

Ejercicios de Python resueltos

Los 68 ejercicios de Python de la web en una sola hoja: cada uno con su enunciado, los datos que necesitas (código de partida, ejemplos de entrada y salida o la base de datos) y, al final, la solución explicada.

Índice

Lenguajes de Marcas

- [1. ¿Están bien anidadas las etiquetas?](#)
- [3. Resolver los espacios de nombres](#)
- [5. Un agregador que mezcla RSS y Atom](#)
- [2. Atributos y referencias de entidad](#)
- [4. Genera un canal RSS 2.0 correcto](#)
- [6. Importación incremental con identificadores externos](#)

Bases de Datos

- [7. Fichero secuencial frente a índice](#)
- [9. Anonimizar datos personales: k-anonimato](#)
- [11. Del diagrama E/R a las tablas](#)
- [13. ¿Qué columnas pueden ser clave?](#)
- [15. Un intérprete de álgebra relacional](#)
- [8. Fragmentación de una base de datos distribuida](#)
- [10. Comprobar cardinalidades con datos reales](#)
- [12. Claves candidatas y forma normal](#)
- [14. Integridad referencial al borrar e insertar](#)

Sistemas Informáticos

- [16. ¿Por qué mi disco de 1 TB tiene 931 GB?](#)
- [18. Ancho de banda y cuellos de botella](#)
- [20. Round Robin con quantum](#)
- [22. Un plan de particiones alineado](#)
- [24. Leer la tabla de particiones de un MBR](#)
- [26. Simulador de estrategias de copia](#)
- [28. ¿Llega este cable?](#)
- [30. Puntos únicos de fallo de una red](#)
- [32. Abreviar y expandir direcciones IPv6](#)
- [34. Permisos efectivos de una carpeta compartida](#)
- [36. ¿Cuánto aguantaría esta contraseña?](#)
- [38. Un índice automático para la documentación](#)
- [17. Dimensionar la fuente de alimentación](#)
- [19. Planificador FCFS y SJF](#)
- [21. Reemplazo de páginas: FIFO, LRU y óptimo](#)
- [23. ¿Arrancará este esquema de particiones?](#)
- [25. Capacidad y tolerancia de un RAID](#)
- [27. ¿Es buena esta copia de seguridad?](#)
- [29. La tabla MAC de un switch](#)
- [31. ¿Pueden hablar sin router?](#)
- [33. Calculadora VLSM](#)
- [35. Un cortafuegos de primera coincidencia](#)
- [37. Informe del inventario](#)
- [39. Un conversor de Markdown a HTML](#)

Sistemas de Gestión Empresarial

- [40. El embudo de ventas del CRM](#)
- [42. Dimensionar servidores para Odoo](#)
- [44. Limpiar los clientes antes de migrarlos](#)
- [46. Encuentra el fallo: el descuento para casi todos](#)
- [48. Encuentra el fallo: el stock que aparece en todos los almacenes](#)
- [50. Encuentra el fallo: el recuento de ventas que se rompe](#)
- [52. Encuentra el fallo: las líneas que se comparten](#)
- [54. Encuentra el fallo: el pedido confirmado que no se factura](#)
- [56. Secuencias de referencias](#)
- [58. Totales de una factura](#)
- [60. Validar NIF y NIE de clientes](#)
- [41. Coste total de propiedad a varios años](#)
- [43. ¿Puede este usuario hacerlo?](#)
- [45. Rotación de copias abuelo-padre-hijo](#)
- [47. Encuentra el fallo: el ranking que no existe](#)
- [49. Un evaluador de dominios de Odoo](#)
- [51. Un campo calculado: la amortización](#)
- [53. Encuentra el fallo: los agotados que se escapan](#)
- [55. Restricciones de un modelo](#)
- [57. Encuentra el fallo: los céntimos que se pierden](#)
- [59. Stock tras los movimientos](#)
- [61. Informe de antigüedad de deuda](#)

Ejercicios largos

62. Corrector de exámenes tipo test

64. Planificador de procesos: FCFS, SJF, SRTF y Round Robin

66. Inventario de almacén con valoración FIFO y reposición

68. Detección de ataques de fuerza bruta en el registro de SSH

63. Informe de ventas a partir de un CSV

65. Plan de direccionamiento con subredes VLSM

67. Resumen de pedidos en XML con ElementTree

Lenguajes de Marcas

1. ¿Están bien anidadas las etiquetas?

Medio · Fundamentos de Lenguajes de Marcas · apuntesdam.com/subject/lenguajes-marcas/topic/fundamentos

Un documento XML bien formado tiene un único elemento raíz y todas sus etiquetas cerradas en orden inverso al de apertura. Lee el documento completo y recorre sus etiquetas (ya están extraídas con una expresión regular; se ignoran `<?...?>`, comentarios y DOCTYPE). Las de cierre son `</nombre>`, las vacías terminan en `/>` y el resto son de apertura (el nombre es lo que va tras `<` hasta un espacio, `/` o `>`). Detén la comprobación en el primer error: «Error en la línea L: `</x>` no tiene etiqueta de apertura», «Error en la línea L: se esperaba `</a>` pero aparece `</b>`», «Error en la línea L: `<x>` es un segundo elemento raíz» (si empieza un elemento cuando la raíz ya se cerró), y al final «Error: falta cerrar `<x>`» (la más interna) o «Error: el documento no tiene elemento raíz». Si todo es correcto: «Bien formado: N elementos, profundidad máxima D, raíz `<r>`» (la raíz tiene profundidad 1). Los nombres distinguen mayúsculas.

Código de partida (python)

```
import re, sys

texto = sys.stdin.read()

def linea(pos):
    return texto.count("\n", 0, pos) + 1

pila = []          # etiquetas abiertas
elementos = 0
profundidad = 0
raiz = None        # nombre de la raíz cuando ya se ha cerrado
for m in re.finditer(r"<[^\>]*>", texto):
    etiqueta = m.group()
    if etiqueta.startswith(("<?", "<!")):
        continue   # declaración, comentarios y DOCTYPE no cuentan
    # TODO: etiquetas de cierre, vacías y de apertura
print("Bien formado")
```

Ejemplo: Catálogo correcto

ENTRADA

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Catálogo de la biblioteca -->
<biblioteca>
  <libro isbn="978-84-1">
    <titulo>El Quijote</titulo>
    <autor>Cervantes</autor>
    <portada archivo="q.jpg"/>
  </libro>
  <libro isbn="978-84-2">
    <titulo>La Regenta</titulo>
  </libro>
</biblioteca>
```

SALIDA ESPERADA

Bien formado: 7 elementos, profundidad máxima 3, raíz `<biblioteca>`

Ejemplo: Cierre cruzado

ENTRADA

```
<pedido>
  <cliente><nombre>Ana</cliente></nombre>
</pedido>
```

SALIDA ESPERADA

Error en la línea 2: se esperaba `</nombre>` pero aparece `</cliente>`

Ejemplo: Dos raíces

ENTRADA

```
<a>
</a>
<b/>
```

SALIDA ESPERADA

Error en la línea 3: es un segundo elemento raíz

Ejemplo: Sin cerrar

ENTRADA

```
<html>
  <body>
    <p>Hola
  </body>
</html>
```

SALIDA ESPERADA

Error en la línea 4: se esperaba </p> pero aparece </body>

2. Atributos y referencias de entidad

Difícil · Fundamentos de Lenguajes de Marcas · apuntesdam.com/subject/lenguajes-marcas/topic/fundamentos

Además de anidar bien, un XML bien formado exige que los atributos tengan valor entre comillas (simples o dobles), sin repetirse en la misma etiqueta y sin «<» en el valor, y que cada «&» empiece una referencia válida: & < > " ';, &#número; o &#xhexadecimal;. Completa las dos funciones. `revisar_etiqueta` recorre los atributos de cada etiqueta de apertura o vacía: el nombre debe empezar por letra o `_` y seguir con letras, dígitos, `_` . : o - (si no, «Línea L, <tag>: nombre de atributo no válido «x»», donde x llega hasta un espacio o =, y se deja de revisar esa etiqueta); sin = → «... el atributo «a» no tiene valor» (y se sigue con el siguiente); sin comillas → «... el valor de «a» no está entre comillas» (se deja la etiqueta); repetido → «... el atributo «a» está repetido»; con < → «... el valor de «a» contiene «<»». Cada atributo con valor cuenta en atributos y su valor pasa por `revisar_referencias`. `revisar_referencias`, para cada &, toma el trozo desde & hasta el primer ; (incluido) sin pasar de un espacio, &, < o ; ; si es válido suma una referencia y si no, «Línea L: referencia no válida «trozo»», con la línea real dentro del texto.

Código de partida (python)

```
import re, sys

texto = sys.stdin.read()
VALIDA = re.compile(r"&(amp|lt|gt|quot|apos|#[0-9]+|#[0-9a-fA-F]+);")

def linea(pos):
    return texto.count("\n", 0, pos) + 1

errores = []
atributos = referencias = 0

def revisar_referencias(cadena, l):
    """Comprueba cada & de la cadena; l es la línea donde se informa."""
    global referencias
    # TODO: referencias de entidad y de carácter

def revisar_etiqueta(etiqueta, l):
    global atributos
    nombre = re.match(r"<\s*([^\s/>]+)", etiqueta).group(1)
    resto = etiqueta[etiqueta.index(nombre) + len(nombre):].rstrip(">").rstrip("/")
    # TODO: atributos de la etiqueta

fin = 0
for m in re.finditer(r"<[>]*>", texto):
    revisar_referencias(texto[fin:m.start()], linea(fin))
    fin = m.end()
    etiqueta = m.group()
    if not etiqueta.startswith(("</", "<?", "<!")):
        revisar_etiqueta(etiqueta, linea(m.start()))
    revisar_referencias(texto[fin:], linea(fin))

if errores:
    print("\n".join(errores))
    print(f"Errores: {len(errores)}")
else:
    print(f"Sin errores: {atributos} atributos y {referencias} referencias correctas")
```

Ejemplo: Tienda con errores

ENTRADA

```
<?xml version="1.0"?>
<tienda nombre="Pan & Café"
web='https://pan.es/?a=1&b=2'>
  <producto id="p1" precio="2,50 &#8364;"
id="p2">Barra & media</producto>
  <producto id=p3 oferta>Croissant &lt;nuevo&gt;
</producto>
  <producto nota="a < b" lid="x">Tarta &copy;
&#x1F370;</producto>
  <producto disponible = "sí" stock="12"/>
</tienda>
```

SALIDA ESPERADA

Línea 3, <producto>: el atributo «id» está repetido
Línea 3: referencia no válida «&»
Línea 4, <producto>: el valor de «id» no está entre comillas
Línea 5, <producto>: el valor de «nota» contiene «<»
Línea 5, <producto>: nombre de atributo no válido «lid»
Línea 5: referencia no válida «©»
Errores: 6

Ejemplo: Documento correcto

ENTRADA

```
<a x="1" y='&quot;dos&quot;';>&lt;3 &#65;&#x42;
</a>
```

SALIDA ESPERADA

Sin errores: 2 atributos y 5 referencias correctas

3. Resolver los espacios de nombres

Muy difícil · Fundamentos de Lenguajes de Marcas · apuntesdam.com/subject/lenguajes-marcas/topic/fundamentos

Un procesador XML no trabaja con prefijos sino con nombres expandidos {URI}local. Lee un documento bien formado y muestra el árbol con el nombre expandido de cada elemento, sangrado dos espacios por nivel; debajo, sangrados dos espacios más, sus atributos «@nombre = valor» (excepto las declaraciones xmlns). Reglas: xmlns="uri" declara el espacio por defecto (xmlns="" lo anula) y xmlns:p="uri" un prefijo; las declaraciones valen para el elemento que las tiene y sus descendientes. Un elemento sin prefijo está en el espacio por defecto; un atributo sin prefijo no está en ningún espacio. El prefijo xml está siempre declarado (<http://www.w3.org/XML/1998/namespace>). Sin espacio de nombres se muestra solo el nombre local. Si un prefijo no está declarado: «Error en la línea L: el prefijo «p» no está declarado» y se termina. Al final, «Espacios de nombres usados: N» y, por orden de primera aparición, « URI: K elemento(s)» contando solo elementos.

Código de partida (python)

```
import re, sys

texto = sys.stdin.read()
XML_NS = "http://www.w3.org/XML/1998/namespace"
ATRIBUTO = re.compile(r'""([\w.-]+\s*=\s*("[^"]*"|'([^']*')')')""')

def linea(pos):
    return texto.count("\n", 0, pos) + 1

ambitos = [{"xml": XML_NS, "": ""}] # pila de ámbitos: prefijo → URI (" es el espacio por defecto)
usos = {} # URI → número de elementos (en orden de primera aparición)
for m in re.finditer(r"<[^>]*>", texto):
    etiqueta = m.group()
    if etiqueta.startswith(("<?", "<!")):
        continue
    if etiqueta.startswith("</"):
        ambitos.pop()
        continue
    nombre = re.match(r"<\s*([^\s/>]+)", etiqueta).group(1)
    atributos = [(a.group(1), a.group(2) if a.group(2) is not None else a.group(3)) for a in
        ATRIBUTO.finditer(etiqueta)]
    actual = dict(ambitos[-1]) # ámbito de este elemento: el del padre más sus declaraciones
    # TODO: declarar, resolver y mostrar el nombre expandido
    if not etiqueta.endswith(">"):
        ambitos.append(actual)
print(f"Espacios de nombres usados: {len(usos)}")
for uri, n in usos.items():
    print(f" {uri}: {n} elemento{'s' if n != 1 else ''}")
```

Ejemplo: Factura con tres vocabularios

ENTRADA

```
<?xml version="1.0"?>
<factura xmlns="urn:facturas"
xmlns:cli="urn:clientes" numero="F-12">
  <cli:cliente cli:nif="12345678Z">
    <cli:nombre xml:lang="es">Ana</cli:nombre>
  </cli:cliente>
  <lineas>
    <linea xmlns="urn:productos"
ref="P1">Teclado</linea>
    <linea xmlns="" ref="P2">Ratón</linea>
  </lineas>
  <total>45.90</total>
</factura>
```

SALIDA ESPERADA

```
{urn:facturas}factura
  @numero = F-12
  {urn:clientes}cliente
    @{urn:clientes}nif = 12345678Z
    {urn:clientes}nombre
      @{http://www.w3.org/XML/1998/namespace}lang
= es
  {urn:facturas}lineas
    {urn:productos}linea
      @ref = P1
    linea
      @ref = P2
  {urn:facturas}total
Espacios de nombres usados: 3
urn:facturas: 3 elementos
urn:clientes: 2 elementos
urn:productos: 1 elemento
```

Ejemplo: Prefijo sin declarar

ENTRADA

```
<doc xmlns:a="urn:a">
  <a:x/>
  <b:y/>
</doc>
```

SALIDA ESPERADA

```
doc
  {urn:a}x
Error en la línea 3: el prefijo «b» no está
declarado
```

4. Genera un canal RSS 2.0 correcto

Difícil · Sindicación de contenidos: RSS y Atom · apuntesdam.com/subject/lenguajes-marcas/topic/sindicacion-contenidos

Una panadería quiere publicar sus novedades como canal RSS. La primera línea tiene «título|enlace|descripción» del canal y cada una de las siguientes una noticia «fecha ISO 8601|título|enlace|descripción|categoría». Construye el documento con `xml.etree.ElementTree` (que escapa solo los caracteres especiales) y escríbelo: primero la línea `<?xml version="1.0" encoding="UTF-8"?>` y después el `<rss version="2.0">` sangrado con dos espacios (`ET.indent`). El `<channel>` contiene, en este orden, `title`, `link`, `description`, `language` (es-es), `lastBuildDate` (la fecha de la noticia más reciente; se omite si no hay noticias) y los `item` de la más reciente a la más antigua. Cada `item` contiene `title`, `link`, `description`, `category`, `guid` (igual al enlace) y `pubDate`. Las fechas del RSS van en formato RFC 822, como «Mon, 29 Sep 2025 09:00:00 +0200» (`email.utils.format_datetime`).

Código de partida (python)

```
import sys
import xml.etree.ElementTree as ET
from datetime import datetime
from email.utils import format_datetime

lineas = [l for l in sys.stdin.read().splitlines() if l.strip()]
titulo, enlace, descripcion = lineas[0].split("|")
noticias = [] # (fecha, titulo, link, descripcion, categoria)
for l in lineas[1:]:
    fecha, t, link, desc, cat = l.split("|")
    noticias.append((datetime.fromisoformat(fecha), t, link, desc, cat))

# TODO: construye el canal con ElementTree y escríbelo
```

Ejemplo: Panadería

ENTRADA

```
Panadería La Espiga|https://laespiga.es|Novedades
& ofertas del obrador
2025-09-26T18:30:00+02:00|Pan de
centeno|https://laespiga.es/centeno|Vuelve el pan
de centeno 100 %|Productos
2025-09-29T09:00:00+02:00|Horario de
otoño|https://laespiga.es/horario|Abrimos a las
7:30 de lunes a sábado|Avisos
2025-09-28T12:15:00+02:00|Taller: masa
madre|https://laespiga.es/taller?
dia=4&plazas=10|Aprende a hacer <b>masa
madre</b>|Talleres
```

SALIDA ESPERADA

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <title>Panadería La Espiga</title>
    <link>https://laespiga.es</link>
    <description>Novedades & ofertas del
    obrador</description>
    <language>es-es</language>
    <lastBuildDate>Mon, 29 Sep 2025 09:00:00
    +0200</lastBuildDate>
    <item>
      <title>Horario de otoño</title>
      <link>https://laespiga.es/horario</link>
      <description>Abrimos a las 7:30 de lunes a
      sábado</description>
      <category>Avisos</category>
      <guid>https://laespiga.es/horario</guid>
      <pubDate>Mon, 29 Sep 2025 09:00:00
      +0200</pubDate>
    </item>
    <item>
      <title>Taller: masa madre</title>
      <link>https://laespiga.es/taller?
      dia=4&plazas=10</link>
      <description>Aprende a hacer <b>masa
      madre</b></description>
      <category>Talleres</category>
      <guid>https://laespiga.es/taller?
      dia=4&plazas=10</guid>
      <pubDate>Sun, 28 Sep 2025 12:15:00
      +0200</pubDate>
    </item>
    <item>
      <title>Pan de centeno</title>
      <link>https://laespiga.es/centeno</link>
      <description>Vuelve el pan de centeno 100 %
      </description>
      <category>Productos</category>
      <guid>https://laespiga.es/centeno</guid>
      <pubDate>Fri, 26 Sep 2025 18:30:00
      +0200</pubDate>
    </item>
  </channel>
</rss>
```

5. Un agregador que mezcla RSS y Atom

Muy difícil · Sindicación de contenidos: RSS y Atom · apuntesdam.com/subject/lenguajes-marcas/topic/sindicacion-contenidos

Un agregador junta varios canales en una sola lista. La entrada empieza con «maximo N» y después vienen los documentos separados por líneas «=====»: unos son RSS 2.0 (raíz <rss>) y otros Atom 1.0 (raíz <feed>) en el espacio de nombres <http://www.w3.org/2005/Atom>, que ElementTree escribe como {<http://www.w3.org/2005/Atom>}feed). De cada entrada obtén: la fecha (pubDate en RFC 822 con email.utils.parsedate_to_datetime, o updated en ISO 8601 con datetime.fromisoformat), el nombre del canal (su title), el título y un identificador (guid o, si no tiene, link en RSS; id en Atom). Descarta las repetidas por identificador conservando la primera que aparece, ordena de la más reciente a la más antigua (a igual fecha, por título) y muestra «Canales: C · Entradas: E (duplicadas descartadas: D)» y las N primeras como «AAAA-MM-DD HH:MM UTC · canal · título», con la hora pasada a UTC.

Código de partida (python)

```
import sys
import xml.etree.ElementTree as ET
from datetime import datetime, timezone
from email.utils import parsedate_to_datetime

ATOM = "{http://www.w3.org/2005/Atom}"
partes = sys.stdin.read().split("====")
maximo = int(partes[0].split()[1]) # «maximo N»
documentos = [ET.fromstring(p.strip()) for p in partes[1:] if p.strip()]

entradas = [] # (fecha UTC, canal, título, identificador)
# TODO: leer RSS y Atom, quitar duplicados, ordenar y mostrar
```

Ejemplo: Dos canales con un duplicado

ENTRADA

```
maximo 10
====
<rss version="2.0"><channel>
  <title>Apuntes DAM</title>
  <link>https://apuntesdam.com</link>
  <description>Novedades</description>
  <item><title>Nuevo tema de XSLT</title>
  <link>https://apuntesdam.com/xslt</link>
  <guid>tag:apuntesdam,2025:41</guid><pubDate>Mon,
29 Sep 2025 09:00:00 +0200</pubDate></item>
  <item><title>Retos SQL</title>
  <link>https://apuntesdam.com/sql</link>
  <pubDate>Thu, 25 Sep 2025 12:00:00
+0200</pubDate></item>
  <item><title>Ejercicios de Entornos</title>
  <link>https://apuntesdam.com/entornos</link>
  <guid>tag:apuntesdam,2025:40</guid><pubDate>Fri,
26 Sep 2025 18:30:00 +0200</pubDate></item>
</channel></rss>
====
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Blog de Marta</title>
  <id>urn:uuid:blog-marta</id>
  <updated>2025-09-29T10:00:00Z</updated>
  <entry><title>Mi primer canal Atom</title>
  <id>urn:uuid:m-3</id><updated>2025-09-
29T07:30:00Z</updated><link
href="https://marta.dev/atom"/></entry>
  <entry><title>XPath sin miedo</title>
  <id>urn:uuid:m-2</id><updated>2025-09-
27T16:45:00+02:00</updated><link
href="https://marta.dev/xpath"/></entry>
  <entry><title>Nuevo tema de XSLT (enlace)
</title><id>tag:apuntesdam,2025:41</id>
<updated>2025-09-29T09:05:00+02:00</updated><link
href="https://apuntesdam.com/xslt"/></entry>
</feed>
```

SALIDA ESPERADA

```
Canales: 2 · Entradas: 5 (duplicadas descartadas:
1)
2025-09-29 07:30 UTC · Blog de Marta · Mi primer
canal Atom
2025-09-29 07:00 UTC · Apuntes DAM · Nuevo tema
de XSLT
2025-09-27 14:45 UTC · Blog de Marta · XPath sin
miedo
2025-09-26 16:30 UTC · Apuntes DAM · Ejercicios
de Entornos
2025-09-25 10:00 UTC · Apuntes DAM · Retos SQL
```

6. Importación incremental con identificadores externos

Muy difícil · Sistemas de Gestión de Información Empresarial · apuntesdam.com/subject/lenguajes-marcas/topic/sistemas-gestion-informacion

Los ERP como Odoó importan datos de forma incremental gracias a un identificador externo (`external_id`): si ya existe, el registro se actualiza; si no, se crea. La entrada tiene una sección `[erp]` con los clientes actuales «`external_id;nif;nombre;email;activo(si/no)`» y una sección `[importar]` con las filas del fichero «`external_id;nif;nombre;email`». Procesa las filas en orden (la línea `N` empieza en 1): primero se valida el NIF (8 dígitos y letra, o letra, 7 dígitos y carácter alfanumérico), después el email y después que haya nombre; si falla, «ERROR línea `N` (`id`): NIF no válido «`x`»» (o «email no válido «`x`»» o «falta el nombre»). Si el NIF ya pertenece a otro registro:

«DUPLICADO línea N (id): el NIF x ya es de otro_id». Si el id existe y nada cambia (y está activo): «SIN CAMBIOS id»; si cambia algo: «ACTUALIZAR id: campo antes → después, ...» (en el orden nif, nombre, email), añadiendo «(reactivado)» si estaba inactivo (y «sin cambios de datos» si solo se reactiva). Si no existe: «CREAR id (nombre)». Los cambios se aplican en el momento, así que afectan a las filas siguientes. Termina con «Creados: a · Actualizados: b · Sin cambios: c · Errores: d» y, si hay clientes activos que no aparecen en el fichero, «No incluidos en el fichero: id, id» en el orden del ERP.

Código de partida (python)

```
import re, sys

NIF = re.compile(r"[0-9]{8}[A-Z]([A-Z][0-9]{7}[A-Z0-9])")
EMAIL = re.compile(r"^[^s@]+@[^\s@]+\.[^\s@]+")

erp = {} # external_id → {"nif", "nombre", "email", "activo"} (el orden de inserción es el del ERP)
filas = [] # filas del fichero a importar: (external_id, nif, nombre, email)
seccion = None
for linea in sys.stdin.read().splitlines():
    linea = linea.strip()
    if not linea:
        continue
    if linea.startswith("["):
        seccion = linea
        continue
    campos = [c.strip() for c in linea.split(";")]
    if seccion == "[erp]":
        ext, nif, nombre, email, activo = campos
        erp[ext] = {"nif": nif, "nombre": nombre, "email": email, "activo": activo == "si"}
    else:
        filas.append(tuple(campos))

# TODO: procesar la importación y mostrar el informe
```

Ejemplo: Clientes de una gestoría

ENTRADA

```
[erp]
cli_001;12345678Z;Ana García;ana@correo.es;si
cli_002;B1234567X;Talleres Norte
S.L.;admin@tnorte.es;si
cli_003;87654321X;Luis Pérez;luis@correo.es;no
cli_004;A7654321B;Academia
Rioja;info@arioja.es;si
[importar]
cli_001;12345678Z;Ana García;ana@correo.es
cli_002;B1234567X;Talleres Norte
S.L.;facturas@tnorte.es
cli_003;87654321X;Luis Pérez Gil;luis@correo.es
cli_005;C1111111D;Tienda Sur;tienda@sur.es
cli_006;1234567;Pepe Ruiz;pepe@correo.es
cli_007;A7654321B;Otra Academia;otra@academia.es
cli_008;Y7654321G;Marta Díaz;marta.correo.es
cli_005;C1111111D;Tienda Sur S.A.;tienda@sur.es
```

SALIDA ESPERADA

```
SIN CAMBIOS cli_001
ACTUALIZAR cli_002: email admin@tnorte.es →
facturas@tnorte.es
ACTUALIZAR cli_003: nombre Luis Pérez → Luis
Pérez Gil (reactivado)
CREAR cli_005 (Tienda Sur)
ERROR línea 5 (cli_006): NIF no válido «1234567»
DUPLICADO línea 6 (cli_007): el NIF A7654321B ya
es de cli_004
ERROR línea 7 (cli_008): email no válido
«marta.correo.es»
ACTUALIZAR cli_005: nombre Tienda Sur → Tienda
Sur S.A.
Creados: 1 · Actualizados: 3 · Sin cambios: 1 ·
Errores: 3
No incluidos en el fichero: cli_004
```

7. Fichero secuencial frente a índice

Medio · Sistemas de Almacenamiento · apuntesdam.com/subject/bases-datos/topic/almacenamiento

¿Por qué una base de datos con índice encuentra un registro entre millones casi al instante? La entrada tiene registros «id nombre» en el orden en que están en el fichero (desordenados) y búsquedas «buscar id». El índice ya está construido: una lista ordenada de (id, posición). Para cada búsqueda cuenta las lecturas de la búsqueda secuencial en el fichero (se leen registros desde el principio hasta encontrarlo; si no está, todos) y las comparaciones de una búsqueda binaria en el índice ($lo = 0$, $hi = n-1$; mientras $lo \leq hi$: $mid = (lo+hi)//2$, se cuenta una comparación y se mira si es el buscado, si es menor o si es mayor), más una lectura del registro si se encuentra. Muestra «buscar x: encontrado (nombre) · secuencial S lecturas · índice C comparaciones + 1 lectura» (o «no existe» y sin «+ 1 lectura»; en singular «1 lectura» y «1 comparación») y al final «Total secuencial: S lecturas · índice: I accesos · ahorro P %» con $P = 100 - 100 \cdot I/S$ redondeado.

Código de partida (python)

```
import sys

registros = []          # (id, nombre) en el orden del fichero (sin ordenar)
búsquedas = []
for linea in sys.stdin.read().splitlines():
    p = linea.split()
    if not p:
        continue
    if p[0] == "buscar":
        búsquedas.append(int(p[1]))
    else:
        registros.append((int(p[0]), p[1]))

indice = sorted((r[0], pos) for pos, r in enumerate(registros))  # (id, posición en el fichero)
# TODO: para cada búsqueda, contar lecturas secuenciales y comparaciones de la búsqueda binaria en el
# índice
```

Ejemplo: 40 registros

ENTRADA

```
431 Ana0
254 Luis1
504 Eva2
766 Juan3
149 Marta4
174 Pablo5
940 Lucía6
648 Hugo7
196 Sara8
474 Iker9
696 Ana10
159 Luis11
619 Eva12
319 Juan13
138 Marta14
188 Pablo15
544 Lucía16
528 Hugo17
171 Sara18
346 Iker19
192 Ana20
664 Luis21
534 Eva22
160 Juan23
946 Marta24
679 Pablo25
226 Lucía26
328 Hugo27
745 Sara28
742 Iker29
163 Ana30
690 Luis31
699 Eva32
506 Juan33
150 Marta34
326 Pablo35
147 Lucía36
670 Hugo37
979 Sara38
236 Iker39
buscar 431
buscar 679
buscar 236
buscar 5
buscar 474
```

SALIDA ESPERADA

```
buscar 431: encontrado (Ana0) · secuencial 1
lectura · índice 1 comparación + 1 lectura
buscar 679: encontrado (Pablo25) · secuencial 26
lecturas · índice 5 comparaciones + 1 lectura
buscar 236: encontrado (Iker39) · secuencial 40
lecturas · índice 6 comparaciones + 1 lectura
buscar 5: no existe · secuencial 40 lecturas ·
índice 5 comparaciones
buscar 474: encontrado (Iker9) · secuencial 10
lecturas · índice 5 comparaciones + 1 lectura
Total secuencial: 117 lecturas · índice: 26
accesos · ahorro 78 %
```

8. Fragmentación de una base de datos distribuida

Difícil · Sistemas de Almacenamiento · apuntesdam.com/subject/bases-datos/topic/almacenamiento

En una base de datos distribuida, una tabla se reparte en fragmentos guardados en distintas sedes. La entrada define la tabla («tabla nombre col1,col2...»), sus filas («fila v1,v2,...»), fragmentos horizontales («fragmento F sede=S campo=v1,v2»: las filas cuyo campo vale uno de esos valores) y consultas. Muestra «F (sede): n filas» por fragmento («1 fila» en singular, también en las consultas); «Compleitud: sí» o «Compleitud: NO, sin fragmento: id X (valor), ...» (las filas que no están en ningún fragmento se pierden; el valor es el del campo del primer fragmento); «Disjunción: sí» o «Disjunción: NO, id X en F1 y F4; ...» (filas en varios fragmentos). Para cada consulta («campo=valor», «campo>número», «campo<número» o «todos»), decide a qué fragmentos hay que preguntar: si la condición es una igualdad sobre el campo de fragmentación, solo a los que incluyen ese valor; si no, a todos. Muestra «consulta C → sedes S1, S2 · n filas: ids» (sedes sin repetir en el orden de los fragmentos, ids ordenados y con repeticiones si una fila está en dos fragmentos; «ninguna» y sin lista si no hay).

```
import sys

columnas, filas, fragmentos, consultas = [], [], [], []
for linea in sys.stdin.read().splitlines():
    p = linea.split(" ", 1)
    if p[0] == "tabla":
        columnas = p[1].split(" ")[1].split(",")
    elif p[0] == "fila":
        valores = p[1].split(",")
        filas.append(dict(zip(columnas, valores)))
    elif p[0] == "fragmento":
        # fragmento F1 sede=Madrid provincia=Madrid,Toledo
        nombre, sede, cond = p[1].split(" ")
        campo, valores = cond.split("=")
        fragmentos.append((nombre, sede.split("=")[1], campo, valores.split(",")
    elif p[0] == "consulta":
        consultas.append(p[1])

# TODO: filas de cada fragmento, completitud, disjunción y consultas
```

Ejemplo: Clientes por provincia

ENTRADA

```
tabla clientes id,nombre,provincia,saldo
fila 1,Ana,Madrid,1200
fila 2,Luis,Sevilla,300
fila 3,Eva,Cádiz,800
fila 4,Iker,Bizkaia,50
fila 5,Marta,Madrid,90
fila 6,Pablo,Lugo,700
fila 7,Sara,Huelva,610
fila 8,Hugo,Toledo,20
fragmento F1 sede=Madrid provincia=Madrid,Toledo
fragmento F2 sede=Sevilla
provincia=Sevilla,Cádiz,Huelva
fragmento F3 sede=Bilbao provincia=Bizkaia
fragmento F4 sede=Sevilla provincia=Toledo
consulta provincia=Cádiz
consulta saldo>500
consulta provincia=Lugo
consulta todos
```

SALIDA ESPERADA

```
F1 (Madrid): 3 filas
F2 (Sevilla): 3 filas
F3 (Bilbao): 1 fila
F4 (Sevilla): 1 fila
Completitud: NO, sin fragmento: id 6 (Lugo)
Disjunción: NO, id 8 en F1 y F4
consulta provincia=Cádiz → sedes Sevilla · 1
fila: 3
consulta saldo>500 → sedes Madrid, Sevilla,
Bilbao · 3 filas: 1, 3, 7
consulta provincia=Lugo → sedes ninguna · 0 filas
consulta todos → sedes Madrid, Sevilla, Bilbao ·
8 filas: 1, 2, 3, 4, 5, 7, 8, 8
```

9. Anonimizar datos personales: k-anonimato

Muy difícil · Sistemas de Almacenamiento · apuntesdam.com/subject/bases-datos/topic/almacenamiento

El RGPD permite publicar o compartir datos para estudios si están anonimizados, pero quitar el nombre no basta: la edad, el código postal y el sexo juntos identifican a mucha gente. La entrada empieza con «k K» y una tabla «nombre;dni;edad;cp;sexo;diagnostico». Anonimiza: elimina nombre y dni («Identificadores eliminados: nombre, dni»); generaliza la edad a su década («34» → «30-39») y el código postal a sus tres primeras cifras («28013» → «280**»). Los cuasi-identificadores son (edad, cp, sexo): las filas con los mismos valores forman un grupo, y k es el tamaño del grupo más pequeño («k antes de suprimir: n»). Suprime las filas de los grupos con menos de K filas; si no queda ninguna, «No queda ningún dato publicable» y termina. Si quedan, muestra la cabecera «edad;cp;sexo;diagnostico», las filas en su orden original y «Grupos: G · k = k (objetivo K) · filas suprimidas: s · l-diversidad: l», donde l es el menor número de diagnósticos distintos en un grupo; si l es 1, añade «Atención: en algún grupo todos tienen el mismo diagnóstico, que se puede deducir».

Código de partida (python)

```
import sys
from collections import Counter, defaultdict

lineas = [l for l in sys.stdin.read().splitlines() if l.strip()]
k_objetivo = int(lineas[0].split()[1])
cabecera = lineas[1].split(";") # nombre;dni;edad;cp;sexo;diagnostico
filas = [dict(zip(cabecera, l.split(";"))) for l in lineas[2:]]

# TODO: eliminar identificadores, generalizar, calcular k y l, y suprimir los grupos pequeños
```

Ejemplo: Estudio médico

ENTRADA

```
k 3
nombre;dni;edad;cp;sexo;diagnostico
Ana López;12345678Z;34;28013;M;asma
Luis Gil;87654321X;37;28015;H;diabetes
Eva Ruiz;11111111H;31;28019;M;migraña
Marta Sanz;22222222J;38;28011;M;asma
Pablo Díaz;33333333P;52;41004;H;hipertensión
Iker Arana;44444444A;55;41007;H;hipertensión
Hugo Vidal;55555555K;58;41002;H;hipertensión
Sara Mora;66666666Q;23;08001;M;alergia
Lucía Paz;77777777B;36;28030;M;diabetes
```

SALIDA ESPERADA

```
Identificadores eliminados: nombre, dni
k antes de suprimir: 1
edad;cp;sexo;diagnostico
30-39;280*;M;asma
30-39;280*;M;migraña
30-39;280*;M;asma
50-59;410*;H;hipertensión
50-59;410*;H;hipertensión
50-59;410*;H;hipertensión
30-39;280*;M;diabetes
Grupos: 2 · k = 3 (objetivo 3) · filas
suprimidas: 2 · l-diversidad: 1
Atención: en algún grupo todos tienen el mismo
diagnóstico, que se puede deducir
```

10. Comprobar cardinalidades con datos reales

Medio · Modelo Entidad-Relación · apuntesdam.com/subject/bases-datos/topic/modelo-er

En el modelo E/R, la cardinalidad (mínimo, máximo) de una entidad en una relación dice en cuántas ocurrencias de la relación participa cada una de sus instancias. La entrada define las instancias de dos entidades («entidad Nombre i1 i2 ...»), la relación («relacion nombre Entidad1(min,max) Entidad2(min,max)», donde max puede ser n) y las ocurrencias («nombre instancia1 instancia2»). Revisa las ocurrencias en orden: si alguna instancia no existe, «Ocurrencia desconocida: a b»; si el par ya apareció, «Par repetido: a b» (una relación no repite la misma pareja); ninguna de las dos cuenta. Muestra «Ocurrencias válidas de nombre: n» y, para cada instancia de la primera entidad y después de la segunda, «Entidad i participa k veces en nombre (mínimo m)» o «(máximo M)» si se sale de su cardinalidad. Termina con «Restricciones: correctas» o «Restricciones: N incumplimientos» («1 incumplimiento»; contando también desconocidas y repetidas).

Código de partida (python)

```
import sys

instancias = {}          # entidad → lista de ocurrencias
relacion = None          # (nombre, entidad1, (min1, max1), entidad2, (min2, max2))
pares = []
for linea in sys.stdin.read().splitlines():
    p = linea.split()
    if not p:
        continue
    if p[0] == "entidad":
        instancias[p[1]] = p[2:]
    elif p[0] == "relacion":          # relacion cursa Alumno(1,n) Modulo(0,2)
        lados = []
        for t in p[2:4]:
            nombre, card = t[:-1].split("(")
            mn, mx = card.split(",")
            lados.append((nombre, (int(mn), None if mx == "n" else int(mx))))
        relacion = (p[1], lados[0][0], lados[0][1], lados[1][0], lados[1][1])
    else:
        pares.append((p[1], p[2]))          # cursa A1 BD

# TODO: comprobar pares repetidos, instancias desconocidas y la participación mínima y máxima
```

Ejemplo: Alumnos y módulos

ENTRADA

```
entidad Alumno A1 A2 A3 A4
entidad Modulo BD PROG LM
relacion cursa Alumno(1,n) Modulo(0,2)
cursa A1 BD
cursa A2 BD
cursa A3 BD
cursa A1 PROG
cursa A1 BD
cursa A5 LM
cursa A2 LM
```

SALIDA ESPERADA

```
Par repetido: A1 BD
Ocurrencia desconocida: A5 LM
Ocurrencias válidas de cursa: 5
Alumno A4 participa 0 veces en cursa (mínimo 1)
Modulo BD participa 3 veces en cursa (máximo 2)
Restricciones: 4 incumplimientos
```

11. Del diagrama E/R a las tablas

Difícil · Modelo Entidad-Relación · apuntesdam.com/subject/bases-datos/topic/modelo-er

Aplica las reglas de transformación del modelo E/R al relacional. La entrada tiene entidades «entidad Nombre *clave atributo ...» (los atributos con * forman la clave primaria) y relaciones binarias «relacion nombre Entidad1 c1 Entidad2 c2 [atributos]» con c1 y c2 iguales a 1, N o M. Reglas: cada entidad es una tabla con sus atributos; en una 1:N (o N:1) la clave de la entidad del lado 1 pasa como clave ajena a la del lado N, junto con los atributos de la relación; en una 1:1 la clave de la primera pasa a la segunda como clave ajena única; una N:M crea una tabla con el nombre de la relación cuya clave primaria es la unión de las dos claves (ambas son claves ajenas), más sus atributos. Si la relación es reflexiva (la misma entidad en los dos lados), la clave ajena se llama clave_relacion; y si la tabla ya tiene una columna con el nombre de la clave ajena, se llama clave_entidad (en minúsculas). Muestra las tablas de las entidades en orden de definición (con las columnas añadidas por las relaciones en el orden de las relaciones) y después las nuevas, con el formato «Tabla(*clave, columna, fk→Tabla)» («*» delante de las columnas de la clave primaria y «fk→Tabla único» en las 1:1), y al final «Tablas: T (k por relaciones N:M)».

Código de partida (python)

```
import sys

entidades = {}          # nombre → (claves, atributos) en orden
relaciones = []         # (nombre, e1, card1, e2, card2, atributos)
for linea in sys.stdin.read().splitlines():
    p = linea.split()
    if not p:
        continue
    if p[0] == "entidad":          # entidad Cliente *dni nombre email
        claves = [a[1:] for a in p[2:] if a.startswith("*")]
        attrs = [a.lstrip("*") for a in p[2:]]
        entidades[p[1]] = (claves, attrs)
    elif p[0] == "relacion":       # relacion realiza Cliente 1 Pedido N [atributos...]
        relaciones.append((p[1], p[2], p[3], p[4], p[5], p[6:]))

# TODO: transformar a tablas y mostrar cada una como Tabla(*clave, columna, fk→Tabla)
```

Ejemplo: Tienda online

ENTRADA

```
entidad Cliente *dni nombre email
entidad Pedido *num fecha
entidad Producto *codigo nombre precio
entidad Tarjeta *numero caducidad
relacion realiza Cliente 1 Pedido N
relacion contiene Pedido N Producto M cantidad
precio_unidad
relacion tiene Cliente 1 Tarjeta 1
```

SALIDA ESPERADA

```
Cliente(*dni, nombre, email)
Pedido(*num, fecha, dni→Cliente)
Producto(*codigo, nombre, precio)
Tarjeta(*numero, caducidad, dni→Cliente único)
contiene(*num→Pedido, *codigo→Producto, cantidad,
precio_unidad)
Tablas: 5 (1 por relaciones N:M)
```

12. Claves candidatas y forma normal

Muy difícil · Modelo Entidad-Relación · apuntesdam.com/subject/bases-datos/topic/modelo-er

Normalizar empieza por saber qué claves tiene una relación y qué dependencias rompen cada forma normal. La primera línea es el esquema «R(A,B,C...)» y las siguientes, dependencias funcionales «X1,X2 → Y1,Y2». Implementa cierre(x) (todos los atributos que se deducen de x aplicando las dependencias hasta que no cambie nada) y después: «Claves candidatas: {...}, {...}» (los conjuntos mínimos cuyo cierre es todo el esquema, por tamaño y en el orden de combinations sobre el esquema) y «Atributos primos: ...» (los que están en alguna clave, en orden del esquema, o «ninguno»). Para cada dependencia, en orden: si la derecha está contenida en la izquierda, «X → Y: trivial»; si X es superclave, «X → Y: correcta (X es superclave)»; si todo lo que determina (sin contar X) es primo, «incumple FNBC (X no es superclave)»; si X es una parte propia de alguna clave, «incumple 2FN (dependencia parcial de la clave {...})» (la primera clave que la contenga); si no, «incumple 3FN (dependencia transitiva)». Termina con «Forma normal: F», la más alta que se cumple: FNBC si no hay incumplimientos; 3FN si solo se incumple FNBC; 2FN si hay alguna transitiva pero no parciales; 1FN si hay alguna parcial.

