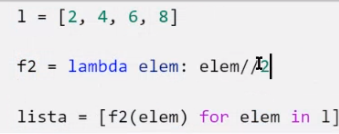
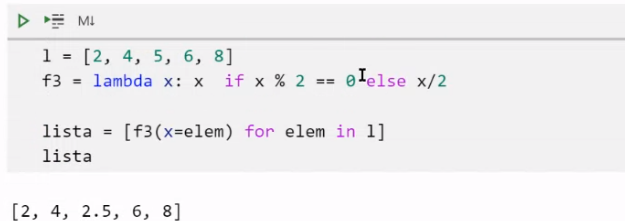
**Lambda**

Las lambda son links a funciones. Tal como se escribe abajo, elaboramos un lambda.



También se pueden hacer con if, en este caso, es obligatorio que tenga un else.



**Seguimiento imports**

Cuando importo un archivo, se ejecuta todo el archivo.

Para acceder a la función de un archivo que está metido en una carpeta de nuestro nivel de ramificación, tenemos que entrar primero a la carpeta, en este caso la llamamos f “f.” y luego al archivo “archivo\_dentro\_f”

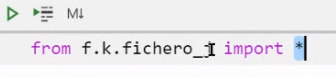


Se puede poner toda la ruta para utilizar la función, pero normalmente utilizamos el “as” para dar un alias que abrevie el nombre de la ruta.

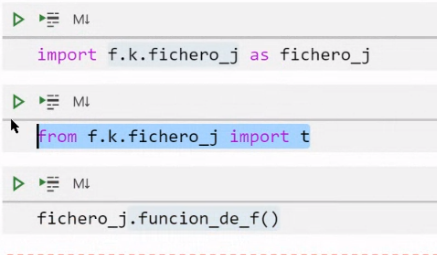


Si creo una función, y luego importo una con el mismo nombre, siempre me quedo con la última que toma el archivo, tengo que considerar, que si importo la ruta al fichero, no estoy sobreponiendo la función, sino que, podré utilizar 2 funciones distintas con el mismo nombre, ya que para el caso importado, tendré que escribir ruta.función. O sea, en este caso, la importada. En resumen, no es aconsejable importar todo porque podemos coger funciones con el nombre de algunas que nosotros ya hemos creado, otro motivo es que cargamos mucho nuestro archivo de funciones.

Cuando importo con un from, pillo funciones específicas. Si importo con estrellita, importo todas las funciones de archivo, y en caso de hacer este from y estrellita, no necesito enrutamiento para llamar a la función (es una mala praxis porque sobrecargo el archivo).

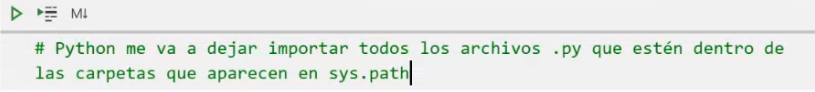


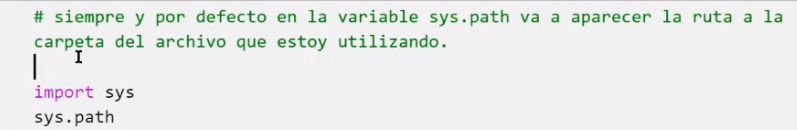
A continuación, se muestran los 2 casos que utilizaremos. Enrutar a otro fichero y aplicar alias, o cargar función específica con from.

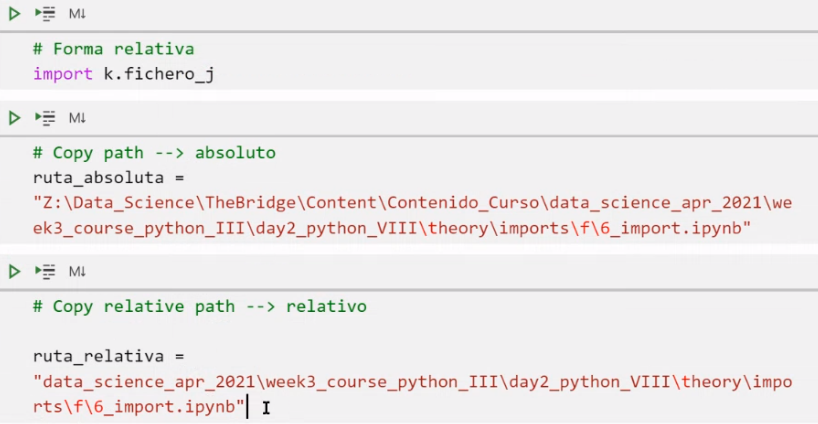


A continuación utilizamos los 2 a la vez, pillamos específico y le damos alias.







