Informacion extra Funciones

Como pudimos ver en las ultimas clases las funciones son muy poderosas cuando se usan de manera correcta.

¿Te acordas como declararlas?

```
def nombre_funcion(parametros):
    #logica de la funcion
    return resultado
```

¿Y los procedimientos?

```
def nombre_funcion(parametros):
    #logica de la funcion
    return
```

La unica diferencia entre una funcion y un procedimiento es que el segundo no retorna un valor cuando la invocamos, hace algo en su interior pero no lo podemos obtener.

Decoradores de Funciones

Las funciones como su definicion explica son cajas negras, porciones de codigo que luego de hacerlas podemos llevarlas a otro codigo y funcionaran perfectamente sin importar la logica interna, esto con el paso del tiempo complicará recordar perfectamente que hace o que parametros espera la funcion, ahi entran en juego los "Decoradores", que son etiquetas para guiarnos como programadores, tenemos de 3 tipos, *Entrada, salida y comentarios*

```
def validar_numero(parametros:)->int:
    """
    Esta funcion validará si la entrada del usuario es un número.
    """
    #Logica de la funcion
    return numero
```

En este ejemplo encontramos los tres:

- Entrada: se coloca en los parametros con ***:*** y seguido del tipo de dato esperado. (int, str, bool, object…)

- Entrada por defecto: podemos agregar un dato de entrada por defecto cuando el usuario coloque la funcion sin parametros, esto puede ser muy util cuando modificamos alguna funcion o queremos que haga cierta funcion por defecto. Esto se hace agregando despues del parametro | **None = valor_por_defecto** puedes reemplazar valor_por_defecto por cualquier cosa, texto, numero, variables, booleanos.
- Salida: se coloca luego de cerrar el parentesis de los parametros y antes de los : la forma de usarlo es -> y seguido el tipo de dato que retornará la funcion, cabe aclarar que si la logica dentro de la misma esta mal, este dato no se modificara, es una simple etiqueta. (int, str, bool, object…)
- **Comentarios: Se coloca luego de la definicion de la funcion con triples comillas (como los comentarios de multiplelinea), nos ayudará a tener una explicacion más detallada de la funcion.**

Pero ¿Y como uso mi funcion en otro archivo?

Con la palabra reservada *import* nos permite importar las funciones de un archivo al otro.

```
import funciones as f

resultado = f.sumar(2+3)
```

En este caso importamos el archivo funciones.py como (as) f para que sea mas sencillo y corto de utilizar. Se utiliza como si fuera un metodo, f.nombre_funcion().

Algunas consideraciones sobre el import:

- Tener en cuenta que cualquier parte del codigo que se encuentre en el archivo que importas que no sea una funcion se **ejecutará** en el archivo donde lo importes. Tene en cuenta esto y en el funciones.py **SOLAMENTE DEJA LAS FUNCIONES.**
- Tener en cuenta que solamente podes importar archivos que esten en la misma carpeta que tu archivo, si tenes funciones.py en otra carpeta no te será sencillo importarlo al archivo donde estas trabajando.
- Podes anidar IMPORT, como vieron en el archivo *funciones.py* que compartí podes colocar import dentro de otro archivos y aun asi funcionará en el que estas trabajando, prueben de importar funciones en un archivo.py y usar la funcion hablar()