
Python

ARCHIVOS

ARCHIVOS

- Datos que se encuentran en sistemas de almacenamiento secundario como los discos
- Mantienen la información aún cuando se apague la computadora

■ Pasos para trabajar con archivos

- 1. Abrir archivo: para indicar que vamos a hacer operaciones sobre él.
- 2. Escribir/Leer datos: mientras el archivo esté abierto se puede hacer operaciones.
- 3. Cerrar el archivo: cuando el archivo no se va a utilizar más.

ARCHIVOS Y FLUJOS DE DATOS

- Python ve cada archivo como un flujo secuencial de caracteres así:



Cada archivo finaliza con un carácter de fin de archivo (EOF: End Of File)

- Hay diferentes tipos de archivos:
 - Texto: strings
 - Texto organizados en líneas: líneas de strings
 - Binarios: números, secuencias, imágenes, etc.

ABRIR ARCHIVOS

- Es la primera operación que se debe hacer antes de usar el archivo:

- Función “open”

open (nombreArchivo [, modo])

- Argumentos de la función open
 - nombre del archivo
 - modo en que se abre: lectura o escritura. Si el modo no se da entonces se asume que el archivo se va a leer (modo "r")
- Cuando se abre un archivo en el programa se crea un objeto tipo archivo el cual se asigna a una variable.
 - Esta variable se usa posteriormente en el programa para hacer referencias al archivo
- Ejemplo: **f = open("datos.txt", "r")**

MODOS PARA ABRIR UN ARCHIVO

Valor Descripción

- “r” Modo lectura: para leer el archivo. Si el archivo no existe se genera la excepción “IOError”
- “w” Modo escritura: para escribir en el archivo.
IMPORTANTE: si el archivo ya existe lo primero que hace Python es borrarlo
- “a” Modo de agregado (“append”): para agregar datos al final de un archivo existente. Si el archivo no existe lo crea

Valor

Descripción

- “b”
Modo binario (no string): se usa en conjunto con otro modo para manejar archivos que no sean grabados como string (ejemplos: imágenes, sonidos, secuencias, etc.)
- “rb”
Abre un archivo binario en modo de lectura
- “wb”
Abre un archivo binario en modo de escritura

■ Ejemplos

```
>>> f1 = open("datos.dat","r")
```

Traceback (most recent call last):

File "<pyshell#35>", line 1, in <module>

f1=open("datos.dat","r")

File "C:\Python30\lib\io.py", line 278, in __new__

return open(*args, **kwargs)

File "C:\Python30\lib\io.py", line 222, in open

closefd)

File "C:\Python30\lib\io.py", line 619, in __init__

_fileio._FileIO.__init__(self, name, mode, closefd)

IOError: [Errno 2] No such file or directory: 'datos.dat'

```
>>> f = open("datos.dat","w")
```

GRABAR DATOS EN ARCHIVOS DE TEXTO

■ Método write

objetoArchivo.**write**(string)

➤ Graba string en el archivo

➤ El archivo debe estar abierto en modo "w"

```
>>> f = open("test.dat","w")
```

```
>>> f.write("Estamos grabando datos")
```

```
>>> f.write("Grabamos mas datos")
```

CERRAR ARCHIVOS

- Método close

objetoArchivo.**close()**

➤ Se le indica al sistema que se ha terminado de usar el archivo indicado

- Ejemplo:
f.close()

LEER DATOS DE LOS ARCHIVOS

- Método `read`
`objetoArchivo.read()`
 - El archivo debe estar abierto en modo "r"
 - Leer todo el contenido del archivo
- ```
>>> f = open("test.dat","r")
>>> texto = f.read()
>>> print (texto)
```
- Estamos grabando datosGrabamos mas datos

- 
- Al método “read” puede especificarse un argumento para indicarle cuántos caracteres se deben leer

---

```
>>> f=open("test.dat","r")
```

```
>>> texto=f.read(22)
```

```
>>> print (texto)
```

Estamos grabando datos

```
>>> texto=f.read(100)
```

```
>>> print (texto)
```

Grabamos mas datos

Cuando no existan mas caracteres que leer, el método "read" retorna una hilera (string) vacía (""), lo cual indica que es fin de archivo (EOF: End Of File)

```
>>> texto=f.read(100)
```

```
>>> print (texto)
```

```
>>>
```

# ARCHIVOS DE TEXTO

## ORGANIZADOS POR LÍNEAS

---

- Estos archivos de texto están organizados en líneas separadas por el carácter de nueva línea (“\n”)
- “readline”: método para leer una línea incluyendo el carácter de nueva línea
- Cuando no hayan más líneas para leer (al final del archivo) el método “readline” retorna una hilera (string) vacía

```
>>> f=open("test.dat","w")
>>> f.write("Línea 1\nLínea 2\nLínea 3\n")
>>> f.close()
>>> f=open("test.dat","r")
>>> print (f.readline())
```

Línea 1

← "\n" caracter de nueva línea

```
>>>
```



- 
- “readlines”: retorna todas las líneas del archivo como una lista de strings o hileras junto con el carácter de nueva línea
  - Al final del archivo el método retorna una lista vacía ([])

---

```
>>> f = open("test.dat","r")
>>> print (f.readlines())
['Línea 1\n', 'Línea 2\n', 'Línea 3\n']
```

```
>>> print (f.readlines())
[]
>>>
```

# Grabando datos no string en los archivos de texto

---

- El argumento del método "write" debe ser un string
- Grabar datos que no sean del tipo string: primero hay que pasarlos a string (función "str")

```
>>> f=open("test2.dat","w")
```

```
>>> f.write(52)
```

Traceback (most recent call last):

File "<pyshell#83>", line 1, in <module>  
f.write(52)

File "C:\Python30\lib\io.py", line 1487, in write  
s.\_\_class\_\_.\_\_name\_\_)

TypeError: can't write int to text stream

```
>>> x=52
```

```
>>> f.write(str(x))
```

```
2
```

Para pasar un dato leído (tipo string) a su tipo original (entero, lista, etc.) se puede usar la función "eval"

```
x = eval(y)
```

# EL MÓDULO PICKLE

---

- Este módulo permite grabar y leer objetos de cualquier tipo de datos sin tener que hacer conversiones de tipos de datos
- Para usar el módulo primero hay que importarlo:

```
import pickle
```

# Grabar datos

---

- Método para grabar: **dump**

`pickle.dump(objetoAgrabar, objetoArchivo)`

El archivo se abre en modo binario

Ejemplo:

```
a = open("datos.dat", "wb")
pickle.dump([1,2,3], a)
```

# Leer datos

---

- El archivo se abre en modo binario
- Método para leer: **load**. Cada vez que se usa este método se lee un valor del archivo

`pickle.load(objetoArchivo)`

Ejemplo: `x = pickle.load(a)`

- Cuando ya no hay más datos para leer encuentra el fin de archivo y da la excepción: `EOFError`

## ■ Ejemplo para grabar datos y luego leer todos esos datos usando pickle

```
import pickle # importar el módulo
f=open("test.dat","wb") # abrir archivo para grabar datos
pickle.dump([1,2,3],f) # grabar una lista
pickle.dump(56.7,f) # grabar un flotante
f.close() # cerrar archivo
print ("Leer archivo")
f=open("test.dat","rb") # abrir archivo para leer datos
while True:
 try: # controlar EOF
 x=pickle.load(f) # leer datos
 print (x)
 except EOFError:
 break
f.close() # cerrar archivo
print ("Fin")
```

Leer archivo

[1, 2, 3]

56.7

Fin

>>>