



NOVIEMBRE 2023

TAREA UNIDAD 1

Sucesión Fibonacci

JUAN MARTÍNEZ GÓMEZ



ÍNDICE

1. [Introducción](#)
2. [Equipos, software y tecnologías utilizadas](#)
3. [Enunciado de la práctica](#)
4. [Desarrollo](#)
 - 4.1 [Crea un script que genere la secuencia de Fibonacci](#)
 - 4.2 [Creación del programa principal](#)
 - 4.3 [Definición de la función test](#)
 - 4.4 [Verificación de software y pregunta final](#)
5. [Bibliografía](#)

1. Introducción

En este ejercicio pondremos en práctica los conocimientos adquiridos en la primera unidad del módulo, *Prueba de aplicaciones web y móviles, analizando el código y su modelo de ejecución*, que incluye un mejor entendimiento de los lenguajes de programación y la gestión de las versiones de un proyecto.

Para ello desarrollaremos un programa desde cero en un lenguaje de programación de alto nivel interpretado, el cual además de hacer una función determinada contará con la posibilidad de hacer pruebas de software para garantizar que la función hace lo que queremos, todo ello con un control de versiones que irá registrando los cambios realizados.

[Volver al índice](#)

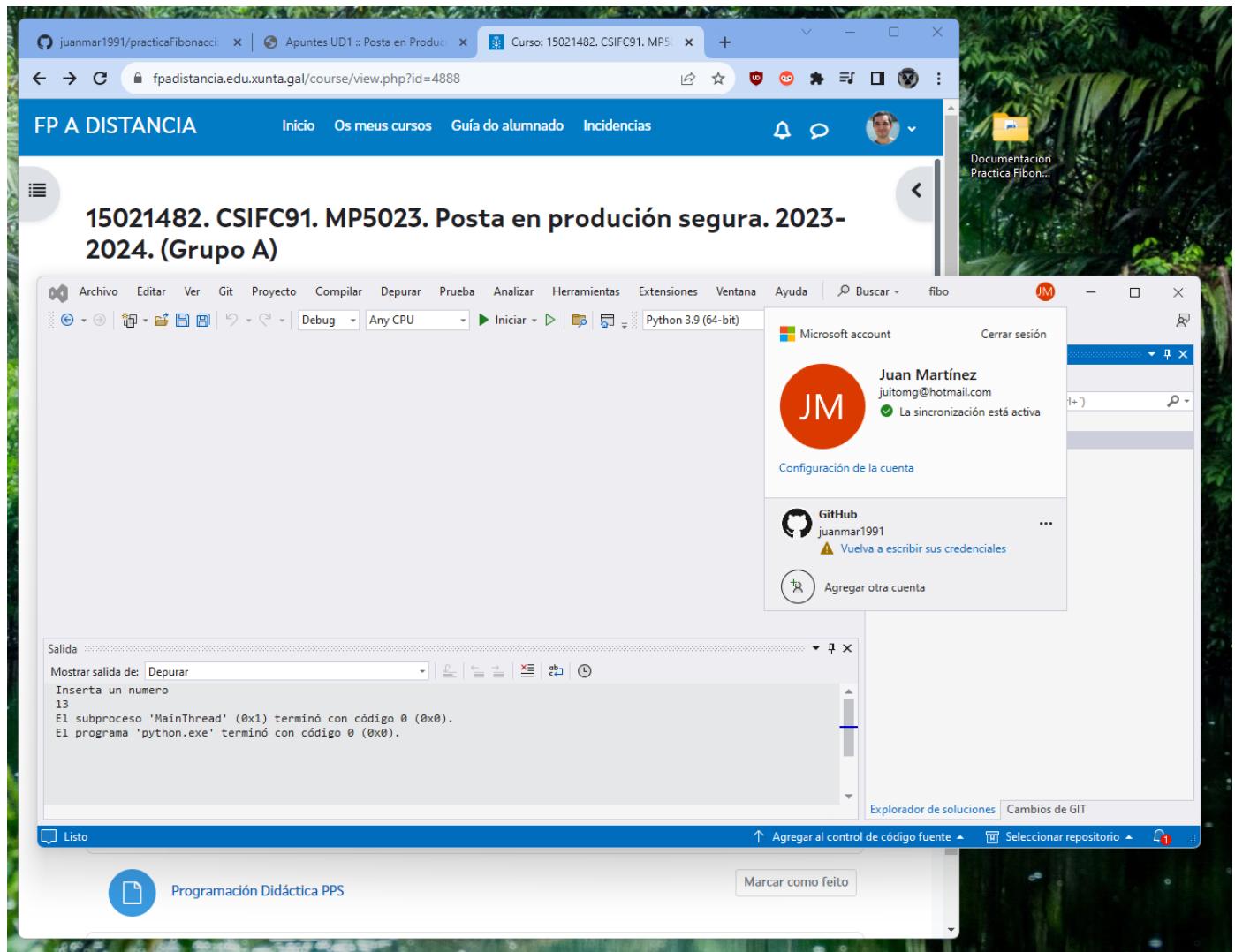
2. Equipos, software y tecnologías utilizadas

Esta práctica requiere aprender a utilizar Git, así que nos obligaremos a utilizar dos equipos físicos para hacer pruebas. Un portátil Omen con Windows 11 y un Macbook Air M1 con MacOS Sonoma. También utilizaremos un smartphone Android para algunas consultas rápidas.

