# Tarea 1

## RA1. Prueba aplicaciones web y aplicaciones para dispositivos móviles analizando la estructura del código y su modelo de ejecución.

Cristian Stan
2023-2024
CE Ciberseguridad en entornos de las tecnologías de la información

# Índice

# Enunciado : Contexto práctico

Julián cuando necesita comprobar parte de un código en Python usa la librería incluida por defecto en las distribuciones de Python llamada unittest.

Tiene un pequeño programa llamado fibo.py que escribe la serie de Fibonacci hasta un número dado y devuelve el valor de esa misma posición.

Decide hacer un test del software de tipo aceptación. Es decir, quiere verificar que una parte del código se comporta de forma esperada. Para ello crea un script que importará la librería unittest y comprobará si el quinto número de la serie Fibonacci, el número 3, es correcto (la serie Fibonacci es 0,1,1,2,3,5,8,13…).

Para ello importa la librería de test de software unittest y define una clase donde incluiría la función para comprobar esta parte específica del código. Va a comprobar que el quinto número de la serie sea igual a 3, lo hace mediante la sentencia self.assertEqual(result, 3).

## 1: Crear un script que genere la secuencia de Fibonacci

El algoritmo ideado e implementado en Python para generar la secuencia de Fibonacci es el siguiente (aunque ya tenga los comentarios del propio código, se detallan ciertos aspectos a continuación):

```
# Se define una lista inicial de Fibonacci con los dos primeros elementos

lista_fibonacci = [0,1]

# Aquí empieza el algoritmo propiamente de Fibonacci

def fibonacci ():
    numero_1 = 0
    numero_2 = 1
    numero_fibonacci = 0

    # Solicita cuántas cifras vamos a querer en nuestra lista de Fibonacci

    numero_sucesion = int(input ("Escriba el número de cifras que tendrá la sucesión de
Fibonacci deseada: "))
    print ("\n")
```

```
# Bucle en el que se calculan los números, iterando numero_sucesion-2 veces ya que los
# dos primeros números de Fibonacci ya están en la lista, y actualizando la lista en cada
# iteración

    for iterador in range (0, numero_sucesion-2):
        numero_fibonacci = numero_1 + numero_2
        lista_fibonacci.append(numero_fibonacci)
        numero_1 = numero_2
        numero_2 = numero_fibonacci

# Se hace la impresión por pantalla de la lista completa

    print (*lista_fibonacci, sep=", ")
    print ("\n")
```

La definición inicial de la lista de Fibonacci se hace fuera de la propia función porque se va a usar en la función de testeo que se detalla más adelante.

Después, en la función en sí, se comienza definiendo las variables iniciales con las que se operará para ir calculando la sucesión.

En el bucle for, como queda comentado en el código, se itera ese número de veces (restando 2 al número total de cifras) ya que las dos primeras cifras ya están definidas. Luego, para el cálculo en sí, se suman los dos números definidos en la variable numero_fibonacci, la cual se añade a la lista ya descrita. Una vez hecho eso, se actualizan los dos valores iniciales para que correspondan con el anterior segundo número y con el anterior “número Fibonacci”, de forma que en la siguiente iteración se calcula correctamente el siguiente número Fibonacci

Finalmente, se imprime por pantalla la lista final completa, separando sus elementos por comas

## 2: Creación del programa principal

Cabe aclarar en este punto, que tanto la función de Fibonacci como lo que se hará a continuación para poder hacer el testeo estarán en el mismo archivo de código Python, formando parte por tanto todo del mismo “programa principal”

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

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

## 2.3. Crear nuestra clase (del tipo unittest.TestCase)

## 2.4. Dentro de esta clase definir una función (se puede llamar como se considere, pero se recomienda un nombre ilustrativo)

Para facilitar la explicación de los puntos 2.1, 2.2, 2.3 2.4 se harán todas juntas a continuación:

En primer lugar vemos el código Python realizando lo requerido (como anteriormente, aparte de los comentarios hechos en el propio código, se detallan algunos aspectos a continuación):

