



**UNIVERSIDAD PRIVADA
DOMINGO SAVIO**

INGENIERIA EN SISTEMAS

GRUPO LOS MAGIOS

NOMBRE DE LOS ESTUDIANTES

Limber David Quispe Osco

Yery Torrico Ribera

Salomon Leon Pesoa

Beymar Ferrufino Peredo

Daniel Alanes Caceres

DOCENTE

Jimmy Nataniel Requena Llorentty

Santa Cruz – Bolivia

16/07/2025

Índice

INTRODUCCION A GITHUB Y REPLIT.....	3
1. Introducción a GitHub.	3
2. Segundo paso para crear cuenta de GitHub.	4
3. Verificación de cuenta.....	5
4. Inicio de sesión en GitHub.	6
5. iniciar sesión ya con nuestra cuenta.....	7
6. Creación de repositorios.	8
7. Creación y colocado de nombre a repositorio.....	9
8. Edición de archivo "Readme.md".	10
9. Editando archivo "Readme.md".....	11
10. Agregado de colaboradores a nuestro repositorio.....	12
11. Creación de carpetas.	13
12. Creacion de carpetas para los integrantes.....	14
13. Introducción a Replit.....	15
14. Vinculación de cuenta.	16
15. Creación de repositorios en Replit.....	17
16. Creación de folder Replit.	18
17. Ejecutar códigos mediante la consola "Shell".....	19
EJECUCION DE CODIGOS.....	20
1. Código Hola mundo.	20
2. Código Tabla de multiplicar.....	21
3. Código Verificador de Edad para Película.	22
4. Código Tabla de Multiplicar.	23
5. Código Área de Rectángulo.....	24
6. Código Promedio de notas.	25
7. Código Presentación de Grupo.....	26
8. Código Lista invertida	27
9. Código Encuentra al Mayor.	28
10. Código Factorial.....	29
11. Código Funciones.....	30

INTRODUCCION A GITHUB Y REPLIT.

1. Introducción a GitHub.

El primer paso para poder tener acceso a GitHub es crear una cuenta. Lo cual tenemos que ir a la página de GitHub.

Después haber buscado la página procedemos a empezar a crear nuestra cuenta y le damos click en en “Sing up”.



2. Segundo paso para crear cuenta de GitHub.

En este parte tenemos rellenar el cual nos pide las siguientes cosas:

- Email, podemos usar nuestro correo personal porque nos van a enviar mensaje de verificación.
- Password, es la contraseña con la cual vamos a acceder después de crear la cuenta.
- Username, es el nombre con el cual va estar tu cuenta de GitHub.
- Your Country/Región, ahí uno tiene el país de donde se encuentra.

Después de haber rellenado todo correctamente le damos click en “Create account” ya para que se cree la cuenta.



Sign up to GitHub

Email*
yerytr1806200@gmail.com ✓

Password*
***** ✓
Password should be at least 10 characters OR at least 8 characters including a number and a lowercase letter.

Username*
yery-torica ✓
Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

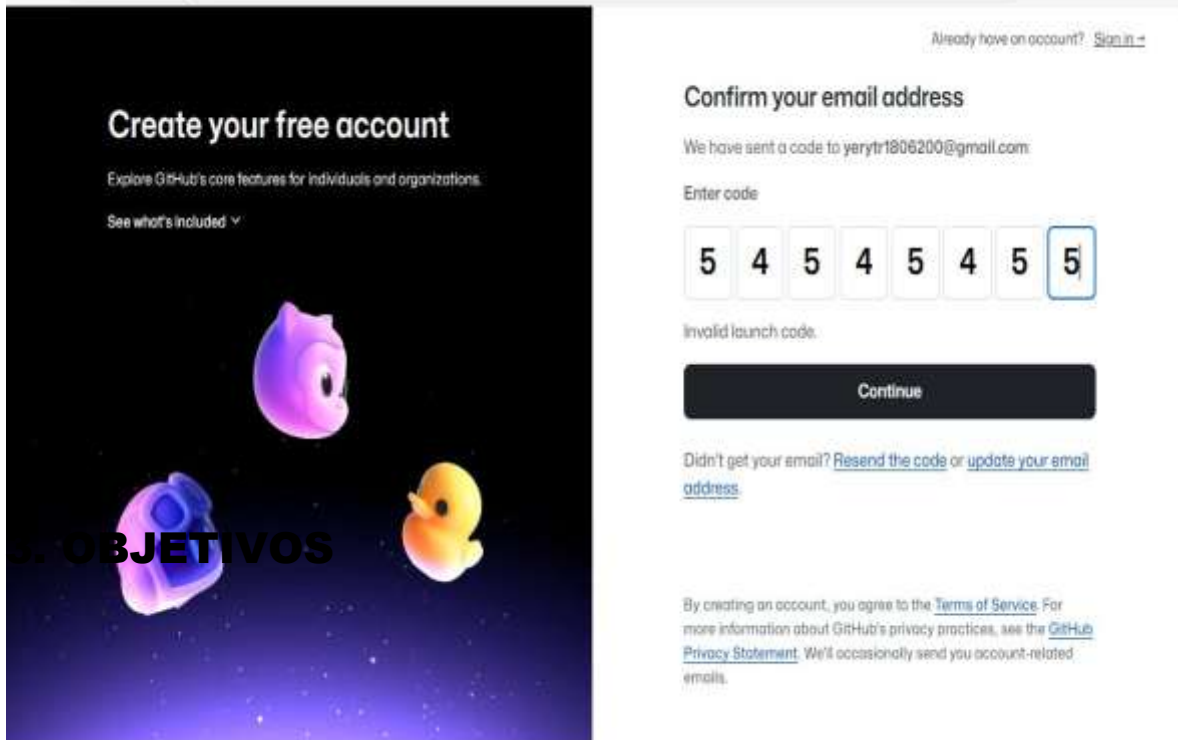
Your Country/Region*
Bolivia ✓
For compliance reasons, we're required to collect country information to send you occasional updates and announcements.

Email preferences
☒ Receive occasional product updates and announcements

Create account >

3. Verificación de cuenta.

Aquí nos van enviar un código de confirmación a nuestro correo y es solo copiar el código de verificación y le damos click en “Continue”.



Create your free account
Explore GitHub's core features for individuals and organizations.
[See what's included](#) ▾

Confirm your email address
We have sent a code to yerytr1806200@gmail.com.
Enter code

5 4 5 4 5 4 5 5

Invalid launch code.

[Continue](#)

Didn't get your email? [Resend the code](#) or [update your email address](#).

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

3. OBJETIVOS

4. Inicio de sesión en GitHub.

Le damos click en “Sing in” ya para poder iniciar sesión.



5. iniciar sesión ya con nuestra cuenta.

En esta parte ponemos nuestro “Username” o “Email” y en la parte de “Password” nuestra contraseña.

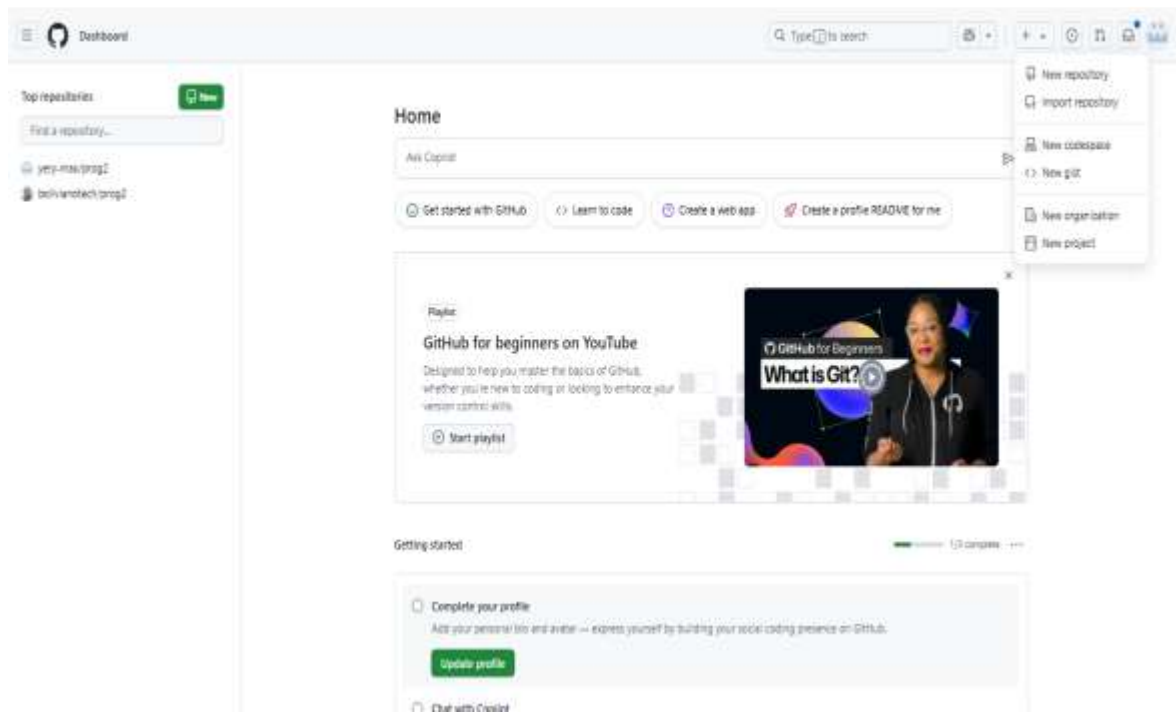
Ya ingresaríamos a nuestra cuenta de GitHub.