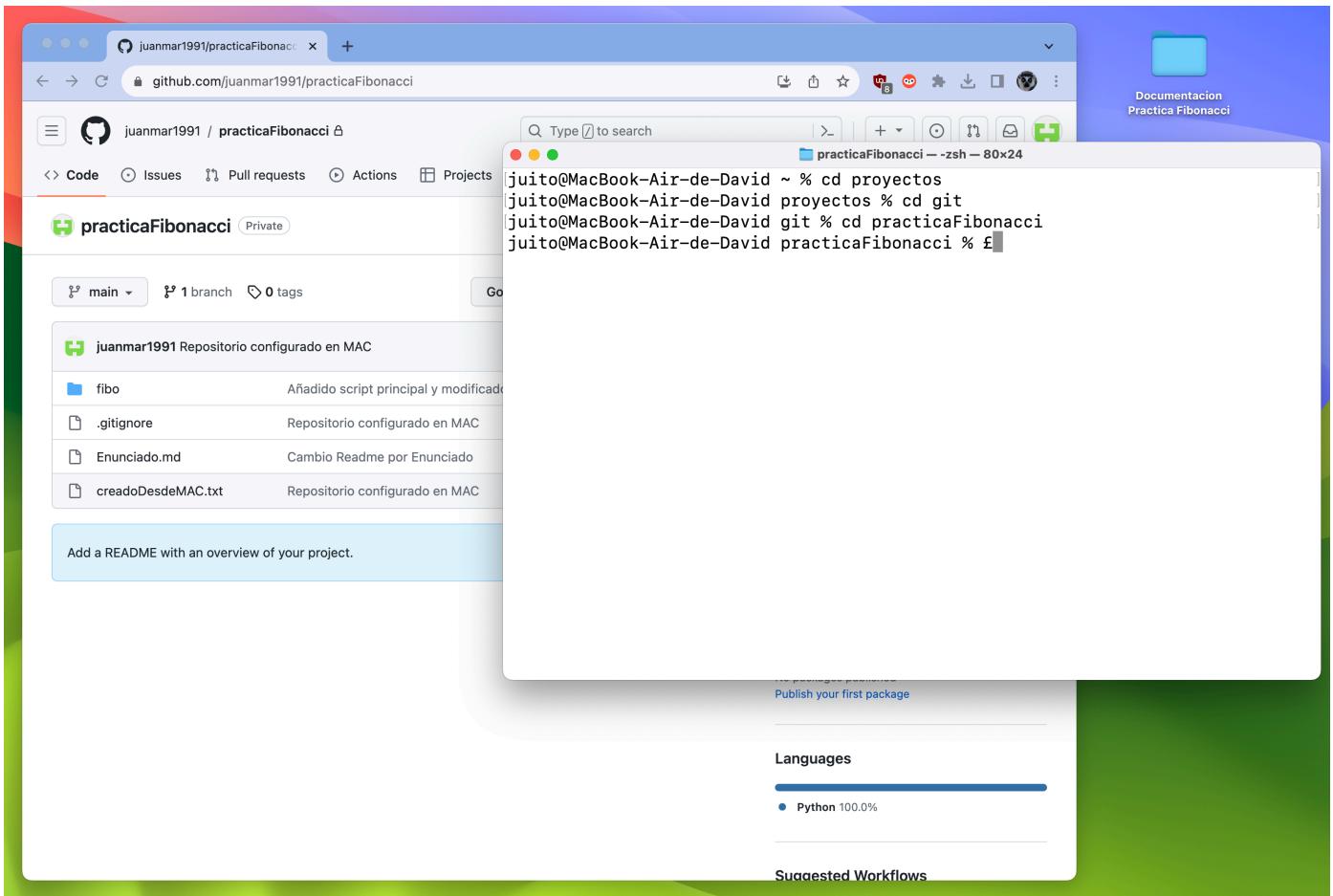
El lenguaje de programación será Python, el IDE Visual Studio. Para el control de versiones utilizaremos GIT, creando además una cuenta en Github. Aprovechando los conocimientos adquiridos en otro módulo, redactaremos la documentación en MarkDown utilizando Typora. A mayores, el navegador Google Chrome para acceder a Github, a la plataforma del instituto, y para hacer varias búsquedas en internet relacionadas con la práctica.

Intentaremos dejar claro en las capturas la autoría con alguna referencia visible cuando se pueda. La cuenta de GitHub será juanmar1991, registrada con la cuenta del instituto. Mientras tanto en Visual Studio aparecerá un "JM" en la parte superior del programa al haberlo registrado con mi cuenta personal de Hotmail (juitomg). Dejo una captura donde se ve más claro. Por último, como aclaración, el Macbook es

compartido y tiene el nombre de otra persona (Macbook de David), pero en la consola figura el usuario Juito utilizando el ordenador, que soy yo.



Captura donde se aprecian varios elementos identificativos mencionados



Más elementos identificativos en MAC

[Volver al índice](#)

3. Enunciado de la práctica

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)`.

¿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?

[Volver al índice](#)

4. Desarrollo

Empezaremos de cero instalando Python y Visual Studio Community en Windows 11, a través de los enlaces que recomiendan en el Curso de Python proporcionado en los apuntes de la Unidad 1. Paralelamente instalaremos y configuraremos GIT tanto en Windows como en MAC, siguiendo también el curso de GIT proporcionado en los apuntes.

Tras documentarnos y entender cómo funciona la sucesión Fibonacci, intentaremos crear la base de la función. Esto nos llevará un tiempo con el método de ensayo y error, hasta dar con la solución.

Fibonacci

1, 1, 2, 3, 5, 8, 13, 21 y 34

```
def fibo(numero):
```

```
    num1=0  
    num2=1
```

```
    for x in range(numero):
```

```
        num3=num1+num2
```

```
        num1=num2
```

```
        num2=num3
```

```
    return num1
```

