Escuela Politécnica Superior

PROGRAMACIÓN
Grado en Ingeniería Biomédica

TEMA . 7

Python – Entrada/Salida

Archivos externos

- El comando print() permite generar información que necesite ser manipulada inmediatamente y se envía al panel de comando de salida ➔ Monitor.
- Sin embargo, en ocasiones es necesario generar información que necesita ser guardada en el disco duro.
- Con frecuencia los ficheros de texto generados se utilizan como datos de entrada (*input*) para un procesamiento posterior por otro programa o conjunto de programas.

Archivos en Python

Abrir archivos en python mediante el objeto file

- Podemos manipular una **variable** como un **objeto file** (archivo) asignándole un archivo mediante la función integrada **open** e indicando **la ruta del mismo seguido del modo en el cual vamos a abrirlo**.
- La variable toma como valor el archivo y se convierte en un **objeto file** permitiéndonos trabajar con ella utilizando los **métodos**.
- Existen dos formas básicas de acceder a un archivo, una es utilizarlo como un **archivo de texto**, que procesaremos línea por línea; la otra es tratarlo como un **archivo binario**, que procesaremos byte por byte.

Archivos en Python

```
fichero=open("fibo.txt", "w")
numero=int(input("Dime el número de elementos de la serie de Fibonacci: "))

lista=[0,1]
for i in range(2,numero):
    lista=lista+[lista[i-1]+lista[i-2]]

fichero.write("Serie Fibonacci de "+str(numero)+ "  \n")
fichero.write(str(lista))
fichero.close()
```

Apertura del fichero en modo escritura (write). La variable fichero es un objeto file.

Se utiliza el método write() para escribir una cadena en el fichero

Todo fichero que se abre debe ser cerrado. Método close() para cerrar el fichero abierto con el método open()

Modo de apertura de archivos

- Lo primero es preguntarnos, **¿para qué vamos a abrir un fichero?**

Por ejemplo, podemos querer abrir un archivo para leerlo, para escribirlo, para leerlo y escribirlo, para crearlo si no existe y luego escribir en él, etc.

- Cada vez que abrimos un archivo estamos creando un ***puntero*** que**:**
  - dependiendo de cómo se abra el archivo el puntero se posicionará en un lugar determinado (al comienzo o al final)
  - el puntero podrá moverse dentro del archivo, eligiendo su nueva posición, mediante el número de byte correspondiente.

Modos apertura de un archivo de texto

Indicador	Modo de apertura	Ubicación del puntero
r	Solo lectura	Al inicio del archivo
r+	Lectura y escritura	Al inicio del archivo
w	Solo escritura. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
w+	Escritura y lectura. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
a	Añadido (agregar contenido). Crea el archivo si éste no existe	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo
a+	Añadido (agregar contenido) y lectura. Crea el archivo si éste no existe.	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo

```
fichero=open("fibo.txt", "w")
```

Modos apertura de un archivo binario

Indicador	Modo de apertura	Ubicación del puntero
rb	Solo lectura en modo binario	Al inicio del archivo
rb+	Lectura y escritura en modo binario	Al inicio del archivo
wb	Solo escritura en modo binario. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
wb+	Escritura y lectura en modo binario. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
ab	Añadido (agregar contenido) en modo binario. Crea el archivo si éste no existe	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo
ab+	Añadido (agregar contenido) y lectura en modo binario. Crea el archivo si éste no existe.	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo

```
fichero=open("fibo.txt", "wb")
```

Métodos del objeto file

buffer
close
closed
detach
encoding
errors
fileno
flush
isatty
line_buffering
mode
name
newlines
read
readable
readline
readlines
reconfigure
seek
seekable
tell
truncate
writable
write
write_through
writelines