The image shows the GitHub login interface. At the top is the GitHub logo and the text "Sign in to GitHub". Below this is a form with two input fields: "Username or email address" and "Password". The first field contains the email "jerrytr1884200@gmail.com". The second field contains a masked password "*****". To the right of the password field is a link that says "forgot password?". Below the password field is a green "Sign in" button. Underneath the button is a horizontal line with the text "or" in the center. Below the line is a button that says "Continue with Google". At the bottom of the form, there are two links: "New to GitHub? Create an account" and "Sign in with a passkey".

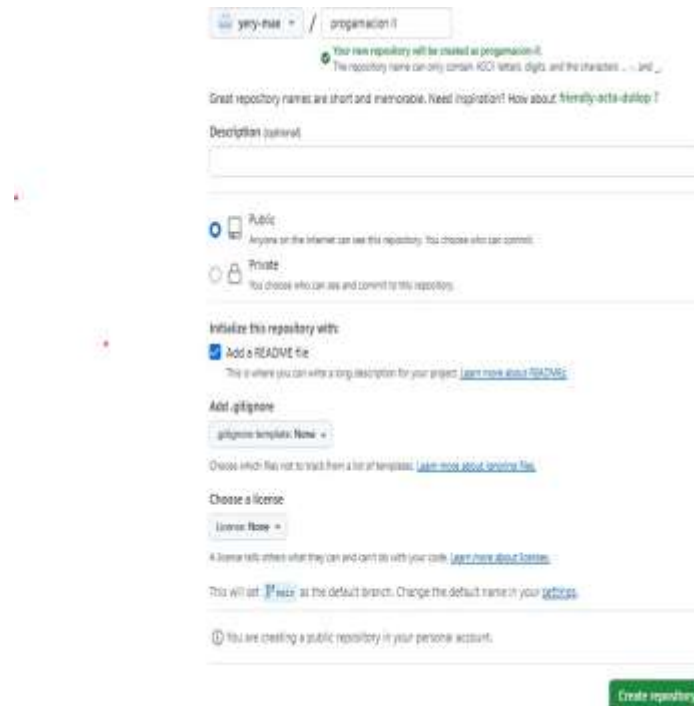
6. Creación de repositorios.

Ya estando dentro de nuestra cuenta procedemos a crear nuestro primer repositorio la cual creamos dándole click en “New repository”.



7. Creación y colocado de nombre a repositorio.

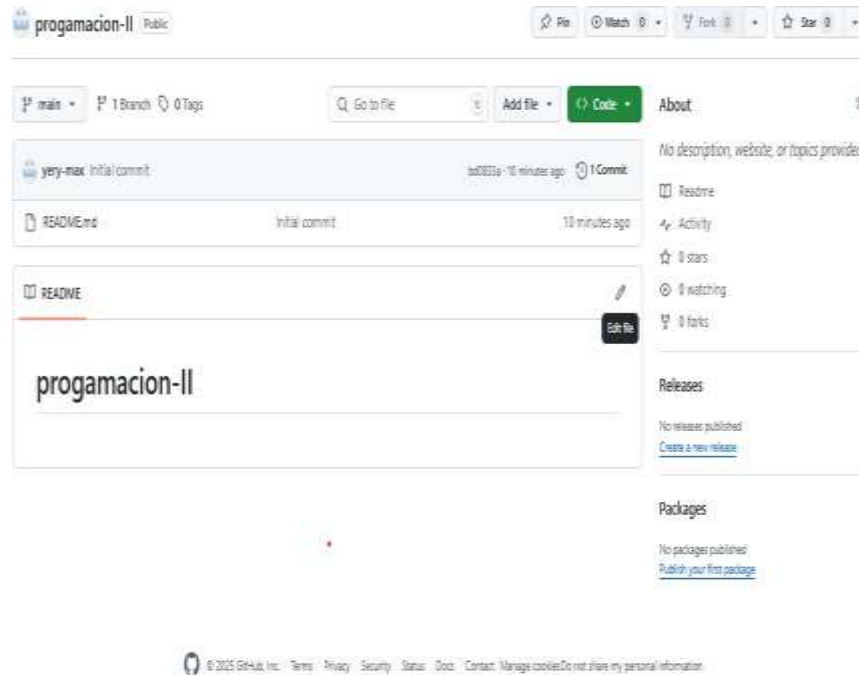
En esta parte tenemos que elegir un nombre para nuestro repositorio. También la vamos a dejar publico nuestro repositorio y también le damos click en la opción “Add a README file” para que nos cree un archivo “Readme.md” que es documentación inicial de un proyecto, ya después haber hecho eso ya le damos click en “Create repository” para que cree el repositorio.



The screenshot shows the GitHub 'Create new repository' form. At the top, the username 'yey-mee' and the repository name 'programacion-II' are entered. A green message states: 'Your new repository will be created as programacion-II. The repository name can only contain ASCII letters, digits, and the characters -, ., and _.' Below this, a tip suggests: 'Great repository names are short and memorable. Need inspiration? How about friendly-acta-develop?'. There is a text input field for the 'Description (optional)'. Under the 'Visibility' section, the 'Public' option is selected with the note 'Anyone on the Internet can see this repository. You choose who can commit.' The 'Private' option is also visible. In the 'Initialize this repository with:' section, the 'Add a README file' checkbox is checked, with a note: 'This is where you can write a long description for your project. [Learn more about READMEs](#)'. Below this is the 'Add .gitignore' section with a dropdown menu showing 'gitignore template: None'. The 'Choose which files not to track from a list of templates' section has a link to 'Learn more about ignoring files'. The 'Choose a license' section has a dropdown menu showing 'License: None'. A note at the bottom states: 'A license tells others what they can and can't do with your code. [Learn more about licenses](#)'. At the very bottom, it says 'This will set `main` as the default branch. Change the default name in your [settings](#).' and a note: 'You are creating a public repository in your personal account.' A green 'Create repository' button is at the bottom right.

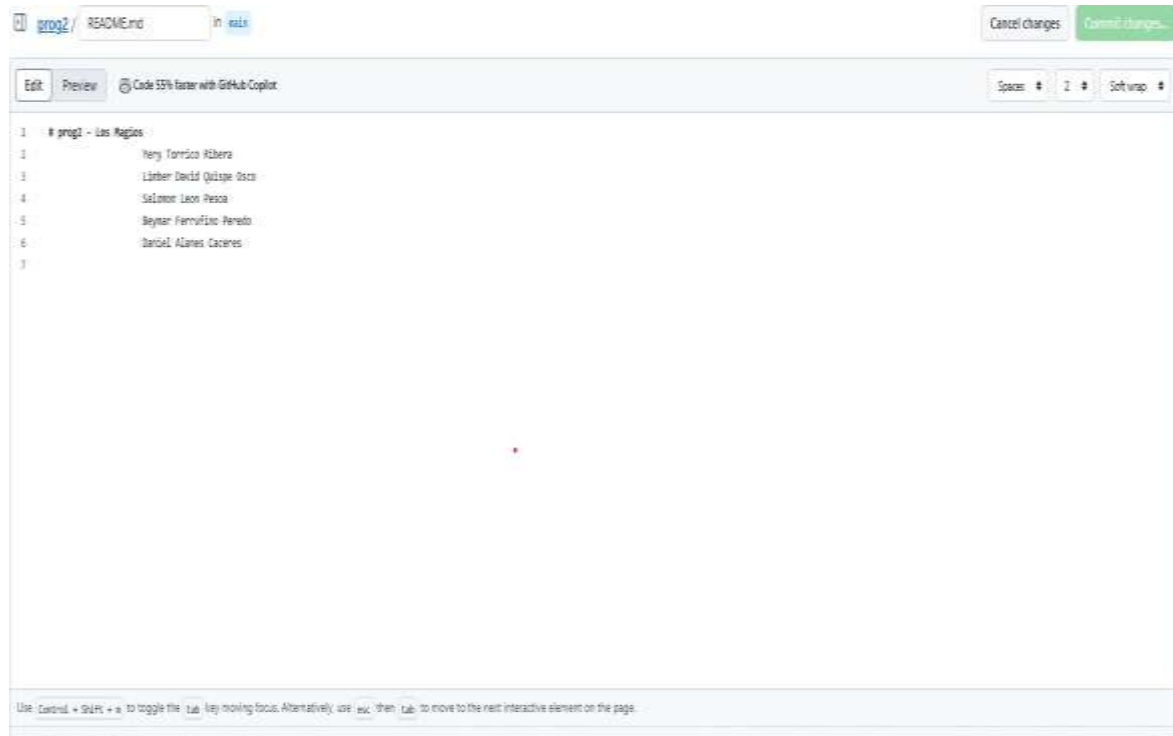
8. Edición de archivo “Readme.md”.

Le damos click en la parte de donde dice “Edit file”.



