**DOCUMENTACION TALLER 02**

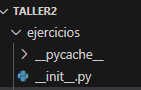
1. Se crea una carpeta madre la cual se llamará taller 2



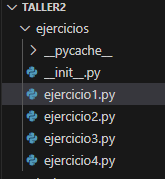
1. Se crea una subcarpeta que será donde estarán los ejercicios



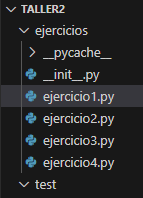
1. Se crear el archivo \_\_init\_\_.py (el archivo \_pycache se crea solo)



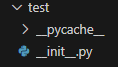
1. Se empieza a crear los archivos donde estarán las diferentes formulas en este caso se llamarán ejercicio1, ejercicio2, ejercicio3, ejercicio4



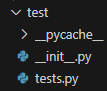
1. Luego crearemos una carpeta donde se realizarán los test esta su raíz es la principal carpeta test



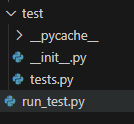
1. Se crea también el archivo \_\_init\_\_.py

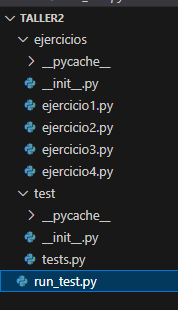


1. Se crea el archivo donde se hará las diferentes comparaciones donde se verificara si no hay errores. Tests.py



1. Se crea el archivo donde se corre los diferentes test Run\_test.py este esta en la carpeta de taller2





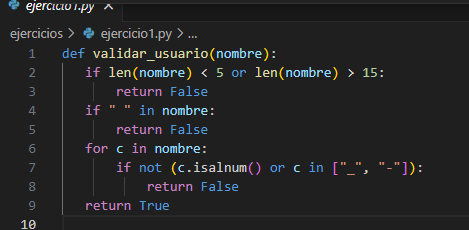
1. explicación de los otros ejercicios

## Dentro del ejercicio 1 se realizará Ejercicio 1: Validación de Nombre de Usuario (30 pts)

Especificaciones del Sistema

La función validar\_usuario(nombre) debe cumplir:

1. Longitud entre 5 y 15 caracteres.
2. Sin espacios.
3. Solo letras (mayúsculas/minúsculas) y números.



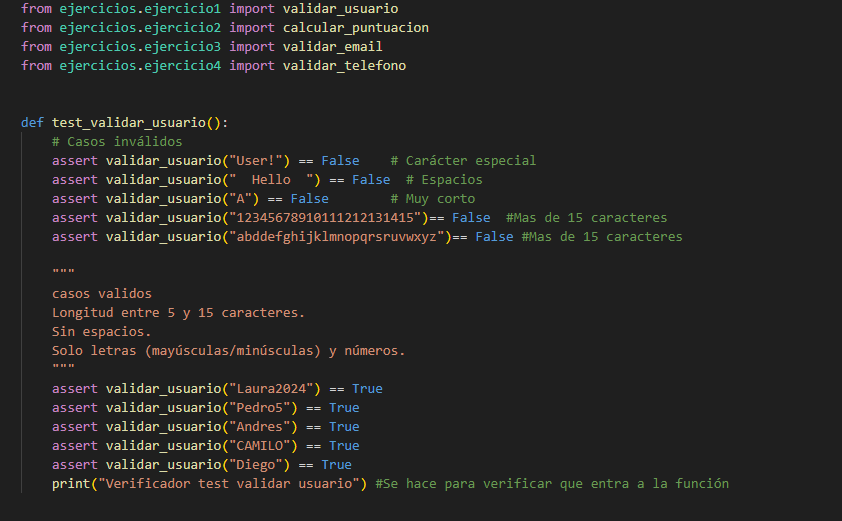
En este código de validara si cumple el tamaño del nombre con len diciendo que si menor que 5 o mayor a 15 sea que el nombre no cumple con lo especificado retornando a falso

Otra condición es si hay espacios en blanco no cumple y retorna a falso

Con el for recorre todo el nombre buscando solo tiene letras y números con , el método **isalnum()** se usa para verificar si la cadena no contiene  **caracteres alfanuméricos** (letras y números).

