**TRABAJO PRÁCTICO N°1 - ING. SOFTWARE II**

**ALUMNO**: FERNANDO PÉREZ

**DOCENTES**: Dr. Pedro E. Colla y Hernán Sanchez

**UADER-FCYT**

**1)** Obtenga una cuenta en www.github.comy a laque llamará UADER\_IS2\_{su\_apellido}, a continuación genere una estructura de carpetas formada por:

* src
* doc
* bin
* script

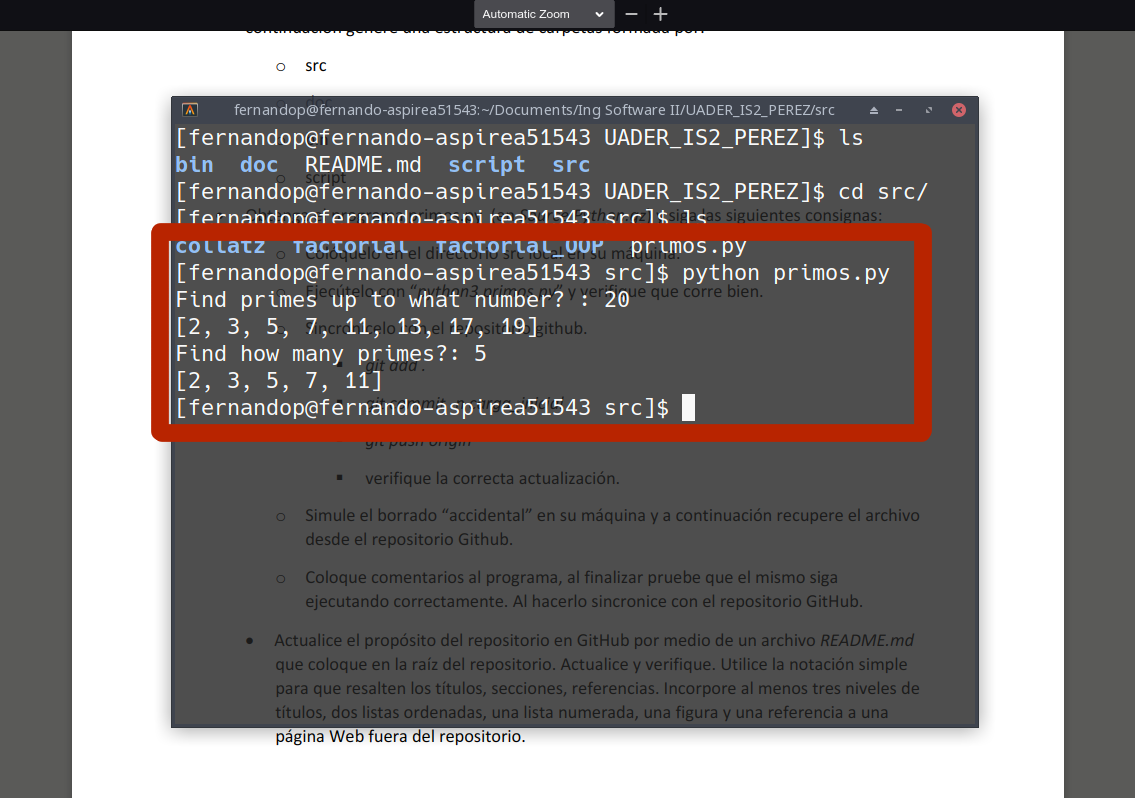
**1\_** Se agregó la estructura de carpetas solicitadas como se ve a continuación:



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

* Colóqueloen el directorio src local en su máquina.
* Ejecútelo con “python3primos.py”y verifique que corre bien.
* Sincronícelo con el repositorio github.
* §git add .
* §git commit -m carga\_inicial
* §git push origin
* §verifique la correcta actualización.

**2\_** Se obtuvo el programa “primos.py” de “Source Python.gz”, se colocó en el repositorio local y se verificó que funcionara correctamente. luego de eso se subió al repositorio de github, a continuación se muestra su correcto funcionamiento:



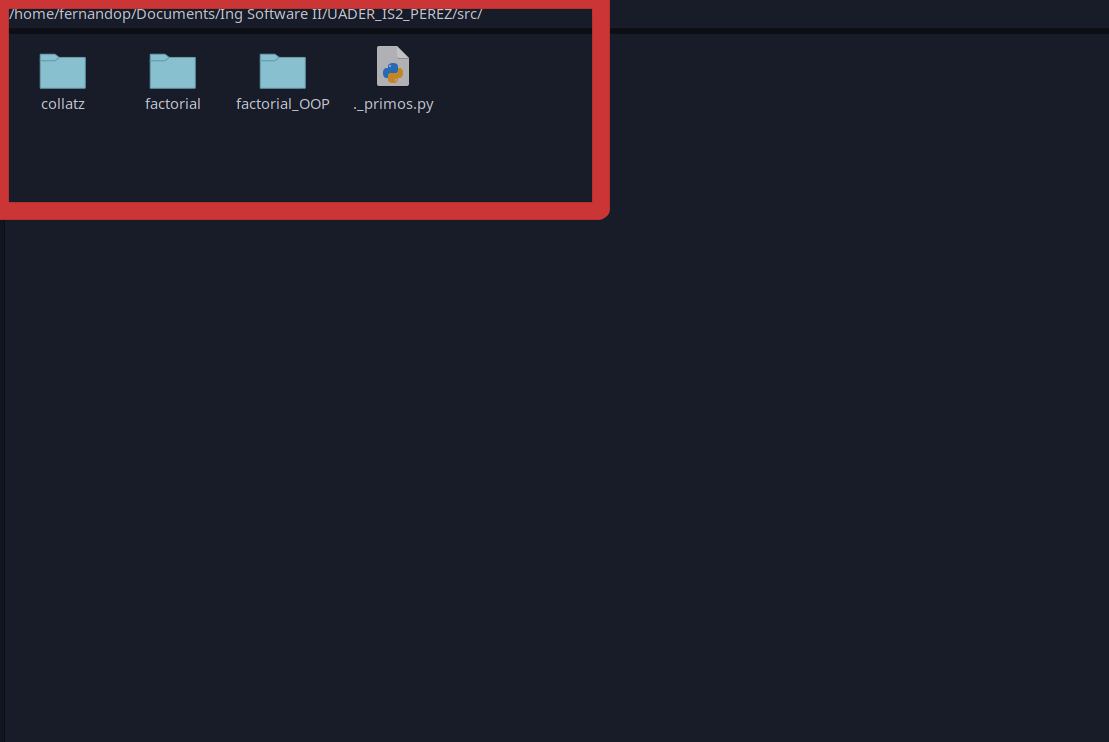
Link para verificar la correcta actualización del repositorio en github:

**GITHUB REPO:** <https://urlis.net/carga-inicial>

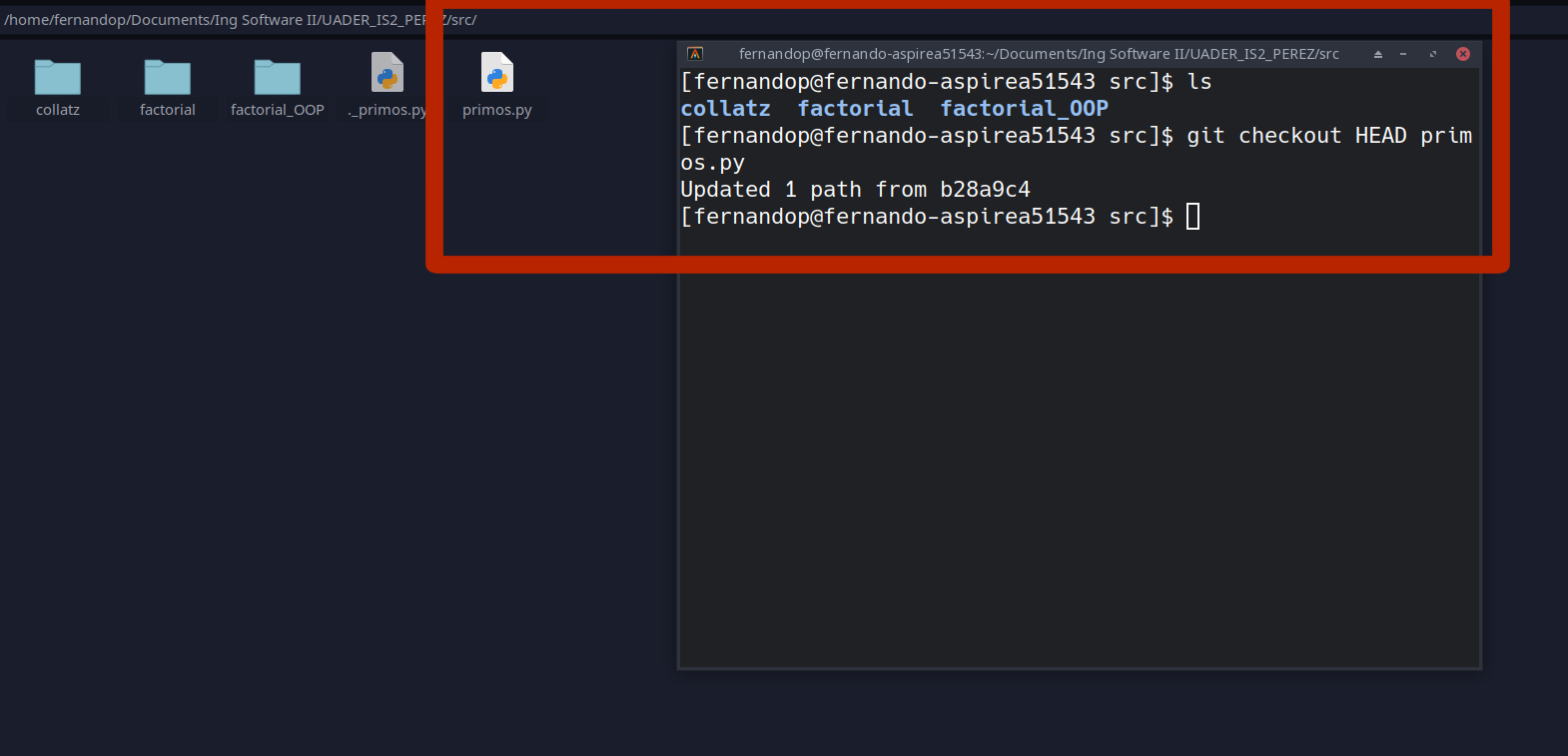
**3)** Simule el borrado “accidental”en su máquina y a continuación recupere el archivo desde el repositorio Github.

Se simula el borrado accidental en la maquina y despues se recupera desde el repositorio de github como se muestra a continuación:

El archivo principal “primos.py” fue eliminado de la máquina como se muestra en la imagen:

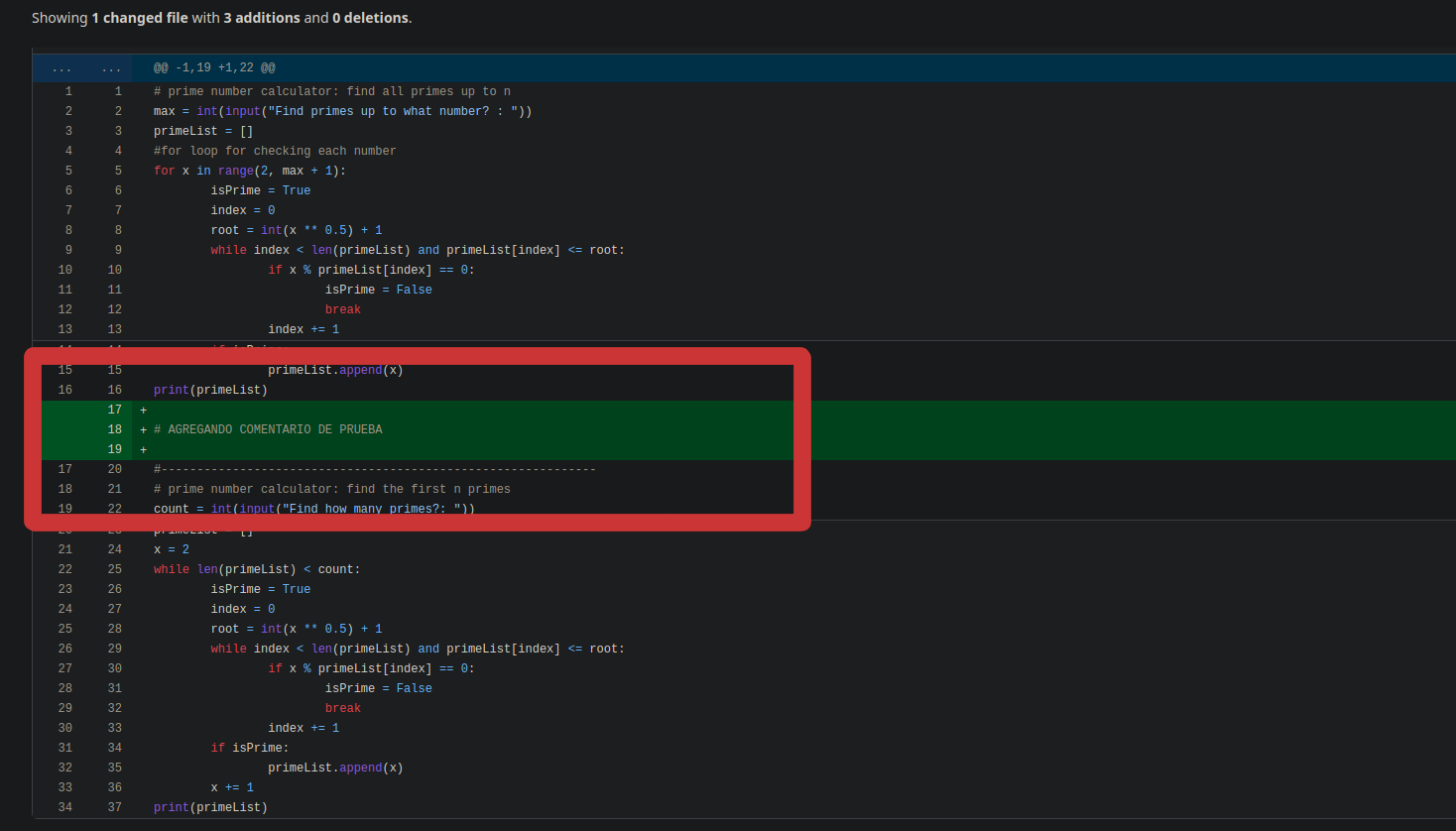


Ahora en la imagen a continuación se muestra al archivo “primos.py” recuperado con git desde el repositorio de github:



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

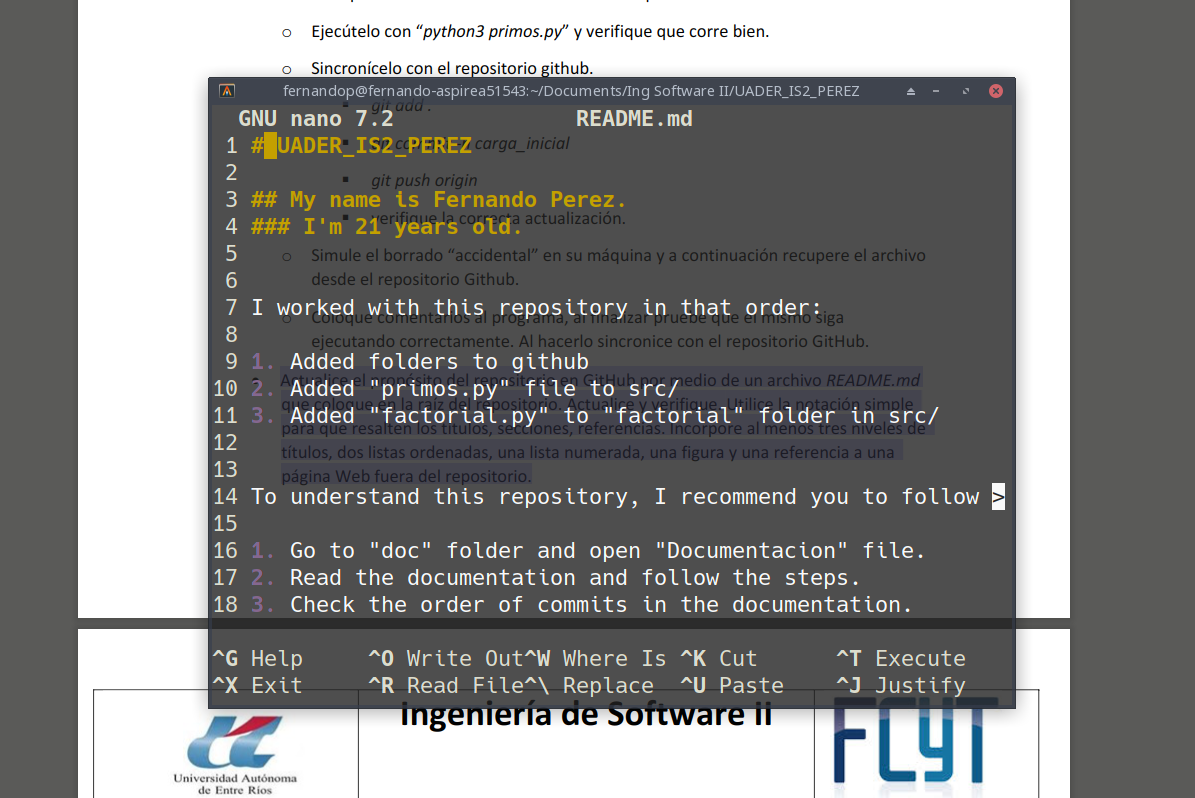
**4\_** Se colocaron comentarios al programa y se provó su funcionamiento, luego se actualizó el repositorio en github, adjunto imagen:



**GITHUB REPO:** <https://urlis.net/primos-comment>

**5)** Actualice el propósito del repositorio en GitHubpor mediode un archivoREADME.mdque coloque en la raíz del repositorio. Actualice y verifique.Utilice la notación simple para que resalten los títulos, secciones, referencias.Incorpore al menos tres nivelesde títulos, dos listas ordenadas, una listanumerada, una figura y una referencia a una página Web fuera del repositorio.

**5\_** Se utlizó en la consola el comando “nano README.md” para editar el archvivo “readme.md”,  como se muestra a continuación y luego se subió al repositorio en github:



**GITHUB REPO:** <https://urlis.net/readme-updt>

**6)** 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”.

**6\_** Se obtuvo el programa fuente “factorial.py” y se comprobó su correcto funcionamiento como se muestra a continuación, luego se subió al repositorio en github:



**GITHUB REPO:** <https://urlis.net/factorial-added>

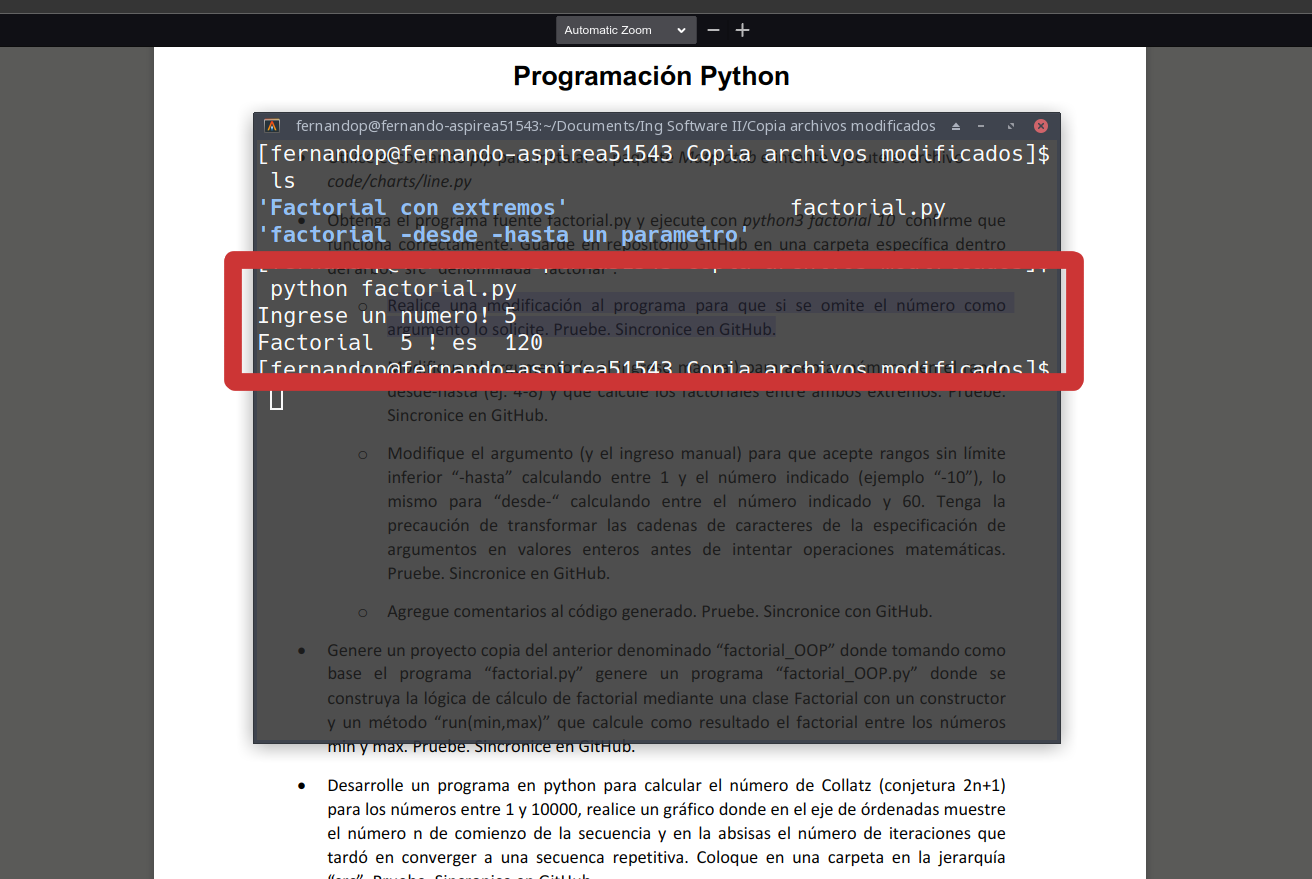
**7)** Realice una modificación al programa para que si se omite el número como argumento lo solicite. Pruebe. Sincronice en GitHub.