9. Editando archivo “Readme.md”.

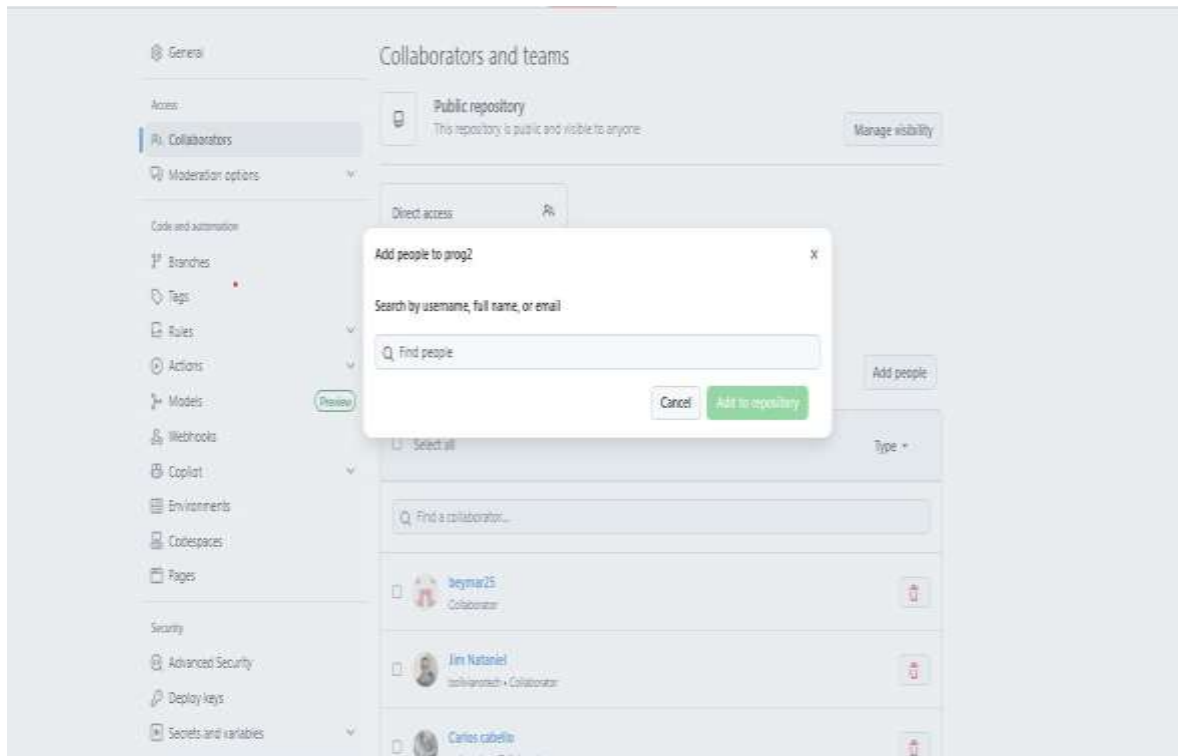
En esta parte hacemos lista con los integrantes del grupo que hicimos en clases y le ponemos el nombre que eligió de como se va a llamar el grupo en este caso Los Magios.



```
1 # prog2 - Los Magios
2
3     Nery Tomsico Ribera
4     Lister David Quijano Ocas
5     Salomon Leon Pesca
6     Beymar Ferrufino Parado
7     Daniel Alanes Caceres
8
```

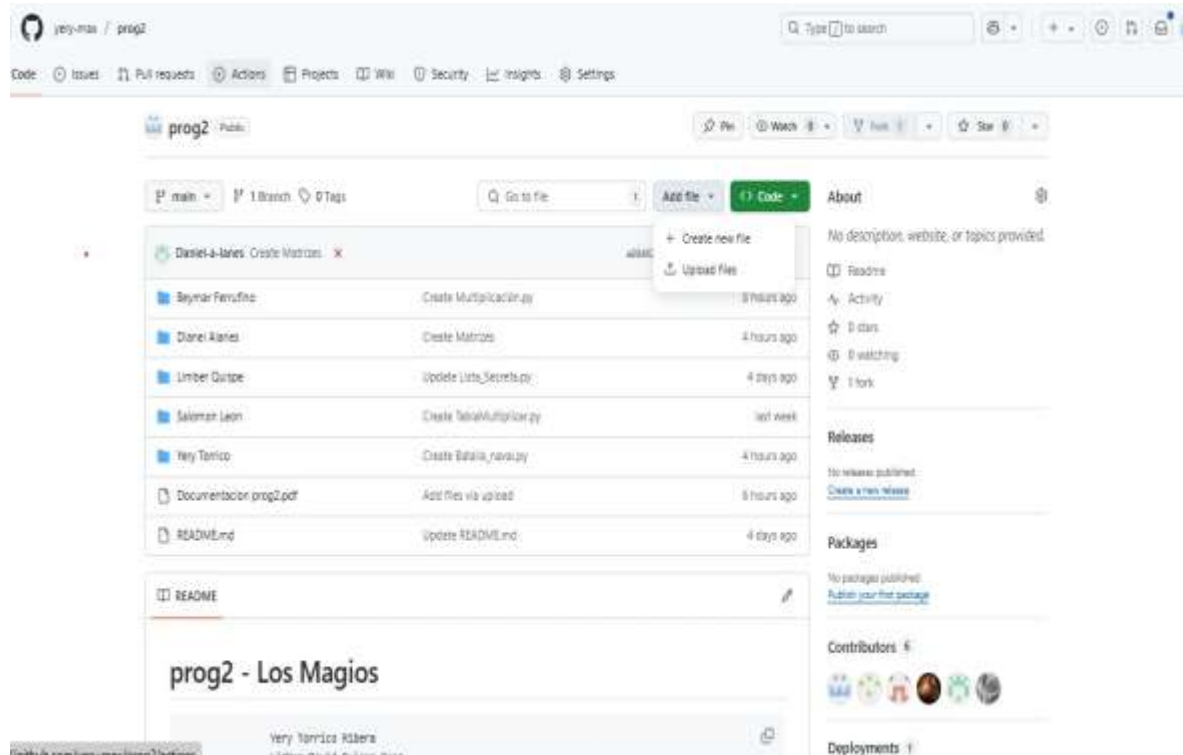
10. Agregado de colaboradores a nuestro repositorio.

En esta parte agregamos a los integrantes del grupo para que ellos pueden ver y editar el repositorio.



11. Creación de carpetas.

Aquí para poder crear carpetas para que cada uno de los integrantes del grupo le dimos click en “Add file” y después en “+ Create new file”.

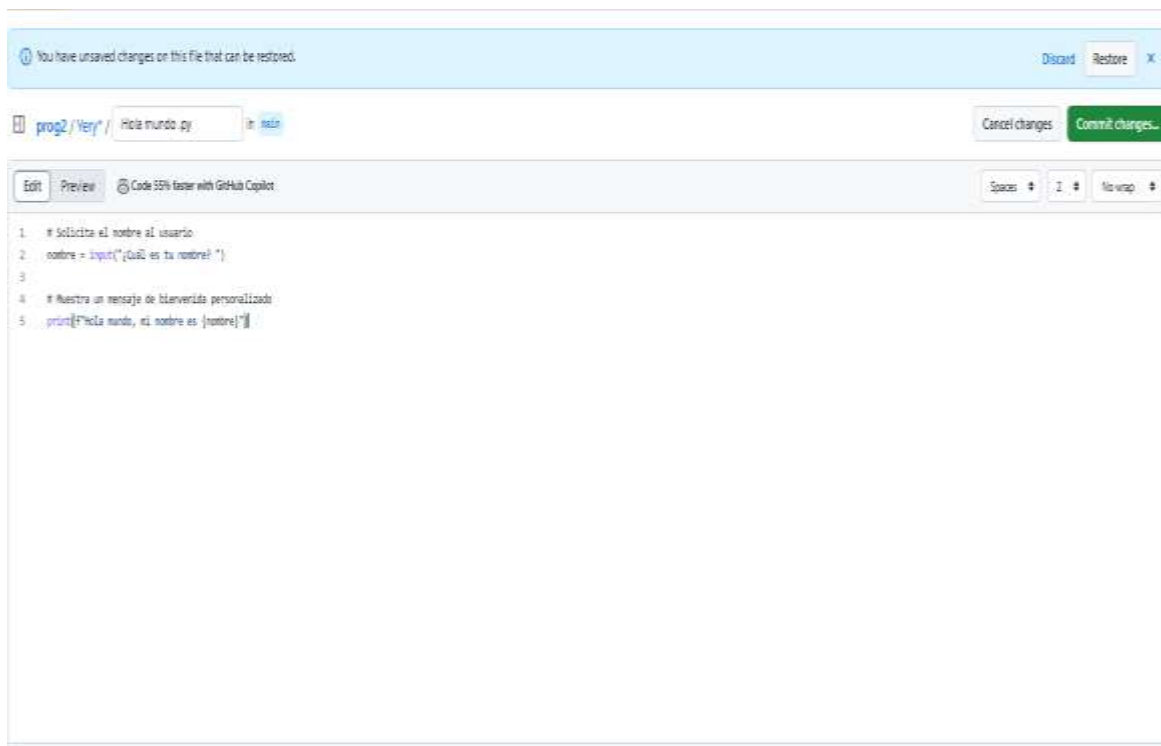


12. Creacion de carpetas para los integrantes.

En esta parte primero creamos la carpeta lo cual llamaremos en este caso “Yery” que es uno de lo integrantes del grupo y dentro de esa carpeta creamos otra carpeta ya para poder copiar los códigos, en este caso copiamos el código que lleva por nombre “Hola_mundo.py”.

Después haber hecho eso le damos click en “Commit changes” para que se guarden los que añadimos.

Ya cada uno de los integrantes del grupo va a ir subiendo sus códigos a sus carpetas.



