



Practica 11 - Ficheros:parte 2



Instrucciones

Entregar los ejercicios en uno o varios `archivos.py` o en un notebook de Jupyter (`.ipynb`).

1. Pickle. Guardar una lista de diccionarios con datos de usuarios y cargarla después.
Salida esperada:

```
[{'nombre': 'Ana', 'edad': 30}, {'nombre': 'Luis', 'edad': 25}, {'nombre': 'Marta', 'edad': 35}]
```

2. Crear un script llamado **personas.py** que lea los datos de un fichero de texto, que transforme cada fila en un diccionario y lo añada a una lista llamada personas. Luego recorre las personas de la lista y para cada una muestra de forma amigable todos sus campos. El fichero de texto se denominará **personas.txt** y tendrá el siguiente contenido en texto plano (créalo previamente):

```
1;Carlos;Pérez;05/01/1989  
2;Manuel;Heredia;26/12/1973  
3;Rosa;Campos;12/06/1961  
4;David;García;25/07/2006
```

Los campos del diccionario serán por orden: **id, nombre, apellido y nacimiento.**

3. Crear un script llamado **contador.py** que realice varias tareas sobre un fichero llamado **contador.txt** que almacenará un contador de visitas (será un número):

- Nuestro script trabajará sobre el fichero **contador.txt**. Si el fichero no existe o está vacío lo crearemos con el número 0. Si existe simplemente leeremos el valor del contador.
- Luego a partir de un argumento:
 - Si se envía el argumento **inc**, se incrementará el contador en uno y se mostrará por pantalla.
 - Si se envía el argumento **dec**, se decrementará el contador en uno y se mostrará por pantalla.
 - Si no se envía ningún argumento (o algo que no sea inc o dec), se mostrará el valor del contador por pantalla.
- Finalmente guardará de nuevo el valor del contador de nuevo en el fichero.
- Utiliza excepciones si crees que es necesario, puedes mostrar el mensaje: **Error: Fichero corrupto.**

3. Subcadena con `seek()` (texto):

Dado el contenido `"Hola mundo desde Python\n"`:

- Lee 4 caracteres → `Hola`
- Salta 2 más usando `seek()` relativo a la **posición actual**.
- Lee 5 caracteres más y muéstralos.

Salida esperada:

```
Hola  
mundo
```

4. Cola de archivo (últimos 5 bytes, binario): El archivo contiene los siguientes bytes (ASCII): `b"abcdeFGHIJklmn"`. Tu tarea es **leer los últimos 5 bytes del archivo** y mostrar su contenido interpretado como texto ASCII.

Salida esperada:

```
lmn
```

Los últimos cinco bytes reales del archivo son `b"J" b"k" b"l" b"m" b"n"`.

5. Releer desde marca: Lee 7 caracteres, guarda `tell()`, avanza 6 más, lee 4; vuelve a la marca guardada y lee 6.

Salida esperada:

```
bloque_1: <primeros 7>  
bloque_2: <4 tras saltar 6>  
bloque_3: <6 desde la marca guardada>
```

El contenido exacto dependerá del texto de entrada; verifica con `tell()`.

6. Delimitador `;` y suma.

Entrada:

```
id;producto;precio  
10;Teclado;25.50  
11;Ratón;15.00  
12;Monitor;120.00
```

Calcula la **suma** de `precio` y muéstralala con **2 decimales**.

Salida esperada:

```
Suma total: 160.50
```

7. `DictReader`: contar registros por categoría.

Entrada:

```
cat,valor
A,3
B,5
A,2
C,7
B,1
```

Cuenta cuántos registros hay por `cat` y muestra un diccionario ordenado por clave.

Salida esperada:

```
{'A': 2, 'B': 2, 'C': 1}
```

8. `DictWriter`: normalizar decimales.

Entrada:

```
id,monto
1,10
2,3.5
3,7
```

Escribe un CSV de salida con `monto` en **dos decimales**.

Salida esperada (contenido):

```
id,monto
1,10.00
2,3.50
3,7.00
```

9. JSON - `loads` + agregación

Entrada JSON:

```
[  
    {"pais": "ES", "ventas": 120.5},  
    {"pais": "FR", "ventas": 99.0},  
    {"pais": "ES", "ventas": 30.0}  
]
```

Suma las ventas por país y muestra un dict ordenado por clave.

Salida esperada:

```
{'ES': 150.5, 'FR': 99.0}
```

10. JSON — `dumps` con `ensure_ascii=False`

Objeto Python:

```
{"mensaje": "¡Bienvenido, José!", "ok": True}
```

Imprime `dumps` con `indent=2`, `ensure_ascii=False`, `sort_keys=True`.

Salida esperada:

```
{  
    "mensaje": "¡Bienvenido, José!",  
    "ok": true  
}
```