Código de partida (python)

```
import sys
from itertools import combinations

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]
atributos = lineas[0][lineas[0].index("(") + 1:-1].split(",") # R(A,B,C,D)
dependencias = []
for l in lineas[1:]:
    izq, der = l.split("->")
    dependencias.append([a.strip() for a in izq.split(",")], [a.strip() for a in der.split(",")])

def ordenar(conjunto):
    return [a for a in atributos if a in conjunto]

def cierre(x):
    # TODO: atributos que se deducen a partir de x con las dependencias
    return set(x)

# TODO: claves candidatas, atributos primos, análisis de cada dependencia y forma normal
```

Ejemplo: Matrícula

ENTRADA

```
R(alumno,modulo,nota,nombre,profesor,despacho)
alumno,modulo -> nota
alumno -> nombre
modulo -> profesor
profesor -> despacho
```

SALIDA ESPERADA

```
Claves candidatas: {alumno,modulo}
Atributos primos: alumno, modulo
alumno,modulo -> nota: correcta (alumno,modulo es superclave)
alumno -> nombre: incumple 2FN (dependencia parcial de la clave {alumno,modulo})
modulo -> profesor: incumple 2FN (dependencia parcial de la clave {alumno,modulo})
profesor -> despacho: incumple 3FN (dependencia transitiva)
Forma normal: 1FN
```

Ejemplo: Transitiva

ENTRADA

```
R(dni,nombre,cp,ciudad)
dni -> nombre,cp
cp -> ciudad
```

SALIDA ESPERADA

```
Claves candidatas: {dni}
Atributos primos: dni
dni -> nombre,cp: correcta (dni es superclave)
cp -> ciudad: incumple 3FN (dependencia transitiva)
Forma normal: 2FN
```

Ejemplo: Casi FNBC

ENTRADA

```
R(A,B,C)
A,B -> C
C -> B
```

SALIDA ESPERADA

```
Claves candidatas: {A,B}, {A,C}
Atributos primos: A, B, C
A,B -> C: correcta (A,B es superclave)
C -> B: incumple FNBC (C no es superclave)
Forma normal: 3FN
```

13. ¿Qué columnas pueden ser clave?

Medio · Modelo Relacional · apuntesdam.com/subject/bases-datos/topic/modelo-relacional

Antes de crear una tabla a partir de datos existentes (una hoja de cálculo, un CSV) hay que decidir la clave primaria. La entrada es un CSV sencillo: la cabecera y las filas, con los valores vacíos como nulos. Muestra para cada columna «col: k valores distintos de n» (sin contar nulos), «Columnas con nulos: ...» (o «ninguna»; una clave primaria no admite nulos) y «Claves candidatas según los datos: {a}, {b,c}»: las combinaciones de columnas sin nulos, de hasta 3

columnas, cuyos valores no se repiten en ninguna fila y que no contienen otra ya encontrada (mínimas), por tamaño y en el orden de las columnas; si no hay, «ninguna con hasta 3 columnas». Termina con el aviso «Aviso: los datos solo descartan claves; que hoy no haya repetidos no garantiza que no los haya mañana».

Código de partida (python)

```
import sys
from itertools import combinations

lineas = [l for l in sys.stdin.read().splitlines() if l.strip()]
columnas = lineas[0].split(",")
filas = [[v if v != "" else None for v in l.split(",")] for l in lineas[1:]]

# TODO: valores distintos por columna, columnas con nulos y combinaciones únicas mínimas (hasta 3
columnas)
```

Ejemplo: Alumnos

ENTRADA

```
dni,nombre,email,telefono,ciudad,curso
12345678Z,Ana
García,ana@correo.es,600111222,Logroño,1DAM
87654321X,Luis Pérez,luis@correo.es,,Haro,1DAM
11111111H,Ana
García,ana.g@correo.es,600333444,Logroño,2DAM
22222222J,Marta
Gil,marta@correo.es,600555666,Arnedo,1DAM
33333333P,Luis
Pérez,lperez@correo.es,600777888,Logroño,2DAM
```

SALIDA ESPERADA

```
dni: 5 valores distintos de 5
nombre: 3 valores distintos de 5
email: 5 valores distintos de 5
telefono: 4 valores distintos de 5
ciudad: 3 valores distintos de 5
curso: 2 valores distintos de 5
Columnas con nulos: telefono
Claves candidatas según los datos: {dni},
{email}, {nombre,curso}
Aviso: los datos solo descartan claves; que hoy
no haya repetidos no garantiza que no los haya
mañana
```

14. Integridad referencial al borrar e insertar

Difícil · Modelo Relacional · apuntesdam.com/subject/bases-datos/topic/modelo-relacional

Una clave ajena impide que queden filas «huérfanas», y su regla ON DELETE decide qué pasa al borrar la fila referenciada: RESTRICT (se impide el borrado), CASCADE (se borran también las filas que la referencian) o SET NULL (su clave ajena pasa a NULL). La entrada define tablas («tabla nombre id col...»; la primera columna es siempre id), claves ajenas («fk tabla.columna -> otra.id ON DELETE REGLA»), filas («fila tabla valores...», NULL para nulo) y órdenes. «borrar tabla id» funciona como una transacción: o se hace con todos sus efectos o no se hace nada; muestra «borrar t id → borradas: t#id, hija#id, ...» (en el orden en que se borran, primero la fila pedida y después, recursivamente, sus dependientes) y, si hay, «; a NULL: tabla#id.columna, ...», o «borrar t id → ERROR: no se puede borrar t id: lo usan n filas de hija» (o «no existe t id»). «insertar tabla valores...» comprueba que el id no exista («ERROR: ya existe t id») y que cada clave ajena no nula apunte a una fila existente («ERROR: no existe otra v»); si no, «correcto». «mostrar» lista cada tabla «nombre: v1 v2 | v1 v2» o «(vacía)».

Código de partida (python)

```
import sys

tablas = {}          # nombre -> (columnas, {id: fila como dict})
claves_ajenas = []   # (tabla, columna, tabla_ref, regla)
ordenes = []
for linea in sys.stdin.read().splitlines():
    p = linea.split()
    if not p:
        continue
    if p[0] == "tabla":
        tablas[p[1]] = (p[2:], {})
    elif p[0] == "fk":          # fk pedido.cliente_id -> cliente.id ON DELETE CASCADE
        t, c = p[1].split(".")
        claves_ajenas.append((t, c, p[3].split(".")[0], " ".join(p[6:])))
    elif p[0] == "fila":
        cols, filas = tablas[p[1]]
        fila = dict(zip(cols, [None if v == "NULL" else v for v in p[2:])))
        filas[fila["id"]] = fila
    else:
        ordenes.append(p)

# TODO: procesar borrar, insertar y mostrar respetando las claves ajenas
```

Ejemplo: Tienda

ENTRADA

```
tabla cliente id nombre
tabla pedido id cliente_id fecha
tabla linea id pedido_id producto
tabla factura id pedido_id importe
fk pedido.cliente_id -> cliente.id ON DELETE
CASCADE
fk linea.pedido_id -> pedido.id ON DELETE CASCADE
fk factura.pedido_id -> pedido.id ON DELETE
RESTRICT
fila cliente 1 Ana
fila cliente 2 Luis
fila cliente 3 Eva
fila pedido 10 1 2025-09-01
fila pedido 11 2 2025-09-03
fila pedido 12 2 2025-09-05
fila linea 100 10 Teclado
fila linea 101 11 Ratón
fila linea 102 12 Monitor
fila linea 103 12 Cable
fila factura 500 10 45.90
borrar cliente 1
borrar cliente 2
insertar pedido 13 9 2025-10-01
insertar pedido 13 3 2025-10-01
insertar pedido 13 3 2025-10-02
borrar linea 999
mostrar
```

SALIDA ESPERADA

```
borrar cliente 1 -> ERROR: no se puede borrar
pedido 10: lo usan 1 fila de factura
borrar cliente 2 -> borradas: cliente#2,
pedido#11, linea#101, pedido#12, linea#102,
linea#103
insertar pedido 13 -> ERROR: no existe cliente 9
insertar pedido 13 -> correcto
insertar pedido 13 -> ERROR: ya existe pedido 13
borrar linea 999 -> ERROR: no existe linea 999
cliente: 1 Ana | 3 Eva
pedido: 10 1 2025-09-01 | 13 3 2025-10-01
linea: 100 10 Teclado
factura: 500 10 45.90
```

15. Un intérprete de álgebra relacional

Muy difícil · Modelo Relacional · apuntesdam.com/subject/bases-datos/topic/modelo-relacional

El álgebra relacional es la base teórica de SQL: cada consulta es una composición de operaciones sobre relaciones que devuelven relaciones. Programa un intérprete. La entrada define tablas («tabla nombre c1,c2,...», sus filas con valores separados por comas y «fin») y después órdenes «R = operación ...» y «mostrar R». Operaciones: «seleccion T col=valor» (también <, > y <; compara como número si los dos lo son); «proyeccion T c1,c2» (solo esas columnas y sin tuplas repetidas, conservando la primera aparición); «join T1 T2 a=b» (combina cada tupla de T1 con las de T2 que cumplan T1.a = T2.b; las columnas de T2 que ya existen en T1 se llaman T2.col); «union», «diferencia» e

«interseccion» de dos relaciones con las mismas columnas en el mismo orden (si no, «esquemas incompatibles (...) y (...)»), sin repetidos; y «renombrar T viejo=nuevo». Si falta una relación o una columna, muestra «ERROR en R: no existe X» o «ERROR en R: no existe la columna c en T» y R no se crea («mostrar» de algo inexistente: «ERROR: no existe X»). «mostrar R» imprime «R(c1, c2)», cada tupla como « v1 | v2» y « n tuplas» («1 tupla»).

Código de partida (python)

```
import sys

relaciones = {}          # nombre → (columnas, lista de tuplas)
lineas = sys.stdin.read().splitlines()
i = 0
while i < len(lineas):
    l = lineas[i].strip()
    if l.startswith("tabla "):
        # tabla nombre col1,col2 ... filas ... fin
        nombre, cols = l.split()[1], l.split()[2].split(",")
        filas = []
        i += 1
        while lineas[i].strip() != "fin":
            filas.append(tuple(lineas[i].strip().split(",")))
            i += 1
        relaciones[nombre] = (cols, filas)
    elif l:
        pass # TODO: asignaciones «R = operación ...» y «mostrar R»
    i += 1
```

Ejemplo: Alumnos y matrículas

ENTRADA

```
tabla alumno id,nombre,curso
1,Ana,1DAM
2,Luis,2DAM
3,Eva,1DAM
4,Iker,1DAW
fin
tabla matricula alumno,module,nota
1,BD,7
1,PROG,4
2,BD,9
3,PROG,6
3,BD,5
fin
R1 = seleccion alumno curso=1DAM
R2 = join R1 matricula id=alumno
R3 = proyeccion R2 nombre,module
mostrar R3
R4 = seleccion R2 nota<5
R5 = proyeccion R4 nombre
mostrar R5
R6 = proyeccion alumno id
R7 = proyeccion matricula alumno
R8 = diferencia R6 R7
R9 = renombrar R7 alumno=id
R10 = diferencia R6 R9
mostrar R10
R11 = proyeccion R2 module
mostrar R11
R12 = seleccion alumno edad>18
R13 = union alumno matricula
mostrar R13
```

SALIDA ESPERADA

```
R3(nombre, module)
Ana | BD
Ana | PROG
Eva | PROG
Eva | BD
4 tuplas
R5(nombre)
Ana
1 tupla
ERROR en R8: esquemas incompatibles (id) y
(alumno)
R10(id)
4
1 tupla
R11(module)
BD
PROG
2 tuplas
ERROR en R12: no existe la columna edad en alumno
ERROR en R13: esquemas incompatibles
(id,nombre,curso) y (alumno,module,nota)
ERROR: no existe R13
```

16. ¿Por qué mi disco de 1 TB tiene 931 GB?

Medio · Componentes Físicos · apuntesdam.com/subject/sistemas/topic/componentes-fisicos

Los fabricantes miden en unidades decimales (1 GB = 10^9 bytes) y muchos sistemas operativos en binarias (1 GiB = 2^{30} bytes, aunque Windows lo muestre como «GB»). Cada línea de la entrada es una cantidad y una unidad (B, KB, MB, GB, TB, KiB, MiB, GiB o TiB). Muestra «cantidad unidad = N bytes = X GB = Y GiB», con los bytes como número entero redondeado y X e Y con 2 decimales. Si la unidad no existe, «línea: unidad desconocida».

Código de partida (python)

```
import sys

DECIMAL = {"B": 1, "KB": 10**3, "MB": 10**6, "GB": 10**9, "TB": 10**12}
BINARIO = {"KiB": 2**10, "MiB": 2**20, "GiB": 2**30, "TiB": 2**40}

# TODO: convierte cada tamaño a bytes, a GB y a GiB
```

Ejemplo: Discos y memoria

ENTRADA

```
1 TB
512 GB
16 GiB
4.7 GB
```

SALIDA ESPERADA

```
1 TB = 1000000000000 bytes = 1000.00 GB = 931.32 GiB
512 GB = 512000000000 bytes = 512.00 GB = 476.84 GiB
16 GiB = 17179869184 bytes = 17.18 GB = 16.00 GiB
4.7 GB = 4700000000 bytes = 4.70 GB = 4.38 GiB
```

Ejemplo: Unidades pequeñas

ENTRADA

```
1536 KiB
700 MB
1 B
```

SALIDA ESPERADA

```
1536 KiB = 1572864 bytes = 0.00 GB = 0.00 GiB
700 MB = 700000000 bytes = 0.70 GB = 0.65 GiB
1 B = 1 bytes = 0.00 GB = 0.00 GiB
```

17. Dimensionar la fuente de alimentación

Difícil · Componentes Físicos · apuntesdam.com/subject/sistemas/topic/componentes-fisicos

Las primeras líneas son los componentes con su consumo máximo, «nombre vatios» (el nombre puede tener espacios). Después vienen cuatro líneas de configuración «clave=valor»: certificación (bronze, gold o platinum, con eficiencias 0.85, 0.90 y 0.92), uso (porcentaje medio de carga respecto al máximo), horas (de uso al día) y precio (€ por kWh). Muestra: «Consumo de los componentes: N W»; «Con un margen del 30 %: N W» (sin decimales); «Fuente recomendada: N W» (la primera del catálogo que llegue a ese valor, o «ninguna del catálogo es suficiente»); «Consumo real de la pared: X W» (consumo × uso / 100 / eficiencia, 1 decimal); y «Coste anual: X € (Y kWh)» con 365 días al año (2 y 1 decimales).

Código de partida (python)

```
import sys

CATALOGO = [450, 550, 650, 750, 850, 1000, 1200]
EFICIENCIA = {"bronze": 0.85, "gold": 0.90, "platinum": 0.92}

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

# TODO: calcula el consumo, elige la fuente y estima el coste anual
```

Ejemplo: Equipo de desarrollo

ENTRADA

Procesador Ryzen 7 105
Gráfica RTX 4060 115
Placa base 50
RAM 2 módulos 10
SSD NVMe 8
Ventiladores 12
certificacion=gold
uso=40
horas=8
precio=0.18

SALIDA ESPERADA

Consumo de los componentes: 300 W
Con un margen del 30 %: 390 W
Fuente recomendada: 450 W
Consumo real de la pared: 133.3 W
Coste anual: 70.08 € (389.3 kWh)

Ejemplo: Estación de trabajo

ENTRADA

CPU 253
GPU RTX 4090 450
Resto 120
certificacion=platinum
uso=70
horas=10
precio=0.2

SALIDA ESPERADA

Consumo de los componentes: 823 W
Con un margen del 30 %: 1070 W
Fuente recomendada: 1200 W
Consumo real de la pared: 626.2 W
Coste anual: 457.12 € (2285.6 kWh)

18. Ancho de banda y cuellos de botella

Muy difícil · Componentes Físicos · apuntesdam.com/subject/sistemas/topic/componentes-fisicos

Cada línea describe un componente. «ram DDR5-6000 2» es memoria del tipo indicado (el número son millones de transferencias por segundo, MT/s) en 2 canales: cada canal transfiere 8 bytes, así que el ancho de banda en GB/s es $\text{MT/s} \times 8 \times \text{canales} / 1000$. «ssd 7000 pcie4 x4» es un SSD que puede leer a 7000 MB/s conectado a una ranura PCIe de 4.ª generación con 4 líneas; cada línea transporta 0.985, 1.969 o 3.938 GB/s en las generaciones 3, 4 y 5. Para la memoria muestra «Memoria DDR5-6000 en 2 canales: 96.0 GB/s» («canal» en singular si es 1). Para el SSD, «SSD de V MB/s en PCIe G.0 xN (E GB/s): sin cuello de botella» o, si el enlace es más lento que el SSD, «...: limitado a E MB/s (D MB/s desaprovechados)», con el enlace en MB/s sin decimales ($\times 1000$) y en GB/s con 1 decimal.

Código de partida (python)

```
import sys

PCIE_GBS_POR_LINEA = {"3": 0.985, "4": 1.969, "5": 3.938}

# TODO: calcula el ancho de banda y detecta cuellos de botella
```

Ejemplo: Equipo actual

ENTRADA

ram DDR5-6000 2
ssd 7000 pcie4 x4
ssd 3500 pcie3 x4

SALIDA ESPERADA

Memoria DDR5-6000 en 2 canales: 96.0 GB/s
SSD de 7000 MB/s en PCIe 4.0 x4 (7.9 GB/s): sin
cuello de botella
SSD de 3500 MB/s en PCIe 3.0 x4 (3.9 GB/s): sin
cuello de botella

Ejemplo: SSD en ranura lenta

ENTRADA

```
ssd 7400 pcie3 x4
ssd 12000 pcie5 x4
ram DDR4-3200 1
```

SALIDA ESPERADA

```
SSD de 7400 MB/s en PCIe 3.0 x4 (3.9 GB/s):
limitado a 3940 MB/s (3460 MB/s desaprovechados)
SSD de 12000 MB/s en PCIe 5.0 x4 (15.8 GB/s): sin
cuello de botella
Memoria DDR4-3200 en 1 canal: 25.6 GB/s
```

19. Planificador FCFS y SJF

Medio · Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/sistemas-operativos

La primera línea es el algoritmo: FCFS (el primero que llega, se ejecuta primero) o SJF (de los que ya han llegado, el de ráfaga más corta; a igualdad, el que llegó antes y después por nombre). Ambos son no expropiativos: un proceso se ejecuta hasta terminar. Cada línea siguiente es un proceso «nombre llegada ráfaga». Muestra el diagrama de Gantt en una línea, con los tramos «nombre inicio-fin» separados por « | » (si la CPU queda libre esperando a que llegue alguien, un tramo «ocioso inicio-fin»); después, para cada proceso en el orden de la entrada, «nombre: espera E, retorno R» (espera = inicio – llegada; retorno = fin – llegada); y por último la espera media y el retorno medio con 2 decimales.

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
algoritmo = lineas[0][0]
procesos = [(nombre, int(llegada), int(rafaga)) for nombre, llegada, rafaga in lineas[1:]]

# TODO: planifica con FCFS o SJF y muestra el diagrama y los tiempos medios
```

Ejemplo: FCFS del ejemplo

ENTRADA

```
FCFS
P1 0 5
P2 1 3
P3 2 1
P4 3 2
```

SALIDA ESPERADA

```
P1 0-5 | P2 5-8 | P3 8-9 | P4 9-11
P1: espera 0, retorno 5
P2: espera 4, retorno 7
P3: espera 6, retorno 7
P4: espera 6, retorno 8
Espera media: 4.00
Retorno medio: 6.75
```

Ejemplo: SJF del ejemplo

ENTRADA

```
SJF
P1 0 5
P2 1 3
P3 2 1
P4 3 2
```

SALIDA ESPERADA

```
P1 0-5 | P3 5-6 | P4 6-8 | P2 8-11
P1: espera 0, retorno 5
P2: espera 7, retorno 10
P3: espera 3, retorno 4
P4: espera 3, retorno 5
Espera media: 3.25
Retorno medio: 6.00
```

Ejemplo: CPU ociosa

ENTRADA

```
FCFS
A 2 3
B 10 2
C 11 1
```

SALIDA ESPERADA

```
ocioso 0-2 | A 2-5 | ocioso 5-10 | B 10-12 | C
12-13
A: espera 0, retorno 3
B: espera 0, retorno 2
C: espera 1, retorno 2
Espera media: 0.33
Retorno medio: 2.33
```

20. Round Robin con quantum

Difícil · Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/sistemas-operativos

La primera línea es «quantum Q»; cada línea siguiente, un proceso «nombre llegada ráfaga». Simula Round Robin: el proceso al frente de la cola se ejecuta como mucho Q unidades; si no termina, vuelve al final de la cola. Los procesos que llegan durante ese tramo (o justo cuando termina) entran en la cola antes que el proceso expulsado. Si la cola está vacía, la CPU queda ociosa hasta la siguiente llegada. Muestra el diagrama de Gantt («nombre inicio-fin» separados por « | », con tramos «ocioso inicio-fin» si hace falta), para cada proceso en el orden de la entrada «nombre: espera E, retorno R» (retorno = fin – llegada; espera = retorno – ráfaga), la espera y el retorno medios con 2 decimales y «Cambios de contexto: N» (veces que en el diagrama un tramo es de un proceso distinto al anterior).

Código de partida (python)

```
import sys
from collections import deque

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
quantum = int(lineas[0][1])
procesos = [(nombre, int(llegada), int(rafaga)) for nombre, llegada, rafaga in lineas[1:]]

# TODO: simula Round Robin y muestra el diagrama y los tiempos
```

Ejemplo: Ejemplo con quantum 2

ENTRADA

```
quantum 2
P1 0 5
P2 1 3
P3 2 1
P4 3 2
```

SALIDA ESPERADA

```
P1 0-2 | P2 2-4 | P3 4-5 | P1 5-7 | P4 7-9 | P2
9-10 | P1 10-11
P1: espera 6, retorno 11
P2: espera 6, retorno 9
P3: espera 2, retorno 3
P4: espera 4, retorno 6
Espera media: 4.50
Retorno medio: 7.25
Cambios de contexto: 6
```

Ejemplo: Quantum grande

ENTRADA

```
quantum 10
P1 0 5
P2 1 3
P3 2 1
P4 3 2
```

SALIDA ESPERADA

```
P1 0-5 | P2 5-8 | P3 8-9 | P4 9-11
P1: espera 0, retorno 5
P2: espera 4, retorno 7
P3: espera 6, retorno 7
P4: espera 6, retorno 8
Espera media: 4.00
Retorno medio: 6.75
Cambios de contexto: 3
```

Ejemplo: CPU ociosa

ENTRADA

```
quantum 3
A 0 4
B 10 5
C 12 2
```

SALIDA ESPERADA

```
A 0-3 | A 3-4 | ocioso 4-10 | B 10-13 | C 13-15 |
B 15-17
A: espera 0, retorno 4
B: espera 2, retorno 7
C: espera 1, retorno 3
Espera media: 1.00
Retorno medio: 4.67
Cambios de contexto: 4
```

21. Reemplazo de páginas: FIFO, LRU y óptimo

Muy difícil · Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/sistemas-operativos

Cuando la memoria está llena y se necesita una página que no está cargada (fallo de página), el sistema operativo elige una víctima. La primera línea es «marcos N» (páginas que caben en memoria, inicialmente vacía) y la segunda, la cadena de referencias separadas por espacios. Cuenta los fallos con tres algoritmos: FIFO (sale la que lleva más tiempo cargada), LRU (sale la que hace más tiempo que no se usa) y óptimo (sale la que tardará más en volver a usarse; si varias no vuelven a usarse, la que lleva más tiempo cargada). Muestra «FIFO: N fallos (P %)», «LRU: ...» y «Óptimo: ...», con el porcentaje sobre el total de referencias con 1 decimal. Prueba la cadena 1 2 3 4 1 2 5 1 2 3 4 5 con 3 y con 4 marcos: verás la anomalía de Bélády.

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
marcos = int(lineas[0][1])
referencias = [int(x) for x in lineas[1]]

# TODO: cuenta los fallos de página con FIFO, LRU y el algoritmo óptimo
```

Ejemplo: Cadena clásica con 3 marcos

ENTRADA

```
marcos 3
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
```

SALIDA ESPERADA

```
FIFO: 15 fallos (75.0 %)
LRU: 12 fallos (60.0 %)
Óptimo: 9 fallos (45.0 %)
```

Ejemplo: Anomalía de Bélády con 3

ENTRADA

```
marcos 3
1 2 3 4 1 2 5 1 2 3 4 5
```

SALIDA ESPERADA

```
FIFO: 9 fallos (75.0 %)
LRU: 10 fallos (83.3 %)
Óptimo: 7 fallos (58.3 %)
```

Ejemplo: Anomalía de Bélády con 4

ENTRADA

```
marcos 4
1 2 3 4 1 2 5 1 2 3 4 5
```

SALIDA ESPERADA

```
FIFO: 10 fallos (83.3 %)
LRU: 8 fallos (66.7 %)
Óptimo: 6 fallos (50.0 %)
```

22. Un plan de particiones alineado

Medio · Instalación de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-operativos

La primera línea es «disco N GiB» (N entero). Cada línea siguiente es una partición «nombre tamaño», con el tamaño en MiB («512MiB»), en GiB («50GiB», puede tener decimales) o «resto» (todo lo que quede). En un disco GPT, el primer MiB queda reservado (tabla y alineación) y también el último (copia de la tabla). Coloca las particiones una detrás de otra empezando en el MiB 1 y muestra «nombre: inicio A MiB, fin B MiB, tamaño T GiB» (T con 2 decimales). Si una partición no cabe, muestra «ERROR: no cabe nombre (faltan N MiB)» y para. Si todas caben, termina con «Sin asignar: X GiB».

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
disco_mib = int(lineas[0][1].replace("GiB", "")) * 1024

# TODO: coloca las particiones alineadas a 1 MiB y muestra dónde empieza y acaba cada una
```

Ejemplo: Ubuntu en 500 GiB

ENTRADA

```
disco 500GiB
efi 1GiB
raiz 50GiB
swap 8GiB
home resto
```

SALIDA ESPERADA

```
efi: inicio 1 MiB, fin 1025 MiB, tamaño 1.00 GiB
raiz: inicio 1025 MiB, fin 52225 MiB, tamaño
50.00 GiB
swap: inicio 52225 MiB, fin 60417 MiB, tamaño
8.00 GiB
home: inicio 60417 MiB, fin 511999 MiB, tamaño
441.00 GiB
Sin asignar: 0.00 GiB
```

Ejemplo: Windows con hueco libre

ENTRADA

```
disco 256GiB
esp 260MiB
msr 16MiB
windows 180GiB
recuperacion 1GiB
```

SALIDA ESPERADA

```
esp: inicio 1 MiB, fin 261 MiB, tamaño 0.25 GiB
msr: inicio 261 MiB, fin 277 MiB, tamaño 0.02 GiB
windows: inicio 277 MiB, fin 184597 MiB, tamaño
180.00 GiB
recuperacion: inicio 184597 MiB, fin 185621 MiB,
tamaño 1.00 GiB
Sin asignar: 74.73 GiB
```

23. ¿Arrancará este esquema de particiones?

Difícil · Instalación de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-operativos

Las tres primeras líneas son «firmware uefi|bios», «tabla gpt|mbr» y «disco N» (GiB). Cada línea siguiente es una partición «nombre clase sistema tamaño punto» (clase: primaria, extendida o logica; sistema: fat32, ntfs, ext4, swap...; tamaño en GiB; punto de montaje o «-»). Comprueba estas reglas en este orden y muestra un mensaje por cada una que falle, o «Esquema válido» si ninguna falla: (1) «UEFI necesita una tabla GPT (o activar el modo CSM)»; con MBR: (2) «MBR solo puede usar 2048 GiB de los N GiB del disco» si el disco es mayor, (3) «MBR admite como máximo 4 particiones primarias o extendidas», (4) «Solo puede haber una partición extendida», (5) «Las particiones lógicas necesitan una partición extendida»; con GPT: (6) «GPT no tiene particiones extendidas ni lógicas»; (7) con UEFI, «Falta la partición EFI (FAT32 montada en /boot/efi)»; (8) «Falta la partición raíz (/)»; (9) «Las particiones ocupan T GiB y el disco tiene N GiB» (la extendida no cuenta, porque contiene a las lógicas). Los números se muestran sin decimales innecesarios.

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
config = {l[0]: l[1] for l in lineas[:3]}          # firmware, tabla, disco
particiones = lineas[3:]                          # nombre clase sistema tamaño_GiB punto_de_montaje

# TODO: comprueba las reglas en orden y muestra los problemas o «Esquema válido»
```

Ejemplo: Esquema correcto

ENTRADA

```
firmware uefi
tabla gpt
disco 500
efi primaria fat32 1 /boot/efi
raiz primaria ext4 50 /
home primaria ext4 400 /home
```

SALIDA ESPERADA

Esquema válido

Ejemplo: UEFI con MBR y sin EFI

ENTRADA

```
firmware uefi
tabla mbr
disco 3000
sistema primaria ntfs 200 C:
datos primaria ntfs 2500 D:
```

SALIDA ESPERADA

UEFI necesita una tabla GPT (o activar el modo CSM)
MBR solo puede usar 2048 GiB de los 3000 GiB del disco
Falta la partición EFI (FAT32 montada en /boot/efi)
Falta la partición raíz (/)

Ejemplo: MBR clásico con lógicas

ENTRADA

```
firmware bios
tabla mbr
disco 1000
win primaria ntfs 300 C:
recup primaria ntfs 1 -
ext extendida - 699 -
raiz logica ext4 100 /
swap logica swap 8 -
home logica ext4 591 /home
```

SALIDA ESPERADA

Esquema válido

24. Leer la tabla de particiones de un MBR

Muy difícil · Instalación de Sistemas Operativos · apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-operativos

El sector 0 de un disco MBR termina con la tabla de particiones: 4 entradas de 16 bytes (desde el byte 446) y la firma 55 AA. La entrada recibe esos 66 bytes en hexadecimal. En cada entrada: el byte 0 es 0x80 si la partición es la activa (arrancable); el byte 4, el tipo; los bytes 8-11, el sector inicial (LBA); y los bytes 12-15, el número de sectores, ambos enteros de 4 bytes en little endian (el byte menos significativo primero). Las entradas con tipo 00 están vacías. Para cada partición muestra «Partición N: [activa,]tipo XX (nombre), sector inicial S, T sectores (G GiB)» con sectores de 512 bytes y G con 2 decimales (tipo en hexadecimal con 2 cifras en mayúsculas; nombre «desconocido» si no está en la tabla). Si el tipo es EE, añade la línea «El disco usa GPT: esta entrada solo protege el disco de herramientas antiguas». Al final, «Firma: correcta (55 AA)» o «Firma: incorrecta (XX XX)».

Código de partida (python)

```
import sys

datos = bytes.fromhex(''.join(sys.stdin.read().split())) # las 4 entradas (64 bytes) y la firma (2 bytes)
TIPOS = {0x07: "NTFS/exFAT", 0x0B: "FAT32", 0x0C: "FAT32 LBA", 0x83: "Linux", 0x82: "Linux swap",
         0x05: "Extendida", 0x0F: "Extendida LBA", 0xEE: "GPT de protección", 0xEF: "Sistema EFI"}

# TODO: interpreta cada entrada de 16 bytes y la firma del sector
```

Ejemplo: Windows y Linux en arranque dual

ENTRADA

```
80 00 00 00 07 00 00 00 00 08 00 00 00 00 40 06
00 00 00 00 83 00 00 00 00 08 40 06 00 00 80 0C
00 00 00 00 82 00 00 00 00 08 C0 12 00 00 00 01
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
55 AA
```

SALIDA ESPERADA

Partición 1: activa, tipo 07 (NTFS/exFAT), sector inicial 2048, 104857600 sectores (50.00 GiB)
Partición 2: tipo 83 (Linux), sector inicial 104859648, 209715200 sectores (100.00 GiB)
Partición 3: tipo 82 (Linux swap), sector inicial 314574848, 16777216 sectores (8.00 GiB)
Firma: correcta (55 AA)

Ejemplo: Disco GPT

ENTRADA

```
00 00 00 00 EE 00 00 00 01 00 00 00 FF FF FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
55 AA
```

SALIDA ESPERADA

Partición 1: tipo EE (GPT de protección), sector inicial 1, 4294967295 sectores (2048.00 GiB)
El disco usa GPT: esta entrada solo protege el disco de herramientas antiguas
Firma: correcta (55 AA)

25. Capacidad y tolerancia de un RAID

Medio · Sistemas de Ficheros y Copias de Seguridad · apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento

Cada línea es «nivel discos tamaño_GB», con el nivel RAID0, RAID1, RAID5, RAID6 o RAID10. Calcula la capacidad útil (RAID0: suma de todos; RAID1: un disco, porque todos son copias; RAID5: $n - 1$ discos; RAID6: $n - 2$; RAID10: la mitad), la eficiencia (útil / total, en % sin decimales) y los fallos de disco que soporta sin perder datos (0, $n - 1$, 1, 2 y en RAID10 «1 fallo por pareja (hasta $n/2$)»). Muestra «RAID5 con 4 × 2000 GB: 6000 GB útiles (75 %), tolera 1 fallo» (y «fallos» en plural salvo cuando es exactamente 1). Si faltan discos (mínimos: 2, 2, 3, 4 y 4; RAID10 además un número par), muestra «RAID10 con 5 discos: no válido (necesita al menos 4 discos y un número par)», y para un nivel que no existe, «nivel: nivel desconocido».

Código de partida (python)

```
import sys

MINIMO = {"RAID0": 2, "RAID1": 2, "RAID5": 3, "RAID6": 4, "RAID10": 4}

# TODO: calcula la capacidad útil, la eficiencia y los fallos que soporta cada configuración
```

Ejemplo: Niveles habituales

ENTRADA

```
RAID0 2 1000
RAID1 2 4000
RAID5 4 2000
RAID6 6 8000
RAID10 4 1000
```

SALIDA ESPERADA

```
RAID0 con 2 x 1000 GB: 2000 GB útiles (100 %),
tolera 0 fallos
RAID1 con 2 x 4000 GB: 4000 GB útiles (50 %),
tolera 1 fallo
RAID5 con 4 x 2000 GB: 6000 GB útiles (75 %),
tolera 1 fallo
RAID6 con 6 x 8000 GB: 32000 GB útiles (67 %),
tolera 2 fallos
RAID10 con 4 x 1000 GB: 2000 GB útiles (50 %),
tolera 1 fallo por pareja (hasta 2)
```

Ejemplo: Configuraciones no válidas

ENTRADA

```
RAID5 2 1000
RAID10 5 2000
RAID7 4 1000
```

SALIDA ESPERADA

```
RAID5 con 2 discos: no válido (necesita al menos
3 discos)
RAID10 con 5 discos: no válido (necesita al menos
4 discos y un número par)
RAID7: nivel desconocido
```

26. Simulador de estrategias de copia

Difícil · Sistemas de Ficheros y Copias de Seguridad · apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento

La primera línea es «datos N» (GB que se copian), la segunda «cambio P» (porcentaje de los datos que se modifica cada día, siempre ficheros distintos) y la tercera «plan ...» con un tipo de copia por día: C (completa: copia todo), I (incremental: lo que ha cambiado desde la copia anterior, P % de los datos) o D (diferencial: todo lo que ha cambiado desde la última completa, P % por cada día transcurrido desde ella, como máximo el 100 %). Muestra «Día N (T): X GB» (1 decimal) por cada día, «Total ocupado: X GB» y «Para restaurar el último día: día A, día B...» con las copias necesarias: la última completa, la última diferencial posterior (si la hay) y las incrementales posteriores a esa diferencial (o a la completa si no hay diferencial). Una I o una D antes de cualquier completa muestran «Día N: ERROR, una copia T necesita una completa anterior» y no ocupan nada.

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
datos = float(lineas[0][1])
cambio = float(lineas[1][1])
plan = lineas[2][1:]          # C = completa, I = incremental, D = diferencial, un día cada una

# TODO: tamaño de cada copia, total ocupado y copias necesarias para restaurar el último día
```

Ejemplo: Semana con incrementales

ENTRADA

```
datos 200
cambio 5
plan C I I I I I I
```

SALIDA ESPERADA

```
Día 1 (C): 200.0 GB
Día 2 (I): 10.0 GB
Día 3 (I): 10.0 GB
Día 4 (I): 10.0 GB
Día 5 (I): 10.0 GB
Día 6 (I): 10.0 GB
Día 7 (I): 10.0 GB
Total ocupado: 260.0 GB
Para restaurar el último día: día 1, día 2, día
3, día 4, día 5, día 6, día 7
```

Ejemplo: Semana con diferenciales

ENTRADA

```
datos 200
cambio 5
plan C D D D D D D
```

SALIDA ESPERADA

```
Día 1 (C): 200.0 GB
Día 2 (D): 10.0 GB
Día 3 (D): 20.0 GB
Día 4 (D): 30.0 GB
Día 5 (D): 40.0 GB
Día 6 (D): 50.0 GB
Día 7 (D): 60.0 GB
Total ocupado: 410.0 GB
Para restaurar el último día: día 1, día 7
```

Ejemplo: Mixta con dos completas

ENTRADA

```
datos 500
cambio 10
plan C I I D I C I D I
```

SALIDA ESPERADA

```
Día 1 (C): 500.0 GB
Día 2 (I): 50.0 GB
Día 3 (I): 50.0 GB
Día 4 (D): 150.0 GB
Día 5 (I): 50.0 GB
Día 6 (C): 500.0 GB
Día 7 (I): 50.0 GB
Día 8 (D): 100.0 GB
Día 9 (I): 50.0 GB
Total ocupado: 1500.0 GB
Para restaurar el último día: día 6, día 8, día 9
```

27. ¿Es buena esta copia de seguridad?

Muy difícil · Sistemas de Ficheros y Copias de Seguridad · apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento

Para comprobar una copia se compara un manifiesto del original con otro de la copia: cada línea es «ruta tamaño hash» (como lo que genera sha256sum, abreviado). La entrada tiene los dos manifiestos separados por una línea «---». Informa, en este orden: los ficheros MOVIDOS (faltan en una ruta pero aparecen en otra de la copia con el mismo hash) como «MOVIDO ruta -> nueva_ruta»; los que FALTAN; los DISTINTOS (misma ruta, distinto hash) con «(X bytes en el original, Y en la copia)»; y los que SOBRAN; cada grupo ordenado alfabéticamente. Termina con «Correctos: N de M · Datos del original: X MiB» (2 decimales) y «Copia VÁLIDA» si no falta ni difiere nada (los movidos y sobrantes no la invalidan) o «Copia NO VÁLIDA».

Código de partida (python)

```
import sys

texto = sys.stdin.read().split("---")
original = [l.split() for l in texto[0].splitlines() if l.strip()] # ruta tamaño hash
copia = [l.split() for l in texto[1].splitlines() if l.strip()]

# TODO: compara los dos manifiestos e informa de las diferencias
```

Ejemplo: Copia perfecta

ENTRADA

```
docs/contrato.pdf 482133 9f2a
docs/nominas/enero.xlsx 22100 77b1
docs/nominas/febrero.xlsx 22340 01ce
fotos/logo.png 150000 aa10
bd/clientes.sql 5242880 c3d4
---
docs/contrato.pdf 482133 9f2a
docs/nominas/enero.xlsx 22100 77b1
docs/nominas/febrero.xlsx 22340 01ce
fotos/logo.png 150000 aa10
bd/clientes.sql 5242880 c3d4
```

SALIDA ESPERADA

```
Correctos: 5 de 5 · Datos del original: 5.65 MiB
Copia VÁLIDA
```

Ejemplo: Con errores

ENTRADA

```
docs/contrato.pdf 482133 9f2a
docs/nominas/enero.xlsx 22100 77b1
docs/nominas/febrero.xlsx 22340 01ce
fotos/logo.png 150000 aa10
bd/clientes.sql 5242880 c3d4
---
docs/contrato.pdf 482133 9f2a
docs/nominas/enero.xlsx 22100 0000
fotos/logo.png 150000 aa10
bd/clientes.sql 5242880 c3d4
tmp/cache.bin 10 ffff
```

SALIDA ESPERADA

```
FALTA docs/nominas/febrero.xlsx
DISTINTO docs/nominas/enero.xlsx (22100 bytes en
el original, 22100 en la copia)
SOBRA tmp/cache.bin
Correctos: 3 de 5 · Datos del original: 5.65 MiB
Copia NO VÁLIDA
```

28. ¿Llega este cable?

Medio · Redes Informáticas · apuntesdam.com/subject/sistemas/topic/redes-informaticas

Cada línea describe un tramo de cable: «medio metros velocidad», con el medio cat5e, cat6, cat6a, cat8 (par trenzado), om3, om4 (fibra multimodo) u os2 (fibra monomodo) y la velocidad en Gbit/s con una G (1G, 10G...). El diccionario MAXIMO indica la distancia máxima de cada medio a cada velocidad que admite. Muestra «medio, M m a VG: SÍ (máximo X m)» si el tramo es válido; «...: NO (máximo X m; sirve a, b...)» si es demasiado largo, con los medios que sí llegarían en orden alfabético (o «ninguno»); «...: NO (no admite VG; velocidades inferiores válidas: ...)» si el medio no admite esa velocidad (o «ninguna»); y «medio: medio desconocido» si no existe.

Código de partida (python)

```
import sys

# Distancia máxima en metros de cada medio para cada velocidad (Gbit/s)
MAXIMO = {
    "cat5e": {1: 100},
    "cat6": {1: 100, 10: 55},
    "cat6a": {1: 100, 10: 100},
    "cat8": {10: 30, 25: 30, 40: 30},
    "om3": {1: 550, 10: 300, 40: 100},
    "om4": {1: 550, 10: 400, 40: 150},
    "os2": {1: 10000, 10: 10000, 40: 10000},
}

# TODO: decide si cada tramo de cable es válido
```

Ejemplo: Oficina

ENTRADA

```
cat6 70 1G
cat6 70 10G
cat6a 90 10G
om4 350 10G
```

SALIDA ESPERADA

```
cat6, 70 m a 1G: SÍ (máximo 100 m)
cat6, 70 m a 10G: NO (máximo 55 m; sirve cat6a,
om3, om4, os2)
cat6a, 90 m a 10G: SÍ (máximo 100 m)
om4, 350 m a 10G: SÍ (máximo 400 m)
```

Ejemplo: Velocidades no admitidas

ENTRADA

```
cat5e 20 10G
cat8 10 1G
coaxial 50 1G
```

SALIDA ESPERADA

```
cat5e, 20 m a 10G: NO (no admite 10G; velocidades
inferiores válidas: 1G)
cat8, 10 m a 1G: NO (no admite 1G; velocidades
inferiores válidas: ninguna)
coaxial: medio desconocido
```

29. La tabla MAC de un switch

Difícil · Redes Informáticas · apuntesdam.com/subject/sistemas/topic/redes-informaticas

Un switch aprende en qué puerto está cada equipo mirando la MAC de origen de las tramas que recibe. La primera línea es «puertos N»; cada línea siguiente, una trama recibida: «puerto MAC_origen MAC_destino». Para cada trama: si la MAC de origen no estaba en la tabla (o estaba en otro puerto, porque el equipo se ha movido), muestra «Aprende MAC en el puerto P». Después, con dos espacios delante: si el destino es FF:FF:FF:FF:FF:FF, «Difusión: inunda por a, b...» (todos los puertos menos el de entrada); si el destino no está en la tabla, «MAC desconocido: inunda por ...»; si está en el mismo puerto de entrada, «MAC está en el mismo puerto: descarta la trama»; y si no, «Reenvía a MAC por el puerto P». Al final muestra «Tabla MAC:» y cada entrada « MAC → puerto P» ordenada por MAC.

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
puertos = int(lineas[0][1])
BROADCAST = "FF:FF:FF:FF:FF:FF"

# TODO: simula cómo aprende y reenvía las tramas un switch
```

Ejemplo: Arranque de una red

ENTRADA

```
puertos 4
1 AA:AA:AA:00:00:01 FF:FF:FF:FF:FF:FF
2 BB:BB:BB:00:00:02 AA:AA:AA:00:00:01
1 AA:AA:AA:00:00:01 BB:BB:BB:00:00:02
3 CC:CC:CC:00:00:03 DD:DD:DD:00:00:04
```

SALIDA ESPERADA

```
Aprende AA:AA:AA:00:00:01 en el puerto 1
  Difusión: inunda por 2, 3, 4
Aprende BB:BB:BB:00:00:02 en el puerto 2
  Reenvía a AA:AA:AA:00:00:01 por el puerto 1
  Reenvía a BB:BB:BB:00:00:02 por el puerto 2
Aprende CC:CC:CC:00:00:03 en el puerto 3
  DD:DD:DD:00:00:04 desconocido: inunda por 1, 2,
  4
Tabla MAC:
  AA:AA:AA:00:00:01 → puerto 1
  BB:BB:BB:00:00:02 → puerto 2
  CC:CC:CC:00:00:03 → puerto 3
```

30. Puntos únicos de fallo de una red

Muy difícil · Redes Informáticas · apuntesdam.com/subject/sistemas/topic/redes-informaticas

La primera línea es «raíz NOMBRE»: el equipo por el que se sale a Internet (normalmente el router). Cada línea siguiente es un enlace «enlace A B» entre dos equipos (los enlaces funcionan en los dos sentidos). Para cada equipo con dos o más enlaces que no sea la raíz, en orden alfabético, calcula qué equipos dejarían de poder llegar a la raíz si ese equipo fallara: muestra «Si falla X: sin red a, b...» (en orden alfabético) o «Si falla X: nadie pierde la red (hay caminos alternativos)». Al final, «Puntos únicos de fallo: N» con los equipos cuyo fallo deja a alguien sin red.