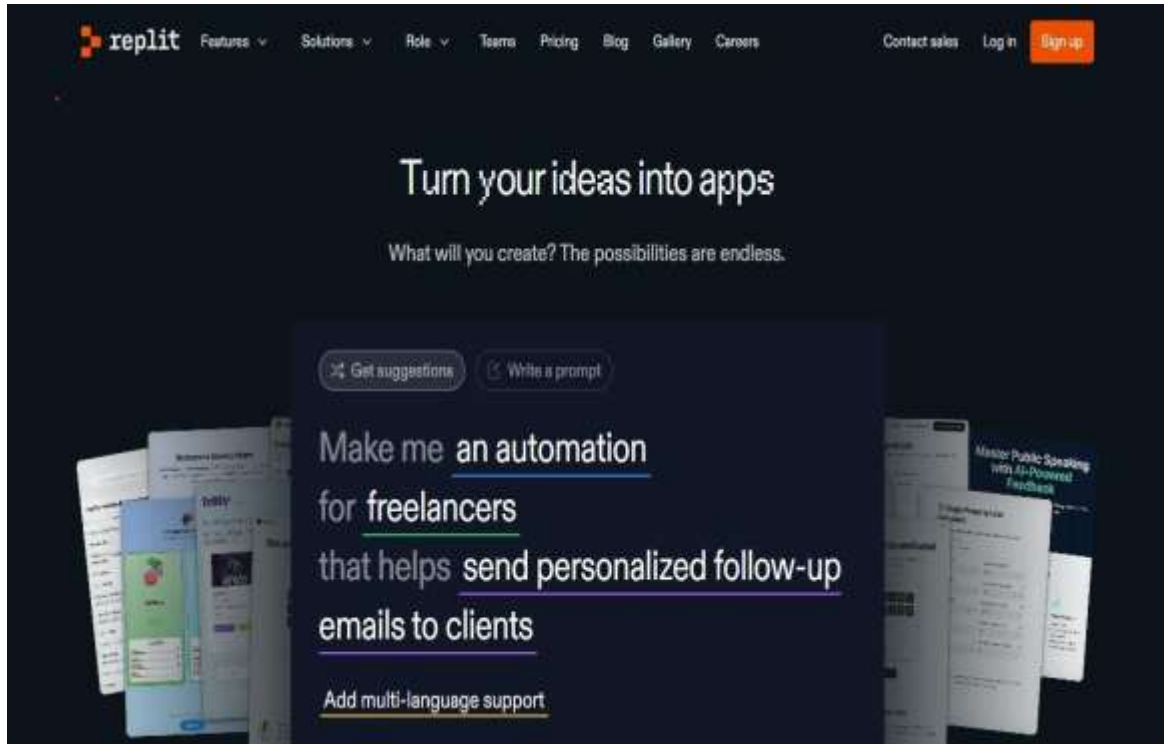
The screenshot shows a code editor interface. At the top, there is a blue notification bar that says "You have unsaved changes on this file that can be restored." with "Discard" and "Restore" buttons. Below this, the file path is shown as "prog2 / Yery / Hola mundo.py". There are "Cancel changes" and "Commit changes..." buttons. The editor has tabs for "Edit" and "Preview", and a link to "Code 55% faster with GitHub Copilot". The code in the editor is as follows:

```
1. # Solicita el nombre al usuario
2. nombre = input("¿Cuál es tu nombre? ")
3.
4. # Muestra un mensaje de bienvenida personalizado
5. print(f"Hola mundo, mi nombre es {nombre}")
```

At the bottom right, there are settings for "Spaces" (4), "Tab" (set), and "No wrap" (set).

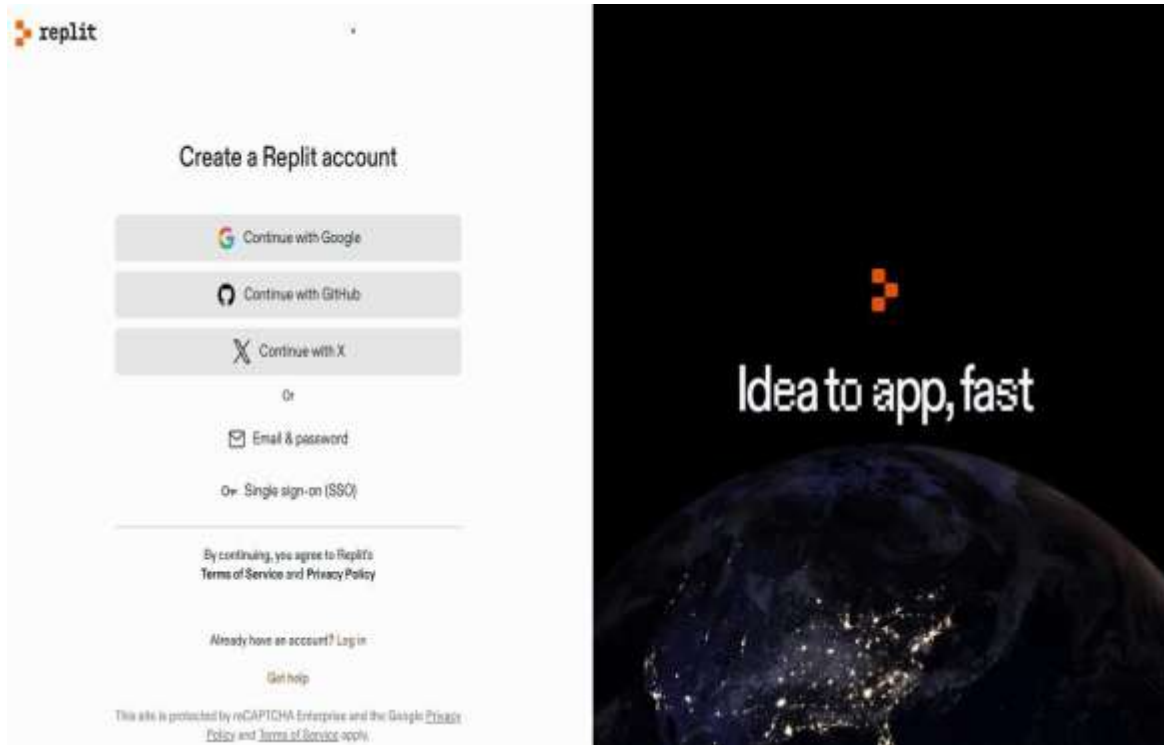
13. Introducción a Replit.

En el navegador buscamos la página de replit y le damos click en “Sing Up”.



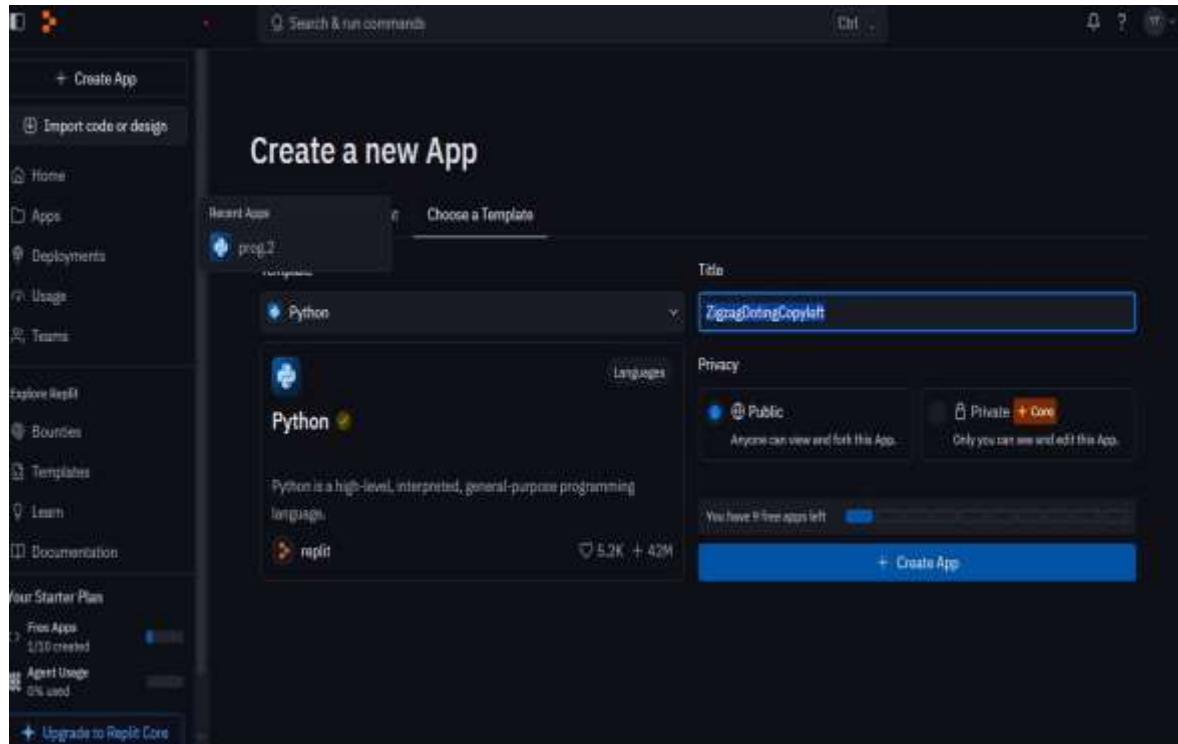
14. Vinculación de cuenta.

En esta parte vamos a vincular con nuestra de cuenta de GitHub para poder iniciar sesión en Replit.



