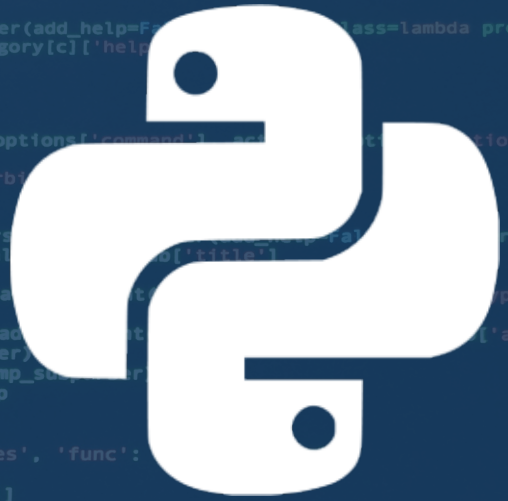


ENTRADA Y SALIDA DE FICHEROS

PROGRAMACIÓN III

Abdel G. Martínez L.

CONCEPTO Y DEFINICIÓN

- Hasta el momento toda entrada de datos ha sido a través del teclado y toda salida de datos, en pantalla
- Python provee funcionalidades básicas e integradas para manipular objetos de ficheros
- No se necesita importar ninguna librería

ABRIR UN FICHERO

- Antes de poder leer o escribir en un archivo, debemos abrirlo
- Python implementa la función `open()`
- Es una función que retorna un objeto fichero
- Utiliza dos argumentos:
  - *Nombre del fichero*
  - *Modo de apertura*

```
>>> archivo = open('data.txt', 'w')
```

MODOS DE APERTURA

Texto	Descripción
`r`	Sólo lectura
`r+`	Lectura y escritura
`w`	Sólo escritura
`w+`	Lectura y escritura
`a`	Adjuntar
`a+`	Adjuntar y lectura

Binario	Descripción
`rb`	Sólo lectura
`rb+`	Lectura y escritura
`wb`	Sólo escritura
`wb+`	Lectura y escritura
`ab`	Adjuntar
`ab+`	Adjuntar y lectura

CREAR UN FICHERO

- En el siguiente ejemplo crearemos un fichero llamado `prueba.txt`

```
>>> archivo = open('prueba.txt', 'w')
>>> archivo.write("Nuevo archivo\n")
>>> archivo.write("de dos lineas\n")
>>> archivo.close()
```

LEER UN FICHERO

- En el siguiente ejemplo leeremos el fichero llamado `prueba.txt`

```
>>> archivo = open('prueba.txt', 'r')
>>> print(archivo.read())
```

- Ahora, los 5 primeros caracteres de datos

```
>>> archivo = open('prueba.txt', 'r')
>>> print(archivo.read(5))
```

ITERAR SOBRE UN OBJETO DE FICHERO

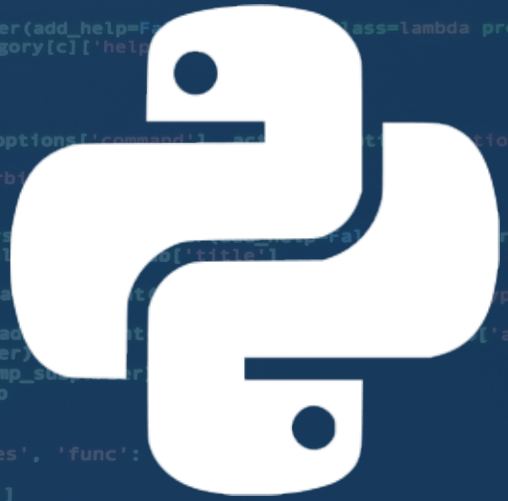
- Para leer cada línea de un fichero, podemos iterar

```
>>> archivo = open('prueba.txt', 'r')
>>> for linea in archivo:
...   print linea
```

CERRAR UN FICHERO

- Cuando se termina de trabajar sobre un fichero, debemos cerrarlo
- Esto libera recursos tomados al abrir el fichero
- Cualquier posible acción, luego de cerrar el fichero, fallará
- La función para cerrar un fichero es `close()`

```
>>> archivo = open('prueba.txt', 'r')
>>> archivo.close()
```



MÓDULOS

PROGRAMACIÓN III

Abdel G. Martínez L.

CONCEPTO Y DEFINICIÓN

- No desarrollemos aplicaciones complejas sin organizar en funciones
- Un módulo es un grupo de funciones alojadas en un archivo `.py`
- Es como un conjunto de herramientas que podemos usar siempre

EJEMPLO

- En un archivo llamado `finanzas.py` creamos la siguiente función

```
def calcularImpuesto(precio, impuesto):
    total = precio + (precio * impuesto)
    return total
```

IMPORTAR UN MÓDULO

- Existen dos posibilidades:
  - `import`
    - Manera más simple y fácil de utilizar
    - Importa todas las funciones dentro del módulo

```
import finanzas
```

IMPORTAR UN MÓDULO

- Existen dos posibilidades:
  - `from`
    - Es más recomendable
    - Ahorra tiempo de procesamiento
    - Importa funciones específicas del módulo

```
from finanzas import calcularImpuesto
```

USO DE MÓDULOS

- Una vez importados, la forma de utilizarlos es el nombre del módulo, un punto, el nombre de la función y los paréntesis (junto a sus argumentos)

```
finanzas.calcularImpuestos(50, 0.1)
```

MÓDULOS INCORPORADOS

- Existen módulos ya preexistentes en Python
- Son más de 60,000 disponibles
- Los más importantes son:
  - `math`: Provee acceso a funciones y constantes matemáticas
  - `os`: Manipula comandos de sistema operativo
  - `datetime`: Contiene información de fechas y tiempos