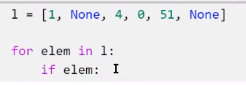
-Con este for, preguntamos si el valor es TRUE

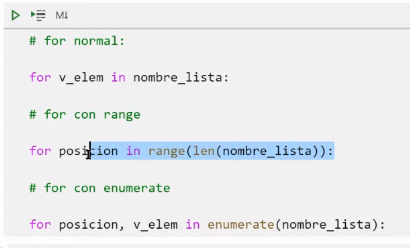


-Uso de for:

-Normal: uso la lista como bucle para cada posición.

-Range: uso la cantidad de elementos en lista commo bucle, ej (7)

-Enumerate: uso tanto el valor, como la posición en lista para programar (muchísimos usos).



**Dia 2 de funciones**

Las funciones con return devuelven el valor que asignamos. Las funciones con return tienen por defecto un valor none. Esto significa, que la función tiene valor 0, y si se la asignamos a una variable, le estaremos asignando a la variable un valor 0; a menos que le asignemos un valor con return a esa función.

¿Qué significa esto?

Def f():

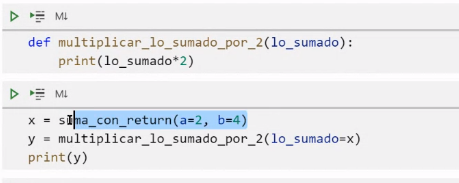
Return none (no hace falta que lo escribamos, por defecto siempre será none)

X = f()

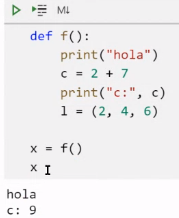
Si escribo x, la función se ejecutará. Si escribo print(x), me devolverá una ejecución de la función el valor de la función.

En este caso le asignamos a x la función suma con return que no aparece aquí, pero está. Despues le asignamos a y la función de arriba con el valor de x. Y luego printeamos el valor de x. El resultado es 6 12 y none, porque llamamos a 2 funciones y printeamos el valor de la última.

IMPORTANTE: Tenemos que al operar con funciones, siempre se imprimen sus prints, y al mostrar la variable final, nos devoilverá su return. Si es none, no aparecerá, a menos que sea con print.



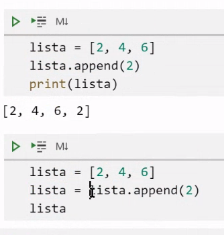
Otro ejemplo: como digo, sin el print final (si hubiese print en esa x final, habría none.



-El return es lo último que ocurre en una función!!!!!!!!!! Si pongo return arriba, la función termina.

Caso ejemplar:

-La función built in append, se muestra si hago print. Abajo, también la activo, pero no muestra nada porque el return por defecto es none. Si no printeo, no muestra.



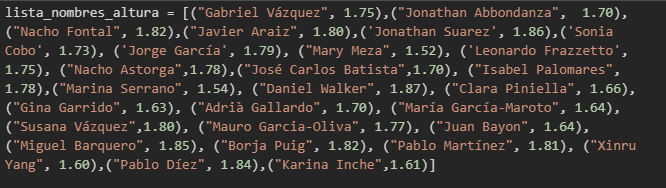
Documentación: Para crear una documentación, escribimos entre 3 comillas. Es un comentario ejecutado.

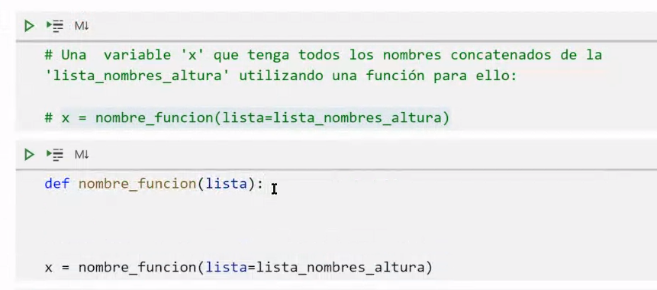
-pass: es para evitar que una función incompleta de error. Se cumplirá el return si se lo asignamos a la función, pero en ejecución la función quedará incompleta pero sin error.

-Cuidao aquí: En el primer caso, se elimina el 4 y se printea el valor de la variable lista. En el segundo caso, creamos una variable con valor 4, se ejecuta la función def resta\_dos (4): con lo que obtenemos x = 2, pero esta x = 2, solo existe dentro de la función, de ahí no va a salir, por lo que no podemos seguir utilizándola para futuros procesos. Para que en el segundo caso, x valga 2, fuera de la función, tengo que instalar un return con el valor de x.



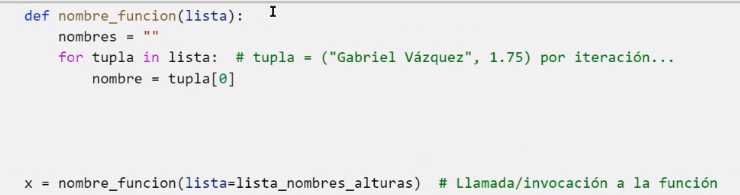
Ejercicio práctico en clase





-El primer paso es saber que nos pide una función y saber que nos tiene q llamar a la función el programa. Como en el enunciado, nos dice que, cuando llamemos a la función, especifiquemos (lista=lista\_nombres\_altura), dentro de la función tenemos que especificar que trabaje con (lista), para que identifique, que queremos trabajar sobre lista, con los datos que nos proporcione lista\_nombre\_altura. “El parámetro que pongamos en el nombre de la función, es el que va a recoger los parámetros de llamada a la función.” Por eso igualamos, en la llamada, para crear una segunda lista modificable.

El siguiente paso, es reconocer los datos de la lista, y saber como queremos modificarlos. Entonces, vemos que la lista, está compuesta por todo tuplas. Bien, hacemos un for que coja cada 1 de las tuplas y buscamos la posición 0 de cada tupla. Tenemos que crear una colección y añadir 1 a 1 el valor de la posición 0 de tupla. col = col + tupla [0].



Pero si printeo esto, me va a devolver none; tengo que poner un return col para darle a la función el valor final de col