

# **PUESTA EN PRODUCCIÓN SEGURA**

## **Tarea 1**

# Indice

	Paxina
1 - Enunciado .....	1
2 - Creación del programa .....	2
3 - Test del programa .....	3
4 - Referencias .....	4

# 1 - Enunciado

¿Qué se pide?

Antes de nada reflejaremos la sucesión de Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34... Tienes mas información sobre esta interesante sucesión y su origen aquí

1: Crea un script que genere la secuencia de Fibonacci

Puedes usar cualquier lenguaje con el que estés familiarizado, pero te recomiendo por facilidad y por el gran acceso a librerías de evaluación de software el lenguaje Python. Si aún no has programado en Python, su inmersión te será sencilla y amigable.

Llamaremos a este script fibo.py (en el caso en que estemos programando en python)

Deberías de crear un programa con su estructura completa. Dentro de este programa crearemos una función llamada fibonacci.

2: Creación del programa principal

2.1. Crearemos un programa principal donde definiremos una clase llamada Test, donde probaremos nuestro software.

2.2. Importaremos la librería de testeo de software, en este caso unittest.

2.3. Crearemos nuestra clase (del tipo unittest.TestCase)

2.4. Dentro de esta clase definiremos una función (podemos llamarla como consideremos, pero es recomendable un nombre ilustrativo)

Dentro de la función reflejaremos el tipo de testeo que vamos a realizar. En este caso nuestro objetivo es verificar si la posición X de la sucesión de Fibonacci coincide con el resultado esperado, es decir, si coincide con la posición X que devuelve nuestro programa. Para ello, comprobamos que el quinto número de la sucesión, que es 3, coincide con el quinto número que devuelve nuestra secuencia. Para hacer esta comprobación usaremos el tipo de comprobación `assertEqual` que comprueba si dos valores son iguales

Mediante esta función comprobaremos si la posición quinta de nuestra función coincide con el valor esperado (el valor esperado es la quinta posición de la sucesión que es 3)

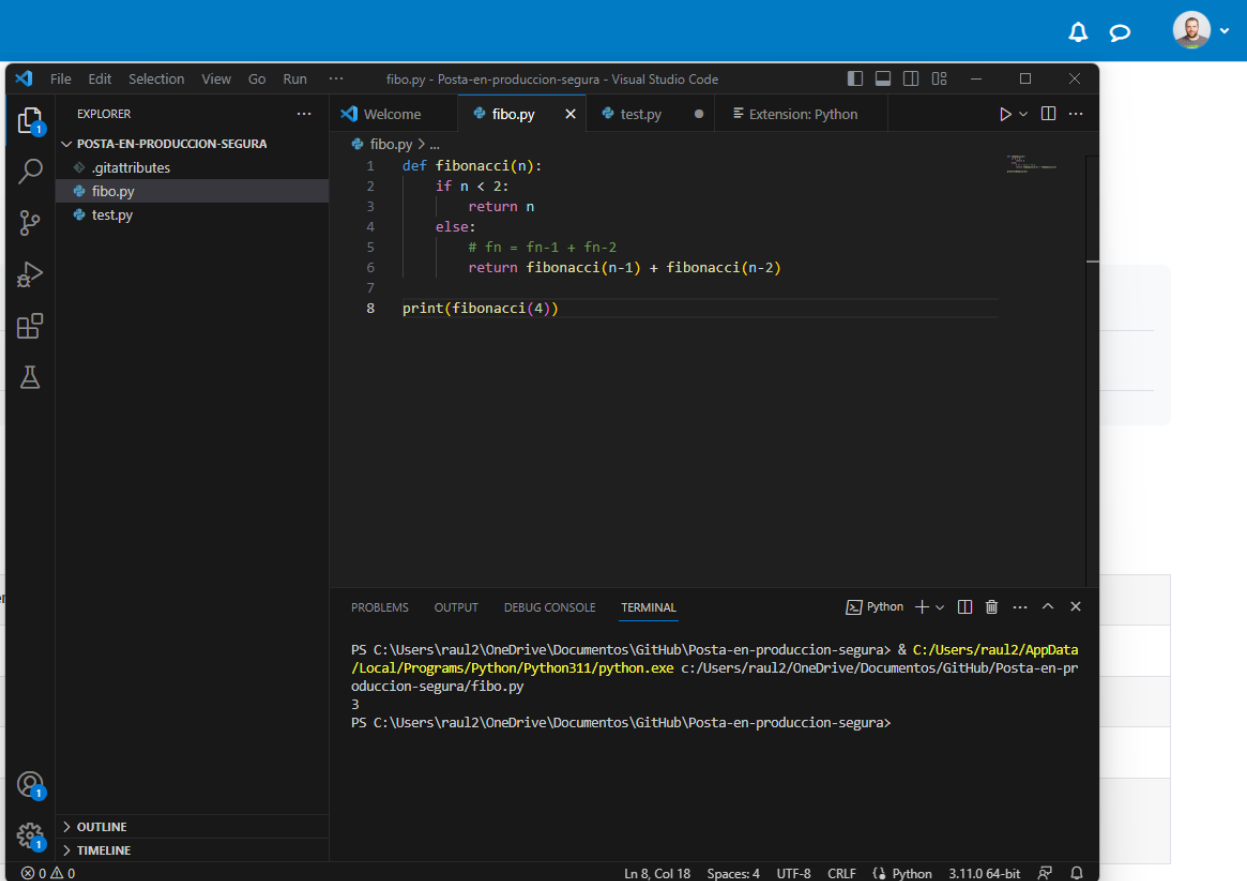
3: Verificación de software y pregunta final