4

$N1 = 1$
 $N2 = 1$
 $N3 = 2$

$1 \times 2 = 3$

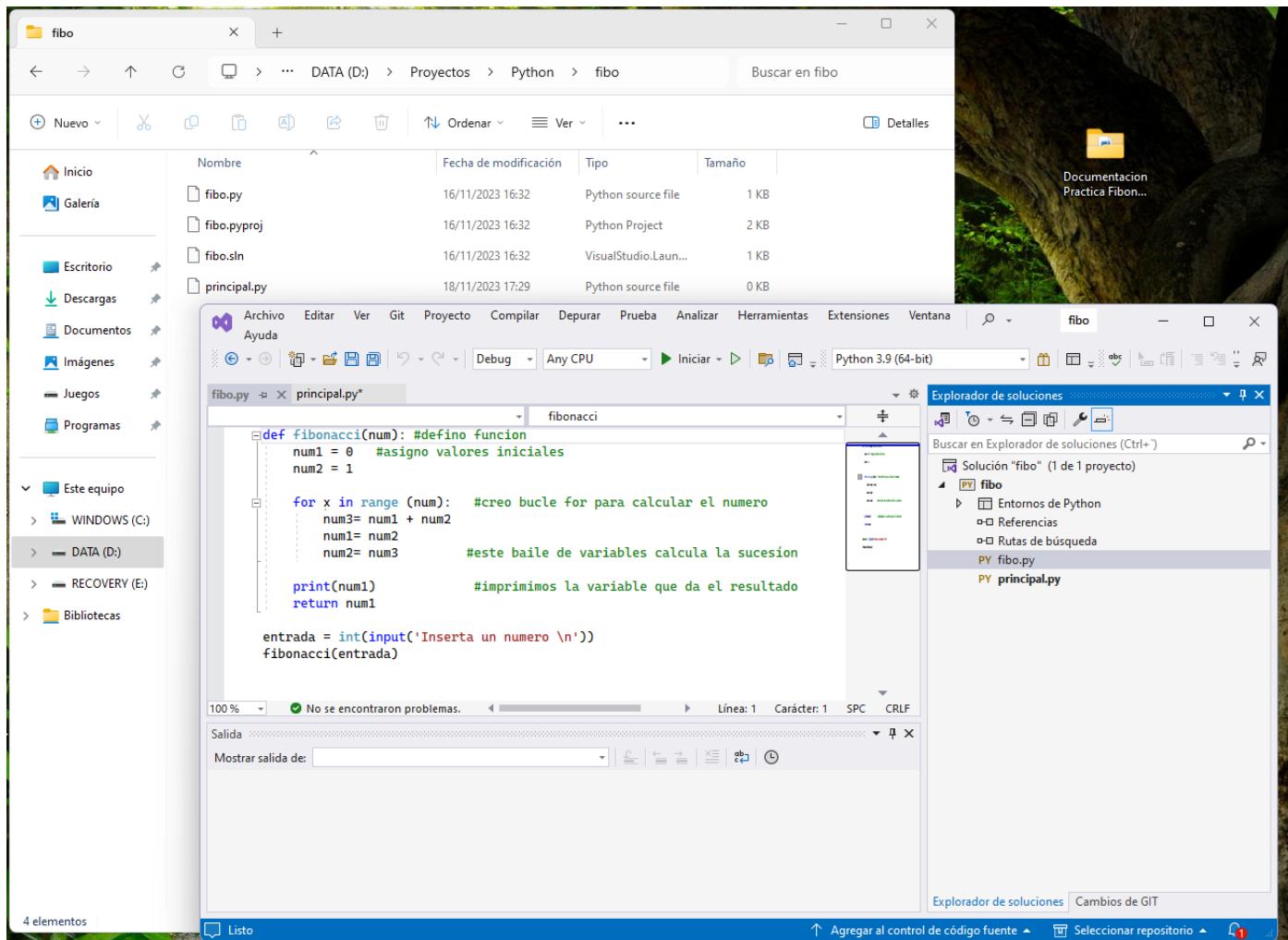


Notas tomadas en el teléfono buscando la solución

[Volver al índice](#)

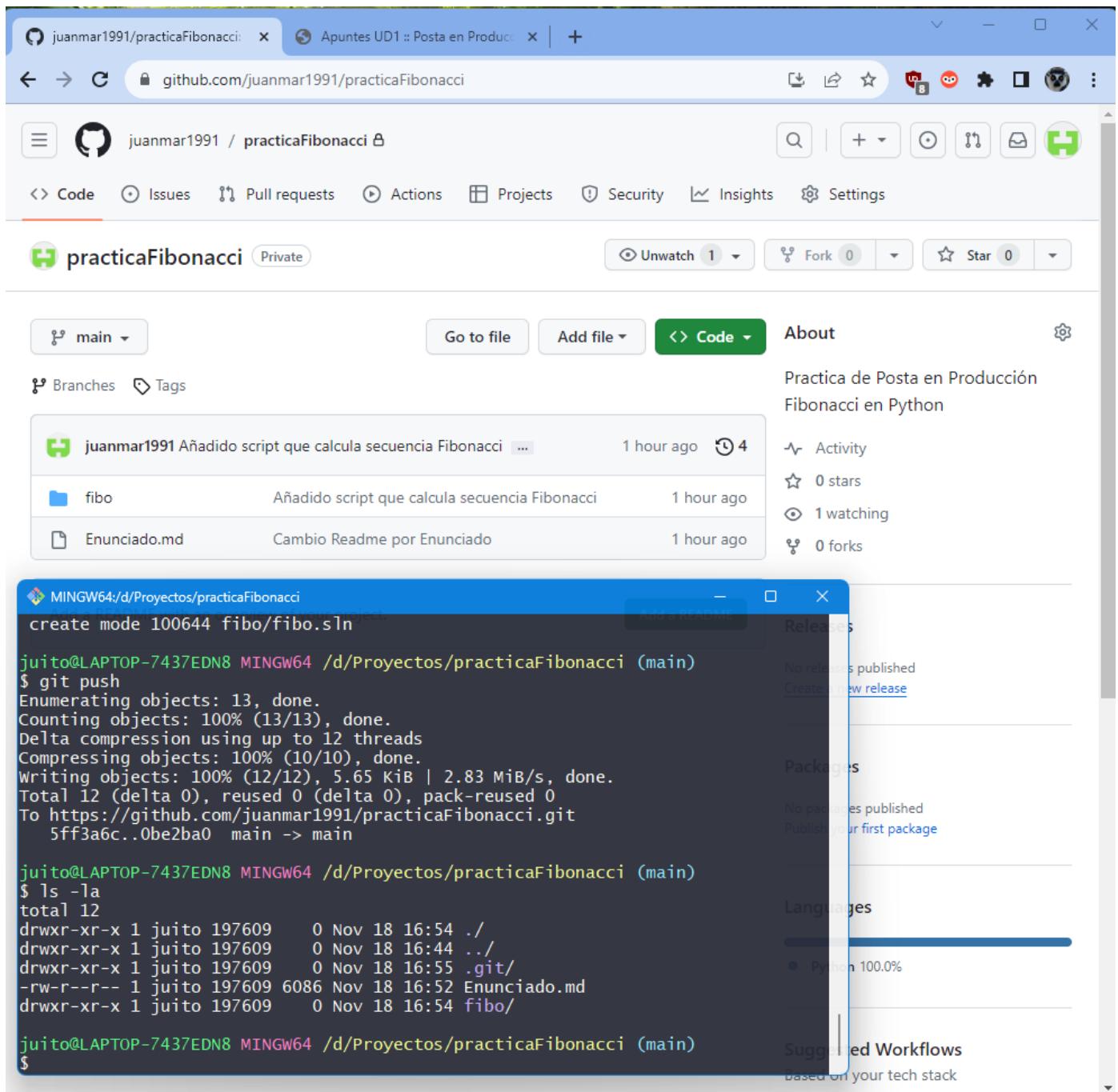
4.1 Crea un script que genere la secuencia de Fibonacci

Tras descubrir algo que puede funcionar acudiremos al IDE y lo probaremos. La función declara dos variables (*num1* y *num2*) con valores 0 y 1 respectivamente, después dentro de un bucle *for* se calcula el número que queramos de la sucesión, apoyándose en una tercera variable (*num3*). Para comprobar que funciona pediremos que nos devuelva por texto el valor. Funciona, podremos dar por cumplido el punto 1 de la tarea.