Métodos del objeto archivo

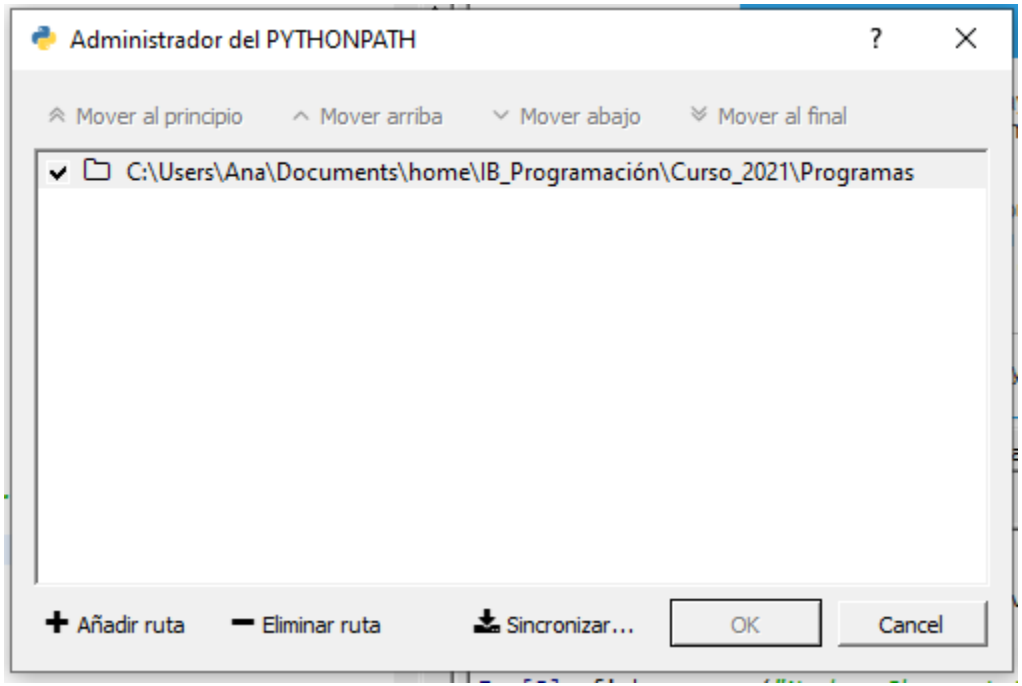
- Cuando abrimos un archivo existe un puntero, que es donde se posiciona el intérprete de Python sobre el archivo.
- En el momento de la apertura el puntero se posiciona al inicio y a medida que trabajamos con el archivo el puntero va modificando su posición.
- Método read([bytes]): permite leer un archivo completo, salvo que indiquemos los bytes entonces solo leerá hasta los bytes determinados.

Modo lectura

```
In [8]: fichero=open("Noches_Claras.txt", "r")

In [9]: poesia=fichero.read()

In [10]: poesia
Out[10]: 'En las noches claras,\nresuelvo el problema de la soledad del ser.\nInvito
a la luna y con mi sombra somos tres.'

In [11]: print(poesia)
En las noches claras,
resuelvo el problema de la soledad del ser.
Invito a la luna y con mi sombra somos tres.
```

Métodos del objeto archivo

- Si el módulo/fichero no es encontrado en el directorio actual, Python entonces busca en cada directorio que se encuentra en la variable de entorno PYTHONPATH.
- PYTHONPATH es una variable de entorno del sistema operativo, consistiendo de una lista de directorios.



Métodos del objeto archivo

- Método **readline([bytes]):**
  - El método readline lee una línea por vez, la línea se recoge como una cadena.
  - Un readline() leerá una sola línea y si no especificamos a partir de donde será la primera por defecto.
  - Cuando se llega al final del fichero nos devolverá una línea vacía "".
  - Una línea en blanco en medio del fichero nos sería devuelta como un "\n", no como una línea vacía "".

```
fichero=open("Piratas.txt")
linea=fichero.readline()
while linea !="":
    print(linea)
    linea=fichero.readline()
fichero.close()
```

Siempre hay que cerrar un fichero. Cuando no se cierra cualquier intento posterior de usar el fichero podría dar errores.