**7\_** Se realizó la modificación correspondiente ya que antes fallaba al intentar introducir un número y ahora ese error está contemplado, evitando la interrupción de la ejecución como se ve a continuación:

**ERROR:**



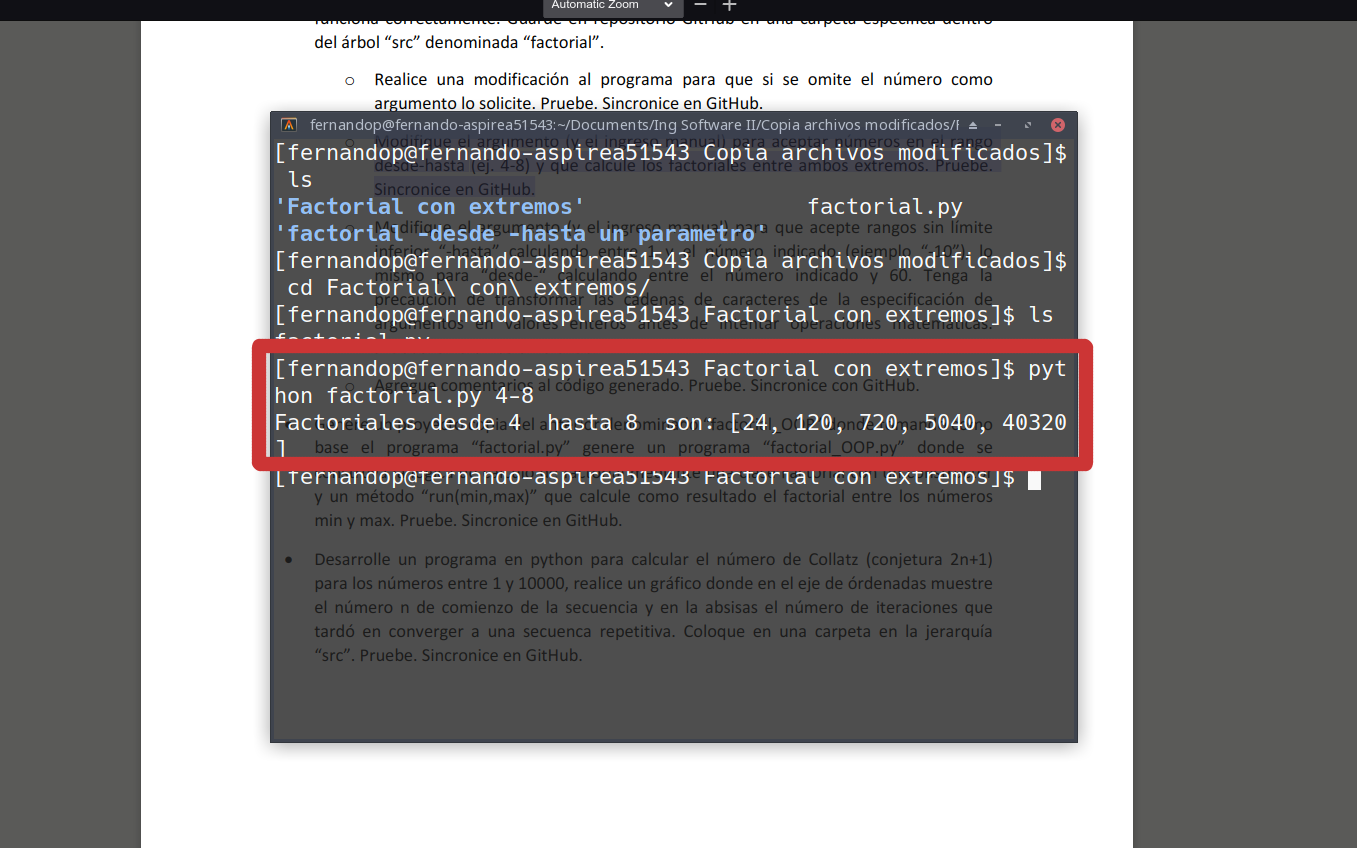
**SOLUCIÓN:**



**GITHUB REPO:** <https://urlis.net/repository>

**8)** Modifique el argumento (yel 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.

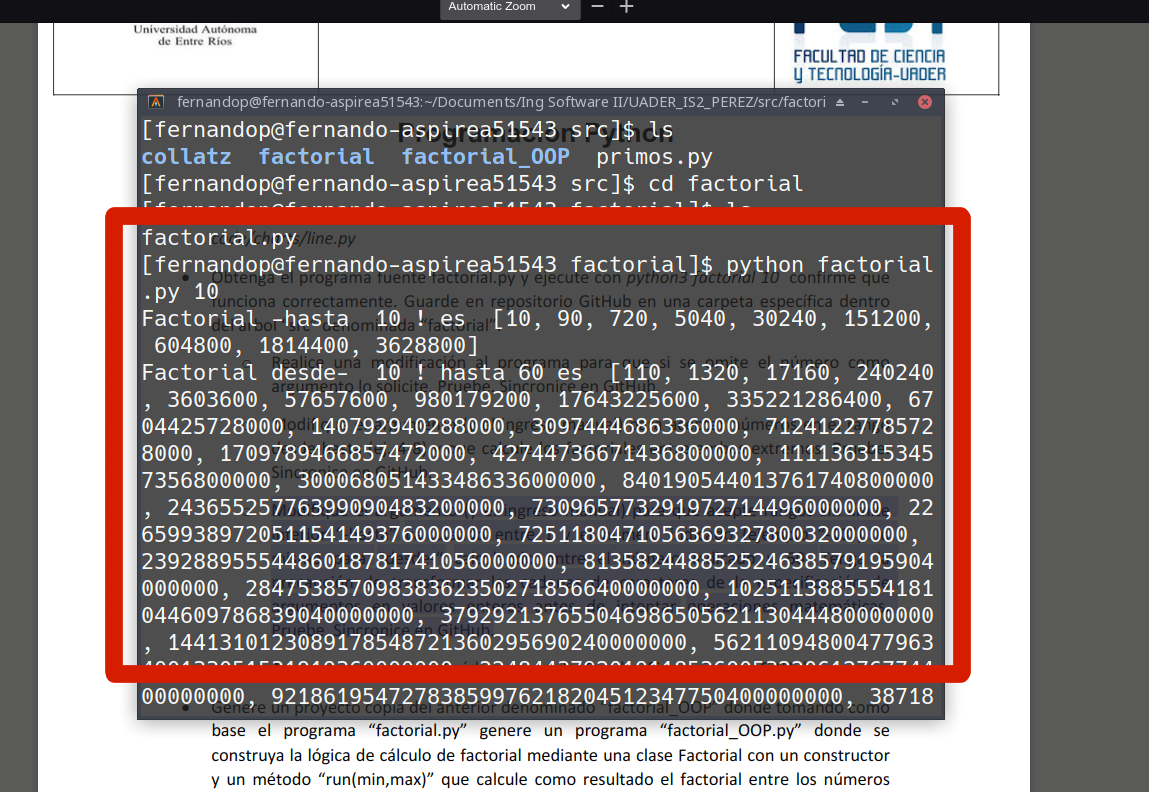
**8\_** Se modificó el ingreso manual y ahora calcula el factorial de lado a lado incluyendo los extremos definidos como se ve a continuación:



**GITHUB REPO:** <https://urlis.net/fact-extremes>

**9)** 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 intentaroperaciones matemáticas. Pruebe. Sincronice en GitHub.

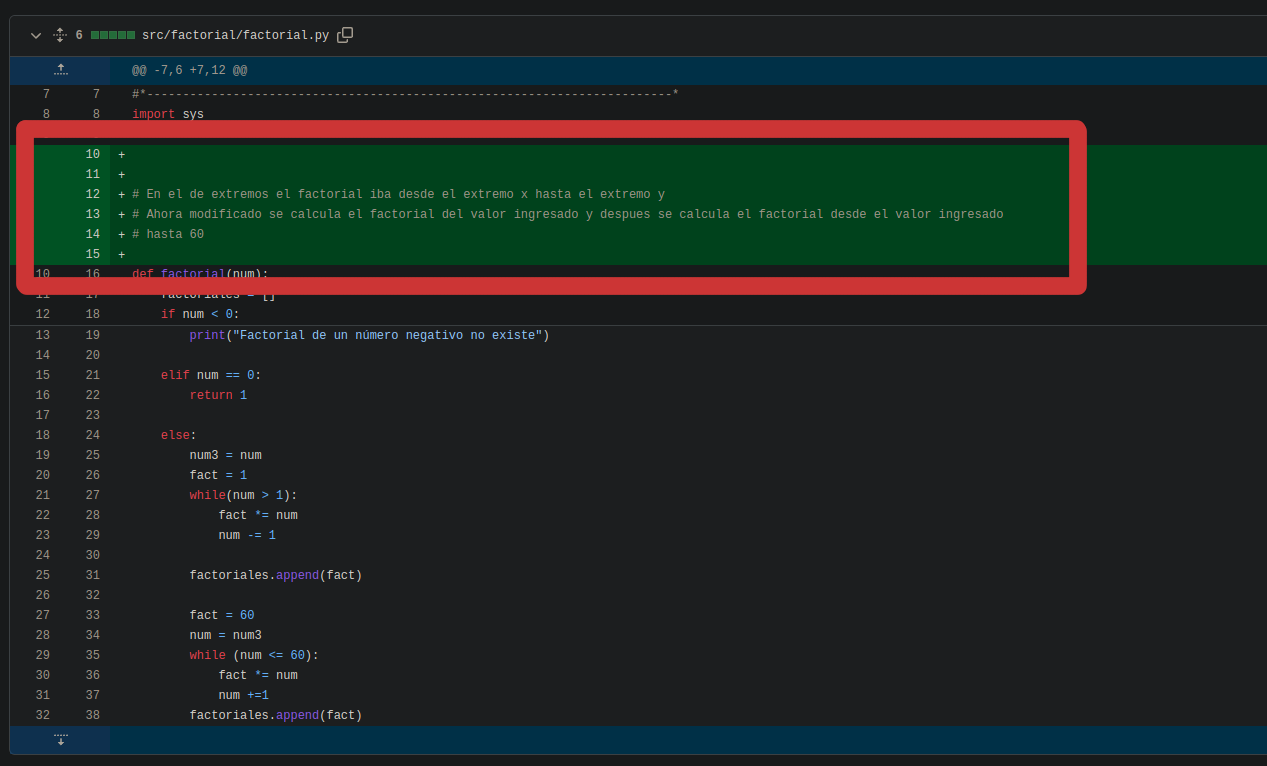
**9\_** Se modifica el argumento y el ingreso manual, y a partir de ahora, al número ingresado se calcula losl factoriales, hasta el número ingresado anteriormente y se muestra. Luego se calcula desde el valor ingresado los factoriales hasta 60, como se muestra a continuación (solo se muestra un parte porque hasta 60 el vector es muy largo):



