

```
cas: CAUsers\jose\examen-1-pruebas-software\casosTest\test_vocales.py • Modified
You, 22 hours ago | 1 author (You)
1 import unittest #-----> para ejecutar pruebas unitarias
2 from fibonacci import fibonacci #-----> Importamos la función que queremos probar desde el archivo fibonacci.py
3
4 You, 22 hours ago | 1 author (You)
5 class TestFibonacci(unittest.TestCase): #-----> lo que nos permite escribir y ejecutar pruebas con unittest
6
7     def test_numeros_positivos(self): #-----> Prueba de números positivos
8         self.assertEqual(fibonacci(2, 3) == 5)
9         self.assertEqual(fibonacci(10, 20) == 30)
10        self.assertEqual(fibonacci(100, 200) == 300)
11
12     def test_numeros_negativos(self): #-----> Prueba con números negativos
13         self.assertEqual(fibonacci(-1, -1) == -2)
14         self.assertEqual(fibonacci(-5, -10) == -15)
15         self.assertEqual(fibonacci(-100, -200) == -300)
16
17     def test_mixto_positivo_negativo(self): #-----> Se verifica que la suma de positivos y negativos sea la correcta
18         self.assertEqual(fibonacci(-10, 10) == 0)
19         self.assertEqual(fibonacci(5, -3) == 2)
20         self.assertEqual(fibonacci(-7, 14) == 7)
21
22     def test_ceros(self): #-----> Verifica que la función maneja correctamente el 0 en diferentes escenarios
23         self.assertEqual(fibonacci(0, 0) == 0)
24         self.assertEqual(fibonacci(0, 5) == 5)
25         self.assertEqual(fibonacci(10, 0) == 10)
26
27     def test_numeros_decimales(self): #-----> Se verifica que la función maneje correctamente los decimales (float)
28         self.assertEqual(fibonacci(1.5, 2.5) == 4.0)
29         self.assertEqual(fibonacci(10.1, 5.2) == 15.3)
30         self.assertEqual(fibonacci(100.75, 200.25) == 301.0)
```

```
casosTest > test_fibonacci.py > TestFibonacci
28 self.assertEqual(fibonacci(10.1, 5.2) == 15.3)
29 self.assertEqual(fibonacci(100.75, 200.25) == 301.0)
30
31 def test_numeros_grandes(self): #-----> Se verifica que la función puede manejar enteros muy grandes sin errores
32     self.assertEqual(fibonacci(10**6, 10**6) == 2 * 10**6)
33     self.assertEqual(fibonacci(10**9, 10**9) == 2 * 10**9)
34     self.assertEqual(fibonacci(10**12, 10**12) == 2 * 10**12)
35
36 def test_valores_invalidos(self): #-----> Espera que la función genere un error si se ingresan strings
37     self.assertRaises(TypeError, fibonacci=="a", "b")
38     self.assertRaises(TypeError, fibonacci== [1, 2], [3, 4])
39     self.assertRaises(TypeError, fibonacci== {"x": 1}, {"y": 2})
40
41 def test_booleanos(self): #-----> True y False no son números válidos así que se espera que la función genere un TypeError
42     self.assertRaises(TypeError, fibonacci==True, False)
43     self.assertRaises(TypeError, fibonacci==True, 10)
44     self.assertRaises(TypeError, fibonacci==10, False)
45
46 def test_tipos_estructurados(self): #-----> Se espera que la función falle cuando recibe tuplas
47     self.assertRaises(TypeError, fibonacci==(1, 2), (3, 4))
48     self.assertRaises(TypeError, fibonacci==None, 5)
49     self.assertRaises(TypeError, fibonacci=="a": 1), {"b": 2})
50
51 You, 22 hours ago • pruebas
52 def test_extremos(self): #-----> maneja correctamente números enormes sin errores de desbordamiento
53     self.assertEqual(fibonacci(10**100, 10**100) == 2 * 10**100)
54     self.assertEqual(fibonacci(-10**100, -10**100) == -2 * 10**100)
55     self.assertEqual(fibonacci(0, 10**100) == 10**100)
56
```

