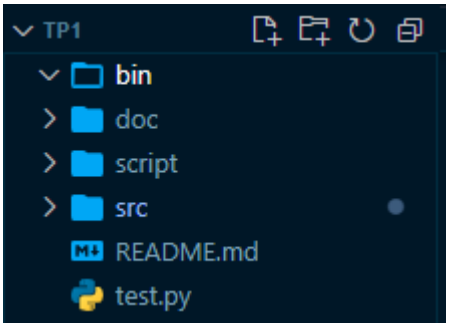
Ingeniería de Software II

TP1

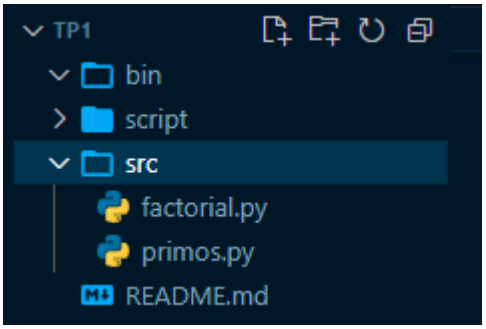
Gestión de la configuración y programación Python

Repositorio: https://github.com/mirkojp/TP1

- Instale los siguientes paquetes de software en la versión apropiada para el sistema operativo que utilice.
    - Git.
    - Python 3 (instalar desde python.org)
    - Pip3 (instalar desde python.org)
- 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:
    - src
    - doc
    - bin
    - script



- Obtenga el programa primos.py (en *Source Python.gz*) y siga las siguientes consignas:
    - Colóquelo en el directorio src local en su máquina.

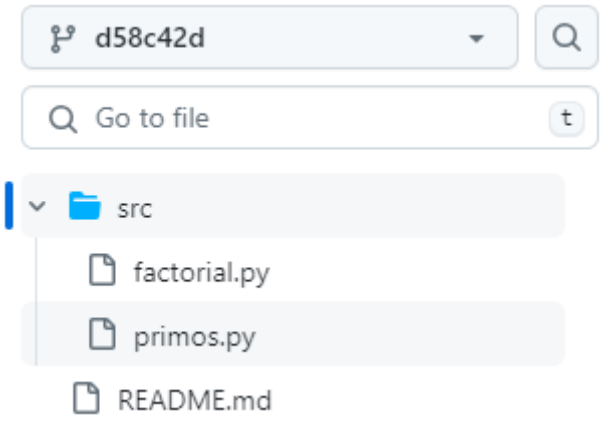


    - Ejecútelo con "*python3 primos.py*" y verifique que corre bien.

```
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1> python src/factorial.py 2
Factorial  2 ! es  2
```

Ingeniería de Software II

- Sincronícelo con el repositorio github.
  - *git add .*
  - *git commit -n carga_inicial*
  - *git push origin*
  - verifique la correcta actualización.



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

```
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:    src/factorial.py

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1>
```

```
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1> git restore src/factorial.py
```

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

```
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1> multimetric src/factorial.py
{
  "files": {
    "C:\\Users\\Admin\\Desktop\\Ing en software\\IS2\\TP1\\src\\factorial.py": {
      "comment_ratio": 51.701,
      "cyclomatic_complexity": 4,
```

```
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1> multimetric src/factorial.py
{
  "files": {
    "C:\\Users\\Admin\\Desktop\\Ing en software\\IS2\\TP1\\src\\factorial.py": {
      "comment_ratio": 62.258,
      "cyclomatic_complexity": 4,
```

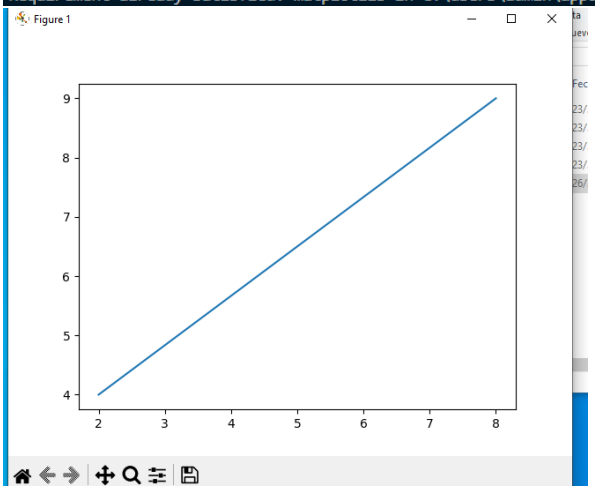
- Actualice el propósito del repositorio en GitHub por medio de un archivo *README.md*

que coloque en la raíz del repositorio. Actualice y verifique. Utilice la notación simple para que resalten los títulos, secciones, referencias. Incorpore al menos tres niveles de títulos, dos listas ordenadas, una lista numerada, una figura y una referencia a una página Web fuera del repositorio.

https://github.com/mirkojp/TP1 Link al repositorio (Contiene el readme)

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

```
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1> pip install matplotlib
Requirement already satisfied: matplotlib in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (3.8.3)
```



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

```
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1> python -u "c:\Users\Admin\Desktop\Ing en software\IS2\TP1\src\factorial.py"
Ingrese un número para calcular su factorial:
Debe ingresar un número válido.
Ingrese un número para calcular su factorial:
Debe ingresar un número válido.
Ingrese un número para calcular su factorial: 10
Factorial 10 ! es 3628800
```