Vista del código provisional que calcula la sucesión. Nota: Esta captura fue tomada en un momento posterior, cuando ya se había creado el archivo principal

En este punto aprovecharemos para subir el progreso del proyecto a nuestro repositorio en GitHub. En un principio el repositorio se llamará *practicaFibonacci* y tendrá un enunciado junto con la carpeta con los scripts.



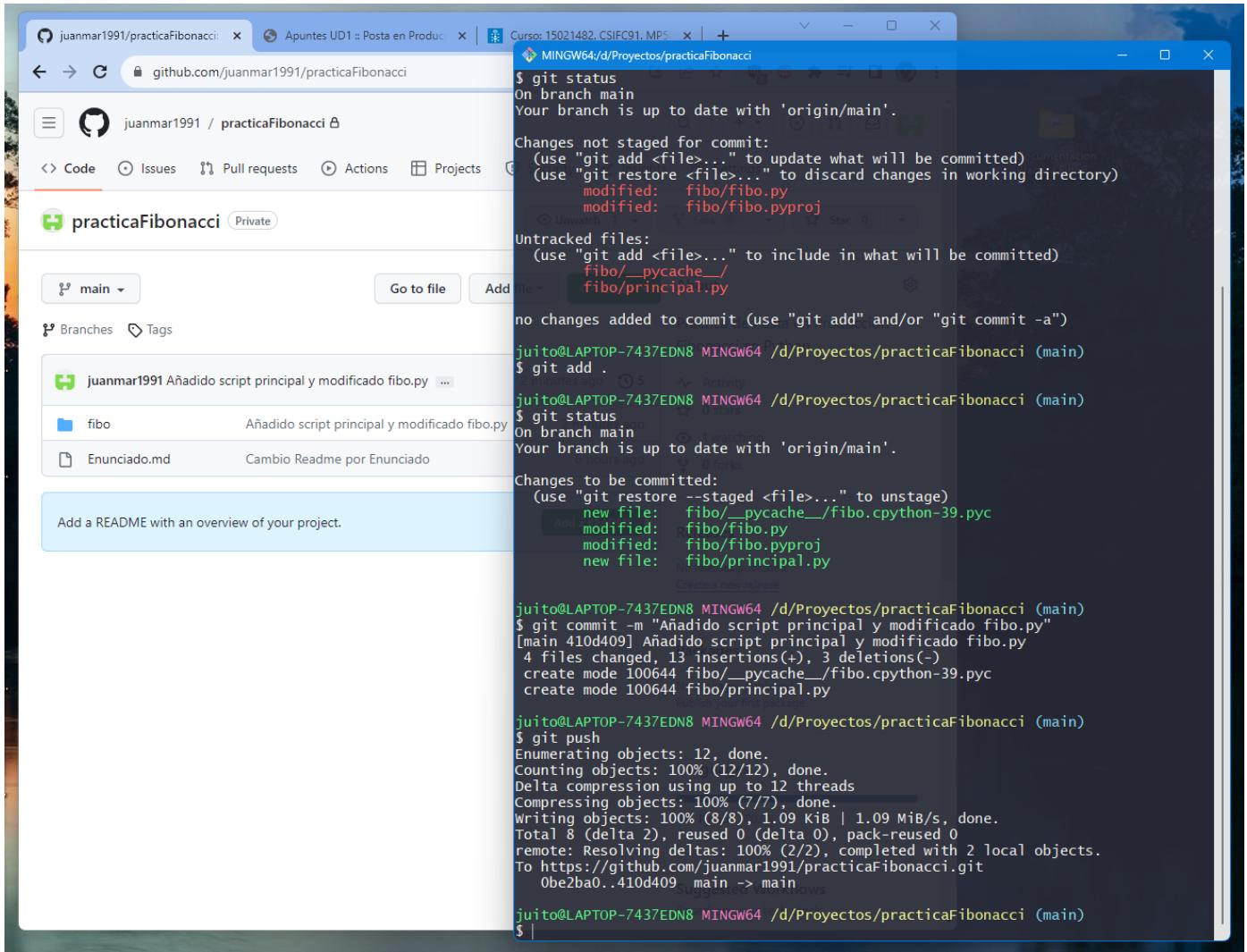
The screenshot shows a browser window with two tabs: 'juanmar1991/practicaFibonacci' and 'Apuntes UD1 :: Posta en Producción'. The GitHub page for 'practicaFibonacci' is visible, showing a private repository with 4 commits. The terminal window below shows the command 'git push' being run, followed by the output of the push operation, including the creation of a new commit and the pushing of changes to the 'main' branch. A tooltip is visible on the right side of the GitHub page, providing information about the repository's activity, stars, watchers, and forks.

```
MINGW64:/d/Proyectos/practicaFibonacci
create mode 100644 fibo/fibo.sln
juito@LAPTOP-7437EDN8 MINGW64 /d/Proyectos/practicaFibonacci (main)
$ git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 12 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (12/12), 5.65 KiB | 2.83 MiB/s, done.
Total 12 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/juanmar1991/practicaFibonacci.git
  5ff3a6c..0be2ba0  main -> main

MINGW64:/d/Proyectos/practicaFibonacci (main)
$ ls -la
total 12
drwxr-xr-x 1 juito 197609 0 Nov 18 16:54 .
drwxr-xr-x 1 juito 197609 0 Nov 18 16:44 ../
drwxr-xr-x 1 juito 197609 0 Nov 18 16:55 .git/
-rw-r--r-- 1 juito 197609 6086 Nov 18 16:52 Enunciado.md
drwxr-xr-x 1 juito 197609 0 Nov 18 16:54 fibo/
MINGW64:/d/Proyectos/practicaFibonacci (main)
$
```

Actualización del proyecto en Github tras un *git push*

Seguiremos con el punto 2 de la tarea, donde nos piden crear el programa principal desde el que llamaremos a nuestra función *fibonacci()*. Tras ello probaremos a hacer un *commit* con la creación del nuevo archivo.



