

## ej01\_def2.py

Prueba correcta

En esta primera prueba se comprueba tanto la función saludo como el método .title()

```
test > test_ej01_def2.py > test_saludo_params
1 import pytest
2 from src.ej01_def2 import saludo
3
4 @pytest.mark.parametrize(
5     'nom, expected',
6     [
7         ('Juan', 'Hola, Juan.'),
8         ('pedro', 'Hola, Pedro.'),
9         ('MANOLITO', 'Hola, Manolito.'),
10        ('rAmIRo', 'Hola, Ramiro.'),
11        ('joSEmi', 'Hola, Josemi.'),
12    ]
13 )
14
15 def test_saludo_params(nom, expected):
16     assert saludo(nom) == expected

PS C:\Users\UsuarioT\prog1> pytest -v ./test/test_ej01_def2.py
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\UsuarioT\prog1\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\UsuarioT\prog1
collected 5 items

test/test_ej01_def2.py::test_saludo_params[Juan-Hola, Juan.] PASSED [ 20%]
test/test_ej01_def2.py::test_saludo_params[pedro-Hola, Pedro.] PASSED [ 40%]
test/test_ej01_def2.py::test_saludo_params[MANOLITO-Hola, Manolito.] PASSED [ 60%]
test/test_ej01_def2.py::test_saludo_params[rAmIRo-Hola, Ramiro.] PASSED [ 80%]
test/test_ej01_def2.py::test_saludo_params[joSEmi-Hola, Josemi.] PASSED [100%]

===== 5 passed in 0.05s =====
PS C:\Users\UsuarioT\prog1>
```

Prueba con error

En esta segunda prueba se fuerza un error eliminando algunas letras de los resultados

```
test > test_ej01_def2.py > test_saludo_params
1 import pytest
2 from src.ej01_def2 import saludo
3
4 @pytest.mark.parametrize(
5     'nom, expected',
6     [
7         ('Juan', 'Hola, Juan.'),
8         ('pedro', 'Hola, Pdro.'),
9         ('MANOLITO', 'Hola, Manolito.'),
10        ('rAmIRo', 'Hola, Ramiro.'),
11        ('joSEmi', 'Hola, Jsemi.'),
12    ]
13 )
14
15 def test_saludo_params(nom, expected):
16     assert saludo(nom) == expected

def test_saludo_params(nom, expected):
> assert saludo(nom) == expected
E       AssertionError: assert 'Hola, Josemi.' == 'Hola, Jsemi.'
E
E       - Hola, Jsemi.
E       + Hola, Josemi.
E       ?      +

test\test_ej01_def2.py:16: AssertionError
===== short test summary info =====
FAILED test/test_ej01_def2.py::test_saludo_params[pedro-Hola, Pdro.] - AssertionError: assert 'Hola, Pedro.' == 'Hola, Pdro.'
FAILED test/test_ej01_def2.py::test_saludo_params[joSEmi-Hola, Jsemi.] - AssertionError: assert 'Hola, Josemi.' == 'Holla, Jsemi.'
===== 2 failed, 3 passed in 0.12s =====
PS C:\Users\UsuarioT\prog1>
```

## ej02\_def2.py

Prueba correcta

En esta primera prueba se comprueba la funcion pagoTotal y que redondee a dos decimales el resultado

```
test > test_ej02_def2.py > test_pagoTotal
1 import pytest
2 from src.ej02_def2 import pagoTotal
3
4 @pytest.mark.parametrize(
5     'horas, precioHoras, expected',
6     [
7         (10, 8, 80),
8         (20.2312, 21.3213, 431.36),
9         (100000000, 1, 100000000),
10        (25.2, 34, 856.8),
11        (-123, 9.4, -1156.2),
12    ]
13 )
14 def test_pagoTotal(horas, precioHoras, expected):
15     assert pagoTotal(horas, precioHoras) == expected
```

```
===== no tests ran in 0.04s =====
PS C:\Users\UsuarioT\prog1> pytest -v ./test/test_ej02_def2.py
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\UsuarioT\prog1\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\UsuarioT\prog1
collected 5 items

test/test_ej02_def2.py::test_pagoTotal[10-8-80] PASSED [ 20%]
test/test_ej02_def2.py::test_pagoTotal[20.2312-21.3213-431.36] PASSED [ 40%]
test/test_ej02_def2.py::test_pagoTotal[100000000-1-100000000] PASSED [ 60%]
test/test_ej02_def2.py::test_pagoTotal[25.2-34-856.8] PASSED [ 80%]
test/test_ej02_def2.py::test_pagoTotal[-123-9.4--1156.2] PASSED [100%]

===== 5 passed in 0.07s =====
PS C:\Users\UsuarioT\prog1>
```