Código de partida (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
raiz = lineas[0][1] # equipo desde el que se sale a Internet
enlaces = [(a, b) for _, a, b in lineas[1:]]

# TODO: para cada equipo de interconexión, ¿quién se queda sin red si falla?
```

Ejemplo: Estrella extendida

ENTRADA

```
raiz router
enlace router sw-core
enlace sw-core sw-planta1
enlace sw-core sw-planta2
enlace sw-planta1 pc1
enlace sw-planta1 pc2
enlace sw-planta2 pc3
enlace sw-planta2 impresora
```

SALIDA ESPERADA

```
Si falla sw-core: sin red impresora, pc1, pc2,
pc3, sw-planta1, sw-planta2
Si falla sw-planta1: sin red pc1, pc2
Si falla sw-planta2: sin red impresora, pc3
Puntos únicos de fallo: 3
```

Ejemplo: Con enlace redundante

ENTRADA

```
raiz router
enlace router sw-a
enlace router sw-b
enlace sw-a sw-b
enlace sw-a pc1
enlace sw-b pc2
enlace sw-b ap
enlace ap portatil
```

SALIDA ESPERADA

```
Si falla ap: sin red portatil
Si falla sw-a: sin red pc1
Si falla sw-b: sin red ap, pc2, portatil
Puntos únicos de fallo: 3
```

31. ¿Pueden hablar sin router?

Medio · Configuración de Redes TCP/IP · apuntesdam.com/subject/sistemas/topic/configuracion-redes

Dos equipos solo se comunican directamente si están en la misma red, es decir, si su IP AND la máscara da el mismo resultado. Cada línea tiene «IP/prefijo IP2»: el primer equipo con su prefijo y la IP del segundo (que usa la misma máscara). Muestra «IP y IP2: misma red RED/prefijo, comunicación directa» o «IP y IP2: redes distintas (RED1 y RED2), hace falta un router». Las funciones a_entero y a_texto convierten entre la notación con puntos y un entero de 32 bits.

Código de partida (python)

```
import sys

def a_entero(ip):
    a, b, c, d = (int(x) for x in ip.split("."))
    return (a << 24) | (b << 16) | (c << 8) | d

def a_texto(n):
    return ".".join(str((n >> s) & 255) for s in (24, 16, 8, 0))

# TODO: ¿están los dos equipos en la misma red?
```

Ejemplo: Redes /24 y /26

ENTRADA

```
192.168.1.10/24 192.168.1.200
192.168.1.10/26 192.168.1.70
192.168.1.65/26 192.168.1.126
```

SALIDA ESPERADA

```
192.168.1.10 y 192.168.1.200: misma red
192.168.1.0/24, comunicación directa
192.168.1.10 y 192.168.1.70: redes distintas
(192.168.1.0 y 192.168.1.64), hace falta un
router
192.168.1.65 y 192.168.1.126: misma red
192.168.1.64/26, comunicación directa
```

Ejemplo: Prefijos que cortan otros octetos

ENTRADA

```
172.16.37.10/20 172.16.40.1
172.16.37.10/20 172.16.48.1
10.1.2.3/8 10.200.0.1
```

SALIDA ESPERADA

```
172.16.37.10 y 172.16.40.1: misma red
172.16.32.0/20, comunicación directa
172.16.37.10 y 172.16.48.1: redes distintas
(172.16.32.0 y 172.16.48.0), hace falta un router
10.1.2.3 y 10.200.0.1: misma red 10.0.0.0/8,
comunicación directa
```

32. Abreviar y expandir direcciones IPv6

Difícil · Configuración de Redes TCP/IP · apuntesdam.com/subject/sistemas/topic/configuracion-redes

Cada línea es una dirección IPv6. Si está completa (8 grupos sin «::»), abréviala: quita los ceros a la izquierda de cada grupo (un grupo todo a ceros queda como 0) y sustituye por «::» la serie más larga de dos o más grupos seguidos a cero (si hay dos igual de largas, la primera; solo se usa «::» una vez). Si está abreviada, expándela a 8 grupos de 4 cifras. Usa siempre minúsculas. Muestra «entrada → resultado».

Código de partida (python)

```
import sys

# TODO: abrevia las direcciones IPv6 completas y expande las abreviadas
```

Ejemplo: Abreviar

ENTRADA

```
2001:0db8:0000:0000:0000:ff00:0042:8329
fe80:0000:0000:0000:0202:b3ff:fe1e:8329
2001:0DB8:0000:0001:0000:0000:0000:0001
```

SALIDA ESPERADA

```
2001:0db8:0000:0000:0000:ff00:0042:8329 →
2001:db8::ff00:42:8329
fe80:0000:0000:0000:0202:b3ff:fe1e:8329 →
fe80::202:b3ff:fe1e:8329
2001:0DB8:0000:0001:0000:0000:0000:0001 →
2001:db8:0:1::1
```

Ejemplo: Expandir

ENTRADA

```
2001:db8::ff00:42:8329
::1
fe80::
```

SALIDA ESPERADA

```
2001:db8::ff00:42:8329 →
2001:0db8:0000:0000:0000:ff00:0042:8329
::1 → 0000:0000:0000:0000:0000:0000:0000:0001
fe80:: → fe80:0000:0000:0000:0000:0000:0000:0000
```

33. Calculadora VLSM

Muy difícil · Configuración de Redes TCP/IP · apuntesdam.com/subject/sistemas/topic/configuracion-redes

La primera línea es «red A.B.C.D/p», la red que hay que repartir. Cada línea siguiente es «nombre hosts»: un departamento y los equipos que necesita. Con VLSM, asigna las subredes de la que más hosts necesita a la que menos (a igualdad, por nombre), cada una del tamaño mínimo que alcance ($2^{\text{bits}} - 2 \geq \text{hosts}$, con al menos 2 bits) y una detrás de otra desde el principio de la red. Para cada una muestra «nombre (hosts): RED/prefijo, hosts PRIMERO - ÚLTIMO, difusión DIF, sobran N»; si ya no cabe, «nombre: no hay espacio para N hosts» (y se sigue con las demás). Termina con «Libre: N direcciones desde IP» o «Libre: 0 direcciones».

Código de partida (python)

```
import sys

def a_entero(ip):
    a, b, c, d = (int(x) for x in ip.split("."))
    return (a << 24) | (b << 16) | (c << 8) | d

def a_texto(n):
    return ".".join(str((n >> s) & 255) for s in (24, 16, 8, 0))

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
base_ip, base_prefijo = lineas[0][1].split("/")
departamentos = [(nombre, int(hosts)) for nombre, hosts in lineas[1:]]

# TODO: reparte la red con VLSM, de la subred que más hosts necesita a la que menos
```

Ejemplo: Ejemplo del tema

ENTRADA

```
red 192.168.50.0/24
Desarrollo 100
Administracion 50
Direccion 10
Enlace 2
```

SALIDA ESPERADA

```
Desarrollo (100): 192.168.50.0/25, hosts
192.168.50.1 - 192.168.50.126, difusión
192.168.50.127, sobran 26
Administracion (50): 192.168.50.128/26, hosts
192.168.50.129 - 192.168.50.190, difusión
192.168.50.191, sobran 12
Direccion (10): 192.168.50.192/28, hosts
192.168.50.193 - 192.168.50.206, difusión
192.168.50.207, sobran 4
Enlace (2): 192.168.50.208/30, hosts
192.168.50.209 - 192.168.50.210, difusión
192.168.50.211, sobran 0
Libre: 44 direcciones desde 192.168.50.212
```

Ejemplo: Instituto en una /22

ENTRADA

```
red 10.20.0.0/22
Aulas 400
Profesorado 120
Wifi 200
Servidores 20
Impresoras 12
```

SALIDA ESPERADA

```
Aulas (400): 10.20.0.0/23, hosts 10.20.0.1 -
10.20.1.254, difusión 10.20.1.255, sobran 110
Wifi (200): 10.20.2.0/24, hosts 10.20.2.1 -
10.20.2.254, difusión 10.20.2.255, sobran 54
Profesorado (120): 10.20.3.0/25, hosts 10.20.3.1
- 10.20.3.126, difusión 10.20.3.127, sobran 6
Servidores (20): 10.20.3.128/27, hosts
10.20.3.129 - 10.20.3.158, difusión 10.20.3.159,
sobran 10
Impresoras (12): 10.20.3.160/28, hosts
10.20.3.161 - 10.20.3.174, difusión 10.20.3.175,
sobran 2
Libre: 80 direcciones desde 10.20.3.176
```

34. Permisos efectivos de una carpeta compartida

Medio · Servicios de Red, Recursos Compartidos y Seguridad · apuntesdam.com/subject/sistemas/topic/servicios-red-seguridad

En Windows, a una carpeta compartida por la red se le aplican dos capas de permisos: los del recurso compartido y los NTFS. Dentro de cada capa, un usuario suma los permisos de todos sus grupos (se queda con el mayor), una denegación explícita gana a cualquier permiso, y el permiso efectivo es el más restrictivo de las dos capas. Las líneas pueden ser «compartido GRUPO nivel», «ntfs GRUPO nivel», «denegar GRUPO» (denegación NTFS total) o «usuario NOMBRE grupo1,grupo2...». Los niveles, de menor a mayor, son ninguno, lectura, modificar y control. Para cada usuario, en el orden de la entrada, muestra «nombre: compartido X, NTFS Y → efectivo Z».

Código de partida (python)

```
import sys

NIVELES = ["ninguno", "lectura", "modificar", "control"]

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]

# TODO: calcula el permiso efectivo de cada usuario sobre la carpeta compartida
```

Ejemplo: Departamentos

ENTRADA

```
compartido Todos modificar
ntfs Ventas modificar
ntfs Direccion control
ntfs Todos lectura
usuario ana Ventas,Todos
usuario luis Todos
usuario marta Direccion,Todos
```

SALIDA ESPERADA

```
ana: compartido modificar, NTFS modificar →
efectivo modificar
luis: compartido modificar, NTFS lectura →
efectivo lectura
marta: compartido modificar, NTFS control →
efectivo modificar
```

Ejemplo: Denegación y compartido restrictivo

ENTRADA

```
compartido Todos lectura
compartido Admins control
ntfs Admins control
ntfs Todos modificar
denegar Becarios
usuario pedro Todos,Becarios
usuario eva Todos
usuario root Admins
```

SALIDA ESPERADA

```
pedro: compartido lectura, NTFS ninguno →
efectivo ninguno
eva: compartido lectura, NTFS modificar →
efectivo lectura
root: compartido control, NTFS control → efectivo
control
```

35. Un cortafuegos de primera coincidencia

Difícil · Servicios de Red, Recursos Compartidos y Seguridad · apuntesdam.com/subject/sistemas/topic/servicios-red-seguridad

Un cortafuegos revisa sus reglas en orden y aplica la primera que coincide; si ninguna coincide, aplica la política por defecto. La entrada tiene dos partes separadas por «---». La primera empieza con «política allow|deny» y sigue con reglas «allow|deny protocolo puerto from red» (protocolo tcp, udp o any; puerto un número o any; red en CIDR, una IP sola o any). La segunda son paquetes entrantes «protocolo ip_origen puerto_destino». Para cada paquete muestra «proto ip → puerto: ALLOW (regla N)», «...: DENY (regla N)» o «...: ACCIÓN (política por defecto)».

Código de partida (python)

```
import sys

def a_entero(ip):
    a, b, c, d = (int(x) for x in ip.split("."))
    return (a << 24) | (b << 16) | (c << 8) | d

texto = sys.stdin.read().split("---")
reglas = [l.split() for l in texto[0].splitlines() if l.strip()] # la primera línea es «política
allow|deny»
paquetes = [l.split() for l in texto[1].splitlines() if l.strip()]

# TODO: aplica las reglas en orden a cada paquete (gana la primera que coincide)
```

Ejemplo: Servidor web

ENTRADA

```
politica deny
allow tcp 22 from 192.168.1.0/24
allow tcp 443 from any
allow tcp 80 from any
deny tcp 3306 from any
allow udp 53 from 10.0.0.0/8
---
tcp 192.168.1.50 22
tcp 203.0.113.7 22
tcp 203.0.113.7 443
tcp 192.168.1.50 3306
udp 10.20.30.40 53
udp 8.8.8.8 53
```

SALIDA ESPERADA

```
tcp 192.168.1.50 → 22: ALLOW (regla 1)
tcp 203.0.113.7 → 22: DENY (política por defecto)
tcp 203.0.113.7 → 443: ALLOW (regla 2)
tcp 192.168.1.50 → 3306: DENY (regla 4)
udp 10.20.30.40 → 53: ALLOW (regla 5)
udp 8.8.8.8 → 53: DENY (política por defecto)
```

36. ¿Cuánto aguantaría esta contraseña?

Muy difícil · Servicios de Red, Recursos Compartidos y Seguridad · apuntesdam.com/subject/sistemas/topic/servicios-red-seguridad

Cada línea es una contraseña. Si en minúsculas está en el DICCIONARIO, o lo está tras quitar los dígitos, puntos y signos de exclamación del final (Barcelona2024!), muestra «clave: muy débil (está en los diccionarios de contraseñas)». Si no, calcula su entropía: longitud × log2(juego), donde el juego suma 26 si hay minúsculas, 26 si hay mayúsculas, 10 si hay dígitos y 33 si hay otros símbolos; si tiene muy pocos caracteres distintos (no más de un tercio de su longitud), divide la entropía entre 2. Clasifícala como débil (< 40 bits), media (< 60), fuerte (< 80) o muy fuerte, y estima el tiempo medio de un ataque de fuerza bruta a 10¹⁰ intentos por segundo: 2^{entropía} / 2 / 10¹⁰ segundos, expresado en años, días, horas o minutos (la mayor unidad con valor ≥ 1; con 1 decimal si es menor que 100, y si no, redondeado y con punto de miles, como «857.012 años»), «más de un millón de ...» si llega al millón, «menos de 1 minuto» o «instantáneo» (< 1 s). Muestra «clave: nivel (X bits, TIEMPO por fuerza bruta)», con los bits con 1 decimal.

Código de partida (python)

```
import sys
import math

DICCIONARIO = {"password", "123456", "qwerty", "admin", "iloveyou", "dragon", "monkey", "letmein",
               "futbol", "barcelona", "madrid", "contraseña", "hola", "princesa", "superman"}

# TODO: estima la fortaleza de cada contraseña
```

Ejemplo: Varias contraseñas

ENTRADA

```
password
Barcelona2024!
Gato
T3rue1!
aaaaaaaaaaaa
caballo-bateria-grapa-correcto
```

SALIDA ESPERADA

```
password: muy débil (está en los diccionarios de
contraseñas)
Barcelona2024!: muy débil (está en los
diccionarios de contraseñas)
Gato: débil (22.8 bits, instantáneo por fuerza
bruta)
T3rue1!: media (46.0 bits, 58.2 minutos por
fuerza bruta)
aaaaaaaaaaaa: débil (28.2 bits, instantáneo por
fuerza bruta)
caballo-bateria-grapa-correcto: muy fuerte (176.5
bits, más de un millón de años por fuerza bruta)
```

Ejemplo: Largas y aleatorias

ENTRADA

```
Xk9#mP2$vL8q
z7Q!r4T@w1E%y6U^
```

SALIDA ESPERADA

```
Xk9#mP2$vL8q: fuerte (78.8 bits, 856.735 años por
fuerza bruta)
z7Q!r4T@w1E%y6U^: muy fuerte (105.1 bits, más de
un millón de años por fuerza bruta)
```

37. Informe del inventario

Medio · Documentación y Aplicaciones Ofimáticas · apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica

La primera línea es «hoy=AAAA-MM-DD» y el resto un CSV separado por punto y coma con cabecera «codigo;tipo;aula;precio;fin_garantia». Agrupa los equipos por aula (en orden alfabético) y muestra «AULA: N equipos, X €» (2 decimales; «1 equipo» en singular), añadiendo «, sin garantía: código1, código2» con los códigos ordenados si alguno tiene la garantía vencida (fecha de fin anterior a hoy). Termina con «TOTAL: N equipos, X €, M sin garantía (P %)» con el porcentaje sin decimales. Es lo que harías con una tabla dinámica en la hoja de cálculo.

Código de partida (python)

```
import sys
import csv
from datetime import date

lineas = sys.stdin.read().splitlines()
hoy = date.fromisoformat(lineas[0].split("=")[1])
filas = list(csv.DictReader(lineas[1:], delimiter=";")) # codigo;tipo;aula;precio;fin_garantia

# TODO: informe por aula con número de equipos, valor y garantías caducadas
```

Ejemplo: Instituto

ENTRADA

```
hoy=2026-10-01
codigo;tipo;aula;precio;fin_garantia
PC-201;sobremesa;2.01;780;2027-05-01
PC-202;sobremesa;2.01;780;2026-03-15
IMP-01;impresora;2.01;320.5;2025-11-30
PC-110;sobremesa;1.10;650;2028-01-10
PRY-01;proyector;1.10;899.99;2026-12-31
SRV-01;servidor;CPD;3400;2026-09-30
```

SALIDA ESPERADA

```
1.10: 2 equipos, 1549.99 €
2.01: 3 equipos, 1880.50 €, sin garantía: IMP-01,
PC-202
CPD: 1 equipo, 3400.00 €, sin garantía: SRV-01
TOTAL: 6 equipos, 6830.49 €, 3 sin garantía (50
%)
```

38. Un índice automático para la documentación

Difícil · Documentación y Aplicaciones Ofimáticas · apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica

La entrada es un documento en Markdown. Genera su índice con los títulos de nivel 2 (##) y 3 (###), numerados (1., 1.1., 1.2., 2....; la numeración de nivel 3 se reinicia en cada nivel 2) y con un enlace a su ancla: «1. [Requisitos previos] (#requisitos-previos)» para el nivel 2 y, con tres espacios delante, « 1.1. [Hardware] (#hardware)» para el nivel 3. El ancla se obtiene del título en minúsculas, sin tildes, eliminando lo que no sea letra, número, espacio o guion, y cambiando los espacios por guiones; si un ancla se repite, la segunda vez se le añade «-1», la tercera «-2»... Ignora el título de nivel 1 y cualquier línea dentro de un bloque de código (entre líneas que empiezan por ```).

Código de partida (python)

```
import sys
import re
import unicodedata

lineas = sys.stdin.read().splitlines()

# TODO: genera el índice numerado de los títulos (## y ###) con su ancla
```

Ejemplo: Manual

ENTRADA

```
# Manual de instalación del servidor

## Requisitos previos
Texto.
### Hardware
### Software
## Instalación
``bash
## esto es un comentario dentro de un bloque de
código, no un título
```

Configuración de la red
Configuración de la red
Copias de seguridad: ¿cómo y cuándo?
```

#### SALIDA ESPERADA

```
1. [Requisitos previos] (#requisitos-previos)
 1.1. [Hardware] (#hardware)
 1.2. [Software] (#software)
2. [Instalación] (#instalacion)
 2.1. [Configuración de la red] (#configuracion-
de-la-red)
 2.2. [Configuración de la red] (#configuracion-
de-la-red-1)
3. [Copias de seguridad: ¿cómo y cuándo?]
 (#copias-de-seguridad-como-y-cuando)
```

## 39. Un conversor de Markdown a HTML

Muy difícil · Documentación y Aplicaciones Ofimáticas · [apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica](https://apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica)

Convierte un documento Markdown a HTML con estas reglas. Bloques: los títulos (# a #####) pasan a <h1>...<h6>; las líneas que empiezan por «- » o «\* » forman una lista <ul> y las que empiezan por «1. », «2. »... una <ol>, con cada elemento en un <li>; las líneas seguidas de texto normal forman un párrafo <p> (unidas con un espacio); una línea en blanco cierra el párrafo o la lista; y lo que hay entre dos líneas que empiezan por `` es un bloque <pre><code>...</code></pre> en el que no se interpreta nada. En línea, dentro de títulos, párrafos y elementos de lista:

código` → `<code>`, **negrita** → `<strong>`, *\*cursiva\** → `<em>` y [texto](url) → `<a href="url">texto</a>`. Antes de nada, escapa los caracteres `<`, `>` y `&` (html.escape con quote=False), también dentro de los bloques de código. Muestra cada bloque en su propia línea, y las etiquetas de apertura y cierre de las listas en líneas aparte.

**Código de partida** (python)

```
import sys
import re
import html

lineas = sys.stdin.read().splitlines()

TODO: convierte el Markdown en HTML
```

### Ejemplo: Procedimiento

#### ENTRADA

```
Alta de un usuario

Este procedimiento lo sigue el técnico de guardia cuando llega una petición aprobada.

Pasos

1. Comprueba la petición en el [gestor de incidencias](https://tickets.empresa.es).
2. Crea la cuenta con `New-ADUser`.
3. Añádela a sus grupos.

Notas:
- Nunca uses contraseñas <123456>.
- Avisa a RR. HH. al terminar.

```powershell
New-ADUser -Name "Ana Ruiz" -Enabled $true
if ($x -lt 5) { "<ok>" }
```
```

#### SALIDA ESPERADA

```
<h1>Alta de un usuario</h1>
<p>Este procedimiento lo sigue el técnico de guardia cuando llega una petición aprobada.</p>
<h2>Pasos</h2>

Comprueba la petición en el gestor de incidencias.
Crea la cuenta con New-ADUser.
Añádela a sus grupos.

<p>Notas:</p>

Nunca uses contraseñas <123456>.
Avisa a RR. HH. al terminar.

<pre><code>New-ADUser -Name "Ana Ruiz" -Enabled $true
if ($x -lt 5) { "<ok>" }</code></pre>
```

## Sistemas de Gestión Empresarial

### 40. El embudo de ventas del CRM

Medio · Sistemas ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/sistemas-erp-crm](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/sistemas-erp-crm)

Cada línea de la entrada es una oportunidad del CRM con el formato «nombre;etapa;importe». Las etapas, en orden, y su probabilidad de cierre son: Nuevo 10 %, Calificado 30 %, Propuesta 50 %, Negociación 75 %, Ganado 100 % y Perdido 0 %. Muestra, para cada etapa en ese orden (aunque no tenga oportunidades), «Etapas: N oportunidades, importe €» con el importe total con 2 decimales. Después, «Previsión ponderada: X €» (la suma de importe × probabilidad de las oportunidades abiertas, es decir, sin contar Ganado ni Perdido) y «Tasa de conversión: X %» (ganadas entre ganadas más perdidas, con 1 decimal; si no hay ninguna cerrada, «Tasa de conversión: -»).

**Código de partida** (python)

```
import sys

ETAPAS = [("Nuevo", 10), ("Calificado", 30), ("Propuesta", 50), ("Negociación", 75), ("Ganado", 100), ("Perdido", 0)]

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

TODO: recuento por etapa, previsión ponderada y tasa de conversión
```

### Ejemplo: Embudo completo

#### ENTRADA

Bodega Ruiz;Propuesta;12000  
Hotel Mar;Nuevo;3000  
Clínica Sol;Ganado;8000  
Taller Paco;Perdido;1500  
Academia Luz;Negociación;6000  
Gimnasio Up;Ganado;4500

#### SALIDA ESPERADA

Nuevo: 1 oportunidades, 3000.00 €  
Calificado: 0 oportunidades, 0.00 €  
Propuesta: 1 oportunidades, 12000.00 €  
Negociación: 1 oportunidades, 6000.00 €  
Ganado: 2 oportunidades, 12500.00 €  
Perdido: 1 oportunidades, 1500.00 €  
Previsión ponderada: 10800.00 €  
Tasa de conversión: 66.7 %

### Ejemplo: Ninguna cerrada

#### ENTRADA

A;Calificado;1000  
B;Calificado;2500.5

#### SALIDA ESPERADA

Nuevo: 0 oportunidades, 0.00 €  
Calificado: 2 oportunidades, 3500.50 €  
Propuesta: 0 oportunidades, 0.00 €  
Negociación: 0 oportunidades, 0.00 €  
Ganado: 0 oportunidades, 0.00 €  
Perdido: 0 oportunidades, 0.00 €  
Previsión ponderada: 1050.15 €  
Tasa de conversión: -

## 41. Coste total de propiedad a varios años

Difícil · Sistemas ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/sistemas-erp-crm](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/sistemas-erp-crm)

La primera línea es «años N». Cada línea siguiente describe una opción de ERP:

«nombre;licencia\_usuario\_mes;usuarios;implantación;formación;hardware;mantenimiento\_anual». El coste total de propiedad (TCO) es:  $\text{licencia} \times \text{usuarios} \times 12 \times \text{años} + \text{implantación} + \text{formación} + \text{hardware} + \text{mantenimiento\_anual} \times \text{años}$ . Muestra las opciones ordenadas de menor a mayor TCO (a igualdad, por nombre) con el formato «nombre: X €» (2 decimales), y después «Recomendada: nombre» (la más barata) y «Ahorro frente a la más cara: X € (Y %)\», con el porcentaje respecto a la más cara y 1 decimal. Si solo hay una opción, no muestres la línea de ahorro.

**Código de partida** (python)

```
import sys

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

TODO: calcula el TCO de cada opción y recomienda
```

### Ejemplo: Tres opciones a 5 años

#### ENTRADA

años 5  
Odoo Enterprise;31.1;20;18000;4000;0;2500  
Business Central;70;20;25000;5000;0;3000  
Odoo Community on-  
premise;0;20;30000;4000;6000;7000

#### SALIDA ESPERADA

Odoo Enterprise: 71820.00 €  
Odoo Community on-premise: 75000.00 €  
Business Central: 129000.00 €  
Recomendada: Odoo Enterprise  
Ahorro frente a la más cara: 57180.00 € (44.3 %)

### Ejemplo: Una sola opción

#### ENTRADA

años 3  
Holded;15;4;500;200;0;0

#### SALIDA ESPERADA

Holded: 2860.00 €  
Recomendada: Holded

## 42. Dimensionar servidores para Odoo

Muy difícil · Sistemas ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/sistemas-erp-crm](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/sistemas-erp-crm)

Cada línea es un escenario: «nombre;usuarios\_concurrentes;núcleos;ram\_gb». Aplica estas reglas: workers = techo(concurrentes / 6), con un mínimo de 2; además hay 1 worker de cron. Núcleos necesarios = techo(workers / 2). Memoria necesaria (MB) = (workers + 1) × 250 + 2048 de PostgreSQL + 1024 del sistema. Para cada escenario muestra, en este orden: «== nombre», «workers = N», «max\_cron\_threads = 1», «CPU: suficiente» o «CPU: insuficiente (faltan N núcleos)», «Memoria: suficiente» o «Memoria: insuficiente (faltan N MB)» y, si algo es insuficiente, «Recomendado: X», el servidor más pequeño del catálogo que cubra ambas cosas: S (2 núcleos, 4 GB), M (4, 8), L (8, 16), XL (16, 32); si ninguno llega, «Recomendado: varios servidores».

Código de partida (python)

```
import sys
import math

CATALOGO = [("S", 2, 4), ("M", 4, 8), ("L", 8, 16), ("XL", 16, 32)]

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

TODO: dimensiona cada escenario
```

### Ejemplo: Servidor justo y otro corto

#### ENTRADA

```
Oficina;20;4;8
Almacén;40;2;4
```

#### SALIDA ESPERADA

```
== Oficina
workers = 4
max_cron_threads = 1
CPU: suficiente
Memoria: suficiente
== Almacén
workers = 7
max_cron_threads = 1
CPU: insuficiente (faltan 2 núcleos)
Memoria: insuficiente (faltan 976 MB)
Recomendado: M
```

### Ejemplo: Pocos usuarios y memoria justa

#### ENTRADA

```
Tienda;3;1;4
```

#### SALIDA ESPERADA

```
== Tienda
workers = 2
max_cron_threads = 1
CPU: suficiente
Memoria: suficiente
```

### Ejemplo: Solo falta memoria

#### ENTRADA

```
Taller;12;2;3
```

#### SALIDA ESPERADA

```
== Taller
workers = 2
max_cron_threads = 1
CPU: suficiente
Memoria: insuficiente (faltan 750 MB)
Recomendado: S
```

### 43. ¿Puede este usuario hacerlo?

Medio · Implantación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/implantacion-erp](http://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/implantacion-erp)

En Odoo, los derechos de acceso (ir.model.access) dan a cada grupo permisos de lectura (r), escritura (w), creación (c) y borrado (d) sobre cada modelo, y un usuario tiene la suma de los permisos de todos sus grupos. La entrada tiene tres secciones. En [accesos], líneas «grupo modelo permisos» (permisos es una combinación de rwcd, o «-» si ninguno; puede haber varias líneas del mismo grupo y modelo, que se suman). En [usuarios], «usuario grupo1,grupo2...». En [consultas], «usuario acción modelo», con acción leer, escribir, crear o borrar. Para cada consulta muestra «usuario acción modelo: PERMITIDO» o «DENEGADO». Un usuario que no existe no tiene ningún permiso.

**Código de partida** (python)

```
import sys

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]
LETRA = {"leer": "r", "escribir": "w", "crear": "c", "borrar": "d"}

TODO: lee las secciones y responde a cada consulta
```

#### Ejemplo: Comercial y administración

##### ENTRADA

```
[accesos]
ventas_usuario sale.order rwc
ventas_usuario res.partner rw
ventas_admin sale.order rwcd
contabilidad account.move rwcd
empleado res.partner r
[usuarios]
ana ventas_usuario,empleado
luis contabilidad,empleado
marta ventas_admin
[consultas]
ana crear sale.order
ana borrar sale.order
marta borrar sale.order
luis leer res.partner
luis escribir res.partner
ana escribir res.partner
marta leer account.move
```

##### SALIDA ESPERADA

```
ana crear sale.order: PERMITIDO
ana borrar sale.order: DENEGADO
marta borrar sale.order: PERMITIDO
luis leer res.partner: PERMITIDO
luis escribir res.partner: DENEGADO
ana escribir res.partner: PERMITIDO
marta leer account.move: DENEGADO
```

#### Ejemplo: Permisos vacíos y usuario desconocido

##### ENTRADA

```
[accesos]
lectura product.product r
nadie product.product -
[usuarios]
pepe nadie
eva lectura,nadie
[consultas]
pepe leer product.product
eva leer product.product
eva borrar product.product
juan leer product.product
```

##### SALIDA ESPERADA

```
pepe leer product.product: DENEGADO
eva leer product.product: PERMITIDO
eva borrar product.product: DENEGADO
juan leer product.product: DENEGADO
```

### 44. Limpiar los clientes antes de migrarlos

Difícil · Implantación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/implantacion-erp](http://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/implantacion-erp)

Antes de importar los clientes del sistema antiguo hay que limpiarlos. La entrada es un CSV con cabecera «nombre;nif;email;telefono» (la línea 1). Para cada fila, normaliza: nombre sin espacios sobrantes; NIF en mayúsculas y sin guiones ni espacios; correo en minúsculas; teléfono sin espacios, puntos ni guiones y sin el prefijo +34 o 0034.

Después valida, en este orden: el NIF (DNI de 8 cifras y letra, o NIE que empiece por X, Y o Z —que valen 0, 1 y 2— seguido de 7 cifras y letra; la letra es TABLA[número % 23]); el correo (algo@algo.dominio con al menos 2 letras en el dominio); y el teléfono (9 cifras empezando por 6, 7, 8 o 9). Si falla, «ERROR línea N: NIF no válido» (o «correo no válido», «teléfono no válido»). Si es válido pero su NIF ya apareció en una fila válida anterior, «DUPLICADO línea N: NIF (ya en la línea M)». Si no, «OK nif;nombre;email;telefono». Al final, «Válidos: A · Errores: B · Duplicados: C». Las líneas se numeran como en el fichero, contando la cabecera.

**Código de partida** (python)

```
import sys
import re

TABLA = "TRWAGMYFPDXBNJZSQVHLCKE"

lineas = [l for l in sys.stdin.read().splitlines() if l.strip()]

TODO: valida y normaliza cada línea (la primera es la cabecera)
```

### Ejemplo: Clientes variados

ENTRADA

```
nombre;nif;email;telefono
Lucía Pérez Gómez ;12345678z;
Lucia.Perez@Correo.ES ;+34 612 345 678
Talleres
Norte;X1234567L;info@talleresnorte.es;942.123.456
Juan Ruiz;12345678A;juan@correo.es;612345678
Marta Díaz;87654321x;marta@correo;612000111
Pedro Gil;00000000T;pedro@correo.es;512345678
L. Pérez;12345678-Z;lucia@otra.es;0034 699 000
111
```

SALIDA ESPERADA

```
OK 12345678Z;Lucía Pérez
Gómez;lucia.perez@correo.es;612345678
OK X1234567L;Talleres
Norte;info@talleresnorte.es;942123456
ERROR línea 4: NIF no válido
ERROR línea 5: correo no válido
ERROR línea 6: teléfono no válido
DUPLICADO línea 7: 12345678Z (ya en la línea 2)
Válidos: 2 · Errores: 3 · Duplicados: 1
```

### Ejemplo: Todo correcto

ENTRADA

```
nombre;nif;email;telefono
Ana;Y1234567X;ana@a.es;700000000
Bea;Z1234567R;bea@b.com;800 00 00 00
```

SALIDA ESPERADA

```
OK Y1234567X;Ana;ana@a.es;700000000
OK Z1234567R;Bea;bea@b.com;800000000
Válidos: 2 · Errores: 0 · Duplicados: 0
```

## 45. Rotación de copias abuelo-padre-hijo

Muy difícil · Implantación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/implantacion-erp](http://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/implantacion-erp)

La estrategia de rotación abuelo-padre-hijo (GFS) conserva copias de seguridad a distintas escalas sin llenar el disco. La entrada son fechas de copias (AAAA-MM-DD), una por línea, desordenadas y quizá repetidas. Se conservan: las 7 copias más recientes (diaria); la copia más reciente de cada una de las 4 semanas ISO más recientes que tengan copias (semanal); y la copia más reciente de cada uno de los 12 meses más recientes que tengan copias (mensual). Muestra todas las fechas, de la más reciente a la más antigua, con «fecha conservar (motivos)» —los motivos en el orden diaria, semanal, mensual, separados por «, »— o «fecha borrar», y al final «Conservadas: N · Borradas: M».

**Código de partida** (python)

```
import sys
from datetime import date

fechas = [date.fromisoformat(l.strip()) for l in sys.stdin.read().split() if l.strip()]

TODO: decide qué copias conservar (diaria, semanal, mensual) y cuáles borrar
```

### Ejemplo: Un mes de copias diarias

#### ENTRADA

2026-09-01  
2026-09-02  
2026-09-03  
2026-09-04  
2026-09-05  
2026-09-06  
2026-09-07  
2026-09-08  
2026-09-09  
2026-09-10  
2026-09-11  
2026-09-12  
2026-09-13  
2026-09-14  
2026-09-15  
2026-09-16  
2026-09-17  
2026-09-18  
2026-09-19  
2026-09-20  
2026-09-21  
2026-09-22  
2026-09-23  
2026-09-24  
2026-09-25  
2026-09-26  
2026-09-27  
2026-09-28  
2026-09-29  
2026-09-30

#### SALIDA ESPERADA

2026-09-30 conservar (diaria, semanal, mensual)  
2026-09-29 conservar (diaria)  
2026-09-28 conservar (diaria)  
2026-09-27 conservar (diaria, semanal)  
2026-09-26 conservar (diaria)  
2026-09-25 conservar (diaria)  
2026-09-24 conservar (diaria)  
2026-09-23 borrar  
2026-09-22 borrar  
2026-09-21 borrar  
2026-09-20 conservar (semanal)  
2026-09-19 borrar  
2026-09-18 borrar  
2026-09-17 borrar  
2026-09-16 borrar  
2026-09-15 borrar  
2026-09-14 borrar  
2026-09-13 conservar (semanal)  
2026-09-12 borrar  
2026-09-11 borrar  
2026-09-10 borrar  
2026-09-09 borrar  
2026-09-08 borrar  
2026-09-07 borrar  
2026-09-06 borrar  
2026-09-05 borrar  
2026-09-04 borrar  
2026-09-03 borrar  
2026-09-02 borrar  
2026-09-01 borrar  
Conservadas: 9 · Borradas: 21

### Ejemplo: Copias sueltas de varios meses

#### ENTRADA

2026-01-15  
2026-01-31  
2026-02-28  
2026-03-01  
2026-03-02  
2026-05-20  
2026-05-21  
2026-05-22

#### SALIDA ESPERADA

2026-05-22 conservar (diaria, semanal, mensual)  
2026-05-21 conservar (diaria)  
2026-05-20 conservar (diaria)  
2026-03-02 conservar (diaria, semanal, mensual)  
2026-03-01 conservar (diaria, semanal)  
2026-02-28 conservar (diaria, mensual)  
2026-01-31 conservar (diaria, semanal, mensual)  
2026-01-15 borrar  
Conservadas: 7 · Borradas: 1

## 46. Encuentra el fallo: el descuento para casi todos

Fácil · Adaptación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp)

A partir de 10 unidades se aplica un descuento del 5 %. Con 12 unidades funciona, pero con 9 también da el descuento. Corrígelo.

#### Código de partida (python)

```
cantidad = input()
if cantidad >= "10":
 print("Descuento del 5 %")
else:
 print("Sin descuento")
```

#### Ejemplo: 12 unidades

ENTRADA

12

SALIDA ESPERADA

Descuento del 5 %

#### Ejemplo: 9 unidades

ENTRADA

9

SALIDA ESPERADA

Sin descuento

#### Ejemplo: 100 unidades

ENTRADA

100

SALIDA ESPERADA

Descuento del 5 %

### 47. Encuentra el fallo: el ranking que no existe

Fácil · Adaptación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp)

El programa ordena los clientes por importe facturado, de mayor a menor, y muestra el ranking. En lugar del ranking da un error. Corrígelo.

Código de partida (python)

```
clientes = [("Carlos", 558.0), ("Ana", 1079.86), ("Javier", 165.4), ("Lucía", 297.0)]
ranking = clientes.sort(key=lambda c: c[1], reverse=True)
for posicion, (nombre, importe) in enumerate(ranking, start=1):
 print(f"{posicion}. {nombre}: {importe:.2f} €")
```

#### Ejemplo: Ranking de facturación

SALIDA ESPERADA

```
1. Ana: 1079.86 €
2. Carlos: 558.00 €
3. Lucía: 297.00 €
4. Javier: 165.40 €
```

### 48. Encuentra el fallo: el stock que aparece en todos los almacenes

Medio · Adaptación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp)

Se crea una tabla de stock con 2 almacenes y 3 productos, todo a 0, y se anotan 5 unidades del producto 1 en el almacén 0. Al imprimirla, las 5 unidades aparecen en los dos almacenes. Corrígelo.

Código de partida (python)

```
almacenes, productos = 2, 3
stock = [[0] * productos] * almacenes
stock[0][1] = 5
for i, fila in enumerate(stock):
 print(f"Almacén {i}: {fila}")
```

#### Ejemplo: Tabla de stock

SALIDA ESPERADA

```
Almacén 0: [0, 5, 0]
Almacén 1: [0, 0, 0]
```

## 49. Un evaluador de dominios de Odoo

Muy difícil · Adaptación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp](http://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp)

Las reglas de automatización, las reglas de registro y los filtros de Odoo usan dominios: listas de condiciones (campo, operador, valor) en notación prefija (polaca), donde '&' (y) y '|' (o) se aplican a las dos expresiones que les siguen, '!' (no) a la siguiente, y los términos seguidos del nivel superior se unen con Y. La primera línea de la entrada es un dominio escrito como lista de Python; cada línea siguiente es un registro en JSON con su id. Muestra los id de los registros que cumplen el dominio, separados por espacios y en el orden de la entrada, o «ninguno». Operadores: =, !=, >, >=, <, <=, in, not in e ilike (contiene, sin distinguir mayúsculas). Un campo que falta vale None: para = False cumple, y para >, <, >=, <= e ilike no cumple.

