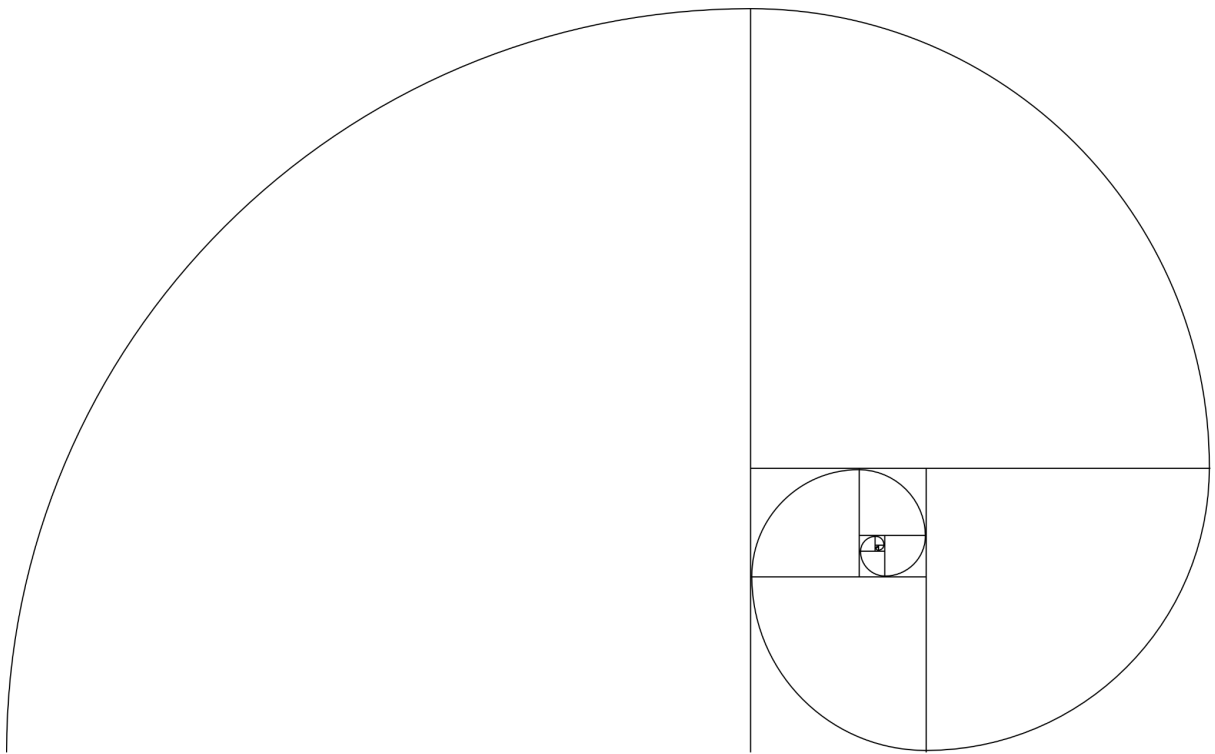


Entrega UD1 - PPS

Secuencia de Fibonacci en Python

Xabier Rodriguez Blanco



Índice

Índice	2
1. Resolución	3
2. Probas	5
3. Que tipo de proba foi realizada?	7

1. Resolución

Abro o meu editor de código, neste caso “Visual Studio Code” e comezo coa resolución de cada apartado. O primer paso foi crear a base do programa “a sucesión de Fibonacci” a partir da cal fago probas e sigo creando o programa, como engadir o módulo de Python “unittest”, o cal uso para comprobar se o resultado que me da o programa é correcto, ata chegar a etapa final, onde podo comprobar que todo o código funciona según o esperado.

