

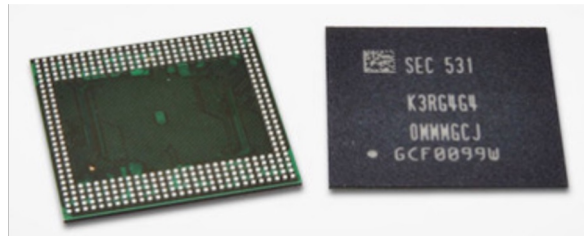
Python (XVI)

Programación (tics 100)
Semestre 02/2020

Datos volátiles y permanentes

Los datos almacenados de forma “volátil” son aquellos que solo “viven” o “existen” mientras un determinado programa este funcionando y mientras el computador tenga energía. Estos datos son generalmente almacenados en la RAM (Random Access Memory). Los datos almacenados en RAM pueden ser operados (leer, escribir, borrar) de forma muy rápida.

Las variables de un programa son datos almacenados de forma volátil en RAM.



Datos volátiles y permanentes

Los computadores necesitan una manera de poder guardar información a largo plazo. Que no sea dependiente de que un determinado programa este funcionando o que el computador tenga energía. Para estos casos tenemos formas de guardar los datos de manera permanente o de manera no volátil. Operar (leer, escribir, borrar) con datos no volátiles es lento.

Estos datos se guardan generalmente en los discos duros.



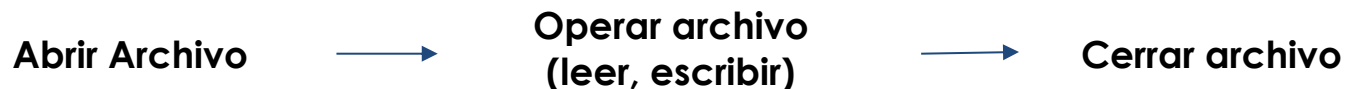
Hoy veremos cómo guardar, leer y borrar datos desde nuestro programa Python de manera no volátil.

Manejo de archivos I/O

Cuando queremos leer, guardar o eliminar datos de forma no volátil, tenemos que hacer referencia a un **archivo** de nuestro computador. Los archivos tienen cientos de formatos, desde imágenes (jpg), pasando por videos (mp4), hasta texto (txt).

Nosotros ocupamos para esta clase archivos de tipo **txt** (texto).

Flujo de operación:

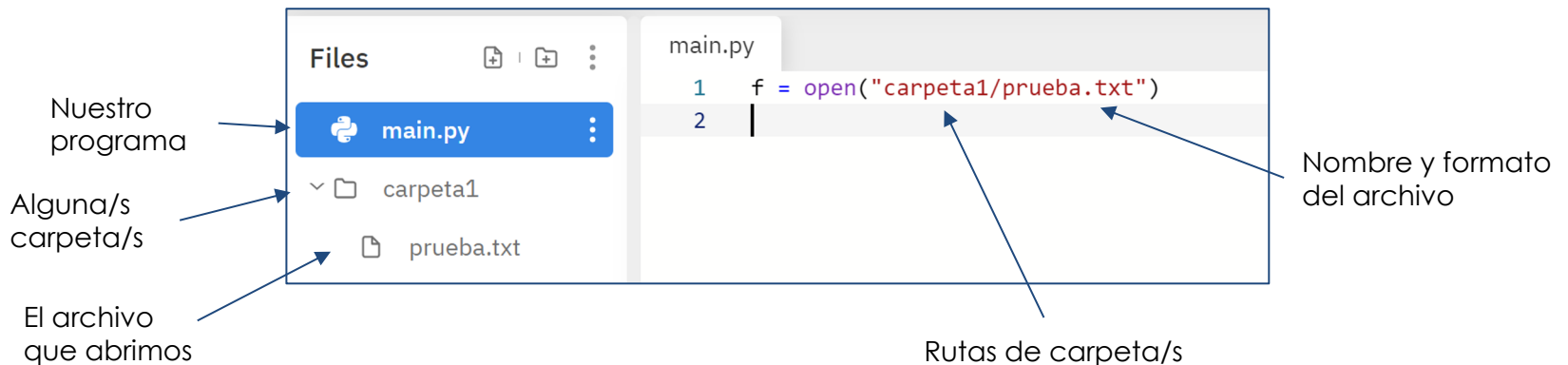
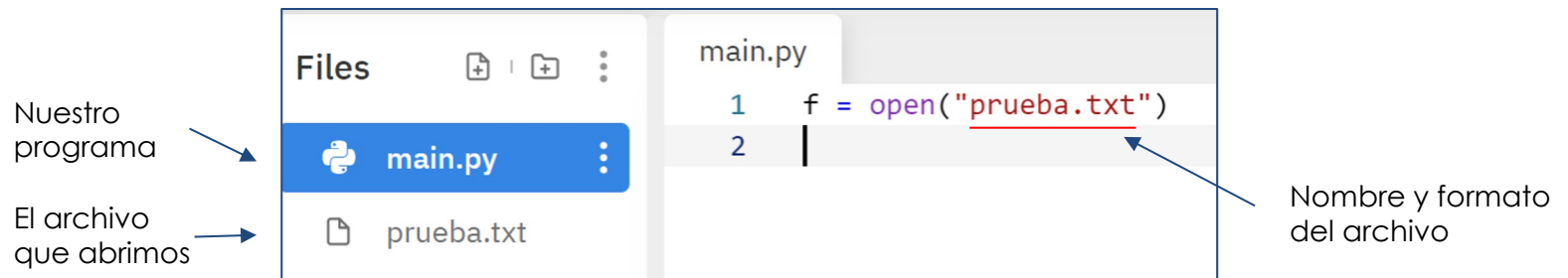


