

Licenciatura en Sistemas de Información

**Trabajo Práctico N°1**

**#Guía de la configuración**

**#Programación Python**

**Materia:** Ingeniería de Software II

**Docente:** Pedro Colla

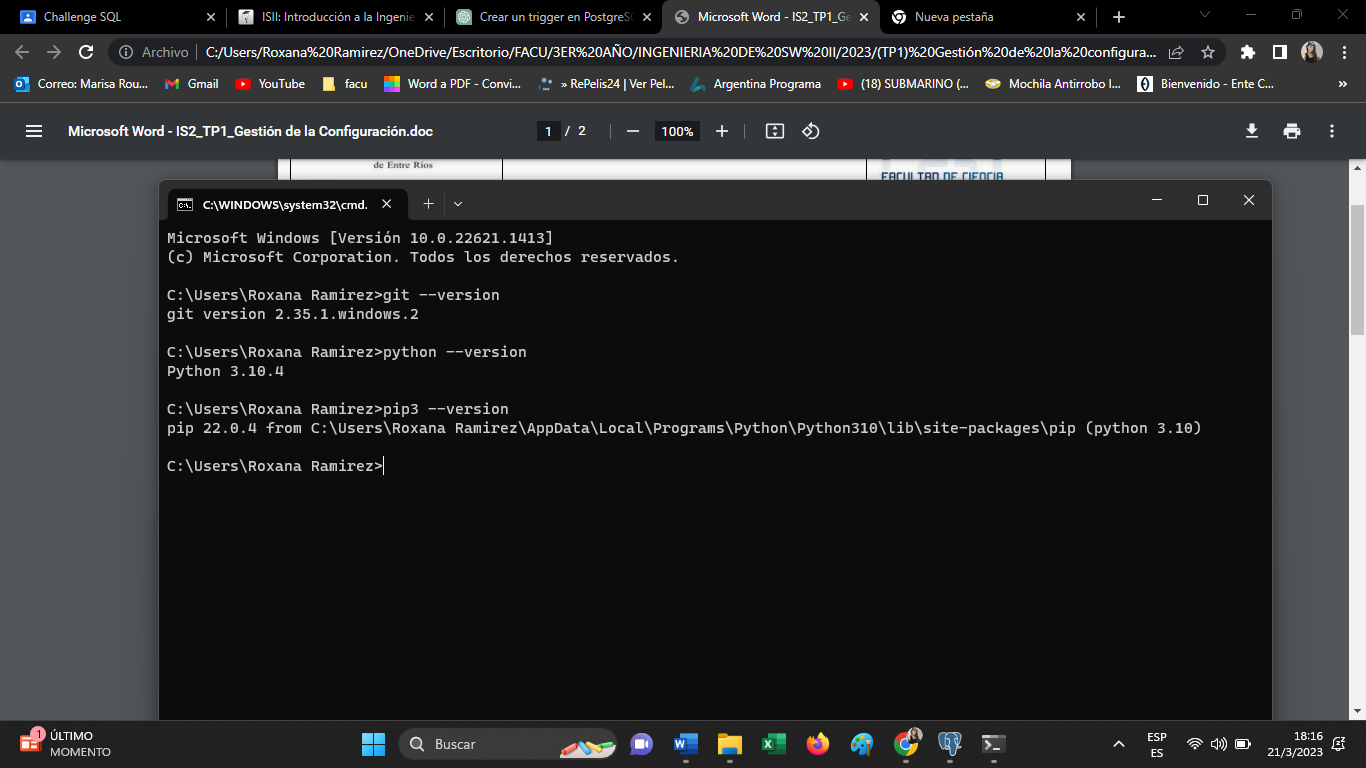
**#GESTION DE LA CONFIGURACIÓN**

**1. Instale los siguientes paquetes de software en la versión apropiada para el sistema operativo que utilice.**

**o Git.**

**o Python 3 (instalar desde python.org)**

**o Pip3 (instalar desde python.org)**



**2. Obtenga una cuenta en www.github.com y a la que llamará UADER\_IS2\_{su\_apellido}, a continuación genere una estructura de carpetas formada por:**

Cree el repositorio UADER\_IS2\_ROUDE en github.

