Acceso a ficheros

En este documento veremos algunas nociones elementales de cómo extraer, procesar y/o guardar información en ficheros de texto empleando el lenguaje Python. También veremos algunas operaciones útiles sobre el sistema de ficheros: crear carpetas, listar ficheros y carpetas de una ruta, copiar ficheros...

1. Gestión de ficheros de texto

1.1. Apertura de ficheros

Lo primero que debemos hacer es abrir el fichero correspondiente con la función `open`. Como primer parámetro indicaremos el nombre del fichero a abrir (con una ruta relativa a donde esté el ejecutable Python), y como segundo parámetro para qué lo vamos a abrir: `r` para lectura, `w` para escritura, `r+` para lectura-escritura combinadas... Aquí vemos unos ejemplos:

```
lectura = open("fichero1.txt", "r")
escritura = open("fichero2.txt", "w")
lectura_escritura = open("fichero3.txt", "r+")
```

En el caso de que queramos leer o escribir en un formato determinado, añadimos un parámetro adicional `encoding` para especificar dicho formato. Esto permitirá procesar correctamente algunos símbolos especiales de ciertos alfabetos, como los acentos en el alfabeto latino. Por ejemplo:

```
lectura = open("fichero1.txt", "r", encoding="utf-8")
```

Al abrir para escritura según el ejemplo anterior, se borrará el contenido previo que pudiera tener el fichero. Si queremos mantenerlo para añadir contenido nuevo al final, empleamos la opción `a` en lugar de `w`. Esta opción creará el fichero si no existe, o respetará su contenido previo si ya existía:

```
escrituraAppend = open("fichero3.txt", "a")
```

1.2. Operaciones básicas sobre ficheros de texto

Las dos operaciones básicas que podemos realizar sobre ficheros de texto son la escritura y la lectura (dependiendo del modo en que hayamos abierto el fichero previamente).

Para **escribir datos** en el fichero de salida, empleamos su método `write`:

```
n = 3
escritura.write("El número vale " + str(n))
```

Para **leer datos** del fichero de entrada, lo más habitual es leer el fichero con la función `readlines`, que nos da ya una lista con todas las líneas para irlas procesando:

```
lineas = lectura.readlines()
for linea in lineas:
    # Procesamiento de la línea en cuestión
```

Existen otros métodos de lectura alternativa, como el método `read`, que se emplea para leer bytes de ficheros binarios (imágenes, por ejemplo).

1.3. Cierre de fichero

Es importante cerrar el fichero una vez finalizado el trabajo con él, empleando la instrucción `close` (tanto para lectura como para escritura):

```
lectura.close()
escritura.close()
```

1.4. Uso de la cláusula *with*

La declaración `with` en Python se utiliza para crear un contexto de administración (*context manager*) que garantiza que ciertos recursos se gestionen adecuadamente, como la apertura y el cierre de archivos, conexiones a bases de datos o la liberación de recursos. Podemos emplearlo, por ejemplo, para abrir un fichero (para lectura o escritura) y despreocuparnos de su cierre:

```
with open("archivo.txt", "r") as archivo:
    # Realizar operaciones de lectura en el archivo
    contenido = archivo.readlines()
    # El archivo se cerrará automáticamente al salir del bloque 'with'

# Fuera del bloque 'with', el archivo ya está cerrado
```

1.5. Ejemplo

El siguiente ejemplo lee todas las líneas de un fichero "*datos.txt*". En cada línea hay números separados por espacios. Tras leer cada línea, procesará todos los números que hay contenidos en ella y mostrará su suma:

```
lectura = open("datos.txt", "r")
lineas = lectura.readlines()
numLinea = 0
for linea in lineas:
    numLinea += 1
    suma = 0
    numeros = linea.split(" ")
    for numero in numeros:
        suma += int(numero)
    print("La línea %d suma %d" % (numLinea, suma))
```

Ejercicio 1:

Crea un programa llamado **FicheroPersonas.py** que lea información de personas (nombre y edad) de un fichero de texto, y muestre por pantalla los datos de la persona más joven y más vieja del fichero. El formato del fichero será como el siguiente, y se deberá almacenar en una lista antes de procesar la información.

```
Nacho;44
Juan;70
Mario;9
Laura;6
Ana;40
```

AYUDA: vídeo con la solución del ejercicio

2. Gestión del sistema de ficheros

Veremos a continuación algunas instrucciones útiles para realizar operaciones típicas sobre el sistema de ficheros. Casi todas ellas forman parte del paquete `os` de Python, por lo que deberemos incorporarlo al inicio de nuestro código, con `import os`. Otras funciones que usaremos pertenecen a otras librerías, como `shutil`, también disponibles en el núcleo de Python.

- Para **crear un directorio o carpeta** usaremos la función `os.mkdir(ruta)`
- Para **listar ficheros y carpetas de una carpeta** usamos la función `os.listdir(ruta)`. Se mostrará el listado de recursos que están directamente en la carpeta indicada (no en subcarpetas)
- Para **borrar un fichero** usaremos el método `os.remove(ruta)`. En el caso de una carpeta, usaremos `os.rmdir(ruta)`, siempre que la carpeta esté vacía.
- Para **copiar un fichero o carpeta** de una ruta a otra, usaremos el método `copy` del paquete `shutil`, indicando ruta origen y destino a copiar.

- Para **comprobar si un recurso existe**, disponemos de `os.path.exists(ruta)`
- Para **determinar si un recurso es fichero o carpeta** podemos usar los métodos `is_dir` e `is_file` del elemento `Path` del módulo `pathlib`. También aquí disponemos del método `iter_dir` para iterar el contenido de una carpeta.

Aquí vemos un ejemplo de algunas de estas funciones:

```
import os
import shutil
from pathlib import Path

# Creamos carpeta D:\pruebas
# La 'r' delante del texto indica que se interprete como
# un texto literal, respetando la barra \
if not os.path.exists(r'D:\Pruebas'):
    os.mkdir(r'D:\Pruebas')

# Listamos ficheros y carpetas de C:\Usuarios\Nacho
path = Path(r'C:\Users\Nacho')
for elemento in path.iterdir():
    if elemento.is_dir():
        print('Carpeta', elemento)
    else:
        print('Fichero', elemento)

# Copiamos fichero "prueba.txt" de un lugar a otro
origen = r'C:\Users\Nacho\prueba.txt'
destino = r'D:\Pruebas\prueba_copia.txt'
shutil.copy(origen, destino)

# Borramos fichero "prueba.txt" de una carpeta
os.remove(r'C:\Users\Nacho\prueba.txt')
```