



PRÁCTICO 1

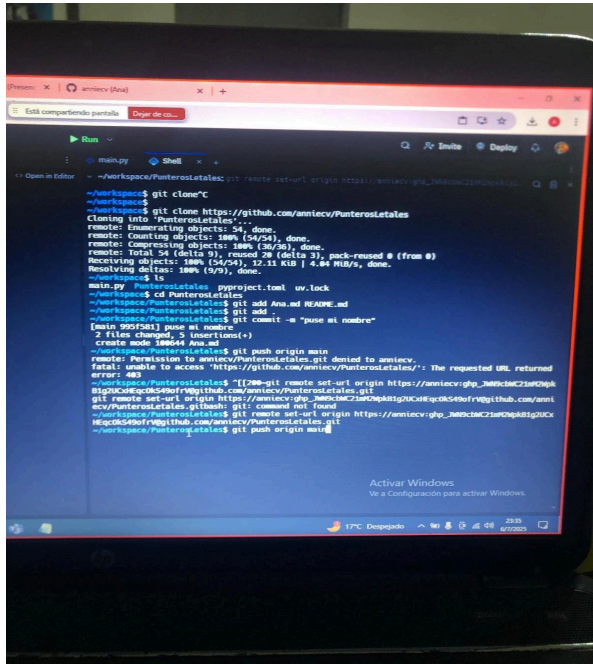
Jorge Luis Esteves Salas
Fernando Navia
Joel Dalton Montero
Ana Laura Cuellar
Weimar Valda
Leonel Eguez Camargo
ALEJANDRO HURTADO

Dirigido por el docente:
JIMMY REQUENA LLORENTY
Materia:
Programación II

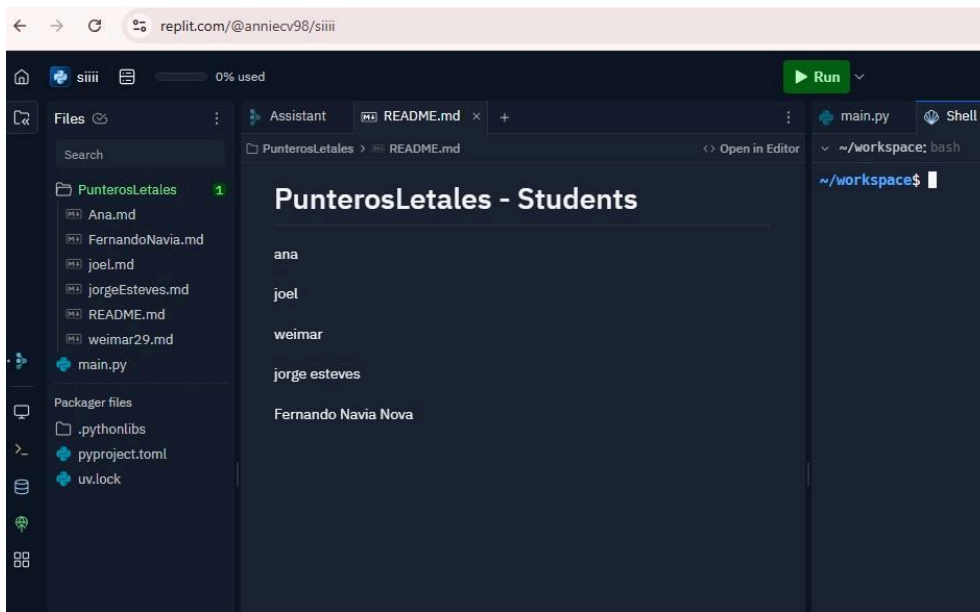
ACTIVIDAD 1

Ana Cuellar, 06/07/2025.

Primer Ejercicio desde Replit.



```
~/workspace$ git clone https://github.com/anniecv/PunterosLetales
Cloning into 'PunterosLetales'...
remote: Enumerating objects: 54, done.
remote: Counting objects: 100% (36/36), done.
remote: Total 54 (delta 3), reused 28 (delta 3), pack-reused 0 (from #)
Receiving objects: 100% (54/54), 12.11 KiB | 4.04 MiB/s, done.
Resolving deltas: 100% (19/19), done.
~/workspace$ ls
main.py  PunterosLetales  pyproject.toml  uv.lock
~/workspace$ cd PunterosLetales
~/workspace/PunterosLetales$ git add Ana.md README.md
~/workspace/PunterosLetales$ git commit -m "puse mi nombre"
[main 995f81] puse mi nombre
2 files changed, 5 insertions(+)
create mode 100644 Ana.md
~/workspace/PunterosLetales$ git push origin main
remote: Permission to anniecv/PunterosLetales.git denied to anniecv.
fatal: unable to access 'https://github.com/anniecv/PunterosLetales/': The requested URL returned error: 403
~/workspace/PunterosLetales$ git remote set-url origin https://anniecv:ghp_3M6t3Mc2se0qpb8ip9iKd6q6K549frv9@github.com:anniecv/PunterosLetales.git
~/workspace/PunterosLetales$ git remote set-url origin https://anniecv:ghp_3M6t3Mc2se0qpb8ip9iKd6q6K549frv9@github.com:anniecv/PunterosLetales.git
~/workspace/PunterosLetales$ git push origin main
```

