Entrada / Salida

Antonio Espín Herranz

Salida por consola

- **print**
  - Imprimir el contenido de variables.
  - Texto fijo entrecomillas.
  - Paréntesis de print obligatorios.
  - Por defecto imprime y salta de línea.
  - Imprimir varios textos o variables separadas por comas.
    - print (‘Hola’, ’mundo’) **# las separa por un espacio en blanco**
    - Hola mundo
  - Pero también se puede utilizar parámetros de una forma similar a printf de C.

    print (‘%d y %d’ % (x,y))

  - Dentro de estos parámetros se puede utilizar números: %6.2f

Salida por consola

- Ejemplo con salto de línea:

```
for i in range(3):
    print( i)
```

- Salida:

```
0
1
2
```

- Ejemplo sin salto de línea:

```
for i in range(3):
    print (i,end=" ")
```

- Salida:

```
0 1 2
```

Salida por consola

- **print al estilo C**:
  - print (“Hola %s” **%** “mundo”)
  - print (“%s %s” **%** (“Hola”, “mundo”))

Especificador	Formato
%s	Cadena
%d	Entero
%o	Octal
%x	Hexadecimal
%f	Real

>>> print (“**%10s** mundo” % “Hola”)

_____Hola mundo

>>> print (“**%-10s** mundo” % “Hola”)

Hola_____mundo

Salida por consola

- En el caso de los reales es posible indicar la precisión a utilizar precediendo la **f** de un punto seguido del número de decimales que queremos mostrar:

```
>>> from math import pi
>>> print ("%.4f" % pi)
3.1416
```

- La **misma sintaxis** se puede utilizar para **indicar el número de caracteres** de la cadena que queremos mostrar

```
>>> print ("%.4s" % "hola mundo")  # trunca la cadena
hola
```

Entrada estándar

- **input()**
  - Para leer datos de teclado.
  - De forma opcional se puede indicar un prompt entre comillas dentro de la función.
  - Devuelve la cadena tecleada por el usuario.

- Ejemplo
  - nombre = input(‘introduce un nombre: ’)
  - print (nombre)

Entrada estándar

- La lectura de teclado implica conversiones a otros tipos como números enteros o reales.
- Ojo con las posibles excepciones:

```
try:
    edad = input(“Cuantos años tienes? “)
    dias = int(edad) * 365
    print (“Has vivido “ + str(dias) + “ dias”)

except ValueError:
    print (“Eso no es un numero”)
```

Conversiones

- **float**
  - Convertir texto o entero a float.
    - float(3) → 3.0
    - float('3.0') → 3.0
- **int**
  - Convertir texto o float a entero.
    - int(2.1) → 2
    - int("234") → 234
- **str**
  - Convertir a texto un número, objeto …
    - str(2.1): "2.1".

Parámetros en línea

- Al igual que en otros lenguajes de programación a un programa python se le pueden pasar parámetros por la línea de comandos:
  - python mi_programa.py **datos.txt**

- En este caso hola.txt sería el parámetro, al que se puede acceder a través de la lista **sys.argv** (antes importar el módulo **sys**).

- Para pasar mas de un parámetro separarlos por espacios en la línea de comandos.

Parámetros en línea

- Uso del paquete sys:
  - argv: Representan los parámetros.

- **sys.argv[0]**
  - Contiene siempre el nombre del programa tal como lo ha ejecutado el usuario.
- **sys.argv[1]**
  - Si existe, sería el primer parámetro.
- **sys.argv[2]**
  - El segundo, y así sucesivamente.

Parámetros en línea

- Ejemplo:

```
import sys
if(len(sys.argv) > 1):
    print (“Abriendo “ + sys.argv[1])
else:
    print (“Debes indicar el nombre del archivo”)
```

Ficheros

- Los ficheros en Python son **objetos de tipo file** creados mediante la función **open** (abrir).