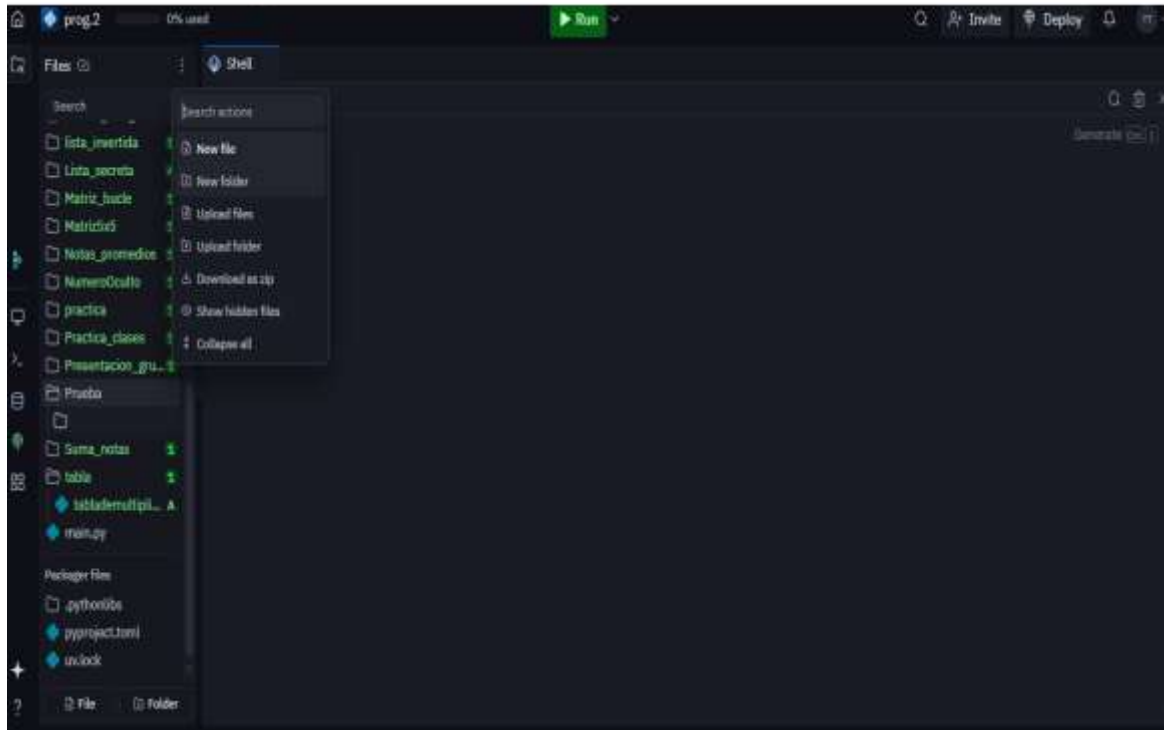
15. Creación de repositorios en Replit.

En esta parte le damos click en “+ Create a new App” y después le damos click en “Choose a Template” y buscamos el lenguaje de programación Python y le asignamos un nombre al repositorio y ya procedemos a darle click en “+ Create App”.



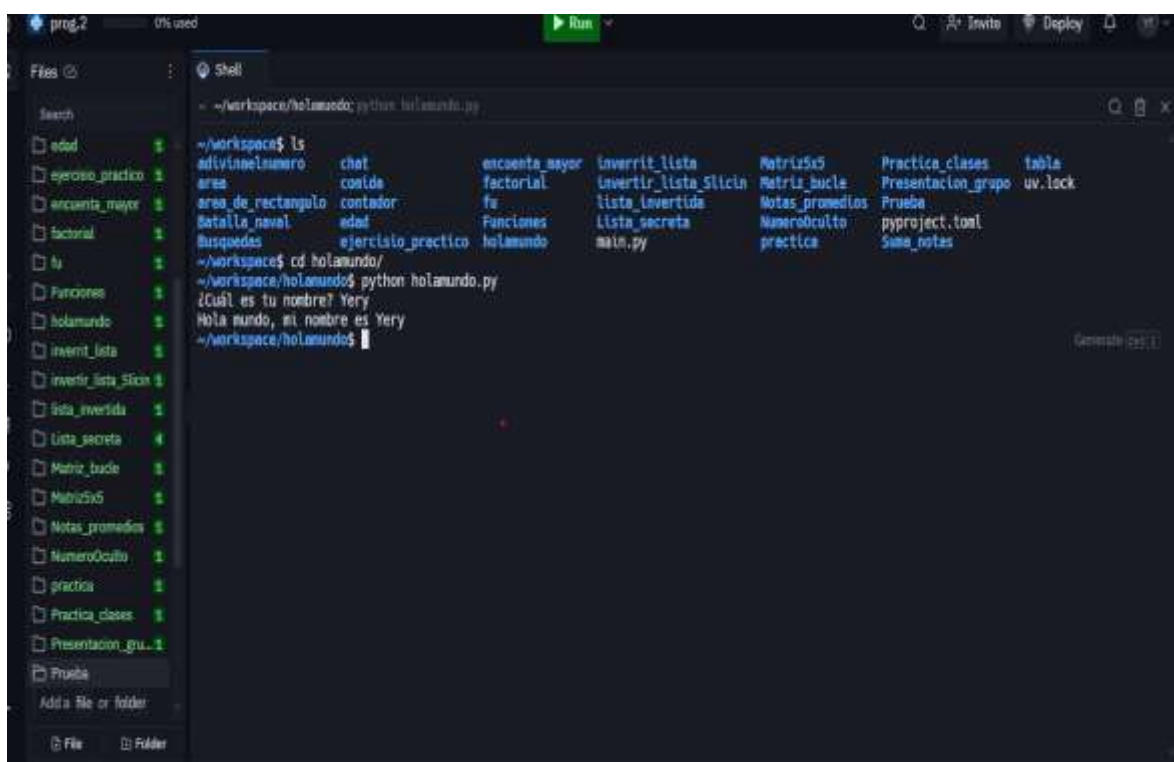
16. Creación de folder Replit.

En esta parte creamos los folder donde vamos a meter vuestros códigos para poder tenerlo de mejor manera organizado para que cada código tenga su folder.



17. Ejecutar códigos mediante la consola “Shell”.

Para poder ejecutar abrimos la consola y escribimos el comando “ls” el cual te va a mostrar todos los folders que uno haya creado ya después de eso para poder entra a un folder se hace con el comando “cd (nombre de folder)” y ya una vez dentro del folder metemos el comando “Python (nombre del código)” y ya de esa manera ya va correr el código y el programa que hallas programado.



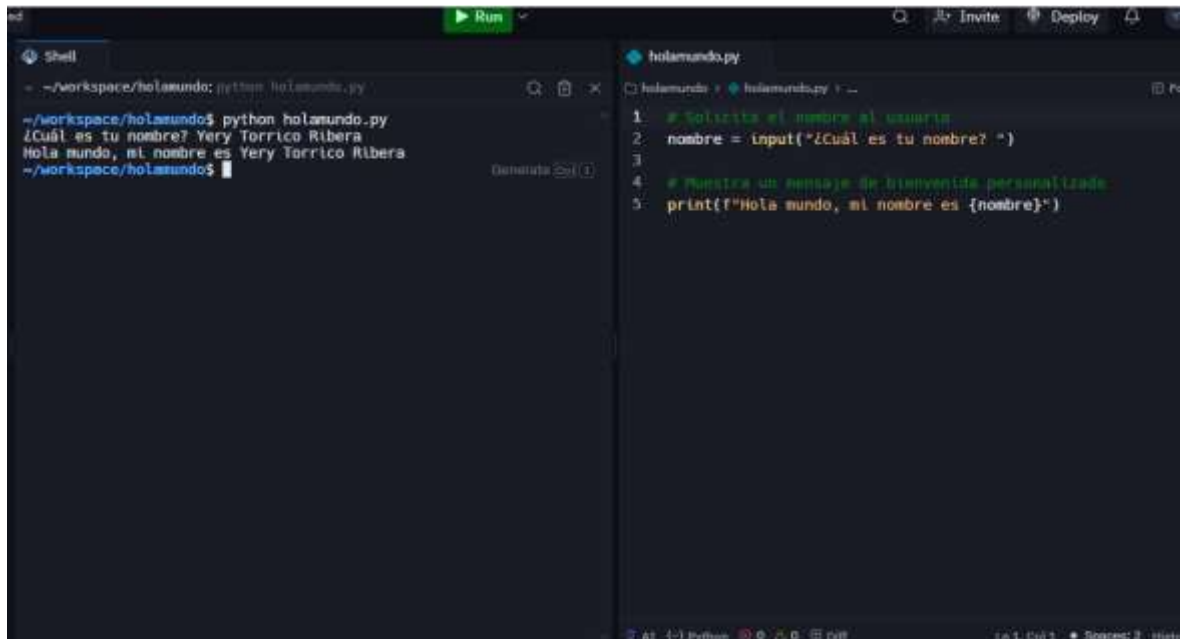
```
~/workspace$ ls
adivinaelnúmero  chat          encuesta_mayor  Invertir_lista  Matriz5x5      Practica_clases  tabla
ejercicio_practico  comida       factorial       Invertir_lista_Slice  Matriz_bucle  Presentacion_grupo  uv.lock
area            contador      fu             lista_invertida  Notas_promedios  Prueba
Batalla_naval    edad          Funciones      Lista_secreta    NumeroOculto     pyproject.toml
Busquedas        ejercicio_practico  holamundo     main.py          practica          Same_notes

~/workspace$ cd holamundo/
~/workspace/holamundo$ python holamundo.py
¿Cuál es tu nombre? Yery
Hola mundo, mi nombre es Yery
~/workspace/holamundo$
```

EJECUCION DE CODIGOS.

