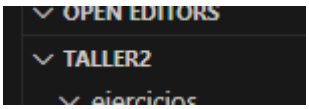
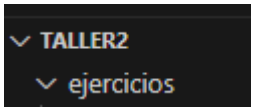
DOCUMENTACION TALLER 02

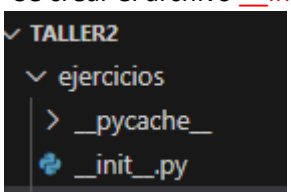
1. Se crea una carpeta madre la cual se llamará taller 2



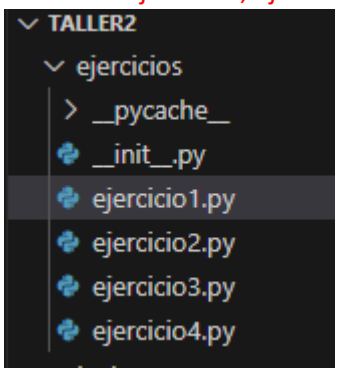
2. Se crea una subcarpeta que será donde estarán los ejercicios



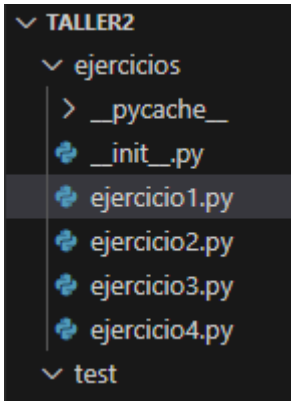
3. Se crear el archivo __init__.py (el archivo _pycache se crea solo)



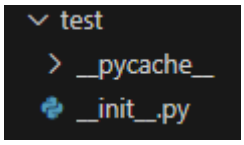
4. Se empieza a crear los archivos donde estarán las diferentes formulas en este caso se llamarán ejercicio1, ejercicio2, ejercicio3, ejercicio4



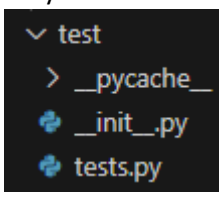
5. Luego crearemos una carpeta donde se realizarán los test esta su raíz es la principal carpeta test



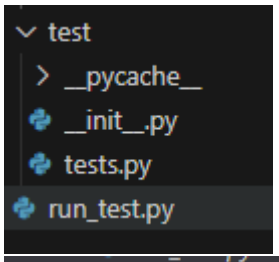
6. Se crea también el archivo __init__.py

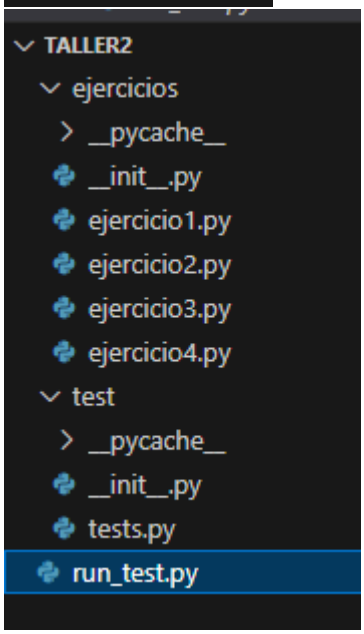


7. Se crea el archivo donde se hará las diferentes comparaciones donde se verificara si no hay errores. Tests.py



8. Se crea el archivo donde se corre los diferentes test Run_test.py este esta en la carpeta de taller2





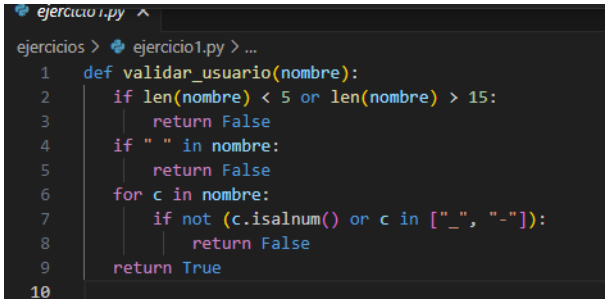
9. explicación de los otros ejercicios

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.