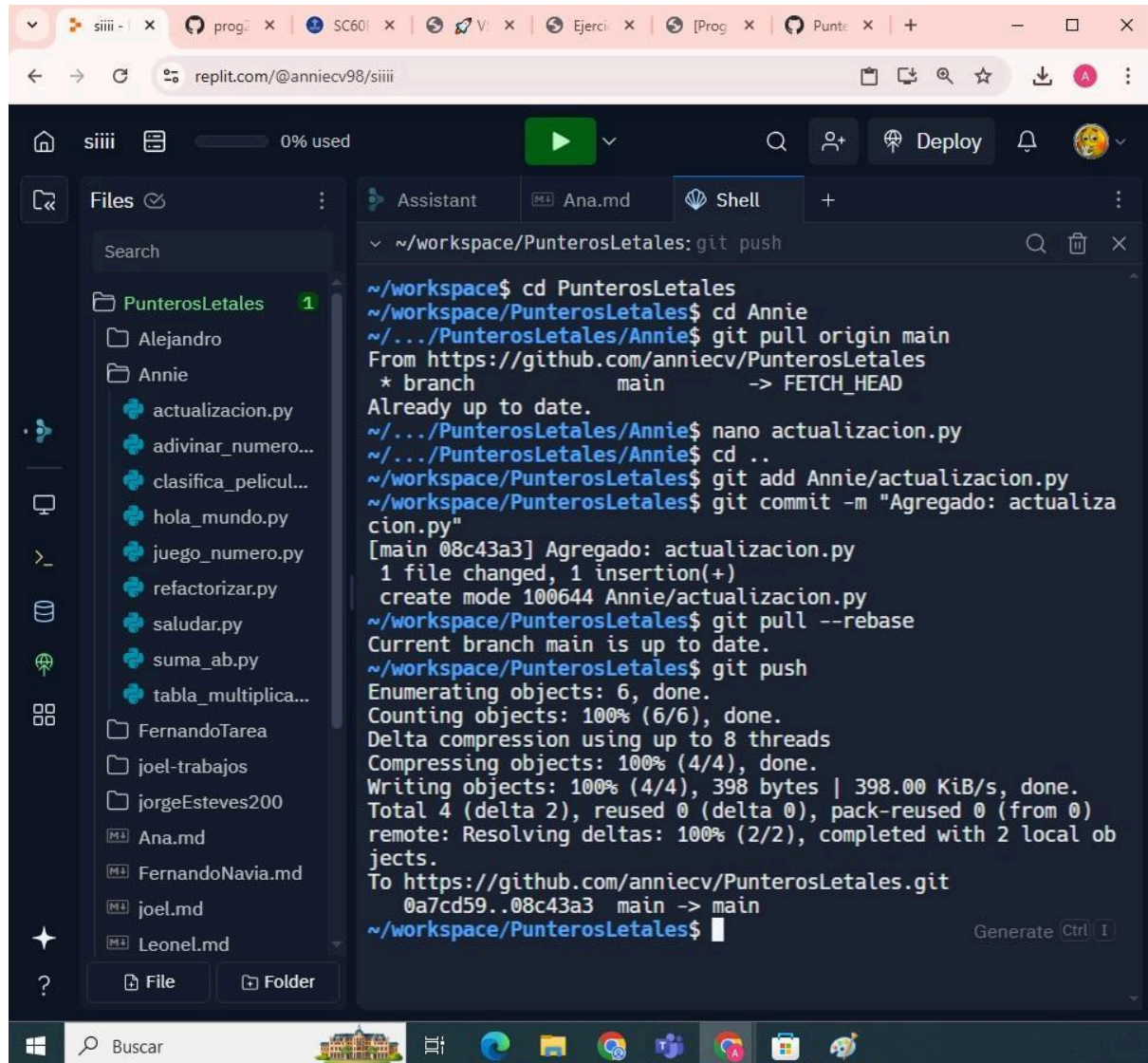


Hice un archivo con mi nombre, fue la primera vez que use comandos como git commit, git push, git add. Un compañero me guió gracias a Dios y así pude subir mi carpeta. Me pareció interesante ver como se puede colaborar de forma organizada y simultánea en un mismo proyecto, ya que cada uno puede subir sus avances sin pisar el trabajo del otro.

Fecha y hora actual: PM-07-08 17:20:58

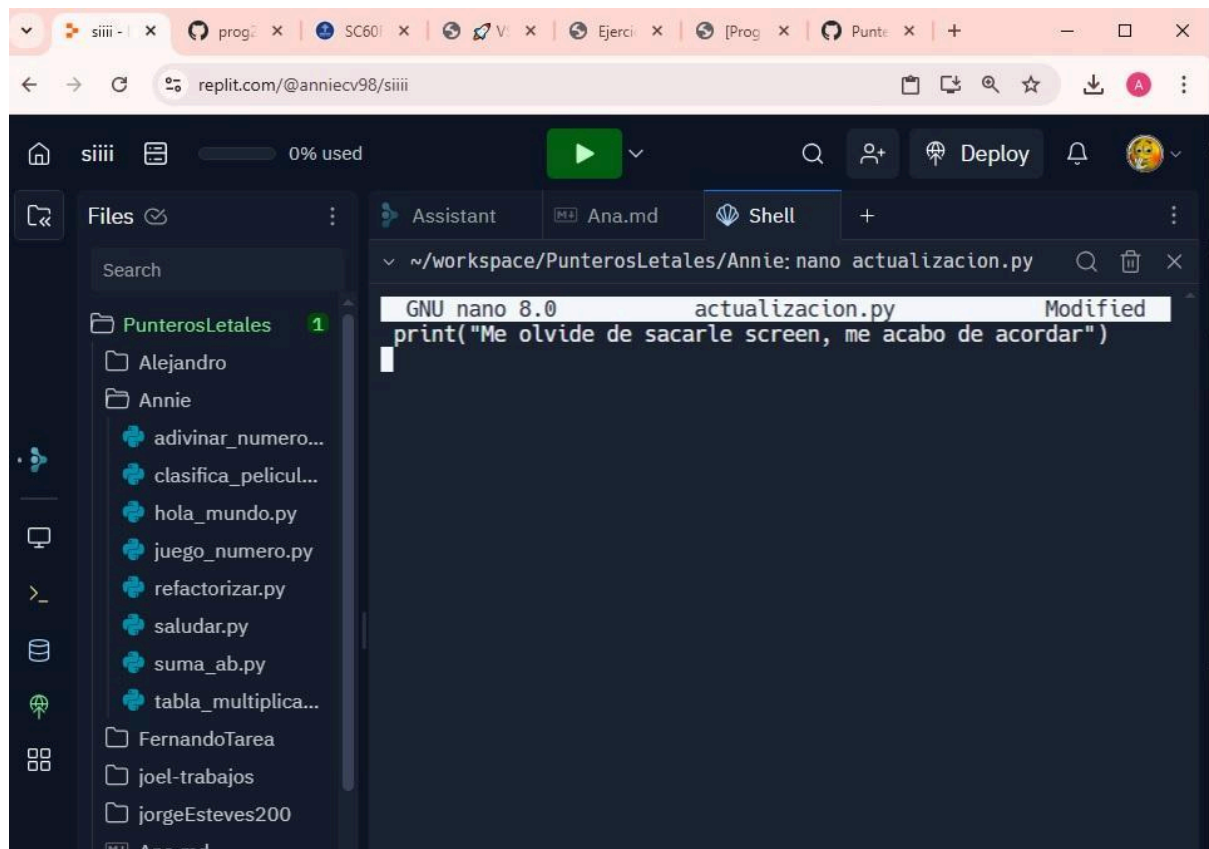
Actualización.