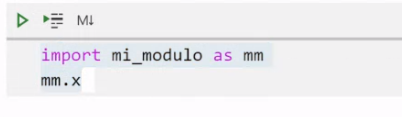


Cuando queramos acceder a una carpeta que esta por encima en ramificación de nuestro archivo, utilizamos ruta absoluta. Cuando utilizamos ruta absoluta, tenemos que cambiar manualmente “\” por “\\”

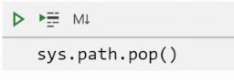


Después hacemos un sys.path.append(ruta\_absoluta) para añadir la ruta

Una vez enrutada la carpeta matriz a la que yo trabajo, Python leerá todo lo de esa carpeta. Por lo que, podemos hacer un import sencillo, y Python lo encontrará.



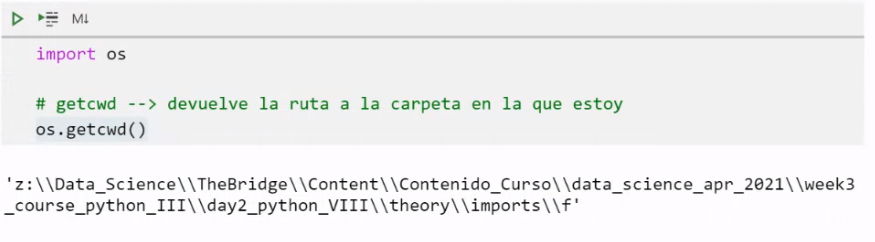
Si queremos sacar la ruta hacemos:



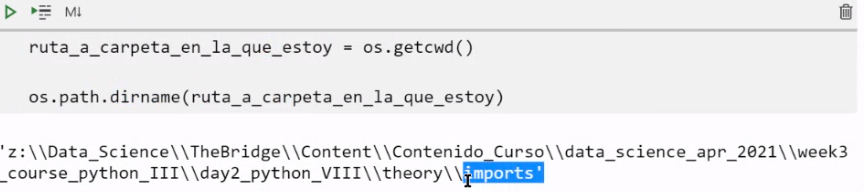
No obstante, la función la seguimos teniendo guardada, almenos hasta que paremos el kernel y reiniciemos las variables y funciones.

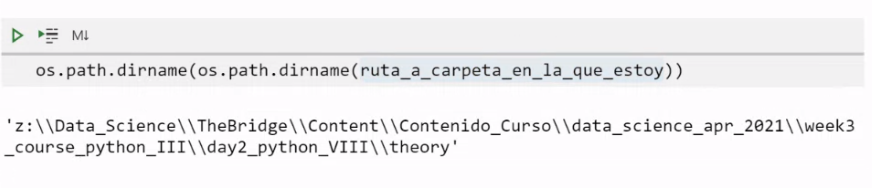
.getcwd



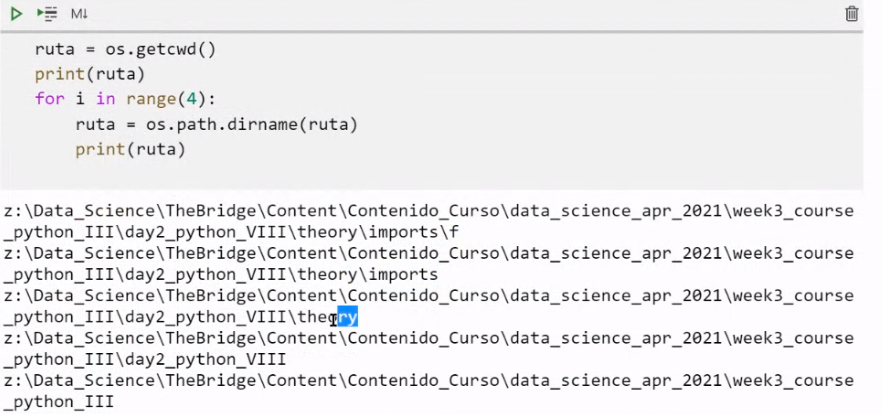


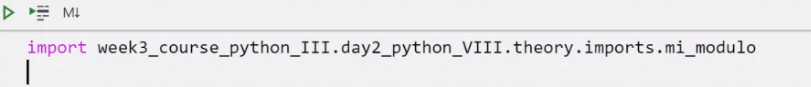
Cada vez que utilizo la función os.path.dirname(), me tira una carpeta atrás en la ramificación.



