

Python Courses

**David Martín**

[Course from zero to Heroe - Udemy - 4](#_Toc38301018)

[Instalación de Python y SetUp 5](#_Toc38301019)

[Data types 7](#_Toc38301020)

[Uso de variables 7](#_Toc38301021)

[Método Print () 7](#_Toc38301022)

[Strings 8](#_Toc38301023)

[Propiedades de los strings 8](#_Toc38301024)

[Métodos de los strings 9](#_Toc38301025)

[List 11](#_Toc38301026)

[Métodos 11](#_Toc38301027)

[Podemos concatenar list sumándolas 11](#_Toc38301028)

[Dictionaries 13](#_Toc38301029)

[Métodos 13](#_Toc38301030)

[Tuples 14](#_Toc38301031)

[Métodos 14](#_Toc38301032)

[Sets 15](#_Toc38301033)

[Métodos 15](#_Toc38301034)

[In/Out with basic Files 16](#_Toc38301035)

[Crear un fichero - %%writefile 16](#_Toc38301036)

[Abrir el fichero - open() 16](#_Toc38301037)

[A (append) mode 17](#_Toc38301038)

[W (write) mode 17](#_Toc38301039)

[Leer el fichero - .read() 18](#_Toc38301040)

[.seek(0) 18](#_Toc38301041)

[.readlines() 19](#_Toc38301042)

[.close() 19](#_Toc38301043)

[With...as 19](#_Toc38301044)

[File location 20](#_Toc38301045)

[Operadores de comparación 21](#_Toc38301046)

[Condicionales e Iteraciones 22](#_Toc38301047)

[If -elif-else statement 22](#_Toc38301048)

[For loops 22](#_Toc38301049)

[While loops 24](#_Toc38301050)

# Course from zero to Heroe - Udemy -

## Instalación de Python y SetUp

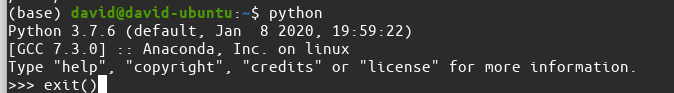
Bajaremos un paqueta q se llama anaconda compuesto por python y una serie de librerías útiles junto con editores de texto como Jupiter. Lo bajamos desde <https://www.anaconda.com/distribution/>