Copiamos y pegamos los ejercicios del Ingeniero. Subiendo a la Carpeta que tiene mi nombre cada uno de ellos, usando Replit.



```
~/workspace/PunterosLetales: git push

~/workspace$ cd PunterosLetales
~/workspace/PunterosLetales$ cd Annie
~/.../PunterosLetales/Annie$ git pull origin main
From https://github.com/anniecv/PunterosLetales
* branch          main          -> FETCH_HEAD
Already up to date.
~/.../PunterosLetales/Annie$ nano actualizacion.py
~/.../PunterosLetales/Annie$ cd ..
~/workspace/PunterosLetales$ git add Annie/actualizacion.py
~/workspace/PunterosLetales$ git commit -m "Agregado: actualizacion.py"
[main 08c43a3] Agregado: actualizacion.py
1 file changed, 1 insertion(+)
create mode 100644 Annie/actualizacion.py
~/workspace/PunterosLetales$ git pull --rebase
Current branch main is up to date.
~/workspace/PunterosLetales$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 398 bytes | 398.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/anniecv/PunterosLetales.git
0a7cd59..08c43a3 main -> main
~/workspace/PunterosLetales$
```



Pero resulta y resalta que me olvide sacarle screen a todo eso jajaja, entonces hice uno con print para mostrarle evidencia al Ingeniero.

Ana Laura Cuéllar – 09/07/2025 – 17:22

Claro, aquí te dejo el texto formateado en estilo **APA**, listo para que puedas pegarlo en tu Word. Le agregué título, subtítulos, una redacción formal y cuidé la estructura para que sea adecuada en un trabajo académico:

Investigación: Argumentos por Posición vs. Argumentos por Nombre (Keyword Arguments) en Python

Cuando creamos funciones en Python, a veces queremos que esas funciones reciban datos desde afuera para procesarlos o utilizarlos. Esos datos que se le entregan a una función se llaman **argumentos**. Por otro lado, los **parámetros** son los nombres que se usan dentro de la función para referirse a esos datos. Por ejemplo, en la definición `def saludar(nombre):`, el término *nombre* es un parámetro. Al llamar a la función, como en `saludar("Ana")`, el valor *"Ana"* es el argumento.

En Python existen dos formas principales de pasar argumentos a las funciones: **por posición** y **por nombre**.

- **Argumentos por posición:** Consiste en pasar los valores en el mismo orden en que los parámetros fueron definidos en la función. Por ejemplo, si una función espera los parámetros (`nombre`, `edad`), al llamarla como `funcion("Juan", 30)`, se asignará `"Juan"` al parámetro `nombre` y `30` al parámetro `edad`.
- **Argumentos por nombre (keyword arguments):** En este caso se indica explícitamente qué valor corresponde a cada parámetro, sin importar el orden en que se coloquen. Por ejemplo, `funcion(edad=30, nombre="Juan")` es una llamada válida y asignará correctamente los valores a cada parámetro.

La elección entre uno y otro método depende de la claridad y la cantidad de parámetros. Los argumentos por nombre se usan cuando se desea mayor legibilidad o cuando la función tiene muchos parámetros para evitar errores por el orden. Los argumentos por posición son más comunes en funciones con pocos parámetros donde el orden es fácil de recordar.

¿Qué es una función recursiva?

Una función recursiva es aquella que se llama a sí misma durante su ejecución. Se emplea cuando un problema puede resolverse dividiéndolo en partes más pequeñas y aplicando la misma solución a cada una.

Por ejemplo, para contar hacia atrás desde un número hasta llegar a cero, en lugar de escribir múltiples líneas de código, la función puede llamarse a sí misma hasta alcanzar la condición de corte. Esta condición de salida es fundamental para evitar que la recursión sea infinita.

Las funciones recursivas son útiles para resolver problemas como calcular factoriales, recorrer estructuras de datos (como carpetas) o dividir problemas complejos en subproblemas similares más manejables.

Métodos importantes de las listas en Python

Las listas en Python son estructuras que permiten almacenar varios elementos dentro de una sola variable. Python ofrece métodos propios para manipular listas de forma eficiente, entre ellos:

- `append(elemento)`: Añade un elemento al final de la lista. Es útil cuando

se desea ir agregando datos uno a uno sin conocer la cantidad total de antemano.

Ejemplo:

```
mi_lista = [1, 2, 3]
```

```
mi_lista.append(4) # Resultado: [1, 2, 3, 4]
```

- `pop()`: Elimina y devuelve el último elemento de la lista. Es útil para manejar la lista como una pila (LIFO, último en entrar, primero en salir).

Ejemplo:

```
mi_lista = [10, 20, 30]
```

```
mi_lista.pop() # Elimina 30, la lista queda [10, 20]
```

- `sort()`: Ordena la lista. Si contiene números, los ordena de menor a mayor; si contiene cadenas, las ordena alfabéticamente.

Ejemplo:

```
mi_lista = [3, 1, 2]
```

```
mi_lista.sort() # Resultado: [1, 2, 3]
```

¿Qué son los índices negativos en listas?

En Python, además de acceder a una lista con índices normales (0, 1, 2, ...), también se pueden usar índices negativos para acceder a los elementos desde el final de la lista:

- `-1` corresponde al último elemento.
- `-2` al penúltimo.
- `-3` al antepenúltimo, y así sucesivamente.