o src

o doc

o bin

script

**3. Obtenga el programa primos.py (en Source Python.gz) y siga las siguientes consignas:**

**o Colóquelo en el directorio src local en su máquina.**

**o Ejecútelo con “python3 primos.py” y verifique que corre bien.**

**o Sincronícelo con el repositorio github.**

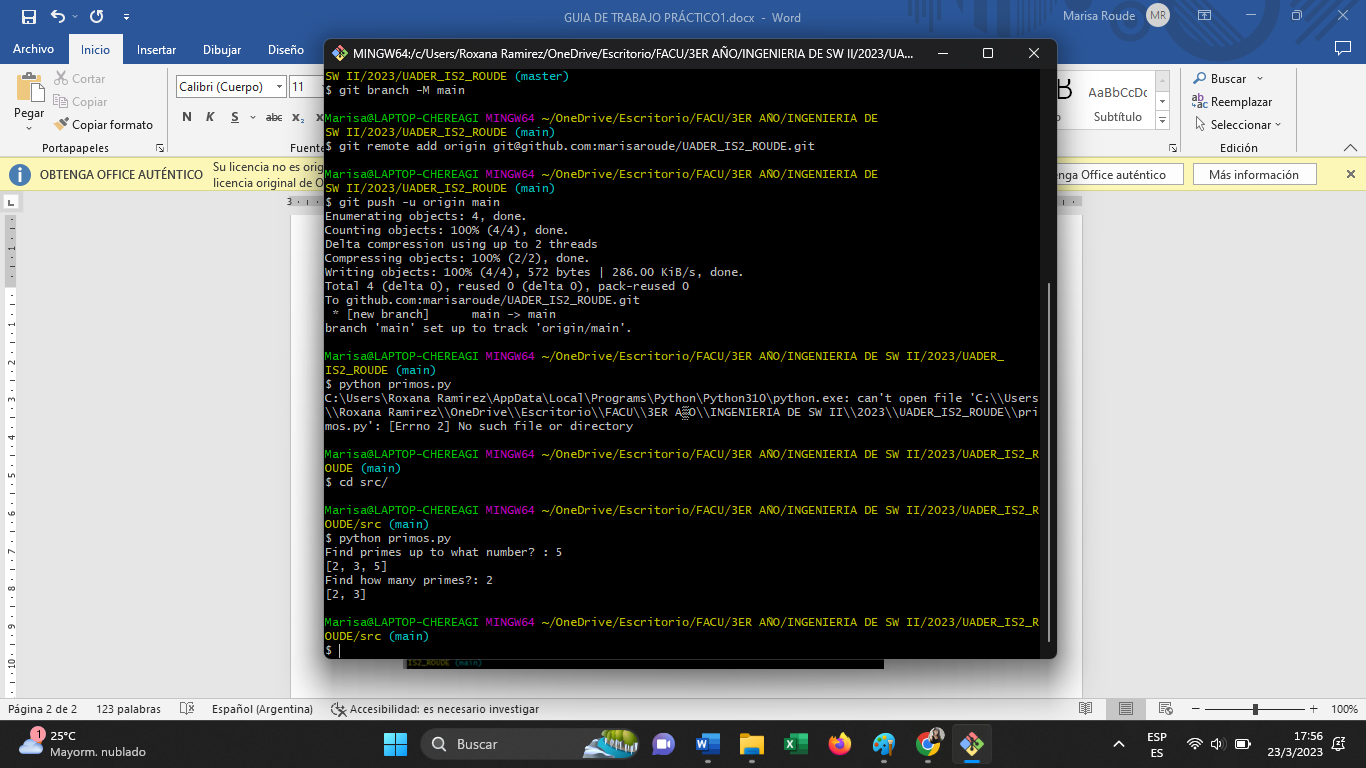
**§ git add .**

**§ git commit -n carga\_inicial**

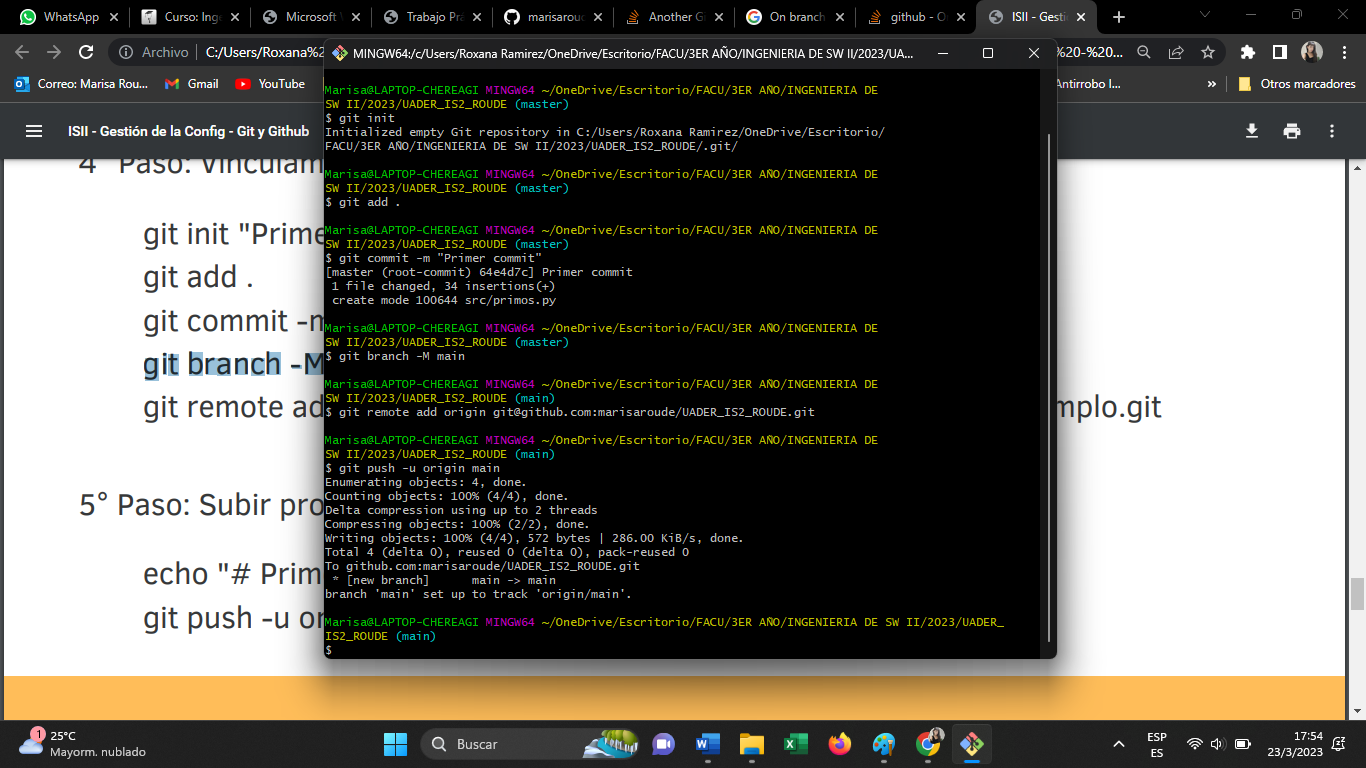
**§ git push origin**

**§ verifique la correcta actualización.**

Funcionamiento del programa primos.py

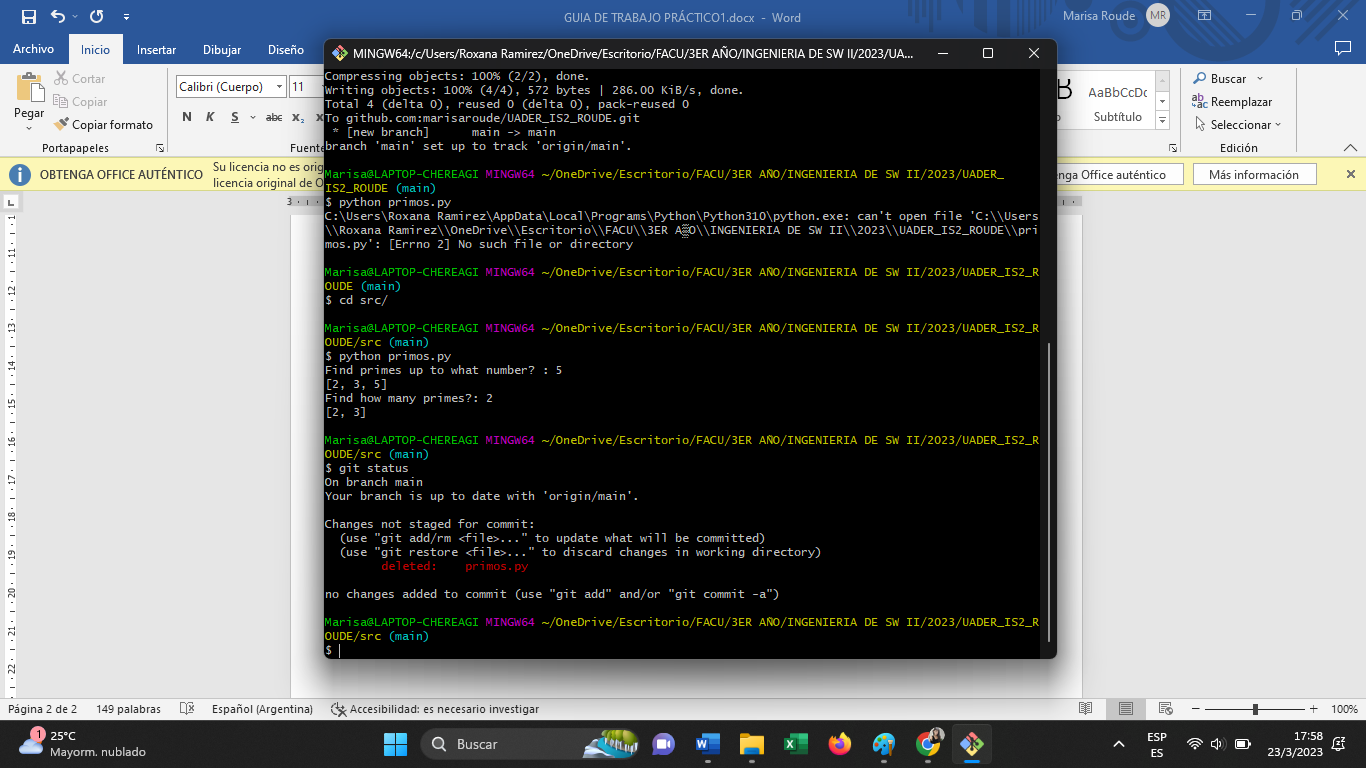


Push en el repositorio:

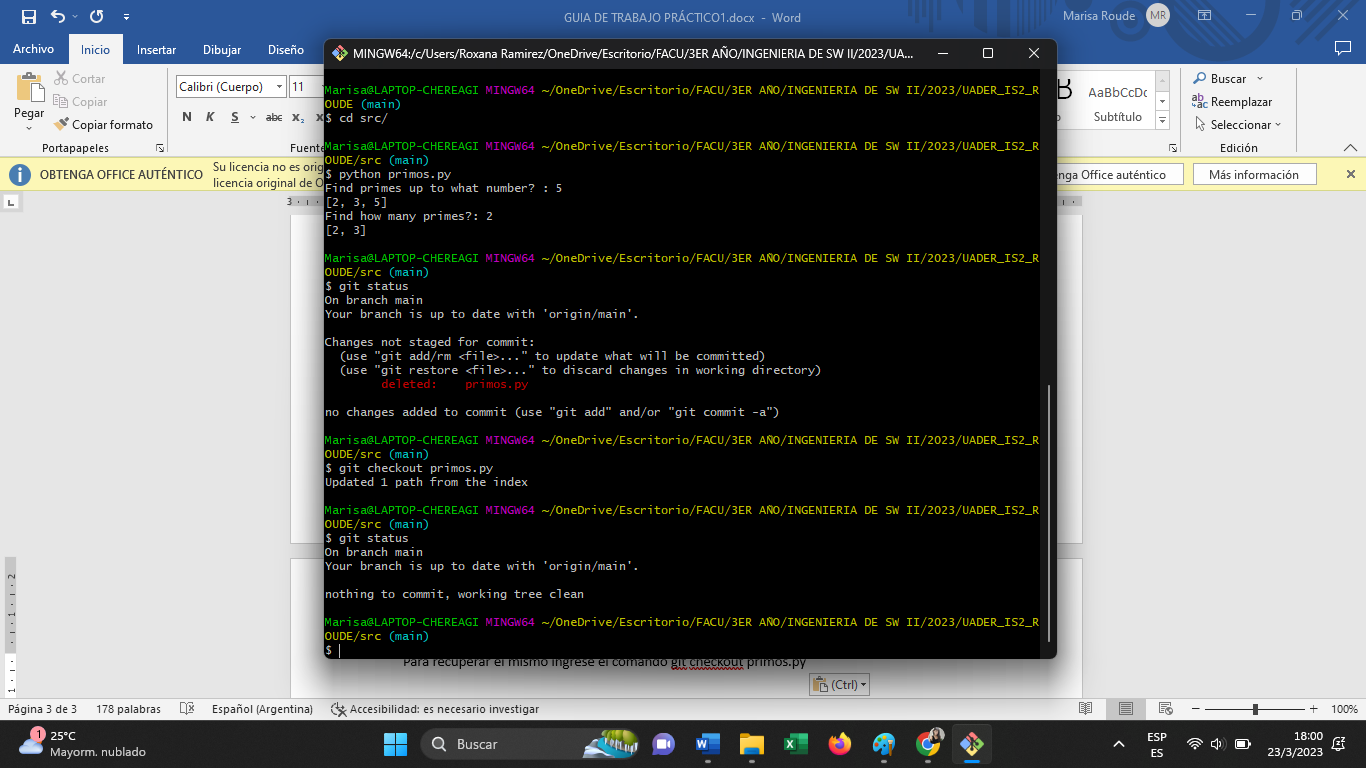


**o Simule el borrado “accidental” en su máquina y a continuación recupere el archivo desde el repositorio Github.**

Borré el archivo desde mi máquina y lo comprobé ingresando en la consola “git status” para ver los cambios

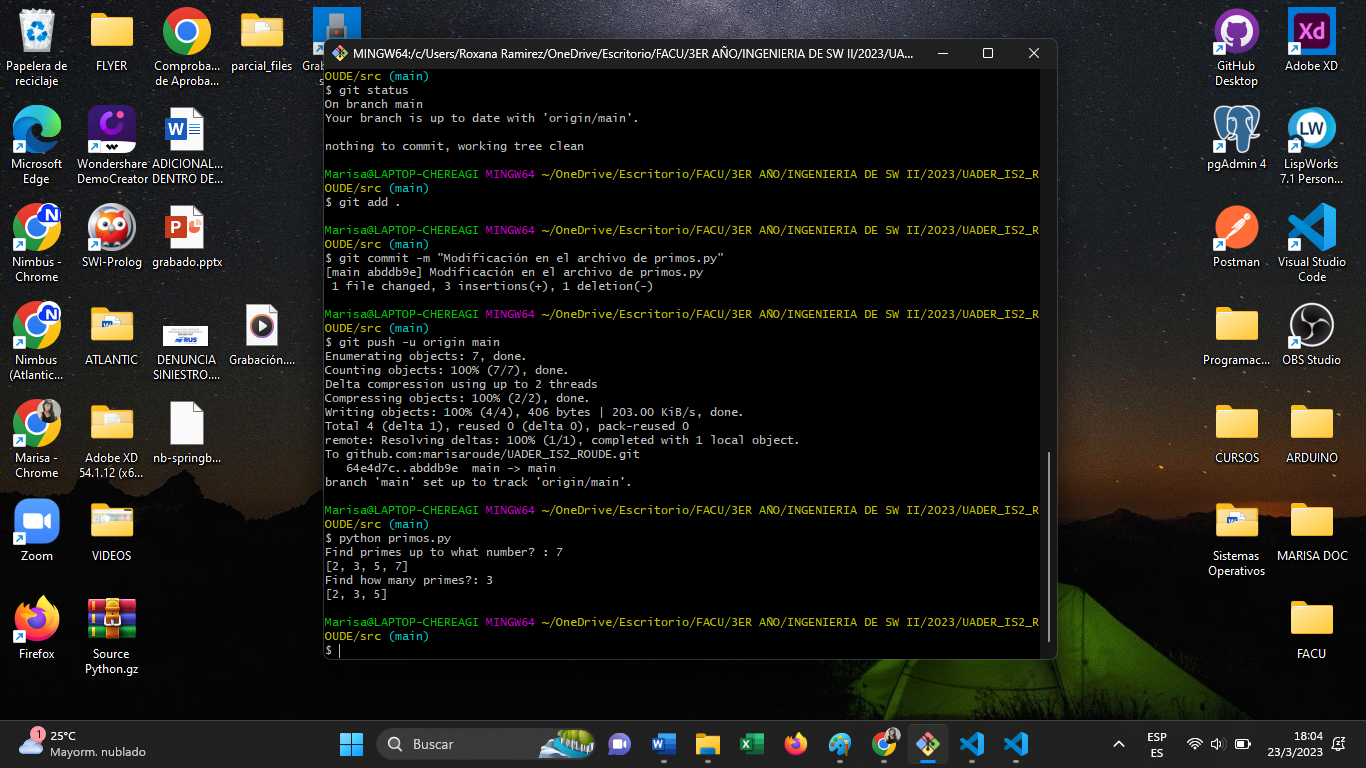


Para recuperar el mismo ingresé el comando git checkout primos.py



**Coloque comentarios al programa, al finalizar pruebe que el mismo siga ejecutando correctamente. Al hacerlo sincronice con el repositorio GitHub.**

Agregué un comentario al programa, lo guardé y luego lo subí al repositorio, al probarlo el archivo andaba perfecto.



