Python 3

8. Entrada/Salida y trabajo con ficheros

Carolina Mañoso, Ángel P. de Madrid y Miguel Romero

Índice

- Entrada estándar
- Salida estándar
- Archivos
  - Abrir y cerrar
  - Lectura
  - Escritura
  - Mover el puntero
  - Lectura desde la web

Entrada estándar

- La forma más sencilla de obtener información por parte del usuario es mediante la función `input()`.
  - Esta función toma como parámetro una cadena, para usar como texto a mostrar al usuario, pide la entrada y *devuelve una cadena con los caracteres introducidos* por el usuario hasta que pulsa la tecla `Enter`.

```
nombre =input("Introduce tu nombre: ")
print("Hola", nombre)
```

  - Si lo que se necesita es otro tipo de datos debemos usar la función conversora que corresponda:

```
edad = int(input('Teclear edad: '))  # entrada de entero
peso = float(input('Teclear peso: '))  # entrada de float
```

Salida estándar (1/4)

- La función `print()` muestra la cadena que encierra entre los paréntesis, omitiendo las comillas que la encierran, en la salida estándar que, normalmente, se corresponde con la pantalla de un ordenador.
  - Después de escribir la cadena el puntero se sitúa en la siguiente línea de la pantalla.
  - En Python 2 no hacen falta paréntesis.
- También produce como salida el valor de una variable pasada como argumento:

```
>>> nombre = 'Felipe'
>>> print(nombre)
```

- Se pueden escribir varias cadenas y variables pasadas todas como argumentos (separadas por comas):

```
>>> print('Hola', nombre, '¿qué tal?')
```

Salida estándar (2/4)

- Si la cadena ocupa varias líneas se encierra entre triple entrecomillado.

```
>>> print("""primera línea
segunda línea""")
```

- `\n` indica cambio de línea.

```
>>> print('primera línea\n segunda línea')
```

- `\t` introduce un tabulador.

```
>>> print('primera parte\t segunda parte')
```

- La función `print()` tiene dos argumentos reservados:
  - `end = " "` (por defecto es nueva línea)
  - `sep = " "` (por defecto espacio)

```
>>> print("hola","hola","hola",sep="***",
  end="...")
>>> print("hola","hola","hola")
```

Salida estándar (3/4)

- Para que la salida esté formateada, se usa el *método* `str.format()`
  - En el `string`, las llaves y caracteres dentro de las mismas (llamados campos de formato) son reemplazadas con los objetos pasados en el método `str.format()`.

```
>>>print('pedro tiene {}'.format('gatos'))
>>>print('pedro tiene {} y {}'.format('gatos', 'perros'))
```

  - Un número en las llaves se refiere a la posición del objeto pasado en el método.

```
>>> print('{0} y {1}'.format('hola', 'adios'))
>>> print('{1} y {0}'.format('hola', 'adios'))
```

  - Si se usan argumentos nombrados en el método `str.format()`, sus valores serán referidos usando ese nombre.

```
>>>print('El télefono de {nombre} es {num}'.format(nombre='Paco', num='666666'))
```

Salida estándar (4/4)

- Podemos utilizar especificadores de formato: `{nombre: conversion}` (`s` para string, `d` para enteros decimales, `f` para floats).

```
>>>num = 3.14159
>>>print('El valor de num es {0:f}'.format(num))
```

- Podemos indicar el número de decimales:

```
>>>print('El valor num es aprox {0:.2f}'.format(num))
```

- Para limitar que el campo sea de un mínimo número de caracteres de ancho se pasa un entero después ':'

```
>>> nombre = "Pedro"
>>> telf = 456777
>>>  print('{0:10} ==> {1:10d}'.format(nombre, telf))
```

  - Esto es útil para hacer tablas.