test\_booleanos (test\_fibonacci.TestFibonacci.test\_booleanos) ... FAIL test\_ceros  
(test\_fibonacci.TestFibonacci.test\_ceros) ... ok test\_extremos  
(test\_fibonacci.TestFibonacci.test\_extremos) ... ok test\_mixto\_positivo\_negativo  
(test\_fibonacci.TestFibonacci.test\_mixto\_positivo\_negativo) ... ok test\_numeros\_decimales  
(test\_fibonacci.TestFibonacci.test\_numeros\_decimales) ... ok test\_numeros\_grandes  
(test\_fibonacci.TestFibonacci.test\_numeros\_grandes) ... ok test\_numeros\_negativos  
(test\_fibonacci.TestFibonacci.test\_numeros\_negativos) ... ok test\_numeros\_positivos  
(test\_fibonacci.TestFibonacci.test\_numeros\_positivos) ... ok test\_tipos\_estructurados  
(test\_fibonacci.TestFibonacci.test\_tipos\_estructurados) ... FAIL test\_valores\_invalidos  
(test\_fibonacci.TestFibonacci.test\_valores\_invalidos) ... FAIL test\_caracteres\_unicode\_y\_emojis  
(test\_vocales.TestContarVocales.test\_caracteres\_unicode\_y\_emojis) ... ok test\_entradas\_invalidas  
(test\_vocales.TestContarVocales.test\_entradas\_invalidas) ... FAIL test\_mayusculas\_y\_minusculas

```

(test_vocales.TestContarVocales.test_mayusculas_y_minusculas) ... FAIL
test_mezcla_vocales_y_consonantes
(test_vocales.TestContarVocales.test_mezcla_vocales_y_consonantes) ... FAIL
test_texto_con_espacios_y_tabulaciones
(test_vocales.TestContarVocales.test_texto_con_espacios_y_tabulaciones) ... ok
test_texto_con_numeros_y_simbolos
(test_vocales.TestContarVocales.test_texto_con_numeros_y_simbolos) ... FAIL
test_texto_con_vocales_normales
(test_vocales.TestContarVocales.test_texto_con_vocales_normales) ... FAIL
test_texto_vacio_y_sin_vocales (test_vocales.TestContarVocales.test_texto_vacio_y_sin_vocales)
... ok test_textos_largos_y_repeticiones
(test_vocales.TestContarVocales.test_textos_largos_y_repeticiones) ... ok
test_vocales_con_tildes_y_caracteres_especiales
(test_vocales.TestContarVocales.test_vocales_con_tildes_y_caracteres_especiales) ... FAIL
===== FAIL:
test_booleanos (test_fibonacci.TestFibonacci.test_booleanos) -----
----- Traceback (most recent call last): File "C:\Users\josel\examen-1-pruebas-
software\casosTest\test_fibonacci.py", line 42, in test_booleanos self.assertRaises(TypeError,
fibonacci, True, False) AssertionError: TypeError not raised by fibonacci
===== FAIL:
test_tipos_estructurados (test_fibonacci.TestFibonacci.test_tipos_estructurados) -----
----- Traceback (most recent call last): File
"C:\Users\josel\examen-1-pruebas-software\casosTest\test_fibonacci.py", line 47, in
test_tipos_estructurados self.assertRaises(TypeError, fibonacci, (1, 2), (3, 4)) AssertionError:
TypeError not raised by fibonacci
===== FAIL:
test_valores_invalidos (test_fibonacci.TestFibonacci.test_valores_invalidos) -----
----- Traceback (most recent call last): File "C:\Users\josel\examen-1-
pruebas-software\casosTest\test_fibonacci.py", line 37, in test_valores_invalidos
self.assertRaises(TypeError, fibonacci, "a", "b") AssertionError: TypeError not raised by fibonacci
===== FAIL:
test_entradas_invalidas (test_vocales.TestContarVocales.test_entradas_invalidas) -----
----- Traceback (most recent call last): File
"C:\Users\josel\examen-1-pruebas-software\casosTest\test_vocales.py", line 47, in
test_entradas_invalidas self.assertEqual(contar_vocales(str(None)), 0) AssertionError: 2 != 0
===== FAIL:
test_mayusculas_y_minusculas (test_vocales.TestContarVocales.test_mayusculas_y_minusculas) -
----- Traceback (most recent call last): File
"C:\Users\josel\examen-1-pruebas-software\casosTest\test_vocales.py", line 19, in
test_mayusculas_y_minusculas self.assertEqual(contar_vocales("PyThOn UnItEsT"), 5)
AssertionError: 4 != 5
===== FAIL:
test_mezcla_vocales_y_consonantes
(test_vocales.TestContarVocales.test_mezcla_vocales_y_consonantes) -----