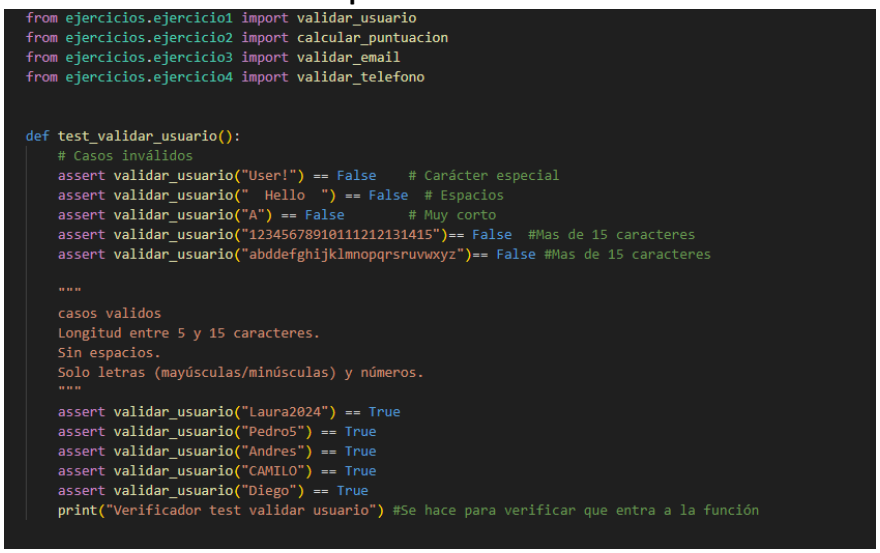
En este código de validara si cumple el tamaño del nombre con len diciendo que si menor que 5 o mayor a 15 sea que el nombre no cumple con lo especificado retornando a falso

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.

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

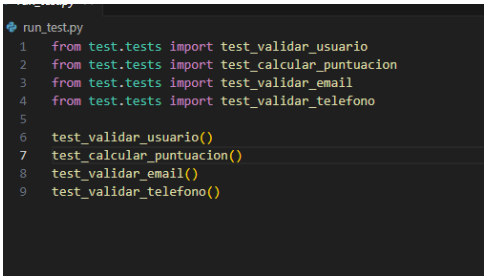


Aquí se hace una comparación donde se hace una función donde están los casos inválidos que no cumpla con lo que indica el ejercicio 1

Y la segunda parte que si cumplan

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

11. Y por ultimo en el archivo run_test se llama la función para que corra

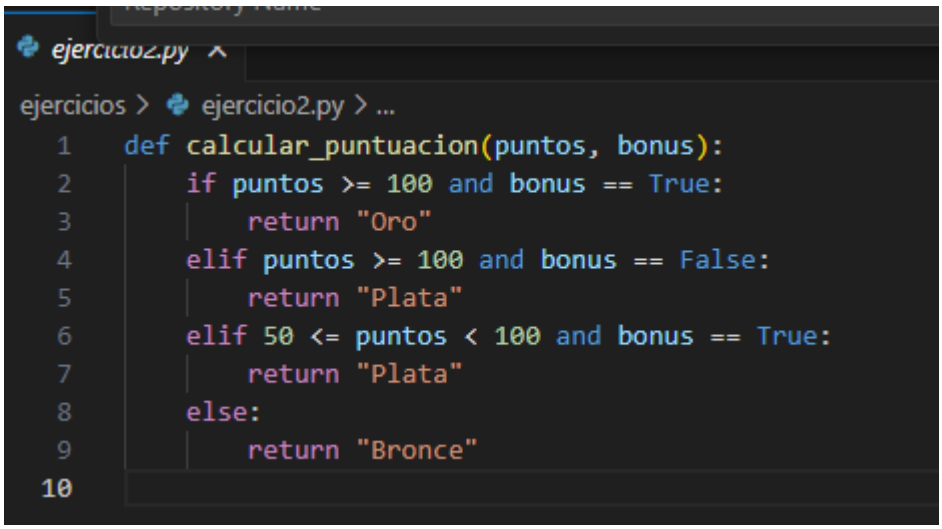


12. Ejercicio 2: Sistema de Puntuación de Juegos (30 pts)

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

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

```python
#ejercicio2**********************************************************************************************

def test_calcular_puntuacion():
    assert calcular_puntuacion(100, True) == "Oro"
    assert calcular_puntuacion(200, True) == "Oro"
    assert calcular_puntuacion(500, True) == "Oro"
    assert calcular_puntuacion(100, False) == "Plata"
    assert calcular_puntuacion(500, False) == "Plata"
    assert calcular_puntuacion(50, True) == "Plata"
    assert calcular_puntuacion(99, True) == "Plata"
    assert calcular_puntuacion(49, False) == "Bronce"
    assert calcular_puntuacion(4, True) == "Bronce"
    assert calcular_puntuacion(1, True) == "Bronce"
    assert calcular_puntuacion(4, True) == "Bronce"
    print("Verificador test calcular puntacion") #Se hace para verificar que entra a la función
```