Práctica (1/2)

- Introduzca en el intérprete los siguientes comandos. Razone la salida que se produce:

```
>>> int("dos")
>>> print(str(3+3)+"3")
>>> type(3>2)
>>> type(3=3)
>>> "Hola"[4]
>>> "Hola"[-4]
>>> "Hola"[1:3]
>>> "hola"[:3]
>>> "hola"[3:]
>>> 2345[4]
>>> (3>2) or (2>3)
>>> str((not True) and (not False))
>>> 10%3
>>> 10//3
```

Práctica (2/2)

```
>>> x = input("Dame un entero: ") # x es una cadena
>>> print("x =", x)
>>> print("Su cuadrado vale:", x**2)   # Da un error
>>> print("Su cuadrado vale:", int(x)**2)   # OK
>>> print("Su cuadrado vale (como flotante):
{0:6.3f}".format(int(x)**2))
>>> x = int(input("Dame otro entero: ")) # x es un entero
>>> print("Su raíz cuadrada es:", x**0.5)
>>> print("Su raíz cuadrada es: {0:5.3}".format(x**0.5))
>>> print('En un lugar de La Mancha\nde cuyo nombre…')
>>> print('En un lugar de La Mancha\tde cuyo nombre…')
>>> print('En un lugar', 'de La Mancha', sep=" ", end="...")
>>> print('En un lugar', 'de La Mancha', sep="... ",
end="!!!")
>>> x = '''En un lugar
de La Mancha'''
>>> print(x)
```

Archivos (1/10)

- Los *ficheros* en Python son objetos de tipo `file` creados mediante la función `open()`.
  - Esta función toma como parámetros el nombre del archivo seguido, opcionalmente, por el modo o tipo de operación a realizar y la codificación que tendrá el archivo.

```
f = open('nombre_archivo', 'modo')
```

  - Si no se indica el tipo de operación el archivo se abrirá en modo de lectura y si se omite la codificación se utilizará la codificación actual del sistema.
  - Si no existe la ruta del archivo o se intenta abrir para lectura un archivo inexistente se producirá una *excepción* del tipo `IOerror`.

```
Objf=open('/home/archivo.txt')
Objf=open('/home/archivo.txt', 'r')
Objf=open('/home/archivo.txt', mode='r', encoding='utf-8')
```

Archivos (2/10)

- El modo de acceso puede ser cualquier combinación de:
  - `'r'`: *read*, leer. Abre el archivo en modo lectura. El archivo tiene que existir previamente, en caso contrario se lanzará una *excepción* de tipo `IOError`.
  - `'w'`: *write*, escribir. Abre el archivo en modo escritura. Si el archivo no existe, se crea. Si existe, se sobreescribe el contenido.
  - `'a'`: *append*, añadir. Abre el archivo en modo escritura. Se diferencia del modo `'w'` en que en este caso no se sobreescribe el contenido del archivo, sino que se comienza a escribir al final del archivo.
  - `'b'`: *binary*, binario.
  - `'r+'`: permite lectura y escritura simultáneas.

- Una vez terminemos de trabajar con el archivo debemos cerrarlo con el método `close()`.

Archivos (3/10)

- Para la lectura de archivos se utilizan los métodos `read()`, `readline()` y `readlines()`:
  - El método `read()` devuelve una cadena con el contenido del archivo o bien el contenido de los *n* primeros bytes, si se especifica el tamaño máximo a leer.

```
archivo = open('archivo.txt','r')  # Abre archivo en modo lectura
cadena1 = archivo.read(9)  # Lee los 9 primeros bytes
cadena2 = archivo.read()  # Lee la información restante
cadena3 = archivo.read()  # Se ha alcanzado el final del archivo
```

Nota: Crear un archivo llamado `archivo.txt` y desde la ventana de comandos ejecutar estas acciones y observar el resultado.

Archivos (4/10)

- El método `readline` sirve para leer las líneas del archivo *una por una*, es decir, devuelve el contenido desde el puntero hasta donde se encuentra un carácter de nueva línea, *incluyendo ese carácter* (a excepción de la última línea del archivo, si éste no lo contuviera).

```
>>>archivo = open('archivo.txt','r')  #abre para lectura
>>>cadena1 = archivo.readline()  # … lee primera línea
>>>cadena2 = archivo.readline()  # … lee segunda línea
>>> …
```

Nota 1: Probar que lea la última línea que tiene el archivo.
Nota 2: Probar que lea más líneas de las que tiene el archivo.