    - 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.

```
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1> python src/factorial.py
Ingrese un rango (desde-hasta) para calcular los factoriales: 10-11
Los factoriales en el rango 10 - 11 son:
Factorial de 10 es 3628800
Factorial de 11 es 39916800
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1>
```

    - 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.

```
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1> python src/factorial.py
Ingrese un rango (desde-hasta) para calcular los factoriales: 10-11
Los factoriales en el rango 10 - 11 son:
Factorial de 10 es 3628800
Factorial de 11 es 39916800
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1> python src/factorial.py
Ingrese un rango (desde-hasta) para calcular los factoriales: -2
Los factoriales en el rango 1 - 2 son:
Factorial de 1 es 1
Factorial de 2 es 2
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1> python src/factorial.py
Ingrese un rango (desde-hasta) para calcular los factoriales: 59-
Los factoriales en el rango 59 - 60 son:
Factorial de 59 es 138683118545689835737939019720389406345902876772687432540821294940160000000000000
Factorial de 60 es 8320987112741390144276341183223364380754172606361245952449277696409600000000000000
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1>
```

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

```
PS C:\Users\Admin\Desktop\Ing en software\IS2\TP1> multimetric src/factorial.py
{
  "files": {
    "C:\\Users\\Admin\\Desktop\\Ing en software\\IS2\\TP1\\src\\factorial.py": {
      "comment_ratio": 39.943,
```

- 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.

```python
class Factorial:
    def __init__(self,min,max):
        self.min = min
        self.max = max

    def calcular_factorial(self,num):
        if num < 0:
            print("Factorial de un número negativo no existe")

        # Manejo de caso de 0
        elif num == 0:
            return 1

        # Manejo de caso numero enteros positivos
        else:
            fact = 1
            # Iteracion calculo factorial
            while(num > 1):
                fact *= num
                num -= 1
            return fact
```

Ingeniería de Software II

```
    def run(self):
        # Itera por el intervalo
        for num in range(self.min,self.max+1):
            # Calcula usando calcular_factorial(num)
            factorial = self.calcular_factorial(num)
            # Si existe lo escribe
            if factorial is not None:
                print(f"Factorial de {num} es {factorial}")
```

```
# Verificar si se proporciona un argumento en la línea de comandos
if len(sys.argv) == 1:
    # Si no se proporciona un argumento, solicitar al usuario que ingrese el rango
    while True:
        try:
            rango = input("Ingrese un rango (desde-hasta) para calcular los factoriales: ")
            # Caso -hasta
            if rango.startswith("-"):
                desde = 1
                hasta = int(rango.split("-")[1])
                break
            # Caso desde-
            elif rango.endswith("-"):
                desde = int(rango.split("-")[0])
                hasta = 60
                break
            # Caso desde-hasta
            if "-" in rango:
                desde, hasta = map(int, rango.split("-"))
                break
            # Levantar error
            else:
                raise ValueError
        except ValueError:
            print("Debe ingresar un rango válido en el formato 'desde-hasta', '-hasta' o 'desde-'.")
else:
```

```
else:

    # Si se proporciona un argumento en la línea de comandos, lo interpreta como un rango automáticamente
    rango = sys.argv[1]

    # Caso -hasta
    if rango.startswith("-"):
        desde = 1
        hasta = int(rango.split("-")[1])
    # Caso desde-
    elif rango.endswith("-"):
        desde = int(rango.split("-")[0])
        hasta = 60
    # Caso desde-hasta
    elif "-" in rango:
        desde, hasta = map(int, rango.split("-"))

# Crear una instancia de la clase Factorial y ejecutar el cálculo
factoriador = Factorial(desde, hasta)
factoriador.run()
```

- 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 secuenca repetitiva. Coloque en una carpeta en la jerarquía

Ingeniería de Software II

"src". Pruebe. Sincronice en GitHub.

```python
import matplotlib.pyplot as plt


# Define una función llamada collatz_secuencia que toma un
# número num como entrada y devuelve la secuencia de Collatz para ese número.
def collatz_secuencia(num):
    secuencia = [num]
    while num !=1:
        if num % 2 == 0:
            num=num //2
        else:
            num=3*num+1
        secuencia.append(num)
    return secuencia
```

```python
# Define una función llamada iteracion_collatz que calcula el número de iteraciones
# necesarias para que la secuencia de Collatz converja a 1
def iteracion_collatz(num):
    iteraciones = 0
    while num != 1:
        if num % 2 ==0:
            num = num // 2
        else:
            num = 3 * num + 1
        iteraciones += 1
    return iteraciones

# Calcular la secuencia de Collatz y el número de iteraciones para cada número en el rango
num_valores = range(1, 10001)
iteraciones_valores = [iteracion_collatz(num) for num in num_valores]
```

```python
# Graficar los resultados
plt.figure(figsize=(10, 6))
plt.scatter(iteraciones_valores, num_valores, marker=".", color="blue")
plt.title("Secuencia de Collatz")
plt.xlabel("Número de iteraciones")
plt.ylabel("Número inicial (n)")
plt.grid(True)
plt.show()
```