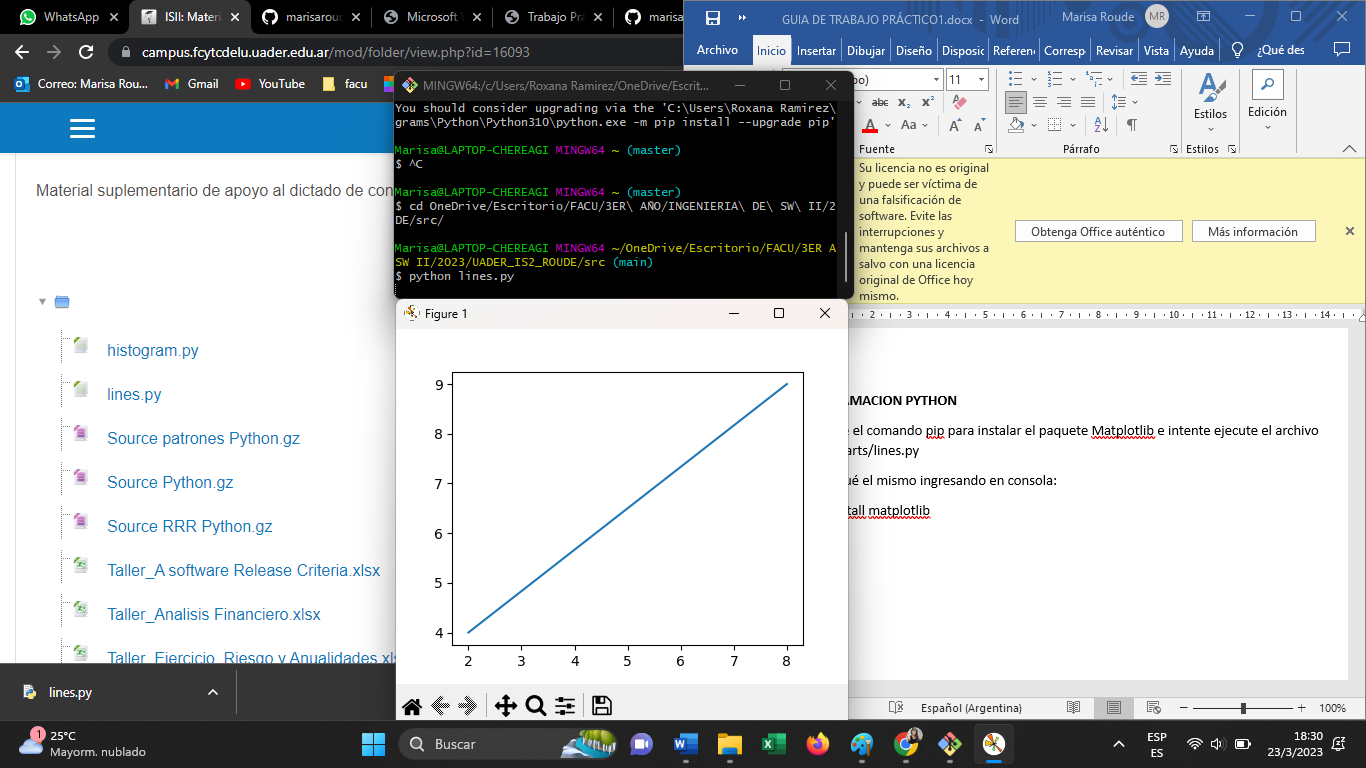
**#PROGRAMACION PYTHON**

**1. Utilice el comando pip para instalar el paquete Matplotlib e intente ejecute el archivo code/charts/lines.py**

Descargué el mismo ingresando en consola:

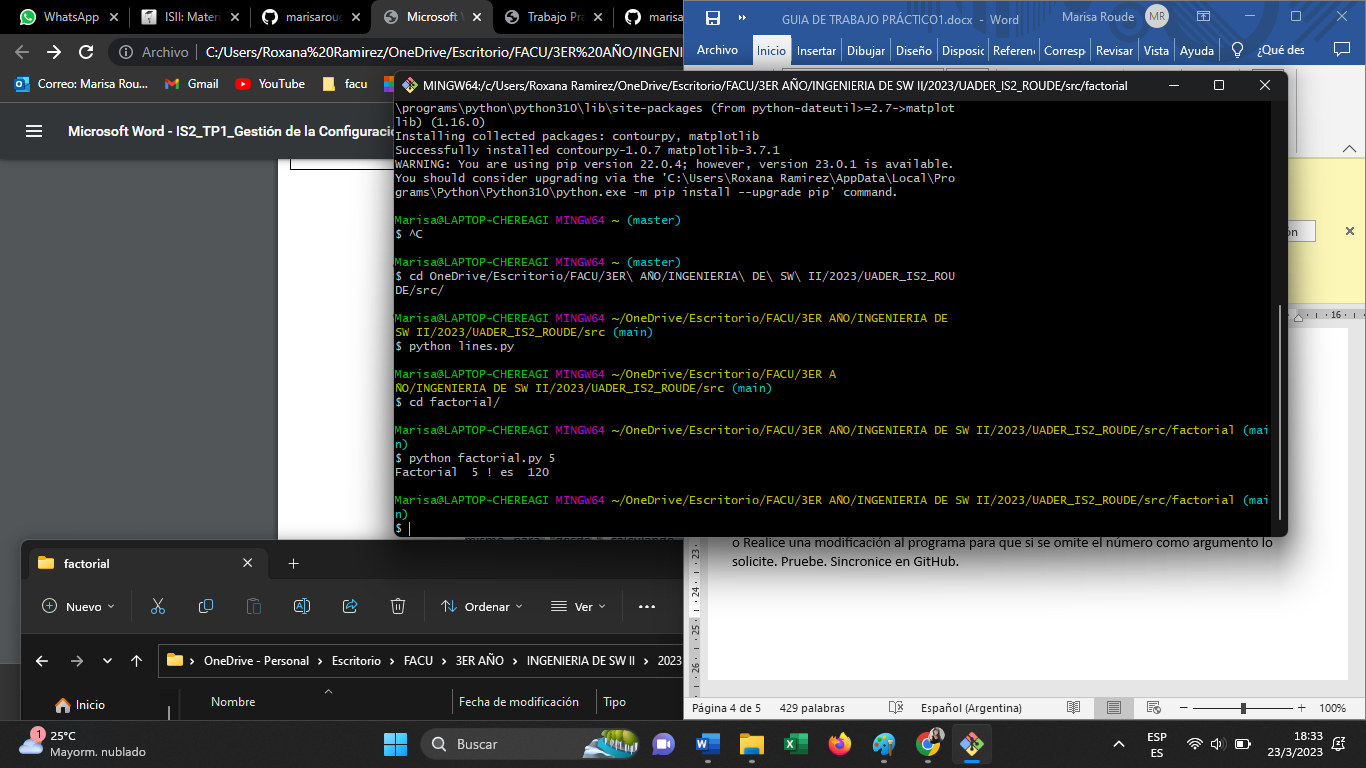
$ pip install matplotlib

Ejecute Python lines.py



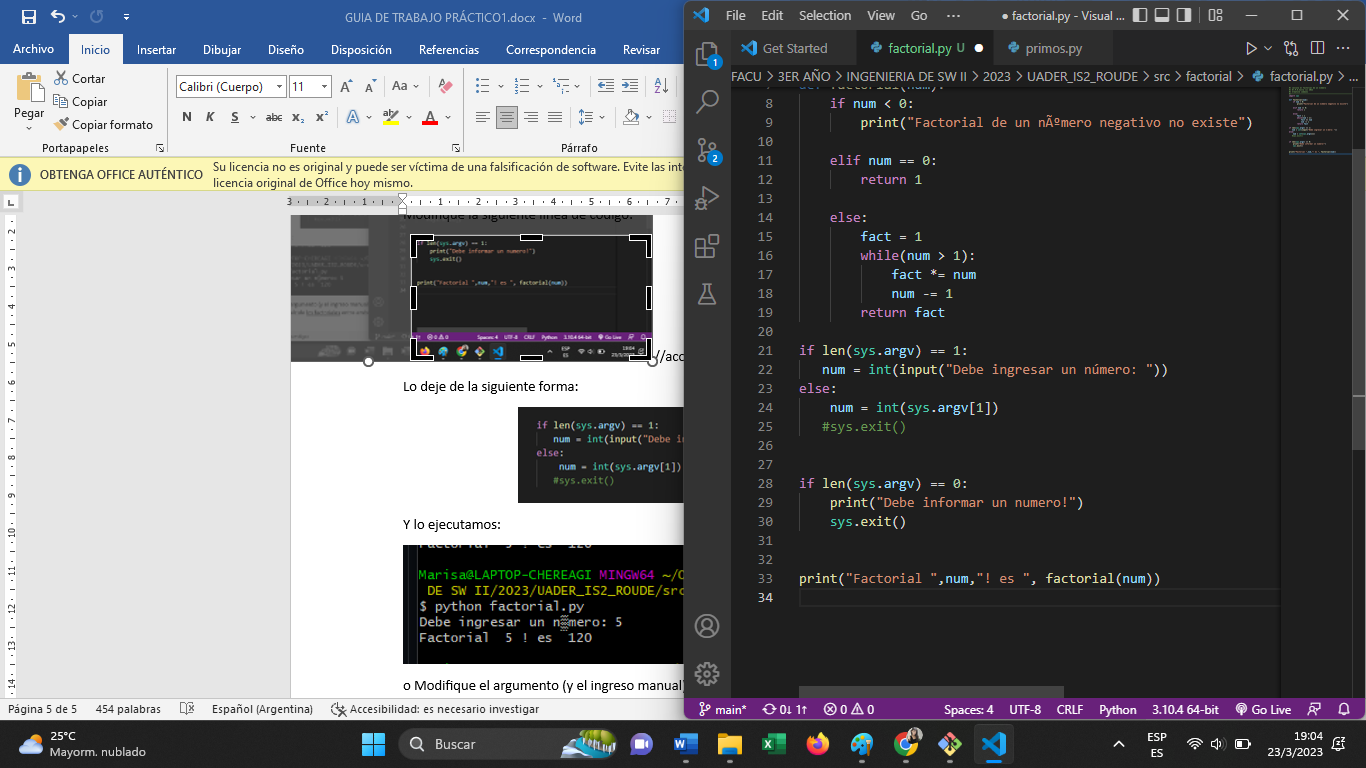
**2. Obtenga el programa fuente factorial.py y ejecute con python3 factorial 10 confirme que funciona correctamente. Guarde en repositorio GitHub en una carpeta específica dentro del árbol “src” denominada “factorial”.**

Cree una nueva carpeta llamada “Factorial” dentro de la carpeta de “src” y guardé el archivo factorial.py, luego de esto, lo ejecute:

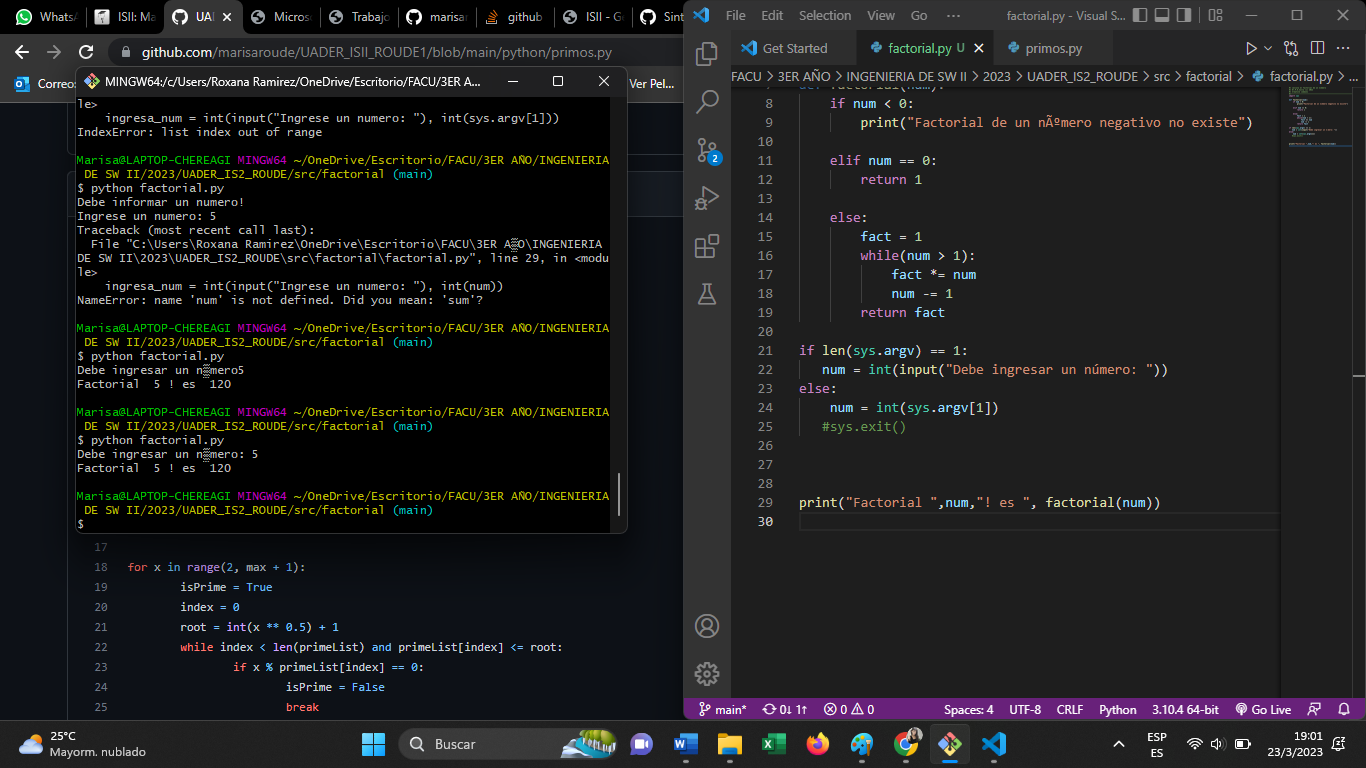


**o Realice una modificación al programa para que si se omite el número como argumento lo solicite. Pruebe. Sincronice en GitHub.**

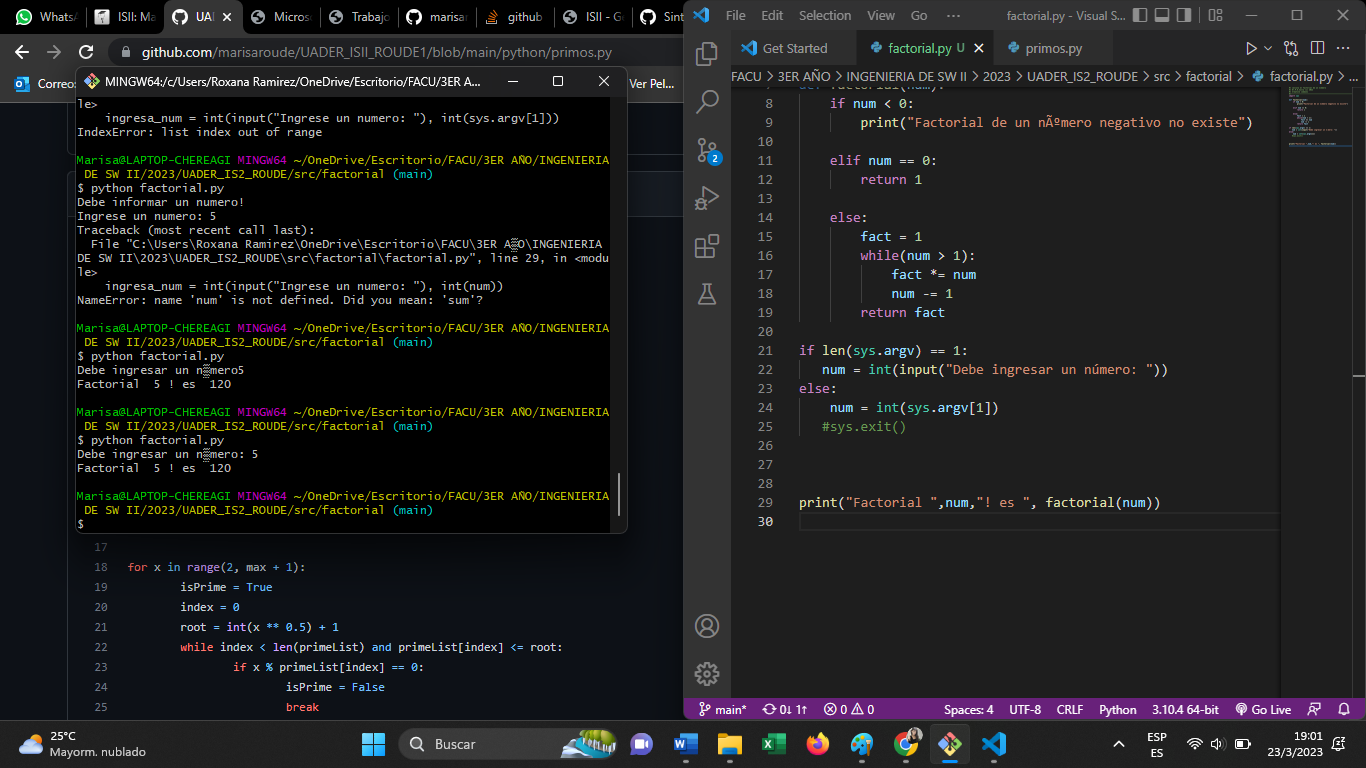
Modifiqué la siguiente línea de código:



Lo deje de la siguiente forma:



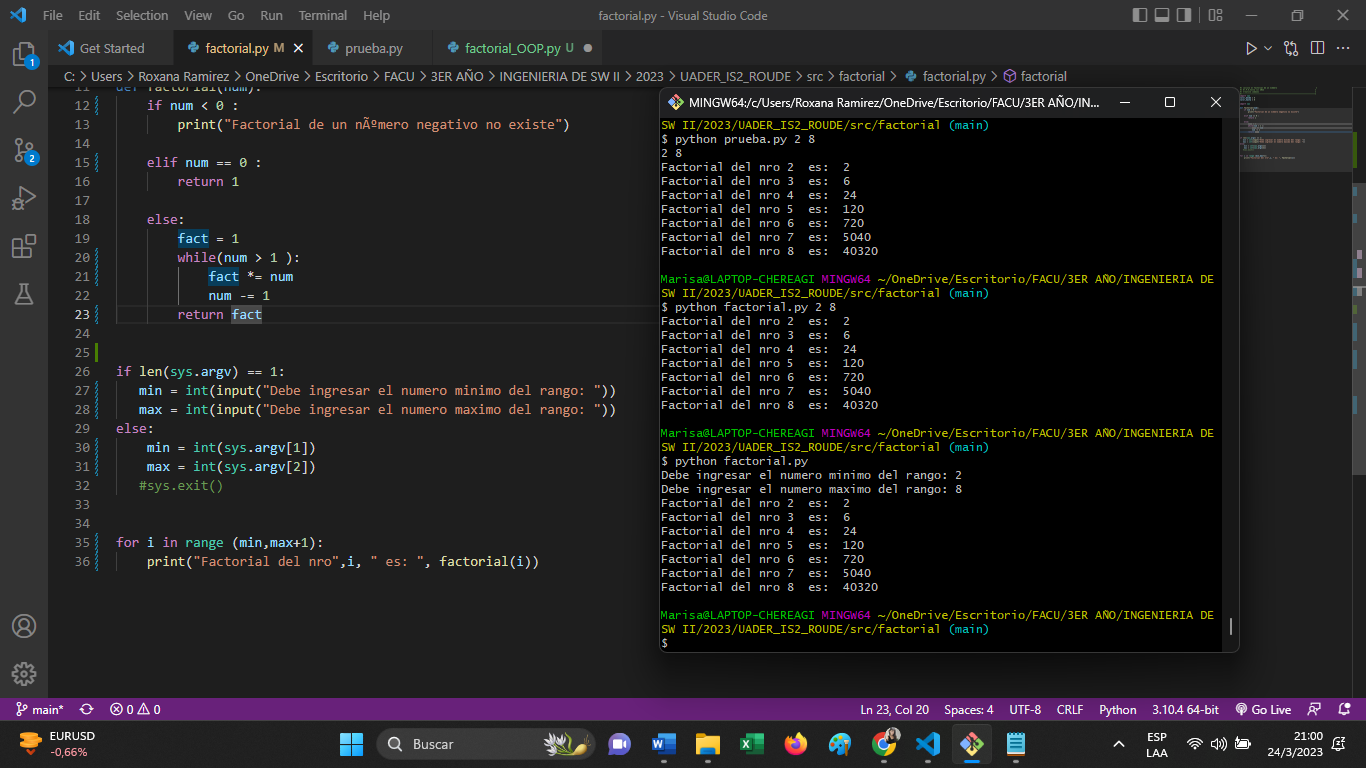
Lo ejecutamos:



Y luego lo sincronicé con el repositorio remoto.

**o Modifique el argumento (y el ingreso manual) para aceptar números en el rango desde-hasta (ej. 4-8) y que calcule los factoriales entre ambos extremos. Pruebe. Sincronice en GitHub.**

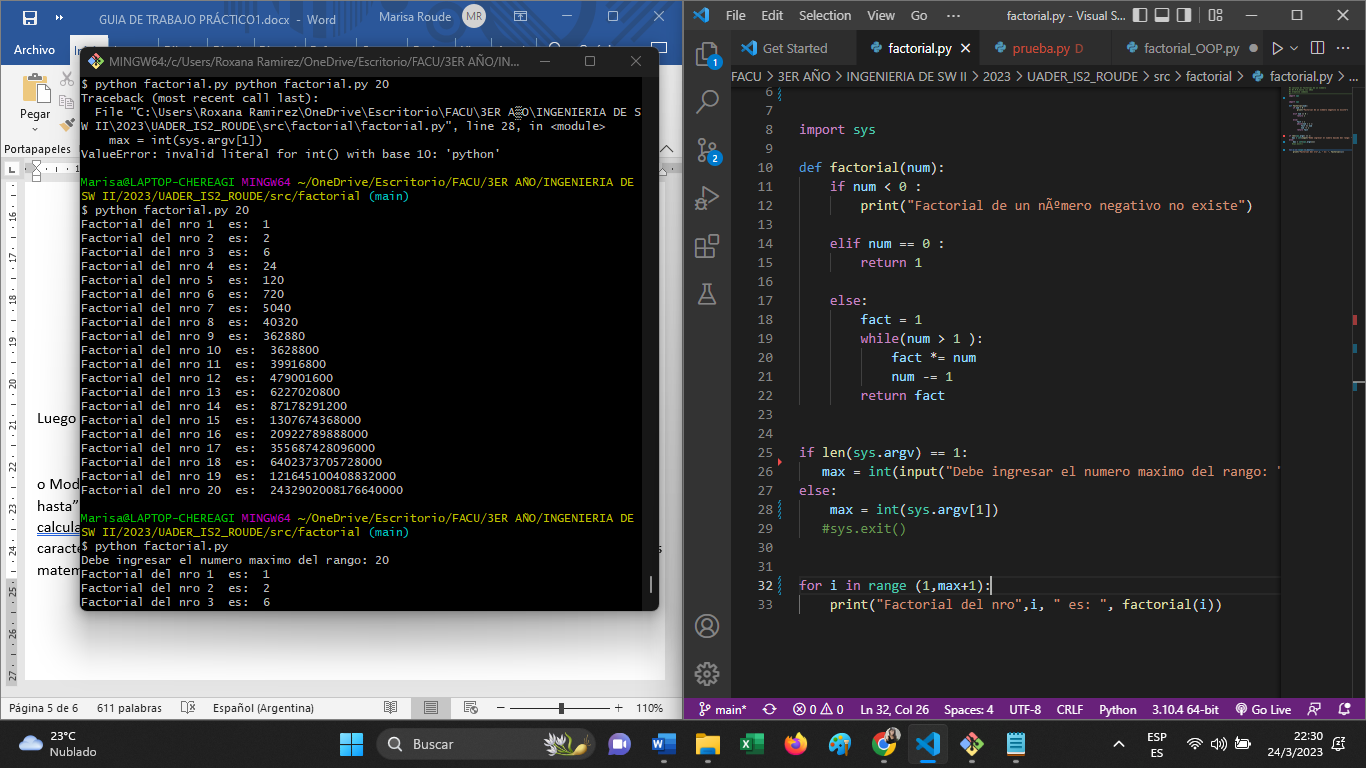
Modifiqué el código, y lo probé a través de la consola de dos maneras, una ingresando los dos números al llamar al programa y otra sin ingresar los números para que el programa me los solicite:



Luego lo sincronice en GitHub.

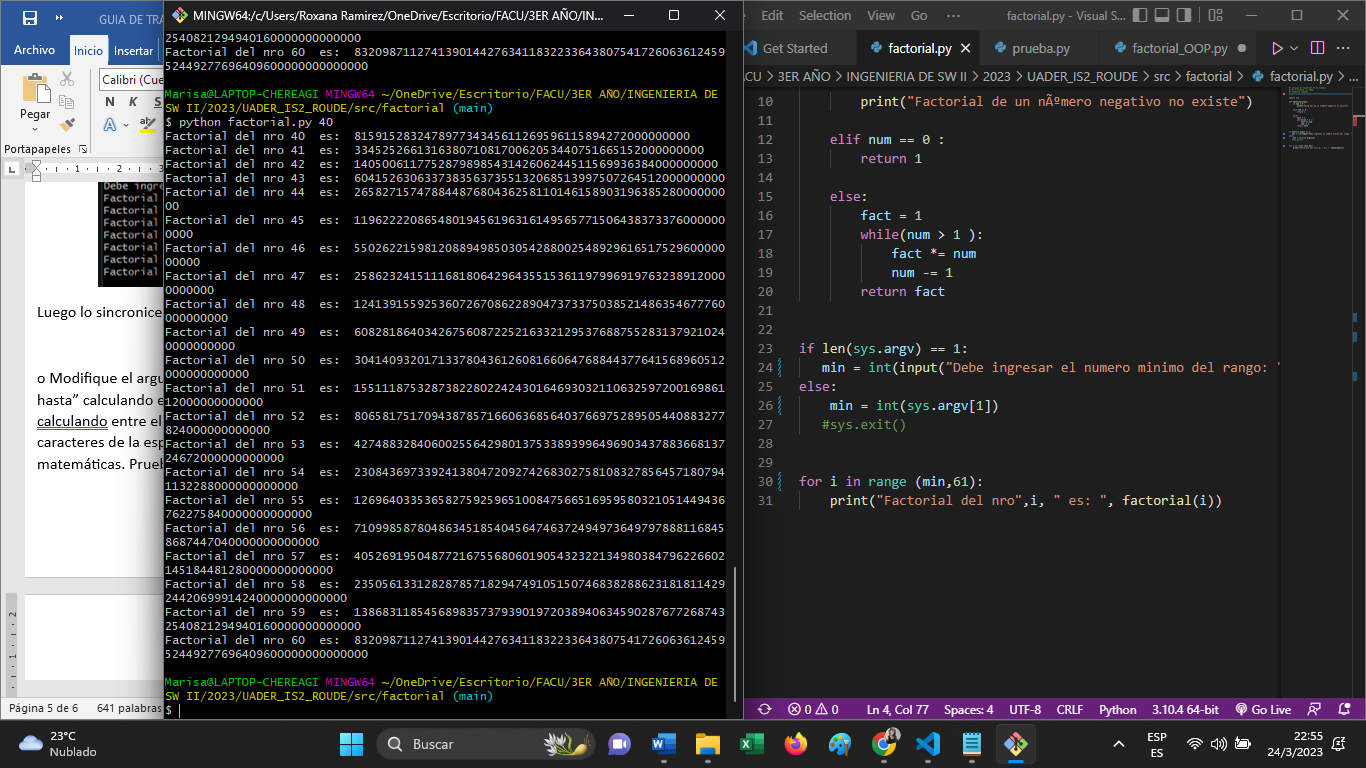
**o Modifique el argumento (y el ingreso manual) para que acepte rangos sin límite inferior “-hasta” calculando entre 1 y el número indicado (ejemplo “-10”), lo mismo para “desde-“ calculando entre el número indicado y 60. Tenga la precaución de transformar las cadenas de caracteres de la especificación de argumentos en valores enteros antes de intentar operaciones matemáticas. Pruebe. Sincronice en GitHub.**