1. Instamos en nuestro direcotrio

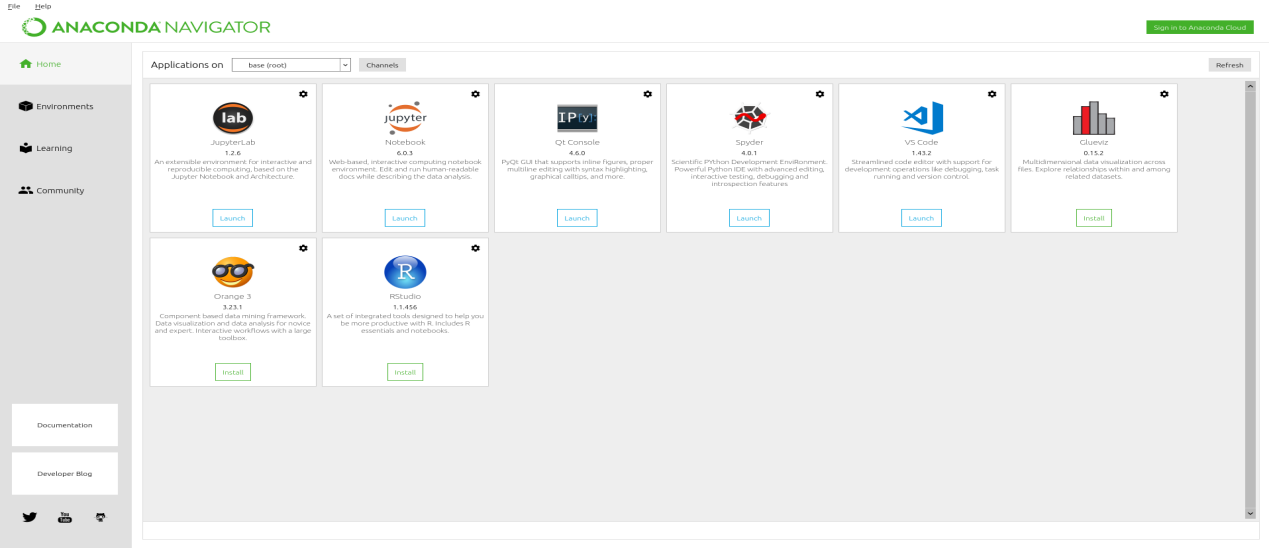
Ejecutamos el archivo de instalación



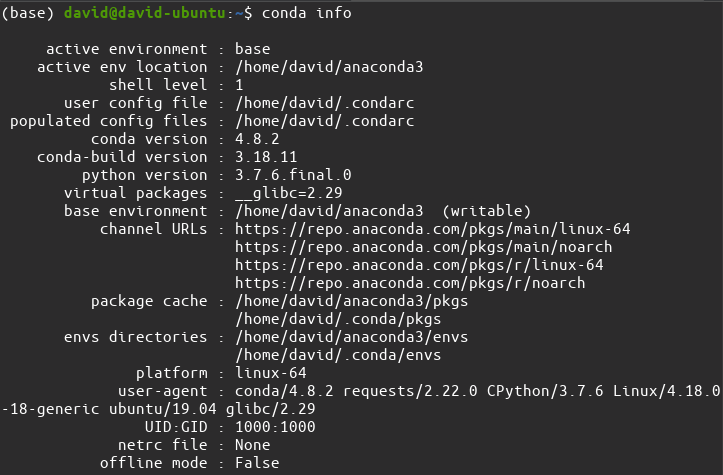
1. Añadimos python al PATH

1. Comprobamos que se ha instalado adecuadamente
2. Lanzamos la interfaz de anaconda





1. Para ver info de anaconda



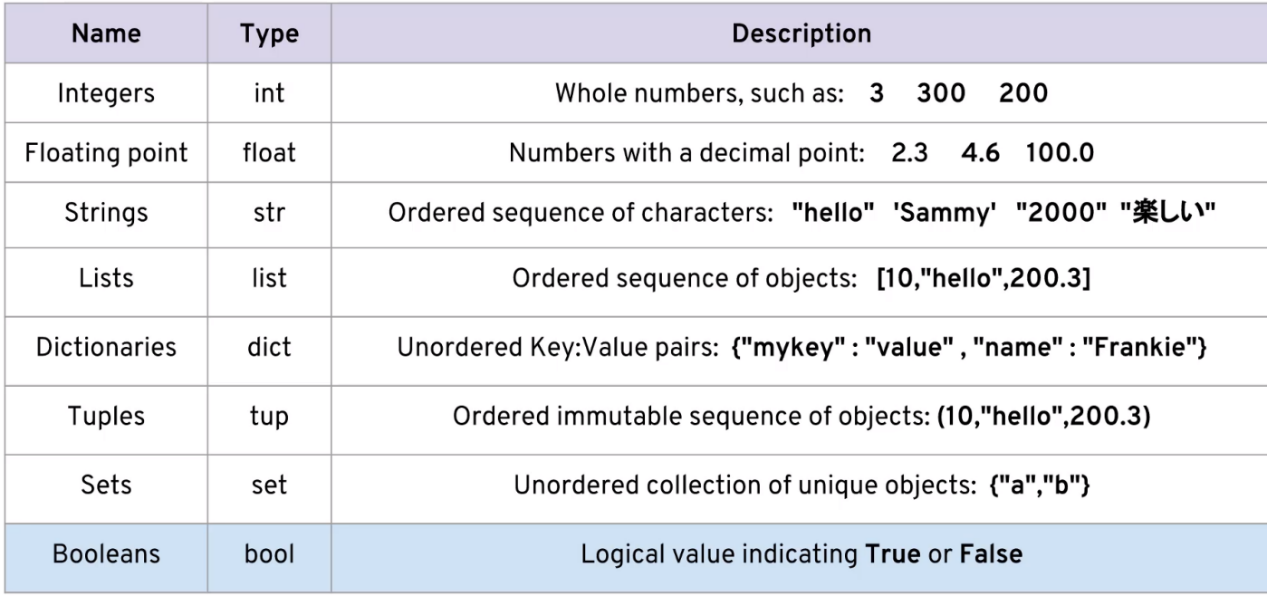
1. Vemos como en el prompt pone base eso es xq es el interprete de anaconda para desactivarlo



1. Para activarlo conda activate



## Data types



### Uso de variables

Python usa tipado dinámico como JavaScript esto significa que no es necesario especificar el tipo de dato que contendrá dicha variable.

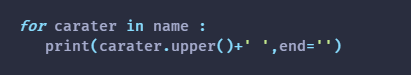
Por ejemplo Java tiene un tipado estático porque requiere que especifiquemos durante la declaración el tipo de dato que contendrá la variable.