The screenshot shows a terminal window with a GitHub repository and a terminal session. The GitHub repository is 'practicaFibonacci' (Private). The terminal session shows the following commands and output:

```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   fibo/fibo.py
      modified:   fibo/fibo.pyproj

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    fibo/__pycache__/
    fibo/principal.py

no changes added to commit (use "git add" and/or "git commit -a")

juito@LAPTOP-7437EDN8 MINGW64 /d/Proyectos/practicaFibonacci (main)
$ git add .
juito@LAPTOP-7437EDN8 MINGW64 /d/Proyectos/practicaFibonacci (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   fibo/__pycache__/_fibo.cpython-39.pyc
    modified:   fibo/fibo.py
    modified:   fibo/fibo.pyproj
    new file:   fibo/principal.py

juito@LAPTOP-7437EDN8 MINGW64 /d/Proyectos/practicaFibonacci (main)
$ git commit -m "Añadido script principal y modificado fibo.py"
[main 410d409] Añadido script principal y modificado fibo.py
4 files changed, 13 insertions(+), 3 deletions(-)
create mode 100644 fibo/__pycache__/_fibo.cpython-39.pyc
create mode 100644 fibo/principal.py
juito@LAPTOP-7437EDN8 MINGW64 /d/Proyectos/practicaFibonacci (main)
$ git push
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 1.09 KiB | 1.09 MiB/s, done.
Total 8 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/juanmar1991/practicaFibonacci.git
  0be2ba0..410d409 main -> main
juito@LAPTOP-7437EDN8 MINGW64 /d/Proyectos/practicaFibonacci (main)
$ |
```

Commit con los nuevos cambios

[Volver al índice](#)

4.2 Creación del programa principal

Siguiendo con el código, en el archivo *principal.py* importaremos las librerías que nos permitirán probar el código más adelante, y la función *fibonacci* del archivo *fibo.py*. Después definiremos la clase *Test*, y en ese punto pediremos un número para activar la función *fibonacci()* asegurándonos de que funciona.

The screenshot shows the Microsoft Visual Studio Code interface. The top menu bar includes Archivo, Editar, Ver, Git, Proyecto, Compilar, Depurar, Prueba, Analizar, Herramientas, Extensiones, Ventana, Ayuda, Buscar, and a Python 3.9 (64-bit) dropdown. The status bar indicates 'Proceso: [4952] principal.py' and 'Eventos del ciclo de vida'. The code editor has tabs for 'fibo.py' and 'principal.py', with 'principal.py' currently selected. The code in 'principal.py' is a test script using unittest. A terminal window below shows the execution of the script, prompting for a number and displaying the result (8, 21). The right side of the interface includes an 'Explorador de soluciones' (Solution Explorer) and a 'Cambiros de GIT' (Git Changes) panel.

```
import unittest
from fibo import fibonacci

class Test(unittest.TestCase):
    def test_fibonacci(self):
        entrada = int(input('Inserta un numero \n'))
        fibonacci(entrada)

#Importamos unittest
#Importamos la funcion Fibonacci del archivo fibo.py

#definimos clase Test
#Pedimos un numero por teclado
#utilizamos el numero en la funcion
```

```
C:\Program Files (x86)\Micros
Inserta un numero
8
21
Press any key to continue . . . |
```

El código funciona. Nos devuelve un número calculado en la función fibonacci() del otro archivo

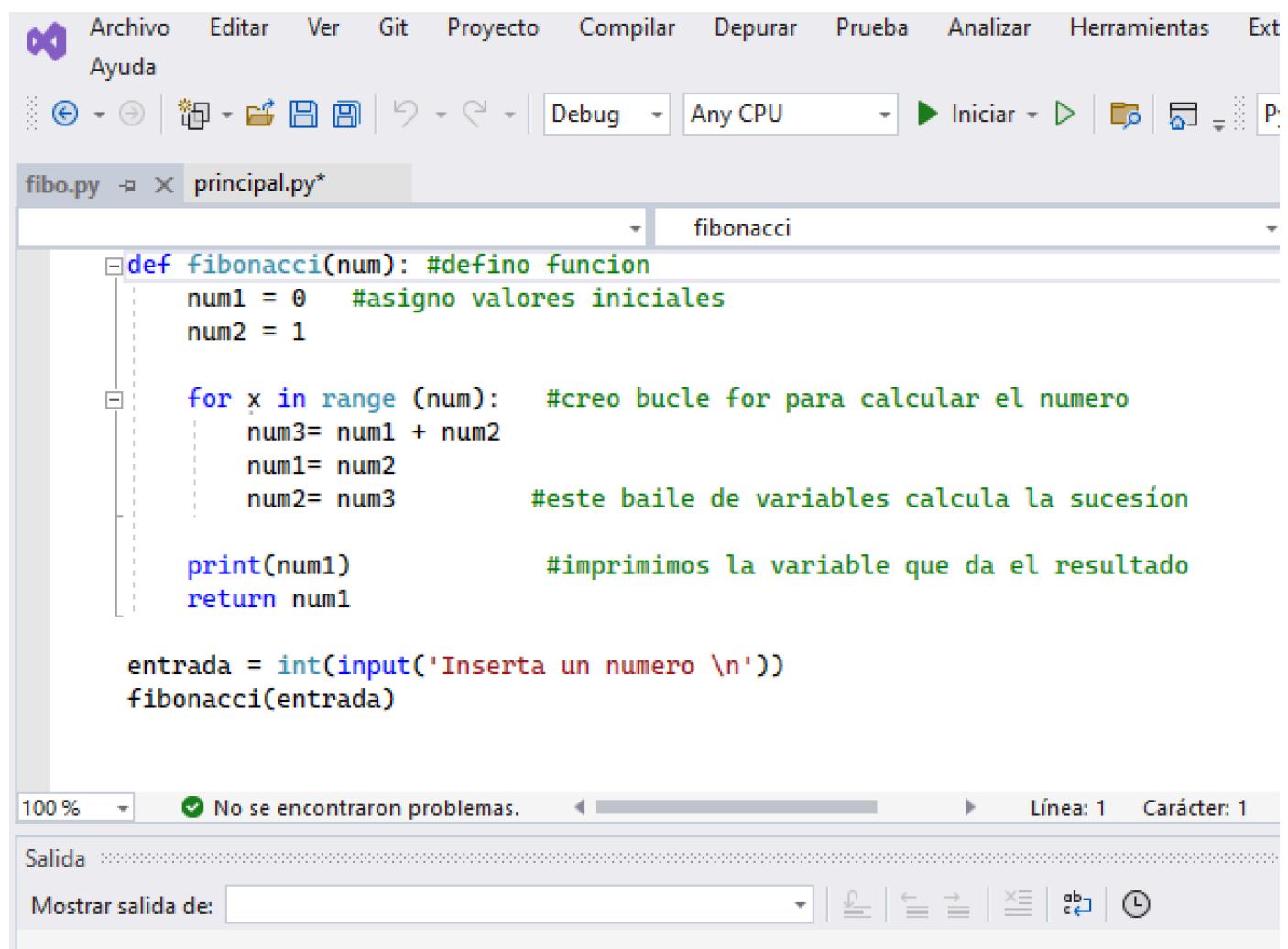
Para seguir practicando las posibilidades de *G/T* que se comentaron en la unidad, intentaremos acceder a nuestro proyecto desde otro ordenador, un Macbook. Tras clonar con éxito el proyecto, crearemos un archivo desde el Macbook y lo añadiremos al proyecto. Por el camino también crearemos el archivo *.gitignore* para evitar que el fichero *.DSStore* se suba al proyecto.

