



Licenciatura en Sistemas de Información

## **Trabajo Práctico N°1**

**#Guía de la configuración  
#Programación Python**

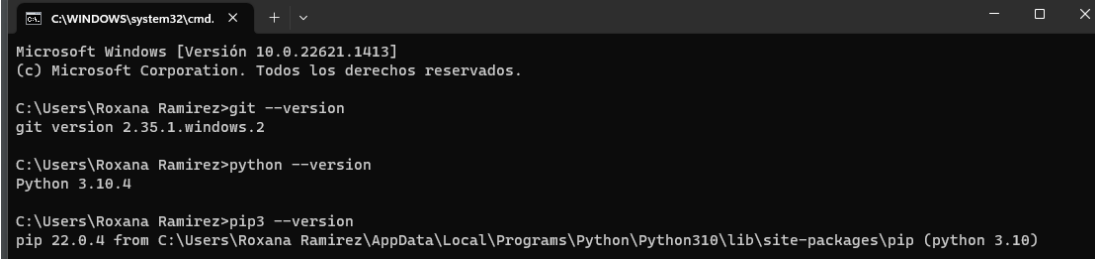
**Materia:** Ingeniería de Software II

**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)



```
C:\WINDOWS\system32\cmd. x + v
Microsoft Windows [Versión 10.0.22621.1413]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Roxana Ramirez>git --version
git version 2.35.1.windows.2

C:\Users\Roxana Ramirez>python --version
Python 3.10.4

C:\Users\Roxana Ramirez>pip3 --version
pip 22.0.4 from C:\Users\Roxana Ramirez\AppData\Local\Programs\Python\Python310\lib\site-packages\pip (python 3.10)
```

2. Obtenga una cuenta en [www.github.com](https://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
- o 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

```

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE (main)
$ cd src/

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE/src (main)
$ python primos.py
Find primes up to what number?: 5
[2, 3, 5]
Find how many primes?: 2
[2, 3]

```

Push en el repositorio:

```

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE (master)
$ git init
Initialized empty Git repository in C:/Users/Roxana Ramirez/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE/.git/

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE (master)
$ git add .

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE (master)
$ git commit -m "Primer commit"
[master (root-commit) 64e4d7c] Primer commit
1 file changed, 34 insertions(+)
create mode 100644 src/primos.py

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE (master)
$ git branch -M main

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE (main)
$ git remote add origin git@github.com:marisaroude/UADER_IS2_ROUDE.git

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 572 bytes | 286.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:marisaroude/UADER_IS2_ROUDE.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE (main)

```

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

```

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE/src (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

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

```

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

```

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_R
OUDE/src (main)
$ git checkout primos.py
Updated 1 path from the index

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_R
OUDE/src (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

```

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.

```

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_R
OUDE/src (main)
$ git add .

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_R
OUDE/src (main)
$ git commit -m "Modificación en el archivo de primos.py"
[main abddb9e] Modificación en el archivo de primos.py
1 file changed, 3 insertions(+), 1 deletion(-)

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_R
OUDE/src (main)
$ git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 406 bytes | 203.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:marisaroude/UADER_IS2_ROUDE.git
64e4d7c..abddb9e main -> main
branch 'main' set up to track 'origin/main'.

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_R
OUDE/src (main)
$ python primos.py
Find primes up to what number? : 7
[2, 3, 5, 7]
Find how many primes?: 3
[2, 3, 5]

```

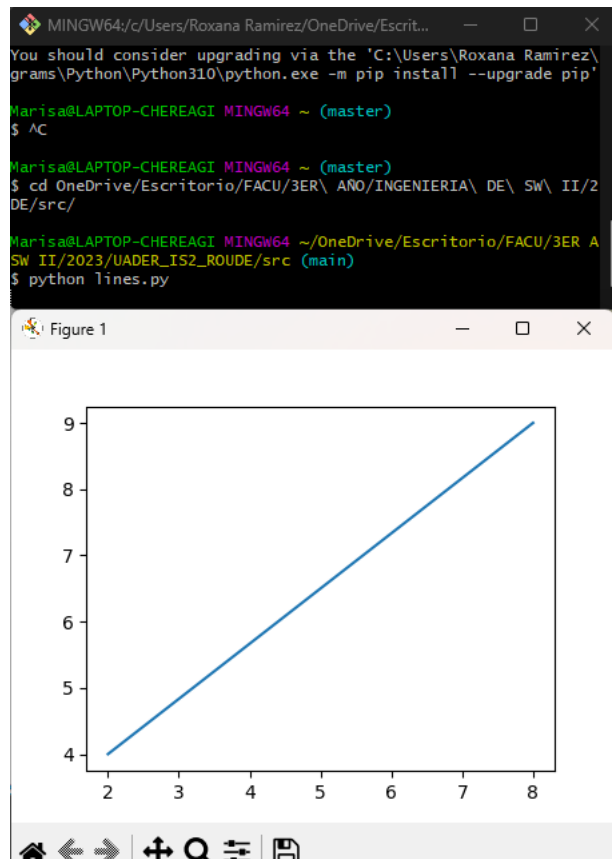
## #PROGRAMACION PYTHON

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:

```
Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER A  
ÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE/src (main)  
$ cd factorial/  
Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE SW II/2023/UADER_IS2_ROUDE/src/factorial (mai  
n)  
$ python factorial.py 5  
Factorial 5 ! es 120
```

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:

```
if len(sys.argv) == 0:  
    print("Debe informar un numero!")  
    sys.exit()
```

Lo deje de la siguiente forma:

```
if len(sys.argv) == 1:  
    num = int(input("Debe ingresar un número: "))  
else:  
    num = int(sys.argv[1])  
    #sys.exit()
```

