Utilidades de entrada y salida

Entrada

Ahora veremos cómo interactuar con el exterior desde y hacia un programa de Python. Para leer información desde la terminal (**importante:** en Python 2 [que muy probablemente ninguno de ustedes usará] deben usar `raw_input`):

```
nombre = input('Cual es tu nombre? ')

> Cual es tu nombre? Cristobal

nombre

> Cristobal
```

Podemos usar algunas opciones comunes para asegurar el resultado esperado:

```
seguir = input('Deseas continuar? [y/N] ')

while seguir and seguir.lower() not in 'yn':

     print(f'{seguir} no es una opción válida')

     seguir = input('Deseas continuar? [y/N] ')

if not seguir or seguir.lower() == 'n':

     print('Chao')

...
```

También podemos acceder al contenido de un archivo de texto de manera muy simple:

```
file = open('lvg_table1.dat')

contenido = file.read()

file.close()

contenido

> ...
```

Noten que una vez leído un *objeto archivo* no puede volver a leerse:

```
file = open('lvg_table1.dat')

contenido1 = file.read()
```

```
type(contenido1), len(contenido1)

> (str, 147045)

contenido2 = file.read()

type(contenido2), len(contenido2)

> (str, 0)

file.close()
```

Debemos recordar *siempre* cerrar el archivo. Es más útil separar el contenido por líneas:

```
file = open('lvg_table1.dat')

contenido = file.readlines()

file.close()

type(contenido), len(contenido)

> (list, 1246)

contenido[0]

> 'AGC102728        000021.4+310119   0.20 0.58 0.17
20.40  20.2         17.7*   18.70  21 10     Ir L  566
8.87 TRGB\n'
```

La función `readlines` lee todo el contenido, entregando una lista de strings, un string para cada línea. También podemos acceder a las líneas una a una:

```
file = open('lvg_table1.dat')

line = file.readline()

line

> 'AGC102728        000021.4+310119   0.20 0.58 0.17
20.40  20.2         17.7*   18.70  21 10     Ir L  566
8.87 TRGB\n'
```

El resultado es un string con el contenido de la primera línea.

Antes de seguir, veamos utilidades para trabajar con esta línea (que son utilidades genéricas de strings que no hemos visto hasta ahora). Los últimos dos caracteres, "`\n`", corresponden a un salto de línea. Podemos quitar este carácter, además de cualquier espacio o tab extra al final del string con:

```
line = line.rstrip()

line

> 'AGC102728        000021.4+310119   0.20 0.58 0.17
20.40  20.2         17.7*   18.70  21 10     Ir L  566
8.87 TRGB'
```

La función `strip()` permite hacer lo mismo a ambos lados del string. Podemos querer acceder a cada entrada en esta línea de manera simple:

```
line.split()

> ['AGC102728', '000021.4+310119', '0.20', '0.58',
'0.17', '20.40', '20.2', '17.7*', '18.70', '21',
'10', 'Ir', 'L', '566', '8.87', 'TRGB']
```

Al llamar readline nuevamente, automáticamente leemos la siguiente línea:

```
file.readline().split()

> ['UGC12894', '000022.5+392944', '1.02', '0.87',
'0.47', '17.60', '16.8', '19.40', '14.0*', '15.70', '34',
'10', 'Ir', 'L', '335', '8.47', 'TF']
```

(La función split hace también el trabajo de strip, *sólo si el argumento del separador no se especifica*.) Es importante cerrar el archivo una vez que terminamos de trabajar con él, como lo hemos estado haciendo hasta ahora:

```
file.close()
```

Este es un buen momento para introducir la declaración `with`, que define un *contexto* de trabajo:

```
with open('lvg_table1.dat') as file:

    file.read()

    > ...
```

Al salir de este contexto (es decir, desindentar) el archivo se cierra solo. Existen otros usos para la declaración `with`, pero esta es la más común.

De hecho, podemos iterar *sobre el archivo* para leer cada línea automáticamente. Por ejemplo, si estamos seguros que nuestro archivo es una tabla regular con columnas separadas por espacios (que *no* es el caso de `lvg_table1.dat`), podríamos hacer

```
with open('lvg_table1.dat') as file:
```

```
        for i, line in enumerate(file):
            line = line.split()
            print(f'línea {i}: {len(line)} columnas')
            if i == 5:
                break
```

```
> línea 0: 16 columnas
> línea 1: 17 columnas
> línea 2: 16 columnas
> línea 3: 17 columnas
> línea 4: 18 columnas
> línea 5: 14 columnas
```

Ejemplo

Usaremos el archivo **`fuerzas.tbl`** disponible en el repositorio del curso. En él, la primera línea corresponde al encabezado de la tabla, y las siguientes son los datos. Las columnas están separadas por "`|`". Podríamos construir un diccionario a partir de este archivo en menos de 5 líneas de código:

```
with open('fuerzas.tbl') as file:
    nombres = file.readline().strip().split('|')
    datos = [line.strip().split('|')
             for line in file.readlines()]
tbl = {nombre: col
       for nombre, col in zip(nombres, datos)}
```

Incluso, para eliminar esos espacios adicionales en los distintos elementos podemos anidar comprensiones de lista (reemplazar esta línea donde corresponde arriba):

```
datos = [[word.strip()
          for word in line.strip().split('|')]
         for line in file]
```

Salida

La contraparte a la función **`open`** (que lee un archivo) es la función **`print`** (que imprime a un archivo). Ya vimos cómo imprimir a la terminal. Veamos esta función más en detalle:

```
help(print)

print(...)

     print(value, ..., sep=' ', end='\n',
file=sys.stdout, flush=False)



     Prints the values to a stream, or to sys.stdout by
default.

     Optional keyword arguments:

     file:  a file-like object (stream); defaults to the
current sys.stdout.

     sep:   string inserted between values, default a
space.

     end:   string appended after the last value, default
a newline.

     flush: whether to forcibly flush the stream.
```

Los puntos suspensivos muestran que la función print puede imprimir cuantos argumentos le entreguemos:

```
print('uno', 'dos', 'tres')

> uno dos tres
```

El argumento que nos permite imprimir a un archivo es `file`. Este especifica a dónde escribir; aquí podemos entregar un objeto archivo escribible (no un *nombre* de archivo):

```
with open('nuevo.txt', 'w') as file:

     print('Probando 1 2 3', file=file)
```

Esto reescribirá el archivo nuevo.txt, si es que ya existía. Para agregar información al final del archivo, podemos abrirlo en modo 'append':

```
with open('nuevo.txt', 'a') as file:

     print('Probando de nuevo', file=file)
```