```

----- Traceback (most recent call last): File "C:\Users\josel\examen-1-pruebas-software\casosTest\test\_vocales.py", line 43, in test\_mezcla\_vocales\_y\_consonantes  
self.assertEqual(contar\_vocales("Supercalifragilisticoespialidoso"), 20) AssertionError: 15 != 20

===== FAIL:

test\_texto\_con\_numeros\_y\_simbolos

(test\_vocales.TestContarVocales.test\_texto\_con\_numeros\_y\_simbolos) -----

----- Traceback (most recent call last): File "C:\Users\josel\examen-1-pruebas-software\casosTest\test\_vocales.py", line 27, in test\_texto\_con\_numeros\_y\_simbolos  
self.assertEqual(contar\_vocales("H0l@, ¿cóm0 estás?"), 6) AssertionError: 3 != 6

===== FAIL:

test\_texto\_con\_vocales\_normales

(test\_vocales.TestContarVocales.test\_texto\_con\_vocales\_normales) -----

----- Traceback (most recent call last): File "C:\Users\josel\examen-1-pruebas-software\casosTest\test\_vocales.py", line 8, in test\_texto\_con\_vocales\_normales  
self.assertEqual(contar\_vocales("Python es genial"), 6) AssertionError: 5 != 6

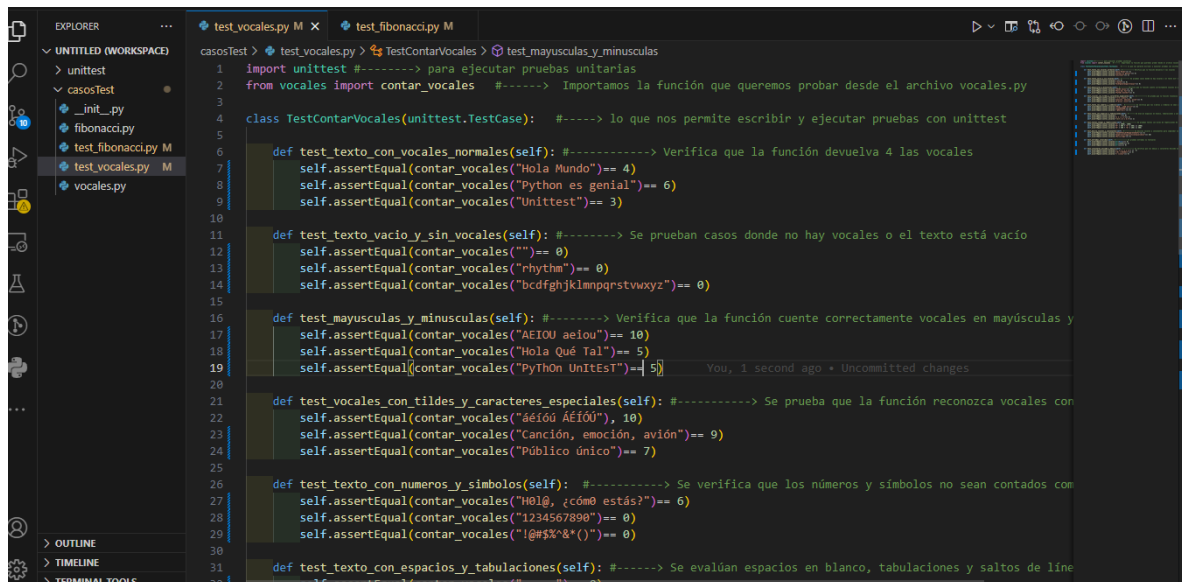
===== FAIL:

test\_vocales\_con\_tildes\_y\_caracteres\_especiales

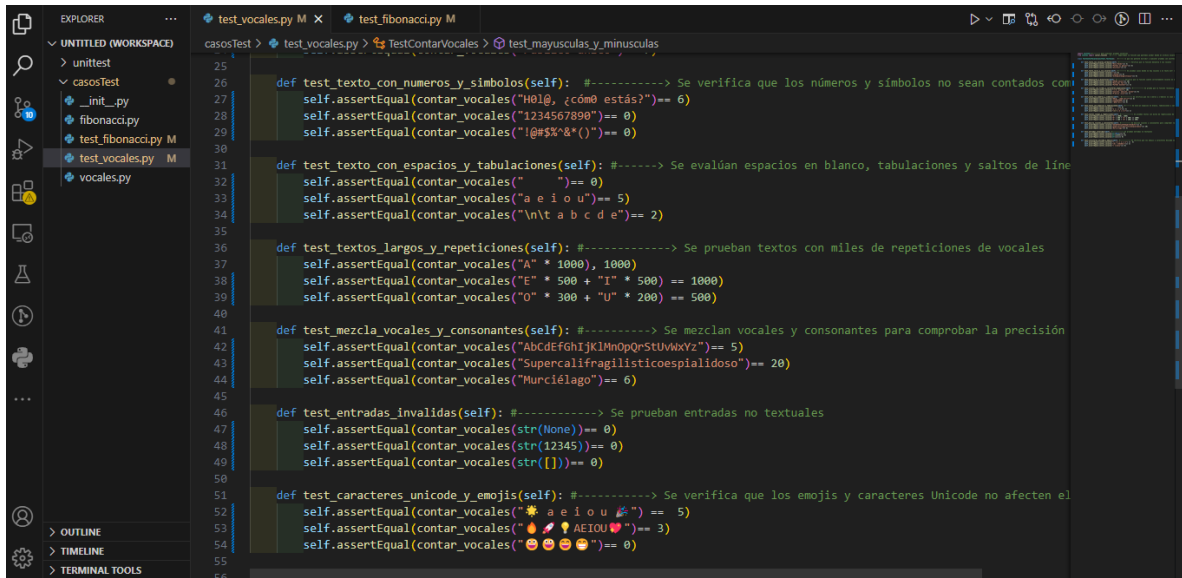
(test\_vocales.TestContarVocales.test\_vocales\_con\_tildes\_y\_caracteres\_especiales) -----

----- Traceback (most recent call last): File  
"C:\Users\josel\examen-1-pruebas-software\casosTest\test\_vocales.py", line 23, in  
test\_vocales\_con\_tildes\_y\_caracteres\_especiales self.assertEqual(contar\_vocales("Canción,  
emoción, avión"), 9) AssertionError: 10 != 9 -----