Y al final si no cumple quiere decir que esta correcto las indicaciones que se pidió en el ejercicio dando True.

1. **Luego en el archivo de verificación de test se importará las diferentes librerías y se realizará las diferentes comparaciones.**

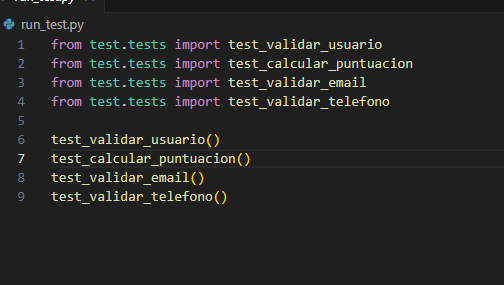
****

Aquí se hace una comparación donde se hace una función donde están los casos inválidos que no cumpla con lo que indica el ejercicio 1

Y la segunda parte que si cumplan

Se coloca un print para verificar que entra a la función.

1. Y por ultimo en el archivo run\_test se llama la función para que corra



1. Ejercicio 2: Sistema de Puntuación de Juegos (30 pts)

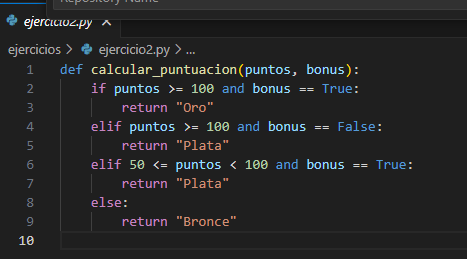
Especificaciones

La función calcular\_puntuacion(puntos, bonus) retorna:

"Oro" si puntos >= 100 y bonus == True.

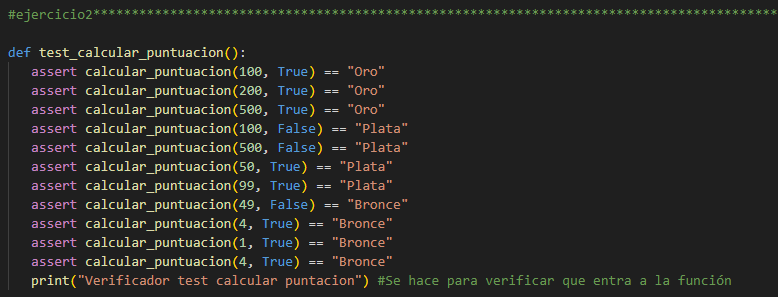
"Plata" si puntos >= 50 y puntos < 100.

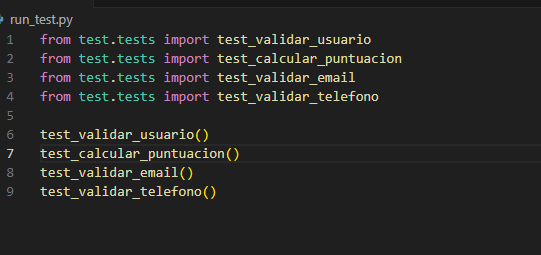
"Bronce" si puntos < 50 o bonus == False.



* Se calcula si los puntos son mayores o igual que 100 y tiene bonus dará oro
* Si los si los puntos son mayores o igual que 100 y no tiene bonus dará plata
* Si están entre 50 y 100 y tiene bonus dará plata
* De lo contrario será bronce

1. Se hace la verificación que cumpla con la puntación el bonus y lo que obtiene



1. Se corre los test 
2. Ejercicio 3 (Avanzado): Validación de Email (60 pts)

Especificaciones

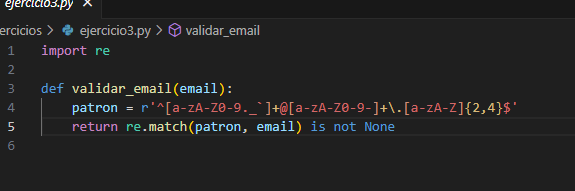
La función validar\_email(email) debe cumplir:

Formato usuario@dominio.extension.

usuario: Solo letras, números, ., \_, y ``.

dominio: Letras, números, y guiones.

extension: 2 a 4 letras.