*Usado : str type()*



### Método Print ()

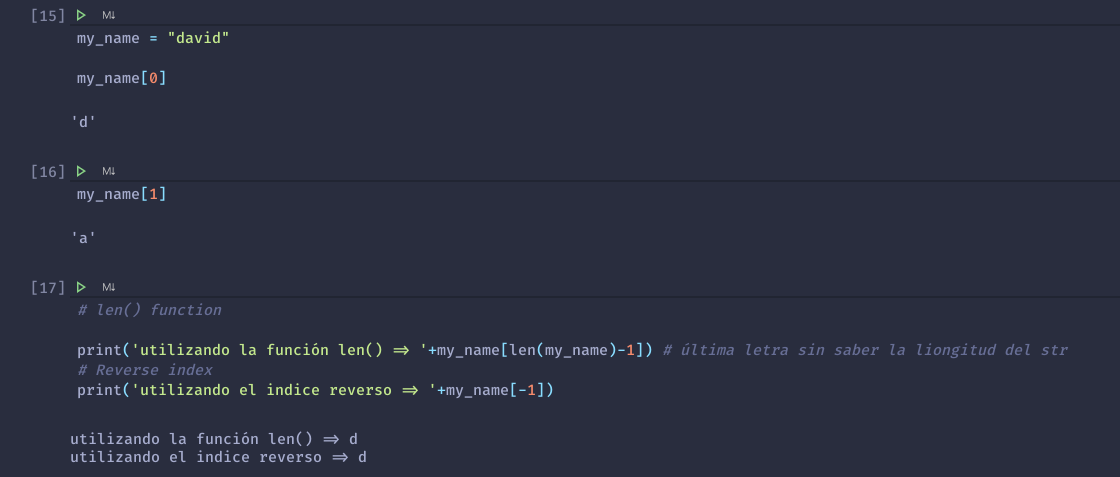
Para imprimir contenido usamos el método print(), éste acepta un atributo para indicar que no haga salto de página end=’’.



### Strings

Funcionan como arrays así que usando el índice podemos extraer letras del string.

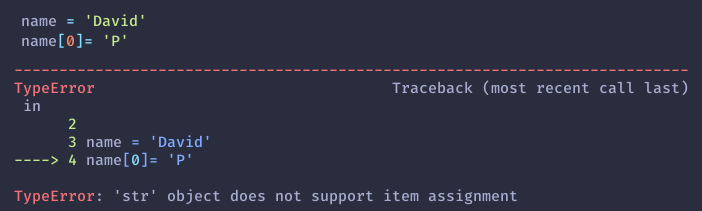
Está habilitado el reverse index(indice reverso) así sin conocer la longitud del string podemos extraer el último carácter.



Tamaño de un str usamos la función len()

#### Propiedades de los strings

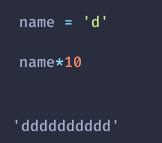
1. Son inmutables, no podemos modificar un string