Esta forma es útil cuando no se conoce la longitud de la lista pero se quiere acceder

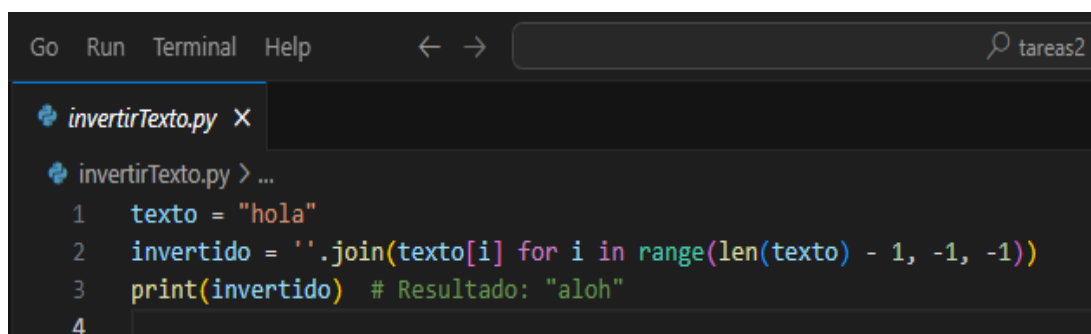
a los elementos finales sin hacer cálculos adicionales.

Ejemplo:

```
mi_lista = ['a', 'b', 'c', 'd']
```

```
print(mi_lista[-1]) # Resultado: 'd'
```

Ana Laura Cuellar – 10/07/2025 – 18:38

A screenshot of a code editor window with a dark theme. The window has a menu bar with 'Go', 'Run', 'Terminal', and 'Help'. Below the menu bar is a search bar with the text 'tareas2'. The editor shows a file named 'invertirTexto.py'. The code inside the file is as follows:

```
invertirTexto.py > ...
1 texto = "hola"
2 invertido = ''.join(texto[i] for i in range(len(texto) - 1, -1, -1))
3 print(invertido) # Resultado: "aloh"
4
```

1. Generación del rango inverso con `range(len(texto) - 1, -1, -1)`

Esta instrucción utiliza la función incorporada `range()` para generar una secuencia de números enteros que comienza en el índice del último carácter de la cadena y termina antes de llegar a -1, decrementándose en pasos de uno. En otras palabras, `len(texto) - 1` calcula la posición del último carácter (dado que los índices comienzan en 0), el valor `-1` indica el límite inferior excluido, y el paso `-1` especifica que la secuencia se recorra de forma decreciente. Así, para una cadena de longitud 4, esta función genera la secuencia `[3, 2, 1, 0]`, que permite acceder a los caracteres de la cadena en orden inverso (Python Software Foundation, 2023).

2. Comprensión de generador `texto[i] for i in range(...)`

Se emplea una expresión generadora que itera sobre la secuencia de índices creada en el punto anterior para extraer cada carácter de la cadena `texto` en orden inverso. Esta expresión produce una serie de caracteres individuales, cada uno obtenido mediante indexación, es decir, accediendo a la posición `i` en la cadena. Este método resulta eficiente porque genera los elementos uno a uno bajo demanda, en lugar de crear una lista completa en memoria (Lutz, 2013).

3. Método de cadena `' '.join(...)`

El método `join()` se utiliza para concatenar todos los caracteres generados por la comprensión en una única cadena de texto. Al invocarse sobre una cadena vacía `''`, indica que los elementos se unen sin separadores adicionales entre ellos. Esto es fundamental para reconstruir la cadena invertida a partir de sus caracteres individuales. En resumen, `join()` convierte un iterable de caracteres en una cadena continua (Beazley, 2009).