1. Código Hola mundo.

En este código se utilizó las funciones “input” y la función “print”. Para poder meter información mediante el teclado usamos la función “input” y para poder ver usamos la función “print” información que nos da el código.



The screenshot shows a code editor with two panels. The left panel, titled 'Shell', displays the execution of a Python script. The right panel, titled 'holamundo.py', shows the source code of the script.

```
~/workspace/holamundo$ python holamundo.py
¿Cuál es tu nombre? Very Torrico Ribera
Hola mundo, mi nombre es Very Torrico Ribera
~/workspace/holamundo$
```

```
1 # Solicita el nombre al usuario
2 nombre = input("¿Cuál es tu nombre? ")
3
4 # Muestra un mensaje de bienvenida personalizada
5 print(f"Hola mundo, mi nombre es {nombre}")
```

2. Código Tabla de multiplicar.

Este código Python solicita al usuario un número, luego usa un bucle “for” para calcular y mostrar la tabla de multiplicar de ese número del 1 al 10. Utiliza “input()” para obtener el número, “int()” para convertirlo a entero y “print()” con f-strings para formatear la salida.

```
1 print("TABLAS DE MULTIPLICAR")
2 print("Introduzca el numero de la tabla:")
3 num_tabla = int(input())
4 print(f"-----Tabla del {num_tabla}-----")
5 for i in range(1, 11):
6     resultado = num_tabla * i
7     print(f"{num_tabla} x {i} = {resultado}")
8
9 print("----Fin del programa----Beymar Ferrufino")
```

C:\Users\USUARIO\PycharmProjects\PythonProject\venv\Scripts>python 3

TABLAS DE MULTIPLICAR

Introduzca el numero de la tabla:

2

-----Tabla del 2-----

2 x 1 = 2

2 x 2 = 4

2 x 3 = 6

2 x 4 = 8

2 x 5 = 10

2 x 6 = 12

2 x 7 = 14

2 x 8 = 16

2 x 9 = 18

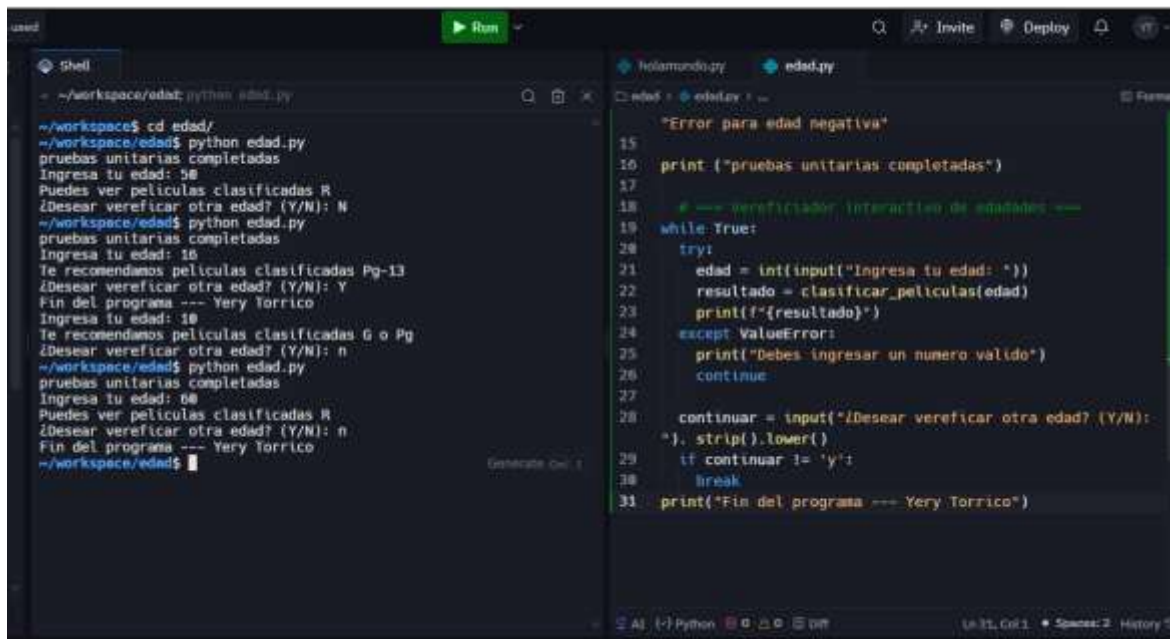
2 x 10 = 20

----Fin del programa----Beymar Ferrufino

Process finished with exit code 0

3. Código Verificador de Edad para Película.

Este script interactivo solicita la edad del usuario para una clasificación de películas. Utiliza un bucle “while True” para permitir múltiples consultas, con un bloque “try-except” para manejar entradas no numéricas. Después de cada clasificación (realizada por una función externa clasificar_peliculas), pregunta al usuario si desea continuar o finalizar el programa.



```
~/workspace/edad: python edad.py
~/workspace/edad$ python edad.py
pruebas unitarias completadas
Ingresa tu edad: 50
Puedes ver películas clasificadas R
¿Desear verificar otra edad? (Y/N): N
~/workspace/edad$ python edad.py
pruebas unitarias completadas
Ingresa tu edad: 10
Te recomendamos películas clasificadas Pg-13
¿Desear verificar otra edad? (Y/N): Y
Fin del programa --- Verry Torrico
Ingresa tu edad: 10
Te recomendamos películas clasificadas G o Pg
¿Desear verificar otra edad? (Y/N): n
~/workspace/edad$ python edad.py
pruebas unitarias completadas
Ingresa tu edad: 60
Puedes ver películas clasificadas R
¿Desear verificar otra edad? (Y/N): n
Fin del programa --- Verry Torrico
~/workspace/edad$
```

```
15 "Error para edad negativa"
16 print("pruebas unitarias completadas")
17
18 # === "verificador" interacción de edades ===
19 while True:
20     try:
21         edad = int(input("Ingresa tu edad: "))
22         resultado = clasificar_peliculas(edad)
23         print(f"{resultado}")
24     except ValueError:
25         print("Debes ingresar un número válido")
26         continue
27
28     continuar = input("¿Desear verificar otra edad? (Y/N): ").strip().lower()
29     if continuar != 'y':
30         break
31 print("Fin del programa --- Verry Torrico")
```

4. Código Tabla de Multiplicar.

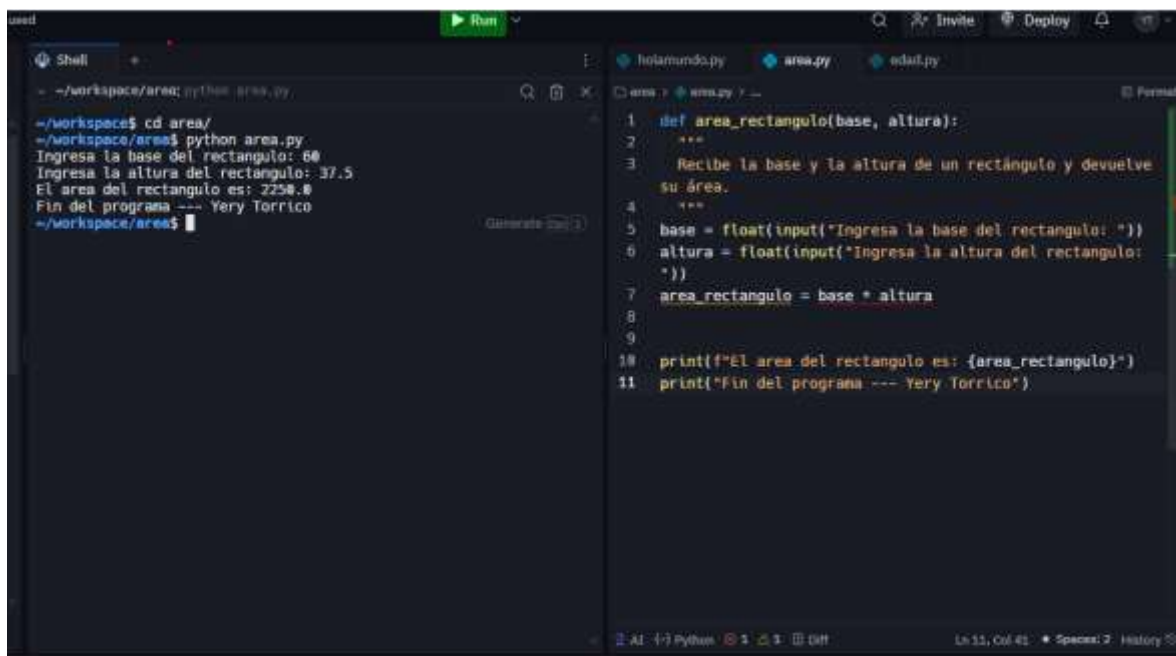
Este programa pide al usuario mediante en “print” un número y luego calcula y muestra su tabla de multiplicar del 1 al 10. Utiliza la función “input” para obtener el número, lo convierte a un valor numérico y luego un ciclo “for” repite las multiplicaciones e imprime cada resultado.

```
1 print("TABLAS DE MULTIPLICAR")
2 print("Introduzca el numero de la tabla:")
3 num_tabla = int(input())
4 print(f"-----Tabla del {num_tabla}-----")
5 for i in range(1, 11):
6     resultado = num_tabla * i
7     print(f"{num_tabla} x {i} = {resultado}")
8
9 print("----Fin del programa----Beymar Ferrufino")
```