Prueba con error

En esta segunda prueba se eliminan tanto dígitos como decimales comprobando ambas.

```
test > test_ej02_def2.py > test_pagoTotal
1 import pytest
2 from src.ej02_def2 import pagoTotal
3
4 @pytest.mark.parametrize(
5     'horas, precioHoras, expected',
6     [
7         (10, 8, 80),
8         (20.2312, 21.3213, 431.3),
9         (100000000, 1, 100000000),
10        (25.2, 34, 856.8),
11        (-123, 9.4, -1156.2),
12    ]
13 )
14 def test_pagoTotal(horas, precioHoras, expected):
15     assert pagoTotal(horas, precioHoras) == expected
```

```

        (100000000, 1, 100000000),
        (25.2, 34, 856.8),
        (-123, 9.4, -1156.2),
    ]
)
def test_pagoTotal(horas, precioHoras, expected):
> assert pagoTotal(horas, precioHoras) == expected
E       assert 100000000 == 100000000
E       + where 100000000 = pagoTotal(100000000, 1)

test/test_ej02_def2.py:15: AssertionError
===== short test summary info =====
FAILED test/test_ej02_def2.py::test_pagoTotal[20.2312-21.3213-431.3] - assert 431.36 == 431.3
FAILED test/test_ej02_def2.py::test_pagoTotal[100000000-1-100000000] - assert 100000000 == 100000000
===== 2 failed, 3 passed in 0.13s =====
PS C:\Users\UsuarioT\prog1>
```

## ej04\_def2.py

### Prueba correcta

En esta primera prueba se prueba tanto la función `grados_celsius` como el redondeo.

```
test > test_ej04_def2.py > test_grados_fahrenheit_params
1 import pytest
2 from src.ej04_def2 import grados_celsius
3
4 @pytest.mark.parametrize(
5     "grados_fahrenheit, expected",
6     [
7         (100.12, 37.84),
8         (21.09, -6.06),
9         (329.12, 165.07),
10        (-21, -29.44),
11        (200, 93.33),
12    ]
13 )
14 def test_grados_fahrenheit_params(grados_fahrenheit, expected):
15     assert grados_celsius(grados_fahrenheit) == expected
```

PS C:\Users\UsuarioT\prog1> pytest ./test/test\_ej04\_def2.py  
===== test session starts =====  
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0  
rootdir: C:\Users\UsuarioT\prog1  
collected 5 items  
  
test\test\_ej04\_def2.py ..... [100%]  
  
===== 5 passed in 0.03s =====  
PS C:\Users\UsuarioT\prog1>

### Prueba con error

En esta segunda prueba se eliminan tanto dígitos como decimales comprobando ambas.

```
test > test_ej04_def2.py > test_grados_fahrenheit_params
1 import pytest
2 from src.ej04_def2 import grados_celsius
3
4 @pytest.mark.parametrize(
5     "grados_fahrenheit, expected",
6     [
7         (100.12, 37.84),
8         (21.09, -6.06),
9         (329.12, 165.07),
10        (-21, -29.44),
11        (200, 91.33),
12    ]
13 )
14 def test_grados_fahrenheit_params(grados_fahrenheit, expected):
15     assert grados_celsius(grados_fahrenheit) == expected
```

test\test\_ej04\_def2.py:15: AssertionError  
===== short test summary info =====  
FAILED test\test\_ej04\_def2.py::test\_grados\_fahrenheit\_params[200-91.33] - assert 93.33 == 91.33  
===== 1 failed, 4 passed in 0.10s =====  
PS C:\Users\UsuarioT\prog1>