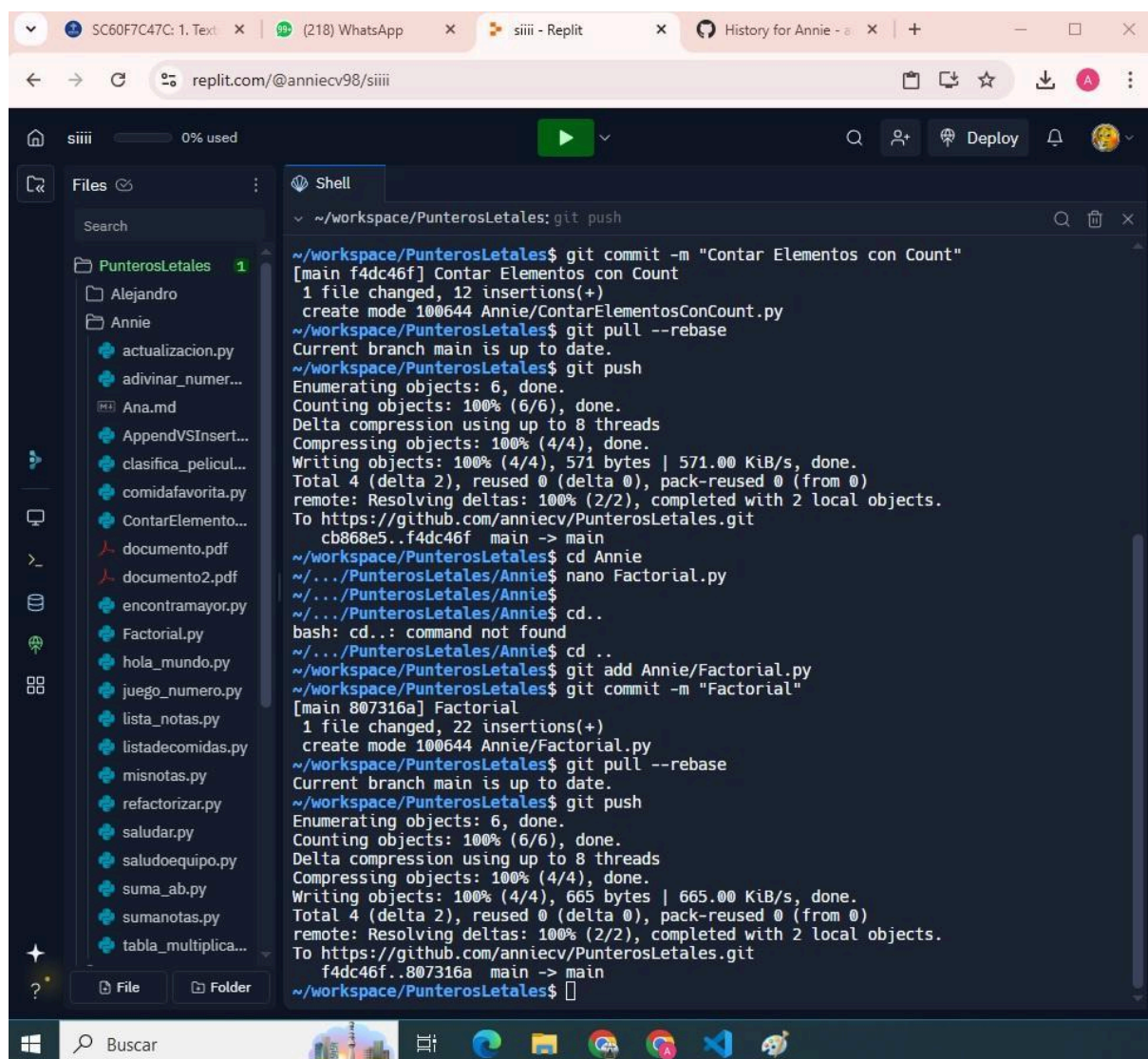
Referencias

Beazley, D. (2009). *Python Essential Reference* (4th ed.). Addison-Wesley.

Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media.

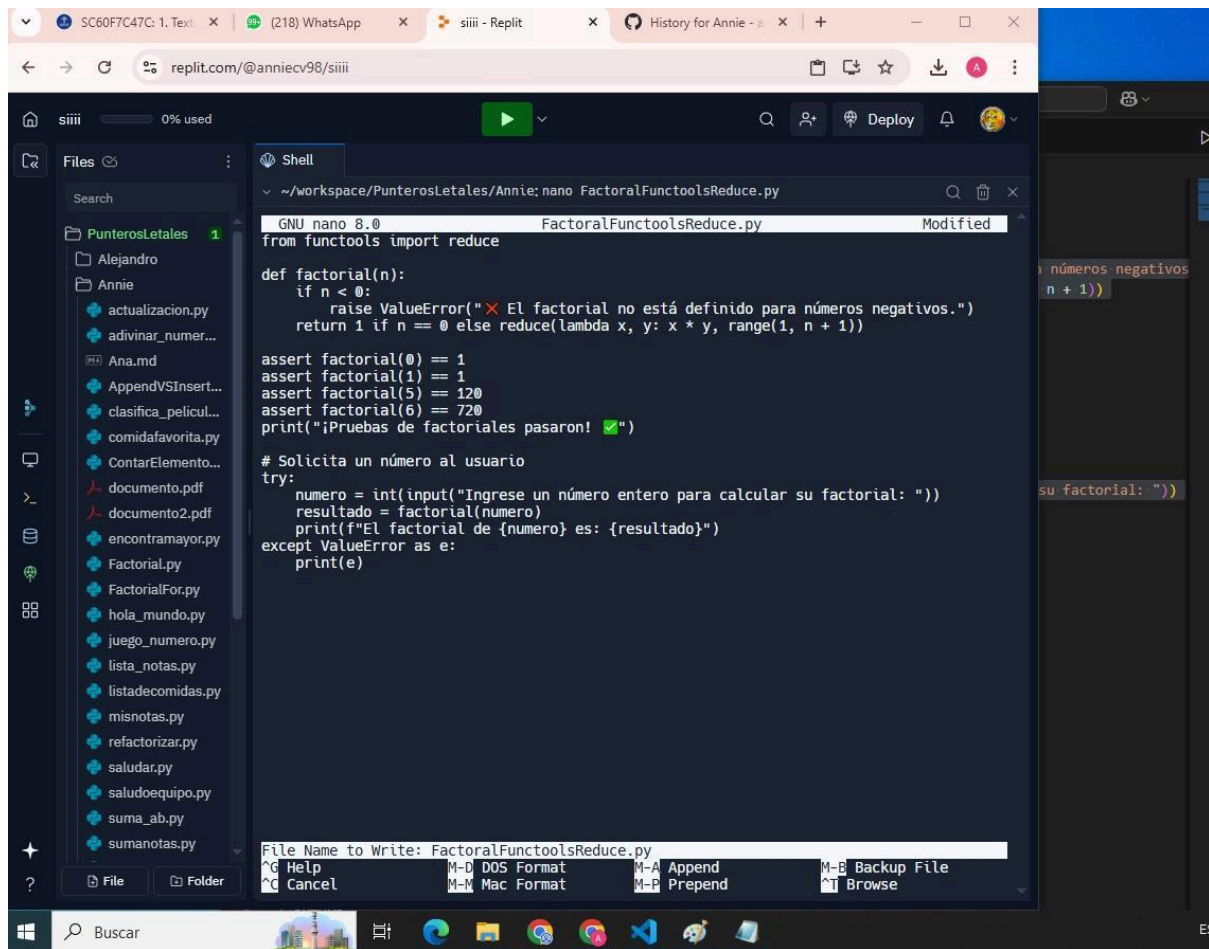
Python Software Foundation. (2023). *The Python Language Reference*, release 3.11. <https://docs.python.org/3/reference/>