C:\Users\USUARIO\PycharmProjects\PythonProject\venv\Sc
TABLAS DE MULTIPLICAR
Introduzca el numero de la tabla:
2
-----Tabla del 2-----
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
----Fin del programa----Beymar Ferrufino
Process finished with exit code 0

5. Código Área de Rectángulo.

En el código se utilizó la función “def” la cual implementamos para definir una función (area_rectangulo), igual utilizamos “input” para poder ingresar los valores de la base y altura, utilizamos igual La función “float” la cual toma esa cadena de texto y la convierte en un número de punto flotante (un número con decimales) y ya de ultimo utilizamos la función “print” para poder mostrar el resultado.



```
Shell
~/workspace/area$ python area.py
~/workspace/area$ cd area/
~/workspace/area$ python area.py
Ingresa la base del rectangulo: 60
Ingresa la altura del rectangulo: 37.5
El area del rectangulo es: 2250.0
Fin del programa --- Yery Torrico
~/workspace/area$
```

```
1 def area_rectangulo(base, altura):
2     """
3     Recibe la base y la altura de un rectángulo y devuelve
4     su área.
5     """
6     base = float(input("Ingresa la base del rectangulo: "))
7     altura = float(input("Ingresa la altura del rectangulo: "))
8     area_rectangulo = base * altura
9
10    print(f"El area del rectangulo es: {area_rectangulo}")
11    print("Fin del programa --- Yery Torrico")
```


6. Código Promedio de notas.

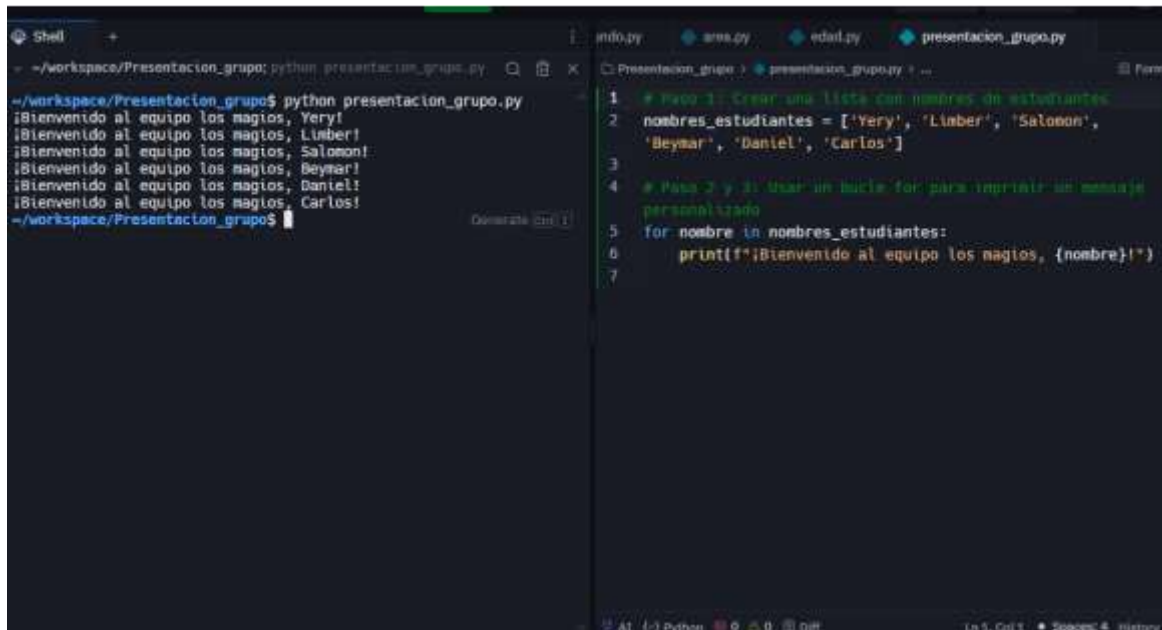
En el código utilizamos “#” para poder implementar la línea de comentario, utilizamos una lista llamada “notas = [80,95,73,60,88]”. Después usamos la función de “sum” que es la función suma y también la función “len” para poder leer las cantidades de valores que tiene la lista y así poder sacar el promedio de notas ya después de haber definido ya podemos poner la operación para poder sacar promedios que es “promedio = suma / Cantidad” y ya utilizamos “print” para poder mostraren pantalla el resultado.

```
1 # Lista con las notas de los parciales
2 notas = [80, 95, 73, 60, 88]
3
4 # Calcular el promedio
5 suma = sum(notas)
6 cantidad = len(notas)
7 promedio = suma / cantidad
8
9 # Mostrar el resultado
10 print("Notas:", notas)
11 print("Promedio:", promedio)
12 print("Fin del programa: Beymar Ferrufino.")
```

C:\Users\USUARIO\PycharmProjects\PythonProject\venv\Scripts\python.exe C:\Users\USUARIO\PycharmProjects\PythonProject\main.py
Notas: [80, 95, 73, 60, 88]
Promedio: 79.2
Fin del programa: Beymar Ferrufino.
Process finished with exit code 0

7. Código Presentación de Grupo.

En el código se utiliza “Listas” para almacenar una colección de nombres de estudiantes de manera ordenada. Bucle “for” para iterar sobre cada nombre en la lista, permitiendo ejecutar un bloque de código para cada uno, “print” para mostrar un mensaje de bienvenida personalizado en la consola para cada estudiante.

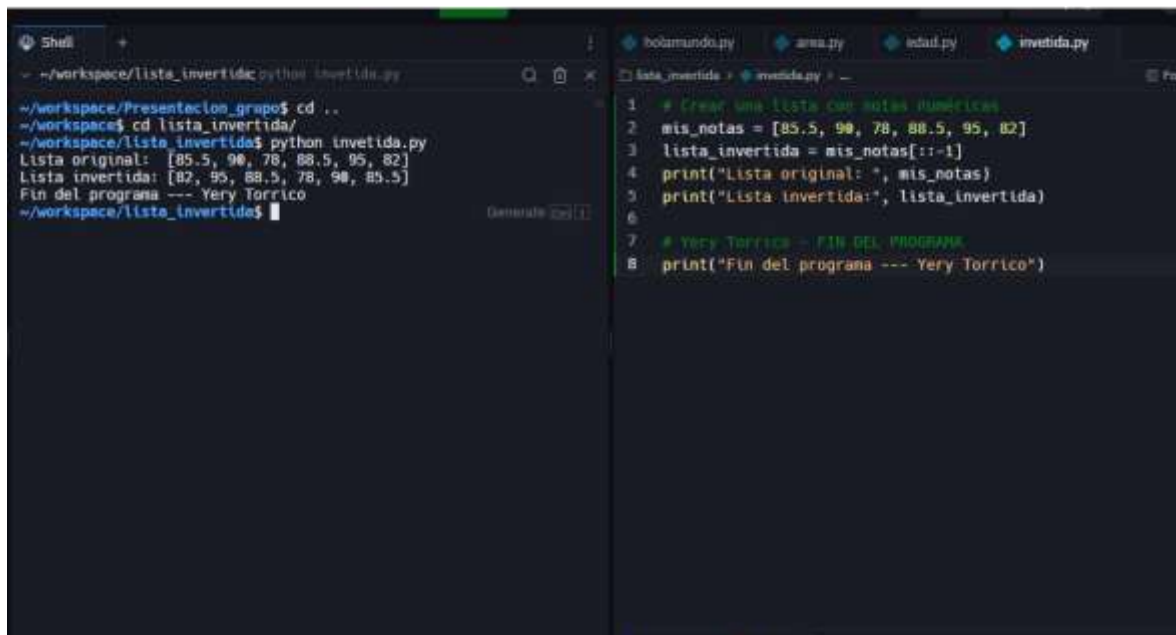


```
~/workspace/Presentacion_grupo$ python presentacion_grupo.py
¡Bienvenido al equipo los magios, Yery!
¡Bienvenido al equipo los magios, Linber!
¡Bienvenido al equipo los magios, Salomon!
¡Bienvenido al equipo los magios, Beymar!
¡Bienvenido al equipo los magios, Daniel!
¡Bienvenido al equipo los magios, Carlos!
~/workspace/Presentacion_grupo$
```

```
1 # Paso 1: Crear una lista con nombres de estudiantes
2 nombres_estudiantes = ['Yery', 'Linber', 'Salomon',
3 'Beymar', 'Daniel', 'Carlos']
4 # Paso 2 y 3: Usar un bucle for para imprimir un mensaje
5 # personalizado
6 for nombre in nombres_estudiantes:
7     print(f'¡Bienvenido al equipo los magios, {nombre}!')
```

8. Código Lista invertida

En este código primeramente tenemos crear un lista en este caso numérica lo cual llamamos “mis_notas”, y después usamos Invertir la lista usando “slicing” y después usamos la función “print” para poder mostrar ambas listas.

