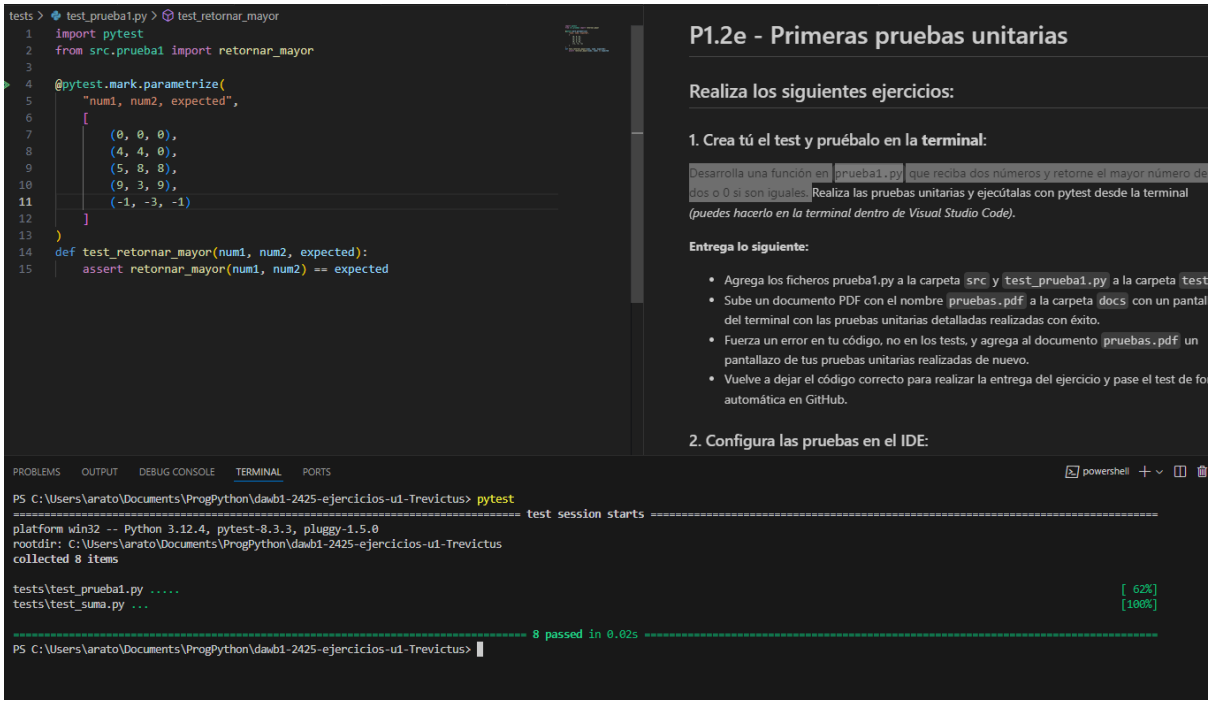
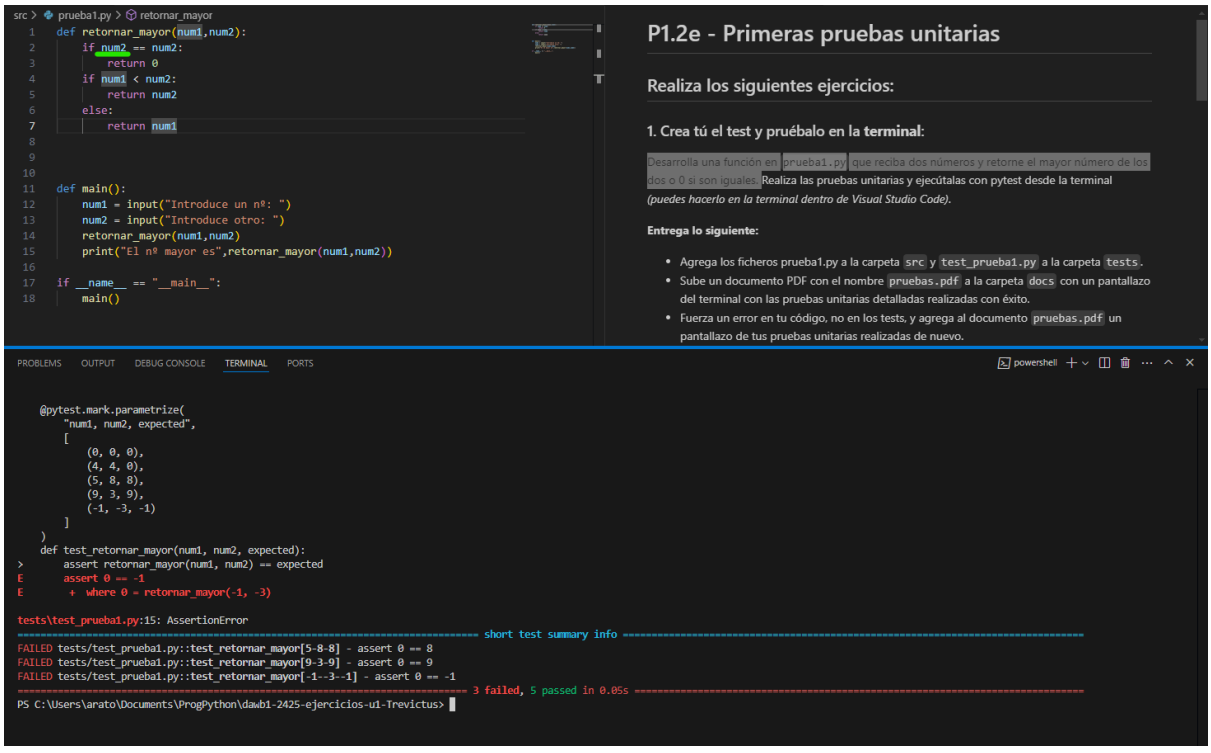

tests > test_prueba1.py > test_retornar_mayor
import pytest
from src.prueba1 import retornar_mayor
@pytest.mark.parametrize(
"num1, num2, expected",
[
(0, 0, 0),
(4, 4, 0),
(5, 8, 8),
(9, 3, 9),
(-1, -3, -1)
]
)
def test_retornar_mayor(num1, num2, expected):
assert retornar_mayor(num1, num2) == expected
P1.2e - Primeras pruebas unitarias
Realiza los siguientes ejercicios:
1. Crea tú el test y pruébalo en la terminal:
Desarrolla una función en prueba1.py que reciba dos números y retorne el mayor número de dos o 0 si son iguales. Realiza las pruebas unitarias y ejecútalas con pytest desde la terminal (puedes hacerlo en la terminal dentro de Visual Studio Code).
Entrega lo siguiente:
Agrega los ficheros prueba1.py a la carpeta src y test_prueba1.py a la carpeta test
Sube un documento PDF con el nombre pruebas.pdf a la carpeta docs con un pantal del terminal con las pruebas unitarias detalladas realizadas con éxito.
Fuerza un error en tu código, no en los tests, y agrega al documento pruebas.pdf un pantallazo de tus pruebas unitarias realizadas de nuevo.
Vuelve a dejar el código correcto para realizar la entrega del ejercicio y pase el test de fo automática en GitHub.
2. Configura las pruebas en el IDE:
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
PS C:\Users\arato\Documents\ProgPython\dawb1-2425-ejercicios-u1-Trevictus> pytest
test session starts
platform win32 -- Python 3.12.4, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\arato\Documents\ProgPython\dawb1-2425-ejercicios-u1-Trevictus
collected 8 items
tests\test_prueba1.py ..... [62%]
tests\test_suma.py ... [100%]
8 passed in 0.02s
PS C:\Users\arato\Documents\ProgPython\dawb1-2425-ejercicios-u1-Trevictus>