1. Concatenables



1. Multiplicables

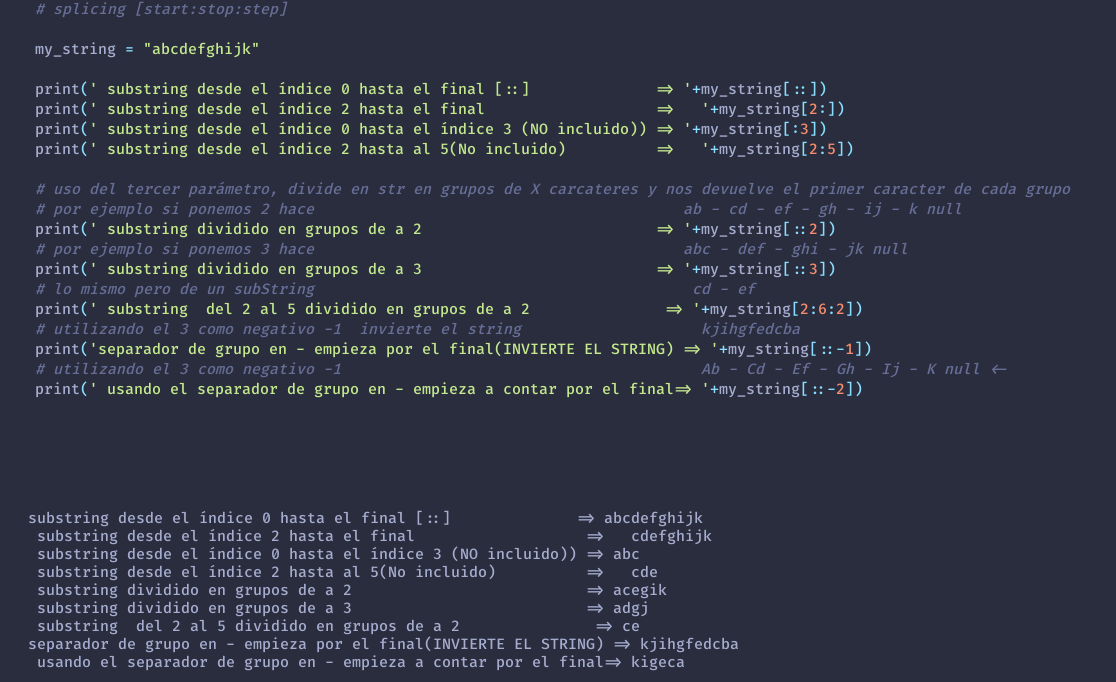


#### Métodos de los strings

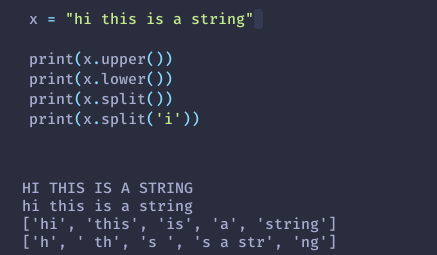
##### Slicing [start:stop:step] obtener un subString

- stop no incluimos ese índice

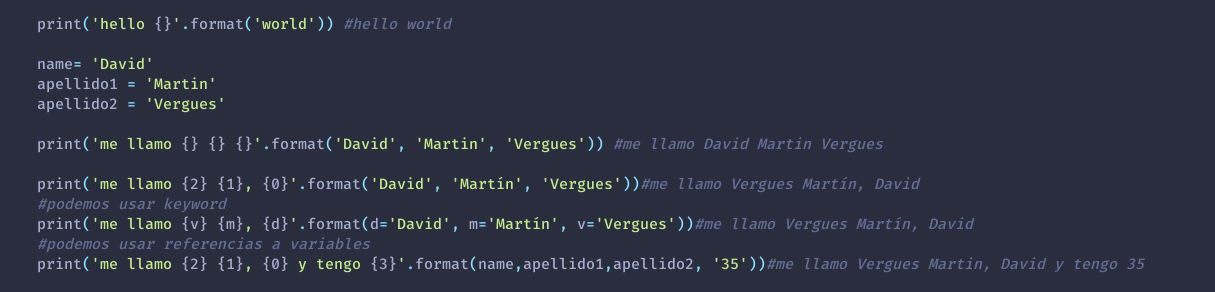
- step tamaño del subString a generar



##### .upper() .lower() .split()



##### .format()

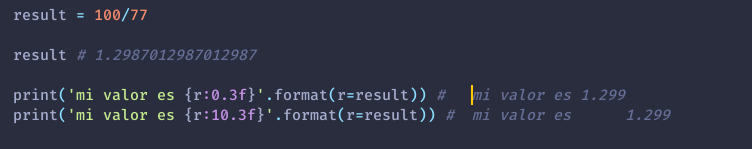


###### 2.1 Float formatting {value:width.precision f}

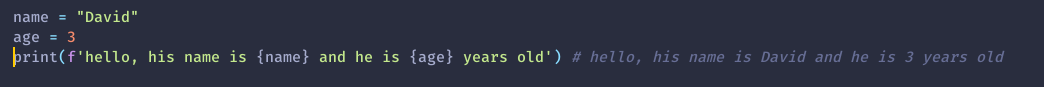
Value => ponemos el valor del número

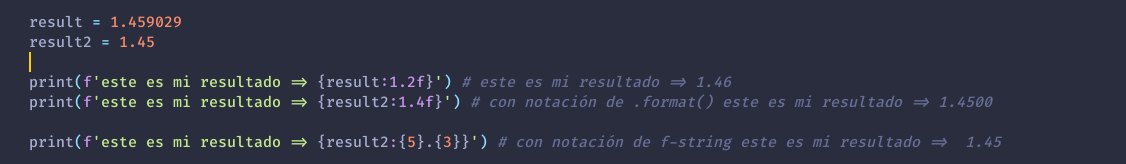
Width => tamaño que ocupará el número hecho string (nº de caracteres)

Precision => número de decimales que tendrá



##### f-string = formated string literal





En este caso hace referencia al número total de dígitos que tendrá el número, no como en .format() que hace referencia al número de dígitos decimales

### List

Secuencia ordenada de elementos que pueden ser de diferentes tipos(numbers, string,obj...). Podemos tener distintos tipos de datos almacenados en un list.

