Introducción al uso de archivos

En nuestros programas, estamos acostumbrados a utilizar un gran número de datos, tanto como datos de entrada como datos de salida.

Es sencillo manejar los datos de manera interactiva, pero cuando tenemos una gran cantidad de datos, esto puede resultar incómodo. Podemos leer y almacenar los datos de manera permanente, utilizando **archivos**, normalmente almacenados en el disco duro del ordenador, o en un pen-drive.

Lectura de archivos

La instrucción `open` abre un archivo, y con la etiqueta `r` se abre para lectura:

In [1]:
```
f = open('texto.txt','r')
for linea in f:
    print(linea)
f.close()
```

```
Con diez cañones por banda,

viento en popa a toda vela,

no corta el mar, sino vuela,

un velero bergantín.
```

Observaciones:

En el ejemplo anterior, asumimos que del programa que contiene la instrucción `open` se encuentra en una carpeta que también contiene el archivo `texto.txt`.

También, es importante cerrar un archivo cuando termina su uso.

Observamos que un archivo se comporta de una manera similar a una lista, y por eso lo hemos recorrido con la instrucción `for`.

Finalmente, vemos que el resultado emite dos fines de línea por cada línea: uno porque cada línea de ese archivo ya contiene su propio fin de línea. Y otro añadido por cada ejecución de la instrucción *print*. Esto se puede evitar de dos maneras:

In [2]:
```
f = open('texto.txt','r')
for linea in f:
    print(linea.strip())
f.close()

print("-----------------------------")

f = open('texto.txt','r')
for linea in f:
    print(linea, end="")
f.close()

print("-----------------------------")

f = open('texto.txt','r')
for linea in f:
    print(linea.strip(), end="")
f.close()
```

```
Con diez cañones por banda,
viento en popa a toda vela,
no corta el mar, sino vuela,
un velero bergantín.
-----------------------------
Con diez cañones por banda,
viento en popa a toda vela,
no corta el mar, sino vuela,
un velero bergantín.
-----------------------------
Con diez cañones por banda,viento en popa a toda vela,no corta el mar, sino vuela,un velero bergantín.
```

La expresión `readline()` permite leer un archivo línea a línea:

In [3]:
```
f = open('texto.txt','r')
a = f.readline()
print(a)
a = f.readline()
print(a)

print("-------------------------")

linea = f.readline()
i = 1
while linea != '':
    print(i, " - ", linea)
    linea = f.readline()
    i += 1
f.close()
```

```
Con diez cañones por banda,

viento en popa a toda vela,

-------------------------
1  -  no corta el mar, sino vuela,

2  -  un velero bergantín.
```

¿Por qué, en el segundo ejemplo, se empieza por la tercera línea, y no por la primera? La respuesta está en la instrucción `close`.

Decíamos antes que un archivo se comporta un poco como una lista. De hecho, podemos leerlo todo a un tiempo en una lista utilizando `readlines()`:

In [4]:
```
f = open('texto.txt', 'r')
lista = f.readlines()
print(type(lista), len(lista))
print(lista)
f.close()
```

```
<class 'list'> 4
['Con diez cañones por banda,\n', 'viento en popa a toda vela,\n', 'no corta el mar, sino vuela,\n', 'un velero bergantín.\n']
```

In [5]:
```
print(lista[2].strip())
```

```
no corta el mar, sino vuela,
```

Datos numéricos y más, en un archivo

Un archivo puede contener datos de cualquier tipo, por ejemplo números, pero inicialmente cada línea es considerada como una **cadena de caracteres**.

Como antes, asumimos que existe un fichero `numeros.txt` en el mismo directorio en el que se ejecuta el programa.

In [6]:
```
g = open('numeros.txt','r')
lista = g.readlines()
print(lista)
g.close()
```

```
['1\n', '2\n', '3\n', '4\n', '5\n']
```

Si queremos recuperar los números como tales, tenemos que convertirlos en enteros:

In [7]:
```
lista_num = []
for cad in lista:
    lista_num.append(int(cad.strip()))
lista_num
```

Out[7]:
```
[1, 2, 3, 4, 5]
```

Es habitual que los ficheros con números haya más de un número por fila, o datos de distinto tipo:

In [8]:
```
f = open("agenda.txt", "r")
linea = f.readline()
print(linea)
elementos = linea.split()
print(elementos)
agenda = {elementos[0]: {
    "peso": int(elementos[1]),
    "altura": float(elementos[2]),
    "dirección": elementos[3],
    }
}
print(agenda)
f.close()
```

```
Fernando 75 1.90 Pozuelo

['Fernando', '75', '1.90', 'Pozuelo']
{'Fernando': {'peso': 75, 'altura': 1.9, 'dirección': 'Pozuelo'}}
```

Lo normal es leer todos los datos de un archivo de una vez:

In [9]:
```
f = open("agenda.txt", "r")
agenda = dict()
for linea in f:
    elementos = linea.split()
    agenda [elementos[0]] = {
        "peso": int(elementos[1]),
        "altura": float(elementos[2]),
        "dirección": elementos[3],
    }
print(agenda)
f.close()
```

```
{'Fernando': {'peso': 75, 'altura': 1.9, 'dirección': 'Pozuelo'}, 'Elena': {'peso': 70, 'altura': 1.7, 'dirección': 'Colón'}}
```