Lo ejecutamos:

```
Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA  
DE SW II/2023/UADER-IS2-ROUDE/src/factorial (main)  
$ python factorial.py  
Debe ingresar un número: 5  
Factorial 5 ! es 120
```

Y luego lo sincronice 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:

```

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE
SW II/2023/UADER-IS2-ROUDE/src/factorial (main)
$ python factorial.py 2 8
Factorial del nro 2 es: 2
Factorial del nro 3 es: 6
Factorial del nro 4 es: 24
Factorial del nro 5 es: 120
Factorial del nro 6 es: 720
Factorial del nro 7 es: 5040
Factorial del nro 8 es: 40320

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE
SW II/2023/UADER-IS2-ROUDE/src/factorial (main)
$ python factorial.py
Debe ingresar el numero mínimo del rango: 2
Debe ingresar el numero máximo del rango: 8
Factorial del nro 2 es: 2
Factorial del nro 3 es: 6
Factorial del nro 4 es: 24
Factorial del nro 5 es: 120
Factorial del nro 6 es: 720
Factorial del nro 7 es: 5040
Factorial del nro 8 es: 40320

```

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:

```

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE
SW II/2023/UADER-IS2-ROUDE/src/factorial (main)
$ python factorial.py 20
Factorial del nro 1 es: 1
Factorial del nro 2 es: 2
Factorial del nro 3 es: 6
Factorial del nro 4 es: 24
Factorial del nro 5 es: 120
Factorial del nro 6 es: 720
Factorial del nro 7 es: 5040
Factorial del nro 8 es: 40320
Factorial del nro 9 es: 362880
Factorial del nro 10 es: 3628800
Factorial del nro 11 es: 39916800
Factorial del nro 12 es: 479001600
Factorial del nro 13 es: 6227020800
Factorial del nro 14 es: 87178291200
Factorial del nro 15 es: 1307674368000
Factorial del nro 16 es: 20922789888000
Factorial del nro 17 es: 355687428096000
Factorial del nro 18 es: 6402373705728000
Factorial del nro 19 es: 121645100408832000
Factorial del nro 20 es: 2432902008176640000

```

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.

```

Marisa@LAPTOP-CHEREAGI MINGW64 ~/OneDrive/Escritorio/FACU/3ER AÑO/INGENIERIA DE
SW II/2023/UADER-IS2-ROUDE/src/factorial (main)
$ python factorial.py 40
Factorial del nro 40 es: 8159152832478977343456112695961158942720000000000
Factorial del nro 41 es: 334525266131638071081700620534407516651520000000000
Factorial del nro 42 es: 14050061177528798985431426062445115699363840000000000
Factorial del nro 43 es: 604152630633738356373551320685139975072645120000000000
Factorial del nro 44 es: 265827157478844876804362581101461589031963852800000000
00
Factorial del nro 45 es: 11962222086548019456196316149565771506438373376000000
0000
Factorial del nro 46 es: 55026221598120889498503054288002548929616517529600000
00000
Factorial del nro 47 es: 25862324151116818064296435515361197996919763238912000
0000000
Factorial del nro 48 es: 12413915592536072670862289047373375038521486354677760
000000000
Factorial del nro 49 es: 60828186403426756087225216332129537688755283137921024
0000000000
Factorial del nro 50 es: 30414093201713378043612608166064768844377641568960512
000000000000
Factorial del nro 51 es: 15511187532873822802242430164693032110632597200169861
1200000000000000
Factorial del nro 52 es: 80658175170943878571660636856403766975289505440883277
8240000000000000
Factorial del nro 53 es: 42748832840600255642980137533893996496903437883668137
246720000000000000
Factorial del nro 54 es: 23084369733924138047209274268302758108327856457180794
11322880000000000000
Factorial del nro 55 es: 12696403353658275925965100847566516959580321051449436
7622758400000000000000
Factorial del nro 56 es: 71099858780486345185404564746372494973649797888116845
868744704000000000000000
Factorial del nro 57 es: 40526919504877216755680601905432322134980384796226602
14518448128000000000000000
Factorial del nro 58 es: 23505613312828785718294749105150746838288623181811429
2442069991424000000000000000
Factorial del nro 59 es: 13868311854568983573793901972038940634590287677268743
254082129494016000000000000000
Factorial del nro 60 es: 83209871127413901442763411832233643807541726063612459
524492776964096000000000000000

```

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

Agregué un comentario sobre el for y luego sincronice

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 corrobore 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 sincronice con GitHub.



```
MINGW64/c/Users/Roxana Ramirez/OneDrive/Escritorio/FACU/3ER AÑO/IN...
SW II/2023/UADER_IS2_ROUDE/src/factorial (main)
$ python factorial_OOP.py 1 20
Factorial del nro 1 es: 1
Factorial del nro 2 es: 2
Factorial del nro 3 es: 6
Factorial del nro 4 es: 24
Factorial del nro 5 es: 120
Factorial del nro 6 es: 720
Factorial del nro 7 es: 5040
Factorial del nro 8 es: 40320
Factorial del nro 9 es: 362880
Factorial del nro 10 es: 3628800
Factorial del nro 11 es: 39916800
Factorial del nro 12 es: 479001600
Factorial del nro 13 es: 6227020800
Factorial del nro 14 es: 87178291200
Factorial del nro 15 es: 1307674368000
Factorial del nro 16 es: 20922789888000
Factorial del nro 17 es: 355687428096000
Factorial del nro 18 es: 6402373705728000
Factorial del nro 19 es: 121645100408832000
Factorial del nro 20 es: 2432902008176640000
```

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 ordenadas 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 sincronice en GitHub.