O resultado é o seguinte:

```
# Programa fibo.py que xenera a sucesión de Fibonacci e posteriormente
comproba si o valor xerado é igual ao valor
# pertencente á serie.
import unittest

# Función chamada "fibonacci" que xenera a sucesión de fibonacci co número de
veces dado
def fibonacci(numVeces):
    secuenciaFibo = [0,1] # Inicio da secuencia cos dous primeiros números da
sucesión
    for x in range(2,numVeces): # Comezo dende o tercer número en diante
        secuenciaFibo.append(secuenciaFibo[-1] + secuenciaFibo[-2]) # Engado o
seguinte número sumando os dous últimos
    return secuenciaFibo # Devolvo a secuencia ata o número dado

# Función adicional para calcular un número específico na secuencia de Fibonacci
def calcularPosicionFibo(posicion):
    if posicion == 0:
        return 0
    elif posicion == 1:
        return 1
    else:
        num1,num2 = 0,1
        for _ in range(2,posicion + 1):
            num1,num2 = num2,num1 + num2
        return num2

# Clase para as probas unitarias
class testFibo(unittest.TestCase):

    # Método para probar si o quinto número da serie é igual a 3
    def testSecuenciaFibo(self):
        quintoNumero = fibonacci(5)[-1] # Obteño o quinto número da serie
        self.assertEqual(quintoNumero,3,"O quinto número da serie de Fibonacci
tendría que ser 3")

    # Método para probar o cálculo de un número específico na secuencia
```

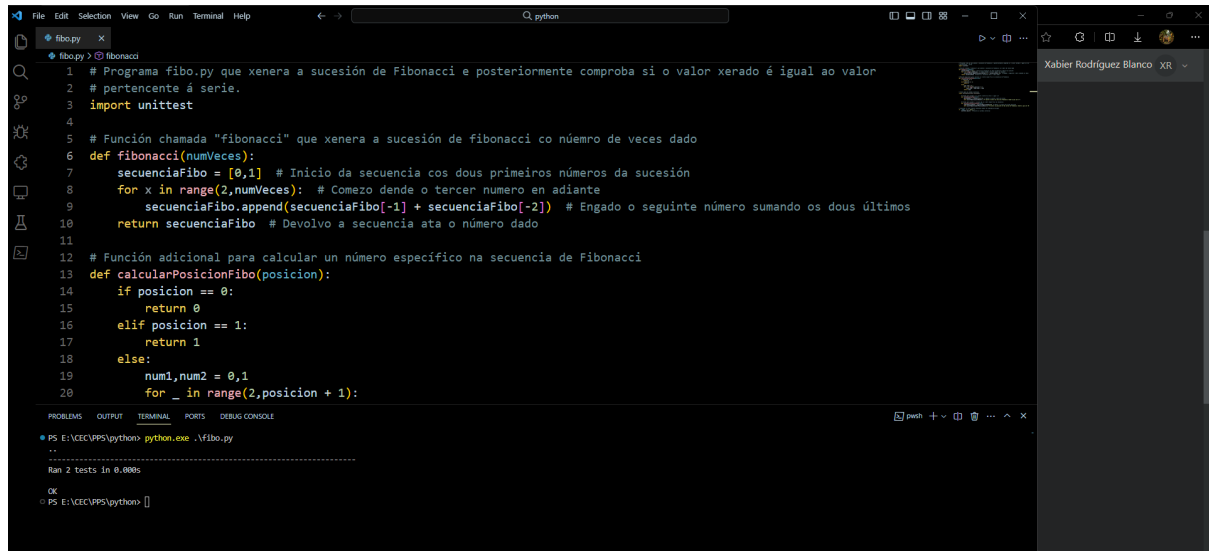
```
def testCalcularPosicionFibo(self):
    numeroPosicionDiez = calcularPosicionFibo(10) # Obteño o número da
    décima posición
    self.assertEqual(numeroPosicionDiez,55,"O número na posición 10 da serie
    de Fibonacci tendría que ser 55")

# Comprobo si este arquivo executase igual ao programa principal
if __name__ == "__main__":
    unittest.main() # Executo as probas unitarias
```

*Todo este código encóntrase dentro do arquivo “fibo.py”, sangrado e comentado correctamente.