src > prueba1.py > retornar_mayor
def retornar_mayor(num1,num2):
if num2 == num2:
return 0
if num1 < num2:
return num2
else:
return num1
def main():
num1 = input("Introduce un nº: ")
num2 = input("Introduce otro: ")
retornar_mayor(num1,num2)
print("El nº mayor es",retornar_mayor(num1,num2))
if __name__ == "__main__":
main()
P1.2e - Primeras pruebas unitarias
Realiza los siguientes ejercicios:
1. Crea tú el test y pruébalo en la terminal:
Desarrolla una función en prueba1.py que reciba dos números y retorne el mayor número de los dos o 0 si son iguales. Realiza las pruebas unitarias y ejecútalas con pytest desde la terminal (puedes hacerlo en la terminal dentro de Visual Studio Code).
Entrega lo siguiente:
Agrega los ficheros prueba1.py a la carpeta src y test_prueba1.py a la carpeta tests.
Sube un documento PDF con el nombre pruebas.pdf a la carpeta docs con un pantallazo del terminal con las pruebas unitarias detalladas realizadas con éxito.
Fuerza un error en tu código, no en los tests, y agrega al documento pruebas.pdf un pantallazo de tus pruebas unitarias realizadas de nuevo.
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
@pytest.mark.parametrize(
"num1, num2, expected",
[
(0, 0, 0),
(4, 4, 0),
(5, 8, 8),
(9, 3, 9),
(-1, -3, -1)
]
)
def test_retornar_mayor(num1, num2, expected):
> assert retornar_mayor(num1, num2) == expected
E assert 0 == -1
E + where 0 = retornar_mayor(-1, -3)
tests\test_prueba1.py:15: AssertionError
short test summary info
FAILED tests/test_prueba1.py::test_retornar_mayor[5-8-8] - assert 0 == 8
FAILED tests/test_prueba1.py::test_retornar_mayor[9-3-9] - assert 0 == 9
FAILED tests/test_prueba1.py::test_retornar_mayor[-1--3--1] - assert 0 == -1
3 failed, 5 passed in 0.05s
PS C:\Users\arato\Documents\ProgPython\dawb1-2425-ejercicios-u1-Trevictus>