

Taller 2

Universidad Antonio Nariño

Construcción de Software

Construcción de Software

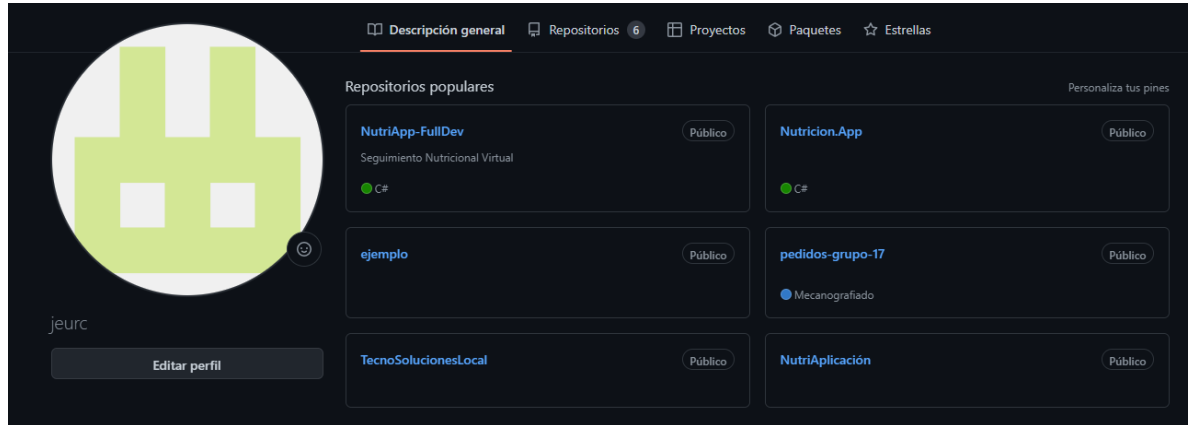
Jeu Rojas Castillo

Bogotá D.C.

2023

En este taller usted va a seguir paso a paso cómo usar git a través de un ejemplo práctico. Para completar este taller, debe haber completado satisfactoriamente el Taller 1 para configurar su ambiente de trabajo.

1. Cree una cuenta en GitHub



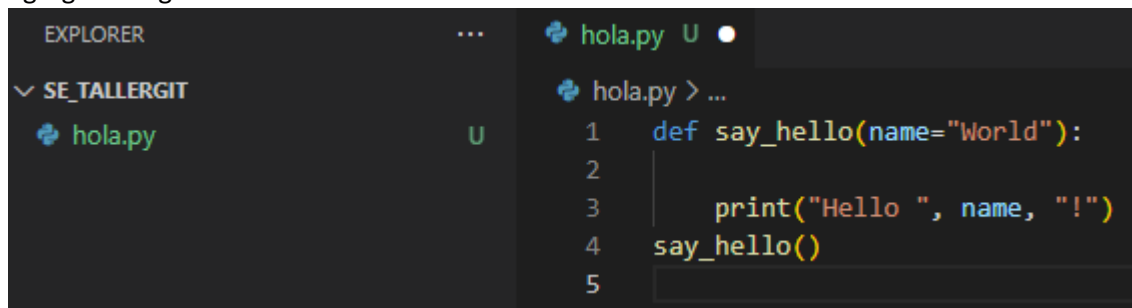
2. Cree una carpeta en su sistema para el nuevo proyecto. Nombre está como SE_TallerGit.

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1>mkdir SE_TallerGit
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1>cd SE_TallerGit
```

3. Desde la línea de comandos (cmd o terminal) inicialice en esta carpeta su repositorio local de Git usando el comando git init.

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\SE_TallerGit>git init
Initialized empty Git repository in D:/Universidad/Construccion de Software/4º Semestre/Construccion de Software/Corte 1/SE_TallerGit/.git/
```

4. Abra la carpeta de su proyecto desde VS Code y cree un nuevo archivo llamado hola.py y agregue el siguiente contenido:



5. Nuevamente desde la consola, verifique los cambios pendientes por agregar al repositorio, usando el comando git status. Debe obtener como resultado algo similar a lo siguiente:

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\SE_TallerGit>git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
```

6. Para agregar el archivo al repositorio use el comando git add. Esto agregara el archivo al área de stage esperando para que este cambio sea confirmado:

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\SE_TallerGit>git add *
```