Ran 20 tests in 0.011s FAILED (failures=9)



```
casosTest > test_vocales.py > TestContarVocales > test_mayusculas_y_minusculas
1 import unittest #-----> para ejecutar pruebas unitarias
2 from vocales import contar_vocales #-----> Importamos la función que queremos probar desde el archivo vocales.py
3
4 class TestContarVocales(unittest.TestCase): #-----> lo que nos permite escribir y ejecutar pruebas con unittest
5
6     def test_texto_con_vocales_normales(self): #-----> Verifica que la función devuelva 4 las vocales
7         self.assertEqual(contar_vocales("Hola Mundo") == 4)
8         self.assertEqual(contar_vocales("Python es genial") == 6)
9         self.assertEqual(contar_vocales("Unittest") == 3)
10
11     def test_texto_vacio_y_sin_vocales(self): #-----> Se prueban casos donde no hay vocales o el texto está vacío
12         self.assertEqual(contar_vocales("") == 0)
13         self.assertEqual(contar_vocales("rhythm") == 0)
14         self.assertEqual(contar_vocales("bcd fghjklmnpqrstvwxyz") == 0)
15
16     def test_mayusculas_y_minusculas(self): #-----> Verifica que la función cuente correctamente vocales en mayúsculas y
17         self.assertEqual(contar_vocales("AEIOU aeiou") == 10)
18         self.assertEqual(contar_vocales("Hola Qué Tal") == 5)
19         self.assertEqual(contar_vocales("PyThOn UnItEst") == 5)
20
21     def test_vocales_con_tildes_y_caracteres_especiales(self): #-----> Se prueba que la función reconozca vocales con
22         self.assertEqual(contar_vocales("áéíóú ÁÉÍÓÚ"), 10)
23         self.assertEqual(contar_vocales("Canción, emoción, avión") == 9)
24         self.assertEqual(contar_vocales("Público único") == 7)
25
26     def test_texto_con_numeros_y_simbolos(self): #-----> Se verifica que los números y símbolos no sean contados como
27         self.assertEqual(contar_vocales("H0l@, ¿cóm0 estás?") == 6)
28         self.assertEqual(contar_vocales("1234567890") == 0)
29         self.assertEqual(contar_vocales("!@#$%^&*()") == 0)
30
31     def test_texto_con_espacios_y_tabulaciones(self): #-----> Se evalúan espacios en blanco, tabulaciones y saltos de línea
32         self.assertEqual(contar_vocales(" ") == 0)
```



The image shows a Visual Studio Code editor window with a Python project. The Explorer sidebar on the left shows a file tree with the following structure:

- UNTITLED (WORKSPACE)
  - unittest
    - casosTest
      - init.py
      - fibonacci.py
      - test\_fibonacci.py M
      - test\_vocales.py M
      - vocales.py

The main editor displays the content of `test_vocales.py`, which contains several unit tests for a `ContarVocales` class. The tests are as follows:

```
25
26 def test_texto_con_numeros_y_simbolos(self): #-----> Se verifica que los números y símbolos no sean contados como
27     self.assertEqual(contar_vocales("H0l0, ¿cómo estás?")== 6)
28     self.assertEqual(contar_vocales("1234567890")== 0)
29     self.assertEqual(contar_vocales("!@#%&*()")== 0)
30
31 def test_texto_con_espacios_y_tabulaciones(self): #-----> Se evalúan espacios en blanco, tabulaciones y saltos de línea
32     self.assertEqual(contar_vocales(" ")== 0)
33     self.assertEqual(contar_vocales("a e i o u")== 5)
34     self.assertEqual(contar_vocales("\n\t a b c d e")== 2)
35
36 def test_textos_largos_y_repeticiones(self): #-----> Se prueban textos con miles de repeticiones de vocales
37     self.assertEqual(contar_vocales("A" * 1000), 1000)
38     self.assertEqual(contar_vocales("E" * 500 + "I" * 500) == 1000)
39     self.assertEqual(contar_vocales("O" * 300 + "U" * 200) == 500)
40
41 def testmezcla_vocales_y_consonantes(self): #-----> Se mezclan vocales y consonantes para comprobar la precisión
42     self.assertEqual(contar_vocales("AbCdEfGhIjKlMnOpQrStUvWxYz")== 5)
43     self.assertEqual(contar_vocales("Supercalifragilisticoespialidoso")== 20)
44     self.assertEqual(contar_vocales("Murciélagos")== 6)
45
46 def test_entradas_invalidas(self): #-----> Se prueban entradas no textuales
47     self.assertEqual(contar_vocales(str(None))== 0)
48     self.assertEqual(contar_vocales(str(12345))== 0)
49     self.assertEqual(contar_vocales(str([]))== 0)
50
51 def test_caracteres_unicode_y_emojis(self): #-----> Se verifica que los emojis y caracteres Unicode no afecten el
52     self.assertEqual(contar_vocales(" a e i o u 🍌 ") == 5)
53     self.assertEqual(contar_vocales("🍌 🍌 AEIOU 🍌 ") == 3)
54     self.assertEqual(contar_vocales("🍌 🍌 🍌 🍌 ") == 0)
55
56
```