The screenshot shows a GitHub repository page for 'practicaFibonacci'. The repository is private and has 1 branch and 0 tags. The 'Code' tab is selected. The repository contains a 'fibo' folder, a '.gitignore' file, an 'Enunciado.md' file, and a 'creadoDesdeMAC.txt' file. A terminal window on the right shows the command 'ls -la' being run, listing the files and their permissions and timestamps. The GitHub interface includes a sidebar with 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings'.

```
juito@MacBook-Air-de-David practicaFibonacci % ls -la
total 48
drwxr-xr-x  8 juito  staff  256 19 nov 11:50 .
drwxr-xr-x  5 juito  staff  160 19 nov 11:34 ..
-rw-r--r--@  1 juito  staff  6148 19 nov 11:38 .DS_Store
drwxr-xr-x 13 juito  staff  416 19 nov 11:56 .git
-rw-r--r--  1 juito  staff  10 19 nov 11:50 .gitignore
-rw-r--r--  1 juito  staff  6027 19 nov 11:34 Enunciado.md
-rw-r--r--  1 juito  staff  25 19 nov 11:47 creadoDesdeMAC.txt
drwxr-xr-x  8 juito  staff  256 19 nov 11:34 fibo
```

4.3 Definición de la función test

Aprovecharemos para continuar nuestro código en MAC. Gracias a las librerías que importamos antes, podremos escribir la función que nos permita hacer la comprobación indicada. Le pediremos que calcule la quinta posición de la sucesión Fibonacci y compruebe si el resultado es tres. Realizando esta parte nos daremos cuenta de un error que hemos arrastrado desde el principio. Nuestro código calcula la sucesión saltándose el cero, de forma que todas las posiciones posteriores son erróneas.



The screenshot shows the Microsoft Visual Studio Code interface. The top menu bar includes Archivo, Editar, Ver, Git, Proyecto, Compilar, Depurar, Prueba, Analizar, Herramientas, and Ext. The toolbar includes icons for file operations, search, and debugging. The status bar at the bottom shows 100% completion, no problems found, Linea: 1, Carácter: 1, and a 'Salida' (Output) section. The code editor displays the following Python script:

```
def fibonacci(num): #defino funcion
    num1 = 0 #asigno valores iniciales
    num2 = 1

    for x in range (num): #creo bucle for para calcular el numero
        num3= num1 + num2
        num1= num2
        num2= num3 #este baile de variables calcula la sucesion

    print(num1) #imprimimos la variable que da el resultado
    return num1

entrada = int(input('Inserta un numero \n'))
fibonacci(entrada)
```

Este código da como resultado la sucesión 1, 1, 2, 3, 5, 8, 13, 21... Nos interesa que empiece desde cero

Acudiremos al código de *fibo.py* para corregirlo. Tras muchas pruebas daremos con la solución. Si le restamos el valor de la variable *num1* a la variable *num2* nos da justo el valor que necesitamos.

