PRÁCTICA 1 PPS

EJERCICIO FIBONACCI CON PYTHON

Bernabé Villaverde Rodríguez

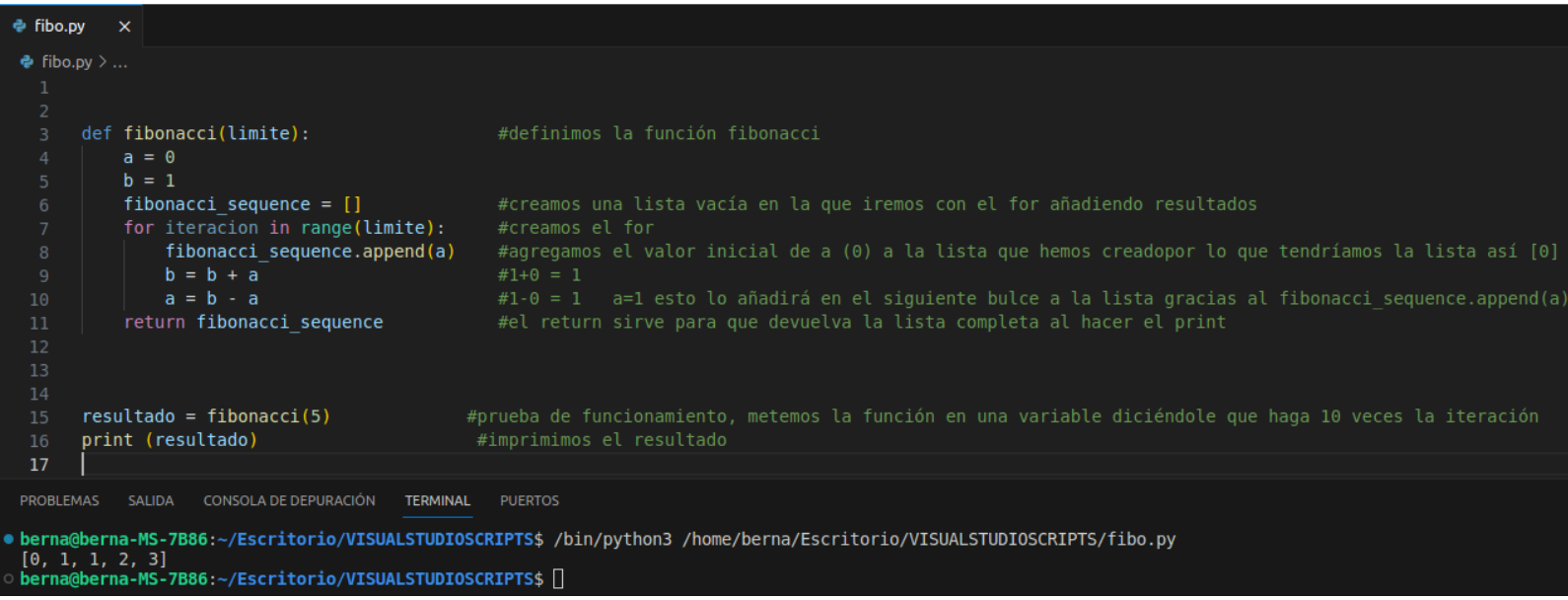
FIBO.PY

Lo primero que haremos será crear el script con la secuencia de fibonacci, esta secuencia es una serie de números en la que cada número es la suma de los dos anteriores, comenzando con 0 y 1. Es decir 0,1,1,2,3,5,8,13,21,34,....

En el ejercicio nos piden que creemos una función para nuestro código por lo que tendríamos algo así:

```
def fibonacci(limite):                  #definimos la función fibonacci
    a = 0
    b = 1
    fibonacci_sequence = []             #creamos una lista vacía en la que iremos con el for añadiendo resultados
    for iteracion in range(limite):     #creamos el for
        fibonacci_sequence.append(a)    #agregamos el valor inicial de a (0) a la lista que hemos creadopor lo que tendríamos la lista así [0]
        b = b + a                       #1+0 = 1
        a = b - a                       #1-0 = 1   a=1 esto lo añadirá en el siguiente bulce a la lista gracias al fibonacci_sequence.append(a)
    return fibonacci_sequence           #el return sirve para que devuelva la lista completa al hacer el print
```

Como podemos observar aquí tenemos la función fibonacci creada y explicada.
Si necesitáramos probarla solo tendríamos que realizar dos cambios



Como se observa guardamos la función indicando que llame a la función fibonacci con el argumento 5. Esto significa que la función generará los primeros 5 números de la secuencia de Fibonacci.