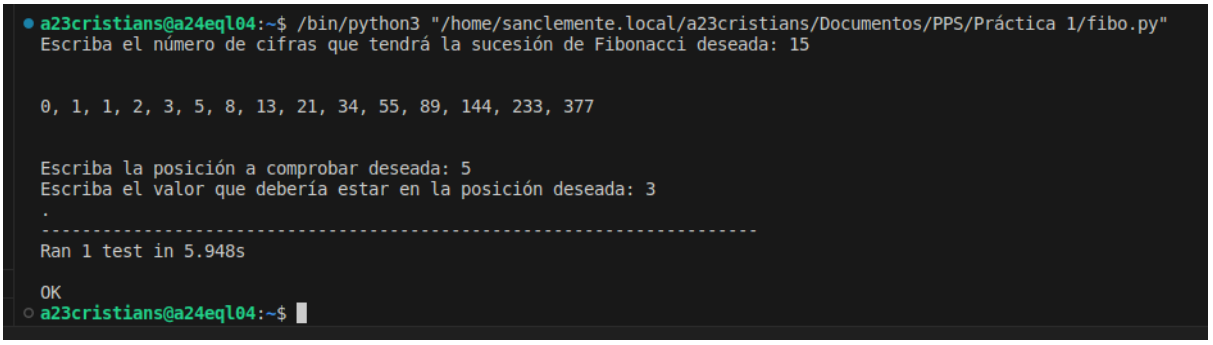
```
# Se importa la librería unittest para poder trabajar con sus funciones

import unittest

# Se declara la clase de testeo para definir en ella la función para comprobar el valor de una
# posición dada

class Test(unittest.TestCase):
    def test_Check_Position(self):
        posicion_pedida = int (input ("Escriba la posición a comprobar deseada: "))
        valor_esperado = int (input ("Escriba el valor que debería estar en la posición deseada:
"))
        self.assertEqual(lista_fibonacci[posicion_pedida-1], valor_esperado)

# Inicio del main del programa, en el que se llama a la función de Fibonacci y al main de
# unittest

if __name__ == "__main__":
    fibonacci ()
    unittest.main()
```

Pasando a comentar la función de testeo tes_Check_Position, lo más destacable es que se solicita al usuario en primer lugar la posición que se quiere comprobar y después se solicita el valor que se cree que debería estar ahí. Para la comprobación en sí con el assertEqual, se comprueba la posicion_pedida-1, ya que las listas comienzan en la posición cero.

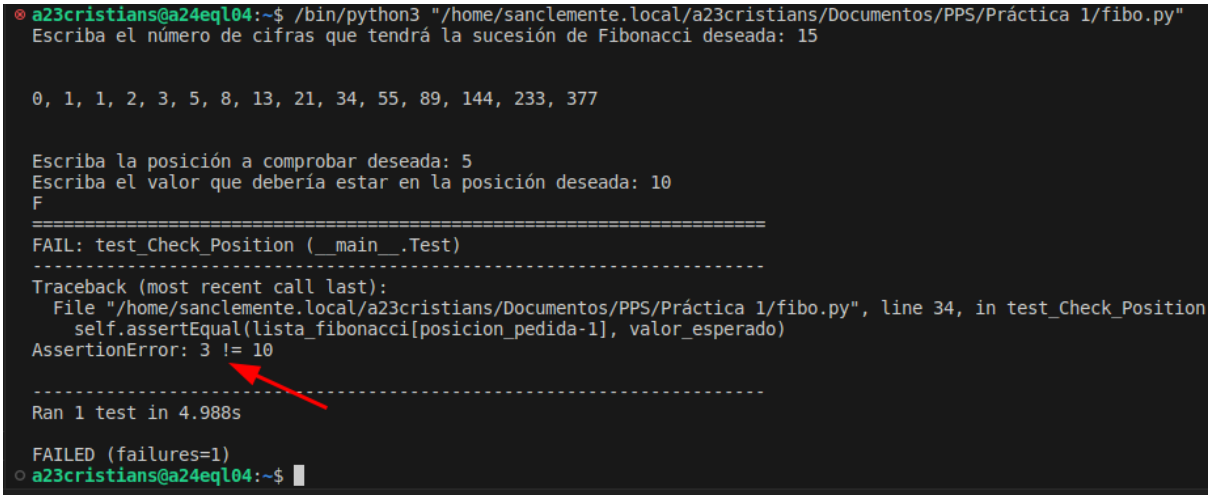
## 3: Verificación de software y pregunta final

*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?*

A continuación vemos una captura de la ejecución del programa para una sucesión de Fibonacci con 15 elementos y comprobando la situación de ejemplo del enunciado, de la 5ª posición de la sucesión




En la siguiente captura se puede ver qué ocurre si en el valor esperado se introduce un valor diferente al correcto, pudiendo observar también la aclaración que hace en el lugar señalado




Por tanto esta prueba realizada se podría considerar de aceptación, ya que se revisa el funcionamiento de todo el conjunto del software, en este caso para el cálculo de la sucesión de Fibonacci, mediante la comprobación de alguno de sus elementos comparándolo con el valor que debería tener sabiendo de antemano la sucesión de Fibonacci real por algún otro medio.