7. Puede volver a verificar el estado de sus cambios con el comando git status.

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\SE_TallerGit>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   hola.py
```

8. Para confirmar sus cambios debe usar el comando git commit. Esto creará una nueva versión en su repositorio con los cambios que tiene en el área de stage. (En git una versión es llamada un commit). Para hacer un commit recuerde que siempre debe agregar un mensaje describiendo qué cambios está versionando.

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\SE_TallerGit>git commit -m "Version inicial"
[master (root-commit) 5a17819] Version inicial
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hola.py
```

9. Es momento de sincronizar sus cambios con un repositorio remoto en GitHub. Para esto vaya a GitHub y cree un nuevo repositorio con el mismo nombre de su proyecto (SE_TallerGit). Una vez haya creado el repositorio en GitHub copie la dirección del repositorio y use el comando git remote para conectar su repositorio local con el repositorio remoto. Esta es una operación que se debe realizar una única vez por cada repositorio remoto con el que se quiera sincronizar. (Recuerde reemplazar en este ejemplo la URL de su repositorio)

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\SE_TallerGit>git remote add origin https://github.com/JeuRC/SE_TallerGit.git
```

10. Ya que ha conectado el repositorio remoto use el comando git push para sincronizar sus cambios. (La opción -u origin master solo se requiere la primera vez que se sincroniza el branch).

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\SE_TallerGit>git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 278 bytes | 92.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/JeuRC/SE_TallerGit.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

11. En este momento sus cambios ya deben estar sincronizados con GitHub. En adelante, cada vez que quiera sincronizar más cambios, deberá agregarlos usando git add, confirmarlos usando git commit, y sincronizarlos con GitHub usando git push.

Trabajo colaborativo

Hasta este punto usted ya puede realizar cambios en un repositorio local y sincronizarlos con un repositorio remoto. Estos cambios pueden ser archivos nuevos, y/o la modificación o eliminación de archivos existentes. En esta sección usará otros comandos simulando ser otro desarrollador que está trabajando en el mismo proyecto.

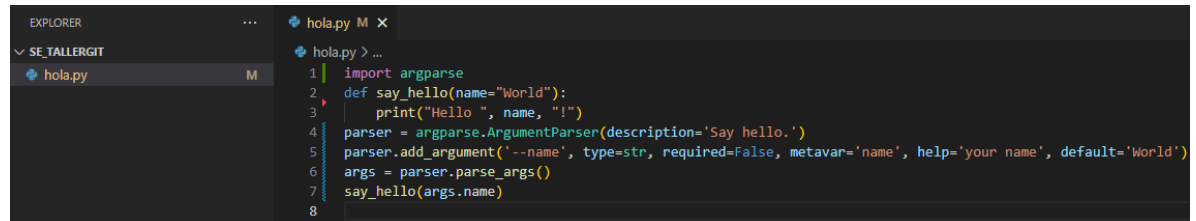
1. Cree una nueva carpeta por fuera de la carpeta actual del proyecto. Llámela usuario2.

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1>mkdir usuario2
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1>cd usuario2
```

2. Ejecute el comando git clone dentro de la carpeta usuario2 para clonar el repositorio remoto. Este paso es el que haría cualquier otro desarrollador para trabajar en el mismo proyecto. Esto creará una copia del proyecto en esta nueva carpeta. (reemplace usuario con su nombre de usuario de GitHub)

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\usuario2>git clone https://github.com/JeuRC/SE_TallerGit.git
Cloning into 'SE_TallerGit'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

3. En VS Code abra la nueva carpeta SE_TallerGit creada dentro de la carpeta usuario2 y modifique el contenido del archivo hola.py para que permita recibir como argumento su nombre:



```
1 | import argparse
2 | def say_hello(name="World"):
3 |     print("Hello ", name, "!")
4 | parser = argparse.ArgumentParser(description='Say hello.')
5 | parser.add_argument('--name', type=str, required=False, metavar='name', help='your name', default='World')
6 | args = parser.parse_args()
7 | say_hello(args.name)
8 |
```