**Código de partida** (python)

```
import sys
import ast
import json

lineas = [l for l in sys.stdin.read().splitlines() if l.strip()]
dominio = ast.literal_eval(lineas[0])
registros = [json.loads(l) for l in lineas[1:]]

TODO: muestra los id de los registros que cumplen el dominio
```

### Ejemplo: AND implícito

ENTRADA

```
[('is_company', '=', True), ('city', '=', 'Logroño')]
{"id": 1, "name": "Talleres Norte", "city": "Logroño", "is_company": true, "credit": 1200.5, "country": "ES"}
{"id": 2, "name": "Ana Ruiz", "city": "Haro", "is_company": false, "credit": 0, "country": "ES"}
{"id": 3, "name": "Clínica Sol", "city": "Logroño", "is_company": true, "credit": 0, "country": "ES"}
{"id": 4, "name": "Weinhaus GmbH", "city": "Berlín", "is_company": true, "credit": 5300, "country": "DE"}
{"id": 5, "name": "Pedro Gil", "is_company": false, "credit": 90, "country": "ES"}
```

SALIDA ESPERADA

1 3

### Ejemplo: OR en notación prefija

ENTRADA

```
['|', ('city', '=', 'Haro'), ('credit', '>', 1000)]
{"id": 1, "name": "Talleres Norte", "city": "Logroño", "is_company": true, "credit": 1200.5, "country": "ES"}
{"id": 2, "name": "Ana Ruiz", "city": "Haro", "is_company": false, "credit": 0, "country": "ES"}
{"id": 3, "name": "Clínica Sol", "city": "Logroño", "is_company": true, "credit": 0, "country": "ES"}
{"id": 4, "name": "Weinhaus GmbH", "city": "Berlín", "is_company": true, "credit": 5300, "country": "DE"}
{"id": 5, "name": "Pedro Gil", "is_company": false, "credit": 90, "country": "ES"}
```

SALIDA ESPERADA

1 2 4

### Ejemplo: Negación, in e ilike

#### ENTRADA

```
[('!', ('country', 'in', ['ES', 'PT']), ('name', 'ilike', 'GMBH'))]
{"id": 1, "name": "Talleres Norte", "city": "Logroño", "is_company": true, "credit": 1200.5, "country": "ES"}
{"id": 2, "name": "Ana Ruiz", "city": "Haro", "is_company": false, "credit": 0, "country": "ES"}
{"id": 3, "name": "Clínica Sol", "city": "Logroño", "is_company": true, "credit": 0, "country": "ES"}
{"id": 4, "name": "Weinhaus GmbH", "city": "Berlín", "is_company": true, "credit": 5300, "country": "DE"}
{"id": 5, "name": "Pedro Gil", "is_company": false, "credit": 90, "country": "ES"}
```

#### SALIDA ESPERADA

4

### Ejemplo: Campo ausente y False

#### ENTRADA

```
[('city', '=', False)]
{"id": 1, "name": "Talleres Norte", "city": "Logroño", "is_company": true, "credit": 1200.5, "country": "ES"}
{"id": 2, "name": "Ana Ruiz", "city": "Haro", "is_company": false, "credit": 0, "country": "ES"}
{"id": 3, "name": "Clínica Sol", "city": "Logroño", "is_company": true, "credit": 0, "country": "ES"}
{"id": 4, "name": "Weinhaus GmbH", "city": "Berlín", "is_company": true, "credit": 5300, "country": "DE"}
{"id": 5, "name": "Pedro Gil", "is_company": false, "credit": 90, "country": "ES"}
```

#### SALIDA ESPERADA

5

## 50. Encuentra el fallo: el recuento de ventas que se rompe

Fácil · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

El programa cuenta las unidades vendidas de cada producto a partir de las líneas de pedido y las muestra por orden alfabético. Se detiene con un error en la primera línea. Corrígelo.

#### Código de partida (python)

```
lineas = ["Portátil", "Ratón", "Ratón", "Webcam", "Ratón", "Portátil"]
ventas = {}
for producto in lineas:
 ventas[producto] += 1
for producto, unidades in sorted(ventas.items()):
 print(producto, unidades)
```

### Ejemplo: Seis líneas

#### SALIDA ESPERADA

```
Portátil 2
Ratón 3
Webcam 1
```

## 51. Un campo calculado: la amortización

Medio · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

En el módulo de equipamiento, el campo calculado `valor_actual` depende de la fecha de compra, la vida útil y el precio (@api.depends). La primera línea es la fecha de hoy (AAAA-MM-DD); cada línea siguiente, un equipo:

«nombre;fecha\_compra;vida\_util\_en\_años;precio». Con amortización lineal, años = días transcurridos / 365 y valor = precio × (1 – años / vida útil), sin bajar de 0. Muestra «nombre: valor € (estado, N años)», con el valor con 2 decimales, el estado «amortizado» si el valor es 0 y «en uso» si no, y los años con 1 decimal. Si la fecha de compra es posterior a hoy, «nombre: fecha de compra futura».

Código de partida (python)

```
import sys
from datetime import date

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]
hoy = date.fromisoformat(lineas[0])

TODO: calcula el valor actual y el estado de cada equipo
```

### Ejemplo: Varios equipos

#### ENTRADA

```
2026-10-01
Portátil Dell;2024-10-01;4;1200
Impresora HP;2019-03-15;5;450
Monitor LG;2026-04-01;6;220
Servidor;2027-01-01;5;3000
```

#### SALIDA ESPERADA

```
Portátil Dell: 600.00 € (en uso, 2.0 años)
Impresora HP: 0.00 € (amortizado, 7.6 años)
Monitor LG: 201.62 € (en uso, 0.5 años)
Servidor: fecha de compra futura
```

### Ejemplo: Recién comprado

#### ENTRADA

```
2026-10-01
Ratón;2026-10-01;3;25
```

#### SALIDA ESPERADA

```
Ratón: 25.00 € (en uso, 0.0 años)
```

## 52. Encuentra el fallo: las líneas que se comparten

Medio · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

Cada pedido debería empezar sin líneas. Ana pide un portátil y Luis dos ratones, pero el programa dice que los dos pedidos tienen 2 líneas. Corrígelo.

Código de partida (python)

```
class Pedido:
 def __init__(self, cliente, lineas=[]):
 self.cliente = cliente
 self.lineas = lineas

 def anadir(self, producto, cantidad):
 self.lineas.append((producto, cantidad))

p1 = Pedido("Ana")
p1.anadir("Portátil", 1)
p2 = Pedido("Luis")
p2.anadir("Ratón", 2)
print(p1.cliente, len(p1.lineas))
print(p2.cliente, len(p2.lineas))
```

### Ejemplo: Dos pedidos

SALIDA ESPERADA

```
Ana 1
Luis 1
```

## 53. Encuentra el fallo: los agotados que se escapan

Medio · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

El programa quita del inventario los productos sin stock. Deberían quedar el ratón y el teclado, pero también queda la webcam, que tiene 0 unidades. Corrígelo.

Código de partida (python)

```
productos = [{"Auriculares", 0}, {"Webcam", 0}, {"Ratón", 120}, {"Altavoz", 0}, {"Teclado", 45}]
for p in productos:
 if p[1] == 0:
 productos.remove(p)
print([p[0] for p in productos])
```

### Ejemplo: Inventario

SALIDA ESPERADA

```
['Ratón', 'Teclado']
```

## 54. Encuentra el fallo: el pedido confirmado que no se factura

Medio · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

El programa lee el estado de un pedido y, si está confirmado, genera la factura. Al escribir «confirmado» dice que todavía no se factura. Corrígelo.

Código de partida (python)

```
estado = input().strip()
if estado is "confirmado":
 print("Pedido confirmado: se genera la factura")
else:
 print(f"El pedido está en estado {estado}: todavía no se factura")
```

### Ejemplo: Pedido confirmado

ENTRADA

```
confirmado
```

SALIDA ESPERADA

```
Pedido confirmado: se genera la factura
```

### Ejemplo: Pedido en borrador

ENTRADA

```
borrador
```

SALIDA ESPERADA

```
El pedido está en estado borrador: todavía no se factura
```

## 55. Restricciones de un modelo

Difícil · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

En Odoo, los métodos decorados con `@api.constrains` lanzan un `ValidationError` y el registro no se guarda. Simula el modelo de mantenimientos: la primera línea es la fecha de hoy y cada línea siguiente un mantenimiento «equipo;fecha;coste;tipo». Comprueba, en este orden: el coste no es negativo; el tipo es preventivo o correctivo; la fecha no es posterior a hoy; y el equipo no tiene ya otro mantenimiento guardado esa misma fecha (una restricción única sobre equipo y fecha). Para cada error muestra «ValidationError en el registro N: mensaje» (N empieza en 1) con los mensajes «el coste no puede ser negativo», «tipo «X» no permitido», «la fecha no puede ser futura» o «EQUIPO ya tiene un mantenimiento el FECHA». Los registros con error no se guardan (no cuentan para los duplicados). Al final, «Guardados: N».

**Código de partida** (python)

```
import sys
from datetime import date

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]
hoy = date.fromisoformat(lineas[0])
TIPOS = ("preventivo", "correctivo")

TODO: valida cada mantenimiento como lo haría @api.constrains
```

### Ejemplo: Mezcla de errores

#### ENTRADA

```
2026-10-01
Portátil Dell;2026-09-10;45.5;preventivo
Impresora HP;2026-09-12;-10;correctivo
Impresora HP;2026-09-12;60;revisión
Servidor;2026-11-02;0;preventivo
Portátil Dell;2026-09-10;20;correctivo
Servidor;2026-10-01;120;correctivo
```

#### SALIDA ESPERADA

```
ValidationError en el registro 2: el coste no
puede ser negativo
ValidationError en el registro 3: tipo «revisión»
no permitido
ValidationError en el registro 4: la fecha no
puede ser futura
ValidationError en el registro 5: Portátil Dell
ya tiene un mantenimiento el 2026-09-10
Guardados: 2
```

### Ejemplo: Todo correcto

#### ENTRADA

```
2026-10-01
A;2026-01-01;0;preventivo
A;2026-01-02;10;preventivo
B;2026-01-01;10;correctivo
```

#### SALIDA ESPERADA

```
Guardados: 3
```

## 56. Secuencias de referencias

Muy difícil · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

Los módulos de Odoo numeran sus registros con secuencias (`ir.sequence`): un prefijo que puede incluir la fecha, un número con ceros a la izquierda y un reinicio del contador. La entrada tiene tres líneas de configuración —«prefijo=...», «relleno=N» y «reinicio=anual|mensual|nunca»— y después las fechas (AAAA-MM-DD) en que se crean los registros, en orden de creación. En el prefijo, `%(year)s` es el año con 4 cifras, `%(y)s` con 2, `%(month)s` el mes y `%(day)s` el día, con sus ceros. El contador empieza en 1 y se reinicia cuando cambia el año (anual) o el año y mes (mensual); si se vuelve a un periodo anterior, continúa su propio contador. Muestra «fecha → referencia» por cada registro.

#### Código de partida (python)

```
import sys

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]
patron = lineas[0].split("=", 1)[1]
relleno = int(lineas[1].split("=")[1])
reinicio = lineas[2].split("=")[1]

TODO: genera la referencia de cada registro creado en las fechas siguientes
```

#### Ejemplo: Reinicio anual

##### ENTRADA

```
prefijo=EQ/(year)s/
relleno=4
reinicio=anual
2025-11-03
2025-12-20
2026-01-02
2026-01-15
2026-02-01
```

##### SALIDA ESPERADA

```
2025-11-03 → EQ/2025/0001
2025-12-20 → EQ/2025/0002
2026-01-02 → EQ/2026/0001
2026-01-15 → EQ/2026/0002
2026-02-01 → EQ/2026/0003
```

#### Ejemplo: Reinicio mensual

##### ENTRADA

```
prefijo=MNT%(y)s%(month)s-
relleno=3
reinicio=mensual
2026-09-29
2026-09-30
2026-10-01
2026-10-01
2026-09-30
```

##### SALIDA ESPERADA

```
2026-09-29 → MNT2609-001
2026-09-30 → MNT2609-002
2026-10-01 → MNT2610-001
2026-10-01 → MNT2610-002
2026-09-30 → MNT2609-003
```

## 57. Encuentra el fallo: los céntimos que se pierden

Muy difícil · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

Las facturas se redondean al céntimo y los medios céntimos van hacia arriba (1,525 € pasa a 1,53 €), como en un tique. El programa redondea 1,525 a 1,52 y 2,675 a 2,67. Corrígelo con el módulo decimal, que ya está importado.

#### Código de partida (python)

```
from decimal import Decimal, ROUND_HALF_UP

lineas = [("Tornillos", 1.15), ("Arandelas", 1.525), ("Tuercas", 2.675)]
for nombre, importe in lineas:
 print(nombre, round(importe, 2))
```

#### Ejemplo: Tres líneas

##### SALIDA ESPERADA

```
Tornillos 1.15
Arandelas 1.53
Tuercas 2.68
```

## 58. Totales de una factura

Fácil · Ejercicios de Sistemas de Gestión Empresarial · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge)

Cada línea de la entrada es una línea de factura con el formato «producto;cantidad;precio;iva» (iva en %: 21, 10 o 4). Muestra la base imponible, la cuota de IVA y el total con 2 decimales, con el formato «Base: 100.00», «IVA: 21.00» y «Total: 121.00».

**Código de partida** (python)

```
import sys

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

TODO
```

### Ejemplo: Factura con dos tipos de IVA

ENTRADA

```
Licencia ERP;3;300;21
Formación (hora);10;60;21
Libro de consulta;2;25;4
```

SALIDA ESPERADA

```
Base: 1550.00
IVA: 317.00
Total: 1867.00
```

### Ejemplo: Una sola línea

ENTRADA

```
Soporte anual;1;900;21
```

SALIDA ESPERADA

```
Base: 900.00
IVA: 189.00
Total: 1089.00
```

## 59. Stock tras los movimientos

Medio · Ejercicios de Sistemas de Gestión Empresarial · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge)

La entrada tiene dos partes separadas por una línea «---». Primero, el stock inicial: «producto cantidad». Después, los movimientos: «entrada producto cantidad» o «salida producto cantidad». Una salida mayor que el stock disponible no se hace y se muestra «ERROR: stock insuficiente de producto». Al final muestra el stock de cada producto ordenado alfabéticamente con el formato «producto: cantidad». Un producto que llega por una entrada sin estar en el stock inicial empieza en 0.

**Código de partida** (python)

```
import sys

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

TODO
```

### Ejemplo: Entradas, salidas y un error

ENTRADA

```
tornillos 500
tuercas 300
arandelas 50

salida tornillos 120
entrada arandelas 200
salida tuercas 400
salida tuercas 100
entrada bisagras 40
```

SALIDA ESPERADA

```
ERROR: stock insuficiente de tuercas
arandelas: 250
bisagras: 40
tornillos: 380
tuercas: 200
```

### Ejemplo: Sin movimientos

#### ENTRADA

```
lector 3
portatil 5

```

#### SALIDA ESPERADA

```
lector: 3
portatil: 5
```

## 60. Validar NIF y NIE de clientes

Medio · Ejercicios de Sistemas de Gestión Empresarial · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge)

Antes de dar de alta clientes en el ERP hay que validar su documento. Cada línea es un NIF (8 cifras y una letra) o un NIE (X, Y o Z, 7 cifras y una letra), en mayúsculas o minúsculas. Muestra «DOCUMENTO válido» o «DOCUMENTO no válido», con el documento en mayúsculas. La letra de control es la posición número % 23 de la cadena TRWAGMYFPDXBNJZSQVHLCKE; en los NIE la X, Y o Z se sustituye por 0, 1 o 2 antes de calcularla.

Código de partida (python)

```
import sys

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

TODO
```

### Ejemplo: NIF y NIE variados

#### ENTRADA

```
12345678Z
48123456A
X1234567L
Y7654321A
1234567Z
30567891v
```

#### SALIDA ESPERADA

```
12345678Z válido
48123456A no válido
X1234567L válido
Y7654321A no válido
1234567Z no válido
30567891V válido
```

## 61. Informe de antigüedad de deuda

Difícil · Ejercicios de Sistemas de Gestión Empresarial · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge)

La primera línea es la fecha del informe (AAAA-MM-DD). Las siguientes son facturas pendientes: «factura;cliente;vencimiento;pendiente». Clasifica el importe pendiente por días de retraso (fecha del informe – vencimiento) en los tramos «No vencido» (0 días o menos), «1-30», «31-60», «61-90» y «Más de 90», y muestra todos los tramos en ese orden con el formato «tramo: importe» (2 decimales). Termina con «Mayor deudor vencido: cliente (importe)», el cliente con más importe vencido (retraso > 0), o «Mayor deudor vencido: ninguno».

Código de partida (python)

```
import sys

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

TODO
```

### Ejemplo: Facturas en todos los tramos

#### ENTRADA

```
2025-06-30
F2025-010;Talleres Ebro;2025-07-15;1200
F2025-004;Calzados Cidacos;2025-06-10;850.5
F2025-002;Talleres Ebro;2025-05-20;300
F2025-001;Hotel Los Cameros;2025-04-15;2100
F2024-090;Calzados Cidacos;2025-01-31;640
F2025-011;Muebles Oja;2025-06-30;99.9
```

#### SALIDA ESPERADA

```
No vencido: 1299.90
1-30: 850.50
31-60: 300.00
61-90: 2100.00
Más de 90: 640.00
Mayor deudor vencido: Hotel Los Cameros (2100.00)
```

### Ejemplo: Nada vencido

#### ENTRADA

```
2025-03-01
F1;Academia Rioja;2025-03-15;400
F2;Farmacia Central;2025-03-01;150
```

#### SALIDA ESPERADA

```
No vencido: 550.00
1-30: 0.00
31-60: 0.00
61-90: 0.00
Más de 90: 0.00
Mayor deudor vencido: ninguno
```

## Ejercicios largos

### 62. Corrector de exámenes tipo test

Difícil · 60 minutos · [apuntesdam.com/ejercicios/python/corrector-de-examenes-tipo-test](https://apuntesdam.com/ejercicios/python/corrector-de-examenes-tipo-test)

Una profesora pasa sus exámenes tipo test a un formulario y obtiene un fichero de texto con las respuestas de cada alumno. Quiere un programa que los corrija, ordene las notas y le diga qué preguntas han salido peor, para repasarlas en clase.

Cada pregunta tiene cuatro opciones (A, B, C y D) y el alumno puede dejarla en blanco. Para que no compense contestar al azar, cada fallo resta un tercio de acierto: tres fallos anulan una pregunta bien contestada. Las preguntas en blanco no suman ni restan, y la puntuación de un examen nunca puede ser negativa.

Algunas líneas del fichero llegan mal (sin el punto y coma, con respuestas de más o de menos o con letras que no existen): el programa debe avisar de cada una y corregir las demás.

#### Requisitos

- La primera línea es la plantilla: `CLAVE` seguido de las respuestas correctas, por ejemplo `CLAVE ABCDA`. El número de preguntas  $N$  es la longitud de la clave.
- Cada línea siguiente es un examen: `nombre;respuestas`, con una letra (A, B, C o D) o un guion (en blanco) por pregunta. Se ignoran las líneas vacías y los espacios al principio y al final de cada parte.
- Si una línea no tiene `;`, escribe `Aviso: la línea L no tiene el formato nombre;respuestas`, donde  $L$  es su número de línea (la de la clave es la 1). Si las respuestas no tienen exactamente  $N$  caracteres o alguno no es A, B, C, D ni guion, escribe `Aviso: las respuestas de nombre no son válidas y no se corrigen`. En los dos casos ese examen no cuenta.
- Puntuación de un examen:  $\text{aciertos} - \text{fallos} \div 3$ , con un mínimo de 0. La nota es esa puntuación  $\times 10 \div N$ .
- Escribe `Corrección de N preguntas` y una línea por alumno, de mayor a menor nota (a igual nota, por orden alfabético): la posición en 2 caracteres alineada a la derecha, un punto y un espacio, el nombre en 22 caracteres alineado a la izquierda y la nota con dos decimales y coma, en 6 caracteres a la derecha.
- Después: `Media: X,XX`, `Aprobados: A de T` (aprueba quien tiene un 5 o más), `Pregunta más fallada: P (F fallos)` (la de más respuestas incorrectas; en blanco no cuenta como fallo; si hay empate, la de número más bajo; «fallo» en singular si es 1; `ninguna` si nadie ha fallado nada) y `Sin aciertos: ...` con los números de las preguntas que nadie ha acertado separados por coma y espacio, o `ninguna`.

- Si no hay ningún examen válido, después de los avisos escribe solo **No hay exámenes que corregir.**

## Formato de la entrada

- Línea 1: **CLAVE** respuestas . Resto de líneas: **nombre;respuestas** (puede haber líneas vacías o mal escritas).

### Ejemplo: Cinco exámenes de diez preguntas

#### ENTRADA

```
CLAVE ABCDABCDAB
Lucía Gómez;ABCDABCDAB
Mario Vega;ABCDABCD--
Nerea Paz;ABCAABCDDBB
Olga Ruiz;DCBADCBADC
Pedro Sanz;-----ABCDAB
```

#### SALIDA ESPERADA

```
Corrección de 10 preguntas
1. Lucía Gómez 10,00
2. Mario Vega 8,00
3. Nerea Paz 7,33
4. Olga Ruiz 0,00
5. Pedro Sanz 0,00
Media: 5,07
Aprobados: 3 de 5
Pregunta más fallada: 9 (3 fallos)
Sin aciertos: ninguna
```

### Ejemplo: Con líneas mal escritas

#### ENTRADA

```
CLAVE CCBA
Ana Ruiz;CCBB
Bruno Díaz;CCB
Carla Mena;CCXA
sin punto y coma
Diego Sola;CA-B
Elena Toro;C-BC
```

#### SALIDA ESPERADA

```
Aviso: las respuestas de Bruno Díaz no son
válidas y no se corrigen
Aviso: las respuestas de Carla Mena no son
válidas y no se corrigen
Aviso: la línea 6 no tiene el formato
nombre;respuestas
Corrección de 4 preguntas
1. Ana Ruiz 6,67
2. Elena Toro 4,17
3. Diego Sola 0,83
Media: 3,89
Aprobados: 1 de 3
Pregunta más fallada: 4 (3 fallos)
Sin aciertos: 4
```

## Código de partida (python)

```
import sys

def formato(numero):
 """Dos decimales con coma, como se escriben las notas en España."""
 return f"{numero:.2f}".replace(".", ",")

def main():
 lineas = sys.stdin.read().split("\n")
 clave = lineas[0].split()[1]
 n = len(clave)
 alumnos = [] # (nombre, nota) de cada examen válido
 fallos = [0] * n # fallos de cada pregunta
 aciertos_pregunta = [0] * n # aciertos de cada pregunta

 for num, linea in enumerate(lineas[1:], start=2):
 # TODO: valida la línea, corrige el examen y guarda el nombre y la nota
 pass

 # TODO: clasificación, media, aprobados y estadísticas de las preguntas

main()
```

### 63. Informe de ventas a partir de un CSV

Difícil · 70 minutos · [apuntesdam.com/ejercicios/python/informe-de-ventas-csv](https://apuntesdam.com/ejercicios/python/informe-de-ventas-csv)

Una papelería exporta las ventas de su caja registradora a un fichero CSV, con una línea por venta: fecha, producto, categoría, unidades y precio unitario. El dueño quiere un informe rápido sin abrir una hoja de cálculo.

El fichero lo genera una caja antigua y a veces trae líneas rotas: columnas que faltan, unidades escritas con letras o negativas. Esas líneas no deben estropear el informe: se descartan y se dice cuáles eran.

Para que los totales cuadren al céntimo, el informe se calcula en céntimos (números enteros) y solo se convierte a euros al escribirlo. Sumar precios con decimales como 0.1 + 0.2 en coma flotante da errores pequeños que, con muchas ventas, acaban notándose.

#### Requisitos

- La primera línea es la cabecera `fecha,producto,categoria,unidades,precio` y no se procesa. Las líneas vacías se ignoran sin más.
- Una línea es válida si tiene exactamente 5 campos separados por comas, las unidades son un entero mayor que 0 y el precio es un número no negativo. El importe de la línea es unidades × precio. Las fechas tienen el formato `AAAA-MM-DD`.
- Escribe `Ventas totales: X €` y, bajo `Por categoría:`, una línea por categoría de mayor a menor importe (a igualdad, alfabético): dos espacios, el nombre en 16 caracteres a la izquierda, el importe en 12 caracteres a la derecha, dos espacios y el porcentaje sobre el total con un decimal en 5 caracteres a la derecha, seguido de `%`.
- `Producto más vendido: producto (N unidades)`: el de más unidades vendidas en total (a igualdad, el primero alfabéticamente; «unidad» en singular si es 1). `Mejor día: fecha (X €)`: el día con más importe (a igualdad, el más antiguo).
- Bajo `Por mes:`, una línea por mes en orden cronológico: dos espacios, el mes (`AAAA-MM`), dos espacios, el importe en 12 caracteres a la derecha, un espacio y una almohadilla `#` por cada 50 € completos (sin espacios al final de la línea).
- Por último, `Líneas descartadas: N` y, si hay alguna, sus números de línea entre paréntesis separados por coma y espacio (la cabecera es la línea 1).
- Los importes se escriben con punto de millar, coma decimal, dos decimales y el símbolo `€` (por ejemplo `1.234,50 €`). Si no hay ninguna venta válida, escribe solo `No hay ventas válidas.`

#### Formato de la entrada

- Un fichero CSV: la cabecera y una venta por línea, con los campos separados por comas y el precio con punto decimal.

#### Ejemplo: Una semana de ventas

##### ENTRADA

```
fecha,producto,categoria,unidades,precio
2026-08-28,Cuaderno A4,Papelería,10,2.50
2026-08-28,Ratón inalámbrico,Informática,2,18.90
2026-08-31,Bolígrafo azul,Papelería,25,0.80
2026-09-01,Cuaderno A4,Papelería,15,2.50
2026-09-01,Memoria USB 64 GB,Informática,4,12.00
2026-09-02,Mochila,Complementos,3,34.95
2026-09-02,Bolígrafo azul,Papelería,40,0.80
2026-09-03,Ratón inalámbrico,Informática,1,18.90
```

##### SALIDA ESPERADA

```
Ventas totales: 324,05 €
Por categoría:
 Papelería 114,50 € 35,3 %
 Complementos 104,85 € 32,4 %
 Informática 104,70 € 32,3 %
Producto más vendido: Bolígrafo azul (65
unidades)
Mejor día: 2026-09-02 (136,85 €)
Por mes:
 2026-08 82,80 € #
 2026-09 241,25 € #####
Líneas descartadas: 0
```

### Ejemplo: Con líneas rotas

#### ENTRADA

```
fecha,producto,categoria,unidades,precio
2026-09-10,Cuaderno A4,Papelería,4,2.50
2026-09-10,Grapadora,Papelería,dos,6.40
2026-09-11,Auriculares,Informática,1,29.99
2026-09-11,Funda portátil,Complementos,1
2026-09-12,Bolígrafo azul,Papelería,-3,0.80
2026-09-12,Bolígrafo azul,Papelería,12,0.80

2026-09-13,Auriculares,Informática,1,29.99
```

#### SALIDA ESPERADA

```
Ventas totales: 79,58 €
Por categoría:
 Informática 59,98 € 75,4 %
 Papelería 19,60 € 24,6 %
Producto más vendido: Bolígrafo azul (12
unidades)
Mejor día: 2026-09-11 (29,99 €)
Por mes:
 2026-09 79,58 € #
Líneas descartadas: 3 (3, 5, 6)
```

#### Código de partida (python)

```
import sys

def euros(centimos):
 """1234.5 € → «1.234,50 €»: puntos de millar y coma decimal."""
 texto = f"{centimos / 100:,.2f}"
 return texto.replace(",", "X").replace(".", ",").replace("X", ".") + " €"

def leer_venta(linea):
 """Devuelve (fecha, producto, categoría, unidades, importe en céntimos) o None si la línea no es
 válida."""
 # TODO
 return None

def main():
 lineas = sys.stdin.read().split("\n")
 # TODO: acumula por categoría, producto, día y mes, y escribe el informe

main()
```

## 64. Planificador de procesos: FCFS, SJF, SRTF y Round Robin

Muy difícil · 100 minutos · [apuntesdam.com/ejercicios/python/planificador-de-procesos](https://apuntesdam.com/ejercicios/python/planificador-de-procesos)

Un ordenador ejecuta muchos más procesos que núcleos tiene, así que el planificador del sistema operativo decide en cada momento cuál usa la CPU. La decisión cambia mucho el tiempo que espera cada programa: es uno de los temas que más cae en los exámenes de Sistemas, siempre con el mismo tipo de ejercicio: dibujar el diagrama de Gantt y calcular tiempos.

En este ejercicio vas a escribir ese planificador. Los procesos llegan en distintos instantes con una ráfaga de CPU (lo que necesitan ejecutar), y el programa simula cuatro algoritmos clásicos: FCFS (por orden de llegada), SJF (el de ráfaga más corta entre los que esperan, sin interrumpirlo), SRTF (el de menor tiempo restante, interrumpiendo si llega uno más corto) y Round Robin (turnos de un quantum fijo).

El resultado de cada proceso se mide con dos tiempos: el de retorno (desde que llega hasta que termina) y el de espera (el retorno menos lo que de verdad ha usado la CPU). En la web tienes también el simulador visual de planificación para comprobar tus resultados.

### Requisitos

- La primera línea es el algoritmo: **FCFS**, **SJF**, **SRTF** o **RR q** (con q, el quantum, un entero mayor que 0). Si no es válida, escribe **Algoritmo no válido** y nada más. Cada línea siguiente es un proceso: **nombre llegada ráfaga**, enteros con la ráfaga mayor que 0. Una línea con otro formato o un nombre repetido escribe **Aviso: línea N ignorada**. Si no queda ningún proceso, **No hay procesos**.

- Desempates: FCFS y Round Robin atienden por orden de llegada y, a igual llegada, por el orden de la entrada. SJF elige la menor ráfaga y SRTF el menor tiempo restante; si empatan, el que llegó antes y luego el de la entrada. En Round Robin, cuando a un proceso se le acaba el quantum, los procesos que han llegado hasta ese instante entran en la cola antes que él.
- SRTF reevalúa cada vez que llega un proceso: si el recién llegado necesita menos que lo que le queda al que está en la CPU, lo expulsa. Si nadie está listo, la CPU queda ociosa hasta la siguiente llegada.
- Escribe **Algoritmo:** ... con su nombre completo (el de la tabla; en Round Robin, **Round Robin con quantum q**) y **Diagrama:** | P1 0-3 | P2 3-7 | ... | : cada tramo con el proceso, el inicio y el fin, -- para la CPU ociosa, y los tramos seguidos del mismo proceso unidos en uno solo.
- Una tabla con la cabecera **Proceso**, **Llegada**, **Ráfaga**, **Fin**, **Retorno**, **Espera** y una fila por proceso en el orden de la entrada: el nombre en 8 caracteres a la izquierda y los números alineados a la derecha en 8, 8, 6, 9 y 8 caracteres.
- Para terminar, **Media de retorno: X** • **Media de espera: Y** y **Uso de la CPU: Z %** (tiempo ocupado entre el instante final), con dos decimales y coma decimal.

### Formato de la entrada

- Línea 1: **FCFS**, **SJF**, **SRTF** o **RR q**. Resto: **nombre llegada ráfaga**.

### Nombre de cada algoritmo en la salida

Entrada	Algoritmo
FCFS	FCFS (primero en llegar, primero en ser servido)
SJF	SJF (el más corto primero, sin expulsión)
SRTF	SRTF (menor tiempo restante, con expulsión)
RR q	Round Robin con quantum q

### Ejemplo: Round Robin con quantum 2

#### ENTRADA

```
RR 2
P1 0 5
P2 1 3
P3 2 8
P4 3 6
```

#### SALIDA ESPERADA

Algoritmo: Round Robin con quantum 2  
Diagrama: | P1 0-2 | P2 2-4 | P3 4-6 | P1 6-8 | P4 8-10 | P2 10-11 | P3 11-13 | P1 13-14 | P4 14-16 | P3 16-18 | P4 18-20 | P3 20-22 |  
Proceso Llegada Ráfaga Fin Retorno Espera  
P1 0 5 14 14 9  
P2 1 3 11 10 7  
P3 2 8 22 20 12  
P4 3 6 20 17 11  
Media de retorno: 15,25 · Media de espera: 9,75  
Uso de la CPU: 100,00 %

### Ejemplo: SRTF con CPU ociosa

#### ENTRADA

```
SRTF
A 0 7
B 2 4
C 4 1
D 5 4
E 30 2
F x 3
A 1 1
```

#### SALIDA ESPERADA

Aviso: línea 7 ignorada  
Aviso: línea 8 ignorada  
Algoritmo: SRTF (menor tiempo restante, con expulsión)  
Diagrama: | A 0-2 | B 2-4 | C 4-5 | B 5-7 | D 7-11 | A 11-16 | -- 16-30 | E 30-32 |  
Proceso Llegada Ráfaga Fin Retorno Espera  
A 0 7 16 16 9  
B 2 4 7 5 1  
C 4 1 5 1 0  
D 5 4 11 6 2  
E 30 2 32 2 0  
Media de retorno: 6,00 · Media de espera: 2,40  
Uso de la CPU: 56,25 %

### Código de partida (python)

```
import sys

NOMBRES = {"FCFS": "FCFS (primero en llegar, primero en ser servido)", "SJF": "SJF (el más corto primero, sin expulsión)", "SRTF": "SRTF (menor tiempo restante, con expulsión)", "RR": "Round Robin"}

def coma(x):
 return f"{x:.2f}".replace(".", ",")

def simular(algoritmo, quantum, procesos):
 """Devuelve el diagrama de Gantt como lista de [nombre, inicio, fin] y el instante en que termina cada proceso."""
 # TODO
 return [], {}

def main():
 lineas = sys.stdin.read().split("\n")
 # TODO: lee el algoritmo y los procesos, simula y escribe el diagrama, la tabla y las medias

main()
```

## 65. Plan de direccionamiento con subredes VLSM

Difícil · 70 minutos · [apuntesdam.com/ejercicios/python/subredes-vlsm](https://apuntesdam.com/ejercicios/python/subredes-vlsm)

Una empresa tiene asignada la red 192.168.10.0/24 y quiere dividirla entre sus departamentos. Con subredes del mismo tamaño se desperdician direcciones (el enlace entre dos routers no necesita 62 direcciones), así que se usa VLSM: cada subred recibe la máscara más ajustada a los equipos que va a tener.

El método que se enseña (y el que sigue este programa) es asignar primero las subredes más grandes: así cada bloque empieza en una dirección múltiplo de su tamaño y las subredes nunca se solapan. De cada subred hay que dar la dirección de red, la máscara, la primera y la última dirección que se pueden asignar a equipos y la de difusión.

Hacerlo a mano es lento y es fácil equivocarse con la última dirección o la de difusión. Una dirección IP no es más que un número de 32 bits escrito en cuatro bytes: con ese número, las cuentas son sumas y operaciones de bits.

### Requisitos

- La primera línea es la red, `IP/prefijo`, con el prefijo entre 8 y 30. Si no tiene ese formato o la IP no es válida, escribe `Red no válida: texto`. Si la IP no es la dirección de red de ese prefijo (tiene bits de host a 1), escribe `IP no es una dirección de red (sería RED/prefijo)`. En los dos casos no se escribe nada más.
- Cada línea siguiente es una subred: un nombre (puede tener espacios) y el número de hosts que necesita, un entero mayor que 0. Una línea con otro formato escribe `Aviso: línea N ignorada`.
- Las subredes se asignan de mayor a menor número de hosts (si empatan, por orden de la entrada), una detrás de otra desde el principio de la red. Cada una recibe el prefijo más largo que tenga sitio para sus hosts más la dirección de red y la de difusión, y como mucho /30 (un enlace de 2 hosts es un /30).
- Para cada subred: `Nombre (H hosts) → RED/p`, y debajo, con dos espacios delante, `máscara M · hosts útiles U` y `primera A · última B · difusión D`. Si no cabe en lo que queda de la red, `Nombre (H hosts) → no cabe: necesita un /p y quedan N direcciones` y se sigue con la siguiente.
- Antes de las subredes, `Red: RED/prefijo (T direcciones)`. Al final, `Sin asignar: N de T direcciones, desde IP` con la primera dirección libre, o `Sin asignar: 0 de T direcciones` si no queda ninguna.

### Formato de la entrada

- Línea 1: la red, `IP/prefijo`. Resto: una subred por línea, `nombre hosts`.

### Tamaño de cada prefijo

Prefijo	Máscara	Direcciones	Hosts útiles
/30	255.255.255.252	4	2
/29	255.255.255.248	8	6
/28	255.255.255.240	16	14
/27	255.255.255.224	32	30
/26	255.255.255.192	64	62
/25	255.255.255.128	128	126
/24	255.255.255.0	256	254

### Ejemplo: Una oficina con cinco subredes

#### ENTRADA

192.168.10.0/24  
Ventas 50  
Administración 20  
Almacén 10  
Enlace router 2  
Dirección 20

#### SALIDA ESPERADA

Red: 192.168.10.0/24 (256 direcciones)  
Ventas (50 hosts) → 192.168.10.0/26  
máscara 255.255.255.192 · hosts útiles 62  
primera 192.168.10.1 · última 192.168.10.62 ·  
difusión 192.168.10.63  
Administración (20 hosts) → 192.168.10.64/27  
máscara 255.255.255.224 · hosts útiles 30  
primera 192.168.10.65 · última 192.168.10.94 ·  
difusión 192.168.10.95  
Dirección (20 hosts) → 192.168.10.96/27  
máscara 255.255.255.224 · hosts útiles 30  
primera 192.168.10.97 · última 192.168.10.126 ·  
difusión 192.168.10.127  
Almacén (10 hosts) → 192.168.10.128/28  
máscara 255.255.255.240 · hosts útiles 14  
primera 192.168.10.129 · última 192.168.10.142 ·  
difusión 192.168.10.143  
Enlace router (2 hosts) → 192.168.10.144/30  
máscara 255.255.255.252 · hosts útiles 2  
primera 192.168.10.145 · última 192.168.10.146 ·  
difusión 192.168.10.147  
Sin asignar: 108 de 256 direcciones, desde  
192.168.10.148

### Ejemplo: Una IP que no es de red

#### ENTRADA

10.0.0.5/24  
A 10

#### SALIDA ESPERADA

10.0.0.5 no es una dirección de red (sería  
10.0.0.0/24)

```
import sys

def a_numero(ip):
 """«192.168.10.0» → entero de 32 bits; None si no es una IPv4."""
 # TODO
 return None

def a_ip(n):
 # TODO
 return ""

def main():
 lineas = sys.stdin.read().split("\n")
 # TODO: valida la red, lee las subredes y asígnalas con VLSM

main()
```

## 66. Inventario de almacén con valoración FIFO y reposición

Muy difícil · 90 minutos · [apuntesdam.com/ejercicios/python/inventario-con-valoracion-fifo](https://apuntesdam.com/ejercicios/python/inventario-con-valoracion-fifo)

En un ERP como Odoo, el módulo de inventario no solo cuenta unidades: también valora el almacén. Si un tornillo se ha comprado una vez a 0,05 € y otra a 0,06 €, ¿cuánto valen los que quedan y cuánto costaron los vendidos? La respuesta depende del método de valoración, y uno de los más usados es FIFO (primero en entrar, primero en salir): cada salida se lleva primero las unidades del lote más antiguo.

Los movimientos del almacén llegan de varios terminales y no siempre en orden, así que hay que aplicarlos por fecha. Una salida de más unidades de las que hay se rechaza (es un error de la persona que la registró) y no cambia nada.

Además, el ERP propone pedidos: cada producto tiene un stock mínimo y uno máximo, y cuando queda por debajo del mínimo hay que pedir lo necesario para volver al máximo.

### Requisitos

- Cada línea es `PRODUCTO código;nombre;mínimo;máximo`, `ENTRADA fecha;código;cantidad;coste unitario` o `SALIDA fecha;código;cantidad`, con las fechas como `AAAA-MM-DD`, cantidades enteras mayores que 0 y el coste con punto decimal. Los productos se declaran antes de sus movimientos.
- Una línea con otro formato, un número no válido, un producto que no existe (o repetido) o un mínimo mayor que el máximo escribe `Aviso: línea N ignorada` al leerla.
- Los movimientos se aplican por fecha y, si coinciden, en el orden de la entrada. Una entrada crea un lote con su cantidad y su coste. Una salida con más unidades que el stock escribe `Aviso: salida de C código el fecha rechazada (stock S)` y no se aplica; si hay stock, consume unidades del lote más antiguo, y cuando se acaba, del siguiente, sumando su coste al coste de lo vendido.
- Escribe `Inventario:` y, por cada producto en el orden en que se declararon, `código nombre: stock S · valor V · coste de lo vendido C`, donde el valor es la suma de las unidades que quedan por el coste de su lote.
- Después, `Reponer:` y una línea por producto con stock menor que su mínimo, `código nombre: pedir P (stock S, mínimo m, máximo M)` con  $P = \text{máximo} - \text{stock}$ ; o `Reponer: nada`. Para terminar, `Valor del almacén: X`.
- Los importes se calculan con decimales exactos, se redondean al céntimo solo al escribirlos (las mitades suben) y se escriben con punto de millar, coma decimal y €. Si no hay ningún producto, `No hay productos`.

### Formato de la entrada

- Una línea por producto o movimiento: `PRODUCTO ...`, `ENTRADA ...` o `SALIDA ...`.