- Parámetros de **open**(path, [modo],[tamaño]):
  - **path**:
    - Una cadena con **la ruta al fichero** a abrir, que puede ser relativa o absoluta;
  - **modo**:
    - Una cadena opcional indicando el modo de acceso (si no se especifica se accede en modo lectura)
  - **tamaño**:
    - por último, un entero opcional para especificar un tamaño de buffer distinto del utilizado por defecto.

Modos de acceso (combinar)

- **‘r’: read, lectura**.
  - Abre el archivo en modo lectura.
  - El archivo tiene que existir previamente, en caso contrario se lanzará una excepción de tipo **IOError**.
- **‘w’: write, escritura.**
  - Abre el archivo en modo escritura. Si el archivo no existe se crea. Si existe, **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.**
- **‘+’: permite lectura y escritura simultáneas.**
- **‘U’: universal newline, saltos de línea universales.**
  - Permite trabajar con archivos que tengan un formato para los saltos de línea que no coincide con el de la plataforma actual (en Windows se utiliza el caracter CR LF, en Unix LF y en Mac OS CR).

Leer ficheros

- Para la lectura de archivos se utilizan los métodos **read, readline y realines**.

- El método **read** devuelve una cadena con el contenido del archivo o bien el contenido de los primeros n bytes, si se especifica el tamaño máximo a leer.
  - A partir de f y f2 dos ficheros abiertos:
    - **completo = f.read()**
    - **parte = f2.read(512)**

- El método **readline** sirve para leer las líneas del fichero una por una.

```
while True:
    linea = f.readline()
    if not linea: break
        print (linea)
```

Leer con un iterador

```
print("Contenido del fichero")
with open("fichero") as fp:
    for  linea in iter(fp.readline,''):
        print(linea)
```

- Itera por las líneas del fichero hasta que encuentre el final.

Escribir ficheros

- Para la escritura de archivos se utilizan los método **write** y **writelines**.

- **write**:
  - Escribe en el archivo una cadena de texto que toma como parámetro.

- **writelines**:
  - Toma como parámetro una lista de cadenas de texto indicando las líneas que queremos escribir en el fichero.

seek / tell

- **seek(desplazamiento [,origen])**
  - Toma como parámetro un número positivo o negativo a utilizar como desplazamiento.
  - También es posible utilizar un segundo parámetro para indicar desde dónde queremos que se haga el desplazamiento:
    - **0** indicará que el desplazamiento se refiere al principio del fichero (comportamiento por defecto),
    - **1** se refiere a la posición actual, y
    - **2**, al final del fichero.

- **tell()**
  - **Determinar la posición** en la que se encuentra actualmente el puntero devuelve un entero indicando la distancia en bytes desde el principio del fichero.

Ejemplo

```
lista=['sota','caballo','rey']
fichero=open('prueba.txt','w')
for x in lista:
    fichero.write(x+"\n")
fichero.close()

fichero=open('prueba.txt','r')
mi_cadena=fichero.read()
fichero.seek(0) # vuelvo al princio del fichero
lista_de_cadenas=fichero.readlines()

# ahora cada elemento incluye \n
fichero.seek(0)
for linea in fichero.readlines():
    print (linea)
fichero.close()
```

Directorios

- Dentro del modulo os tenemos la función listdir(ruta) que nos devuelve una lista con todos los directorios y ficheros de una carpeta …

- Ejemplo:

```
import os
print(os.listdir())
```

Directorios II

- **Para leer los archivos de un directorio:**

La función next()
Recupera el siguiente
Elemento de un iterador

```
from os import walk

def ls(ruta = "."):
        #Devuelve una tupla de 3 elementos.
        dir, subdirs, archivos = next(walk(ruta))
        print("Actual: ", dir)
        print("Subdirectorios: ", subdirs)
        print("Archivos: ", archivos) # Devuelve la lista de archivos.

if __name__ == '__main__':
        ls()
```