Métodos del objeto archivo

- Método **readlines():** lee todas las líneas en forma de lista de cadenas separando las líneas con el carácter de escape para los saltos de línea "\n"

```
fichero=open("Noches_Claras.txt", "r")

lista=fichero.readlines()

print("Imprimiendo la lista:")
print(lista)

print("\nImprimiendo elemento a elemento la lista:")
for i in lista:
    print(i)
```

```
Imprimiendo la lista:
['En las noches claras,\n', 'resuelvo el problema de la soledad del ser.\n', 'Invito a la
luna y con mi sombra somos tres.']

Imprimiendo elemento a elemento la lista:
En las noches claras,

resuelvo el problema de la soledad del ser.

Invito a la luna y con mi sombra somos tres.
```

Métodos del objeto archivo

- Método **read([bytes]):** lee todo el contenido de un fichero.
  - Si no tiene argumentos, lee hasta el final, quedando el puntero del fichero al final del fichero
  - Si tiene como argumentos un número de bytes, leerá solo el contenido hasta la longitud dada (número de bytes)

```
# Apertura de ficheros
fichero = open("Piratas.txt","r")
fichero_out=open("Piratas_upper.txt","w")

contenido=fichero.read() #Leer el fichero al completo

#Convierto todo a mayúsculas
new_contenido=contenido.upper()
fichero_out.write(new_contenido)

#Cierre de los ficheros
fichero.close()
fichero_out.close()
```

Métodos del objeto archivo

- Método **write():** escribe una cadena en un fichero abierto en modo escritura

```
numero=int(input("Dime que tabla de multiplicar quieres saber: "))
tabla="Tabla"+str(numero)+".txt"
fichero=open(tabla,"w")
cadena="Tabla del número "+str(numero)+"\n"
fichero.write(cadena)
for j in range(1,11):
    valor=numero*j
    fichero.write(str(numero)+ "x"+str( j) + "=" + str(valor)+"\n")
fichero.close()
```

Apertura en modo escritura

w	Solo escritura. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
w+	Escritura y lectura. Sobreescribe el archivo si existe. Crea el archivo si no existe	Al inicio del archivo
a	Añadido (agregar contenido). Crea el archivo si éste no existe	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo
a+	Añadido (agregar contenido) y lectura. Crea el archivo si éste no existe.	Si el archivo existe, al final de éste. Si el archivo no existe, al comienzo

Métodos del objeto archivo

- Método **writelines**() escribe una lista de cadenas a un archivo.

```
# Declara lista
lista = ['lunes\t', 'martes\t', 'miercoles\t', 'jueves\t', 'viernes\t']

# Abre archivo en modo escritura
archivo = open('semana.txt','w')

# Escribe toda la lista en el archivo
archivo.writelines(lista)

# Cierra archivo
archivo.close()
```

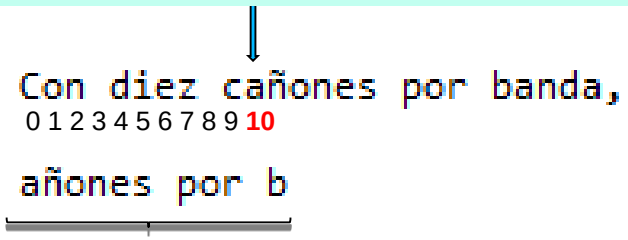
Métodos del objeto archivo

- Método **seek(byte):** mueve el puntero hacia el byte indicado

```
fichero = open("Piratas.txt","r")
linea=fichero.readline() # Lectura de la primera línea
print(linea)
fichero.seek(10)         # Coloca el puntero en el byte numero 10
linea=fichero.read(12)   # Lee 12 bytes que equivalen a 12 caracteres
print(linea)             # Imprime los 12 caracteres que ha leido
fichero.close()
```