### Ejemplo: Tornillos y tuercas

#### ENTRADA

```
PRODUCTO T01;Tornillo M6;100;500
PRODUCTO T02;Tuerca M6;100;400
ENTRADA 2026-09-03;T01;200;0.06
ENTRADA 2026-09-01;T01;300;0.05
SALIDA 2026-09-05;T01;350
ENTRADA 2026-09-02;T02;150;0.035
SALIDA 2026-09-06;T02;80
SALIDA 2026-09-07;T02;100
```

#### SALIDA ESPERADA

```
Aviso: salida de 100 T02 el 2026-09-07 rechazada
(stock 70)
Inventario:
 T01 Tornillo M6: stock 150 · valor 9,00 € ·
coste de lo vendido 18,00 €
 T02 Tuerca M6: stock 70 · valor 2,45 € · coste
de lo vendido 2,80 €
Reponer:
 T02 Tuerca M6: pedir 330 (stock 70, mínimo 100,
máximo 400)
Valor del almacén: 11,45 €
```

### Ejemplo: Líneas incorrectas

#### ENTRADA

```
PRODUCTO A;Arandela;10;50
PRODUCTO A;Repetido;1;2
PRODUCTO B;Bisagra;20;10
ENTRADA 2026-10-01;A;40;0.10
SALIDA 2026-10-01;A;15
SALIDA 2026-10-01;A;30
SALIDA 2026-10-02;Z;1
ENTRADA 2026-10-03;A;-5;0.10
AJUSTE 2026-10-03;A;3
```

#### SALIDA ESPERADA

```
Aviso: línea 2 ignorada
Aviso: línea 3 ignorada
Aviso: línea 7 ignorada
Aviso: línea 8 ignorada
Aviso: línea 9 ignorada
Aviso: salida de 30 A el 2026-10-01 rechazada
(stock 25)
Inventario:
 A Arandela: stock 25 · valor 2,50 € · coste de
lo vendido 1,50 €
Reponer: nada
Valor del almacén: 2,50 €
```

#### Código de partida (python)

```
import sys
from decimal import Decimal, ROUND_HALF_UP

def euros(x):
 """Decimal → «1.234,56 €» redondeado al céntimo."""
 x = x.quantize(Decimal("0.01"), rounding=ROUND_HALF_UP)
 return f"{x:,.2f}".replace(",", "X").replace(".", ",").replace("X", ".") + " €"

def main():
 productos = {} # código → datos del producto
 movimientos = [] # (fecha, orden de la entrada, tipo, código, cantidad, coste)
 # TODO: lee productos y movimientos, aplícalos por fecha con FIFO y escribe el inventario

main()
```

## 67. Resumen de pedidos en XML con ElementTree

Difícil · 70 minutos · [apuntesdam.com/ejercicios/python/resumen-de-pedidos-xml](https://apuntesdam.com/ejercicios/python/resumen-de-pedidos-xml)

La tienda online de una empresa exporta cada noche sus pedidos en XML para el sistema de gestión. El departamento comercial quiere un resumen del día, también en XML, para cargarlo en su cuadro de mando: cuántos pedidos hay en cada estado, cuánto ha comprado cada cliente y cuál es el producto que más se ha vendido.

El fichero lo genera un sistema antiguo y a veces trae pedidos incompletos: sin cliente, con un estado que no existe o con cantidades y precios mal escritos. Esos pedidos se descartan con un aviso que explique el motivo, y el resumen se hace con el resto.

Para leer y escribir XML no se trata como texto con `split`: se usa un analizador. Python trae `xml.etree.ElementTree`, que convierte el documento en un árbol de elementos con sus atributos, y que también sabe construir un árbol nuevo y escribirlo con la sangría correcta y los caracteres especiales escapados.

### Requisitos

- La entrada es un documento XML con la raíz `<pedidos fecha="...">` y elementos `<pedido id cliente estado>` que contienen elementos `<línea ref cantidad precio/>`. Si el XML no está bien formado, escribe `XML mal formado (línea N)` con la línea del error; si la raíz no es `pedidos`, El elemento raíz debe ser `<pedidos>`. En los dos casos no se escribe nada más.
- Un pedido se ignora, con el aviso `Aviso: pedido ID ignorado: motivo` (o `?` si no tiene id), cuando: le falta id, cliente o estado (`falta el atributo X`, comprobados en ese orden); su estado no es pendiente, enviado, entregado ni cancelado (`estado desconocido X`); no tiene líneas (`no tiene líneas`); una línea no tiene ref o su cantidad no es un entero mayor que 0 (`línea REF con una cantidad no válida`, con `?` si falta la ref); o un precio no es un número (`línea REF con un precio no válido`).
- Los pedidos válidos cuentan en su estado. Los cancelados no suman importe, ni unidades, ni cuentan como pedidos del cliente. El importe de un pedido es la suma de `cantidad × precio` de sus líneas, con decimales exactos.
- La salida es un XML `<resumen fecha pedidos importe>` (la fecha de la raíz, el número de pedidos válidos y el importe total con dos decimales y punto), con dos espacios de sangría por nivel, que contiene: `<estados>` con un `<estado nombre pedidos/>` por estado en el orden pendiente, enviado, entregado, cancelado; `<clientes>` con un `<cliente nombre pedidos importe/>` por cliente, de mayor a menor importe (a igualdad, por nombre); y `<masVendido ref unidades/>` con la referencia de más unidades (a igualdad, la menor alfabéticamente), solo si se ha vendido algo.
- Los avisos se escriben antes del XML.

### Formato de la entrada

- Un documento XML de pedidos.

### Ejemplo: Los pedidos de un día

#### ENTRADA

```
<?xml version="1.0" encoding="UTF-8"?>
<pedidos fecha="2026-09-30">
 <pedido id="P-101" cliente="Ana Gil"
estado="enviado">
 <linea ref="CAB-HDMI" cantidad="2"
precio="7.95"/>
 <linea ref="RATON-01" cantidad="1"
precio="24.99"/>
 </pedido>
 <pedido id="P-102" cliente="Muebles Ruiz &
Hijos" estado="entregado">
 <linea ref="SILLA-07" cantidad="4"
precio="89.90"/>
 </pedido>
 <pedido id="P-103" cliente="Ana Gil"
estado="cancelado">
 <linea ref="SILLA-07" cantidad="10"
precio="89.90"/>
 </pedido>
 <pedido id="P-104" cliente="Luis Mora"
estado="enviado">
 <linea ref="CAB-HDMI" cantidad="3"
precio="7.95"/>
 <linea ref="TECLADO" cantidad="1"
precio="abc"/>
 </pedido>
 <pedido id="P-105" cliente="Ana Gil"
estado="pendiente">
 <linea ref="CAB-HDMI" cantidad="3"
precio="7.95"/>
 </pedido>
</pedidos>
```

#### SALIDA ESPERADA

```
Aviso: pedido P-104 ignorado: línea TECLADO con
un precio no válido
<resumen fecha="2026-09-30" pedidos="4"
importe="424.34">
 <estados>
 <estado nombre="pendiente" pedidos="1" />
 <estado nombre="enviado" pedidos="1" />
 <estado nombre="entregado" pedidos="1" />
 <estado nombre="cancelado" pedidos="1" />
 </estados>
 <clientes>
 <cliente nombre="Muebles Ruiz & Hijos"
pedidos="1" importe="359.60" />
 <cliente nombre="Ana Gil" pedidos="2"
importe="64.74" />
 </clientes>
 <masVendido ref="CAB-HDMI" unidades="5" />
</resumen>
```

### Ejemplo: Pedidos con errores

#### ENTRADA

```
<pedidos fecha="2026-10-01">
 <pedido cliente="Sin id" estado="enviado">
<linea ref="A" cantidad="1" precio="1"/></pedido>
 <pedido id="P-2" estado="enviado"><linea
ref="A" cantidad="1" precio="1"/></pedido>
 <pedido id="P-3" cliente="Eva"
estado="perdido"><linea ref="A" cantidad="1"
precio="1"/></pedido>
 <pedido id="P-4" cliente="Eva"
estado="enviado"/>
 <pedido id="P-5" cliente="Eva"
estado="enviado"><linea cantidad="1" precio="1"/>
</pedido>
 <pedido id="P-6" cliente="Eva"
estado="cancelado"><linea ref="B" cantidad="2"
precio="5"/></pedido>
</pedidos>
```

#### SALIDA ESPERADA

```
Aviso: pedido ? ignorado: falta el atributo id
Aviso: pedido P-2 ignorado: falta el atributo
cliente
Aviso: pedido P-3 ignorado: estado desconocido
perdido
Aviso: pedido P-4 ignorado: no tiene líneas
Aviso: pedido P-5 ignorado: línea ? con una
cantidad no válida
<resumen fecha="2026-10-01" pedidos="1"
importe="0.00">
 <estados>
 <estado nombre="pendiente" pedidos="0" />
 <estado nombre="enviado" pedidos="0" />
 <estado nombre="entregado" pedidos="0" />
 <estado nombre="cancelado" pedidos="1" />
 </estados>
 <clientes />
</resumen>
```

```
import sys
import xml.etree.ElementTree as ET
from decimal import Decimal, InvalidOperation

ESTADOS = ("pendiente", "enviado", "entregado", "cancelado")

def leer_pedido(nodo):
 """Comprueba un <pedido> y devuelve su id, cliente, estado e importe; lanza ValueError con el
 motivo."""
 # TODO
 raise ValueError("sin hacer")

def main():
 # TODO: analiza el XML, valida cada pedido y escribe el resumen como XML

main()
```

## 68. Detección de ataques de fuerza bruta en el registro de SSH

Difícil · 70 minutos · [apuntesdam.com/ejercicios/python/deteccion-de-ataques-ssh](https://apuntesdam.com/ejercicios/python/deteccion-de-ataques-ssh)

Cualquier servidor con SSH abierto a Internet recibe cientos de intentos de entrada al día: programas que prueban contraseñas para root, admin o test desde máquinas comprometidas. Cada intento queda en el registro de autenticación ( /var/log/auth.log en Debian y Ubuntu) con una línea como `Oct 3 10:15:01 web1 sshd[2231]: Failed password for root from 203.0.113.5 port 52314 ssh2`.

Herramientas como fail2ban leen ese registro y bloquean en el cortafuegos las IP que fallan demasiadas veces en poco tiempo. Vas a escribir su lógica: con la configuración típica, 5 fallos dentro de una ventana de 10 minutos bloquean la IP.

Lo más peligroso no son los fallos, sino un acceso correcto desde una IP que antes ha fallado: puede ser alguien que ha acertado la contraseña. Esos casos se marcan como alertas para revisarlos.

### Requisitos

- Cada línea empieza con la fecha de syslog en sus primeros 15 caracteres ( `Oct 3 10:15:01` , mes en inglés y sin año: se toma 2026). Las líneas vacías o con una fecha que no se puede leer se ignoran, igual que las que no son ni un fallo ni un acceso correcto.
- Un fallo contiene `Failed password for USUARIO from IP port` (o `for invalid user USUARIO` , si el usuario no existe en el sistema). Un acceso correcto contiene `Accepted password for USUARIO from IP port` o `Accepted publickey for ...`.
- Una IP se bloquea en el instante de un fallo si, contando ese fallo, tiene al menos 5 fallos en los últimos 10 minutos (con una diferencia de 10 minutos exactos todavía cuenta). Cada IP se bloquea una sola vez, en el primer instante en que se cumple.
- Un acceso correcto desde una IP que ya ha tenido algún fallo (en cualquier momento anterior) genera la alerta `USUARIO entró desde IP a las HH:MM:SS tras N fallos desde esa IP` («fallo» si es 1).
- Informe: `Intentos fallidos: F · Accesos correctos: A ; Bloquear: con una línea IP a las HH:MM:SS (5 fallos desde las HH:MM:SS)` por IP, en el orden en que se bloquearon, indicando el primer fallo de la ventana (o `Bloquear: ninguna IP`); `Usuarios inexistentes probados: a, b` en orden alfabético (o `ninguno`); si ha habido fallos, `Usuarios más atacados: root (6), admin (3), ...` con los tres con más fallos (a igualdad, alfabético); y `Alertas: con las alertas en orden` (o `Alertas: ninguna`).

### Formato de la entrada

- Las líneas del registro de autenticación de SSH.

### Ejemplo: Una mañana en el servidor web

#### ENTRADA

```
Oct 3 10:15:01 web1 sshd[2231]: Failed password
for root from 203.0.113.5 port 52314 ssh2
Oct 3 10:15:04 web1 sshd[2231]: Failed password
for invalid user admin from 203.0.113.5 port
52320 ssh2
Oct 3 10:15:09 web1 sshd[2232]: Failed password
for root from 203.0.113.5 port 52331 ssh2
Oct 3 10:16:10 web1 sshd[2240]: Accepted
password for ana from 198.51.100.7 port 50022
ssh2
Oct 3 10:16:12 web1 sshd[2241]: Failed password
for invalid user test from 203.0.113.5 port 52340
ssh2
Oct 3 10:16:30 web1 sshd[2242]: Accepted
publickey for luis from 192.0.2.10 port 40122
ssh2
Oct 3 10:17:02 web1 sshd[2243]: Failed password
for root from 203.0.113.5 port 52351 ssh2
Oct 3 10:20:45 web1 sshd[2250]: Failed password
for ana from 198.51.100.99 port 50100 ssh2
Oct 3 10:21:00 web1 sshd[2251]: Accepted
password for ana from 198.51.100.99 port 50101
ssh2
Oct 3 10:22:10 web1 CRON[2260]:
pam_unix(cron:session): session opened for user
root
```

#### SALIDA ESPERADA

Intentos fallidos: 6 · Accesos correctos: 3  
Bloquear:  
203.0.113.5 a las 10:17:02 (5 fallos desde las 10:15:01)  
Usuarios inexistentes probados: admin, test  
Usuarios más atacados: root (3), admin (1), ana (1)  
Alertas:  
ana entró desde 198.51.100.99 a las 10:21:00 tras 1 fallo desde esa IP

### Ejemplo: La ventana de 10 minutos

#### ENTRADA

```
Oct 3 22:00:00 srv sshd[1]: Failed password for
root from 10.0.0.9 port 1 ssh2
Oct 3 22:03:00 srv sshd[1]: Failed password for
root from 10.0.0.9 port 1 ssh2
Oct 3 22:06:00 srv sshd[1]: Failed password for
root from 10.0.0.9 port 1 ssh2
Oct 3 22:09:00 srv sshd[1]: Failed password for
root from 10.0.0.9 port 1 ssh2
Oct 3 22:12:00 srv sshd[1]: Failed password for
root from 10.0.0.9 port 1 ssh2
Oct 3 22:13:00 srv sshd[1]: Failed password for
root from 10.0.0.9 port 1 ssh2
línea basura
```

#### SALIDA ESPERADA

Intentos fallidos: 6 · Accesos correctos: 0  
Bloquear:  
10.0.0.9 a las 22:13:00 (5 fallos desde las 22:03:00)  
Usuarios inexistentes probados: ninguno  
Usuarios más atacados: root (6)  
Alertas: ninguna

#### Código de partida (python)

```
import re
import sys
from datetime import datetime, timedelta

FALLO = re.compile(r"Failed password for (invalid user)?(\\S+) from (\\S+) port")
EXITO = re.compile(r"Accepted (password|publickey) for (\\S+) from (\\S+) port")
VENTANA = timedelta(minutes=10)
MAX_FALLOS = 5

def main():
 # TODO: recorre el registro, detecta los ataques y escribe el informe
 pass

main()
```

# Soluciones

## 1. ¿Están bien anidadas las etiquetas?

Medio · Fundamentos de Lenguajes de Marcas · [apuntesdam.com/subject/lenguajes-marcas/topic/fundamentos](https://apuntesdam.com/subject/lenguajes-marcas/topic/fundamentos)

Solución (python)

```
import re, sys

texto = sys.stdin.read()

def linea(pos):
 return texto.count("\n", 0, pos) + 1

pila = [] # etiquetas abiertas
elementos = 0
profundidad = 0
raiz = None # nombre de la raíz cuando ya se ha cerrado
for m in re.finditer(r"<[^\>]*>", texto):
 etiqueta = m.group()
 if etiqueta.startswith(("<?", "<!")):
 continue # declaración, comentarios y DOCTYPE no cuentan
 l = linea(m.start())
 nombre = re.match(r"</?\s*([^\s/>]+)", etiqueta).group(1)
 if etiqueta.startswith("</"):
 # cierre
 if not pila:
 print(f"Error en la línea {l}: </{nombre}> no tiene etiqueta de apertura")
 sys.exit()
 if pila[-1] != nombre:
 print(f"Error en la línea {l}: se esperaba </{pila[-1]}> pero aparece </{nombre}>")
 sys.exit()
 pila.pop()
 if not pila:
 raiz = nombre
 continue
 if raiz is not None and not pila:
 # apertura tras cerrar la raíz
 print(f"Error en la línea {l}: <{nombre}> es un segundo elemento raíz")
 sys.exit()
 elementos += 1
 profundidad = max(profundidad, len(pila) + 1)
 if etiqueta.endswith("/>"):
 # vacía: se abre y se cierra a la vez
 if not pila:
 raiz = nombre
 else:
 pila.append(nombre)

if pila:
 print(f"Error: falta cerrar <{pila[-1]}>")
elif raiz is None:
 print("Error: el documento no tiene elemento raíz")
else:
 print(f"Bien formado: {elementos} elementos, profundidad máxima {profundidad}, raíz <{raiz}>")
```

La estructura de datos natural para comprobar el anidamiento es una pila: cada etiqueta de apertura se apila y cada cierre debe coincidir con la cima, que es la última que se abrió. Si no coincide, las etiquetas se cruzan; si la pila está vacía, sobra el cierre; y si al final queda algo, falta cerrarlo.

La profundidad de un elemento es el tamaño de la pila al abrirlo más uno. Para detectar una segunda raíz basta con recordar el nombre de la raíz cuando la pila vuelve a quedar vacía: cualquier apertura posterior con la pila vacía es otro elemento raíz.

## 2. Atributos y referencias de entidad

Difícil · Fundamentos de Lenguajes de Marcas · [apuntesdam.com/subject/lenguajes-marcas/topic/fundamentos](https://apuntesdam.com/subject/lenguajes-marcas/topic/fundamentos)

Solución (python)

```
import re, sys

texto = sys.stdin.read()
VALIDA = re.compile(r"&(amp|lt|gt|quot|apos|#[0-9]+|#[0-9a-fA-F]+);")

def linea(pos):
 return texto.count("\n", 0, pos) + 1

errores = []
atributos = referencias = 0

def revisar_referencias(cadena, l):
 """Comprueba cada & de la cadena; l es la línea donde se informa."""
 global referencias
 for i, c in enumerate(cadena):
 if c != "&":
 continue
 j = i + 1
 while j < len(cadena) and not cadena[j].isspace() and cadena[j] not in "<>":
 j += 1
 trozo = cadena[i:j + 1] if j < len(cadena) and cadena[j] == ";" else cadena[i:j]
 if VALIDA.fullmatch(trozo):
 referencias += 1
 else:
 errores.append(f"Línea {l + cadena.count(chr(10), 0, i)}: referencia no válida «{trozo}»")

def revisar_etiqueta(etiqueta, l):
 global atributos
 nombre = re.match(r"<\s*([^\s/>]+)", etiqueta).group(1)
 resto = etiqueta[etiqueta.index(nombre) + len(nombre):].rstrip(">").rstrip("/")
 lugar = f"Línea {l}, <{nombre}>"
 vistos = set()
 i = 0
 while True:
 while i < len(resto) and resto[i].isspace():
 i += 1
 if i >= len(resto):
 return
 j = i
 while j < len(resto) and not resto[j].isspace() and resto[j] != "=":
 j += 1
 atributo = resto[i:j]
 if not re.fullmatch(r"[A-Za-z_][\w.-]*", atributo):
 errores.append(f"{lugar}: nombre de atributo no válido «{atributo}»")
 return # no se sigue con esta etiqueta
 i = j
 while i < len(resto) and resto[i].isspace():
 i += 1
 if i >= len(resto) or resto[i] != "=":
 errores.append(f"{lugar}: el atributo «{atributo}» no tiene valor")
 continue # se sigue con el siguiente
 i += 1
 while i < len(resto) and resto[i].isspace():
 i += 1
 comilla = resto[i] if i < len(resto) else ""
 cierre = resto.find(comilla, i + 1) if comilla in ("'", '"') else -1
 if cierre < 0:
 errores.append(f"{lugar}: el valor de «{atributo}» no está entre comillas")
 return
 valor = resto[i + 1:cierre]
 i = cierre + 1
 atributos += 1
 if atributo in vistos:
 errores.append(f"{lugar}: el atributo «{atributo}» está repetido")
 vistos.add(atributo)
 if "<" in valor:
```

```

 errores.append(f"{lugar}: el valor de «{atributo}» contiene «<»")
 revisar_referencias(valor, l)

fin = 0
for m in re.finditer(r"<[^>]*>", texto):
 revisar_referencias(texto[fin:m.start()], linea(fin))
 fin = m.end()
 etiqueta = m.group()
 if not etiqueta.startswith(("</", "<?", "<!")):
 revisar_etiqueta(etiqueta, linea(m.start()))
 revisar_referencias(texto[fin:], linea(fin))

if errores:
 print("\n".join(errores))
 print(f"Errores: {len(errores)}")
else:
 print(f"Sin errores: {atributos} atributos y {referencias} referencias correctas")

```

Los atributos se analizan con un pequeño analizador manual que avanza un índice por el texto de la etiqueta: salta espacios, lee el nombre hasta un espacio o «=», comprueba el «=», y lee el valor hasta la comilla que cierra (la misma que abrió). Cada fallo tiene su mensaje y decide si se puede seguir: sin «=» se continúa con el siguiente atributo; sin comillas o con un nombre inválido ya no se sabe dónde empieza el siguiente, así que se abandona la etiqueta.

Para las referencias, cada «&» se extiende hasta el primer «;» sin cruzar un espacio, otro «&» o un «<», y el trozo se valida con la expresión regular de las referencias permitidas. Por eso «Pan & Café» es un error en XML aunque un navegador lo acepte en HTML: el «&» suelto debe escribirse «&amp;».

La línea de cada error de referencia se calcula sumando a la línea donde empieza el trozo de texto los saltos de línea que hay antes del «&».



Las llaves del nombre expandido se escriben dobladas ( `{{` y `}}` ) dentro de la f-string, porque una llave sola marcaría una expresión.

## 4. Genera un canal RSS 2.0 correcto

Difícil · Sindicación de contenidos: RSS y Atom · [apuntesdam.com/subject/lenguajes-marcas/topic/sindicacion-contenidos](https://apuntesdam.com/subject/lenguajes-marcas/topic/sindicacion-contenidos)

Solución (python)

```
import sys
import xml.etree.ElementTree as ET
from datetime import datetime
from email.utils import format_datetime

lineas = [l for l in sys.stdin.read().splitlines() if l.strip()]
titulo, enlace, descripcion = lineas[0].split("|")
noticias = [] # (fecha, titulo, link, descripcion, categoria)
for l in lineas[1:]:
 fecha, t, link, desc, cat = l.split("|")
 noticias.append((datetime.fromisoformat(fecha), t, link, desc, cat))

def hijos(padre, pares):
 for etiqueta, valor in pares:
 ET.SubElement(padre, etiqueta).text = valor

rss = ET.Element("rss", version="2.0")
canal = ET.SubElement(rss, "channel")
hijos(canal, [("title", titulo), ("link", enlace), ("description", descripcion), ("language", "es-es")])
noticias.sort(key=lambda n: n[0], reverse=True) # de la más reciente a la más antigua
if noticias:
 hijos(canal, [("lastBuildDate", format_datetime(noticias[0][0]))])
for fecha, t, link, desc, cat in noticias:
 item = ET.SubElement(canal, "item")
 hijos(item, [("title", t), ("link", link), ("description", desc), ("category", cat),
 ("guid", link), ("pubDate", format_datetime(fecha))])

ET.indent(rss, space=" ")
print('<?xml version="1.0" encoding="UTF-8"?>')
print(ET.tostring(rss, encoding="unicode"))
```

Construir el XML con ElementTree en lugar de concatenar texto garantiza que el resultado esté bien formado:

`SubElement` crea cada hijo en su sitio y, al escribirlo, los caracteres especiales se escapan solos (el «&» del enlace pasa a `&amp;` y el `<b>` de la descripción a `&lt;b&gt;`). Si se pegaran a mano, el canal sería XML inválido y los lectores de RSS lo rechazarían.

Las fechas se ordenan como objetos `datetime` (con su zona horaria) y se escriben en el formato RFC 822 que exige RSS 2.0 con `format_datetime`. La función `hijos` evita repetir la misma línea para cada elemento de texto.

## 5. Un agregador que mezcla RSS y Atom

Muy difícil · Sindicación de contenidos: RSS y Atom · [apuntesdam.com/subject/lenguajes-marcas/topic/sindicacion-contenidos](https://apuntesdam.com/subject/lenguajes-marcas/topic/sindicacion-contenidos)

**Solución (python)**

```
import sys
import xml.etree.ElementTree as ET
from datetime import datetime, timezone
from email.utils import parsedate_to_datetime

ATOM = "{http://www.w3.org/2005/Atom}"
partes = sys.stdin.read().split("=====")
maximo = int(partes[0].split()[1]) # «maximo N»
documentos = [ET.fromstring(p.strip()) for p in partes[1:] if p.strip()]

entradas = [] # (fecha UTC, canal, título, identificador)
for doc in documentos:
 if doc.tag == "rss":
 canal = doc.find("channel")
 nombre = canal.findtext("title")
 for item in canal.findall("item"):
 fecha = parsedate_to_datetime(item.findtext("pubDate"))
 ident = item.findtext("guid") or item.findtext("link")
 entradas.append((fecha, nombre, item.findtext("title"), ident))
 elif doc.tag == ATOM + "feed":
 nombre = doc.findtext(ATOM + "title")
 for entry in doc.findall(ATOM + "entry"):
 fecha = datetime.fromisoformat(entry.findtext(ATOM + "updated").replace("Z", "+00:00"))
 entradas.append((fecha, nombre, entry.findtext(ATOM + "title"), entry.findtext(ATOM + "id")))

unicas, vistos = [], set()
for e in entradas: # se conserva la primera de cada identificador
 if e[3] not in vistos:
 vistos.add(e[3])
 unicas.append(e)
unicas.sort(key=lambda e: (-e[0].timestamp(), e[2]))

print(f"Canales: {len(documentos)} · Entradas: {len(unicas)} (duplicadas descartadas: {len(entradas) - len(unicas)})")
for fecha, canal, titulo, _ in unicas[:maximo]:
 print(f"{fecha.astimezone(timezone.utc):%Y-%m-%d %H:%M} UTC · {canal} · {titulo}")
```

RSS y Atom guardan lo mismo con nombres distintos, así que lo primero es convertir las entradas de los dos formatos en tuplas iguales (fecha, canal, título, identificador). En Atom todos los elementos están en su espacio de nombres, y ElementTree los nombra con la URI entre llaves: por eso se busca `ATOM + "entry"`.

Las fechas llegan en formatos y zonas horarias distintos; convertidas a `datetime` con zona se pueden comparar directamente, y `astimezone(timezone.utc)` las muestra todas en UTC. El orden usa la tupla (–marca de tiempo, título): de más reciente a más antigua y, a igualdad, por título.

El identificador (guid o id) es lo que permite reconocer la misma noticia publicada en dos canales, aunque el título cambie.

## 6. Importación incremental con identificadores externos

Muy difícil · Sistemas de Gestión de Información Empresarial · [apuntesdam.com/subject/lenguajes-](https://apuntesdam.com/subject/lenguajes-)

## Solución (python)

```

import re, sys

NIF = re.compile(r"[0-9]{8}[A-Z]|[A-Z][0-9]{7}[A-Z0-9]")
EMAIL = re.compile(r"^[^s@]+@[^s@]+\.[^s@]+")

erp = {} # external_id → {"nif", "nombre", "email", "activo"} (el orden de inserción es el del ERP)
filas = [] # filas del fichero a importar: (external_id, nif, nombre, email)
seccion = None
for linea in sys.stdin.read().splitlines():
 linea = linea.strip()
 if not linea:
 continue
 if linea.startswith("["):
 seccion = linea
 continue
 campos = [c.strip() for c in linea.split(";")]
 if seccion == "[erp]":
 ext, nif, nombre, email, activo = campos
 erp[ext] = {"nif": nif, "nombre": nombre, "email": email, "activo": activo == "si"}
 else:
 filas.append(tuple(campos))

creados = actualizados = sin_cambios = errores = 0
en_fichero = set()
for n, (ext, nif, nombre, email) in enumerate(filas, start=1):
 en_fichero.add(ext)
 error = None
 if not NIF.fullmatch(nif):
 error = f"NIF no válido «{nif}»"
 elif not EMAIL.fullmatch(email):
 error = f"email no válido «{email}»"
 elif not nombre:
 error = "falta el nombre"
 if error:
 print(f"ERROR línea {n} ({ext}): {error}")
 errores += 1
 continue
 otro = next((e for e, r in erp.items() if e != ext and r["nif"] == nif), None)
 if otro:
 print(f"DUPLICADO línea {n} ({ext}): el NIF {nif} ya es de {otro}")
 errores += 1
 continue
 nuevo = {"nif": nif, "nombre": nombre, "email": email}
 if ext in erp:
 actual = erp[ext]
 cambios = [f"{campo} {actual[campo]} → {nuevo[campo]}" for campo in ("nif", "nombre", "email")
 if actual[campo] != nuevo[campo]]
 reactivado = not actual["activo"]
 if not cambios and not reactivado:
 print(f"SIN CAMBIOS {ext}")
 sin_cambios += 1
 continue
 print(f"ACTUALIZAR {ext}: {' '.join(cambios) or 'sin cambios de datos'}" + (" (reactivado)" if reactivado else ""))
 actual.update(nuevo, activo=True) # se aplica ya: afecta a las filas siguientes
 actualizados += 1
 else:
 erp[ext] = {**nuevo, "activo": True}
 print(f"CREAR {ext} ({nombre})")
 creados += 1

print(f"Creados: {creados} · Actualizados: {actualizados} · Sin cambios: {sin_cambios} · Errores: {errores}")
fuera = [e for e, r in erp.items() if r["activo"] and e not in en_fichero]
if fuera:
 print(f"No incluidos en el fichero: {' '.join(fuera)}")

```

El identificador externo es lo que hace que una importación se pueda repetir sin duplicar datos: si ya existe, la fila actualiza el registro; si no, lo crea. Las comprobaciones van en orden (formato, después duplicados con otro registro) y, en los registros existentes, se compara campo a campo para informar exactamente de qué cambia.

Los cambios se aplican en el momento sobre el diccionario `erp`, así que una fila posterior ya ve el estado actualizado: la segunda fila de `cli_005` encuentra el cliente recién creado y lo actualiza. Al final se avisa de los clientes activos que no venían en el fichero, que podrían ser bajas que alguien debe revisar.

## 7. Fichero secuencial frente a índice

Medio · Sistemas de Almacenamiento · [apuntesdam.com/subject/bases-datos/topic/almacenamiento](https://apuntesdam.com/subject/bases-datos/topic/almacenamiento)

Solución (python)

```
import sys

registros = [] # (id, nombre) en el orden del fichero (sin ordenar)
busquedas = []
for linea in sys.stdin.read().splitlines():
 p = linea.split()
 if not p:
 continue
 if p[0] == "buscar":
 busquedas.append(int(p[1]))
 else:
 registros.append((int(p[0]), p[1]))

indice = sorted((r[0], pos) for pos, r in enumerate(registros)) # (id, posición en el fichero)

def plural(n, palabra, varias):
 return f"{n} {palabra if n == 1 else varias}"

total_sec = total_ind = 0
for x in busquedas:
 # Secuencial: se leen registros desde el principio hasta encontrarlo
 pos = next((i for i, r in enumerate(registros) if r[0] == x), None)
 lecturas = pos + 1 if pos is not None else len(registros)
 # Índice: búsqueda binaria sobre la lista ordenada
 lo, hi, comparaciones, encontrado = 0, len(indice) - 1, 0, None
 while lo <= hi:
 mid = (lo + hi) // 2
 comparaciones += 1
 if indice[mid][0] == x:
 encontrado = indice[mid][1]
 break
 if indice[mid][0] < x:
 lo = mid + 1
 else:
 hi = mid - 1
 total_sec += lecturas
 total_ind += comparaciones + (1 if encontrado is not None else 0)
 sec = f"secuencial {plural(lecturas, 'lectura', 'lecturas')}}"
 ind = f"índice {plural(comparaciones, 'comparación', 'comparaciones')}}"
 if encontrado is not None:
 print(f"buscar {x}: encontrado ({registros[encontrado][1]}) · {sec} · {ind} + 1 lectura")
 else:
 print(f"buscar {x}: no existe · {sec} · {ind}")
print(f"Total secuencial: {total_sec} lecturas · índice: {total_ind} accesos · ahorro {round(100 - 100 * total_ind / total_sec)} %")
```

La búsqueda secuencial lee el fichero desde el principio y, en el peor caso (el registro no está o es el último), lo lee entero: con un millón de registros, un millón de lecturas. El índice es una lista ordenada de (clave, posición) en la que se puede hacer una búsqueda binaria: cada comparación descarta la mitad de lo que queda, y la posición encontrada lleva al registro con una sola lectura.

Con 40 registros la binaria nunca necesita más de 6 comparaciones ( $\log_2 40 \approx 5,3$ ), y con un millón bastarían 20. Esa es la idea de los índices B-tree de cualquier base de datos, que además organizan el índice en bloques de disco.

## 8. Fragmentación de una base de datos distribuida

Difícil · Sistemas de Almacenamiento · [apuntesdam.com/subject/bases-datos/topic/almacenamiento](http://apuntesdam.com/subject/bases-datos/topic/almacenamiento)

Solución (python)

```
import sys

columnas, filas, fragmentos, consultas = [], [], [], []
for linea in sys.stdin.read().splitlines():
 p = linea.split(" ", 1)
 if p[0] == "tabla":
 columnas = p[1].split(" ")[1].split(",")
 elif p[0] == "fila":
 valores = p[1].split(",")
 filas.append(dict(zip(columnas, valores)))
 elif p[0] == "fragmento":
 # fragmento F1 sede=Madrid provincia=Madrid, Toledo
 nombre, sede, cond = p[1].split(" ")
 campo, valores = cond.split("=")
 fragmentos.append((nombre, sede.split("=")[1], campo, valores.split(",")))
 elif p[0] == "consulta":
 consultas.append(p[1])

def filas_texto(n):
 return f"{n} fila" if n == 1 else f"{n} filas"

contenido = {f[0]: [r for r in filas if r[f[2]] in f[3]] for f in fragmentos}
for nombre, sede, _, _ in fragmentos:
 print(f"{nombre} ({sede}): {filas_texto(len(contenido[nombre]))}")

Completitud: cada fila debe estar en algún fragmento
campo0 = fragmentos[0][2]
perdidas = [r for r in filas if not any(r in contenido[f[0]] for f in fragmentos)]
if perdidas:
 print("Completitud: NO, sin fragmento: " + ", ".join(f"id {r['id']} ({r[campo0]})" for r in
 perdidas))
else:
 print("Completitud: sí")

Disjunción: ninguna fila en dos fragmentos
repetidas = []
for r in filas:
 donde = [f[0] for f in fragmentos if r in contenido[f[0]]]
 if len(donde) > 1:
 repetidas.append(f"id {r['id']} en " + ", ".join(donde[:-1]) + " y " + donde[-1])
print("Disjunción: " + ("NO, " + "; ".join(repetidas) if repetidas else "sí"))

def cumple(r, campo, op, valor):
 if op == "=":
 return r[campo] == valor
 return float(r[campo]) > float(valor) if op == ">" else float(r[campo]) < float(valor)

for c in consultas:
 if c == "todos":
 campo = op = valor = None
 else:
 op = next(o for o in "<=>" if o in c)
 campo, valor = c.split(op)
 elegidos = [f for f in fragmentos
 if not (op == "=" and campo == f[2]) or valor in f[3]] # poda solo en igualdades del
 campo
 sedes = list(dict.fromkeys(f[1] for f in elegidos)) # sin repetir, en orden
 ids = sorted((r["id"] for f in elegidos for r in contenido[f[0]] if campo is None or cumple(r, campo,
 op, valor)), key=int)
 texto = f"consulta {c} → sedes {'', '.join(sedes) or 'ninguna'} · {filas_texto(len(ids))}"
 print(texto + (": " + ", ".join(ids) if ids else ""))
```

Cada fragmento horizontal es una selección de la tabla: las filas cuyo campo toma alguno de sus valores. Con el contenido de cada fragmento calculado, las dos propiedades de una buena fragmentación se comprueban directamente: completitud (toda fila está en algún fragmento; si no, se pierde) y disjunción (ninguna está en dos; si no,

hay que mantener dos copias coherentes).

En las consultas se ve la ventaja de fragmentar: una igualdad sobre el campo de fragmentación solo necesita preguntar a las sedes cuyo fragmento incluye ese valor, y si ninguno lo incluye, a ninguna. Cualquier otra condición obliga a preguntar a todas. `dict.fromkeys` quita las sedes repetidas conservando el orden.

## 9. Anonimizar datos personales: k-anonimato

Muy difícil · Sistemas de Almacenamiento · [apuntesdam.com/subject/bases-datos/topic/almacenamiento](https://apuntesdam.com/subject/bases-datos/topic/almacenamiento)

Solución (python)

```
import sys
from collections import Counter, defaultdict

lineas = [l for l in sys.stdin.read().splitlines() if l.strip()]
k_objetivo = int(lineas[0].split()[1])
cabecera = lineas[1].split(";") # nombre;dni;edad;cp;sexo;diagnostico
filas = [dict(zip(cabecera, l.split(";"))) for l in lineas[2:]]

print("Identificadores eliminados: nombre, dni")
anonimas = []
for f in filas:
 decada = int(f["edad"]) // 10 * 10
 anonimas.append({"edad": f"{decada}-{decada + 9}", "cp": f["cp"][:3] + "***",
 "sexo": f["sexo"], "diagnostico": f["diagnostico"]})

clave = lambda f: (f["edad"], f["cp"], f["sexo"]) # cuasi-identificadores
tamanos = Counter(clave(f) for f in anonimas)
print(f"k antes de suprimir: {min(tamanos.values())}")

publicables = [f for f in anonimas if tamanos[clave(f)] >= k_objetivo]
if not publicables:
 print("No queda ningún dato publicable")
 sys.exit()

print("edad;cp;sexo;diagnostico")
for f in publicables:
 print(f"{f['edad']}";{f['cp']}";{f['sexo']}";{f['diagnostico']}")
diagnosticos = defaultdict(set)
for f in publicables:
 diagnosticos[clave(f)].add(f["diagnostico"])
k = min(tamanos[g] for g in diagnosticos)
l_div = min(len(d) for d in diagnosticos.values())
print(f"Grupos: {len(diagnosticos)} · k = {k} (objetivo {k_objetivo}) · filas suprimidas: {len(anonimas) - len(publicables)} · l-diversidad: {l_div}")
if l_div == 1:
 print("Atención: en algún grupo todos tienen el mismo diagnóstico, que se puede deducir")
```

Quitar el nombre y el DNI no basta: la combinación de edad, código postal y sexo identifica a mucha gente. Por eso se generalizan (década y tres cifras del código postal) hasta que cada combinación la comparten varias personas;

`Counter` cuenta el tamaño de cada grupo, y el más pequeño es la k del conjunto. Los grupos con menos de K personas se suprimen, porque en ellos alguien sería reconocible.

La l-diversidad muestra el límite del k-anonimato: si todos los de un grupo tienen el mismo diagnóstico, saber que alguien está en ese grupo basta para conocer su enfermedad, aunque no se sepa cuál de las filas es la suya. Un `defaultdict(set)` reúne los diagnósticos distintos de cada grupo.

## 10. Comprobar cardinalidades con datos reales

Medio · Modelo Entidad-Relación · [apuntesdam.com/subject/bases-datos/topic/modelo-er](https://apuntesdam.com/subject/bases-datos/topic/modelo-er)

Solución (python)

```
import sys

instancias = {} # entidad → lista de ocurrencias
relacion = None # (nombre, entidad1, (min1, max1), entidad2, (min2, max2))
pares = []
for linea in sys.stdin.read().splitlines():
 p = linea.split()
 if not p:
 continue
 if p[0] == "entidad":
 instancias[p[1]] = p[2:]
 elif p[0] == "relacion":
 # relacion cursa Alumno(1,n) Modulo(0,2)
 lados = []
 for t in p[2:4]:
 nombre, card = t[:-1].split("(")
 mn, mx = card.split(",")
 lados.append((nombre, (int(mn), None if mx == "n" else int(mx))))
 relacion = (p[1], lados[0][0], lados[0][1], lados[1][0], lados[1][1])
 else:
 pares.append((p[1], p[2])) # cursa A1 BD

nombre, e1, card1, e2, card2 = relacion
incumplimientos = 0
validos = []
for a, b in pares:
 if a not in instancias[e1] or b not in instancias[e2]:
 print(f"Ocurrencia desconocida: {a} {b}")
 incumplimientos += 1
 elif (a, b) in validos:
 print(f"Par repetido: {a} {b}")
 incumplimientos += 1
 else:
 validos.append((a, b))
print(f"Ocurrencias válidas de {nombre}: {len(validos)}")

for entidad, (minimo, maximo), lado in ((e1, card1, 0), (e2, card2, 1)):
 for i in instancias[entidad]:
 k = sum(1 for par in validos if par[lado] == i) # en cuántas ocurrencias participa
 if k < minimo:
 print(f"{entidad} {i} participa {k} veces en {nombre} (mínimo {minimo})")
 incumplimientos += 1
 elif maximo is not None and k > maximo:
 print(f"{entidad} {i} participa {k} veces en {nombre} (máximo {maximo})")
 incumplimientos += 1

if incumplimientos:
 print(f"Restricciones: {incumplimientos} {'incumplimiento' if incumplimientos == 1 else 'incumplimientos'}")
else:
 print("Restricciones: correctas")
```

La cardinalidad (mínimo, máximo) de una entidad dice en cuántas ocurrencias de la relación puede aparecer cada una de sus instancias. Comprobarla con datos es contar: para cada instancia, cuántos pares válidos la contienen en su lado, y compararlo con el mínimo (participación obligatoria si es 1) y el máximo (n significa sin límite, que se guarda como `None`).

Los pares desconocidos o repetidos no cuentan como ocurrencias, igual que en una base de datos una clave ajena inexistente o una fila duplicada en una tabla con clave primaria serían rechazadas. Recorrer las dos entidades con el mismo bucle, indicando qué lado del par mirar, evita escribir dos veces el mismo código.