**GITHUB REPO:** [https://urlis.net/-desde-hasta](https://urlis.net/fact-desd-hast)

**10)** Agregue comentarios al código generado. Pruebe. Sincronice con GitHub.

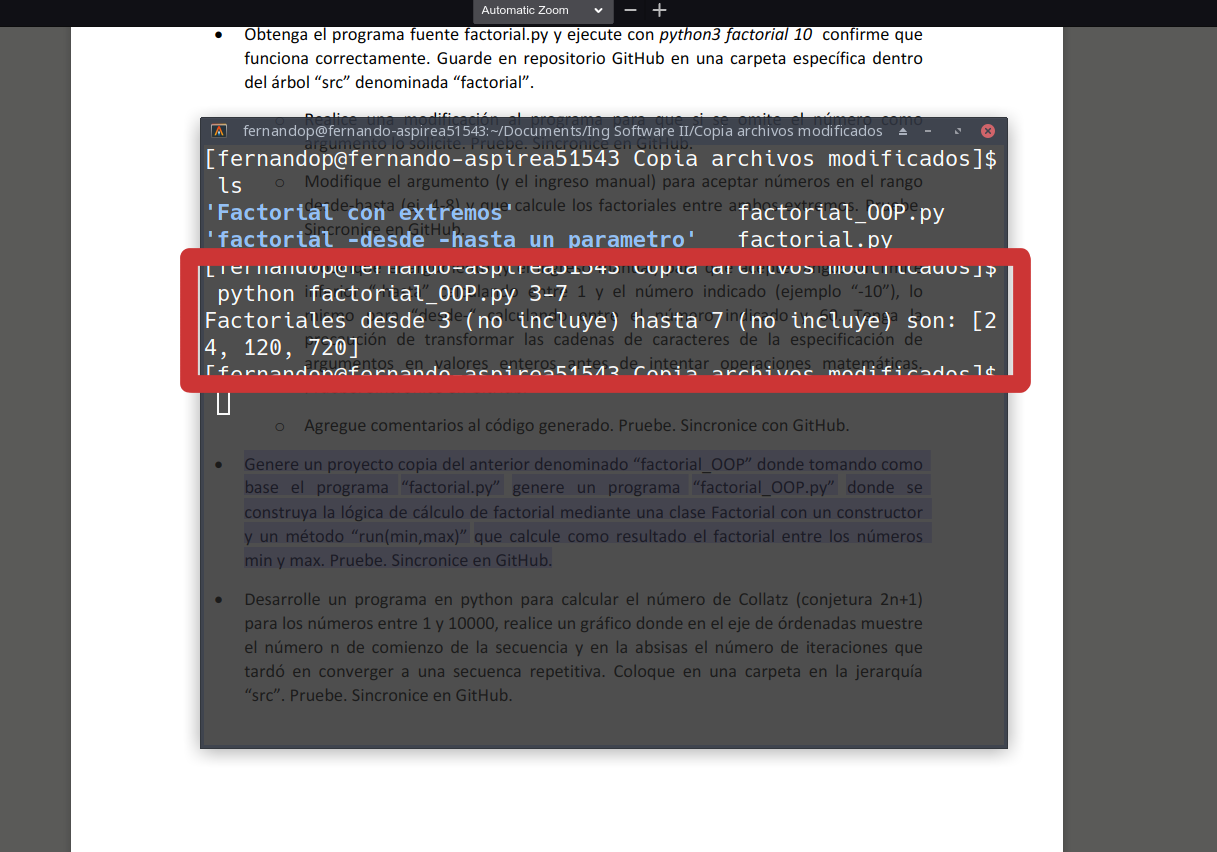
**10\_** Se ha agregados comentarios al ejercicio como se muestra a continuación:



**GITHUB REPO:** <https://urlis.net/d-h-comment>

**11)** 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.

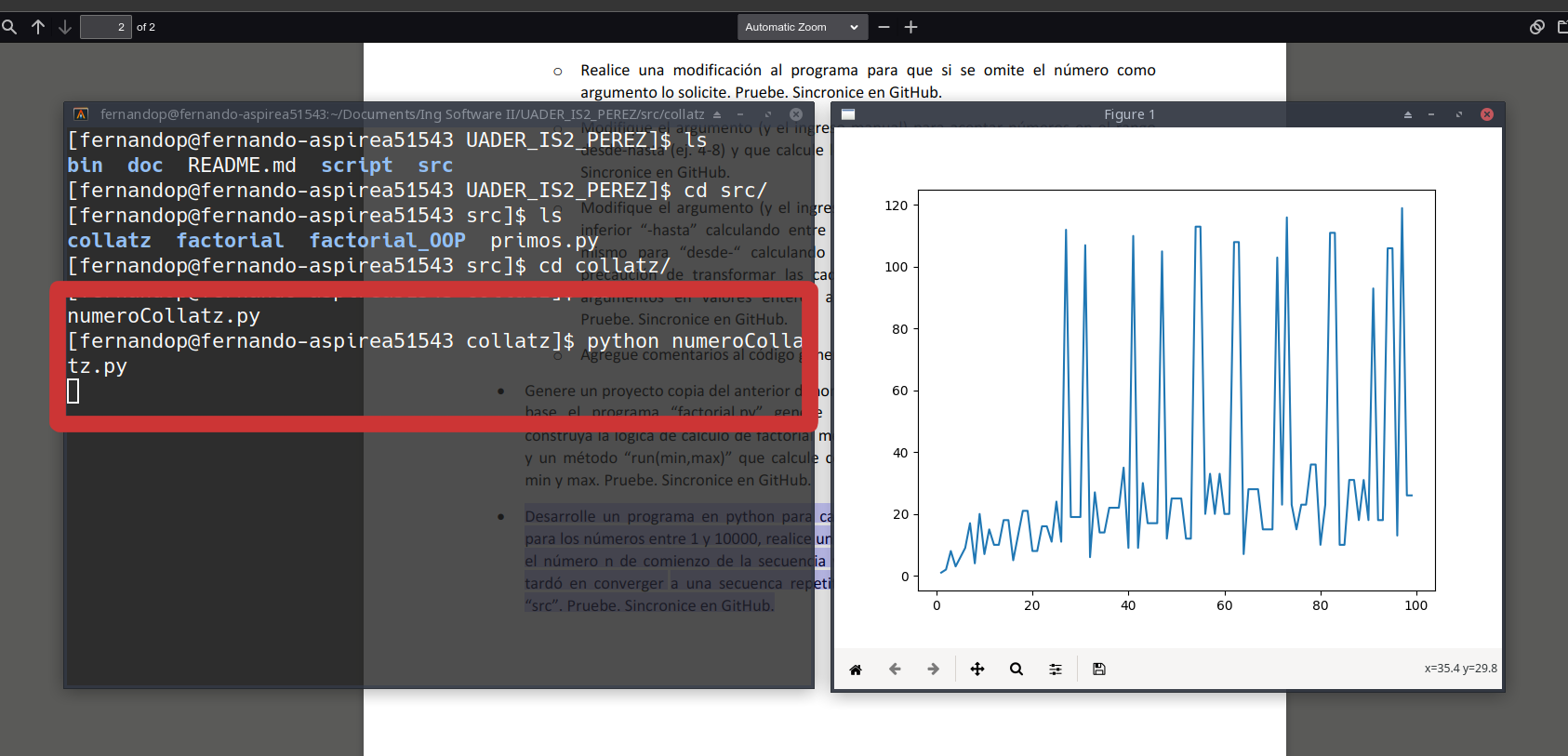
**11\_**  Se generó la carpeta “factorial\_OOP” con el archivo dentro “factorial\_OOP.py” que tiene de base el código del archivo “factorial.py” en la cual se introduce una clase Factorial con un constructor y un método que calcule factorial entre dos numeros (sin incluir los extremos) que se pasen como parámetro como se muestra a continuación:



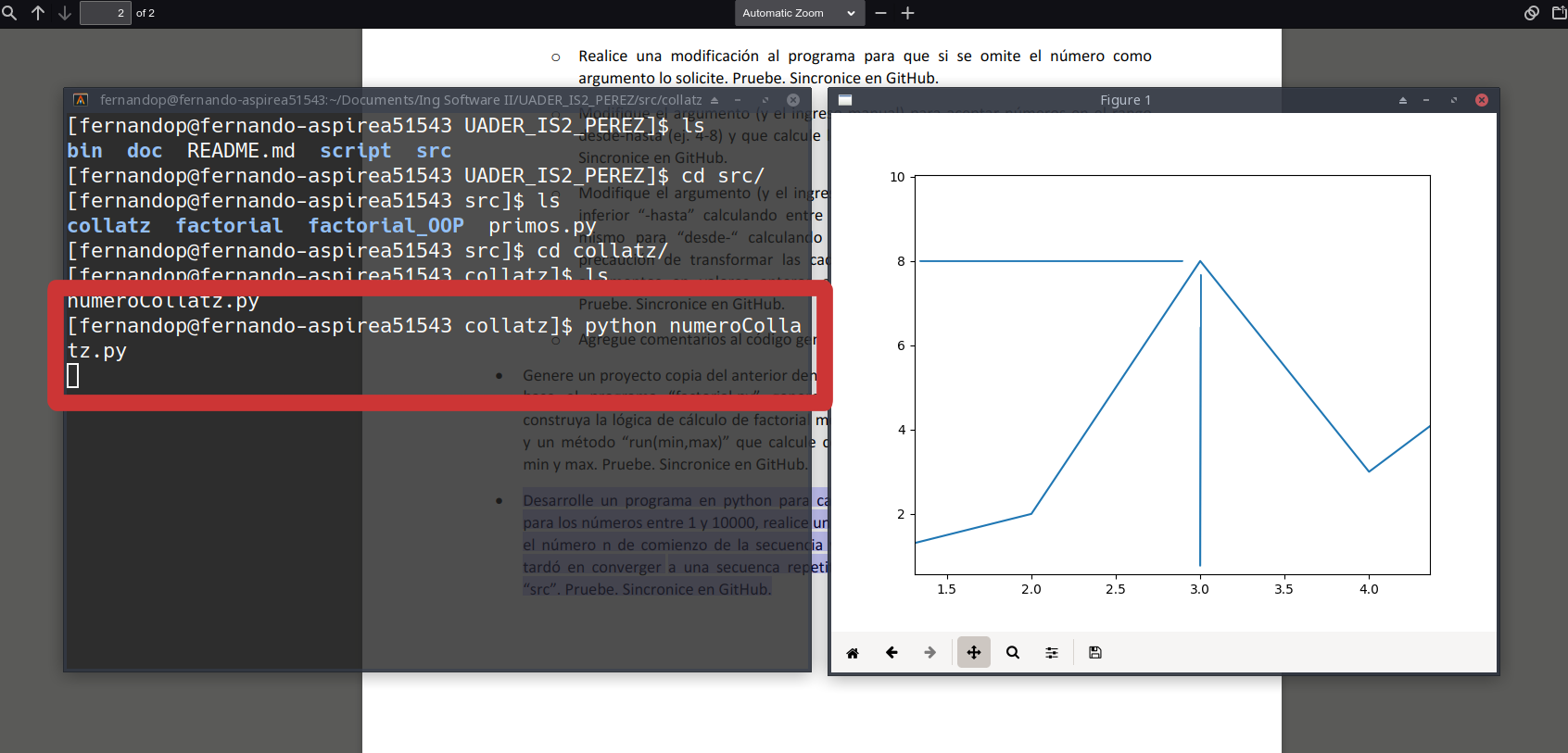
**GITHUB REPO:** <https://urlis.net/fact_OOP+folder>

**12)** 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 iteracionesque tardó en convergera una secuenca repetitiva. Coloque en una carpeta en la jerarquía “src”. Pruebe. Sincronice en GitHub.

**12\_** Se desarrolló un programa en python en el cual calcula el número de Collatz (conjetura 2n+1)  para los números entre 1 y 1000 y los grafica con Matplotlib (para graficar se utilizó solo 100 numeros) , tomando como eje X el número natural dispuesto (ejemplo: 3) y en el eje Y la cantidad de interacciones que necesitó para entrar en la secuencia 4 2 1, (ejemplo: de 3 es 8 la cantidad de interacciones) como se muestra a continuación:



Ampliación de la imagen, siguiendo el ejemplo del número 3 cuyas interacciones son 8 para demostrar que el calculo funciona:



**GITHUB REPO:** [https://urlis.net/-numcollatz](https://urlis.net/NCollatz-grafico)