2. Probas

Para realizar as probas executo o programa de diferentes maneiras para comprobar se funciona según o esperado.

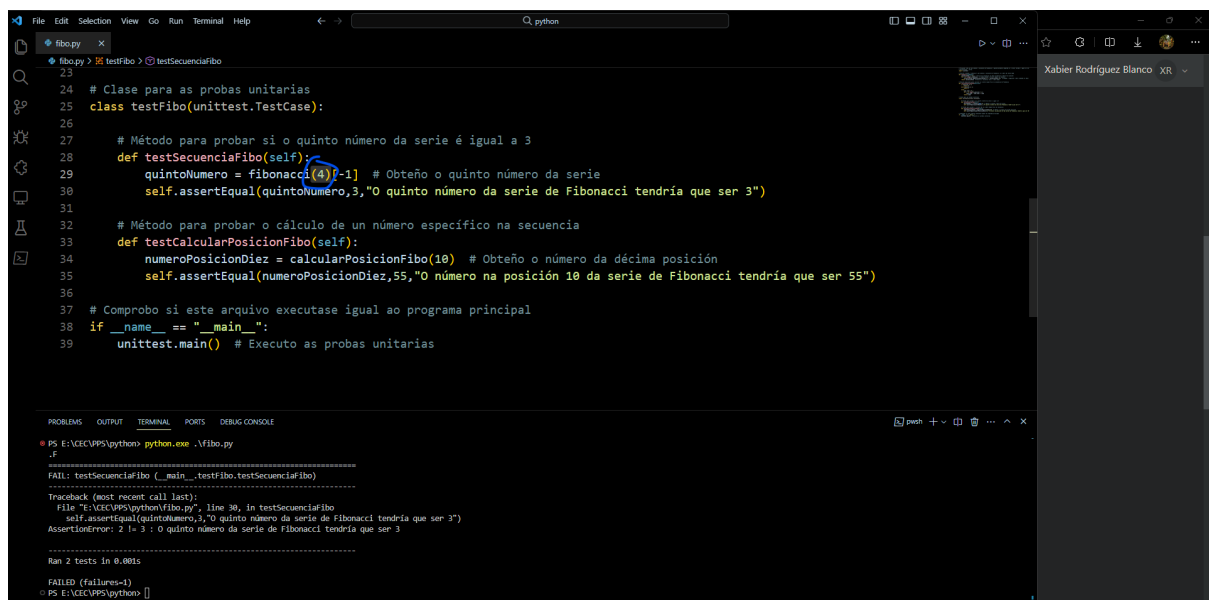


```
1 # Programa fibo.py que xenera a sucesión de Fibonacci e posteriormente comproba si o valor xerado é igual ao valor
2 # pertencente á serie.
3 import unittest
4
5 # Función chamada "fibonacci" que xenera a sucesión de fibonacci co número de veces dado
6 def fibonacci(numVeces):
7     secuenciaFibo = [0,1] # Inicio da secuencia cos dous primeiros números da sucesión
8     for x in range(2,numVeces): # Comezo dende o tercer número en diante
9         secuenciaFibo.append(secuenciaFibo[-1] + secuenciaFibo[-2]) # Engado o seguinte número sumando os dous últimos
10    return secuenciaFibo # Devolvo a secuencia ata o número dado
11
12 # Función adicional para calcular un número específico na secuencia de Fibonacci
13 def calcularPosicionFibo(posicion):
14     if posicion == 0:
15         return 0
16     elif posicion == 1:
17         return 1
18     else:
19         num1,num2 = 0,1
20         for _ in range(2,posicion + 1):
```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

```
PS E:\CEC\PPS\python> python.exe .\fibo.py
..
Ran 2 tests in 0.000s
OK
PS E:\CEC\PPS\python>
```