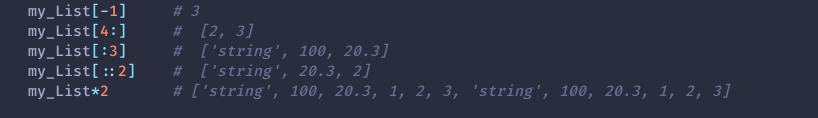
#### Métodos

##### Len()

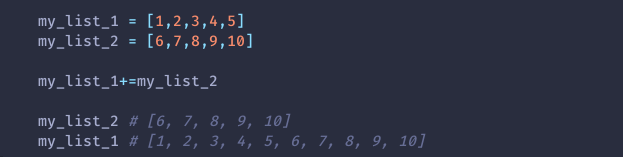
Conocer el número de elementos de la list.

##### Slicing [start:stop:step]

Funciona igual que en los strings



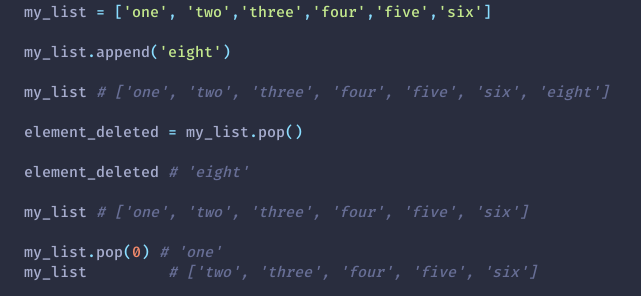
#### Podemos concatenar list sumándolas



##### Append() pop()

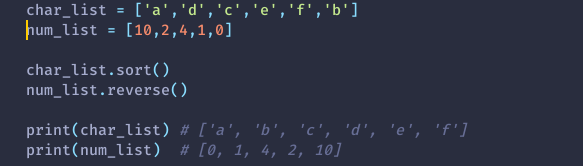
Nos permite añadir(append) o eliminar (pop) un elemento al final de la list.

El método pop() no solo elimina el último elemento sino que también lo devuelve. Una función más es que puedes pasar el índice del elementos a eliminar, por defecto es el -1 (último elemento).

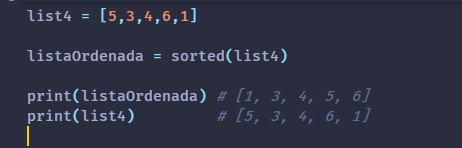


##### Sort() / sorted() / reverse()

Ninguno de los dos métodos devuelve nada, actúan sobre la list y la modifican



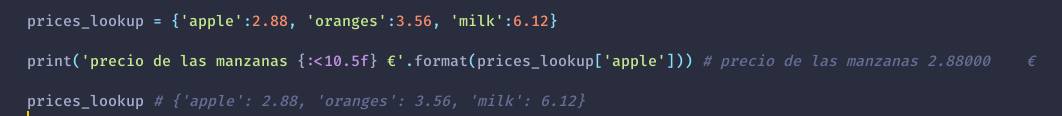
El método sorted() devuelve la lista ordenada pero no modifica la original.



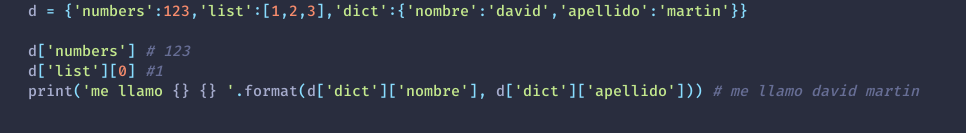
### Dictionaries

Son mapas desordenados (no pueden ser ordenador) para almacenar objetos usando los pares c**lave-valor.**

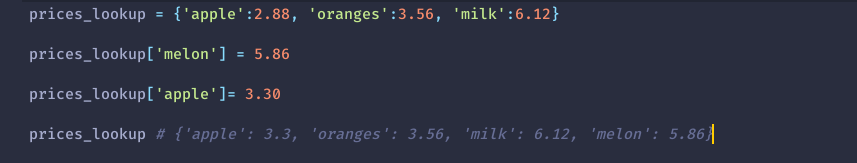
Normalmente los usamos cuando queremos tener dos valores que están relacionados por ejemplo precios de productos, así no necesitamos saber el índice del producto para saber el precio.

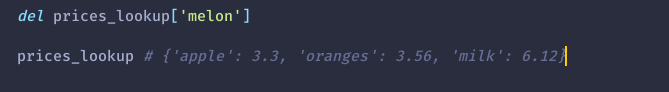


Dentro de los diccionarios podemos almacenar listas y otros diccionarios.