Abrir

open(ruta y nombre del archivo, <modo>, <formato>): Con open podemos abrir un archivo dejar su referencia almacenada en una variable. Por defecto podemos abrir archivos que estén en la misma ruta que nuestro fichero .py

Como opciones adicionales podemos especificar el modo y formato con el cual abrir el archivo.

Por defecto, los archivos se abren en modo "r" (leer)

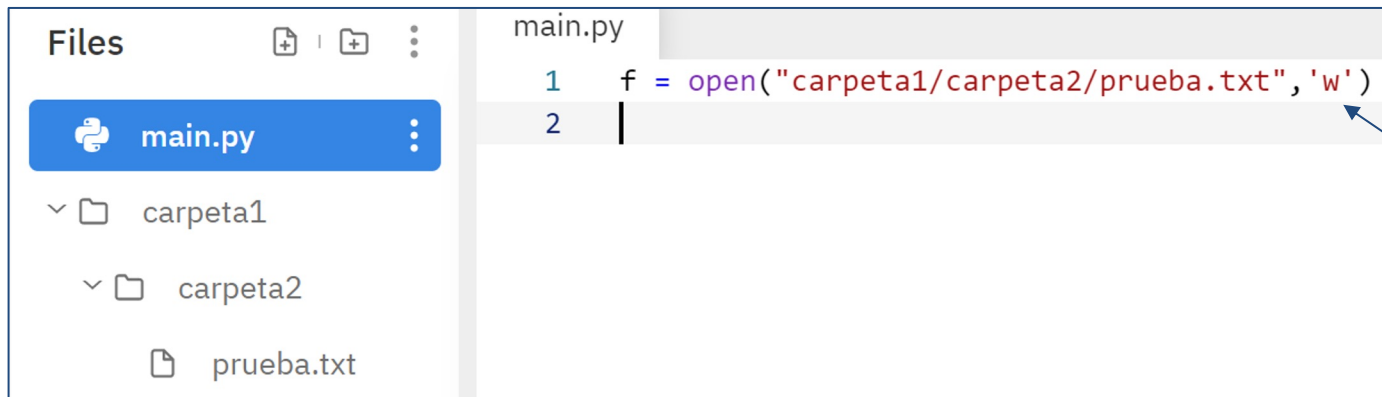


Abrir

Modos basicos:

- r → Abre un archivo para leer (por defecto)
- w → Abre un archivo para escribir. Si no existe, lo crea. Si ya existe, borra el contenido e inicia de 0.
- x → Crea un archivo para luego escribir. Si ya existe, el sistema falla.
- a → Abre un archivo para añadir contenido al mismo. Si no existe, lo crea.