Para datos más complejps, podríamos desear usar un separador distinto del espacio:

```
Fernando # 75 # 1.90 # carretera de Húmera, Pozuelo        # 28223
Elena    # 70 # 1.70 # calle Bárbara de Braganza, Madrid   # 28004
```

Prueba tú con las funciones `split("#")`, o con `split(" # ")` y las funciones `strip`, `rstrip` y `lstrip`, que toman otras cadenas de caracteres como separador y que limpian de espacios y otras cosas el inicio y final de una cadena de caracteres.

Nota: `map` y listas intensionales

Cuando todos los datos de un archivo son del mismo tipo, es adecuado tratarlos genéricamente, mediante el uso de la función `map` o de la notación de listas intensionales:

In [10]:
```
# Uso de la función `map`:

lista_cadenas = ['1\n', '2\n', '3\n', '4\n', '5\n']
lista_num = list(map(int, lista_cadenas))
print(lista_num)
```

```
[1, 2, 3, 4, 5]
```

In [11]:
```
# Listas intensionales:

lista_cadenas = ['1\n', '2\n', '3\n', '4\n', '5\n']
lista_num = [int(cad) for cad in lista_cadenas]
print(lista_num)
```

```
[1, 2, 3, 4, 5]
```

Escritura de archivos

También podemos guardar datos en un archivo:

In [12]:
```
f = open('mi_texto.txt','w')
f.write('Con cien cañones por banda\n')
f.write('Viento en popa a toda vela...\n')
f.close()
```

Observaciones:

Al abrir un archivo para escribir pueden ocurrir dos cosas:

- Si el archivo existía pierde su contenido anterior.
- Si el archivo no existía se crea con el contenido que le demos.

Por supuesto podemos crear un archivo para guardar números... pero como cadenas!

In [13]:
```
l = range(50)
h = open('lista_numeros.txt','w')
for x in l:
    h.write('{0}\n'.format(x))
h.close()
```

Apéndice 1. Funciones y métodos específicos para listas y cadenas

Repasamos seguidamente algunas funciones y métodos para manejar listas y cadenas.

strip: es un método de las cadenas que devuelve una copia sin *blancos* (espacios, tabuladores o saltos de línea) delante o detrás.

In [14]:
```
a = '  hola   ' + '\n'
a
```

Out[14]:
```
'  hola   \n'
```

In [15]:
```
a.strip()
```

Out[15]:
```
'hola'
```

split (partir): es un método de las cadenas que *divide* una cadena de caracteres en trozos y devuelve una lista que contiene los trozos. Si no se indica un separador, se asume que los trozos están separados por espacios en blanco.

In [16]:
```
a = 'maneras de vivir'
l = a.split()
l
```

Out[16]:
```
['maneras', 'de', 'vivir']
```

In [17]:
```
b = 'maneras.de.vivir'
l2 = b.split('.')
l2
```

Out[17]:
```
['maneras', 'de', 'vivir']
```

join (unir): es un método de las cadenas que a partir de una cadena, y dada una lista, une los elementos de la lista para crear una cadena formada por los elementos de la lista conectados por la cadena original.

In [18]:
```
lista = ['uno', 'dos', 'tres', 'cuatro']
''.join(lista)
```

Out[18]:
```
'unodostrescuatro'
```

In [19]:
```
' '.join(lista)
```

Out[19]:
```
'uno dos tres cuatro'
```

In [20]:
```
'-=-'.join(lista)
```

Out[20]:
```
'uno-=-dos-=-tres-=-cuatro'
```

Apéndice 2: instrucción *with*, para evitar algunos errores típicos con el manejo de archivos

La instrucción **with** evita errores de programación.

- Hace el close del fichero de forma automática
- Si se produce cualquier error, se llama a la instrucción close.

In [21]:
```
with open('texto.txt', 'r') as f:
    for linea in f:
        print(linea.strip())
```

```
Con diez cañones por banda,
viento en popa a toda vela,
no corta el mar, sino vuela,
un velero bergantín.
```

In [22]:
```
with open('mi_texto2','w') as f:
    f.write('Con cien cañones por banda\n')
    f.write('Viento en popa a toda vela...\n')
```

Si se produce un error dentro del cuerpo del **with**, el fichero se cierra.

In [23]:
```
lst = [3, 4, 5, 0, 6, 8]
with open('inverses.txt', 'w') as f:
    for number in lst:
        f.write('{0}\n'.format(1/number))
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-23-c25f8957b440> in <module>
      2 with open('inverses.txt', 'w') as f:
      3     for number in lst:
----> 4         f.write('{0}\n'.format(1/number))

ZeroDivisionError: division by zero
```

In [24]:
```
with open('inverses.txt', 'r') as f:
    for linea in f:
        print(linea.strip())
```

```
0.3333333333333333
0.25
0.2
```