Nesta primeira execución vexo que o programa funciona correctamente e que o código coma a función unittest funcionan según o esperado xa que non devolven ningún error.



```
23
24 # Clase para as probas unitarias
25 class testFibo(unittest.TestCase):
26
27     # Método para probar si o quinto número da serie é igual a 3
28     def testSecuenciaFibo(self):
29         quintoNumero = fibonacci(4)[-1] # Obteño o quinto número da serie
30         self.assertEqual(quintoNumero,3,"O quinto número da serie de Fibonacci tendría que ser 3")
31
32     # Método para probar o cálculo de un número específico na secuencia
33     def testCalcularPosicionFibo(self):
34         numeroPosicionDiez = calcularPosicionFibo(10) # Obteño o número da décima posición
35         self.assertEqual(numeroPosicionDiez,55,"O número na posición 10 da serie de Fibonacci tendría que ser 55")
36
37 # Comprobo si este arquivo executase igual ao programa principal
38 if __name__ == "__main__":
39     unittest.main() # Executo as probas unitarias
```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

```
PS E:\CEC\PPS\python> python.exe .\fibo.py
.F
FAIL: testSecuenciaFibo (__main__.testFibo.testSecuenciaFibo)
Traceback (most recent call last):
  File "E:\CEC\PPS\python\fibo.py", line 30, in testSecuenciaFibo
    self.assertEqual(quintoNumero,3,"O quinto número da serie de Fibonacci tendría que ser 3")
AssertionError: 2 != 3 : O quinto número da serie de Fibonacci tendría que ser 3
Ran 2 tests in 0.001s
FAILED (failures=1)
PS E:\CEC\PPS\python>
```

Para esta segunda execución modifíco un dos parámetros e vexo que o programa responde cun error, mostrande que o número que lle chega non é igual ao que eu lle proporciono dentro das probas de unittest.

The screenshot shows a VS Code editor with a Python file named `fib.py`. The code defines a `testFibo` class with two methods: `testSecuenciaFibo` and `testCalcularPosicionFibo`. The `testCalcularPosicionFibo` method is highlighted with a blue circle around the argument `11` in the `calcularPosicionFibo` function call. The terminal output shows a failure in the `testCalcularPosicionFibo` test, with the error message: `AssertionError: 89 != 55 : 0 número na posición 10 da serie de Fibonacci tendría que ser 55`. The terminal also shows the command `python.exe .\fib.py` and the output `FF`.

Nesta terceira execución modifíco un parámetro diferente para ver que as distintas probas de unittest están funcionando según o esperado e os diferentes resultados non se solapan.

The screenshot shows the same VS Code editor with the `fib.py` file. The `testCalcularPosicionFibo` method is highlighted with a blue circle around the argument `9` in the `calcularPosicionFibo` function call. The terminal output shows two failures: `AssertionError: 34 != 55 : 0 número na posición 10 da serie de Fibonacci tendría que ser 55` and `AssertionError: 1 != 3 : 0 quinto número da serie de Fibonacci tendría que ser 3`. The terminal also shows the command `python.exe .\fib.py` and the output `FF`.

Nesta última proba modifíco ambos valores e observo que o módulo unittest funciona correctamente mostrándome por texto que o resultado non é o esperado, o que significa que está no correcto.

Despois destas probas realizadas podo certificar que o código funciona según o pedido e que todas as probas foron exitosas.

3. Que tipo de proba foi realizada?

A proba realizada clasifícase como unha proba unitaria. As probas unitarias verifican un trozo de código en específico, neste caso a función de Fibonacci, para comprobar que o resultado é o esperado nun caso concreto.

Neste caso foron realizadas dúas comprobacións. A primeira foi verificar se o quinto valor da sucesión coincidía co quinto valor que proporcionaba a primeira función. A segunda foi poder verificar calquer valor da sucesión.

Estas probas son necesarias xa que permiten ir comprobando trozos e funcións do código e así poder vendo se está dando o resultado esperado. Pode ser que en código pequenos non sexa necesario, pero en grandes o máis seguro sexa que sí.