Añadir / sobreescribir / borrar (del) elementos de un diccionario

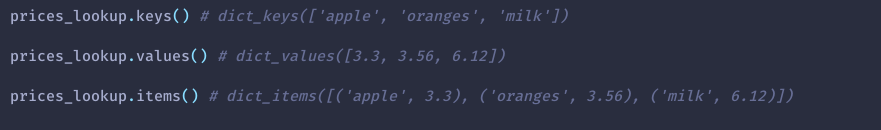




#### Métodos

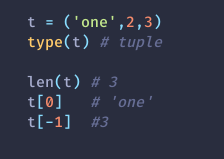
##### keys() / values() / items()

Obtener todas las claves/ valores del diccionario y obtener una array de los pares clave-valor en forma de duplas.



### Tuples

Son muy similares a las listas pero tiene la diferencia que son inmutables.

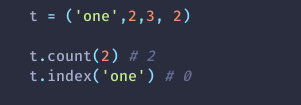


#### Métodos

Sólo hay dos métodos asociados a tuplas.

##### Count() / index()

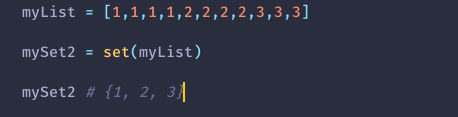
Count devulve cuantas veces se encuentra un elemento en la tupla y index cual es la posición de un elemento dado si aparece más de una vez nos devuelve el índice del primero que encuentra.



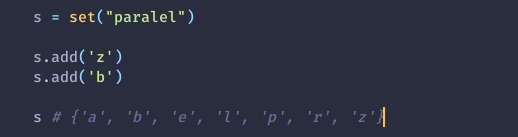
### Sets

Son colecciones sin un orden y de elementos no repetidos.

Podemos crear un set a partir de una list, e esta manera nos aseguramos que los elementos repetidos de la list no se guardan en el set



Si hacemos un set de un string éste guardará cada carácter por separado sin repeticiones.



#### Métodos

##### Add()

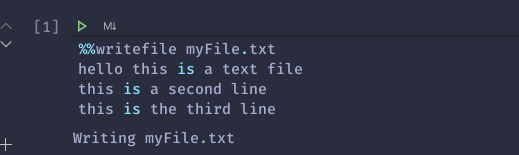
Añadir nuevo elemento



## In/Out with basic Files

Como generar entrada y salida de datos usando un fichero .txt

### Crear un fichero - %%writefile



Después de la sentencia podemos escribir directamente que se nos guardará en el archivo.

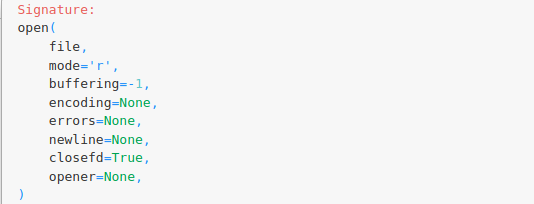
Éste se genera en el path donde se encuentra nuestro script de python. Para saber cual es nuestro path usamos el comando pwd.



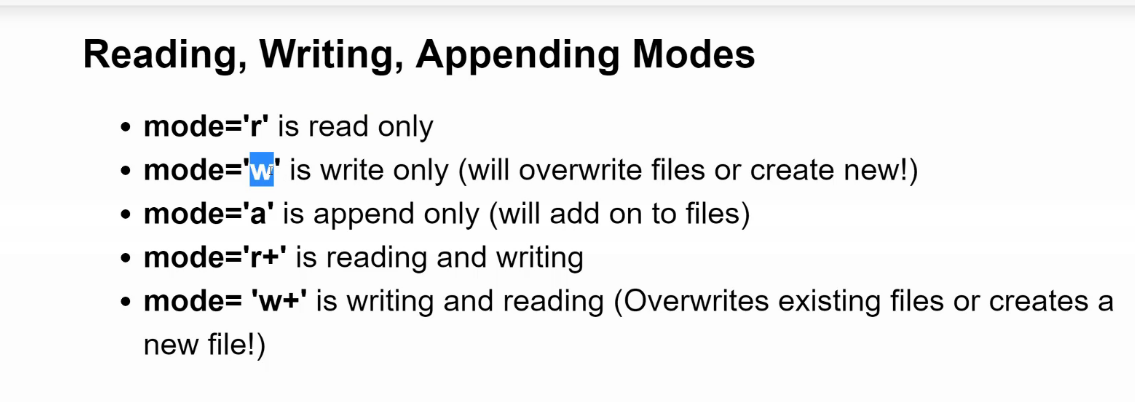
### Abrir el fichero - open()