Ejecutaremos el programa final y verificaremos si realmente nuestro programa que calcula la sucesión de Fibonacci (fibo.py) se comporta como esperamos. Si el programa que comprueba el código detecta un error, nos reflejará que dato está esperando y que ha recibido. ¿Qué tipo de prueba hemos realizado?

## 2 - Creación del programa

Para la creación del programa usamos Python y como IDE Visual Studio Code.

Realizamos el script figo.py y probamos su funcionamiento:



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a folder named 'POSTA-EN-PRODUCCION-SEGURA' containing files '.gitattributes', 'fibo.py', and 'test.py'. The main editor window displays the code in 'fibo.py':

```
1 def fibonacci(n):
2     if n < 2:
3         return n
4     else:
5         # fn = fn-1 + fn-2
6         return fibonacci(n-1) + fibonacci(n-2)
7
8 print(fibonacci(4))
```

The bottom panel shows the TERMINAL output:

```
PS C:\Users\raul2\OneDrive\Documentos\GitHub\Posta-en-produccion-segura> & C:/Users/raul2/AppData/Local/Programs/Python/Python311/python.exe c:/Users/raul2/OneDrive/Documentos/GitHub/Posta-en-produccion-segura/fibo.py
3
PS C:\Users\raul2\OneDrive\Documentos\GitHub\Posta-en-produccion-segura>
```

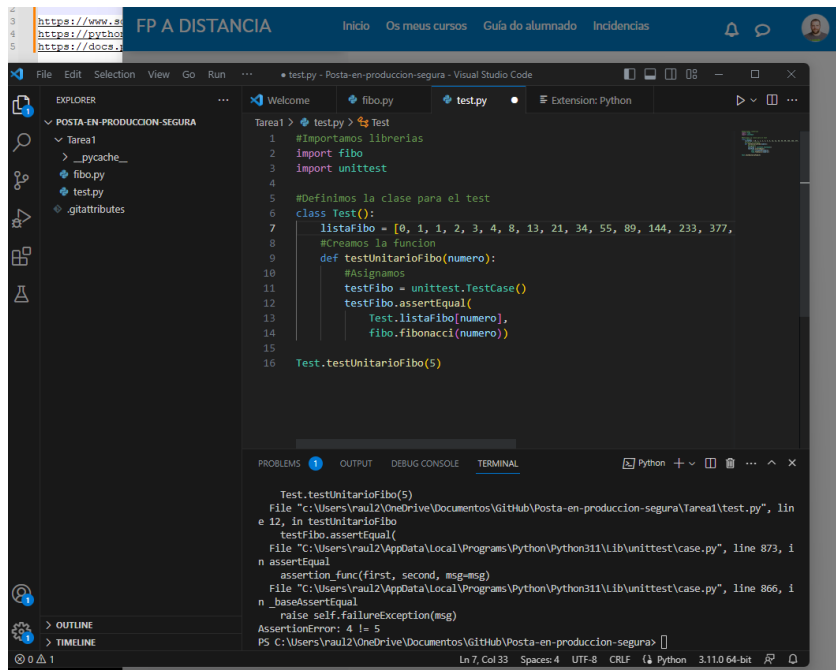
The status bar at the bottom indicates 'Ln 8, Col 18', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', and '3.11.0 64-bit'.

Comprobado que funciona como se espera, finalizamos los comentarios del archivo y continuamos.

### 3 - Test del programa

Tal como indica el ejercicio creamos una nueva clase para realizar los test unitarios de la aplicación con unittest, para este test buscamos una lista de numero Fibonacci correctos y lo guardamos en una variable, después para el test importamos nuestro programa que calcula el numero según la posición y se compara con la lista, si es correcto no devuelve nada y si falla devuelve el error con la diferencia entre los números.