## 11. Del diagrama E/R a las tablas

Difícil · Modelo Entidad-Relación · [apuntesdam.com/subject/bases-datos/topic/modelo-er](https://apuntesdam.com/subject/bases-datos/topic/modelo-er)

Solución (python)

```
import sys

entidades = {} # nombre → (claves, atributos) en orden
relaciones = [] # (nombre, e1, card1, e2, card2, atributos)
for linea in sys.stdin.read().splitlines():
 p = linea.split()
 if not p:
 continue
 if p[0] == "entidad": # entidad Cliente *dni nombre email
 claves = [a[1:] for a in p[2:] if a.startswith("*")]
 attrs = [a.lstrip("*") for a in p[2:]]
 entidades[p[1]] = (claves, attrs)
 elif p[0] == "relacion": # relacion realiza Cliente 1 Pedido N [atributos...]
 relaciones.append((p[1], p[2], p[3], p[4], p[5], p[6]))

Cada tabla: nombre → lista de columnas (nombre, es_clave, tabla_referenciada, única)
tablas = {e: [(a, a in claves, None, False) for a in attrs] for e, (claves, attrs) in entidades.items()}
nm = 0

def clave_ajena(tabla, clave, relacion, origen, destino):
 """Nombre de la columna que recibe la clave de origen en la tabla destino."""
 if origen == destino:
 return f"{clave}_{relacion}" # relación reflexiva
 if any(col[0] == clave for col in tablas[tabla]):
 return f"{clave}_{origen.lower()}" # el nombre ya está ocupado
 return clave

for nombre, e1, c1, e2, c2, attrs in relaciones:
 if c1 == "1" and c2 == "1": # 1:1 → clave ajena única en la segunda
 for k in entidades[e1][0]:
 tablas[e2].append((clave_ajena(e2, k, nombre, e1, e2), False, e1, True))
 tablas[e2] += [(a, False, None, False) for a in attrs]
 elif c1 == "1" or c2 == "1": # 1:N → la clave del lado 1 pasa al lado N
 uno, muchos = (e1, e2) if c1 == "1" else (e2, e1)
 for k in entidades[uno][0]:
 tablas[muchos].append((clave_ajena(muchos, k, nombre, uno, muchos), False, uno, False))
 tablas[muchos] += [(a, False, None, False) for a in attrs]
 else: # N:M → tabla nueva
 nm += 1
 tablas[nombre] = []
 for e in (e1, e2):
 for k in entidades[e][0]:
 col = f"{k}_{nombre}" if e1 == e2 and e is e2 else clave_ajena(nombre, k, nombre, e,
 nombre)
 tablas[nombre].append((col, True, e, False))
 tablas[nombre] += [(a, False, None, False) for a in attrs]

for t, cols in tablas.items():
 textos = [(f"*" if pk else "") + col + (f"→{ref}" if ref else "") + (" único" if unico else "")
 for col, pk, ref, unico in cols]
 print(f"{t}({', '.join(textos)})")
print(f"Tablas: {len(tablas)} ({nm} por relaciones N:M)")
```

Cada tabla se representa como una lista de columnas con lo necesario para escribirla después: nombre, si es clave primaria, a qué tabla referencia y si es única. Las entidades dan las tablas iniciales y cada relación, según su tipo, añade columnas a una tabla existente o crea una nueva: en 1:N la clave del lado 1 viaja al lado N (cada pedido guarda su cliente); en 1:1, igual pero única; y en N:M hace falta una tabla intermedia cuya clave es la unión de las dos claves.

La función `clave_ajena` resuelve los conflictos de nombres: en una relación reflexiva (un empleado dirige a empleados) la columna se llama por la relación, `id_dirige`; y si el nombre ya existe en la tabla, se le añade la entidad de origen, como `nombre_departamento`. Los diccionarios de Python conservan el orden de inserción, así que las tablas de las relaciones N:M salen detrás de las de las entidades.

## 12. Claves candidatas y forma normal

Muy difícil · Modelo Entidad-Relación · [apuntesdam.com/subject/bases-datos/topic/modelo-er](http://apuntesdam.com/subject/bases-datos/topic/modelo-er)

Solución (python)

```
import sys
from itertools import combinations

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]
atributos = lineas[0][lineas[0].index("(") + 1:-1].split(",") # R(A,B,C,D)
dependencias = []
for l in lineas[1:]:
 izq, der = l.split("->")
 dependencias.append([a.strip() for a in izq.split(",")], [a.strip() for a in der.split(",")]))

def ordenar(conjunto):
 return [a for a in atributos if a in conjunto]

def cierre(x):
 resultado = set(x)
 cambia = True
 while cambia:
 cambia = False
 for izq, der in dependencias:
 if set(izq) <= resultado and not set(der) <= resultado:
 resultado |= set(der)
 cambia = True
 return resultado

todos = set(atributos)
claves = []
for tam in range(1, len(atributos) + 1):
 for combo in combinations(atributos, tam):
 if cierre(combo) == todos and not any(set(k) <= set(combo) for k in claves): # mínimas
 claves.append(combo)
primos = ordenar(set().union(*map(set, claves)))
print("Claves candidatas: " + ", ".join("{ " + ", ".join(k) + "}" for k in claves))
print("Atributos primos: " + (", ".join(primos) or "ninguno"))

fallos = set()
for izq, der in dependencias:
 texto = f"{' '.join(izq)} -> {' '.join(der)}"
 x = ", ".join(izq)
 if set(der) <= set(izq):
 print(f"{texto}: trivial")
 elif cierre(izq) == todos:
 print(f"{texto}: correcta ({x} es superclave)")
 elif set(der) - set(izq) <= set(primos):
 print(f"{texto}: incumple FNBC ({x} no es superclave)")
 fallos.add("FNBC")
 else:
 parcial = next((k for k in claves if set(izq) < set(k)), None)
 if parcial:
 print(f"{texto}: incumple 2FN (dependencia parcial de la clave {{{ ' '.join(parcial)}}})")
 fallos.add("2FN")
 else:
 print(f"{texto}: incumple 3FN (dependencia transitiva)")
 fallos.add("3FN")

if "2FN" in fallos:
 forma = "1FN"
elif "3FN" in fallos:
 forma = "2FN"
elif "FNBC" in fallos:
 forma = "3FN"
else:
 forma = "FNBC"
print(f"Forma normal: {forma}")
```

El cierre de un conjunto de atributos es todo lo que se puede deducir de él aplicando las dependencias una y otra vez hasta que no se añada nada. Con él se responde a todo lo demás: un conjunto es superclave si su cierre es el esquema entero, y las claves candidatas son las superclaves mínimas, que se buscan probando combinaciones de menor a mayor tamaño y descartando las que contienen una clave ya encontrada.

Cada dependencia no trivial cuya parte izquierda no es superclave rompe alguna forma normal, y la clasificación sigue la teoría: si solo determina atributos primos, rompe únicamente la FNBC; si su izquierda es parte de una clave, es una dependencia parcial (rompe la 2FN); y si no, es transitiva (rompe la 3FN). La forma normal de la relación es la más alta que no se incumple.

En la f-string, `{{ y }}` escriben llaves literales, necesarias para mostrar una clave como `{alumno,modulo}`.

### 13. ¿Qué columnas pueden ser clave?

Medio · Modelo Relacional · [apuntesdam.com/subject/bases-datos/topic/modelo-relacional](https://apuntesdam.com/subject/bases-datos/topic/modelo-relacional)

Solución (python)

```
import sys
from itertools import combinations

lineas = [l for l in sys.stdin.read().splitlines() if l.strip()]
columnas = lineas[0].split(",")
filas = [[v if v != "" else None for v in l.split(",")] for l in lineas[1:]]

n = len(filas)
for i, col in enumerate(columnas):
 distintos = {f[i] for f in filas if f[i] is not None}
 print(f"{col}: {len(distintos)} valores distintos de {n}")

con_nulos = [col for i, col in enumerate(columnas) if any(f[i] is None for f in filas)]
print("Columnas con nulos: " + (", ".join(con_nulos) or "ninguna"))

candidatas = [i for i, col in enumerate(columnas) if col not in con_nulos] # una clave no admite nulos
claves = []
for tam in range(1, 4):
 for combo in combinations(candidatas, tam):
 if any(set(k) <= set(combo) for k in claves):
 continue # no es mínima
 valores = {tuple(f[i] for i in combo) for f in filas}
 if len(valores) == n: # ninguna combinación de valores se repite
 claves.append(combo)
if claves:
 print("Claves candidatas según los datos: " + ", ".join("{ " + ", ".join(columnas[i] for i in k) + "}"
 for k in claves))
else:
 print("Claves candidatas según los datos: ninguna con hasta 3 columnas")
print("Aviso: los datos solo descartan claves; que hoy no haya repetidos no garantiza que no los haya mañana")
```

Una combinación de columnas puede ser clave si no tiene nulos y sus valores no se repiten en ninguna fila: basta con meter las tuplas de valores en un conjunto y comprobar que tiene tantos elementos como filas. Se prueban combinaciones de 1, 2 y 3 columnas en orden, descartando las que contienen una clave ya encontrada, para quedarse solo con las mínimas.

El aviso final es la lección importante: los datos pueden demostrar que algo no es clave (hay un repetido), pero nunca que lo sea. Que hoy no haya dos alumnos con el mismo nombre y curso no significa que no los vaya a haber; la clave se decide por el significado de los datos, y los datos solo sirven para descartar.

## 14. Integridad referencial al borrar e insertar

Difícil · Modelo Relacional · [apuntesdam.com/subject/bases-datos/topic/modelo-relacional](https://apuntesdam.com/subject/bases-datos/topic/modelo-relacional)

Solución (python)

```
import sys
import copy

tablas = {} # nombre -> (columnas, {id: fila como dict})
claves_ajenas = [] # (tabla, columna, tabla_ref, regla)
ordenes = []
for linea in sys.stdin.read().splitlines():
 p = linea.split()
 if not p:
 continue
 if p[0] == "tabla":
 tablas[p[1]] = (p[2:], {})
 elif p[0] == "fk": # fk pedido.cliente_id -> cliente.id ON DELETE CASCADE
 t, c = p[1].split(".")
 claves_ajenas.append((t, c, p[3].split(".")[0], " ".join(p[6:])))
 elif p[0] == "fila":
 cols, filas = tablas[p[1]]
 fila = dict(zip(cols, [None if v == "NULL" else v for v in p[2:])))
 filas[fila["id"]] = fila
 else:
 ordenes.append(p)

class Rechazo(Exception):
 pass

def borrar(datos, tabla, ident, borradas, anuladas):
 """Borra la fila y aplica las reglas ON DELETE de quienes la referencian (recursivamente)."""
 del datos[tbl][tbl][ident]
 borradas.append(f"{tbl}#{ident}")
 for hija, columna, ref, regla in claves_ajenas:
 if ref != tbl:
 continue
 usan = [f for f in datos[hija][1].values() if f[columna] == ident]
 if not usan:
 continue
 if regla == "RESTRICT":
 n = len(usan)
 raise Rechazo(f"no se puede borrar {tbl} {ident}: lo usan {n} {'fila' if n == 1 else 'filas'} de {hija}")
 for f in usan:
 if regla == "CASCADE":
 if f["id"] in datos[hija][1]:
 borrar(datos, hija, f["id"], borradas, anuladas)
 else:
 f[columna] = None
 anuladas.append(f"{hija}#{f['id']}.{columna}")

for orden in ordenes:
 if orden[0] == "borrar":
 tabla, ident = orden[1], orden[2]
 prefijo = f"borrar {tbl} {ident} ->"
 if ident not in tablas[tbl][tbl]:
 print(f"{prefijo} ERROR: no existe {tbl} {ident}")
 continue
 copia = copy.deepcopy(tablas) # transacción: se trabaja sobre una copia
 borradas, anuladas = [], []
 try:
 borrar(copia, tabla, ident, borradas, anuladas)
 except Rechazo as e:
 print(f"{prefijo} ERROR: {e}") # rollback: la copia se descarta
 continue
 tablas = copia # commit
 print(f"{prefijo} borradas: {' '.join(borradas)}" + (f"; a NULL: {' '.join(anuladas)}" if anuladas else ""))
 elif orden[0] == "insertar":
 tabla = orden[1]
```

```

cols, filas = tablas[tbl]
fila = dict(zip(cols, [None if v == "NULL" else v for v in orden[2:]]))
prefijo = f"insertar {tbl} {fila['id']} →"
if fila["id"] in filas:
 print(f"{prefijo} ERROR: ya existe {tbl} {fila['id']}")
 continue
falta = next(((ref, fila[col]) for t, col, ref, _ in claves_ajenas
 if t == tbl and fila[col] is not None and fila[col] not in tablas[ref][1]), None)
if falta:
 print(f"{prefijo} ERROR: no existe {falta[0]} {falta[1]}")
 continue
filas[fila["id"]] = fila
print(f"{prefijo} correcto")
elif orden[0] == "mostrar":
 for nombre, (cols, filas) in tablas.items():
 texto = " | ".join(" ".join("NULL" if f[c] is None else f[c] for c in cols) for f in
 filas.values())
 print(f"{nombre}: {texto or '(vacía)'}")

```

El borrado es recursivo: se borra la fila y, para cada clave ajena que apunta a su tabla, se aplica su regla a las filas que la usan; con CASCADE se las borra a ellas, con lo que la regla se aplica también a sus dependientes. Así se recorre en profundidad todo lo que cuelga de la fila, en el orden que muestra el resultado.

Para que el borrado sea «todo o nada» se trabaja sobre una copia profunda de las tablas: si en cualquier punto aparece un RESTRICT, se lanza una excepción, la copia se descarta y nada ha cambiado (rollback); si todo va bien, la copia sustituye a las tablas (commit). Es lo que hace un SGBD con una transacción.

Insertar comprueba lo contrario: que el id no exista y que cada clave ajena no nula apunte a una fila existente, porque una fila hija no puede quedar huérfana.

## 15. Un intérprete de álgebra relacional

Muy difícil · Modelo Relacional · [apuntesdam.com/subject/bases-datos/topic/modelo-relacional](http://apuntesdam.com/subject/bases-datos/topic/modelo-relacional)

Solución (python)

```
import sys

relaciones = {} # nombre → (columnas, lista de tuplas)

class ErrorAlgebra(Exception):
 pass

def obtener(nombre):
 if nombre not in relaciones:
 raise ErrorAlgebra(f"no existe {nombre}")
 return relaciones[nombre]

def columna(nombre, cols, c):
 if c not in cols:
 raise ErrorAlgebra(f"no existe la columna {c} en {nombre}")
 return cols.index(c)

def sin_repetidos(tuplas):
 return list(dict.fromkeys(tuplas)) # conserva la primera aparición

def compara(a, op, b):
 try:
 a, b = float(a), float(b) # como números si los dos lo son
 except ValueError:
 pass
 return {"=": a == b, "<>": a != b, ">": a > b, "<": a < b}[op]

def operar(op, args):
 if op == "seleccion":
 cols, filas = obtener(args[0])
 for o in ("<>", "=", ">", "<"):
 if o in args[1]:
 c, valor = args[1].split(o, 1)
 break
 i = columna(args[0], cols, c)
 return cols, [t for t in filas if compara(t[i], o, valor)]
 if op == "proyeccion":
 cols, filas = obtener(args[0])
 indices = [columna(args[0], cols, c) for c in args[1].split(",")]
 return [cols[i] for i in indices], sin_repetidos(tuple(t[i] for i in indices) for t in filas)
 if op == "join":
 c1, f1 = obtener(args[0])
 c2, f2 = obtener(args[1])
 a, b = args[2].split("=")
 i, j = columna(args[0], c1, a), columna(args[1], c2, b)
 cols = c1 + [c if c not in c1 else f"{args[1]}.{c}" for c in c2]
 return cols, [t1 + t2 for t1 in f1 for t2 in f2 if t1[i] == t2[j]]
 if op in ("union", "diferencia", "interseccion"):
 c1, f1 = obtener(args[0])
 c2, f2 = obtener(args[1])
 if c1 != c2:
 raise ErrorAlgebra(f"esquemas incompatibles ({','.join(c1)}) y ({','.join(c2)})")
 if op == "union":
 return c1, sin_repetidos(f1 + f2)
 if op == "diferencia":
 return c1, sin_repetidos(t for t in f1 if t not in f2)
 return c1, sin_repetidos(t for t in f1 if t in f2)
 if op == "renombrar":
 cols, filas = obtener(args[0])
 viejo, nuevo = args[1].split("=")
 i = columna(args[0], cols, viejo)
 return cols[:i] + [nuevo] + cols[i + 1:], list(filas)
 raise ErrorAlgebra(f"operación desconocida {op}")

lineas = sys.stdin.read().splitlines()
i = 0
```

```

while i < len(lineas):
 l = lineas[i].strip()
 if l.startswith("tabla "):
 # tabla nombre col1,col2 ... filas ... fin
 nombre, cols = l.split()[1], l.split()[2].split(",")
 filas = []
 i += 1
 while lineas[i].strip() != "fin":
 filas.append(tuple(lineas[i].strip().split(",")))
 i += 1
 relaciones[nombre] = (cols, filas)
 elif l.startswith("mostrar "):
 nombre = l.split()[1]
 if nombre not in relaciones:
 print(f"ERROR: no existe {nombre}")
 else:
 cols, filas = relaciones[nombre]
 print(f"{nombre}({', '.join(cols)})")
 for t in filas:
 print(" " + " | ".join(t))
 print(f" {len(filas)} {'tupla' if len(filas) == 1 else 'tuplas'}")
 elif l:
 destino, expresion = [x.strip() for x in l.split("=", 1)]
 op, *args = expresion.split()
 try:
 relaciones[destino] = operar(op, args)
 except ErrorAlgebra as e:
 print(f"ERROR en {destino}: {e}")
 i += 1

```

Cada operación del álgebra relacional recibe relaciones (columnas y tuplas) y devuelve otra relación, así que se pueden encadenar guardando cada resultado con un nombre. La función `operar` implementa cada una en pocas líneas: la selección filtra tuplas, la proyección se queda con columnas, el join combina las tuplas que cumplen la condición y las operaciones de conjuntos exigen el mismo esquema.

Una relación es un conjunto: no tiene tuplas repetidas, y por eso la proyección y las operaciones de conjuntos las eliminan con `dict.fromkeys`, que además conserva el orden de la primera aparición. Los errores se lanzan como excepción desde donde se detectan y se capturan en un solo sitio; así el resultado no se guarda y un `mostrar` posterior dice que no existe.

Esta es la teoría que hay debajo de SQL: `SELECT nombre FROM alumno JOIN matricula ... WHERE curso = '1DAM'` es una selección, un join y una proyección.

## 16. ¿Por qué mi disco de 1 TB tiene 931 GB?

Medio · Componentes Físicos · [apuntesdam.com/subject/sistemas/topic/componentes-fisicos](https://apuntesdam.com/subject/sistemas/topic/componentes-fisicos)

**Solución** (python)

```

import sys

DECIMAL = {"B": 1, "KB": 10**3, "MB": 10**6, "GB": 10**9, "TB": 10**12}
BINARIO = {"KiB": 2**10, "MiB": 2**20, "GiB": 2**30, "TiB": 2**40}
UNIDADES = {**DECIMAL, **BINARIO}

for linea in sys.stdin.read().splitlines():
 if not linea.strip():
 continue
 cantidad, unidad = linea.split()
 if unidad not in UNIDADES:
 print(f"{linea.strip()}: unidad desconocida")
 continue
 bytes_ = round(float(cantidad) * UNIDADES[unidad])
 print(f"{cantidad} {unidad} = {bytes_} bytes = {bytes_ / 10**9:.2f} GB = {bytes_ / 2**30:.2f} GiB")

```

Todo pasa por los bytes: primero se convierte la cantidad a bytes con el factor de su unidad y después se divide entre  $10^9$  (GB) y entre  $2^{30}$  (GiB). Unir los dos diccionarios en uno (`{**DECIMAL, **BINARIO}`) permite buscar cualquier unidad de una vez, y que distinga mayúsculas hace que «Gb» (gigabits) se rechace.

Así se ve de dónde salen los «GB que faltan»:  $10^{12}$  bytes son 1000 GB para el fabricante, pero 931,32 GiB para el sistema operativo, porque un GiB es casi un 7,4 % mayor que un GB.

## 17. Dimensionar la fuente de alimentación

Difícil · Componentes Físicos · [apuntesdam.com/subject/sistemas/topic/componentes-fisicos](https://apuntesdam.com/subject/sistemas/topic/componentes-fisicos)

Solución (python)

```
import sys

CATALOGO = [450, 550, 650, 750, 850, 1000, 1200]
EFICIENCIA = {"bronze": 0.85, "gold": 0.90, "platinum": 0.92}

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

consumo = 0
config = {}
for linea in lineas:
 if "=" in linea:
 clave, valor = linea.split("=")
 config[clave] = valor
 else:
 _, vatios = linea.rsplit(" ", 1) # el nombre puede tener espacios: los vatios son lo último
 consumo += float(vatios)

margen = consumo * 1.3
fuente = next((f for f in CATALOGO if f >= margen), None)
pared = consumo * float(config["uso"]) / 100 / EFICIENCIA[config["certificacion"]]
kwh = pared * float(config["horas"]) * 365 / 1000

print(f"Consumo de los componentes: {consumo:g} W")
print(f"Con un margen del 30 %: {margen:.0f} W")
print(f"Fuente recomendada: {f'{fuente} W' if fuente else 'ninguna del catálogo es suficiente'}")
print(f"Consumo real de la pared: {pared:.1f} W")
print(f"Coste anual: {kwh * float(config['precio']):.2f} € ({kwh:.1f} kWh)")
```

Las líneas de componentes y las de configuración se distinguen por el «=». En los componentes, `rsplit(" ", 1)` parte solo por el último espacio, porque el nombre puede tener espacios y los vatios siempre van al final. `next(..., None)` devuelve la primera fuente del catálogo que llega al consumo con margen, o `None` si ninguna llega.

El margen se deja porque los componentes tienen picos de consumo y las fuentes rinden peor cerca de su máximo. El consumo real es menor que el máximo (el equipo no siempre va a tope), pero de la pared sale más de lo que llega a los componentes: se divide entre la eficiencia, y la diferencia se pierde en calor.

## 18. Ancho de banda y cuellos de botella

Muy difícil · Componentes Físicos · [apuntesdam.com/subject/sistemas/topic/componentes-fisicos](https://apuntesdam.com/subject/sistemas/topic/componentes-fisicos)

Solución (python)

```
import sys

PCIE_GBS_POR_LINEA = {"3": 0.985, "4": 1.969, "5": 3.938}

for linea in sys.stdin.read().splitlines():
 p = linea.split()
 if not p:
 continue
 if p[0] == "ram":
 mts = int(p[1].split("-")[1])
 canales = int(p[2])
 gbs = mts * 8 * canales / 1000
 print(f"Memoria {p[1]} en {canales} {'canal' if canales == 1 else 'canales'}: {gbs:.1f} GB/s")
 elif p[0] == "ssd":
 ssd = int(p[1])
 gen = p[2].replace("pcie", "")
 lineas = int(p[3].replace("x", ""))
 enlace_gbs = PCIE_GBS_POR_LINEA[gen] * lineas
 enlace_mbs = round(enlace_gbs * 1000)
 texto = f"SSD de {ssd} MB/s en PCIe {gen}.0 x{lineas} ({enlace_gbs:.1f} GB/s)"
 if enlace_mbs < ssd:
 print(f"{texto}: limitado a {enlace_mbs} MB/s ({ssd - enlace_mbs} MB/s desaprovechados)")
 else:
 print(f"{texto}: sin cuello de botella")
```

Cada línea dice primero qué componente es, y el resto se interpreta según el tipo. En la memoria, los MT/s son transferencias por segundo y cada una mueve 8 bytes por canal, así que doble canal duplica el ancho de banda. En el SSD, el enlace es lo que transporta una línea PCIe por el número de líneas.

El cuello de botella aparece cuando el enlace es más lento que el SSD: un SSD de 7400 MB/s en una ranura PCIe 3.0 x4 nunca pasará de unos 3940 MB/s. Se compara todo en MB/s, redondeando el enlace a un número entero.

## 19. Planificador FCFS y SJF

Medio · Sistemas Operativos · [apuntesdam.com/subject/sistemas/topic/sistemas-operativos](https://apuntesdam.com/subject/sistemas/topic/sistemas-operativos)

Solución (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
algoritmo = lineas[0][0]
procesos = [(nombre, int(llegada), int(rafaga)) for nombre, llegada, rafaga in lineas[1:]]

t = 0
gantt = []
inicio, fin = {}, {}
pendientes = list(procesos)
while pendientes:
 llegados = [p for p in pendientes if p[1] <= t]
 if not llegados:
 siguiente = min(p[1] for p in pendientes)
 gantt.append(f"ocioso {t}-{siguiente}")
 t = siguiente
 continue
 if algoritmo == "FCFS":
 p = min(llegados, key=lambda p: p[1])
 else:
 p = min(llegados, key=lambda p: (p[2], p[1], p[0]))
 pendientes.remove(p)
 inicio[p[0]] = t
 t += p[2]
 fin[p[0]] = t
 gantt.append(f"{p[0]} {inicio[p[0]]}-{t}")

print(" | ".join(gantt))
esperas, retornos = [], []
for nombre, llegada, _ in procesos:
 esperas.append(inicio[nombre] - llegada)
 retornos.append(fin[nombre] - llegada)
 print(f"{nombre}: espera {esperas[-1]}, retorno {retornos[-1]}")
print(f"Espera media: {sum(esperas) / len(esperas):.2f}")
print(f"Retorno medio: {sum(retornos) / len(retornos):.2f}")
```

La simulación avanza un reloj `t`. Cada vez que la CPU queda libre se miran los procesos que ya han llegado y se elige uno con `min` y una clave: en FCFS, la llegada; en SJF, la tupla (ráfaga, llegada, nombre), que resuelve los empates en ese orden porque las tuplas se comparan elemento a elemento. Si no ha llegado nadie, se añade un tramo ocioso hasta la siguiente llegada.

Como son no expropiativos, el elegido se ejecuta entero y basta con apuntar cuándo empieza y termina; de ahí salen la espera (inicio – llegada) y el retorno (fin – llegada). SJF da una espera media menor que FCFS porque los procesos cortos no se quedan atascados detrás de uno largo.

## 20. Round Robin con quantum

Difícil · Sistemas Operativos · [apuntesdam.com/subject/sistemas/topic/sistemas-operativos](https://apuntesdam.com/subject/sistemas/topic/sistemas-operativos)

Solución (python)

```
import sys
from collections import deque

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
quantum = int(lineas[0][1])
procesos = [(nombre, int(llegada), int(rafaga)) for nombre, llegada, rafaga in lineas[1:]]

llegadas = sorted(procesos, key=lambda p: p[1]) # sorted es estable: empates en orden de entrada
resto = {nombre: rafaga for nombre, _, rafaga in procesos}
fin = {}
cola = deque()
tramos = []
t = i = 0

def admitir():
 """Mete en la cola a los procesos que ya han llegado en el instante t."""
 global i
 while i < len(llegadas) and llegadas[i][1] <= t:
 cola.append(llegadas[i][0])
 i += 1

while i < len(llegadas) or cola:
 admitir()
 if not cola:
 # CPU ociosa hasta la siguiente llegada
 tramos.append(("ocioso", t, llegadas[i][1]))
 t = llegadas[i][1]
 continue
 nombre = cola.popleft()
 duracion = min(quantum, resto[nombre])
 tramos.append((nombre, t, t + duracion))
 t += duracion
 resto[nombre] -= duracion
 admitir() # los que llegan durante el tramo van antes...
 if resto[nombre] > 0:
 cola.append(nombre) # ...que el proceso expulsado
 else:
 fin[nombre] = t

print(" | ".join(f"{n} {a}-{b}" for n, a, b in tramos))
esperas, retornos = [], []
for nombre, llegada, rafaga in procesos:
 retornos.append(fin[nombre] - llegada)
 esperas.append(retornos[-1] - rafaga)
 print(f"{nombre}: espera {esperas[-1]}, retorno {retornos[-1]}")
print(f"Espera media: {sum(esperas) / len(esperas):.2f}")
print(f"Retorno medio: {sum(retornos) / len(retornos):.2f}")
print(f"Cambios de contexto: {sum(1 for a, b in zip(tramos, tramos[1:]) if a[0] != b[0])}")
```

Round Robin es una cola: el proceso del frente se ejecuta como mucho un quantum y, si le queda ráfaga, vuelve al final. El orden de las dos operaciones tras cada tramo es lo que decide el resultado: primero se admiten los procesos que han llegado durante el tramo (función `admitir`) y después se reencola el expulsado.

En Round Robin un proceso se ejecuta a trozos, así que la espera no es «inicio – llegada»: se calcula como el tiempo total en el sistema menos lo que de verdad usó la CPU. Los cambios de contexto se cuentan comparando cada tramo con el siguiente con `zip(tramos, tramos[1:])`; con un quantum pequeño hay muchos, y cada uno tiene un coste real para el sistema operativo.

## 21. Reemplazo de páginas: FIFO, LRU y óptimo

Muy difícil · Sistemas Operativos · [apuntesdam.com/subject/sistemas/topic/sistemas-operativos](https://apuntesdam.com/subject/sistemas/topic/sistemas-operativos)

Solución (python)

```
import sys
from collections import deque

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
marcos = int(lineas[0][1])
referencias = [int(x) for x in lineas[1]]

def fifo(refs):
 memoria, fallos = deque(), 0
 for p in refs:
 if p not in memoria:
 fallos += 1
 if len(memoria) == marcos:
 memoria.popleft() # sale la que lleva más tiempo cargada
 memoria.append(p)
 return fallos

def lru(refs):
 memoria, fallos = [], 0 # de la menos a la más recientemente usada
 for p in refs:
 if p in memoria:
 memoria.remove(p)
 else:
 fallos += 1
 if len(memoria) == marcos:
 memoria.pop(0) # sale la que hace más tiempo que no se usa
 memoria.append(p)
 return fallos

def optimo(refs):
 memoria, fallos = [], 0 # en orden de carga
 for i, p in enumerate(refs):
 if p in memoria:
 continue
 fallos += 1
 if len(memoria) == marcos:
 def proximo_uso(q):
 futuro = refs[i + 1:]
 return futuro.index(q) if q in futuro else float("inf")
 # la que tardará más en usarse; en empate (no vuelven), la cargada antes: max devuelve la primera
 memoria.remove(max(memoria, key=proximo_uso))
 memoria.append(p)
 return fallos

total = len(referencias)
for nombre, algoritmo in (("FIFO", fifo), ("LRU", lru), ("Óptimo", optimo)):
 f = algoritmo(referencias)
 print(f"{nombre}: {f} fallos ({f / total * 100:.1f} %)")
```

Los tres algoritmos solo se diferencian en cómo eligen la víctima cuando la memoria está llena. FIFO usa una cola (sale la primera que entró); LRU mantiene la lista ordenada por uso, moviendo al final cada página que se usa; el óptimo mira el futuro y expulsa la página cuyo próximo uso está más lejos (infinito si no vuelve a usarse).

El óptimo no se puede implementar en un sistema real, porque nadie conoce las referencias futuras, pero sirve de referencia para saber cuánto se aleja un algoritmo del mejor posible. La cadena 1 2 3 4 1 2 5... muestra la anomalía de Bélády: con FIFO, pasar de 3 a 4 marcos aumenta los fallos (de 9 a 10), algo que nunca le pasa a LRU.

## 22. Un plan de particiones alineado

Medio · Instalación de Sistemas Operativos · [apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-operativos](https://apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-operativos)

Solución (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
disco_mib = int(lineas[0][1].replace("GiB", "")) * 1024
ultimo = disco_mib - 1 # el último MiB es para la copia de la tabla GPT

def mib(tamano):
 if tamano.endswith("GiB"):
 return round(float(tamano[:-3]) * 1024)
 return int(tamano[:-3]) # «...MiB»

pos = 1 # el primer MiB es para la tabla y la alineación
for nombre, tamano in lineas[1:]:
 tam = ultimo - pos if tamano == "resto" else mib(tamano)
 if pos + tam > ultimo:
 print(f"ERROR: no cabe {nombre} (faltan {pos + tam - ultimo} MiB)")
 break
 print(f"{nombre}: inicio {pos} MiB, fin {pos + tam} MiB, tamaño {tam / 1024:.2f} GiB")
 pos += tam
else:
 print(f"Sin asignar: {(ultimo - pos) / 1024:.2f} GiB")
```

Trabajar en MiB enteros garantiza la alineación: cada partición empieza justo donde acaba la anterior, siempre en un múltiplo de 1 MiB, que es lo que hacen los instaladores para que las particiones coincidan con los bloques físicos del disco (sobre todo en SSD). El espacio útil va del MiB 1 al penúltimo, porque GPT reserva el principio y el final del disco.

«resto» ocupa lo que quede hasta el final útil. El `else` del `for` solo se ejecuta si el bucle termina sin `break`: así el resumen de espacio sin asignar no aparece cuando una partición no cabe.

## 23. ¿Arrancará este esquema de particiones?

Difícil · Instalación de Sistemas Operativos · [apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-operativos](https://apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-operativos)

Solución (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
config = {l[0]: l[1] for l in lineas[:3]} # firmware, tabla, disco
particiones = lineas[3:] # nombre clase sistema tamaño_GiB punto_de_montaje

def num(x):
 return f"{x:g}" # 109.0 → «109», 120.5 → «120.5»

uefi = config["firmware"] == "uefi"
mbr = config["tabla"] == "mbr"
disco = float(config["disco"])
clases = [p[1] for p in particiones]
problemas = []

if uefi and config["tabla"] != "gpt":
 problemas.append("UEFI necesita una tabla GPT (o activar el modo CSM)")
if mbr:
 if disco > 2048:
 problemas.append(f"MBR solo puede usar 2048 GiB de los {num(disco)} GiB del disco")
 if clases.count("primaria") + clases.count("extendida") > 4:
 problemas.append("MBR admite como máximo 4 particiones primarias o extendidas")
 if clases.count("extendida") > 1:
 problemas.append("Solo puede haber una partición extendida")
 if "logica" in clases and "extendida" not in clases:
 problemas.append("Las particiones lógicas necesitan una partición extendida")
elif "extendida" in clases or "logica" in clases:
 problemas.append("GPT no tiene particiones extendidas ni lógicas")
if uefi and not any(p[2] == "fat32" and p[4] == "/boot/efi" for p in particiones):
 problemas.append("Falta la partición EFI (FAT32 montada en /boot/efi)")
if not any(p[4] == "/" for p in particiones):
 problemas.append("Falta la partición raíz (/)")
ocupado = sum(float(p[3]) for p in particiones if p[1] != "extendida") # la extendida contiene a las lógicas
if ocupado > disco:
 problemas.append(f"Las particiones ocupan {num(ocupado)} GiB y el disco tiene {num(disco)} GiB")

print("\n".join(problemas) if problemas else "Esquema válido")
```

Las reglas se comprueban en orden y cada una que falla añade su mensaje a una lista; al final se muestran todos o «Esquema válido». Contar las clases con `clases.count(...)` resuelve casi todas las reglas de MBR: el límite de 4 entradas primarias o extendidas, una sola extendida y que las lógicas necesiten una extendida que las contenga.

En el espacio ocupado no se suma la extendida, porque es solo un contenedor: su tamaño ya es el de las lógicas que lleva dentro. El formato `:g` muestra los números sin decimales innecesarios (109 en vez de 109.0).

## 24. Leer la tabla de particiones de un MBR

Muy difícil · Instalación de Sistemas Operativos · [apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-](https://apuntesdam.com/subject/sistemas/topic/instalacion-sistemas-)

## Solución (python)

```
import sys

datos = bytes.fromhex("".join(sys.stdin.read().split())) # las 4 entradas (64 bytes) y la firma (2 bytes)
TIPOS = {0x07: "NTFS/exFAT", 0x0B: "FAT32", 0x0C: "FAT32 LBA", 0x83: "Linux", 0x82: "Linux swap",
 0x05: "Extendida", 0x0F: "Extendida LBA", 0xEE: "GPT de protección", 0xEF: "Sistema EFI"}

for n in range(4):
 entrada = datos[n * 16:(n + 1) * 16]
 tipo = entrada[4]
 if tipo == 0:
 continue # entrada vacía
 activa = "activa, " if entrada[0] == 0x80 else ""
 inicio = int.from_bytes(entrada[8:12], "little")
 sectores = int.from_bytes(entrada[12:16], "little")
 gib = sectores * 512 / 2**30
 print(f"Partición {n + 1}: {activa}tipo {tipo:02X} ({TIPOS.get(tipo, 'desconocido')}), "
 f"sector inicial {inicio}, {sectores} sectores ({gib:.2f} GiB)")
 if tipo == 0xEE:
 print("El disco usa GPT: esta entrada solo protege el disco de herramientas antiguas")

firma = datos[64:66]
print("Firma: correcta (55 AA)" if firma == b"\x55\xaa" else f"Firma: incorrecta ({firma[0]:02X} {firma[1]:02X})")
```

Cada entrada ocupa 16 bytes, así que la entrada  $n$  está en `datos[n*16:(n+1)*16]`, y dentro de ella cada campo tiene una posición fija. Indexar un objeto `bytes` da directamente el número (0x80, el tipo...), y `int.from_bytes(..., "little")` monta los enteros de 4 bytes leyendo el byte menos significativo primero, como lo guarda el PC.

Con 4 bytes se pueden contar como mucho  $2^{32} - 1$  sectores de 512 bytes, unos 2 TiB: esa es la razón del límite de MBR, y por eso los discos GPT ponen una única entrada de tipo EE que ocupa todo lo posible, para que las herramientas antiguas vean el disco lleno y no lo toquen.

## 25. Capacidad y tolerancia de un RAID

Medio · Sistemas de Ficheros y Copias de Seguridad · [apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento](http://apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento)

## Solución (python)

```
import sys

MINIMO = {"RAID0": 2, "RAID1": 2, "RAID5": 3, "RAID6": 4, "RAID10": 4}

for linea in sys.stdin.read().splitlines():
 if not linea.strip():
 continue
 nivel, n, tam = linea.split()
 n, tam = int(n), int(tam)
 if nivel not in MINIMO:
 print(f"{nivel}: nivel desconocido")
 continue
 if n < MINIMO[nivel] or (nivel == "RAID10" and n % 2):
 par = " y un número par" if nivel == "RAID10" else ""
 print(f"{nivel} con {n} discos: no válido (necesita al menos {MINIMO[nivel]} discos{par})")
 continue
 util = {"RAID0": n, "RAID1": 1, "RAID5": n - 1, "RAID6": n - 2, "RAID10": n // 2}[nivel] * tam
 if nivel == "RAID10":
 tolera = f"1 fallo por pareja (hasta {n // 2})"
 else:
 f = {"RAID0": 0, "RAID1": n - 1, "RAID5": 1, "RAID6": 2}[nivel]
 tolera = f"{f} {'fallo' if f == 1 else 'fallos'}"
 print(f"{nivel} con {n} x {tam} GB: {util} GB útiles ({util / (n * tam) * 100:.0f} %), tolera {tolera}")
```

Cada nivel se resume en dos números: cuántos discos de datos útiles quedan y cuántos fallos aguanta. Escribirlos en diccionarios indexados por el nivel evita una cadena de `if`. RAID5 dedica el equivalente a un disco a la paridad y RAID6 a dos, por eso toleran 1 y 2 fallos.

RAID10 es espejos (RAID1) unidos con bandas (RAID0): pierde la mitad de la capacidad y aguanta un fallo en cada pareja, pero si fallan los dos discos de la misma pareja se pierde todo. Ninguno sustituye a una copia de seguridad: un borrado accidental se replica en todos los discos al instante.

## 26. Simulador de estrategias de copia

Difícil · Sistemas de Ficheros y Copias de Seguridad · [apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento](http://apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento)

Solución (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
datos = float(lineas[0][1])
cambio = float(lineas[1][1])
plan = lineas[2][1:] # C = completa, I = incremental, D = diferencial, un día cada una

total = 0
ultima_completa = None
validas = [] # (día, tipo) de las copias que se han hecho
for dia, tipo in enumerate(plan, 1):
 if tipo != "C" and ultima_completa is None:
 print(f"Día {dia}: ERROR, una copia {tipo} necesita una completa anterior")
 continue
 if tipo == "C":
 tam = datos
 ultima_completa = dia
 elif tipo == "I":
 tam = datos * cambio / 100 # solo lo de un día
 else:
 tam = min(datos, datos * cambio / 100 * (dia - ultima_completa)) # todo desde la completa
 total += tam
 validas.append((dia, tipo))
 print(f"Día {dia} ({tipo}): {tam:.1f} GB")
print(f"Total ocupado: {total:.1f} GB")

Restaurar: última completa, última diferencial posterior y las incrementales que vengan detrás
c = max(d for d, t in validas if t == "C")
difs = [d for d, t in validas if t == "D" and d > c]
desde = difs[-1] if difs else c
necesarias = [c] + difs[-1:] + [d for d, t in validas if t == "I" and d > desde]
print("Para restaurar el último día: " + " ".join(f"día {d}" for d in necesarias))
```

Las tres estrategias se diferencian en desde cuándo copian los cambios: la incremental, desde la copia anterior (siempre un día de cambios); la diferencial, desde la última completa (cada día un poco más, hasta el total). Por eso las incrementales ocupan poco pero hay que aplicarlas todas para restaurar, y la diferencial ocupa más pero basta con la última.

Para saber qué copias hacen falta se parte de la última completa, se añade la última diferencial posterior (que ya incluye todo lo cambiado hasta ella) y después las incrementales que vengan detrás. `difs[-1:]` es una lista con la última diferencial, o vacía si no hay ninguna.

## 27. ¿Es buena esta copia de seguridad?

Muy difícil · Sistemas de Ficheros y Copias de Seguridad · [apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento](http://apuntesdam.com/subject/sistemas/topic/sistemas-ficheros-almacenamiento)

## almacenamiento

### Solución (python)

```
import sys

texto = sys.stdin.read().split("---")
original = [l.split() for l in texto[0].splitlines() if l.strip()] # ruta tamaño hash
copia = [l.split() for l in texto[1].splitlines() if l.strip()]

orig = {ruta: (int(tam), h) for ruta, tam, h in original}
cop = {ruta: (int(tam), h) for ruta, tam, h in copia}

movidos, faltan, distintos = [], [], []
sobran = [r for r in cop if r not in orig]
for ruta, (tam, h) in orig.items():
 if ruta in cop:
 if cop[ruta][1] != h:
 distintos.append(f"DISTINTO {ruta} ({tam} bytes en el original, {cop[ruta][0]} en la copia)")
 continue
 nueva = next((r for r in sobran if cop[r][1] == h), None) # ¿está en otra ruta con el mismo hash?
 if nueva:
 movidos.append(f"MOVIDO {ruta} -> {nueva}")
 sobran.remove(nueva)
 else:
 faltan.append(f"FALTA {ruta}")

for grupo in (movidos, faltan, distintos, [f"SOBRA {r}" for r in sobran]):
 for linea in sorted(grupo):
 print(linea)

correctos = sum(1 for r, (_, h) in orig.items() if r in cop and cop[r][1] == h)
mib = sum(t for t, _ in orig.values()) / 2**20
print(f"Correctos: {correctos} de {len(orig)} · Datos del original: {mib:.2f} MiB")
print("Copia VÁLIDA" if not faltan and not distintos else "Copia NO VÁLIDA")
```

Los dos manifiestos se convierten en diccionarios ruta → (tamaño, hash), y así comparar es buscar por clave. Lo que se compara es el hash, no el tamaño: un fichero puede tener el mismo tamaño y un contenido distinto (como enero.xlsx en el ejemplo), y solo el hash lo detecta.