Existe otros modos para trabajar en modo texto o en modo binario.



```

Files
├── main.py
├── carpeta1
├── carpeta2
└── prueba.txt

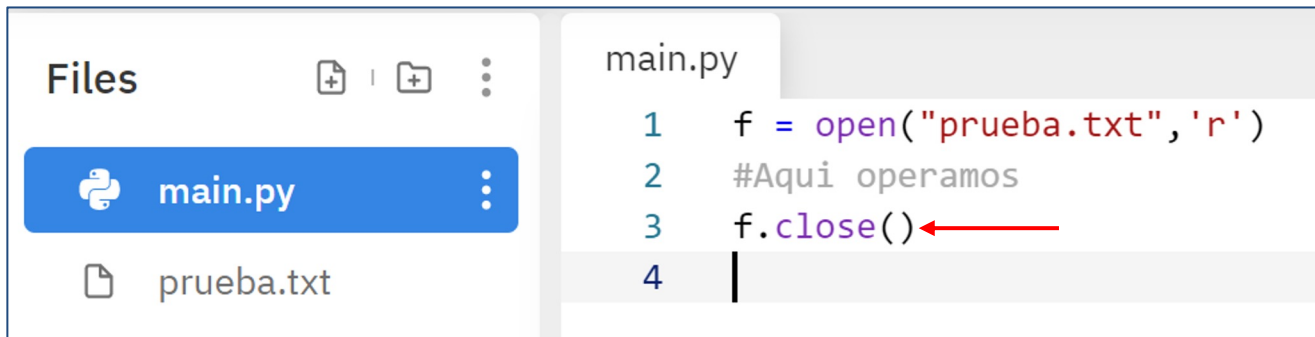
main.py
1  f = open("carpeta1/carpeta2/prueba.txt", 'w')
2  |
  
```

Abriendo
archivo en
modo
escritura "w"

Ojo: Podemos ver que el archivo se encuentra a 2 carpetas de profundidad de nuestro .py

Cerrar

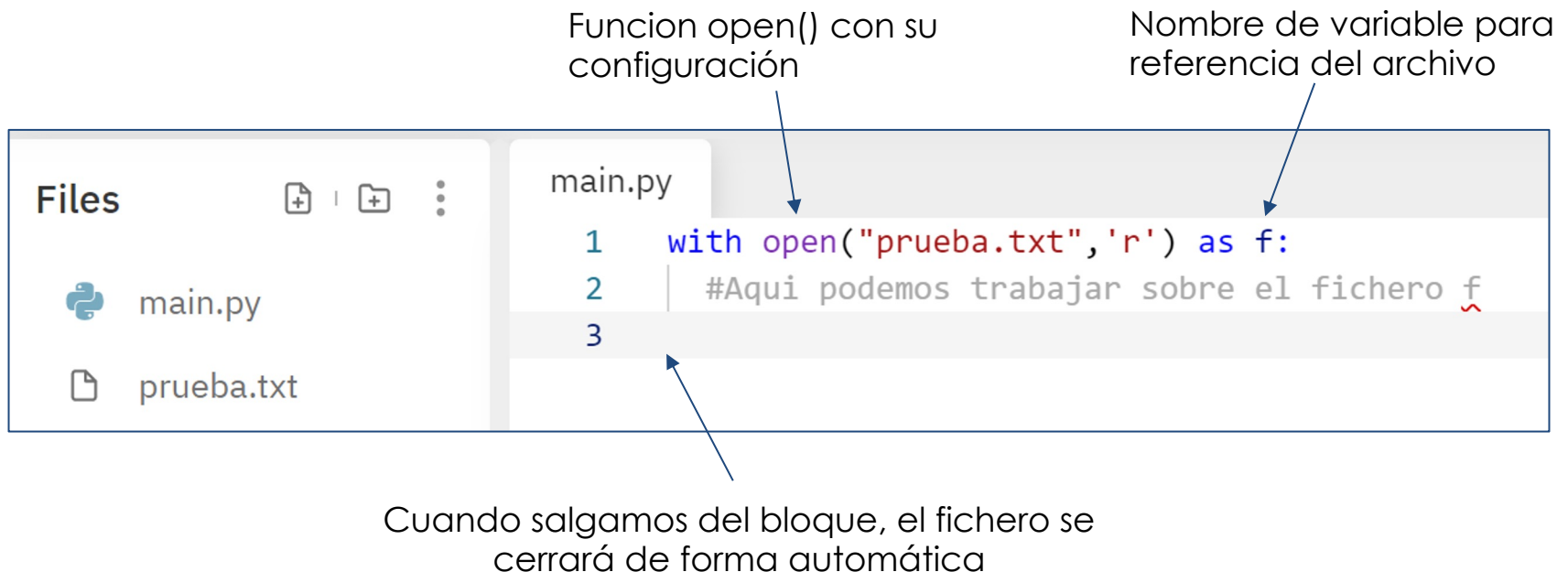
Luego de abrir y operar sobre el archivo. Podemos ocupar **close()** para cerrar el archivo. Si no cerramos el archivo, los datos pueden quedar corruptos y errores pueden ocurrir si otros programas tratan de operar el mismo archivo.



```
main.py
1  f = open("prueba.txt", 'r')
2  #Aqui operamos
3  f.close()
4
```