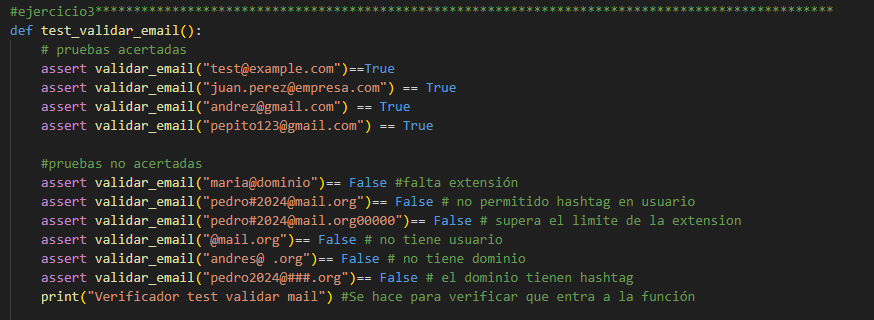
Esta función verifica cada parte separados por un +

Formato de usuario 

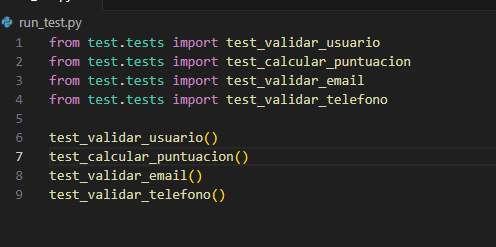
Formato de dominio 

Formato de etension 

1. Se hace la función para verificar que este correcto

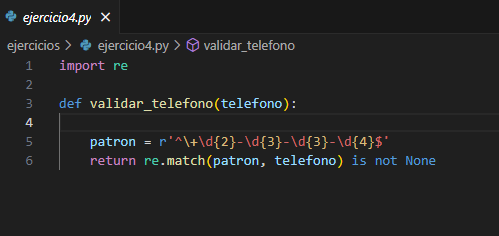


1. Se hace correr las fuciones



1. Ejercicio 4: Validación de teléfono (40 pts)

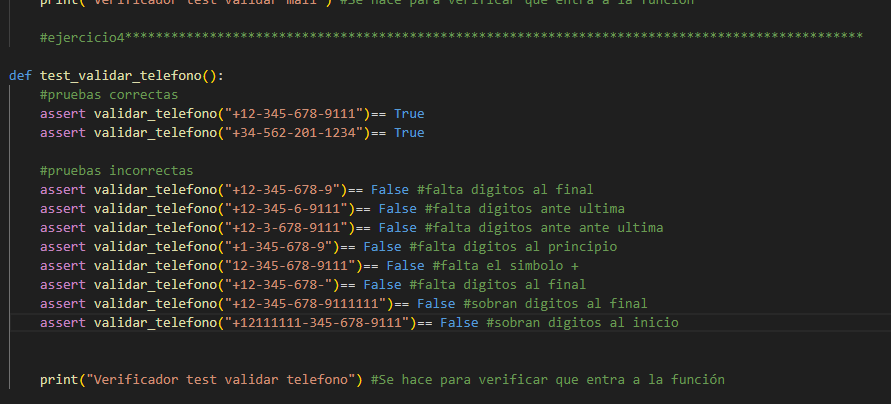
Ejercicio: Diseñar casos de prueba para una función que valide números de teléfono con formato +XX-XXX-XXX-XXXX.



Aquí se verifica que

* Empiece con el símbolo + 
* Luego tenga 2 dígitos 
* Luego 3 dígitos 
* Luego 4 digitos 

1. Se realiza todas las verificaciones necesarias para comprobar que no exista errores



1. Se realiza el correr el test