The screenshot shows a web-based development environment for a course. The top navigation bar includes links for 'Inicio', 'Os meus cursos', 'Guía do alumnado', and 'Incidencias'. A user profile is visible on the right. The main content area is titled '15021482. CSIFC91. MP5023. Posta en producción segura. 2023-2024. (Grupo A)'. On the left, there's a sidebar with sections for 'Xeral' and 'Área de comunicacions' (including 'Foro de noticias' and 'Foro de presentación'). The central area features a code editor with Python code for calculating a Fibonacci sequence. Below the code editor is a terminal window showing a test failure for the fifth Fibonacci number. The file explorer on the left shows the project structure with files like 'principal.py', 'fibo.py', and 'fibo.sln'.

```
def fibonacci(num): #defino función
    num1 = 0 #asigno valores iniciales
    num2 = 1
    for x in range (num): #creo bucle for para calcular el numero
        num3= num1 + num2
        num1= num2
        num2= num3 #este baile de variables calcula la sucesión
    print(num2 - num1) #imprimimos el resultado de la resta
    return num2 - num1 # Devolvemos el valor
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL ... Python + ...

FAIL: test_comprobación_fibonacci (test_fibonacci.Test)

Traceback (most recent call last):

File "/Users/juitor/Proyectos/GIT/practicaFibonacci/fibo/principal.py", line 9, in test_comprobación_fibonacci

self.assertEqual (fibonacci(4),3)

AssertionError: 2 != 3

Ran 1 test in 0.000s

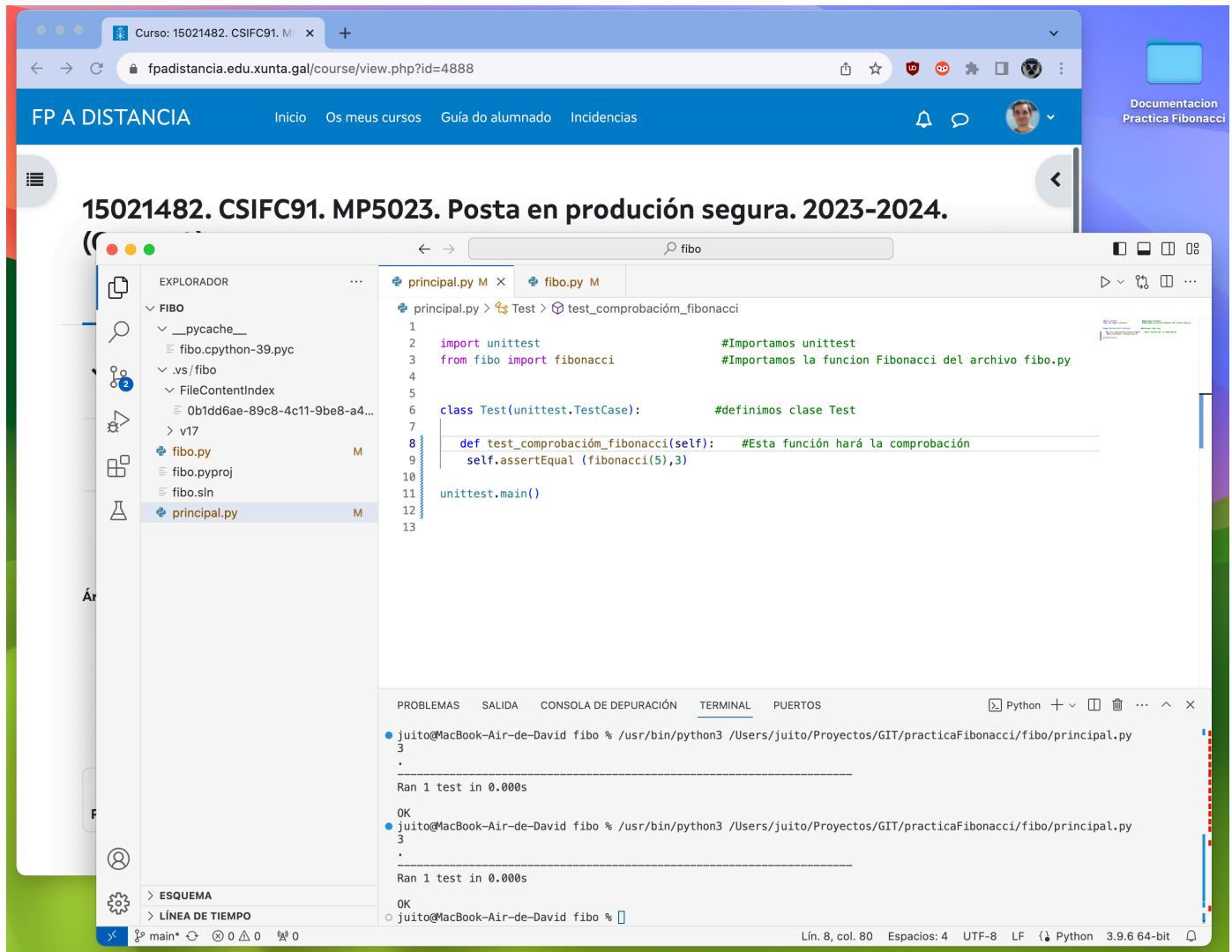
FAILED (failures=1)

juito@MacBook-Air-de-David fibo %

Lín. 9, col. 71 Espacios: 4 UTF-8 LF { Python 3.9.6 64-bit

Este código sí devuelve los valores correspondientes: **0, 1, 1, 2, 3, 5, 8, 13, 21...**

Tras este traspies podremos volver al archivo *principal.py* y ejecutar el código para comprobar que todo está correcto. Cuando le preguntamos si la quinta posición de la sucesión Fibonacci es tres, nos da un Ok



FP A DISTANCIA

15021482. CSIFC91. MP5023. Posta en producción segura. 2023-2024.

EXPLORADOR

- ✓ FIBO
 - ✓ __pycache__
 - ✓ fibo.cpython-39.pyc
 - ✓ .vs/fibo
 - ✓ FileContentIndex
 - ✓ 0b1dd6ae-89c8-4c11-9be8-a4...
 - > v17
 - ✓ fibo.py
 - ✓ fibo.pyproj
 - ✓ fibo.sln
 - ✓ principal.py M

principal.py M fibo.py M

```
1  import unittest
2  from fibo import fibonacci
3
4
5  class Test(unittest.TestCase):
6      #definimos clase Test
7
8      def test_comprobación_fibonacci(self):
9          #Esta función hará la comprobación
10         self.assertEqual (fibonacci(5),3)
11
12         unittest.main()
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
juito@MacBook-Air-de-David fibo % /usr/bin/python3 /Users/juito/Proyectos/GIT/practicaFibonacci/fibo/principal.py
3
.
Ran 1 test in 0.000s
OK
juito@MacBook-Air-de-David fibo % /usr/bin/python3 /Users/juito/Proyectos/GIT/practicaFibonacci/fibo/principal.py
3
.
Ran 1 test in 0.000s
OK
juito@MacBook-Air-de-David fibo %
```

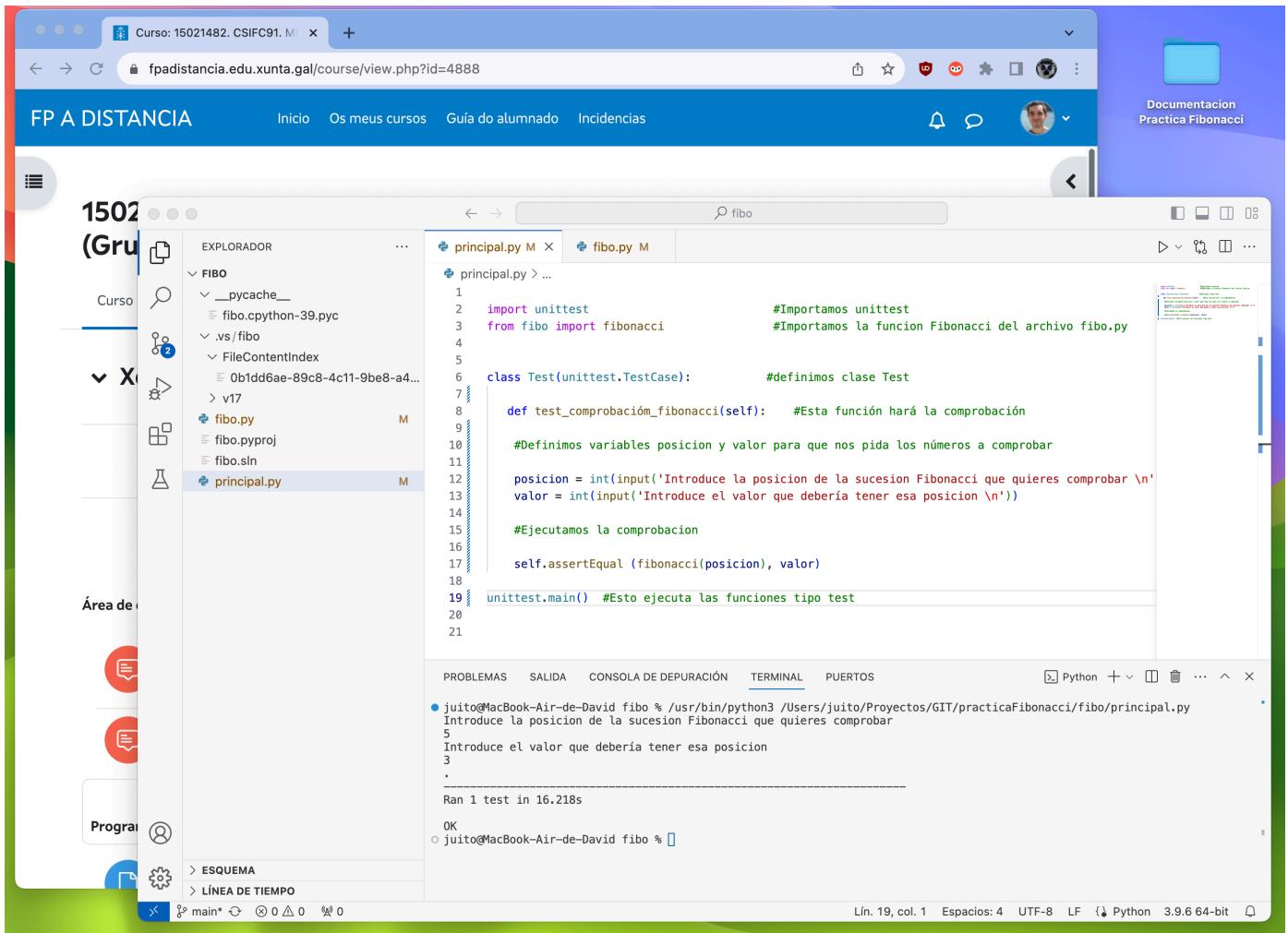
Lín. 8, col. 80 Espacios: 4 UTF-8 LF Python 3.9.6 64-bit

Ahora sí, recibimos el Ok que esperábamos

[Volver al índice](#)

4.4 Verificación de software y pregunta final

Solo quedará poner algunas líneas de código para que el programa nos solicite los datos a comprobar, en lugar de estar ya escritos, para poder comprobar otros valores si se desea. Primero nos pedirá la posición de la sucesión Fibonacci que queremos comprobar, y después el valor que debería tener dicha posición.



The screenshot shows a web-based development environment. On the left, a sidebar titled 'FP A DISTANCIA' displays a course structure with a 'CURSO' section and a 'ÁREA DE TRABAJO' section containing icons for file management, terminal, and help. The main area is titled '1502 (Gr...' and shows a file explorer with a 'FIBO' folder containing files like 'fibo.py', 'fibo.pyproj', and 'fibo.sln'. Two tabs are open in the code editor: 'principal.py' and 'fibo.py'. The 'principal.py' tab contains the following code:

```
1  import unittest
2  from fibo import fibonacci
3
4
5  class Test(unittest.TestCase):
6      #definimos clase Test
7
8      def test_comprobacion_fibonacci(self):
9          #Esta función hará la comprobación
10
11         #Definimos variables posicion y valor para que nos pida los números a comprobar
12
13         posicion = int(input('Introduce la posicion de la sucesion Fibonacci que quieras comprobar \n'))
14         valor = int(input('Introduce el valor que debería tener esa posicion \n'))
15
16         #Ejecutamos la comprobacion
17
18         self.assertEqual (fibonacci(posicion), valor)
19
20
21         unittest.main() #Esto ejecuta las funciones tipo test
```

The 'fibo.py' tab contains the following code:

```
1  import fibo
2
3  print(fibo.fibonacci(10))
```

Below the code editor is a terminal window showing the execution of the code:

```
juito@MacBook-Air-de-David fibo % /usr/bin/python3 /Users/juito/Proyectos/GIT/practicaFibonacci/fibo/principal.py
Introduce la posicion de la sucesion Fibonacci que quieras comprobar
5
Introduce el valor que debería tener esa posicion
3
.
-----
Ran 1 test in 16.218s
OK
juito@MacBook-Air-de-David fibo %
```

At the bottom, status information is displayed: Lín. 19, col. 1 Espacios: 4 UTF-8 LF Python 3.9.6 64-bit.

Si le preguntamos si tres es el valor adecuado para la posición cinco, nos da Ok

The screenshot shows a web-based development environment. On the left, a sidebar displays course information: '1502 (Gru)' and 'Curso'. The main area is a code editor with two tabs: 'principal.py' and 'fibo.py'. The code in 'principal.py' is as follows:

```

1  import unittest
2  from fibo import fibonacci
3
4
5  class Test(unittest.TestCase):
6      #definimos clase Test
7
8      def test_comprobació_fibonacci(self):
9          #Esta función hará la comprobación
10
11         #Definimos variables posicion y valor para que nos pida los números a comprobar
12
13         posicion = int(input('Introduce la posicion de la sucesión Fibonacci que quieres comprobar \n'))
14         valor = int(input('Introduce el valor que debería tener esa posicion \n'))
15
16         #Ejecutaremos la comprobacion
17
18         self.assertEqual (fibonacci(posicion), valor)
19
20         unittest.main() #Esto ejecuta las funciones tipo test
21

```

Below the code editor is a terminal window showing the execution of the test. The output is:

```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
2 F
=====
FAIL: test_comprobació_fibonacci (Test)
=====
Traceback (most recent call last):
  File "/Users/juuto/Proyectos/GIT/practicaFibonacci/fibo/principal.py", line 17, in test_comprobació_fibonacci
    self.assertEqual (fibonacci(posicion), valor)
AssertionError: 3 != 2
=====
Ran 1 test in 9.775s
FAILED (failures=1)
juuto@MacBook-Air-de-David fibo %

```

At the bottom of the terminal, status information is displayed: Lín. 19, col. 1 Espacios: 4 UTF-8 LF Python 3.9.6 64-bit

En cambio, si le decimos que el valor en la quinta posición es dos, da error, $3 \neq 2$

Tras estas comprobaciones solo nos quedará responder a la pregunta final:

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?

Hemos realizado una prueba de aceptación, la cual ha verificado que el código se comporta según lo esperado. Se le dio un valor al programa y ha comprobado que el código calcula correctamente el segundo valor correspondiente.

[Volver al índice](#)

5. Bibliografía

- [Curso de programación sobre Python - programacionfacil.org](#)
- [Libro de GIT desde 0 - Uniwebsidad.com](#)
- [Videos da profesora Sabela Sobrino sobre GIT - Youtube](#)
- [Sucesión de Fibonacci - Wikipedia](#)
- [Sintaxis Markdown - Markdown.es](#)

- [Búsquedas puntuales en Google](#)

[Volver al índice](#)