## ej05\_def2.py

Prueba correcta

En esta primera prueba se prueba tanto la función `calcula_precio` como el redondeo.

```
test > test_ej05_def2.py > test_calcula_precio_params
1 import pytest
2 from src.ej05_def2 import calcula_precio
3
4 @pytest.mark.parametrize(
5     "importe, iva, expected",
6     [
7         (100, 21, 121),
8         (150, 21, 181.5),
9         (123.1231, 39.21312321, 171.4),
10        (11.21, 300, 44.84),
11        (1920.1281, 30, 2496.17),
12    ]
13 )
14 def test_calcula_precio_params(importe, iva, expected):
15     assert calcula_precio(importe, iva) == expected
```

```
PS C:\Users\UsuarioT\prog1> pytest ./test/test_ej05_def2.py
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\UsuarioT\prog1
collected 5 items

test\test_ej05_def2.py ..... [100%]

===== 5 passed in 0.05s =====
PS C:\Users\UsuarioT\prog1>
```

Prueba con error

En esta segunda prueba se eliminan tanto dígitos como decimales comprobando ambas.

```
test > test_ej05_def2.py > test_calcula_precio_params
1 import pytest
2 from src.ej05_def2 import calcula_precio
3
4 @pytest.mark.parametrize(
5     "importe, iva, expected",
6     [
7         (100, 21, 121),
8         (150, 21, 181.5),
9         (123.1231, 39.21312321, 172.4),
10        (11.21, 300, 44.84),
11        (1920.1281, 30, 2496.27),
12    ]
13 )
14 def test_calcula_precio_params(importe, iva, expected):
15     assert calcula_precio(importe, iva) == expected
```

```
test\test_ej05_def2.py:15: AssertionError
===== short test summary info =====
FAILED test\test_ej05_def2.py::test_calcula_precio_params[123.1231-39.21312321-172.4] - assert 171.4 == 172.4
FAILED test\test_ej05_def2.py::test_calcula_precio_params[1920.1281-30-2496.27] - assert 2496.17 == 2496.27
===== 2 failed, 3 passed in 0.12s =====
PS C:\Users\UsuarioT\prog1>
```

## ej11\_def2.py

Prueba correcta

En esta primera prueba se prueba tanto la función sumNum y la conversión a INT

```
test > test_ej11_def2.py > test_sumNum
1 import pytest
2 from src.ej11_def2 import sumNum
3
4 @pytest.mark.parametrize(
5     'num, expected',
6     [
7         (1, 1),
8         (2, 3),
9         (3, 6),
10        (4, 10),
11        (5, 15),
12    ]
13 )
14 def test_sumNum(num, expected):
15     assert sumNum(num) == expected
```

```
===== 1 failed, 4 passed in 0.10s =====
PS C:\Users\UsuarioT\prog1> pytest -v ./test/test_ej11_def2.py
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\UsuarioT\prog1\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\UsuarioT\prog1
collected 5 items

test/test_ej11_def2.py::test_sumNum[1-1] PASSED [ 20%]
test/test_ej11_def2.py::test_sumNum[2-3] PASSED [ 40%]
test/test_ej11_def2.py::test_sumNum[3-6] PASSED [ 60%]
test/test_ej11_def2.py::test_sumNum[4-10] PASSED [ 80%]
test/test_ej11_def2.py::test_sumNum[5-15] PASSED [100%]

===== 5 passed in 0.05s =====
PS C:\Users\UsuarioT\prog1>
```

Prueba con error

En esta segunda prueba se eliminan tanto dígitos como INTs comprobando ambas.

```
test > test_ej11_def2.py > test_sumNum
1 import pytest
2 from src.ej11_def2 import sumNum
3
4 @pytest.mark.parametrize(
5     'num, expected',
6     [
7         (1, 1),
8         (2, 3),
9         (3, 6),
10        (4, 10),
11        (5, 16),
12    ]
13 )
14 def test_sumNum(num, expected):
15     assert sumNum(num) == expected
```

```

(1, 1),
(3, 6),
(4, 10),
(5, 16),
]
)
def test_sumNum(num, expected):
> assert sumNum(num) == expected
E       assert 15 == 16
E       + where 15 = sumNum(5)

test\test_ej11_def2.py:15: AssertionError
===== short test summary info =====
FAILED test/test_ej11_def2.py::test_sumNum[5-16] - assert 15 == 16
===== 1 failed, 4 passed in 0.10s =====
PS C:\Users\UsuarioT\prog1>
```