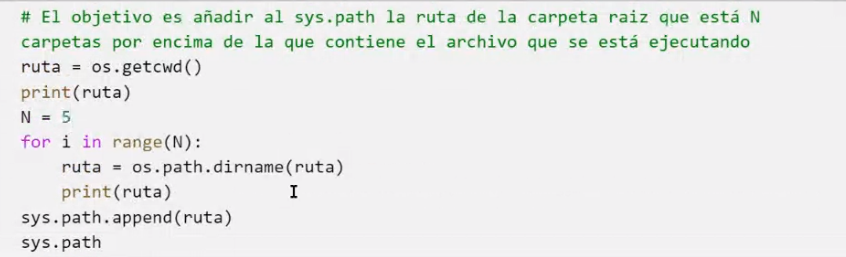


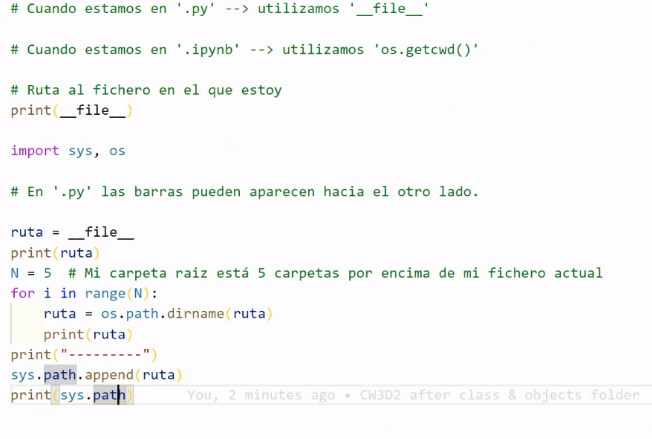
Se puede hasta hacer en bucle!



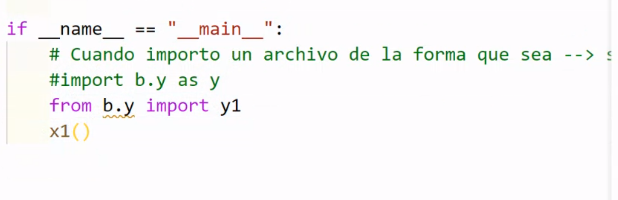
Y una vez llegamos a la carpeta raíz, podemos hacer un import enrutado! Con esto podemos acceder a las funciones de cualquier archivo del pc.

En resumen :



Cuando utilizamos jupyter, utilizamos os.getcwd, cuando utilizamos Python utilizamos \_\_file\_\_

Caso ejercicio: Para evitar la reciprocidad de los imports entre archivos (que da error), escribimos if \_\_name\_\_ == “\_\_main\_\_” encima del import para asignarle que solo importe cuando ejecutamos desde esa página. Es decir, ese parámetro de if nos indica que lo que hay debajo solo sucederá si corremos ese archivo. No sucederá si ese archivo se corre desde otro archivo. De esta manera evitamos los bucles de imports.

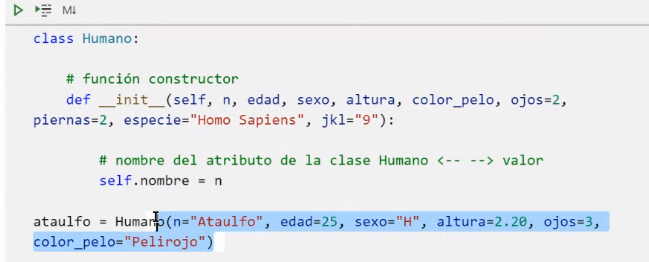


**Clases**

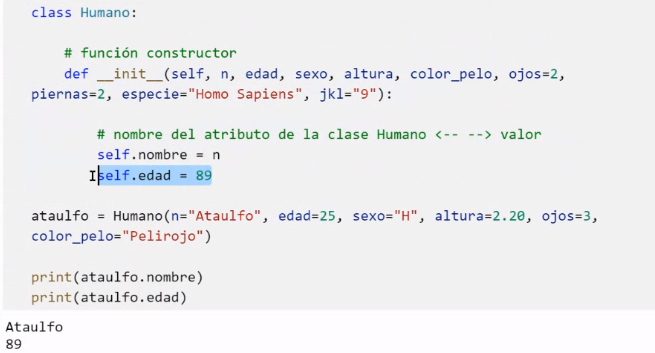
Creamos clase, creamos función en clase que crea atributos (función constructor) con parámetros.



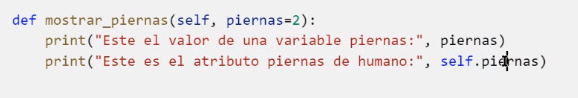
Le damos los atributos a la clase utilizando los parámetros. Es muy importante asignar a la instancia con self. Si no hay self, no tendrá asignado del constructor el atributo.