Antes de declarar que un fichero falta se busca su hash entre los que sobran en la copia: si aparece, el fichero se ha movido o renombrado, pero sus datos están, y no invalida la copia. Al emparejarlo se quita de los sobrantes para que no salga dos veces.

## 28. ¿Llega este cable?

Medio · Redes Informáticas · [apuntesdam.com/subject/sistemas/topic/redes-informaticas](https://apuntesdam.com/subject/sistemas/topic/redes-informaticas)

Solución (python)

```
import sys

Distancia máxima en metros de cada medio para cada velocidad (Gbit/s)
MAXIMO = {
 "cat5e": {1: 100},
 "cat6": {1: 100, 10: 55},
 "cat6a": {1: 100, 10: 100},
 "cat8": {10: 30, 25: 30, 40: 30},
 "om3": {1: 550, 10: 300, 40: 100},
 "om4": {1: 550, 10: 400, 40: 150},
 "os2": {1: 10000, 10: 10000, 40: 10000},
}

for linea in sys.stdin.read().splitlines():
 if not linea.strip():
 continue
 medio, metros, velocidad = linea.split()
 if medio not in MAXIMO:
 print(f"{medio}: medio desconocido")
 continue
 metros, v = int(metros), int(velocidad.rstrip("G"))
 tramo = f"{medio}, {metros} m a {v}G"
 if v not in MAXIMO[medio]:
 inferiores = ", ".join(f"{x}G" for x in sorted(MAXIMO[medio]) if x < v) or "ninguna"
 print(f"{tramo}: NO (no admite {v}G; velocidades inferiores válidas: {inferiores})")
 elif metros <= MAXIMO[medio][v]:
 print(f"{tramo}: SÍ (máximo {MAXIMO[medio][v]} m)")
 else:
 sirven = ", ".join(sorted(m for m, d in MAXIMO.items() if d.get(v, 0) >= metros)) or "ninguno"
 print(f"{tramo}: NO (máximo {MAXIMO[medio][v]} m; sirve {sirven})")
```

Un diccionario de diccionarios responde directamente a la pregunta «¿hasta dónde llega este medio a esta velocidad?» con `MAXIMO[medio][v]`. Las alternativas salen de recorrer todos los medios con `d.get(v, 0)`, que da 0 a los que no admiten esa velocidad para que nunca cumplan.

`", ".join(...)` or `"ninguno"` usa que una cadena vacía es falsa: si la lista no tiene nada, se muestra el texto alternativo. El ejercicio refleja una decisión real al cablear: el cat6 sirve para 10 Gbit/s solo en tramos cortos, y a partir de unos 100 m hay que pasar a fibra.

## 29. La tabla MAC de un switch

Difícil · Redes Informáticas · [apuntesdam.com/subject/sistemas/topic/redes-informaticas](https://apuntesdam.com/subject/sistemas/topic/redes-informaticas)

Solución (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
puertos = int(lineas[0][1])
BROADCAST = "FF:FF:FF:FF:FF:FF"

tabla = {} # MAC → puerto
for puerto, origen, destino in lineas[1:]:
 p = int(puerto)
 if tabla.get(origen) != p: # nueva, o el equipo se ha movido de puerto
 tabla[origen] = p
 print(f"Aprende {origen} en el puerto {p}")
 otros = ", ".join(str(x) for x in range(1, puertos + 1) if x != p)
 if destino == BROADCAST:
 print(f"Difusión: inunda por {otros}")
 elif destino not in tabla:
 print(f"{destino} desconocido: inunda por {otros}")
 elif tabla[destino] == p:
 print(f"{destino} está en el mismo puerto: descarta la trama")
 else:
 print(f"Reenvía a {destino} por el puerto {tabla[destino]}")

print("Tabla MAC:")
for mac in sorted(tabla):
 print(f" {mac} → puerto {tabla[mac]}")
```

Un switch hace dos cosas con cada trama. Aprende: apunta en su tabla que la MAC de origen está en el puerto por el que ha entrado (y si ya estaba en otro, lo actualiza, porque el equipo se ha movido). Y reenvía: busca la MAC de destino en la tabla; si no la conoce, o es una difusión, la manda por todos los puertos menos el de entrada.

Si el destino está en el mismo puerto por el que llegó la trama (por ejemplo, detrás de un hub), no hace falta enviarla a ningún sitio: el destinatario ya la ha recibido. Por eso, tras unas pocas tramas, un switch deja de inundar la red y cada trama va solo a quien le corresponde.

### 30. Puntos únicos de fallo de una red

Muy difícil · Redes Informáticas · [apuntesdam.com/subject/sistemas/topic/redes-informaticas](https://apuntesdam.com/subject/sistemas/topic/redes-informaticas)

Solución (python)

```
import sys

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
raiz = lineas[0][1] # equipo desde el que se sale a Internet
enlaces = [(a, b) for _, a, b in lineas[1:]]

vecinos = {}
for a, b in enlaces:
 vecinos.setdefault(a, set()).add(b)
 vecinos.setdefault(b, set()).add(a)

def alcanzables(sin):
 """Equipos a los que se llega desde la raíz sin pasar por el equipo «sin»."""
 vistos, pila = {raiz}, [raiz]
 while pila:
 x = pila.pop()
 for v in vecinos[x]:
 if v != sin and v not in vistos:
 vistos.add(v)
 pila.append(v)
 return vistos

puntos = 0
for equipo in sorted(vecinos):
 if equipo == raiz or len(vecinos[equipo]) < 2:
 continue
 aislados = sorted(set(vecinos) - alcanzables(equipo) - {equipo})
 if aislados:
 puntos += 1
 print(f"Si falla {equipo}: sin red {'', '.join(aislados)}")
 else:
 print(f"Si falla {equipo}: nadie pierde la red (hay caminos alternativos)")
print(f"Puntos únicos de fallo: {puntos}")
```

La red es un grafo y la pregunta es de conectividad: para cada equipo de interconexión se recorre el grafo desde la raíz como si ese equipo no existiera, y los que no se alcanzan son los que se quedarían sin red. El recorrido es una búsqueda en profundidad con una pila y un conjunto de visitados.

Los equipos con un solo enlace (los PC) no se analizan porque su fallo no aísla a nadie más. En una estrella cada switch es un punto único de fallo; en un anillo, en cambio, siempre hay otro camino, y solo los equipos que cuelgan de un único switch dependen de él.

### 31. ¿Pueden hablar sin router?

Medio · Configuración de Redes TCP/IP · [apuntesdam.com/subject/sistemas/topic/configuracion-redes](https://apuntesdam.com/subject/sistemas/topic/configuracion-redes)

Solución (python)

```
import sys

def a_entero(ip):
 a, b, c, d = (int(x) for x in ip.split("."))
 return (a << 24) | (b << 16) | (c << 8) | d

def a_texto(n):
 return ".".join(str((n >> s) & 255) for s in (24, 16, 8, 0))

for linea in sys.stdin.read().splitlines():
 if not linea.strip():
 continue
 primero, segundo = linea.split()
 ip, prefijo = primero.split("/")
 prefijo = int(prefijo)
 mascara = (0xFFFFFFFF << (32 - prefijo)) & 0xFFFFFFFF # prefijo unos seguidos de ceros
 red1 = a_entero(ip) & mascara
 red2 = a_entero(segundo) & mascara
 if red1 == red2:
 print(f"{ip} y {segundo}: misma red {a_texto(red1)}/{prefijo}, comunicación directa")
 else:
 print(f"{ip} y {segundo}: redes distintas ({a_texto(red1)} y {a_texto(red2)}), hace falta un router")
```

La máscara de un prefijo /p son p unos seguidos de 32 – p ceros: desplazando 32 unos hacia la izquierda 32 - p posiciones y quedándose con 32 bits ( & 0xFFFFFFFF ) se obtiene sin tablas. El AND con la máscara borra la parte de host y deja la dirección de red.

Si las dos IP dan la misma red, el equipo puede entregar el paquete directamente; si no, se lo envía a su puerta de enlace. Por eso dos equipos con IP «parecidas» (192.168.1.10 y 192.168.1.70) pueden necesitar un router con una máscara /26.

## 32. Abreviar y expandir direcciones IPv6

Difícil · Configuración de Redes TCP/IP · [apuntesdam.com/subject/sistemas/topic/configuracion-redes](https://apuntesdam.com/subject/sistemas/topic/configuracion-redes)

Solución (python)

```
import sys

for linea in sys.stdin.read().splitlines():
 d = linea.strip()
 if not d:
 continue
 if "::" in d or len(d.split(":")) != 8:
 # Expandir: los grupos que faltan son los que representa «::»
 izq, _, der = d.partition("::")
 gi = [g for g in izq.split(":") if g]
 gd = [g for g in der.split(":") if g]
 grupos = gi + ["0"] * (8 - len(gi) - len(gd)) + gd
 resultado = ":".join(g.lower().zfill(4) for g in grupos)
 else:
 # Abreviar: sin ceros a la izquierda y «::» en la serie de ceros más larga
 grupos = [format(int(g, 16), "x") for g in d.split(":")]
 mejor, largo, i = -1, 0, 0
 while i < 8:
 if grupos[i] == "0":
 j = i
 while j < 8 and grupos[j] == "0":
 j += 1
 if j - i > largo:
 # con «>», en empate gana la primera
 mejor, largo = i, j - i
 i = j
 else:
 i += 1
 if largo >= 2:
 resultado = ":".join(grupos[:mejor]) + "::" + ":".join(grupos[mejor + largo:])
 else:
 resultado = ":".join(grupos)
 print(f"{d} → {resultado}")
```

Para expandir, `partition("::")` separa lo que hay a cada lado; los grupos que faltan hasta 8 son los ceros que representa «::», y `zfill(4)` rellena cada grupo con ceros a la izquierda. Para abreviar, `format(int(g, 16), "x")` quita los ceros a la izquierda (un grupo «0000» queda como «0») y pasa a minúsculas a la vez.

Luego se busca la serie de grupos a cero más larga recorriendo la lista. Las reglas de la RFC 5952 son las que fija el enunciado: «::» solo una vez (si no, no se sabría cuántos ceros hay en cada hueco), solo para dos o más grupos, y en empate la primera serie.

### 33. Calculadora VLSM

Muy difícil · Configuración de Redes TCP/IP · [apuntesdam.com/subject/sistemas/topic/configuracion-redes](https://apuntesdam.com/subject/sistemas/topic/configuracion-redes)

Solución (python)

```
import sys

def a_entero(ip):
 a, b, c, d = (int(x) for x in ip.split("."))
 return (a << 24) | (b << 16) | (c << 8) | d

def a_texto(n):
 return ".".join(str((n >> s) & 255) for s in (24, 16, 8, 0))

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]
base_ip, base_prefijo = lineas[0][1].split("/")
departamentos = [(nombre, int(hosts)) for nombre, hosts in lineas[1:]]

inicio = a_entero(base_ip)
fin_red = inicio + 2 ** (32 - int(base_prefijo))
pos = inicio
for nombre, hosts in sorted(departamentos, key=lambda d: (-d[1], d[0])):
 bits = 2
 while 2 ** bits - 2 < hosts: # la subred más pequeña que alcanza
 bits += 1
 tam = 2 ** bits
 if pos + tam > fin_red:
 print(f"{nombre}: no hay espacio para {hosts} hosts")
 continue
 print(f"{nombre} ({hosts}): {a_texto(pos)}/{32 - bits}, hosts {a_texto(pos + 1)} - {a_texto(pos + tam - 2)}, "
 f"difusión {a_texto(pos + tam - 1)}, sobran {tam - 2 - hosts}")
 pos += tam
libre = fin_red - pos
print(f"Libre: {libre} direcciones desde {a_texto(pos)}" if libre else "Libre: 0 direcciones")
```

Con las IP convertidas en enteros, repartir la red es sumar: cada subred empieza donde acaba la anterior. El tamaño de cada una es la menor potencia de 2 cuyos hosts útiles ( $2^{\text{bits}} - 2$ , por la dirección de red y la de difusión) alcanzan, y su prefijo es  $32 - \text{bits}$ .

Ordenar de mayor a menor no es un capricho: como todos los tamaños son potencias de 2 y van decreciendo, cada subred empieza siempre en un múltiplo de su tamaño, que es la condición para que sea una subred válida. Si se colocaran primero las pequeñas, una grande podría quedar desalineada.

### 34. Permisos efectivos de una carpeta compartida

Medio · Servicios de Red, Recursos Compartidos y Seguridad · [apuntesdam.com/subject/sistemas/topic/servicios-red-](https://apuntesdam.com/subject/sistemas/topic/servicios-red-)

## seguridad

### Solución (python)

```
import sys

NIVELES = ["ninguno", "lectura", "modificar", "control"]

lineas = [l.split() for l in sys.stdin.read().splitlines() if l.strip()]

compartido, ntfs, denegados, usuarios = {}, {}, set(), []
for l in lineas:
 if l[0] in ("compartido", "ntfs"):
 capa = compartido if l[0] == "compartido" else ntfs
 capa[l[1]] = max(capa.get(l[1], 0), NIVELES.index(l[2]))
 elif l[0] == "denegar":
 denegados.add(l[1])
 elif l[0] == "usuario":
 usuarios.append((l[1], l[2].split(",")))

for nombre, grupos in usuarios:
 c = max((compartido.get(g, 0) for g in grupos), default=0) # dentro de una capa, se suman
 if any(g in denegados for g in grupos): # la denegación gana a todo
 n = 0
 else:
 n = max((ntfs.get(g, 0) for g in grupos), default=0)
 efectivo = min(c, n) # entre capas, la más restrictiva
 print(f"{nombre}: compartido {NIVELES[c]}, NTFS {NIVELES[n]} → efectivo {NIVELES[efectivo]}")
```

Guardar los niveles como posiciones de la lista NIVELES convierte las reglas en operaciones con números: dentro de una capa, el usuario se queda con el mayor de sus grupos ( `max` ); entre las dos capas, con el menor ( `min` ). La denegación explícita se comprueba aparte, porque anula cualquier permiso que den los demás grupos.

Es la causa de muchas consultas de soporte: alguien tiene «control total» en NTFS y aun así solo puede leer, porque el recurso compartido solo da lectura. Lo habitual es dar control al compartido y ajustar los permisos con NTFS.

## 35. Un cortafuegos de primera coincidencia

Difícil · Servicios de Red, Recursos Compartidos y Seguridad · [apuntesdam.com/subject/sistemas/topic/servicios-red-](https://apuntesdam.com/subject/sistemas/topic/servicios-red-)

## seguridad

### Solución (python)

```
import sys

def a_entero(ip):
 a, b, c, d = (int(x) for x in ip.split("."))
 return (a << 24) | (b << 16) | (c << 8) | d

texto = sys.stdin.read().split("---")
reglas = [l.split() for l in texto[0].splitlines() if l.strip()] # la primera línea es «politica
allow|deny»
paquetes = [l.split() for l in texto[1].splitlines() if l.strip()]

politica = reglas[0][1]

def en_red(ip, red):
 if red == "any":
 return True
 base, _, prefijo = red.partition("/")
 prefijo = int(prefijo) if prefijo else 32 # una IP sola es un /32
 mascara = (0xFFFFFFFF << (32 - prefijo)) & 0xFFFFFFFF
 return a_entero(ip) & mascara == a_entero(base) & mascara

def coincide(regla, proto, ip, puerto):
 _, r_proto, r_puerto, _, r_red = regla
 return (r_proto in ("any", proto)
 and (r_puerto == "any" or int(r_puerto) == int(puerto))
 and en_red(ip, r_red))

for proto, ip, puerto in paquetes:
 for n, regla in enumerate(reglas[1:], 1):
 if coincide(regla, proto, ip, puerto):
 print(f"{proto} {ip} → {puerto}: {regla[0].upper()} (regla {n})")
 break
 else:
 print(f"{proto} {ip} → {puerto}: {politica.upper()} (política por defecto)")
```

Cada regla coincide si coinciden sus tres condiciones (protocolo, puerto y red de origen), y «any» funciona como comodín. La pertenencia a una red se comprueba como siempre en IP: aplicando la máscara a las dos direcciones y comparando; una IP sin prefijo se trata como /32 (solo ella).

El `for ... else` expresa justo la lógica del cortafuegos: se recorren las reglas en orden y la primera que coincide decide (`break`); el `else` solo se ejecuta si ninguna coincidió, y entonces se aplica la política por defecto. Por eso el orden importa tanto: una regla general colocada antes tapa a una más específica.

## 36. ¿Cuánto aguantaría esta contraseña?

Muy difícil · Servicios de Red, Recursos Compartidos y Seguridad · [apuntesdam.com/subject/sistemas/topic/servicios-](https://apuntesdam.com/subject/sistemas/topic/servicios-)

## Solución (python)

```

import sys
import math

DICCIONARIO = {"password", "123456", "qwerty", "admin", "iloveyou", "dragon", "monkey", "letmein",
 "futbol", "barcelona", "madrid", "contraseña", "hola", "princesa", "superman"}

UNIDADES = [("años", 365 * 86400), ("días", 86400), ("horas", 3600), ("minutos", 60)]

def tiempo(segundos):
 if segundos < 1:
 return "instantáneo"
 for nombre, s in UNIDADES:
 valor = segundos / s
 if valor >= 1:
 if valor >= 1_000_000:
 return f"más de un millón de {nombre}"
 if valor < 100:
 return f"{valor:.1f} {nombre}"
 return f"{round(valor):,} {nombre}".replace(",", ".") # punto de miles
 return "menos de 1 minuto"

for clave in sys.stdin.read().splitlines():
 if not clave:
 continue
 base = clave.lower()
 if base in DICCIONARIO or base.rstrip("0123456789.!") in DICCIONARIO:
 print(f"{clave}: muy débil (está en los diccionarios de contraseñas)")
 continue
 juego = (26 * any(c.islower() for c in clave) + 26 * any(c.isupper() for c in clave)
 + 10 * any(c.isdigit() for c in clave) + 33 * any(not c.isalnum() for c in clave))
 bits = len(clave) * math.log2(juego)
 if len(set(clave)) <= len(clave) / 3: # muchos caracteres repetidos
 bits /= 2
 nivel = "débil" if bits < 40 else "media" if bits < 60 else "fuerte" if bits < 80 else "muy fuerte"
 segundos = 2 ** bits / 2 / 1e10
 print(f"{clave}: {nivel} ({bits:.1f} bits, {tiempo(segundos)} por fuerza bruta)")

```

La entropía mide cuántas contraseñas posibles hay con esa forma: si se usan caracteres de un juego de  $N$  símbolos, cada carácter aporta  $\log_2(N)$  bits. `any(...)` devuelve un booleano que, multiplicado por el tamaño de cada grupo, suma solo los grupos que aparecen. Un atacante prueba de media la mitad de las  $2^{\text{bits}}$  combinaciones, de ahí el `/ 2`.

Pero la entropía solo vale para contraseñas aleatorias: «Barcelona2024!» parece fuerte por su longitud y variedad, y cae en segundos porque los atacantes prueban primero palabras conocidas con números al final. Por eso se comprueba el diccionario antes que nada. La conclusión práctica sale del último ejemplo: cuatro palabras al azar son más fuertes y más fáciles de recordar que una clave corta llena de símbolos.

La función `tiempo` busca la mayor unidad en la que el valor llega a 1 y aplica las reglas de formato; `f"{n:,}"` pone comas de miles, que se cambian por puntos.

## 37. Informe del inventario

Medio · Documentación y Aplicaciones Ofimáticas · [apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica](https://apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica)

Solución (python)

```
import sys
import csv
from datetime import date

lineas = sys.stdin.read().splitlines()
hoy = date.fromisoformat(lineas[0].split("=")[1])
filas = list(csv.DictReader(lineas[1:], delimiter=";")) # codigo;tipo;aula;precio;fin_garantia

aulas = {}
for f in filas:
 aulas.setdefault(f["aula"], []).append(f)

total_equipos, total_valor, total_vencidos = 0, 0.0, 0
for aula in sorted(aulas):
 equipos = aulas[aula]
 valor = sum(float(e["precio"]) for e in equipos)
 vencidos = sorted(e["codigo"] for e in equipos if date.fromisoformat(e["fin_garantia"]) < hoy)
 linea = f"{aula}: {len(equipos)} {'equipo' if len(equipos) == 1 else 'equipos'}, {valor:.2f} €"
 if vencidos:
 linea += ", sin garantía: " + ", ".join(vencidos)
 print(linea)
 total_equipos += len(equipos)
 total_valor += valor
 total_vencidos += len(vencidos)

porcentaje = total_vencidos / total_equipos * 100
print(f"TOTAL: {total_equipos} equipos, {total_valor:.2f} €, {total_vencidos} sin garantía
({porcentaje:.0f} %)")
```

`csv.DictReader` lee el CSV y da cada fila como un diccionario con los nombres de la cabecera, así el código habla de `e["precio"]` en lugar de posiciones. Agrupar es crear un diccionario aula → lista de equipos con `setdefault`, y después cada aula se resume con `sum` y una comprensión.

Las fechas se comparan como objetos `date`, no como texto: así «antes de hoy» funciona sin trucos. Es exactamente lo que hace una tabla dinámica (agrupar por aula y sumar), pero repetible: el mismo programa sirve para el inventario del año que viene.

### 38. Un índice automático para la documentación

Difícil · Documentación y Aplicaciones Ofimáticas · [apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica](https://apuntesdam.com/subject/sistemas/topic/documentacion-ofimatica)

Solución (python)

```
import sys
import re
import unicodedata

lineas = sys.stdin.read().splitlines()

def ancla(titulo):
 s = unicodedata.normalize("NFD", titulo.lower())
 s = "".join(c for c in s if unicodedata.category(c) != "Mn") # quita las tildes
 s = re.sub(r"^[a-z0-9 -]", "", s)
 return s.replace(" ", "-")

usadas = {}
en_codigo = False
n2 = n3 = 0
for linea in lineas:
 if linea.startswith("```"):
 en_codigo = not en_codigo
 continue
 if en_codigo:
 continue
 m = re.match(r"^(#{2,3})\s+(.*)$", linea)
 if not m:
 continue
 titulo = m.group(2).strip()
 a = ancla(titulo)
 if a in usadas:
 usadas[a] += 1
 a = f"{a}-{usadas[a]}" # la segunda vez -1, la tercera -2...
 else:
 usadas[a] = 0
 if len(m.group(1)) == 2:
 n2 += 1
 n3 = 0
 print(f"{n2}. [{titulo}](#{a})")
 else:
 n3 += 1
 print(f" {n2}.{n3}. [{titulo}](#{a})")
```

El ancla se construye en cuatro pasos: minúsculas, quitar tildes (la normalización NFD separa cada letra de su tilde, y las marcas, de categoría «Mn», se descartan), eliminar lo que no sea letra, número, espacio o guion, y cambiar los espacios por guiones. Es el mismo algoritmo que usan GitHub y la mayoría de generadores de documentación.

Un diccionario cuenta cuántas veces ha salido cada ancla para añadir el sufijo -1, -2... a las repetidas. La numeración usa dos contadores, y el de nivel 3 se reinicia cada vez que aparece un título de nivel 2. Las líneas dentro de bloques de código se saltan con una variable que cambia de estado en cada «```».

### 39. Un conversor de Markdown a HTML

Muy difícil · Documentación y Aplicaciones Ofimáticas · [apuntesdam.com/subject/sistemas/topic/documentacion-](https://apuntesdam.com/subject/sistemas/topic/documentacion-)

## Solución (python)

```

import sys
import re
import html

lineas = sys.stdin.read().splitlines()

def en_linea(texto):
 """Formato dentro de una línea; el código se aparta primero para que no se interprete."""
 codigos = []
 def guardar(m):
 codigos.append(m.group(1))
 return f"\x00{len(codigos) - 1}\x00"
 texto = re.sub(r"``([^\`]+)`", guardar, texto)
 texto = re.sub(r"\s**(.+?)\s*", r"\1", texto)
 texto = re.sub(r"*(.+?)\s*", r"\1", texto)
 texto = re.sub(r"\[[^\]]+\]\s*\([^\)]+\)", r"\1", texto)
 return re.sub(r"\x00(\d+)\x00", lambda m: f"<code>{codigos[int(m.group(1))]}</code>", texto)

salida, parrafo, codigo = [], [], []
lista = None # "ul", "ol" o None
en_codigo = False

def cerrar_parrafo():
 if parrafo:
 salida.append("<p>" + en_linea(" ".join(parrafo)) + "</p>")
 parrafo.clear()

def cerrar_lista():
 global lista
 if lista:
 salida.append(f"</{lista}>")
 lista = None

for linea in lineas:
 if en_codigo:
 if linea.startswith("```"):
 salida.append("<pre><code>" + "\n".join(codigo) + "</code></pre>")
 en_codigo = False
 else:
 codigo.append(html.escape(linea, quote=False))
 continue
 if linea.startswith("```"):
 cerrar_parrafo(); cerrar_lista()
 en_codigo, codigo = True, []
 continue
 texto = html.escape(linea, quote=False)
 titulo = re.match(r"^(#{1,6})\s+(.*)$", texto)
 elemento = re.match(r"^(?:([-*])|\d+\.)\s+(.*)$", texto)
 if not texto.strip():
 cerrar_parrafo(); cerrar_lista()
 elif titulo:
 cerrar_parrafo(); cerrar_lista()
 n = len(titulo.group(1))
 salida.append(f"<h{n}>{en_linea(titulo.group(2))}</h{n}>")
 elif elemento:
 cerrar_parrafo()
 tipo = "ul" if elemento.group(1) else "ol"
 if lista != tipo:
 cerrar_lista()
 salida.append(f"<{tipo}>")
 lista = tipo
 salida.append(f"{en_linea(elemento.group(2))}")
 else:
 cerrar_lista()
 parrafo.append(texto)
 cerrar_parrafo(); cerrar_lista()
if en_codigo:
 salida.append("<pre><code>" + "\n".join(codigo) + "</code></pre>")

```

```
print("\n".join(salida))
```

El conversor trabaja por bloques con un pequeño estado: si estamos en un párrafo (las líneas acumuladas), en qué lista (ul, ol o ninguna) y si estamos dentro de un bloque de código. Cada línea decide qué bloque abre o cierra; las funciones `cerrar_parrafo` y `cerrar_lista` se llaman siempre que cambia el tipo de bloque, y una vez más al final.

Lo primero con cada línea es escapar `<`, `>` y `&`, para que el texto del documento no se convierta en HTML de verdad. El formato en línea se aplica con expresiones regulares en orden (`**` antes que `*`, para que la negrita no se lea como dos cursivas), y el código en línea se aparta antes con un marcador para que un `*` dentro de él no se interprete.

## 40. El embudo de ventas del CRM

Medio · Sistemas ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/sistemas-erp-crm](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/sistemas-erp-crm)

Solución (python)

```
import sys

ETAPAS = [("Nuevo", 10), ("Calificado", 30), ("Propuesta", 50), ("Negociación", 75), ("Ganado", 100),
 ("Perdido", 0)]

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

cuenta = {etapa: 0 for etapa, _ in ETAPAS}
importe = {etapa: 0.0 for etapa, _ in ETAPAS}
for linea in lineas:
 _, etapa, valor = linea.split(";")
 cuenta[etapa] += 1
 importe[etapa] += float(valor)

prevision = 0.0
for etapa, probabilidad in ETAPAS:
 print(f"{etapa}: {cuenta[etapa]} oportunidades, {importe[etapa]:.2f} €")
 if etapa not in ("Ganado", "Perdido"): # solo las abiertas
 prevision += importe[etapa] * probabilidad / 100
print(f"Previsión ponderada: {prevision:.2f} €")

cerradas = cuenta["Ganado"] + cuenta["Perdido"]
if cerradas:
 print(f"Tasa de conversión: {cuenta['Ganado'] / cerradas * 100:.1f} %")
else:
 print("Tasa de conversión: -")
```

Los dos diccionarios se inicializan con todas las etapas a cero, así se muestran también las que no tienen oportunidades y en el orden del embudo. Al recorrer las etapas se acumula a la vez la previsión ponderada, que cuenta cada oportunidad abierta por su importe multiplicado por la probabilidad de cerrarla.

La previsión ponderada es la cifra que un director comercial usa para estimar los ingresos: 12 000 € en «Propuesta» valen, en promedio, 6000 €. La tasa de conversión solo tiene sentido con oportunidades cerradas; sin ninguna, dividir daría error, por eso se comprueba antes.

## 41. Coste total de propiedad a varios años

Difícil · Sistemas ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/sistemas-erp-crm](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/sistemas-erp-crm)

Solución (python)

```
import sys

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

años = int(lineas[0].split()[1])
opciones = []
for linea in lineas[1:]:
 nombre, licencia, usuarios, implantacion, formacion, hardware, mantenimiento = linea.split(";")
 tco = (float(licencia) * int(usuarios) * 12 * años + float(implantacion) + float(formacion)
 + float(hardware) + float(mantenimiento) * años)
 opciones.append((tco, nombre))

opciones.sort() # por TCO y, a igualdad, por nombre
for tco, nombre in opciones:
 print(f"{nombre}: {tco:.2f} €")
print(f"Recomendada: {opciones[0][1]}")
if len(opciones) > 1:
 barata, cara = opciones[0][0], opciones[-1][0]
 print(f"Ahorro frente a la más cara: {cara - barata:.2f} € ({(cara - barata) / cara * 100:.1f} %)")
```

Guardar cada opción como una tupla (TCO, nombre) hace que `sort()` ordene justo como pide el enunciado: primero por el coste y, si empatan, por el nombre, porque las tuplas se comparan elemento a elemento. La más barata queda la primera y la más cara la última.

El ejercicio muestra por qué se compara el coste total y no el precio de la licencia: una opción «gratuita» (Odoo Community) puede salir más cara a cinco años por la implantación, el hardware y el mantenimiento, que es lo que mide el TCO.

## 42. Dimensionar servidores para Odoo

Muy difícil · Sistemas ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/sistemas-erp-crm](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/sistemas-erp-crm)

Solución (python)

```
import sys
import math

CATALOGO = [("S", 2, 4), ("M", 4, 8), ("L", 8, 16), ("XL", 16, 32)]

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

for linea in lineas:
 nombre, concurrentes, nucleos, ram = linea.split(";")
 nucleos, ram_mb = int(nucleos), int(ram) * 1024
 workers = max(2, math.ceil(int(concurrentes) / 6))
 nucleos_necesarios = math.ceil(workers / 2)
 memoria_necesaria = (workers + 1) * 250 + 2048 + 1024 # workers + cron, PostgreSQL y sistema

 print(f"== {nombre}")
 print(f"workers = {workers}")
 print("max_cron_threads = 1")
 if nucleos >= nucleos_necesarios:
 print("CPU: suficiente")
 else:
 print(f"CPU: insuficiente (faltan {nucleos_necesarios - nucleos} núcleos)")
 if ram_mb >= memoria_necesaria:
 print("Memoria: suficiente")
 else:
 print(f"Memoria: insuficiente (faltan {memoria_necesaria - ram_mb} MB)")
 if nucleos < nucleos_necesarios or ram_mb < memoria_necesaria:
 servidor = next((n for n, c, g in CATALOGO if c >= nucleos_necesarios and g * 1024 >=
 memoria_necesaria), None)
 print(f"Recomendado: {servidor or 'varios servidores'}")
```

Las reglas son las que recomienda la documentación de Odoo para el modo multiproceso: un worker atiende unos 6 usuarios concurrentes, cada núcleo puede llevar unos 2 workers y cada worker necesita su memoria, a la que se suman PostgreSQL y el sistema. `math.ceil` redondea hacia arriba, porque no se puede tener medio worker.

Para recomendar un servidor, `next()` recorre el catálogo de menor a mayor y devuelve el primero que cubre a la vez CPU y memoria, o `None` si ninguno llega; en ese caso la solución ya no es un servidor más grande, sino repartir la carga entre varios.

### 43. ¿Puede este usuario hacerlo?

Medio · Implantación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/implantacion-erp](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/implantacion-erp)

Solución (python)

```
import sys

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]
LETRA = {"leer": "r", "escribir": "w", "crear": "c", "borrar": "d"}

accesos = {} # (grupo, modelo) → letras de permiso
usuarios = {} # usuario → grupos
seccion = ""
for linea in lineas:
 if linea.startswith("["):
 seccion = linea
 continue
 p = linea.split()
 if seccion == "[accesos]":
 grupo, modelo, permisos = p
 clave = (grupo, modelo)
 accesos[clave] = accesos.get(clave, set()) | set(permisos.replace("-", "")) # varias filas se suman
 elif seccion == "[usuarios]":
 usuarios[p[0]] = p[1].split(",")
 elif seccion == "[consultas]":
 usuario, accion, modelo = p
 permisos = set()
 for grupo in usuarios.get(usuario, []): # un usuario desconocido no tiene grupos
 permisos |= accesos.get((grupo, modelo), set())
 estado = "PERMITIDO" if LETRA[accion] in permisos else "DENEGADO"
 print(f"{usuario} {accion} {modelo}: {estado}")
```

Los permisos de cada (grupo, modelo) se guardan como un conjunto de letras, y la unión de conjuntos ( `|` ) expresa exactamente la regla de Odoo: los permisos se suman, tanto entre varias filas del mismo grupo como entre los grupos de un usuario. «-» se quita antes, porque no es ningún permiso.

La consulta solo tiene que reunir los permisos de todos los grupos del usuario sobre ese modelo y comprobar si contiene la letra de la acción. Lo que no se ha concedido explícitamente está denegado, y un usuario que no existe no tiene grupos.

## 44. Limpiar los clientes antes de migrarlos

Difícil · Implantación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/implantacion-erp](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/implantacion-erp)

Solución (python)

```
import sys
import re

TABLA = "TRWAGMYFPDXBNJZSQVHLCKE"

lineas = [l for l in sys.stdin.read().splitlines() if l.strip()]

def nif_valido(nif):
 m = re.fullmatch(r"([XYZ]\d{7}|\d{8})([A-Z])", nif)
 if not m:
 return False
 numero = int(str("XYZ".index(m.group(1)[0])) + m.group(1)[1:]) if m.group(1)[0] in "XYZ" else m.group(1)
 return TABLA[numero % 23] == m.group(2)

validos = errores = duplicados = 0
vistos = {} # NIF → línea donde apareció válido
for n, linea in enumerate(lineas[1:], start=2):
 nombre, nif, email, telefono = linea.split(";")
 # Normalizar
 nombre = " ".join(nombre.split())
 nif = nif.upper().replace("-", "").replace(" ", "")
 email = email.strip().lower()
 telefono = re.sub(r"[.-]", "", telefono.strip())
 for prefijo in ("+34", "0034"):
 if telefono.startswith(prefijo):
 telefono = telefono[len(prefijo):]
 # Validar
 error = None
 if not nif_valido(nif):
 error = "NIF no válido"
 elif not re.fullmatch(r"^[^\s]+@[^\s]+\.[A-Za-z]{2,}$", email):
 error = "correo no válido"
 elif not re.fullmatch(r"[6789]\d{8}", telefono):
 error = "teléfono no válido"
 if error:
 print(f"ERROR línea {n}: {error}")
 errores += 1
 elif nif in vistos:
 print(f"DUPLICADO línea {n}: {nif} (ya en la línea {vistos[nif]})")
 duplicados += 1
 else:
 vistos[nif] = n
 print(f"OK {nif};{nombre};{email};{telefono}")
 validos += 1
print(f"Válidos: {validos} · Errores: {errores} · Duplicados: {duplicados}")
```

El orden normalizar → validar → detectar duplicados es la clave de una migración: «12345678z» y «12345678-Z» son el mismo cliente, y solo se ve después de normalizarlos. `" ".join(nombre.split())` quita los espacios de los extremos y deja uno solo entre palabras, y el teléfono se limpia de separadores y del prefijo internacional.

Las validaciones usan expresiones regulares con `re.fullmatch`, que exige que encaje la cadena entera. En el NIE, la letra inicial se sustituye por su cifra (X=0, Y=1, Z=2) antes de calcular el resto entre 23. Solo los registros válidos se apuntan en `vistos`, así un error no puede provocar un falso duplicado.

## 45. Rotación de copias abuelo-padre-hijo

Muy difícil · Implantación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/implantacion-erp](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/implantacion-erp)

## Solución (python)

```
import sys
from datetime import date

fechas = [date.fromisoformat(l.strip()) for l in sys.stdin.read().split() if l.strip()]

copias = sorted(set(fechas), reverse=True) # sin repetir, de la más reciente a la más antigua

diaria = set(copias[:7])
semanal, mensual = set(), set()
semanas, meses = [], []
for f in copias:
 semana = f.isocalendar()[2] # la primera de cada periodo es la más reciente
 # (año ISO, semana ISO)
 if semana not in semanas:
 semanas.append(semana)
 if len(semanas) <= 4:
 semanal.add(f)
 mes = (f.year, f.month)
 if mes not in meses:
 meses.append(mes)
 if len(meses) <= 12:
 mensual.add(f)

conservadas = 0
for f in copias:
 motivos = [m for m, grupo in (("diaria", diaria), ("semanal", semanal), ("mensual", mensual)) if f in grupo]
 if motivos:
 conservadas += 1
 print(f"{f} conservar ({', '.join(motivos)})")
 else:
 print(f"{f} borrar")
print(f"Conservadas: {conservadas} · Borradas: {len(copias) - conservadas}")
```

Ordenando las fechas de la más reciente a la más antigua, la primera copia que aparece de cada semana o de cada mes es justo la más reciente de ese periodo. Basta con ir anotando los periodos que ya se han visto: los 4 primeros son las semanas que se conservan y los 12 primeros, los meses. Las diarias son simplemente las 7 primeras.

La semana ISO se obtiene con `isocalendar()`, y se usa el par (año ISO, semana) porque el número de semana solo no distingue años. Con esta rotación, unas 23 copias cubren un año entero: mucho detalle reciente y cada vez menos hacia atrás.

## 46. Encuentra el fallo: el descuento para casi todos

Fácil · Adaptación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp)

## Solución (python)

```
cantidad = int(input())
if cantidad >= 10:
 print("Descuento del 5 %")
else:
 print("Sin descuento")
```

`input()` devuelve siempre texto, y comparar textos es compararlos en orden alfabético, carácter a carácter: «9» es mayor que «10» porque «9» va después de «1». Convirtiendo la entrada a entero con `int()`, la comparación es numérica.

## 47. Encuentra el fallo: el ranking que no existe

Fácil · Adaptación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp)

Solución (python)

```
clientes = [("Carlos", 558.0), ("Ana", 1079.86), ("Javier", 165.4), ("Lucía", 297.0)]
ranking = sorted(clientes, key=lambda c: c[1], reverse=True)
for posicion, (nombre, importe) in enumerate(ranking, start=1):
 print(f"{posicion}. {nombre}: {importe:.2f} €")
```

`lista.sort()` ordena la lista en el sitio y devuelve `None`, así que `ranking` valía `None` y el `for` fallaba al intentar recorrerlo. `sorted()` hace lo mismo pero devuelve una lista nueva, que es lo que se quería guardar.

También valdría llamar a `clientes.sort(...)` en una línea y recorrer después `clientes`.

## 48. Encuentra el fallo: el stock que aparece en todos los almacenes

Medio · Adaptación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp)

Solución (python)

```
almacenes, productos = 2, 3
stock = [[0] * productos for _ in range(almacenes)]
stock[0][1] = 5
for i, fila in enumerate(stock):
 print(f"Almacén {i}: {fila}")
```

`[[0] * productos] * almacenes` no crea dos filas: crea una lista con dos referencias a la misma fila. Cambiar `stock[0][1]` cambia esa única fila, y se ve desde las dos posiciones.

Con una comprensión, `[[0] * productos for _ in range(almacenes)]`, la expresión `[0] * productos` se evalúa una vez por cada almacén y cada uno tiene su propia lista.

## 49. Un evaluador de dominios de Odoo

Muy difícil · Adaptación de un ERP-CRM · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/adaptacion-erp)

Solución (python)

```
import sys
import ast
import json

lineas = [l for l in sys.stdin.read().splitlines() if l.strip()]
dominio = ast.literal_eval(lineas[0])
registros = [json.loads(l) for l in lineas[1:]]

def cumple(condicion, registro):
 campo, op, valor = condicion
 v = registro.get(campo) # un campo que falta vale None
 if op in ("=", "!="):
 igual = (v is None or v is False) if valor is False else v == valor
 return igual if op == "=" else not igual
 if op == "in":
 return v in valor
 if op == "not in":
 return v not in valor
 if v is None:
 return False
 if op == "ilike":
 return str(valor).lower() in str(v).lower()
 return {">": v > valor, ">=": v >= valor, "<": v < valor, "<=": v <= valor}[op]

def evalua(i, registro):
 """Evalúa la expresión que empieza en la posición i; devuelve (resultado, siguiente posición)."""
 t = dominio[i]
 if t in ("&", "|"):
 a, i = evalua(i + 1, registro)
 b, i = evalua(i, registro)
 return (a and b) if t == "&" else (a or b), i
 if t == "!":
 a, i = evalua(i + 1, registro)
 return not a, i
 return cumple(t, registro), i + 1

ids = []
for r in registros:
 i, resultado = 0, True
 while i < len(dominio): # los términos del nivel superior se unen con Y
 valor, i = evalua(i, r)
 resultado = resultado and valor
 if resultado:
 ids.append(str(r["id"]))
print(" ".join(ids) or "ninguno")
```

La notación prefija se evalúa con una función recursiva que recibe la posición donde empieza una expresión y devuelve su valor y dónde termina: un operador «&» o «|» evalúa las dos expresiones siguientes (la segunda empieza donde acabó la primera), «!» niega la siguiente, y una tupla es una condición simple. Así, «& A | B ! C» se lee como A y (B o no C) sin paréntesis.

`cumple` implementa cada operador, con los detalles de Odoo: un campo ausente vale None, «= False» encuentra los campos vacíos y las comparaciones con None no se cumplen. `ast.literal_eval` convierte el texto del dominio en una lista de Python de forma segura, sin ejecutar código.

## 50. Encuentra el fallo: el recuento de ventas que se rompe

Fácil · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-)

## componentes

### Solución (python)