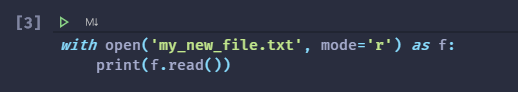


Cuando abrimos un archivo la función acepta estos parámetros:



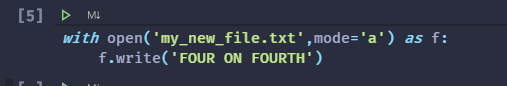
El modo puede ser:





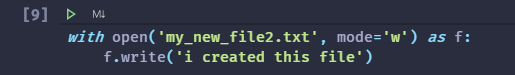
#### A (append) mode

Añade texto al final del documento



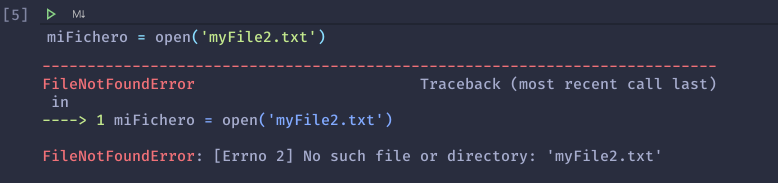
#### W (write) mode

Abrirá o creará en su defecto un archivo con ese nombre, si ya existe lo sobreescribe



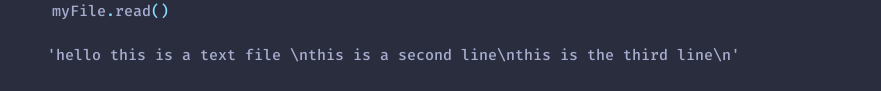
Almacenamos el contenido en una variable. Aquí podemos cometer dos errores:

1. Que escribamos mal el nombre del fichero obteniendo un Errno 2

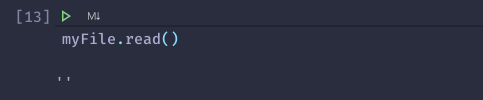


1. Que lo busquemos en un path equivocado

### Leer el fichero - .read()

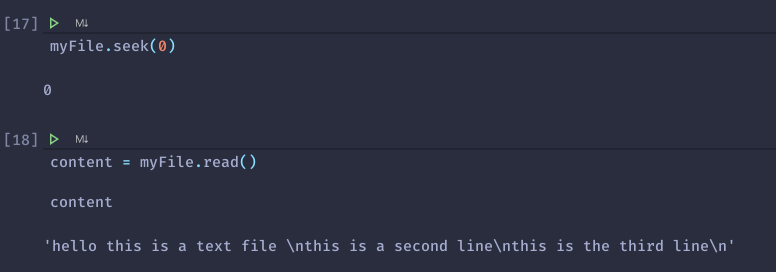


Nos devuelve en un único string todo el contenido del fichero. Este método funciona con un cursor de tal modo que cuando lo utilizamos por primera vez el cursor va desde el inicio al final del texto, así si volvemos a utilizar el método, como el cursor está al final, no nos devolverá nada.



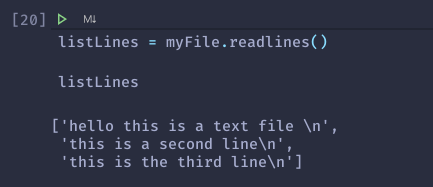
### .seek(0)

Si queremos resetear este cursor utilizamos el método:



### .readlines()

Permite guardar en una lista cada línea del texto. Tenemos que tener en cuenta que al final de cada línea hay un salto de línea \n



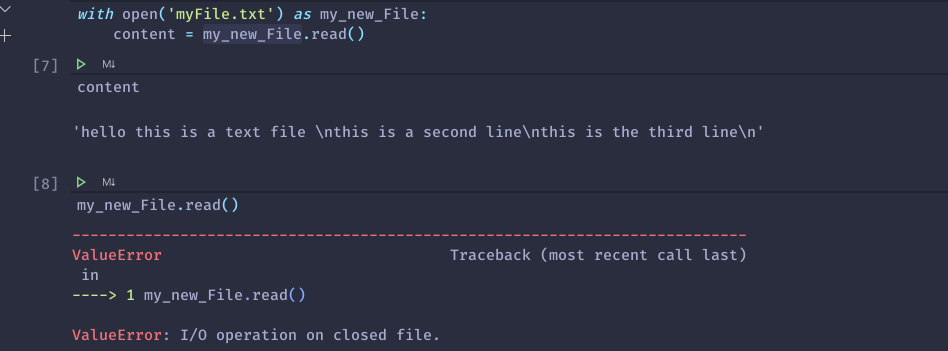
### .close()

