

Ejercicios resueltos de Sistemas de Gestión Empresarial

Los 25 ejercicios de Sistemas de Gestión Empresarial de la web, tema a tema: 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

Sistemas ERP-CRM

1. El embudo de ventas del CRM

3. Dimensionar servidores para Odoo

Implantación de un ERP-CRM

4. ¿Puede este usuario hacerlo?

6. Rotación de copias abuelo-padre-hijo

Adaptación de un ERP-CRM

7. Encuentra el fallo: el descuento para casi todos

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

Desarrollo de Componentes ERP

11. Encuentra el fallo: el recuento de ventas que se rompe

13. Encuentra el fallo: las líneas que se comparten

15. Encuentra el fallo: el pedido confirmado que no se factura

17. Secuencias de referencias

Ejercicios de Sistemas de Gestión Empresarial

19. Totales de una factura

21. Validar NIF y NIE de clientes

Ejercicios largos

23. Corrector de exámenes tipo test

25. Inventario de almacén con valoración FIFO y reposición

2. Coste total de propiedad a varios años

5. Limpiar los clientes antes de migrarlos

8. Encuentra el fallo: el ranking que no existe

10. Un evaluador de dominios de Odoo

12. Un campo calculado: la amortización

14. Encuentra el fallo: los agotados que se escapan

16. Restricciones de un modelo

18. Encuentra el fallo: los céntimos que se pierden

20. Stock tras los movimientos

22. Informe de antigüedad de deuda

24. Informe de ventas a partir de un CSV

Sistemas ERP-CRM

1. El embudo de ventas del CRM

Medio · Sistemas ERP-CRM · 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: -

2. Coste total de propiedad a varios años

Difícil · Sistemas ERP-CRM · 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: licencia × usuarios × 12 × años + implantación + formación + hardware + mantenimiento_anual × 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
```

3. Dimensionar servidores para Odoo

Muy difícil · Sistemas ERP-CRM · 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
```

Implantación de un ERP-CRM

4. ¿Puede este usuario hacerlo?

Medio · Implantación de un ERP-CRM · 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
```

5. Limpiar los clientes antes de migrarlos

Difícil · Implantación de un ERP-CRM · 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 empieza 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
```

6. Rotación de copias abuelo-padre-hijo

Muy difícil · Implantación de un ERP-CRM · 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

Adaptación de un ERP-CRM

7. Encuentra el fallo: el descuento para casi todos

Fácil · Adaptación de un ERP-CRM · 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 %

8. Encuentra el fallo: el ranking que no existe

Fácil · Adaptación de un ERP-CRM · 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 €
```

9. 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

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]
```

10. Un evaluador de dominios de Odoo

Muy difícil · Adaptación de un ERP-CRM · 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

Desarrollo de Componentes ERP

11. Encuentra el fallo: el recuento de ventas que se rompe

Fácil · Desarrollo de Componentes ERP · 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
```

12. Un campo calculado: la amortización

Medio · Desarrollo de Componentes ERP · 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)
```

13. Encuentra el fallo: las líneas que se comparten

Medio · Desarrollo de Componentes ERP · 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
```

14. Encuentra el fallo: los agotados que se escapan

Medio · Desarrollo de Componentes ERP · 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']
```

15. Encuentra el fallo: el pedido confirmado que no se factura

Medio · Desarrollo de Componentes ERP · 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

16. Restricciones de un modelo

Difícil · Desarrollo de Componentes ERP · 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

17. Secuencias de referencias

Muy difícil · Desarrollo de Componentes ERP · 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
```

18. Encuentra el fallo: los céntimos que se pierden

Muy difícil · Desarrollo de Componentes ERP · 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
```

Ejercicios de Sistemas de Gestión Empresarial

19. Totales de una factura

Fácil · Ejercicios de Sistemas de Gestión Empresarial · 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
```

20. Stock tras los movimientos

Medio · Ejercicios de Sistemas de Gestión Empresarial · 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
```

21. Validar NIF y NIE de clientes

Medio · Ejercicios de Sistemas de Gestión Empresarial · 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
```

22. Informe de antigüedad de deuda

Difícil · Ejercicios de Sistemas de Gestión Empresarial · 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

23. Corrector de exámenes tipo test

Difícil · Python · 60 minutos · 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;ABCAABCDDB
Olga Ruiz;DCBADCBADC
Pedro Sanz;-----ABCD
```

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
```

```
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()
```

24. Informe de ventas a partir de un CSV

Difícil · Python · 70 minutos · 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()
```

25. Inventario de almacén con valoración FIFO y reposición

Muy difícil · Python · 90 minutos · 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()
```

Soluciones

1. El embudo de ventas del CRM

Medio · Sistemas ERP-CRM · 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.

2. Coste total de propiedad a varios años

Difícil · Sistemas ERP-CRM · 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.

3. Dimensionar servidores para Odoo

Muy difícil · Sistemas ERP-CRM · 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.

4. ¿Puede este usuario hacerlo?

Medio · Implantación de un ERP-CRM · 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.

5. Limpiar los clientes antes de migrarlos

Difícil · Implantación de un ERP-CRM · 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.

6. Rotación de copias abuelo-padre-hijo

Muy difícil · Implantación de un ERP-CRM · 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.

7. Encuentra el fallo: el descuento para casi todos

Fácil · Adaptación de un ERP-CRM · 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.

8. Encuentra el fallo: el ranking que no existe

Fácil · Adaptación de un ERP-CRM · 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`.

9. 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

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.

10. Un evaluador de dominios de Odoo

Muy difícil · Adaptación de un ERP-CRM · 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.

11. Encuentra el fallo: el recuento de ventas que se rompe

Fácil · Desarrollo de Componentes ERP · 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)`.

12. Un campo calculado: la amortización

Medio · Desarrollo de Componentes ERP · 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).

13. Encuentra el fallo: las líneas que se comparten

Medio · Desarrollo de Componentes ERP · 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.

14. Encuentra el fallo: los agotados que se escapan

Medio · Desarrollo de Componentes ERP · 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.

15. Encuentra el fallo: el pedido confirmado que no se factura

Medio · Desarrollo de Componentes ERP · 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`.

16. Restricciones de un modelo

Difícil · Desarrollo de Componentes ERP · 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.

17. Secuencias de referencias

Muy difícil · Desarrollo de Componentes ERP · 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.

18. Encuentra el fallo: los céntimos que se pierden

Muy difícil · Desarrollo de Componentes ERP · 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`.

19. Totales de una factura

Fácil · Ejercicios de Sistemas de Gestión Empresarial · 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.

20. Stock tras los movimientos

Medio · Ejercicios de Sistemas de Gestión Empresarial · 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.

21. 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.

22. Informe de antigüedad de deuda

Difícil · Ejercicios de Sistemas de Gestión Empresarial · 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.

23. Corrector de exámenes tipo test

Difícil · Python · 60 minutos · 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`.

24. Informe de ventas a partir de un CSV

Difícil · Python · 70 minutos · 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.

25. Inventario de almacén con valoración FIFO y reposición

Muy difícil · Python · 90 minutos · 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["lotes"])
    valor = sum((c * coste for c, coste in p["lotes"]), 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.