Cerrar

Existe un modo alternativo de abrir y cerrar un archivo. Es ocupando el bloque **with**. Al interior de bloque podemos operar sobre el archivo. Al terminar el bloque el sistema cierra el archivo de forma automática.



Funcion open() con su configuración

Nombre de variable para referencia del archivo

```

1  with open("prueba.txt", 'r') as f:
2      #Aquí podemos trabajar sobre el fichero f
3

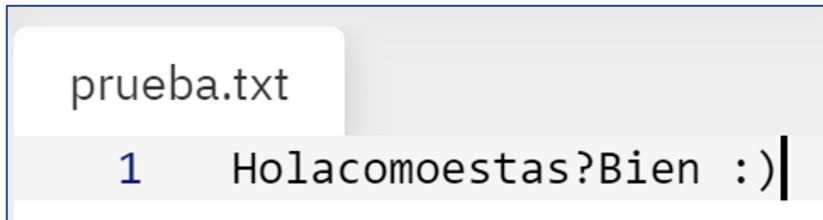
```

Cuando salgamos del bloque, el fichero se cerrará de forma automática

Operar - Escribir

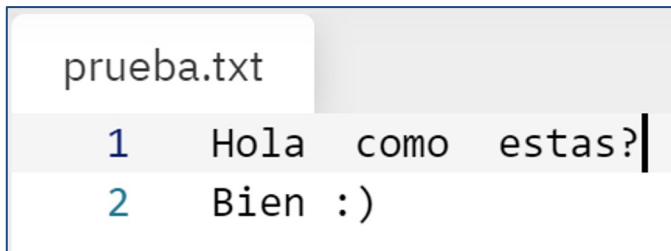
.write(string): Escribimos el la cadena de caracteres en un archivo. Ojo: Al final de la cadena no se genera una nueva línea de manera automática.

```
f = open("prueba.txt", 'w')
f.write("Hola")
f.write("como")
f.write("estas?")
f.write("Bien :)")
f.close()
```



```
prueba.txt
1 Holacomoestas?Bien :)|
```

```
f = open("prueba.txt", 'w')
f.write("Hola ")
f.write(" como ")
f.write(" estas?\n")
f.write("Bien :)")
f.close()
```



```
prueba.txt
1 Hola como estas?|
2 Bien :)|
```

Con `\n` generamos una nueva línea

Operar - Escribir

.writelines(lista de strings): Escribiremos en un archivo todos los string almacenados en una lista. Ojo: Al final de cada cadena no se genera una nueva línea de manera automática.

```
a = "Pepe vive en Arica\n"
b = "Maria vive en Santiago\n"
c = "Juan vive en Punta Arenas\n"
mi_lista = [a,b,c]
print(mi_lista)

f = open("prueba.txt",'w')
f.writelines(mi_lista)
f.close()
```

```
['Pepe vive en Arica\n', 'M
aria vive en Santiago\n', '
Juan vive en Punta Arenas\n
']
```

prueba.txt

```
1  Pepe vive en Arica
2  Maria vive en Santiago
3  Juan vive en Punta Arenas
4
```

Ojo que cada string tiene su término con nueva línea `\n`. Esto provoca que incluso se genere una línea 4 en el fichero txt, producto del `\n` de la variable c

Operar - Leer

.read(<numero de caracteres>): Leemos un número determinado de caracteres. Avanzamos el puntero de lectura esa misma cantidad. Sino se especifica un número de caracteres, se lee todo el fichero.