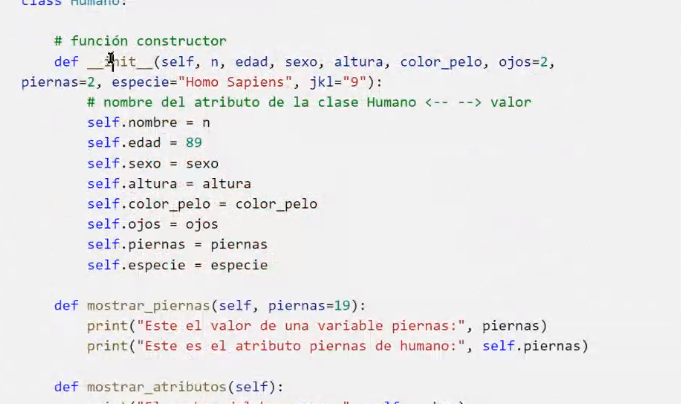
Self.wdaihdwa = / Es el último paso para otorgar a la instancia el atributo, por lo q si no lo vinculo al parámetro, me dará el valor que pueda darle directamente. Ejemplo: en este caso, me coje 89 y no 25, porque coje lo último que le doy.



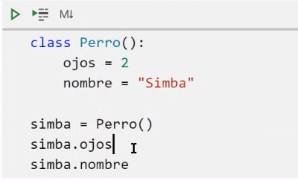
Para trabajar con atributos a funciones externas, tengo que utilizar el self. Siempre se obligatorio usar self:



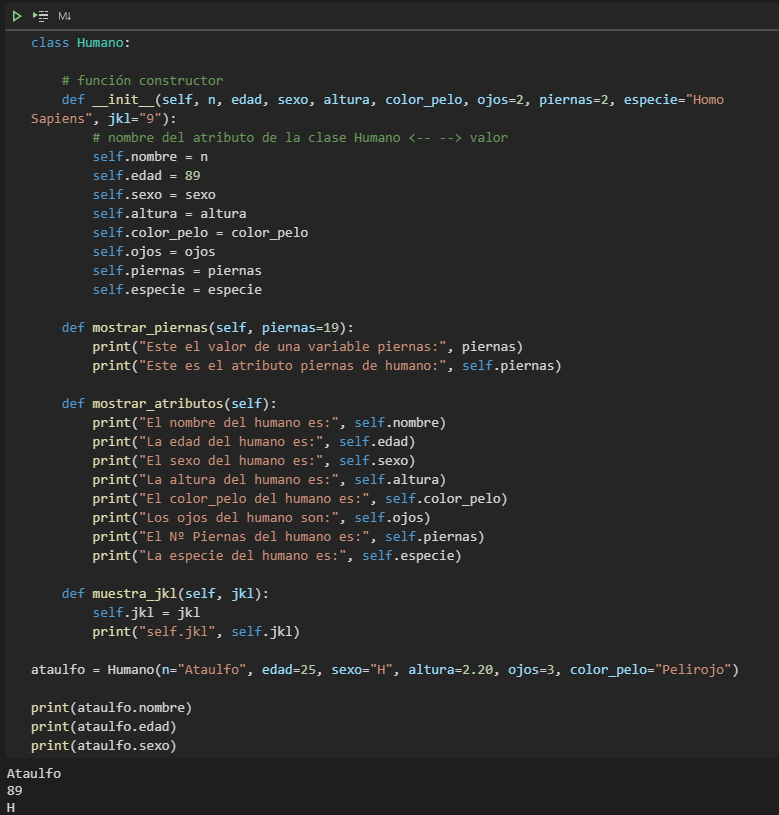
Así nos queda desarrollado:

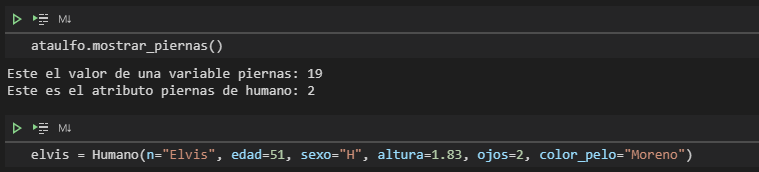


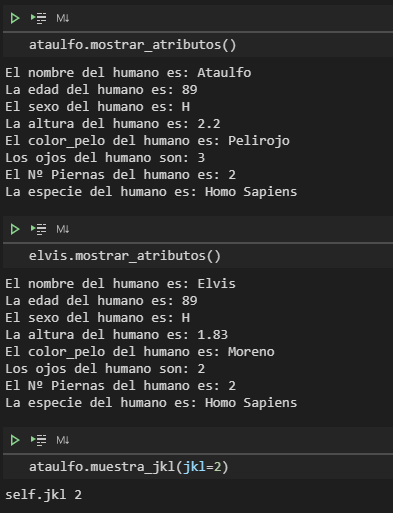
También, podemos llamar a la instancia sin función constructor



Todo humano:







Imports hacemos primero imports, objects es para jueves. En objects, el humano es el compañero, yo el orco. Enviar por github el ejercicio completado