La explicación línea por línea detallada (aunque se vea en la imagen sería):

#definimos la función fibonacci

```
def fibonacci(limite):
```

Le damos a “a” un valor inicial de 0 y a “b” un valor inicial de 1

```
a = 0
b = 1
```

Creamos una lista vacía en la que iremos con el for añadiendo resultados

```
fibonacci_sequence = []
```

Creamos el for

```
for iteracion in range(limite):
```

Agregamos el valor inicial de a (0) a la lista que hemos creadopor lo que tendríamos la lista así [0]

```
fibonacci_sequence.append(a)
```

1+0 = 1

```
b = b + a
```

1-0 = 1 a=1 esto lo añadirá en el siguiente bulce a la lista gracias al fibonacci_sequence.append(a)

```
a = b - a
```

El return sirve para que devuelva la lista completa al hacer el print

```
return fibonacci_sequence
```

Aquí tendríamos ya la mitad del ejercicio hecho ahora falta crear el script test_fibo.py

TEST_FIBO.PY

Lo crearemos ahora será un programa principal donde definiremos una clase llamada Test, donde probaremos nuestro software que hicimos en el apartado anterior

```
# test_fibo.py
import unittest                                 #Importamos la librería unittest
from fibo import fibonacci                      #Importamos la función que creamos

class Test(unittest.TestCase):                  #definimos la clase llamada Test que hereda las funcionalidades de la librería unittest

    def test_quinto_numero_fibonacci(self):     #nombro la función con el prefijo "test_" para que el descubrimiento automático de pruebas de unittest funcione correcta
        resultado = fibonacci(5)                #llama a la funcion indicando que haga 5 veces la iteración lo que hará que la lista tenga 5 números [0,1,1,2,3]
        self.assertEqual(resultado, [0, 1, 1, 2, 3])   #con assertEqual (método que viene de la librería unittest) verofica si el resultado de la función al darle el valor 5 s

if __name__ == '__main__':                      #Ejecutamos la función quinto_numero_fibonacci y con el unittest
    unittest.main()                             #nos verifica si se genera correctamente la secuencia hasta el quinto número devolviendo un OK
```

El cual si ejecutamos imprime lo siguiente:

```
18  # test_fibo.py
19  import unittest                                  #Importamos la librería unittest
20  from pruebafibo import fibonacci                       #Importamos la función que creamos
21
22  class Test(unittest.TestCase):                   #definimos la clase llamada Test que hereda las fun
23
24      def test_quinto_numero_fibonacci(self):      #nombro la función con el prefijo "test_" para que
25          resultado = fibonacci(5)                 #llama a la funcion indicando que haga 5 veces la i
26          self.assertEqual(resultado, [0, 1, 1, 2, 3])    #con assertEqual (método que viene de la librería u
27
28  if __name__ == '__main__':                       #Ejecutamos la función quinto_numero_fibonacci y co
29      unittest.main()                              #nos verifica si se genera correctamente la secuenc
30
31
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL   PUERTOS
berna@berna-MS-7B86:~/Escritorio/VISUALSTUDIOSCRIPTS$ /bin/python3 /home/berna/Escritorio/VISUALSTUDIOSCRIPTS/fibo.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

Ahora explicaremos qué hace cada línea:

Importamos la librería unittest:

```
import unittest
```

Importamos la función que creamos en el anterior apartado

```
from fibo import fibonacci
```

Definimos la clase llamada Test que hereda las funcionalidades de la librería unittest

```
class Test(unittest.TestCase):
```

Nombro la función con el prefijo "test_" para que el descubrimiento automático de pruebas de unittest funcione.

```
def test_quinto_numero_fibonacci(self):
```

Llama a la funcion indicando que haga 5 veces la iteración lo que hará que la lista tenga 5 números [0,1,1,2,3]

```
resultado = fibonacci(5)
```

assertEqual (método proveniente de la librería unittest) verifica si el resultado de la función al darle

```
self.assertEqual(resultado, [0, 1, 1, 2, 3])
```

Ejecutamos la función quinto_numero_fibonacci y con el unittest
nos verifica si se genera correctamente la secuencia hasta el quinto número devolviendo un OK

```
if __name__ == '__main__':
    unittest.main()
```

Cuando ejecutas este script, unittest ejecutará la prueba test_quinto_numero_fibonacci y te dará un resultado indicando si la prueba ha pasado o no.

Si todo está bien, deberías ver un mensaje "OK". Si hay algún problema, unittest te informará sobre la falla (FAILED).

TIPO DE PRUEBA

El tipo de prueba realizada es un Test de aceptación.

El Test de aceptación, se centra en verificar que una parte de un programa funcione según lo esperado, aquí por ejemplo, que la función fibonacci devuelva el quinto número correcto de la secuencia de Fibonacci.

BIBLIOGRAFÍA

https://www.mclibre.org/consultar/python/ejercicios/ej-for-1.html#
https://recursospython.com/guias-y-manuales/unit-testing-doc-testing/
https://chat.openai.com/
https://www.w3schools.com/python/