.readline(): Leemos una línea del fichero (hasta el próximo \n). Avanzamos el puntero de lectura hasta al final de esa línea.

.readlines(): Leemos todas las líneas del fichero y las dejamos almacenadas en una lista.

```
f = open("prueba.txt", 'r')
a = f.read()
print(a)
f.close()
```

```
Pepe vive en Arica
Maria vive en Santiago
Juan vive en Punta Arenas
```



```
f = open("prueba.txt", 'r')
a = f.read(6)
print(a)
b = f.read(3)
print(b)
c = f.read(15)
print(c)
f.close()
```

```
Pepe v
ive
  en Arica
Maria
❖ □
```

Los espacios y \n se consideran caracteres al momento de avanzar el puntero de lectura.

Operar - Leer

.read(<numero de caracteres>): Leemos un número determinado de caracteres. Avanzamos el puntero de lectura esa misma cantidad. Sino se especifica un número de caracteres, se lee todo el fichero.

.readline(): Leemos una línea del fichero (hasta el próximo \n). Avanzamos el puntero de lectura hasta al final de esa línea.

.readlines(): Leemos todas las líneas del fichero y las dejamos almacenadas en una lista.

```
f = open("prueba.txt", 'r')
a = f.readline()
print(a)
b = f.readline()
print(b)
c = f.readline()
print(c)
f.close()
```

```
Pepe vive en Arica
Maria vive en Santiago
Juan vive en Punta Arenas
❖ □
```

Print por defecto nos añade un \n al final. Como los string que leemos del txt también traen su propio \n se provoca que en consola veamos líneas sin nada.

Operar - Leer

.read(<numero de caracteres>): Leemos un número determinado de caracteres. Avanzamos el puntero de lectura esa misma cantidad. Sino se especifica un número de caracteres, se lee todo el fichero.

.readline(): Leemos una línea del fichero (hasta el próximo \n). Avanzamos el puntero de lectura hasta al final de esa línea.

.readlines(): Leemos todas las líneas del fichero y las dejamos almacenadas en una lista.

```
f = open("prueba.txt", 'r')
a = f.readline()
print(a, end="")
b = f.readline()
print(b, end="")
c = f.readline()
print(c, end="")
f.close()
```

```
Pepe vive en Arica
Maria vive en Santiago
Juan vive en Punta Arenas
❖ □
```

Con `end=""`, en el `print`; podemos especificar que al final de cada escritura por consola, no se añada el `\n` por defecto.

Operar - Leer

.read(<numero de caracteres>): Leemos un número determinado de caracteres. Avanzamos el puntero de lectura esa misma cantidad. Sino se especifica un número de caracteres, se lee todo el fichero.

.readline(): Leemos una línea del fichero (hasta el próximo \n). Avanzamos el puntero de lectura hasta al final de esa línea.

.readlines(): Leemos todas las líneas del fichero y las dejamos almacenadas en una lista.

```
f = open("prueba.txt", 'r')
a = f.readlines()
print(a)
print(len(a))
print(a[0], end="")
print(a[1], end="")
print(a[2], end="")
f.close()
```

```
['Pepe vive en Arica\n', 'Maria viv
e en Santiago\n', 'Juan vive en Pun
ta Arenas\n']
3
Pepe vive en Arica
Maria vive en Santiago
Juan vive en Punta Arenas
❖ □
```

} Lista completa
 → N° elementos
 → Acceso por
 → índice a cada
 → elemento

Ejercicio

Crea un programa que le pide a un usuario un número **n** de nombres de ciudades. El número **n** se debe generar de forma aleatoria y puede ir entre 2 y 10.