```
lineas = ["Portátil", "Ratón", "Ratón", "Webcam", "Ratón", "Portátil"]
ventas = {}
for producto in lineas:
 ventas[producto] = ventas.get(producto, 0) + 1
for producto, unidades in sorted(ventas.items()):
 print(producto, unidades)
```

`ventas[producto] += 1` necesita leer primero `ventas[producto]`, y la primera vez que aparece un producto esa clave no existe: salta un `KeyError`. `ventas.get(producto, 0)` devuelve 0 si la clave no está, así el primer recuento parte de cero.

Otras opciones son `collections.Counter(lineas)`, que cuenta directamente, o un `defaultdict(int)`.

## 51. Un campo calculado: la amortización

Medio · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

### Solución (python)

```
import sys
from datetime import date

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]
hoy = date.fromisoformat(lineas[0])

for linea in lineas[1:]:
 nombre, compra, vida, precio = linea.split(";")
 dias = (hoy - date.fromisoformat(compra)).days
 if dias < 0:
 print(f"{nombre}: fecha de compra futura")
 continue
 anios = dias / 365
 valor = max(0.0, float(precio) * (1 - anios / float(vida))) # nunca por debajo de 0
 estado = "amortizado" if valor == 0 else "en uso"
 print(f"{nombre}: {valor:.2f} € ({estado}, {anios:.1f} años)")
```

Restar dos fechas da un `timedelta`, y su atributo `.days` son los días transcurridos. Con amortización lineal el equipo pierde cada año la misma fracción de su precio, y `max(0.0, ...)` evita valores negativos cuando ya ha superado su vida útil.

Es lo mismo que haría un campo calculado de Odoo: una función que se recalcula cuando cambian los campos de los que depende (fecha de compra, vida útil y precio).

## 52. Encuentra el fallo: las líneas que se comparten

Medio · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

## componentes

### Solución (python)

```
class Pedido:
 def __init__(self, cliente, lineas=None):
 self.cliente = cliente
 self.lineas = [] if lineas is None else lineas

 def anadir(self, producto, cantidad):
 self.lineas.append((producto, cantidad))

p1 = Pedido("Ana")
p1.anadir("Portátil", 1)
p2 = Pedido("Luis")
p2.anadir("Ratón", 2)
print(p1.cliente, len(p1.lineas))
print(p2.cliente, len(p2.lineas))
```

Los valores por defecto de los parámetros se crean una sola vez, al definir la función, no en cada llamada. Con `lineas=[]`, todos los pedidos creados sin líneas comparten la misma lista, y lo que añade uno lo ven todos.

La forma correcta es usar `None` como valor por defecto y crear la lista vacía dentro del método, de modo que cada pedido tenga la suya.

## 53. Encuentra el fallo: los agotados que se escapan

Medio · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

### Solución (python)

```
productos = [
 ["Auriculares", 0], ["Webcam", 0], ["Ratón", 120], ["Altavoz", 0], ["Teclado", 45]
]
productos = [p for p in productos if p[1] != 0]
print([p[0] for p in productos])
```

Borrar elementos de una lista mientras se recorre desplaza los que quedan una posición a la izquierda, y el bucle se salta el siguiente: al quitar los auriculares, la webcam pasa a su posición y el bucle continúa por la siguiente, sin revisarla.

La solución idiomática es crear una lista nueva solo con los elementos que se quieren conservar, con una comprensión.

## 54. Encuentra el fallo: el pedido confirmado que no se factura

Medio · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

### Solución (python)

```
estado = input().strip()
if estado == "confirmado":
 print("Pedido confirmado: se genera la factura")
else:
 print(f"El pedido está en estado {estado}: todavía no se factura")
```

`is` comprueba si dos variables son el mismo objeto en memoria, no si tienen el mismo valor. El texto leído con `input()` es un objeto nuevo, distinto del literal "confirmado", aunque contenga las mismas letras. Para comparar valores se usa `==`; `is` se reserva para `None`.

## 55. Restricciones de un modelo

Difícil · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

## componentes

### Solución (python)

```
import sys
from datetime import date

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]
hoy = date.fromisoformat(lineas[0])
TIPOS = ("preventivo", "correctivo")

guardados = set() # (equipo, fecha) de los registros guardados
for n, linea in enumerate(lineas[1:], start=1):
 equipo, fecha, coste, tipo = linea.split(";")
 if float(coste) < 0:
 error = "el coste no puede ser negativo"
 elif tipo not in TIPOS:
 error = f"tipo «{tipo}» no permitido"
 elif date.fromisoformat(fecha) > hoy:
 error = "la fecha no puede ser futura"
 elif (equipo, fecha) in guardados:
 error = f"{equipo} ya tiene un mantenimiento el {fecha}"
 else:
 guardados.add((equipo, fecha)) # solo se guarda si pasa todas las restricciones
 continue
 print(f"ValidationError en el registro {n}: {error}")
print(f"Guardados: {len(guardados)}")
```

Las restricciones se comprueban en orden con `if/elif` y la primera que falla da el mensaje, igual que cuando un `@api.constrains` lanza `ValidationError` y el registro no llega a guardarse. Si ninguna falla, el registro se guarda y `continue` salta el mensaje de error.

La restricción única se comprueba contra un conjunto de pares (equipo, fecha) con solo los registros guardados; por eso un registro rechazado por otro motivo no impide guardar después otro con el mismo equipo y fecha.

## 56. Secuencias de referencias

Muy difícil · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

### Solución (python)

```
import sys
from datetime import date

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]
patron = lineas[0].split("=", 1)[1]
relleno = int(lineas[1].split("= ")[1])
reinicio = lineas[2].split("= ")[1]

contadores = {} # periodo → último número usado
for texto in lineas[3:]:
 f = date.fromisoformat(texto)
 if reinicio == "anual":
 periodo = f.year
 elif reinicio == "mensual":
 periodo = (f.year, f.month)
 else:
 periodo = None # un único contador
 contadores[periodo] = contadores.get(periodo, 0) + 1
 prefijo = patron % {"year": f"{f.year:04d}", "y": f"{f.year % 100:02d}",
 "month": f"{f.month:02d}", "day": f"{f.day:02d}"}
 print(f"{texto} → {prefijo}{str(contadores[periodo]).zfill(relleno)}")
```

Cada periodo tiene su propio contador en un diccionario: la clave es el año, el par (año, mes) o `None` si nunca se reinicia. Así «reiniciar» no es poner a cero, sino empezar a usar otra clave, y volver a un periodo anterior continúa su numeración donde se quedó.

El prefijo usa la misma sintaxis que Odoo, `%(year)s`, que es el formato `%` de Python con un diccionario: `patron % {...}` sustituye cada marcador por su valor. `zfill` rellena el número con ceros a la izquierda.

## 57. Encuentra el fallo: los céntimos que se pierden

Muy difícil · Desarrollo de Componentes ERP · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/desarrollo-componentes)

Solución (python)

```
from decimal import Decimal, ROUND_HALF_UP

lineas = [("Tornillos", 1.15), ("Arandelas", 1.525), ("Tuercas", 2.675)]
for nombre, importe in lineas:
 print(nombre, Decimal(str(importe)).quantize(Decimal("0.01"), rounding=ROUND_HALF_UP))
```

Hay dos problemas juntos. 1,525 no se puede guardar exactamente en binario: el `float` es en realidad 1,52499999..., así que redondea hacia abajo. Y aunque fuera exacto, `round` usa el redondeo del banquero (al par más cercano), no «los medios hacia arriba».

`Decimal(str(importe))` crea el número a partir de su texto, que sí es exactamente 1,525, y `quantize(Decimal("0.01"), rounding=ROUND_HALF_UP)` lo redondea a céntimos con la regla de los tiques. Por eso el dinero se maneja con `Decimal` y nunca con `float`.

## 58. Totales de una factura

Fácil · Ejercicios de Sistemas de Gestión Empresarial · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge)

Solución (python)

```
import sys

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

base = iva = 0.0
for linea in lineas:
 _, cantidad, precio, tipo = linea.split(";")
 importe = float(cantidad) * float(precio)
 base += importe
 iva += importe * float(tipo) / 100 # cada línea con su tipo de IVA
print(f"Base: {base:.2f}")
print(f"IVA: {iva:.2f}")
print(f"Total: {base + iva:.2f}")
```

La cuota de IVA se calcula línea a línea, porque cada producto puede tener un tipo distinto (21 % general, 10 % reducido, 4 % superreducido), y se suma. El total es la base más la cuota. `:.2f` redondea a dos decimales solo al mostrar.

## 59. Stock tras los movimientos

Medio · Ejercicios de Sistemas de Gestión Empresarial · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge)

**Solución (python)**

```
import sys

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

separador = lineas.index("---")
stock = {}
for linea in lineas[:separador]:
 producto, cantidad = linea.split()
 stock[producto] = int(cantidad)

for linea in lineas[separador + 1:]:
 tipo, producto, cantidad = linea.split()
 cantidad = int(cantidad)
 if tipo == "entrada":
 stock[producto] = stock.get(producto, 0) + cantidad
 elif cantidad > stock.get(producto, 0):
 print(f"ERROR: stock insuficiente de {producto}")
 else:
 stock[producto] -= cantidad

for producto in sorted(stock):
 print(f"{producto}: {stock[producto]}")
```

`lineas.index("---")` encuentra dónde acaban las existencias iniciales y empiezan los movimientos, y con dos rebanadas de la lista se procesa cada parte. `stock.get(producto, 0)` trata como 0 un producto que todavía no existe, tanto para darle entrada como para rechazar una salida.

Una salida que no se puede servir no se aplica a medias: se rechaza entera y el stock queda igual, que es lo que haría un ERP.

## 60. Validar NIF y NIE de clientes

Medio · Ejercicios de Sistemas de Gestión Empresarial · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge](#)

**Solución (python)**

```
import sys
import re

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]
TABLA = "TRWAGMYFPDXBNJZSQVHLCKE"

for linea in lineas:
 doc = linea.upper()
 m = re.fullmatch(r"([XYZ]\d{7}|\d{8})([A-Z])", doc) # NIE: letra y 7 cifras; NIF: 8 cifras
 valido = False
 if m:
 numero = m.group(1)
 if numero[0] in "XYZ":
 numero = str("XYZ".index(numero[0])) + numero[1:] # X=0, Y=1, Z=2
 valido = TABLA[int(numero) % 23] == m.group(2)
 print(f"{doc} {'válido' if valido else 'no válido'}")
```

La expresión regular reconoce las dos formas con una alternativa: una X, Y o Z seguida de 7 cifras (NIE) o 8 cifras (NIF), y después la letra de control. En el NIE la letra inicial se sustituye por su cifra (X=0, Y=1, Z=2) delante del número.

La letra de control es la posición `número % 23` en la cadena TABLA: si la letra escrita no coincide, el documento tiene una errata. Se pasa todo a mayúsculas antes de comprobarlo, porque la letra puede venir en minúscula.

## 61. Informe de antigüedad de deuda

Difícil · Ejercicios de Sistemas de Gestión Empresarial · [apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge](https://apuntesdam.com/subject/sistemas-gestion-empresarial/topic/ejercicios-sge)

Solución (python)

```
import sys
from datetime import date

lineas = [l.strip() for l in sys.stdin.read().splitlines() if l.strip()]

informe = date.fromisoformat(lineas[0])
TRAMOS = ["No vencido", "1-30", "31-60", "61-90", "Más de 90"]
totales = {t: 0.0 for t in TRAMOS}
vencido = {} # cliente → importe vencido
for linea in lineas[1:]:
 _, cliente, vencimiento, pendiente = linea.split(";")
 dias = (informe - date.fromisoformat(vencimiento)).days
 importe = float(pendiente)
 if dias <= 0:
 tramo = "No vencido"
 elif dias <= 30:
 tramo = "1-30"
 elif dias <= 60:
 tramo = "31-60"
 elif dias <= 90:
 tramo = "61-90"
 else:
 tramo = "Más de 90"
 totales[tramo] += importe
 if dias > 0:
 vencido[cliente] = vencido.get(cliente, 0) + importe

for t in TRAMOS:
 print(f"{t}: {totales[t]:.2f}")
if vencido:
 cliente = max(vencido, key=vencido.get)
 print(f"Mayor deudor vencido: {cliente} ({vencido[cliente]:.2f})")
else:
 print("Mayor deudor vencido: ninguno")
```

Los días de retraso son la fecha del informe menos la de vencimiento (negativos si aún no ha vencido), y una cadena de `elif` asigna el tramo comprobando solo el límite superior de cada uno. Todos los tramos se inicializan a 0 para mostrarlos aunque estén vacíos.

A la vez se acumula lo vencido por cliente, y `max(vencido, key=vencido.get)` devuelve el cliente con el mayor importe. Este informe de antigüedad es el que usa el departamento financiero para decidir a quién reclamar primero: cuanto más vieja es una deuda, más difícil es cobrarla.

## 62. Corrector de exámenes tipo test

Difícil · 60 minutos · [apuntesdam.com/ejercicios/python/corrector-de-examenes-tipo-test](https://apuntesdam.com/ejercicios/python/corrector-de-examenes-tipo-test)

Solución (python)

```
import sys

def formato(numero):
 """Dos decimales con coma, como se escriben las notas en España."""
 return f"{numero:.2f}".replace(".", ",")

def main():
 lineas = sys.stdin.read().split("\n")
 clave = lineas[0].split()[1]
 n = len(clave)
 alumnos = [] # (nombre, nota) de cada examen válido
 fallos = [0] * n # fallos de cada pregunta
 aciertos_pregunta = [0] * n # aciertos de cada pregunta

 for num, linea in enumerate(lineas[1:], start=2):
 linea = linea.strip()
 if not linea:
 continue
 if ";" not in linea:
 print(f"Aviso: la línea {num} no tiene el formato nombre;respuestas")
 continue
 nombre, respuestas = (parte.strip() for parte in linea.split(";", 1))
 if len(respuestas) != n or any(r not in "ABCD-" for r in respuestas):
 print(f"Aviso: las respuestas de {nombre} no son válidas y no se corrigen")
 continue
 aciertos = mal = 0
 for i, (respuesta, correcta) in enumerate(zip(respuestas, clave)):
 if respuesta == "-":
 continue
 if respuesta == correcta:
 aciertos += 1
 aciertos_pregunta[i] += 1
 else:
 mal += 1
 fallos[i] += 1
 # Tres fallos restan un acierto, pero la puntuación nunca baja de 0
 puntos = max(0, aciertos - mal / 3)
 alumnos.append((nombre, puntos * 10 / n))

 if not alumnos:
 print("No hay exámenes que corregir.")
 return

 alumnos.sort(key=lambda a: (-a[1], a[0]))
 print(f"Corrección de {n} preguntas")
 for pos, (nombre, nota) in enumerate(alumnos, start=1):
 print(f"{pos:>2}. {nombre:<22}{formato(nota):>6}")

 media = sum(nota for _, nota in alumnos) / len(alumnos)
 aprobados = sum(1 for _, nota in alumnos if nota >= 5)
 print(f"Media: {formato(media)}")
 print(f"Aprobados: {aprobados} de {len(alumnos)}")

 # max devuelve el primero de los empatados: la pregunta de número más bajo
 peor = max(range(n), key=lambda i: fallos[i])
 if fallos[peor] == 0:
 print("Pregunta más fallada: ninguna")
 else:
 print(f"Pregunta más fallada: {peor + 1} ({fallos[peor]} {'fallo' if fallos[peor] == 1 else 'fallos'})")
 sin_aciertos = [str(i + 1) for i in range(n) if aciertos_pregunta[i] == 0]
 print("Sin aciertos: " + (", ".join(sin_aciertos) if sin_aciertos else "ninguna"))
```

`main()`

El programa recorre la entrada una sola vez y, por cada examen, actualiza tres cosas: la nota del alumno y los contadores de aciertos y fallos de cada pregunta. Así todas las estadísticas finales salen sin volver a leer nada.

La validación se hace antes de tocar ningún contador: un examen mal escrito no debe afectar a la estadística de las preguntas.

La nota se calcula con la fórmula de la penalización ( `aciertos - mal / 3` ) y `max(0, ...)` impide que sea negativa. Ordenar por la tupla `(-nota, nombre)` resuelve el orden descendente y los empates en una sola línea.

La conversión a coma decimal se concentra en una función, `formato` , para no repetirla en cada `print` .

## 63. Informe de ventas a partir de un CSV

Difícil · 70 minutos · [apuntesdam.com/ejercicios/python/informe-de-ventas-csv](https://apuntesdam.com/ejercicios/python/informe-de-ventas-csv)

Solución (python)

```
import sys

def euros(centimos):
 """1234.5 € → «1.234,50 €»: puntos de millar y coma decimal."""
 texto = f"{centimos / 100:,.2f}"
 return texto.replace(",", "X").replace(".", ",").replace("X", ".") + " €"

def leer_venta(linea):
 """Devuelve (fecha, producto, categoría, unidades, importe en céntimos) o None si la línea no es válida."""
 partes = [p.strip() for p in linea.split(",")]
 if len(partes) != 5:
 return None
 fecha, producto, categoria, unidades, precio = partes
 try:
 unidades = int(unidades)
 precio_cent = round(float(precio) * 100)
 except ValueError:
 return None
 if unidades <= 0 or precio_cent < 0:
 return None
 return fecha, producto, categoria, unidades, unidades * precio_cent

def main():
 lineas = sys.stdin.read().split("\n")
 por_categoria, por_producto, por_dia, por_mes = {}, {}, {}, {}
 total = 0
 descartadas = []
 for num, linea in enumerate(lineas[1:], start=2):
 if not linea.strip():
 continue
 venta = leer_venta(linea)
 if venta is None:
 descartadas.append(num)
 continue
 fecha, producto, categoria, unidades, importe = venta
 total += importe
 por_categoria[categoria] = por_categoria.get(categoria, 0) + importe
 por_producto[producto] = por_producto.get(producto, 0) + unidades
 por_dia[fecha] = por_dia.get(fecha, 0) + importe
 mes = fecha[:7]
 por_mes[mes] = por_mes.get(mes, 0) + importe

 if total == 0 and not por_producto:
 print("No hay ventas válidas.")
 return

 print(f"Ventas totales: {euros(total)}")
 print("Por categoría:")
 for categoria, importe in sorted(por_categoria.items(), key=lambda c: (-c[1], c[0])):
 porcentaje = f"{100 * importe / total:.1f}".replace(".", ",")
 print(f" {categoria:<16}{euros(importe):>12} {porcentaje:>5} %")

 producto, unidades = min(por_producto.items(), key=lambda p: (-p[1], p[0]))
 print(f"Producto más vendido: {producto} ({unidades} {'unidad' if unidades == 1 else 'unidades'})")
 dia, importe = min(por_dia.items(), key=lambda d: (-d[1], d[0]))
 print(f"Mejor día: {dia} ({euros(importe)})")

 print("Por mes:")
 for mes in sorted(por_mes):
 barra = "#" * (por_mes[mes] // 5000) # una almohadilla por cada 50 €
 print(f" {mes} {euros(por_mes[mes]):>12} {barra}".rstrip())
```

```
lista = f" ({', '.join(map(str, descartadas))}" if descartadas else ""
print(f"Líneas descartadas: {len(descartadas)}{lista}")
```

```
main()
```

La validación vive en `leer_venta`, que convierte los campos y devuelve `None` ante cualquier problema. El bucle principal solo decide qué hacer con el resultado: descartar la línea o acumularla.

Todo se acumula en céntimos enteros. Así  $0,80 \times 40$  es exactamente 3200 céntimos y las sumas no arrastran errores de coma flotante; la conversión a euros ocurre una sola vez, en `euros`.

Cada pregunta del informe es un diccionario: por categoría, por producto, por día y por mes. Una sola pasada por las ventas basta para rellenarlos todos.

Los empates se resuelven con claves de ordenación en forma de tupla: el primer elemento es el importe en negativo (para ir de mayor a menor) y el segundo el nombre o la fecha. `min` con esa clave devuelve directamente el ganador.

## 64. Planificador de procesos: FCFS, SJF, SRTF y Round Robin

Muy difícil · 100 minutos · [apuntesdam.com/ejercicios/python/planificador-de-procesos](https://apuntesdam.com/ejercicios/python/planificador-de-procesos)

Solución (python)

```
import sys

NOMBRES = {"FCFS": "FCFS (primero en llegar, primero en ser servido)", "SJF": "SJF (el más corto primero, sin expulsión)",
 "SRTF": "SRTF (menor tiempo restante, con expulsión)", "RR": "Round Robin"}

def coma(x):
 return f"{x:.2f}".replace(".", ",")

def simular(algoritmo, quantum, procesos):
 """Devuelve el diagrama de Gantt como lista de [nombre, inicio, fin] y el instante en que termina cada proceso."""
 restante = {p["nombre"]: p["rafaga"] for p in procesos}
 orden = {p["nombre"]: i for i, p in enumerate(procesos)}
 pendientes = sorted(procesos, key=lambda p: (p["llegada"], orden[p["nombre"]]))
 listos, gantt, fin = [], [], {}
 t = 0

 def admitir(hasta):
 """Pasa a la cola de listos los procesos que han llegado hasta el instante «hasta»."""
 while pendientes and pendientes[0]["llegada"] <= hasta:
 listos.append(pendientes.pop(0))

 def ejecutar(p, duracion):
 nonlocal t
 if gantt and gantt[-1][0] == p["nombre"] and gantt[-1][2] == t:
 gantt[-1][2] += duracion # el mismo proceso sigue: se alarga el tramo
 else:
 gantt.append([p["nombre"], t, t + duracion])
 t += duracion
 restante[p["nombre"]] -= duracion

 admitir(t)
 while listos or pendientes:
 if not listos: # CPU ociosa hasta la siguiente llegada
 gantt.append(["--", t, pendientes[0]["llegada"]])
 t = pendientes[0]["llegada"]
 admitir(t)
 continue
 if algoritmo == "FCFS":
 p = listos.pop(0)
 ejecutar(p, restante[p["nombre"]])
 elif algoritmo == "SJF":
 p = min(listos, key=lambda q: (q["rafaga"], q["llegada"], orden[q["nombre"]]))
 listos.remove(p)
 ejecutar(p, restante[p["nombre"]])
 elif algoritmo == "SRTF":
 p = min(listos, key=lambda q: (restante[q["nombre"]], q["llegada"], orden[q["nombre"]]))
 # Se ejecuta hasta que acabe o hasta la próxima llegada, que puede expulsarlo
 hasta = t + restante[p["nombre"]]
 if pendientes:
 hasta = min(hasta, pendientes[0]["llegada"])
 ejecutar(p, hasta - t)
 if restante[p["nombre"]] > 0:
 admitir(t)
 continue
 listos.remove(p)
 else: # RR
 p = listos.pop(0)
 ejecutar(p, min(quantum, restante[p["nombre"]]))
 admitir(t) # los que llegan entran en la cola antes que el expulsado
 if restante[p["nombre"]] > 0:
 listos.append(p)
 continue
```

```

 fin[p["nombre"]] = t
 admitir(t)
 return gantt, fin

def main():
 lineas = sys.stdin.read().split("\n")
 cabecera = lineas[0].split()
 algoritmo = cabecera[0].upper() if cabecera else ""
 quantum = 0
 if algoritmo == "RR":
 if len(cabecera) != 2 or not cabecera[1].isdigit() or int(cabecera[1]) == 0:
 print("Algoritmo no válido")
 return
 quantum = int(cabecera[1])
 elif algoritmo not in NOMBRES or len(cabecera) != 1:
 print("Algoritmo no válido")
 return

 procesos = []
 for num, linea in enumerate(lineas[1:], start=2):
 partes = linea.split()
 if not partes:
 continue
 if len(partes) != 3 or not partes[1].isdigit() or not partes[2].isdigit() or int(partes[2]) == 0 \
 or any(p["nombre"] == partes[0] for p in procesos):
 print(f"Aviso: línea {num} ignorada")
 continue
 procesos.append({"nombre": partes[0], "llegada": int(partes[1]), "rafaga": int(partes[2])})
 if not procesos:
 print("No hay procesos")
 return

 gantt, fin = simular(algoritmo, quantum, procesos)
 print("Algoritmo: " + NOMBRES[algoritmo] + (f" con quantum {quantum}" if algoritmo == "RR" else ""))
 print("Diagrama: " + " ".join(f"| {n} {a}-{b}" for n, a, b in gantt) + " |")
 print(f'{"Proceso":<8>{"Llegada":>8>{"Ráfaga":>8>{"Fin":>6>{"Retorno":>9>{"Espera":>8}}')
 retornos, esperas = [], []
 for p in procesos:
 retorno = fin[p["nombre"]] - p["llegada"]
 espera = retorno - p["rafaga"]
 retornos.append(retorno)
 esperas.append(espera)
 print(f'{p["nombre"]:<8}{p["llegada"]:>8}{p["rafaga"]:>8}{fin[p["nombre"]]:>6}{retorno:>9}{espera:>8}')
 print(f'Media de retorno: {coma(sum(retornos) / len(procesos))} · Media de espera: {coma(sum(esperas) / len(procesos))}')
 ocupada = sum(b - a for n, a, b in gantt if n != "--")
 print(f'Uso de la CPU: {coma(100 * ocupada / gantt[-1][2])} %')

main()

```

La simulación avanza por eventos: en lugar de mirar el reloj de unidad en unidad, salta de una decisión a la siguiente (fin de un proceso, fin de un quantum o llegada de un proceso). Es más rápido y deja claro en qué instantes decide el planificador.

Los cuatro algoritmos comparten el bucle y las funciones `admitir` y `ejecutar`; solo cambia la elección del proceso y el tiempo que se le deja. Es la misma estructura que usan los sistemas operativos reales, donde la política de planificación es un módulo intercambiable.

Las tuplas como clave de `min` codifican la política y los desempates en una sola expresión, y hacen que el resultado sea siempre el mismo para la misma entrada, sin depender del orden en que se guardan los procesos.

Retorno y espera salen solos al final:  $\text{retorno} = \text{fin} - \text{llegada}$  y  $\text{espera} = \text{retorno} - \text{ráfaga}$ . Comparar las medias entre algoritmos con los mismos procesos muestra por qué SJF y SRTF minimizan la espera media y por qué Round Robin, aunque espere más de media, responde antes a los procesos cortos.

## 65. Plan de direccionamiento con subredes VLSM

Difícil · 70 minutos · [apuntesdam.com/ejercicios/python/subredes-vlsm](http://apuntesdam.com/ejercicios/python/subredes-vlsm)

Solución (python)

```
import sys

def a_numero(ip):
 """«192.168.10.0» → entero de 32 bits; None si no es una IPv4."""
 partes = ip.split(".")
 if len(partes) != 4 or not all(p.isdigit() and int(p) <= 255 for p in partes):
 return None
 n = 0
 for p in partes:
 n = n * 256 + int(p)
 return n

def a_ip(n):
 return ".".join(str((n >> desplazamiento) & 255) for desplazamiento in (24, 16, 8, 0))

def mascara(prefijo):
 return (0xFFFFFFFF << (32 - prefijo)) & 0xFFFFFFFF

def prefijo_para(hosts):
 """El prefijo más largo cuyo bloque tiene sitio para «hosts» (sin contar la red y la difusión), como
 mucho /30."""
 prefijo = 30
 while 2 ** (32 - prefijo) - 2 < hosts:
 prefijo -= 1
 return prefijo

def main():
 lineas = sys.stdin.read().split("\n")
 texto = lineas[0].strip()
 if texto.count("/") != 1 or a_numero(texto.split("/")[0]) is None or not texto.split("/")
 [1].isdigit() \
 or not 8 <= int(texto.split("/")[1]) <= 30:
 print(f"Red no válida: {texto}")
 return
 ip, prefijo = texto.split("/")
 prefijo = int(prefijo)
 base = a_numero(ip)
 if base & mascara(prefijo) != base:
 print(f"{ip} no es una dirección de red (sería {a_ip(base & mascara(prefijo))}/{prefijo}")
 return
 total = 2 ** (32 - prefijo)

 subredes = []
 for num, linea in enumerate(lineas[1:], start=2):
 partes = linea.split()
 if not partes:
 continue
 if len(partes) < 2 or not partes[-1].isdigit() or int(partes[-1]) == 0:
 print(f"Aviso: línea {num} ignorada")
 continue
 subredes.append((" ".join(partes[:-1]), int(partes[-1])))

 print(f"Red: {a_ip(base)}/{prefijo} ({total} direcciones)")
 # VLSM: de la más grande a la más pequeña, así cada bloque queda alineado con su tamaño
 siguiente = base
 for nombre, hosts in sorted(subredes, key=lambda s: -s[1]):
 p = prefijo_para(hosts)
 tamaño = 2 ** (32 - p)
 if p < prefijo or siguiente + tamaño > base + total:
 print(f"{nombre} ({hosts} hosts) → no cabe: necesita un /{p} y quedan {base + total -
 siguiente} direcciones")
```

```

 continue
 print(f"{nombre} ({hosts} hosts) → {a_ip(siguiete)}/{p}")
 print(f" máscara {a_ip(mascara(p))} · hosts útiles {tamano - 2}")
 print(f" primera {a_ip(siguiete + 1)} · última {a_ip(siguiete + tamano - 2)} · difusión
 {a_ip(siguiete + tamano - 1)}")
 siguiete += tamano

libres = base + total - siguiete
if libres:
 print(f"Sin asignar: {libres} de {total} direcciones, desde {a_ip(siguiete)}")
else:
 print(f"Sin asignar: 0 de {total} direcciones")

main()

```

Tratar las direcciones como enteros de 32 bits convierte todo el subnetting en aritmética: la máscara es un desplazamiento, la red un `&` y las direcciones de cada subred, sumas. La conversión a texto solo ocurre al escribir.

Asignar de mayor a menor garantiza la alineación: después de bloques de 64, 32 y 16, la siguiente dirección libre es siempre múltiplo del tamaño del bloque que toca, que es exactamente lo que exige una dirección de red.

La validación tiene dos niveles: que el texto sea una IP con prefijo, y que la IP sea de verdad una dirección de red. Es un error frecuente escribir 192.168.10.5/24 queriendo decir la red 192.168.10.0/24.

El módulo `ipaddress` de Python hace todo esto (`ip_network`, `subnets`, `broadcast_address`). Escribirlo a mano enseña lo que hay debajo; en un programa real conviene usar la librería.

## 66. Inventario de almacén con valoración FIFO y reposición

Muy difícil · 90 minutos · [apuntesdam.com/ejercicios/python/inventario-con-valoracion-fifo](https://apuntesdam.com/ejercicios/python/inventario-con-valoracion-fifo)

Solución (python)

```
import sys
from decimal import Decimal, ROUND_HALF_UP

def euros(x):
 """Decimal → «1.234,56 €» redondeado al céntimo."""
 x = x.quantize(Decimal("0.01"), rounding=ROUND_HALF_UP)
 return f"{x:,.2f}".replace(",","X").replace(".",",").replace("X",".") + " €"

def main():
 productos = {} # código → datos del producto
 movimientos = [] # (fecha, orden de la entrada, tipo, código, cantidad, coste)
 for num, linea in enumerate(sys.stdin.read().split("\n"), start=1):
 linea = linea.strip()
 if not linea:
 continue
 tipo, _, resto = linea.partition(" ")
 campos = [c.strip() for c in resto.split(";")]
 try:
 if tipo == "PRODUCTO" and len(campos) == 4:
 codigo, nombre, minimo, maximo = campos[0], campos[1], int(campos[2]), int(campos[3])
 if codigo in productos or not 0 <= minimo <= maximo:
 raise ValueError
 productos[codigo] = {"nombre": nombre, "minimo": minimo, "maximo": maximo, "lotes": [],
 "vendido": Decimal(0)}
 elif tipo == "ENTRADA" and len(campos) == 4:
 cantidad, coste = int(campos[2]), Decimal(campos[3])
 if cantidad <= 0 or coste < 0 or campos[1] not in productos:
 raise ValueError
 movimientos.append((campos[0], num, tipo, campos[1], cantidad, coste))
 elif tipo == "SALIDA" and len(campos) == 3:
 cantidad = int(campos[2])
 if cantidad <= 0 or campos[1] not in productos:
 raise ValueError
 movimientos.append((campos[0], num, tipo, campos[1], cantidad, None))
 else:
 raise ValueError
 except (ValueError, ArithmeticError):
 print(f"Aviso: línea {num} ignorada")

 # Los movimientos llegan desordenados: se aplican por fecha (y, el mismo día, en el orden de la
 # entrada)
 for fecha, _, tipo, codigo, cantidad, coste in sorted(movimientos):
 p = productos[codigo]
 if tipo == "ENTRADA":
 p["lotes"].append([cantidad, coste])
 continue
 stock = sum(c for c, _ in p["lotes"])
 if cantidad > stock:
 print(f"Aviso: salida de {cantidad} {codigo} el {fecha} rechazada (stock {stock})")
 continue
 # FIFO: se gastan primero las unidades de los lotes más antiguos, a su coste
 while cantidad:
 lote = p["lotes"][0]
 usadas = min(cantidad, lote[0])
 p["vendido"] += usadas * lote[1]
 lote[0] -= usadas
 cantidad -= usadas
 if lote[0] == 0:
 p["lotes"].pop(0)

 if not productos:
 print("No hay productos")
 return
 total = Decimal(0)
```

```

reponer = []
print("Inventario:")
for codigo, p in productos.items():
 stock = sum(c for c, _ in p["lotos"])
 valor = sum((c * coste for c, coste in p["lotos"]), Decimal(0))
 total += valor
 print(f" {codigo} {p['nombre']}: stock {stock} · valor {euros(valor)} · coste de lo vendido {euros(p['vendido'])}")
 if stock < p["minimo"]:
 reponer.append(f" {codigo} {p['nombre']}: pedir {p['maximo'] - stock} (stock {stock}, mínimo {p['minimo']}, máximo {p['maximo']})")
print("Reponer:" if reponer else "Reponer: nada")
for r in reponer:
 print(r)
print(f"Valor del almacén: {euros(total)}")

main()

```

Separar la lectura (que valida y guarda) de la aplicación (que ordena y ejecuta) permite procesar los movimientos en el orden correcto aunque lleguen desordenados, y que un error de formato no deje el almacén a medias.

La lista de lotes de cada producto es una cola: FIFO literalmente. El bucle de la salida recorre los lotes desde el más antiguo y reparte la cantidad entre ellos, que es lo que hace el ERP para calcular el coste de cada venta.

Con `Decimal` todos los cálculos son exactos y el redondeo ocurre una sola vez, al mostrar el resultado. Es lo que exige la contabilidad: el valor del almacén tiene que cuadrar con la suma de lo comprado menos lo vendido.

El coste de lo vendido depende del método: con FIFO las salidas se valoran a los precios antiguos, y si los precios suben, el stock que queda vale más. Es una decisión contable de la empresa, y por eso los ERP permiten elegir entre FIFO, coste medio y otros.

## 67. Resumen de pedidos en XML con ElementTree

Difícil · 70 minutos · [apuntesdam.com/ejercicios/python/resumen-de-pedidos-xml](https://apuntesdam.com/ejercicios/python/resumen-de-pedidos-xml)

Solución (python)

```
import sys
import xml.etree.ElementTree as ET
from decimal import Decimal, InvalidOperation

ESTADOS = ("pendiente", "enviado", "entregado", "cancelado")

def leer_pedido(nodo):
 """Comprueba un <pedido> y devuelve su id, cliente, estado e importe; lanza ValueError con el
 motivo."""
 for atributo in ("id", "cliente", "estado"):
 if not nodo.get(atributo):
 raise ValueError(f"falta el atributo {atributo}")
 if nodo.get("estado") not in ESTADOS:
 raise ValueError(f"estado desconocido {nodo.get('estado')}")
 lineas = nodo.findall("linea")
 if not lineas:
 raise ValueError("no tiene líneas")
 importe = Decimal(0)
 unidades = {}
 for linea in lineas:
 ref, cantidad, precio = linea.get("ref"), linea.get("cantidad", ""), linea.get("precio", "")
 if not ref or not cantidad.isdigit() or int(cantidad) == 0:
 raise ValueError(f"línea {ref or '?'} con una cantidad no válida")
 try:
 precio = Decimal(precio)
 except InvalidOperation:
 raise ValueError(f"línea {ref} con un precio no válido")
 importe += int(cantidad) * precio
 unidades[ref] = unidades.get(ref, 0) + int(cantidad)
 return nodo.get("id"), nodo.get("cliente"), nodo.get("estado"), importe, unidades

def main():
 try:
 raiz = ET.fromstring(sys.stdin.read())
 except ET.ParseError as e:
 print(f"XML mal formado (línea {e.position[0]})")
 return
 if raiz.tag != "pedidos":
 print("El elemento raíz debe ser <pedidos>")
 return

 por_cliente = {} # cliente → [pedidos, importe]
 por_estado = {e: 0 for e in ESTADOS}
 ventas = {}
 total = Decimal(0)
 validos = 0
 for nodo in raiz.findall("pedido"):
 try:
 id_, cliente, estado, importe, unidades = leer_pedido(nodo)
 except ValueError as e:
 print(f"Aviso: pedido {nodo.get('id', '?')} ignorado: {e}")
 continue
 validos += 1
 por_estado[estado] += 1
 if estado == "cancelado":
 continue # se cuenta, pero no suma importe ni unidades
 datos = por_cliente.setdefault(cliente, [0, Decimal(0)])
 datos[0] += 1
 datos[1] += importe
 total += importe
 for ref, n in unidades.items():
 ventas[ref] = ventas.get(ref, 0) + n

 # El resumen se construye como un árbol y se serializa con sangría
```

```

resumen = ET.Element("resumen", fecha=raiz.get("fecha", ""), pedidos=str(validos), importe=f"
 {total:.2f}")
estados = ET.SubElement(resumen, "estados")
for estado in ESTADOS:
 ET.SubElement(estados, "estado", nombre=estado, pedidos=str(por_estado[estado]))
clientes = ET.SubElement(resumen, "clientes")
for cliente, (n, importe) in sorted(por_cliente.items(), key=lambda c: (-c[1][1], c[0])):
 ET.SubElement(clientes, "cliente", nombre=cliente, pedidos=str(n), importe=f"{importe:.2f}")
if ventas:
 ref, n = min(ventas.items(), key=lambda v: (-v[1], v[0]))
 ET.SubElement(resumen, "masVendido", ref=ref, unidades=str(n))
ET.indent(resumen, space=" ")
print(ET.tostring(resumen, encoding="unicode"))

```

```
main()
```

Un analizador de XML comprueba que el documento está bien formado antes de que el programa vea nada, y da acceso a los datos por su estructura (elementos y atributos), no por su posición en el texto. Por eso no le afectan los saltos de línea, el orden de los atributos ni las comillas simples o dobles.

La validación vive en `leer_pedido`, que traduce cada problema en una excepción con un mensaje claro. El bucle principal decide qué hacer con un pedido malo (avisar y seguir) sin mezclarse con las reglas.

Generar el XML con `Element` y `SubElement` en lugar de con `print` garantiza que la salida está bien formada: los nombres con `&` o comillas se escapan automáticamente, algo que es muy fácil olvidar al escribir etiquetas a mano.

Los importes se acumulan con `Decimal` y se formatean con dos decimales y punto, el formato que espera cualquier programa que lea el XML (la coma decimal es para las personas, no para los datos).

## 68. Detección de ataques de fuerza bruta en el registro de SSH

Difícil · 70 minutos · [apuntesdam.com/ejercicios/python/deteccion-de-ataques-ssh](https://apuntesdam.com/ejercicios/python/deteccion-de-ataques-ssh)

Solución (python)

```
import re
import sys
from datetime import datetime, timedelta

FALLO = re.compile(r"Failed password for (invalid user)?(\\S+) from (\\S+) port")
EXITO = re.compile(r"Accepted (password|publickey) for (\\S+) from (\\S+) port")
VENTANA = timedelta(minutes=10)
MAX_FALLOS = 5

def main():
 fallos_ip = {} # ip → lista de instantes de fallo
 bloqueos = {} # ip → (instante del bloqueo, primer fallo de la ventana)
 inexistentes = set()
 fallos_usuario = {}
 alertas = []
 n_fallos = n_exitos = 0

 for linea in sys.stdin.read().split("\n"):
 if not linea.strip():
 continue
 # Las líneas de syslog empiezan con «Oct 3 10:15:01» (sin año)
 try:
 instante = datetime.strptime(linea[:15], "%b %d %H:%M:%S").replace(year=2026)
 except ValueError:
 continue
 if m := FALLO.search(linea):
 n_fallos += 1
 invalido, usuario, ip = m.groups()
 if invalido:
 inexistentes.add(usuario)
 fallos_usuario[ip] = fallos_usuario.get(ip, 0) + 1
 lista = fallos_ip.setdefault(ip, [])
 lista.append(instante)
 # Ventana deslizante: solo cuentan los fallos de los últimos 10 minutos
 recientes = [t for t in lista if instante - t <= VENTANA]
 if len(recientes) >= MAX_FALLOS and ip not in bloqueos:
 bloqueos[ip] = (instante, recientes[0])
 elif m := EXITO.search(linea):
 n_exitos += 1
 _, usuario, ip = m.groups()
 previos = len(fallos_ip.get(ip, []))
 if previos:
 alertas.append(f" {usuario} entró desde {ip} a las {instante:%H:%M:%S} tras {previos} {'fallo' if previos == 1 else 'fallos'} desde esa IP")

 print(f"Intentos fallidos: {n_fallos} · Accesos correctos: {n_exitos}")
 if bloqueos:
 print("Bloquear:")
 for ip, (cuando, desde) in sorted(bloqueos.items(), key=lambda b: b[1][0]):
 print(f" {ip} a las {cuando:%H:%M:%S} ({MAX_FALLOS} fallos desde las {desde:%H:%M:%S})")
 else:
 print("Bloquear: ninguna IP")
 print("Usuarios inexistentes probados: " + ("", ".join(sorted(inexistentes)) if inexistentes else "ninguno"))
 if fallos_usuario:
 ranking = sorted(fallos_usuario.items(), key=lambda u: (-u[1], u[0]))[:3]
 print("Usuarios más atacados: " + " ", ".join(f"{u} ({n})" for u, n in ranking))
 print("Alertas:" if alertas else "Alertas: ninguna")
 for a in alertas:
 print(a)

main()
```

Las expresiones regulares con grupos convierten una línea de texto libre en datos estructurados (usuario, IP, si el usuario existe). Es la base de cualquier análisis de registros, desde un script como este hasta herramientas como Logstash.

La ventana deslizante es la clave de fail2ban: no importa cuántos fallos haya en total, sino cuántos caben en 10 minutos. Recalcularla en cada fallo con una lista por IP es sencillo y suficiente para un registro diario; con millones de líneas se usaría una cola que descarta los fallos antiguos.

Separar la recogida de datos (un único recorrido del registro) de la presentación (el informe al final) permite añadir estadísticas nuevas sin volver a leer nada.

Las alertas cruzan dos tipos de eventos: un acceso correcto no es sospechoso por sí solo, pero sí después de fallos desde la misma IP. Ese tipo de correlación es lo que hacen los sistemas de detección de intrusiones (SIEM).