4. Guarde sus cambios y agréguelos al área de stage usando git add

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\usuario2\SE_TallerGit>git add *
```

5. Luego confirme sus cambios y sincronice estos con GitHub usando los comandos commit y push.

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\usuario2\SE_TallerGit>git commit -m "Se agrega --name como argumento del programa"
[main 030e0fc] Se agrega --name como argumento del programa
1 file changed, 5 insertions(+), 3 deletions(-)

D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\usuario2\SE_TallerGit>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 455 bytes | 227.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/JeuRC/SE_TallerGit.git
a1b4488..030e0fc main -> main
```

6. En este momento ya existen dos versiones del proyecto (dos commits), puede verificarlo usando el comando git log o revisando la lista de commits en GitHub (dar clic sobre el enlace commits).

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\usuario2\SE_TallerGit>git log
commit 030e0fc0405130070b40b3a0818c65e27adf248c (HEAD -> main, origin/main, origin/HEAD)
Author: JeuRC <jeurojascastillo@gmail.com>
Date: Thu Mar 2 12:40:31 2023 -0500

    Se agrega --name como argumento del programa

commit a1b44886e8e1a8278e197ec8d48885adf9856bf4
Author: JeuRC <jeurojascastillo@gmail.com>
Date: Thu Mar 2 12:29:53 2023 -0500

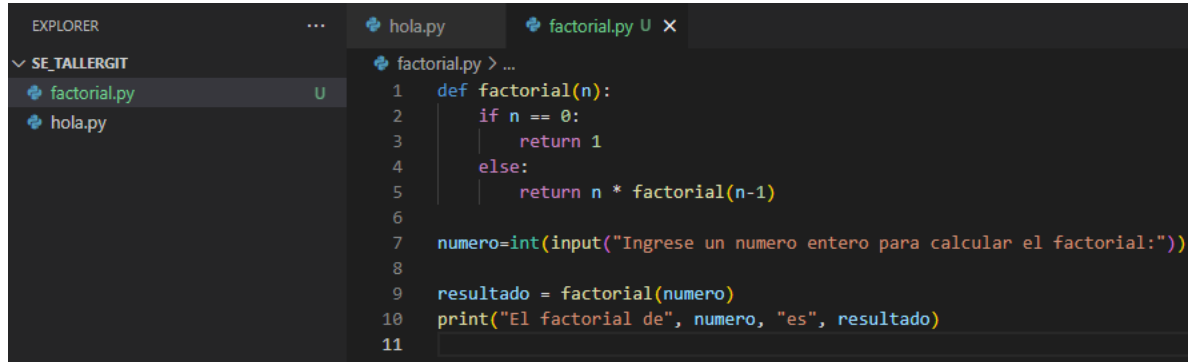
    Version inicial
```



Agregar un nuevo programa al proyecto

Agregue un nuevo archivo al proyecto llamado factorial.py. En este archivo escriba un programa que calcule la factorial de un número. El programa debe recibir un argumento N que será el número para el cual se calculará la factorial. Agregue sus cambios a git y sincronice estos con GitHub. Al terminar comparta la dirección web de su repositorio en GitHub en Moodle. Asegúrese de que sus cambios si están publicados en GitHub.

1. En este archivo escriba un programa que calcule la factorial de un número. El programa debe recibir un argumento N que será el número para el cual se calculará la factorial.



The screenshot shows a code editor with two files: `hola.py` and `factorial.py`. The `factorial.py` file is open and contains the following Python code:

```
1 def factorial(n):
2     if n == 0:
3         return 1
4     else:
5         return n * factorial(n-1)
6
7 numero=int(input("Ingrese un numero entero para calcular el factorial:"))
8
9 resultado = factorial(numero)
10 print("El factorial de", numero, "es", resultado)
11
```

2. Guarde sus cambios y agréguelos al área de stage usando git add

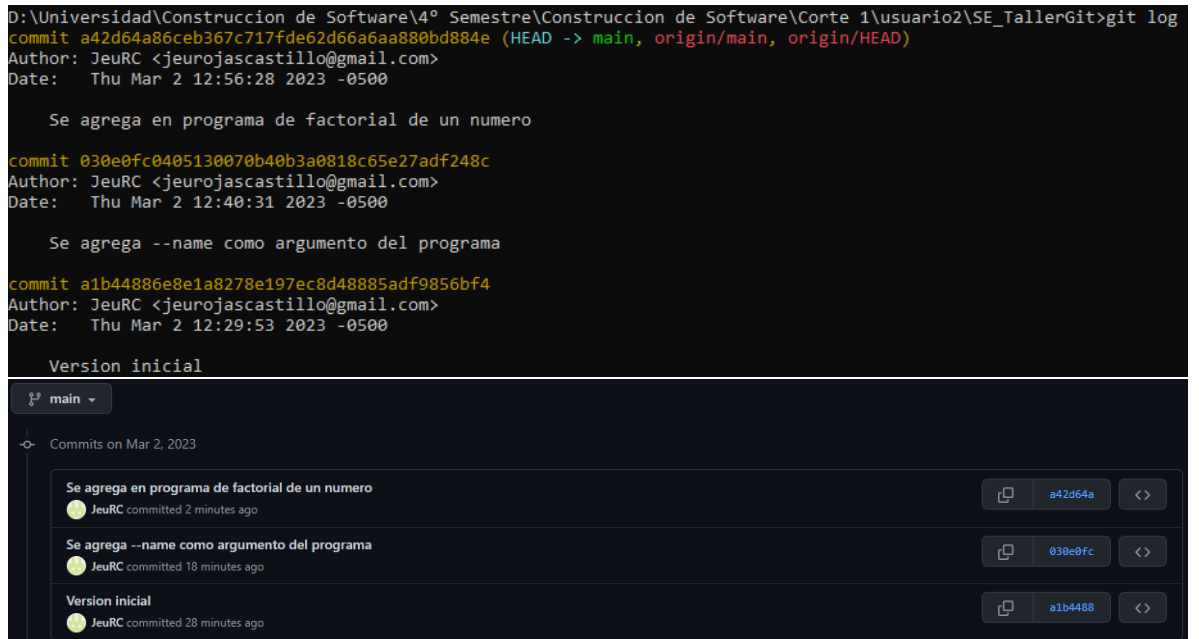
```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\usuario2\SE_TallerGit>git add *
```

3. Luego confirme sus cambios y sincronice estos con GitHub usando los comandos commit y push.

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\usuario2\SE_TallerGit>git commit -m "Se agrega en programa de factorial de un numero"
[main a42d64a] Se agrega en programa de factorial de un numero
1 file changed, 10 insertions(+)
create mode 100644 factorial.py

D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\usuario2\SE_TallerGit>git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 446 bytes | 446.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/JeuRC/SE_TallerGit.git
030e0fc..a42d64a main -> main
```

7. En este momento ya existen dos versiones del proyecto (dos commits), puede verificarlo usando el comando git log o revisando la lista de commits en GitHub (dar clic sobre el enlace commits).



The screenshot shows the output of the `git log` command and the GitHub commit history page.

git log output:

```
D:\Universidad\Construccion de Software\4º Semestre\Construccion de Software\Corte 1\usuario2\SE_TallerGit>git log
commit a42d64a86ceb367c717fde62d66a6aa880bd884e (HEAD -> main, origin/main, origin/HEAD)
Author: JeuRC <jeurojascastillo@gmail.com>
Date: Thu Mar 2 12:56:28 2023 -0500

    Se agrega en programa de factorial de un numero

commit 030e0fc0405130070b40b3a0818c65e27adf248c
Author: JeuRC <jeurojascastillo@gmail.com>
Date: Thu Mar 2 12:40:31 2023 -0500

    Se agrega --name como argumento del programa

commit a1b44886e8e1a8278e197ec8d48885adf9856bf4
Author: JeuRC <jeurojascastillo@gmail.com>
Date: Thu Mar 2 12:29:53 2023 -0500

    Version inicial
```

GitHub commit history:

The GitHub page shows the commit history for the repository. The commits are listed in chronological order, with the most recent commit at the top.

Commit Message	Author	Date	Commit Hash
Se agrega en programa de factorial de un numero	JeuRC	committed 2 minutes ago	a42d64a
Se agrega --name como argumento del programa	JeuRC	committed 18 minutes ago	030e0fc
Version inicial	JeuRC	committed 28 minutes ago	a1b4488