Guarda cada uno de esos nombres en un archivo llamado “ciudades” de tipo texto (txt). Cada ciudad debe ir en una línea distinta.

```
Te vamos a pedir 4 nombres de ciudades.
Dime una ciudad: Temuco
Dime una ciudad: Arica
Dime una ciudad: Viña del Mar
Dime una ciudad: Santiago

```

Files



main.py



ciudades.txt

ciudades.txt

```
1 Temuco
2 Arica
3 Viña del Mar
4 Santiago
5
```

Ejercicio alternativa 1

Usando una lista y writelines

```
import random

n = random.randint(2,11)

lista_ciudades = []
print("Te vamos a pedir", n, "nombres de ciudades.")
while len(lista_ciudades) < n:
    ciudad = input("Dime una ciudad: ")
    ciudad = ciudad + '\n'
    lista_ciudades.append(ciudad)

f = open("ciudades.txt", 'w')
f.writelines(lista_ciudades)
f.close()
```


Ejercicio alternativa 2

Usando directamente un write en el ciclo

```
import random

n = random.randint(2,11)

print("Te vamos a pedir", n, "nombres de ciudades.")
f = open("ciudades.txt",'w')
contador = 0
while contador < n:
    ciudad = input("Dime una ciudad: ")
    f.write(ciudad)
    f.write("\n")
    contador += 1
f.close()
```

Ejercicio alternativa 3

Usando una lista. Luego un for con write. Abriendo el fichero con with

```
import random

n = random.randint(2,11)

lista_ciudades = []
print("Te vamos a pedir", n, "nombres de ciudades.")
while len(lista_ciudades) < n:
    ciudad = input("Dime una ciudad: ")
    ciudad = ciudad + '\n'
    lista_ciudades.append(ciudad)

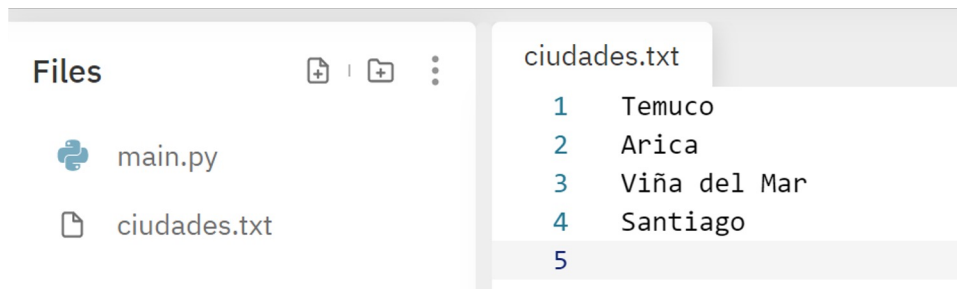
with open("ciudades.txt","w") as f:
    for e in lista_ciudades:
        f.write(e)
```

Ejercicio

Crea un programa que lea el archivo “ciudades.txt” y muestre por pantalla:

1. Imprima por pantalla cada ciudad.
2. Imprime la cantidad de caracteres que componen el nombre de cada ciudad.
3. La cantidad de ciudades que tiene el fichero.

Asume que cada línea tiene una ciudad distinta. Ojo con la última línea.



```
Temuco
Largo: 7
Arica
Largo: 6
Viña del Mar
Largo: 13
Santiago
Largo: 9
Hay 4 ciudades

```

Ejercicio alternativa 1 y 2

Usando **readline()** y un **ciclo**. Ojo: En cada iteración tenemos que verificar si la línea posee contenido. Cuando lleguemos a una línea sin contenido, o sea el fin del archivo, dejamos de leer.

```
f = open("ciudades.txt","r")
cantidad_lineas = 0
while True:
    line = f.readline()
    if not line:
        break
    print(line,end="")
    print("Largo:", len(line))
    cantidad_lineas += 1
print("Hay", cantidad_lineas, "ciudades")
f.close()
```

Usando **readlines()**. De esta forma el sistema automáticamente detecta y almacena en una lista las líneas con algún contenido.

```
f = open("ciudades.txt","r")
list_of_lines = f.readlines()

for e in list_of_lines:
    print(e, end="")
    print("Largo:", len(e))

print("Hay", len(list_of_lines), "ciudades")
f.close()
```

Podemos ver que al contar los caracteres de cada línea, parece que exista uno "fantasma". Ese caracter es el `\n`.

Python (XVI)

Programación (tics 100)
Semestre 02/2020