- Para leer líneas de un archivo se puede iterar sobre el objeto archivo.

```
archivo = open('archivo.txt','r')  # abre en modo lectura
for linea in archivo:
        print(linea, end=' ')
```

Archivos (5/10)

- El método `readlines` funciona leyendo todas las líneas del archivo y *devolviendo una lista* con ellas.

```
# Leemos todas las lineas
archivo = open('archivo.txt','r')
lineas = archivo.readlines()  # Lee todas la líneas a una lista
numlin = 0   # Inicializa un contador
for linea in lineas:  # Recorre todas los elementos de la lista
    numlin += 1   # Incrementa en 1 el contador
    print(numlin, linea)# Muestra contador y elemento de lista, línea
archivo.close()# Cierra archivo
```

Archivos (6/10)

- La declaración `with` tiene la característica de que el archivo es cerrado de forma automática cuando el bloque termina, incluso si ha habido una excepción:

```
with open('nombre_archivo', 'modo') as variable:
    bloque_de_instrucciones
```

Ejemplo:

```
with open('archivo.txt','r') as archivo:
    lineas = archivo.readlines()
```

Nota: Probar desde la línea de comandos.

Archivos (7/10)

- Para la escritura se utilizan los métodos `write()` y `writelines()`:
  - `f = write('cadena')` escribe en el archivo una cadena de texto que toma como parámetro. Devuelve el número de caracteres escrito.
  - El método `writelines()` escribe una lista de cadenas de texto indicando las líneas que queremos escribir.

Archivos (8/10)

```
cadena1 = 'Datos'        # Declara cadena1
cadena2 = 'Secretos'     # Declara cadena2
archivo = open('datos1.txt','w') # Abre archivo para escribir
archivo.write(cadena1 + '\n') # Escribe cadena1 con salto de línea
archivo.write(cadena2)   # Escribe cadena2 en archivo
archivo.close()          # Cierra archivo

lista = ['lunes', 'martes', 'miercoles', 'jueves', 'viernes']
# Declara lista
archivo = open('datos2.txt','w') # Abre archivo en modo escritura
archivo.writelines(lista) # Escribe toda la lista en el archivo
archivo.close()           # Cierra archivo
```

Archivos (9/10)

- Para mover el *puntero de lectura/escritura* a una posición determinada está el método `seek()`:
  - Toma como parámetro un número que indica el desplazamiento. También se le puede indicar desde donde queremos que se haga el desplazamiento: 0 desde el principio, 1 posición actual y 2 final.
  - El método `tell()` determina la posición actual del puntero.

```
archivo = open('datos2.txt','r')  # Abre archivo en modo r
archivo.seek(5)  # Mueve puntero al quinto byte
cadena1 = archivo.read(5)  # Lee los siguientes 5 bytes
print(cadena1)             # Muestra cadena
print(archivo.tell())      # Muestra posición del puntero
archivo.close()            # Cierra archivo
```

Archivos (10/10)

- Para leer un archivo de la web necesitamos el módulo `urllib.request`:
  - Utilizamos la función `urlopen(url)`.

```
from urllib.request import urlopen
f = urlopen('http://gaia.cs.umass.edu/wireshark-labs/alice.txt')
linea = f.readline()
print(linea)
```

```
import urllib.request
url = 'http://jidx.org/ardxjidx.dat'
with urllib.request.urlopen(url) as webpage:
    for line in webpage:
        print(line)
```

Aviso

Esta colección de transparencias se distribuye con fines meramente docentes.

Todas las marcas comerciales y nombres propios de sistemas operativos, programas, hardware, etc. que aparecen en el texto son marcas registradas propiedad de sus respectivas compañías u organizaciones.