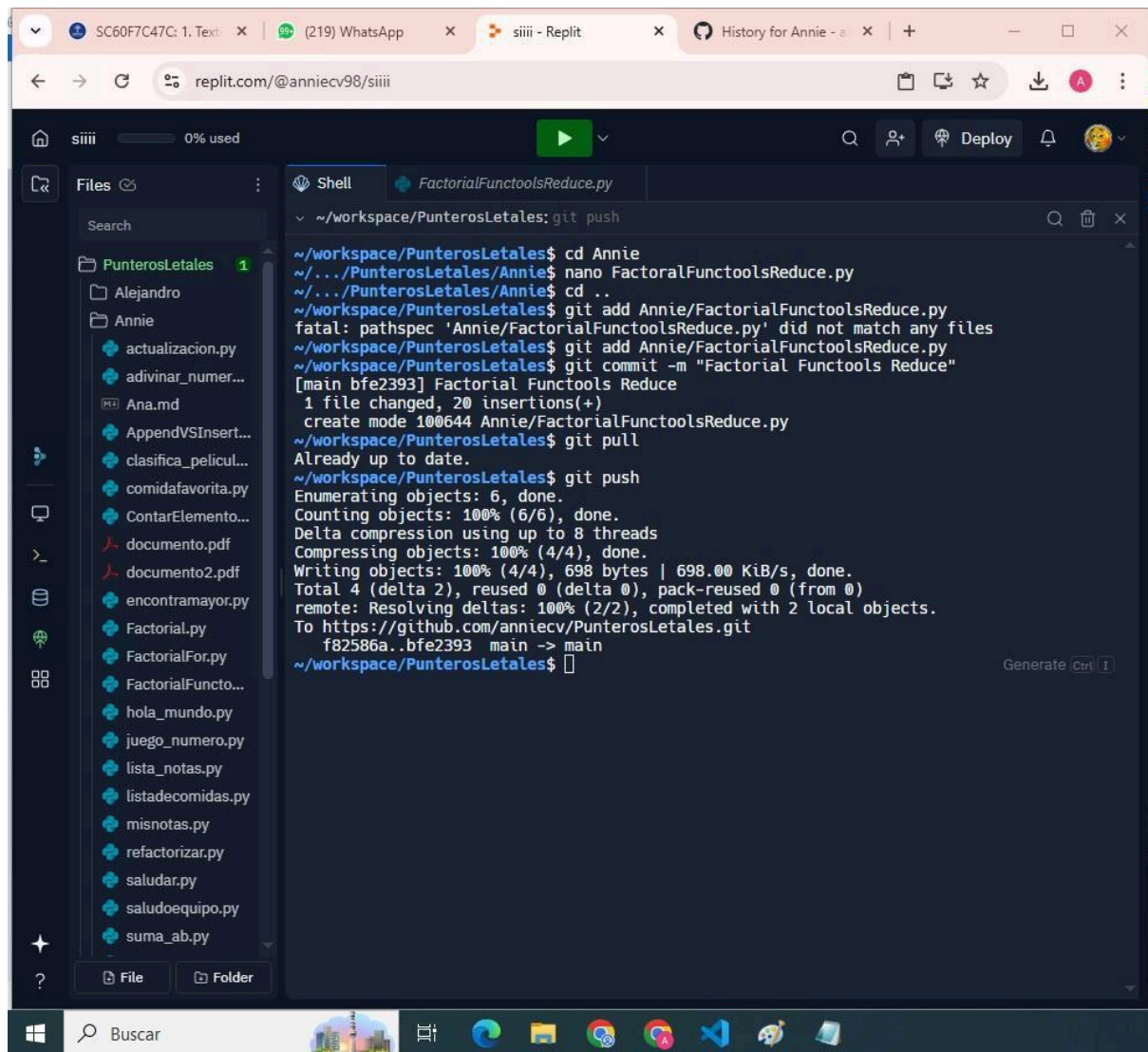
Ana Laura Cuellar – 11/07/2025 – 17:26

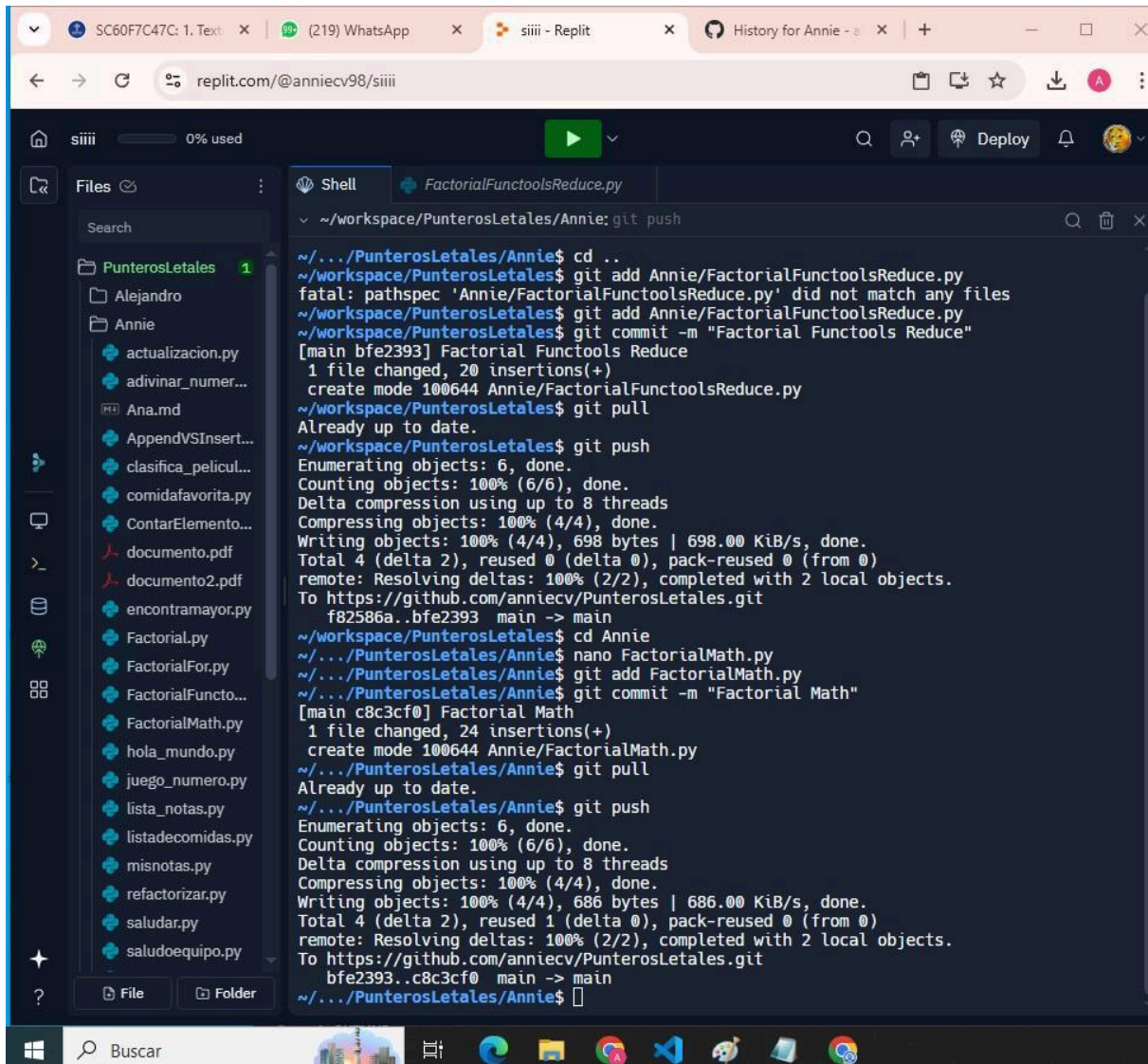


