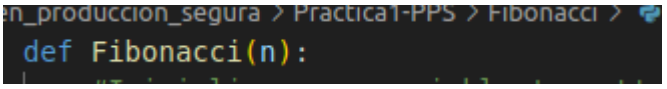
Práctica 1
-
Posta en producción segura

Pablo Sánchez Otero

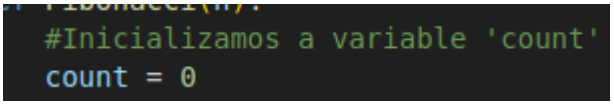
Índice

1. Sucesión de fibonacci

Para a realización de unha fución que calcule a sucesión de Fibonacci dado un número creamos o script fibo.py, no interior do script declaramos unha función chamada “Fibonacci(n)” que toma un paramétro ‘n’ que determinará o número de elementos da sucesión.

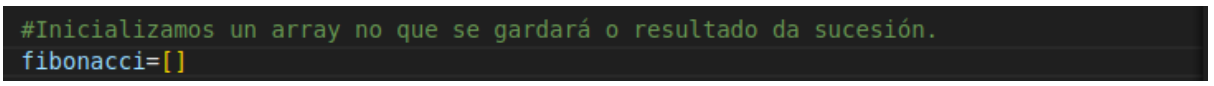


A cotinuación declaramos unha variable ‘count’ que contará as veces que se executa un bucle while aumentando o seu valor en 1 ata que sexa maior ou igual ao valor de ‘n’.



No paso posterior inicializamos dúas variables ‘a’ e ‘b’ que serán os dous primeiros elementos da sucesión de fibonacci que serán 0 e 1.

Para continuar declaramos unha lista chamada ‘fibonacci’ no que se irán gardando en orde os diferentes elementos da sucesión de fibonacci, está lista será o resultado que devolva a nosa función.



Antes de declarar o bucle do que falamos anteriormente declaramos un condicional if a modo de excepción, que se encargará de distinguir si o ‘n’ e menor ou igual a 0. Si este é o caso a función devolverá unha cadena de texto no que se indica que ‘n’ debe ser maior que 0 para que a sucesión fibonacci devolva máis de 0 elementos, en caso contrario comezará a executarse un bucle tantas veces como ‘n’ (mentres ‘count’ sexa menor a ‘n’) realizando as seguintes acción en cada iteración; a variable ‘a’ gardarase dentro da lista ‘fibonacci’, unha variabe ‘result’ gardará o resultado da suma das variables ‘a’ + ‘b’ de forma que na primeira iteración será igual a 1, a variable ‘a’ muta o seu valor ao valor de ‘b’, b muta o seu valor ao valor de result de forma que a vaia gardando e añadindo á lista os valores da suma dos dous anteriores dende 0 ata o elemento dado e para rematar aumentamoso valor de

‘count’ en 1 e fora do bucle, unha vez terminou todas as iteracións devolvems a lista ‘fibonacci’ no que se foron gardando todos os valores de ‘a’ completando a sucesión.

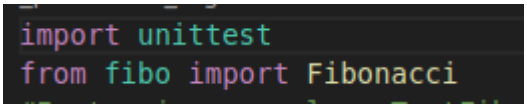
```
    if(n<=0):
        return("Ingrese un numero mayor a 0")
    else:
    #Mentres a conta sea inferior ao número introducido volvera a executarse a suma máis a reasignación das variables 'a=b' e 'b=result(a+b)'.
        while count<n:
            fibonacci.append(a)
            result=a+b
            a=b
            b=result
            #Incrementamos a conta en 1.
            count+=1
        return(fibonacci)
```

Así quedaría o script competo:

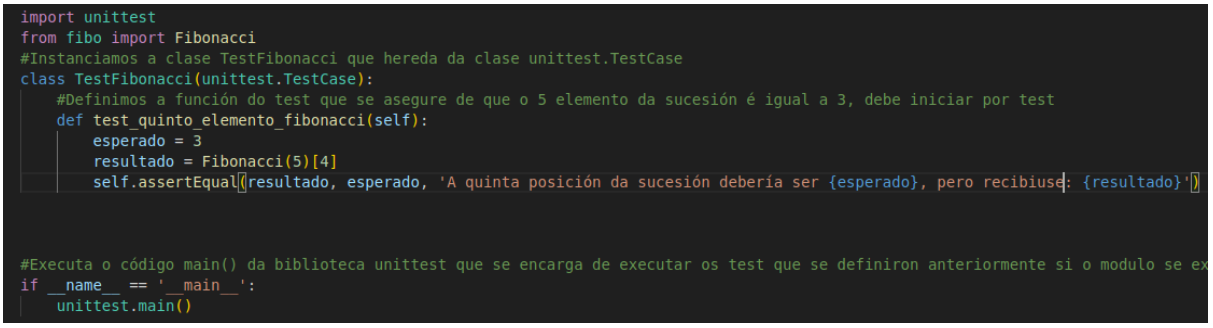
```
def Fibonacci(n):
    #Inicializamos a variable 'count' para contar as veces que se rep
    count = 0
    #Se inicializan as variables 'a' e 'b' cos 2 primeiros valores da
    a = 0
    b = 1
    #Inicializamos un array no que se gardará o resultado da sucesión
    fibonacci=[]
    #Si o número introducido é negativo non se pode realizar a sucesi
    if(n<=0):
        return("Ingrese un numero mayor a 0")
    else:
    #Mentres a conta sea inferior ao número introducido volvera a exe
        while count<n:
            fibonacci.append(a)
            result=a+b
            a=b
            b=result
            #Incrementamos a conta en 1.
            count+=1
        return(fibonacci)


num=int(input("Introduce un numero: "))
print(Fibonacci(num))
```