14. Se corre los test

run_test.py

```python
from test.tests import test_validar_usuario
from test.tests import test_calcular_puntuacion
from test.tests import test_validar_email
from test.tests import test_validar_telefono

test_validar_usuario()
test_calcular_puntuacion()
test_validar_email()
test_validar_telefono()
```

15. 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.

```python
import re

def validar_email(email):
    patron = r'^[a-zA-Z0-9._`]+@[a-zA-Z0-9-]+\.[a-zA-Z]{2,4}$'
    return re.match(patron, email) is not None
```

Esta función verifica cada parte separados por un +

Formato de usuario `r'^[a-zA-Z0-9._`]`

Formato de dominio `+@[a-zA-Z0-9-]`

Formato de etension `+\.[a-zA-Z]{2,4}$'`

16. Se hace la función para verificar que este correcto

```python
#ejercicio3**********************************************************************************************
def test_validar_email():
    # pruebas acertadas
    assert validar_email("test@example.com")==True
    assert validar_email("juan.perez@empresa.com") == True
    assert validar_email("andrez@gmail.com") == True
    assert validar_email("pepito123@gmail.com") == True

    #pruebas no acertadas
    assert validar_email("maria@dominio")== False #falta extensión
    assert validar_email("pedro#2024@mail.org")== False # no permitido hashtag en usuario
    assert validar_email("pedro#2024@mail.org00000")== False # supera el limite de la extension
    assert validar_email("@mail.org")== False # no tiene usuario
    assert validar_email("andres@ .org")== False # no tiene dominio
    assert validar_email("pedro2024@###.org")== False # el dominio tienen hashtag
    print("Verificador test validar mail") #Se hace para verificar que entra a la función
```

17. Se hace correr las fuciones

run_test.py

```python
from test.tests import test_validar_usuario
from test.tests import test_calcular_puntuacion
from test.tests import test_validar_email
from test.tests import test_validar_telefono

test_validar_usuario()
test_calcular_puntuacion()
test_validar_email()
test_validar_telefono()
```

18. 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.

ejercicio4.py

ejercicios > ejercicio4.py > validar_telefono

```python
import re

def validar_telefono(telefono):

    patron = r'^\+\d{2}-\d{3}-\d{3}-\d{4}$'
    return re.match(patron, telefono) is not None
```

Aquí se verifica que

- Empiece con el símbolo + `r'^\+\`
- Luego tenga 2 dígitos `+\d{2}`
- Luego 3 dígitos `\d{3}`
- Luego 4 digitos `\d{4}$'`

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

```python
    print("Verificador test validar mail") #Se hace para verificar que entra a la función

    #ejercicio4*****************************************************************************************

def test_validar_telefono():
    #pruebas correctas
    assert validar_telefono("+12-345-678-9111")== True
    assert validar_telefono("+34-562-201-1234")== True

    #pruebas incorrectas
    assert validar_telefono("+12-345-678-9")== False #falta digitos al final
    assert validar_telefono("+12-345-6-9111")== False #falta digitos ante ultima
    assert validar_telefono("+12-3-678-9111")== False #falta digitos ante ante ultima
    assert validar_telefono("+1-345-678-9")== False #falta digitos al principio
    assert validar_telefono("12-345-678-9111")== False #falta el simbolo +
    assert validar_telefono("+12-345-678-")== False #falta digitos al final
    assert validar_telefono("+12-345-678-9111111")== False #sobran digitos al final
    assert validar_telefono("+12111111-345-678-9111")== False #sobran digitos al inicio


    print("Verificador test validar telefono") #Se hace para verificar que entra a la función
```

20. Se realiza el correr el test

run_test.py

```python
from test.tests import test_validar_usuario
from test.tests import test_calcular_puntuacion
from test.tests import test_validar_email
from test.tests import test_validar_telefono

test_validar_usuario()
test_calcular_puntuacion()
test_validar_email()
test_validar_telefono()
```