Modifiqué el código para que acepte rangos sin límite inferior calculando entre 1 y cualquier número ingresado a través de consola:



Hice un push para que se guarde la versión.

Luego modifiqué el programa para que realice las factoriales a partir del número ingresado por consola hasta 60.



**o Agregue comentarios al código generado. Pruebe. Sincronice con GitHub**

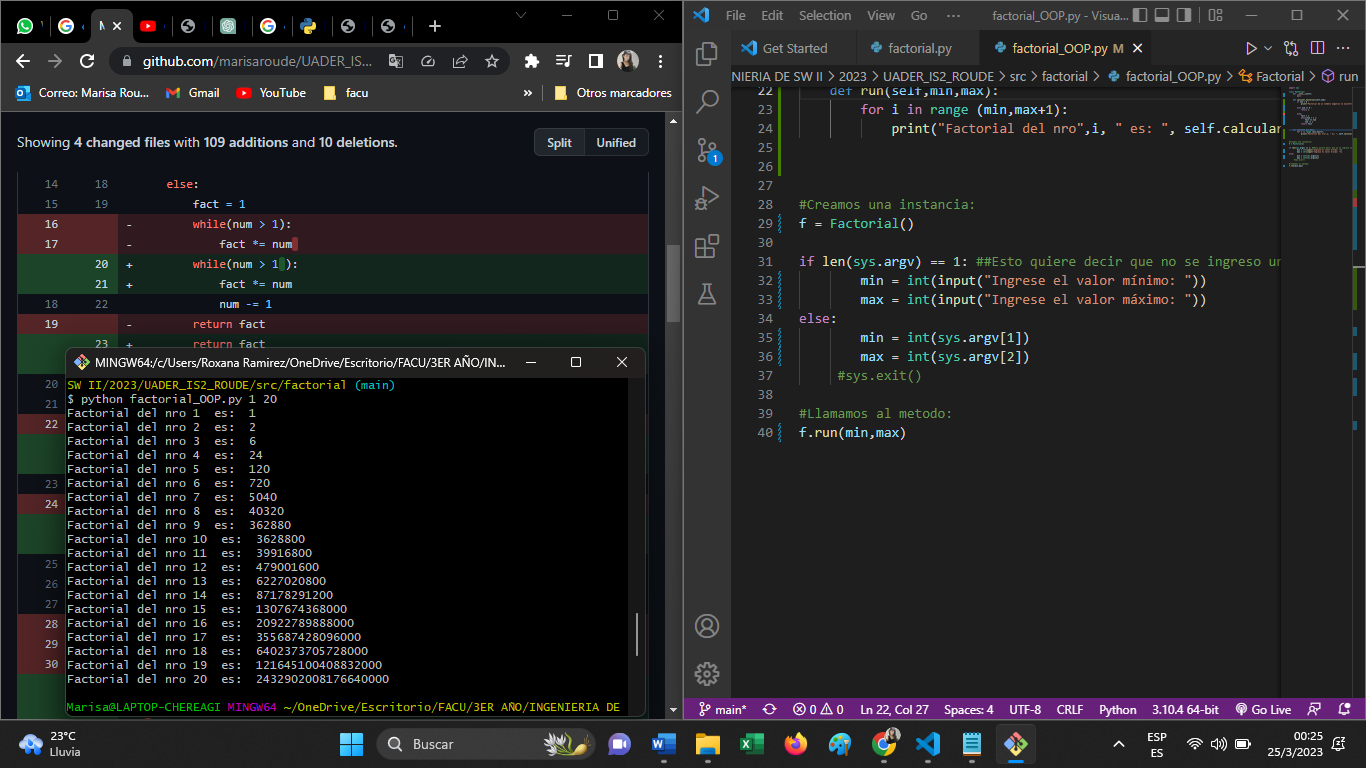
Agregué un comentario sobre el for y luego sincronicé

**3. Genere un proyecto copia del anterior denominado “factorial\_OOP” donde tomando como base el programa “factorial.py” genere un programa “factorial\_OOP.py” donde se construya la lógica de cálculo de factorial mediante una clase Factorial con un constructor y un método “run(min,max)” que calcule como resultado el factorial entre los números min y max. Pruebe. Sincronice en GitHub.**

Cree un archivo llamado factorial\_OOP.py. En él cree una clase “Factorial” con tres métodos: “\_\_init\_\_()”, “calcular\_factorial()” y “run”().

Cree una instancia de la clase, luego corroboré si se habían ingresado valores y por último llamamos al método run() con los parámetros min y max.

Lo probé a través de la consola y luego lo sincronicé con GitHub.



**4. Desarrolle un programa en python para calcular el número de Collatz (conjetura 2n+1) para los números entre 1 y 10000, realice un gráfico donde en el eje de órdenadas muestre el número n de comienzo de la secuencia y en la absisas el número de iteraciones que tardó en converger a una secuencia repetitiva. Coloque en una carpeta en la jerarquía “src”. Pruebe. Sincronice en GitHub.**

Cree un archivo en la carpeta src llamado collatz.py, en él cree una función de collatz y utilicé matplotlib para crear el gráfico obteniendo x e y, siendo X los n números de 1 a 10000 y siendo y el número de iteraciones.

Probé a través de consola y sincronicé en GitHub.