2. Test unitario de la función Fibonacci

Comenzamos creando o script test.py e importando a biblioteca de ‘unittest’ que importa funciones e clases útiles para a realización de tests de código, tamén importamos o módulo a función ‘Fibonacci’ desde o módulo creado anteriormente ‘fibo’.



Continuamos creando unha clase cun nombre descriptivo ‘TestFibonacci’ que herda da clase ‘TestCase’ da biblioteca ‘unittest’. Dentro de esta clase declaramos unha función que debe comezar pola palabra ‘test’. y se le pasa como parámetro la palabra reservada ‘self’ que llama a una instancia de la clase en la que se definió, en este caso TestFibonacci. Para asegurarnos de que o valor esperado (3) é igual a recibido utilizaremos a función assertEqual da clase TestFibonacci que herda da clase TestCase. Neste caso o comprobaremos que o quinto elemtno (Fibonacci(5)[4]) é igual a 3:



O mensaxe que lle pasamos como 3º parámetro á clase assertEqual trátase dun mensaxe de error que se mostrará se o valor recibido non coincide co esperado.

Seguindo o mismo principio añadense dous tests máis que se executarán comprobando diferentes elementos da sucesión fibonacci, asegurándose de que está ben programada.

```
Fibonacci > test.py > ...
1  import unittest
2  from fibo import Fibonacci
3  #Instanciamos a clase TestFibonacci que hereda da clase unittest.TestCase
4  class TestFibonacci(unittest.TestCase):
5      #Definimos a función do test que se asegure de que o 5 elemento da sucesión é igual a 3.
6
7          def test_quinto_elemento_fibonacci(self):
8              esperado = 3
9              resultado = Fibonacci(5)[4]
10             self.assertEqual(resultado, esperado, f'A quinta posición da sucesión debería ser {esperado}, pero recibiuse: {resultado}')
11
12         def test_segundo_elemento_fibonacci(self):
13             esperado = 0
14             resultado = Fibonacci(1)[0]
15             self.assertEqual(resultado, esperado, f'A primeira posición da sucesión debería ser {esperado}, pero recibiuse: {resultado}')
16
17         def test_decimo_elemento_fibonacci(self):
18             esperado = 34
19             resultado = Fibonacci(10)[9]
20             self.assertEqual(resultado, esperado, f'A decima posición da sucesión debería ser {esperado}, pero recibiuse: {resultado}')
21
22
23
24 #Executa o código main() da biblioteca unittest que se encarga de executar os test que se definiron anteriormente si o modulo se executa direc
25 if __name__ == '__main__':
26     unittest.main()
27
```

test.py | fibo.py | .gitignore

Fibonacci > test.py > TestFibonacci > test_segundo_elemento_fibonacci

```python
import unittest
from fibo import Fibonacci
from fibo import num
#Instanciamos a clase TestFibonacci que hereda da clase unittest.Test
class TestFibonacci(unittest.TestCase):
    #Definimos a función do test que se asegure de que o 5 elemento d
    if num>=5:
        def test_quinto_elemento_fibonacci(self):
            esperado = 3
            resultado = Fibonacci(5)[4]
            self.assertEqual(resultado, esperado, f'A quinta posición
    if num<5 and num>1:
        def test_segundo_elemento_fibonacci(self):
            esperado = 1
            resultado = Fibonacci(4)[1]
            self.assertEqual(resultado, esperado, f'A primeira posici
    if num>=10:
        def test_decimo_elemento_fibonacci(self):
            esperado = 34
            resultado = Fibonacci(10)[9]
            self.assertEqual(resultado, esperado, f'A decima posición



#Executa o código main() da biblioteca unittest que se encarga de exe
if __name__ == '__main__':
    unittest.main()

```