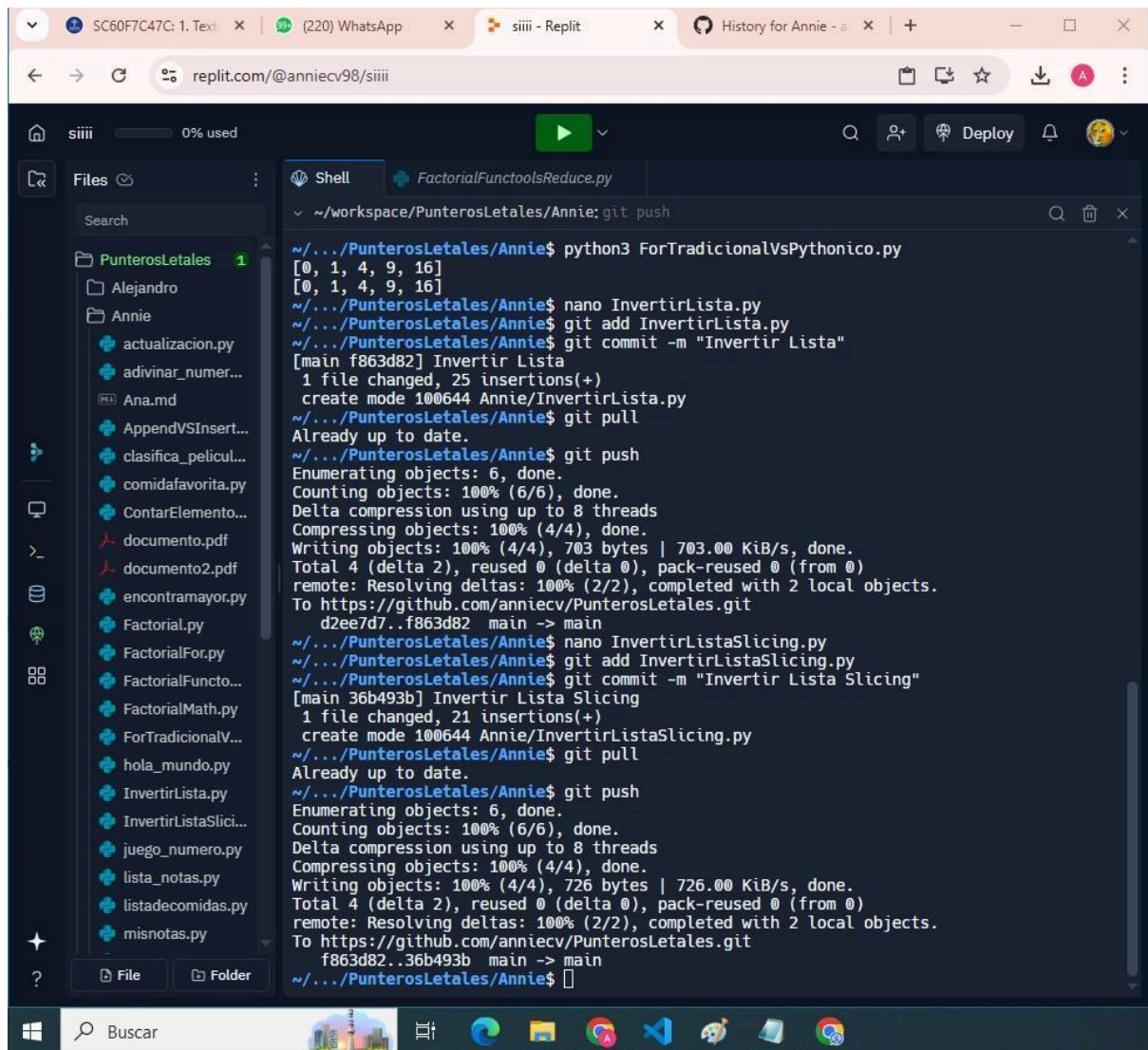
The screenshot shows a Replit terminal window with the following content:

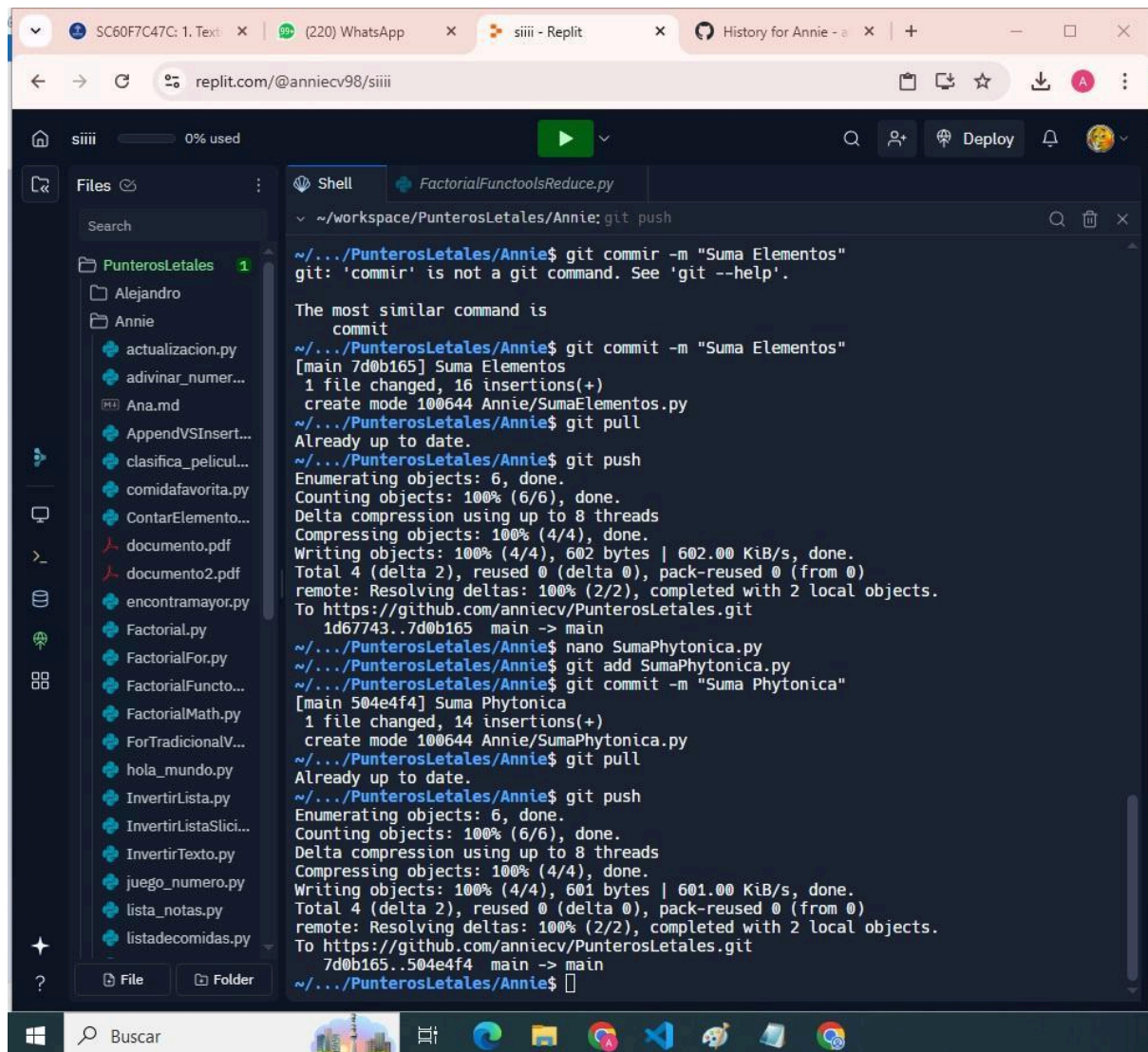
```
~/workspace/PunterosLetales$ git push
~/workspace/PunterosLetales$ git commit -m "Contar Elementos con Count"
[main f4dc46f] Contar Elementos con Count
1 file changed, 12 insertions(+)
create mode 100644 Annie/ContarElementosConCount.py
~/workspace/PunterosLetales$ git pull --rebase
Current branch main is up to date.
~/workspace/PunterosLetales$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 571 bytes | 571.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/anniecv/PunterosLetales.git
cb868e5..f4dc46f main -> main
~/workspace/PunterosLetales$ cd Annie
~/.../PunterosLetales/Annie$ nano Factorial.py
~/.../PunterosLetales/Annie$ cd..
bash: cd.: command not found
~/.../PunterosLetales/Annie$ cd ..
~/workspace/PunterosLetales$ git add Annie/Factorial.py
~/workspace/PunterosLetales$ git commit -m "Factorial"
[main 807316a] Factorial
1 file changed, 22 insertions(+)
create mode 100644 Annie/Factorial.py
~/workspace/PunterosLetales$ git pull --rebase
Current branch main is up to date.
~/workspace/PunterosLetales$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 665 bytes | 665.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/anniecv/PunterosLetales.git
f4dc46f..807316a main -> main
~/workspace/PunterosLetales$
```









SC60F7C47C: 1. Texto guía | UPI (219) WhatsApp PunterosLetales/Annie at main

github.com/anniecv/PunterosLetales/tree/main/Annie

anniecv / PunterosLetales

Type to search

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

main

Go to file

Alejandro

Annie

Ana.md

AppendVInsert.py

ContarElementosConCount.py

Factorial.py

FactorialFor.py

FactorialFunctoolsReduce.py

FactorialMath.py

ForTradicionalVsPythonico.py

InvertirLista.py

InvertirListaSlicing.py

InvertirTexto.py

SumaElementos.py

SumaPythonica.py

actualizacion.py

adivinar_numero.py

clasifica_pelicula.py

comidafavorita.py

PunterosLetales / Annie

anniecv Borrado por falla ortográfica. b8e1c6a · 4 minutes ago History

Name	Last commit message	Last commit date
..		
Ana.md	documento jorge e	yesterday
AppendVInsert.py	AppendVInsert.py	1 hour ago
ContarElementosConCount.py	Contar Elementos con Count	1 hour ago
Factorial.py	Factorial	1 hour ago
FactorialFor.py	Factorial For	1 hour ago
FactorialFunctoolsReduce.py	Factorial Functools Reduce	54 minutes ago
FactorialMath.py	Factorial Math	51 minutes ago
ForTradicionalVsPythonico.py	For Tradicional Vs Pythonico	45 minutes ago
InvertirLista.py	Invertir Lista	40 minutes ago
InvertirListaSlicing.py	Invertir Lista Slicing	38 minutes ago
InvertirTexto.py	Invertir Texto	36 minutes ago
SumaElementos.py	Suma Elementos	33 minutes ago
SumaPythonica.py	Suma Pythonica	28 minutes ago

Buscar

Ana Laura Cuellar – 11/07/2025 – 17:29