

## PRUEBA UNITARIA 1

### TERMINAL

Código sin error:

```
src > prueba1.py > ...
1  def retornar_numero(num1, num2):
2      if num1 == num2:
3          return 0
4      elif num1 < num2:
5          return num2
6      else:
7          return num1
8
9  def main():
10     num1 = input("Introduce el primer número: ")
11     num2 = input("Introduce el otro número: ")
12
13     retornar_numero(num1, num2)
14
15     print("El número mayor es: ", retornar_numero(num1, num2))
16
17 if __name__ == "__main__":
18     main()
```

Código prueba:

```
tests > test_prueba1.py > ...
1  import pytest
2  from src.prueba1 import retornar_numero
3
4  @pytest.mark.parametrize("num1, num2, expected", [
5
6      (0, 0, 0),
7      (4, 4, 0),
8      (5, 8, 8)
9
10 ])
11
12 def test_retornar_numero(num1, num2, expected):
13     assert retornar_numero(num1, num2) == expected
```

Prueba:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\Usuario\Documents\ReposGit\dawb1-2425-ejercicios-u1-PabloFdez06> pytest
===== test session starts =====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\Usuario\Documents\ReposGit\dawb1-2425-ejercicios-u1-PabloFdez06
collected 3 items

tests\test_prueba1.py ... [100%]

===== 3 passed in 0.05s =====
PS C:\Users\Usuario\Documents\ReposGit\dawb1-2425-ejercicios-u1-PabloFdez06>
```

Código forzando error:

```
src > prueba1.py > retornar_numero
1  def retornar_numero(num1, num2):
2      if num1 == num2:
3          return 0
4      elif num1 < num2:
5          return num1,2
6      else:
7          return num1
8
9  def main():
10     num1 = input("Introduce el primer número: ")
11     num2 = input("Introduce el otro número: ")
12
13     resultado = retornar_numero(num1, num2)
14
15     print("El número mayor es: ", resultado)
16
17 if __name__ == "__main__":
18     main()
```

Prueba:

```
===== FAILURES =====
test_retornar_numero[5-8-8]

num1 = 5, num2 = 8, expected = 8

@pytest.mark.parametrize("num1, num2, expected", [
    (0, 0, 0),
    (4, 4, 0),
    (5, 8, 8)
])

def test_retornar_numero(num1, num2, expected):
    assert retornar_numero(num1, num2) == expected
>
E       assert (5, 2) == 8
E       + where (5, 2) = retornar_numero(5, 8)

tests/test_prueba1.py:13: AssertionError
----- short test summary info -----
FAILED tests/test_prueba1.py::test_retornar_numero[5-8-8] - assert (5, 2) == 8
----- 1 failed, 2 passed in 0.09s -----
PS C:\Users\UsuarioT\Documents\ReposGit\dawb1-2425-ejercicios-u1-PabloFdez06>
```

Desde el IDE:

```
File Edit Selection View Go Run Terminal Help
prueba1.py U x test_ej11.py U ej05_def2.py U
src > prueba1.py > ...
1  def retornar_numero(num1, num2):
2      if num1 == num2:
3          return 0
4      elif num1 < num2:
5          return num2
6      else:
7          return num1
8
9  def main():
10     num1 = input("Introduce el primer número: ")
11     num2 = input("Introduce el otro número: ")
12
13     resultado = retornar_numero(num1, num2)
14
15     print("El número mayor es: ", resultado)
16
17 if __name__ == "__main__":
18     main()
```

Desde el IDE forzando error:

```
File Edit Selection View Go Run Terminal Help
dawb1-2425-ejercicios-u1-PabloDef06

pruebas1.py x test_g11.py U ej05_def2.py U ej04_def2.py U ej04_def.py test_g05.py U test_g04.py U test_g02.py U test_g01.py U test_prueba1.py U
5/6 533ms
dawb1-2425-ejercicios-u1-PabloDef06
tests
test_g14.py
test_prueba1.py
test_retornar_numero
[0-0-0]
[4-4-0]
[5-8-8]

src > pruebas1.py > retornar_numero
1 def retornar_numero(num1, num2):
2     if num1 == num2:
3         return 0
4     elif num1 < num2:
5         return num1,2
6     else:
7         return num1
8
9 def main():
10     num1 = input("Introduce el primer número: ")
11     num2 = input("Introduce el otro número: ")
12
13     resultado = retornar_numero(num1, num2)
14     print("El número mayor es: ", resultado)
15
16
17 if __name__ == "__main__":
18     main()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS
tests/test_g14.py ... [ 50%]
tests/test_prueba1.py ..F [100%]

===== FAILURES =====
test_retornar_numero[5-8-8]

num1 = 5, num2 = 8, expected = 8

@pytest.mark.parametrize("num1, num2, expected", [
    (0, 0, 0),
    (4, 4, 0),
    (5, 8, 8)
])
def test_retornar_numero(num1, num2, expected):
    assert retornar_numero(num1, num2) == expected
E       assert (5, 2) == 8
E        + where (5, 2) = retornar_numero(5, 8)

tests/test_prueba1.py:12: AssertionError

===== short test summary info =====
FAILED tests/test_prueba1.py::test_retornar_numero[5-8-8] - assert (5, 2) == 8
===== 1 failed, 5 passed in 0.10s =====
Finished running tests!
```