Byte número 10, comenzando a contar por 0

El tamaño de un carácter es un byte



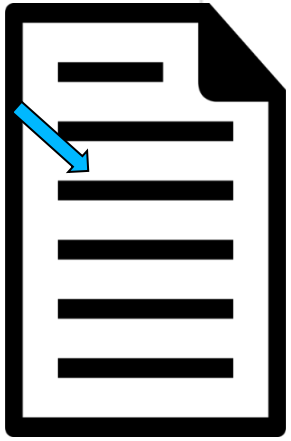
12 Bytes == 12 Caracteres

Métodos del objeto archivo

- Método **tell( ):** devuelve la posición en la que se encuentra el puntero en el fichero.
- La combinación de los métodos seek() y tell() permite posicionarnos en el fichero para poder leer/escribir.

```
fichero = open("Piratas.txt","r")
linea=fichero.readline() # Lectura de la primera línea
print(linea)
fichero.seek(10)         # Coloca el puntero en el byte numero 10
linea=fichero.read(12)   # Lee 12 bytes que equivalen a 12 caracteres
print(linea)             # Imprime los 12 caracteres que ha leido
print("Nueva posición",fichero.tell()) # Imprime la nueva posición del fichero
fichero.close()
```

```
Con diez cañones por banda,

añones por b
Nueva posición 22
```



Atributos del objeto archivo

- Una vez que un fichero está abierto y se tiene un objeto file se tiene información relacionada con ese fichero.
- Atributos del objeto file:
  - fichero.closed, devuelve True si el fichero está cerrado, False en caso contrario.
  - f.mode, devuelve el modo de acceso con el que el fichero ha sido abierto.
  - f.name, devuelve el nombre del fichero.
- Observa que los atributos no tienen () como ocurren en los métodos

Atributo	**Método**
fichero.closed	fichero.close()
fichero.mode	fichero.readline()
fichero.name	fichero.writeline()
	fichero.read()
	fichero.seek()

Atributos del objeto archivo

```
def fichero_abierto(fichero):
    if(not fichero.closed):
        print("Tienes un fichero abierto de nombre \"%s\" y en modo %s \n"
              %( fichero.name,fichero.mode))
    else:
        print("No tienes fichero abierto")

fichero = open("Piratas.txt","r")

print("\nMientras que el fichero está abierto")
fichero_abierto(fichero)
fichero.close()

print("\nCuando ya he cerrado el fichero")
fichero_abierto(fichero)
```

Try/Except para atrapar excepciones

- Sentencia try except:
  - Se comienza a ejecutar las instrucciones que se encuentran dentro de un bloque try.
  - Si durante la ejecución de esas instrucciones se levanta una excepción, Python interrumpe la ejecución en el punto exacto en que surgió la excepción y pasa a la ejecución del bloque except correspondiente.

```
In [40]: dividiendo=20

In [41]: divisisor=0

In [42]: try:
    ...:     resultado=dividdendo/divisor
    ...: except:
    ...:     print("No se permite la división por cero")
    ...:
No se permite la división por cero
```

Try/Except para atrapar excepciones

Ejemplo de Sintaxis:

```
try:
    # aquí ponemos el código que puede lanzar excepciones
except IOError:
    # entrará aquí en caso que se haya producido
    # una excepción IOError
except ZeroDivisionError:
    # entrará aquí en caso que se haya producido
    # una excepción ZeroDivisionError
except:
    # entrará aquí en caso que se haya producido
    # una excepción que no corresponda a ninguno
    # de los tipos especificados en los except previos
finally:
    #se ejecuta siempre, haya surgido una excepción o no
    #son típicamente acciones de limpieza
```

Excepción Input/Output

Sin especificar la excepción

Opcional

Try/Except para atrapar excepciones

```
def apertura_fichero(nombre, modo):
    try: # Abajo el código que podría generar una excepción
        fichero=open(nombre, modo)
        return fichero
    except IOError: #Procesamiento de la excepción
        print("Error de apertura del fichero ", nombre)
        return

import sys  #Permite llamar a la función exit
nombre_fichero="Pratas.txt"
modo_apertura="r"
fichero=apertura_fichero(nombre_fichero, modo_apertura)
if(fichero==None):
    sys.exit()

contenido=fichero.read()
print(contenido)
fichero.close()
```