Test con fallo a propósito:



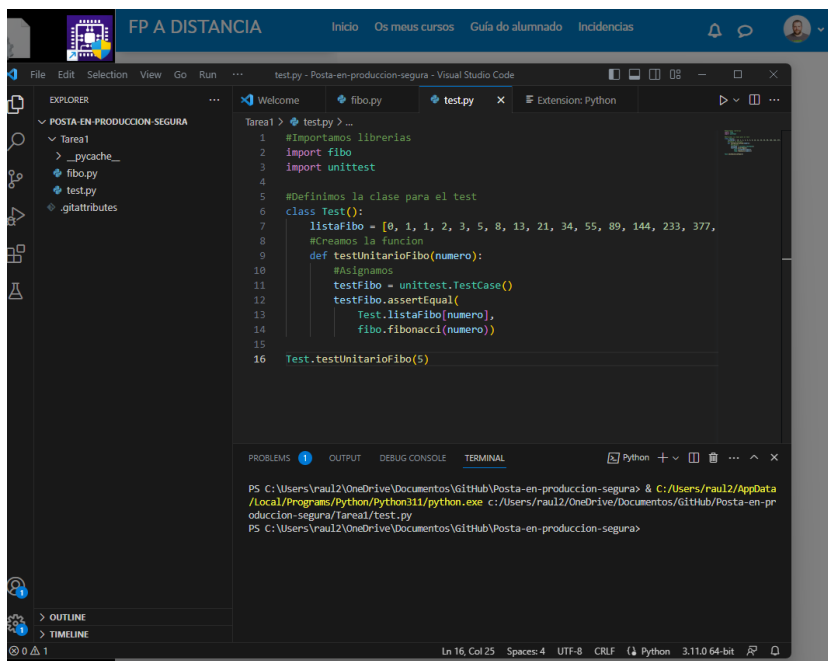
The screenshot shows the Visual Studio Code interface with a file named `test.py` open. The code defines a `Test` class with a `testUnitarioFibo` method. The method compares the output of `fibonacci` with a predefined list `listaFibo`. The test is run with the argument `5`. The terminal shows an `AssertionError` because the output of `fibonacci(5)` is `5`, but the expected value from `listaFibo` is `4`.

```
1 #Importamos librerias
2 import fibo
3 import unittest
4
5 #Definimos la clase para el test
6 class Test():
7     listaFibo = [0, 1, 1, 2, 3, 4, 8, 13, 21, 34, 55, 89, 144, 233, 377,
8     #Creamos la funcion
9     def testUnitarioFibo(numero):
10         #Asignamos
11         testFibo = unittest.TestCase()
12         testFibo.assertEqual(
13             Test.listaFibo[numero],
14             fibo.fibonacci(numero))
15
16 Test.testUnitarioFibo(5)
```

Terminal output:

```
Test.testUnitarioFibo(5)
File "C:\Users\raul2\OneDrive\Documents\GitHub\Posta-en-produccion-segura\Tarea1\test.py", line 12, in testUnitarioFibo
testFibo.assertEqual(
File "C:\Users\raul2\AppData\Local\Programs\Python\Python311\Lib\unittest\case.py", line 873, in assertEqual
assertion_func(first, second, msg=msg)
File "C:\Users\raul2\AppData\Local\Programs\Python\Python311\Lib\unittest\case.py", line 866, in _baseAssertEqual
raise self.failureException(msg)
AssertionError: 4 != 5
PS C:\Users\raul2\OneDrive\Documents\GitHub\Posta-en-produccion-segura>
```

Test sin fallos:



The screenshot shows the same Visual Studio Code interface with the `test.py` file. The terminal now shows the command to run the test successfully, without any errors.

```
PS C:\Users\raul2\OneDrive\Documents\GitHub\Posta-en-produccion-segura> & C:\Users\raul2\AppData\Local\Programs\Python\Python311\python.exe c:\Users\raul2\OneDrive\Documents\GitHub\Posta-en-pr
oduction-segura\Tarea1\test.py
PS C:\Users\raul2\OneDrive\Documents\GitHub\Posta-en-produccion-segura>
```

## 4 - Referencias

Para la realización de la tarea se han usado estas referencias:

Para la lista de números correcta:

<https://www.sdelsol.com/glosario/sucesion-fibonacci/#:~:text=0%2C%201%2C%201%2C%202,as%C3%AD%20sucesivamente%20hasta%20el%20infinito.>

Para el script de obtención del numero en referencia a su posición:

<https://pythondiario.com/2018/08/sucesion-de-fibonacci-con-python.html>

Para comprobar como se usaba unittest:

<https://docs.python.org/es/3.10/library/unittest.html>