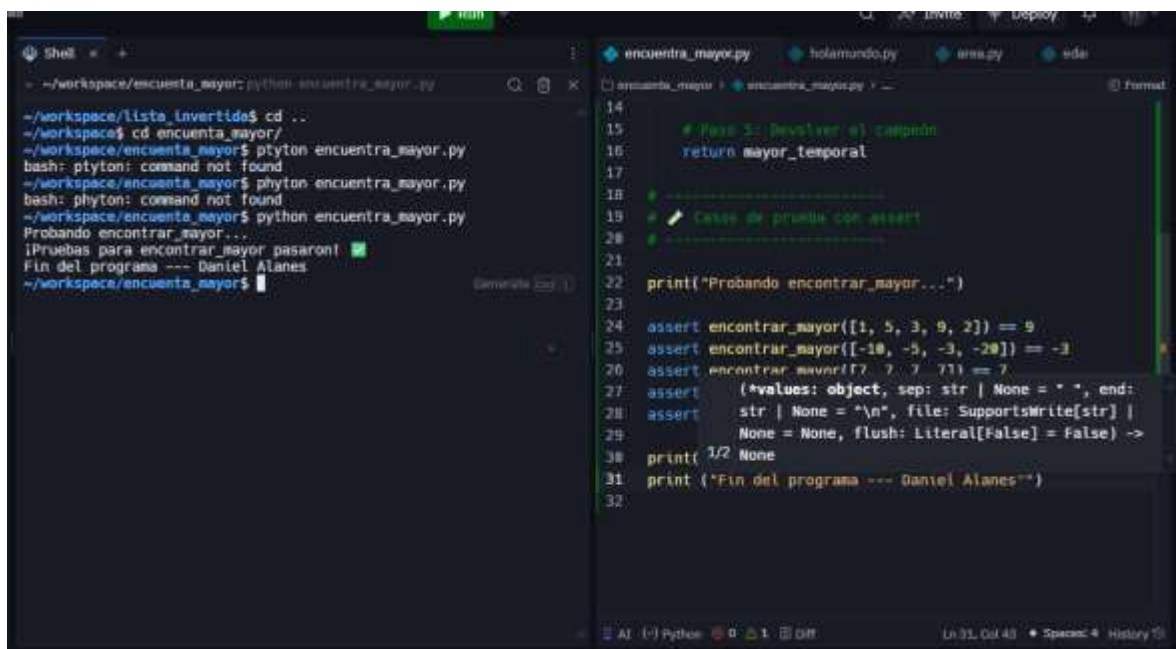


```
~/workspace/Lista_invertida$ python invetida.py
~/workspace/Lista_invertida$ cd ..
~/workspace$ cd lista_invertida/
~/workspace/Lista_invertida$ python invetida.py
Lista original: [85.5, 90, 78, 88.5, 95, 82]
Lista invertida: [82, 95, 88.5, 78, 90, 85.5]
Fin del programa --- Yery Torrico
~/workspace/Lista_invertida$
```

```
1 # Crear una lista con notas numéricas
2 mis_notas = [85.5, 90, 78, 88.5, 95, 82]
3 lista_invertida = mis_notas[::-1]
4 print("Lista original: ", mis_notas)
5 print("Lista invertida:", lista_invertida)
6
7 # Yery Torrico - FIN DEL PROGRAMA
8 print("Fin del programa --- Yery Torrico")
```

9. Código Encuentra al Mayor.

El código utiliza “def” define una función (encontrar mayor) para encapsular la lógica de búsqueda del número más grande, haciéndola reutilizable, bucle “for” itera a través de cada elemento en la lista para comparar y encontrar el valor mayor, “return” devuelve el resultado final (el número mayor encontrado) de la función, “print” muestra mensajes al usuario, incluyendo el estado de las pruebas y el final del programa y “assert” el cual se usa para realizar pruebas unitarias, verificando que la función encontrar mayor produce los resultados esperados para diferentes entradas.



```
~/workspace/encuentra_mayor: python encuentra_mayor.py
~/workspace/encuentra_mayor$ cd ..
~/workspace$ cd encuentra_mayor/
~/workspace/encuentra_mayor$ pyton encuentra_mayor.py
bash: pyton: command not found
~/workspace/encuentra_mayor$ phython encuentra_mayor.py
bash: phython: command not found
~/workspace/encuentra_mayor$ python encuentra_mayor.py
Probando encontrar_mayor...
¡Pruebas para encontrar_mayor pasaron!
Fin del programa --- Daniel Alanes
~/workspace/encuentra_mayor$
```

```
14
15 # Paso 3: Devolver el candidato
16 return mayor_temporal
17
18 # Casos de prueba con assert
19
20
21 print("Probando encontrar_mayor...")
22
23 assert encontrar_mayor([1, 5, 3, 9, 2]) == 9
24 assert encontrar_mayor([-10, -5, -3, -20]) == -3
25 assert encontrar_mayor([7, 7, 7, 7]) == 7
26
27 assert (*values: object, sep: str | None = " ", end:
28 str | None = "\n", file: SupportsWrite[str] |
29 None = None, flush: Literal[False] = False) ->
30 print(1/2 None
31 print("Fin del programa --- Daniel Alanes")
32
```

10. Código Factorial

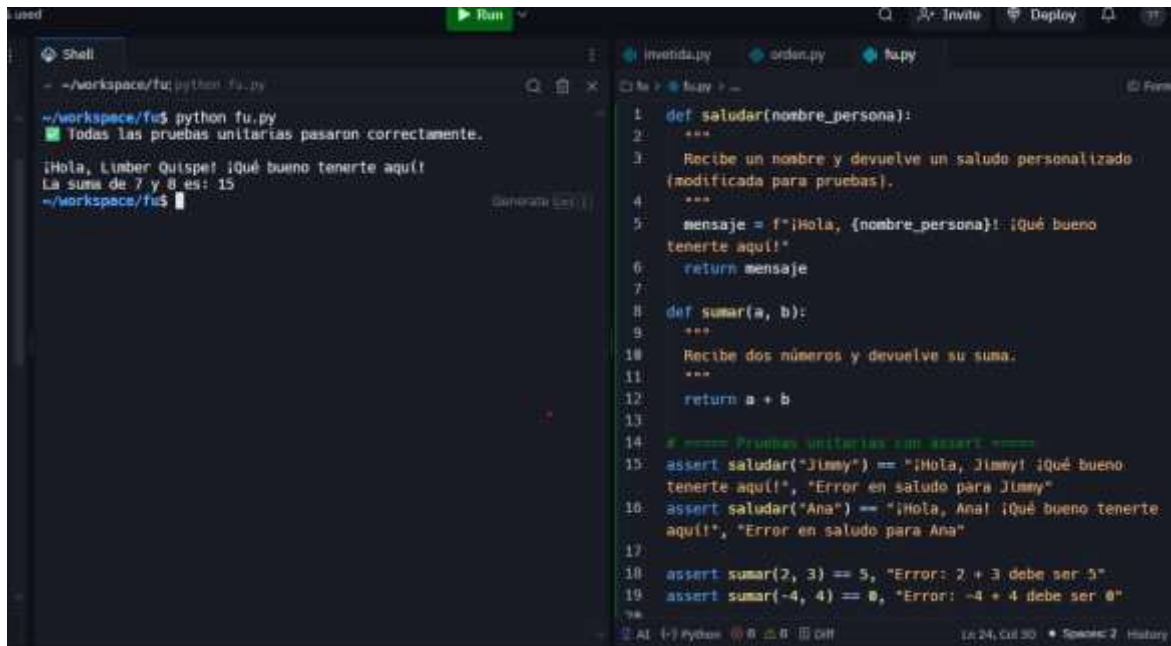
En este código usamos una función “recursiva”. Verifica si el número es negativo y lanza un error si lo es. Si es 0 o 1, retorna 1. Para otros casos, multiplica el número por el factorial del anterior. Usa try-except para manejar errores de entrada.

```
Shell +  
~/workspace/factorial: python factorial.py  
~/workspace/factorial$ python factorial.py  
Ingresa un número para calcular su factorial: 50  
El factorial de 50 es 30414093201713378043612608166064768844377641  
5689605120986000000000000  
~/workspace/factorial$ python factorial.py  
Ingresa un número para calcular su factorial: 10  
El factorial de 10 es 3628800  
~/workspace/factorial$
```

```
factorial.py  
1 # Definición de la función recursiva  
2 def factorial(n):  
3     if n < 0:  
4         raise ValueError("El factorial no está definido para  
números negativos")  
5     elif n == 0 or n == 1:  
6         return 1  
7     else:  
8         return n * factorial(n - 1)  
9  
10 # Solicitar al usuario un número  
11 try:  
12     numero = int(input("Ingresa un número para calcular su  
factorial: "))  
13     resultado = factorial(numero)  
14     print(f"El factorial de {numero} es {resultado}")  
15 except ValueError as e:  
16     print(e)  
17  
18
```

11. Código Funciones.

En este código implementamos los “assert” para poder para verificar si una condición es verdadera



```
Shell
~/workspace/fu$ python fu.py
~ Todas las pruebas unitarias pasaron correctamente.
¡Hola, Limber Quispe! ¡Qué bueno tenerte aquí!
La suma de 7 y 8 es: 15
~/workspace/fu$

def saludar(nombre_persona):
    """
    Recibe un nombre y devuelve un saludo personalizado
    (modificada para pruebas).
    """
    mensaje = f"¡Hola, {nombre_persona}! ¡Qué bueno
    tenerte aquí!"
    return mensaje

def sumar(a, b):
    """
    Recibe dos números y devuelve su suma.
    """
    return a + b

# ===== Pruebas unitarias con assert =====
assert saludar("Jimmy") == "¡Hola, Jimmy! ¡Qué bueno
tenerte aquí!", "Error en saludo para Jimmy"
assert saludar("Ana") == "¡Hola, Ana! ¡Qué bueno tenerte
aquí!", "Error en saludo para Ana"

assert sumar(2, 3) == 5, "Error: 2 + 3 debe ser 5"
assert sumar(-4, 4) == 0, "Error: -4 + 4 debe ser 0"
```