Una vez terminamos el trabajo con el fichero debemos cerrarlo



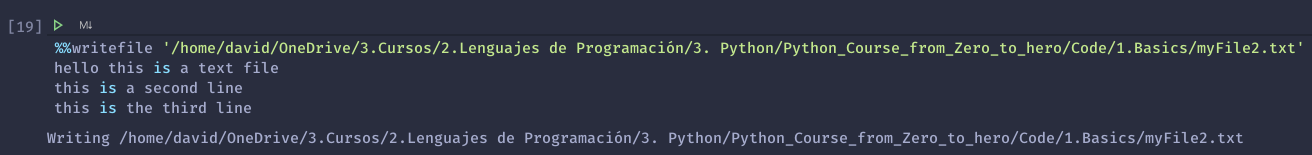
### With...as

Si no nos queremos preocupar por cerrar archivos podemos utilizar esta sentencia que abrirá el archivo y después de hacer las operaciones pertinentes lo vuelve a cerrar automáticamente.

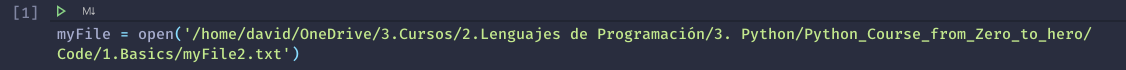


### File location

Crear el fichero en una localización concreta



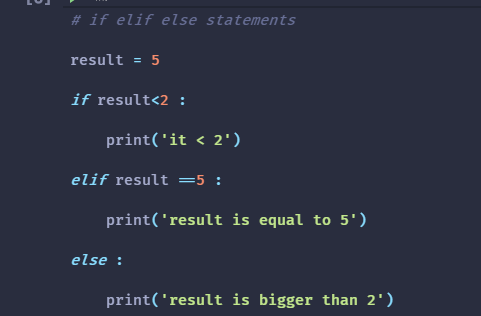
Para abrir un archivo es lo mismo, hay que tener en cuenta si trabajamos en windows entonces usamos los ‘\\’ como separadores o si utilizamos linux o MacOS que usaremos ‘/’



## Operadores de comparación

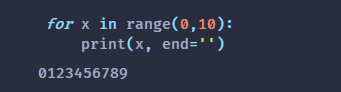
## Condicionales e Iteraciones

### If -elif-else statement

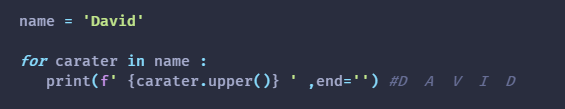


### For loops

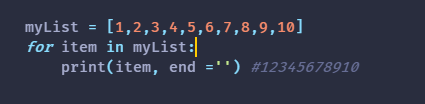
El más sencillo, establecemos un rango.



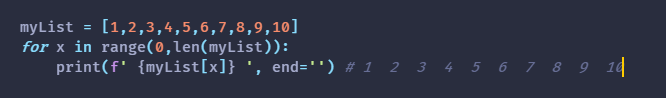
Podemos iterar un string como array de caracteres.



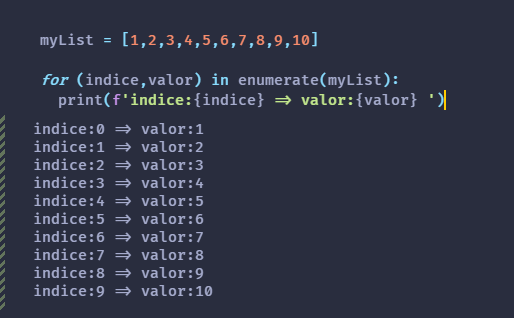
Una list



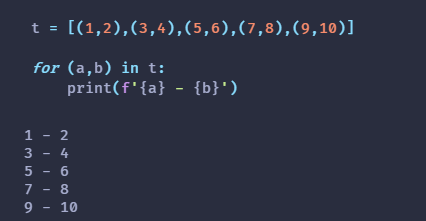
Para conocer el índice de cada elemento lo podemos hacer así:



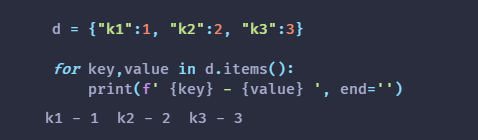
Un truco para obtener el índice de cada elemento de cualquier iterable es transformarlo previamente a un enumerado – enumerate()



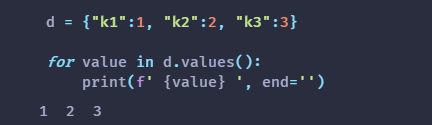
Podemos iterar tuples



Un dictionary



Podemos obtener solo los valores con .values()



### While loops