

UNIVERSIDAD PRIVADA

DOMINGO SAVIO

“ PROGRAMACIÓN II ”

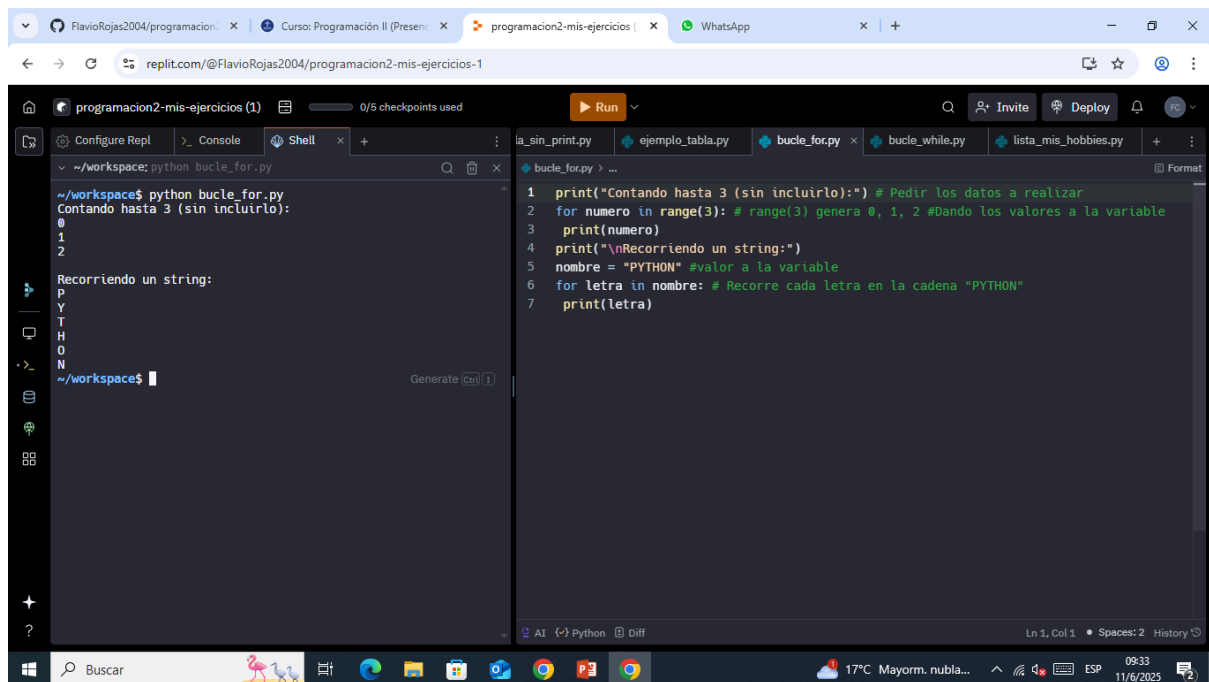
“MUESTRAS DE EJECUCIÓN DE PROGRAMAS REALIZADOS”

Estudiante: Flavio Cesar Rojas Vargas

Ingeniero: Jimmy Nataniel Requena Llorentty

**SANTA CRUZ - BOLIVIA
JUNIO 2025**

1. Bucle For



```
~/workspace$ python bucle_for.py
Contando hasta 3 (sin incluirlo):
0
1
2

Recorriendo un string:
P
Y
T
H
O
N

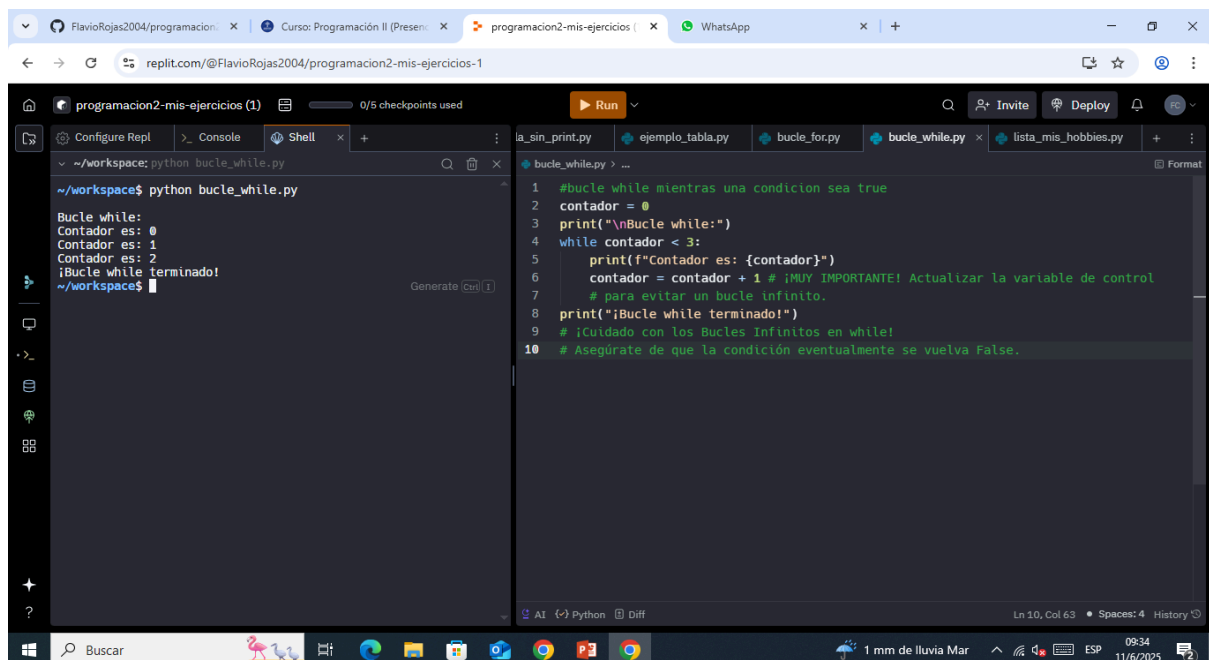
~/workspace$
```

```
1 print("Contando hasta 3 (sin incluirlo):") # Pedir los datos a realizar
2 for numero in range(3): # range(3) genera 0, 1, 2 #Dando los valores a la variable
3     print(numero)
4 print("\nRecorriendo un string:")
5 nombre = "PYTHON" #valor a la variable
6 for letra in nombre: # Recorre cada letra en la cadena "PYTHON"
7     print(letra)
```

Reseña y aprendizaje

El código realizado es un bucle repetitivo al recorrer cada caso en cadena. De este tipo de código aprendí a hacer secuencias para no hacer muy tedioso al reescribir una línea de código

2. Bucle While



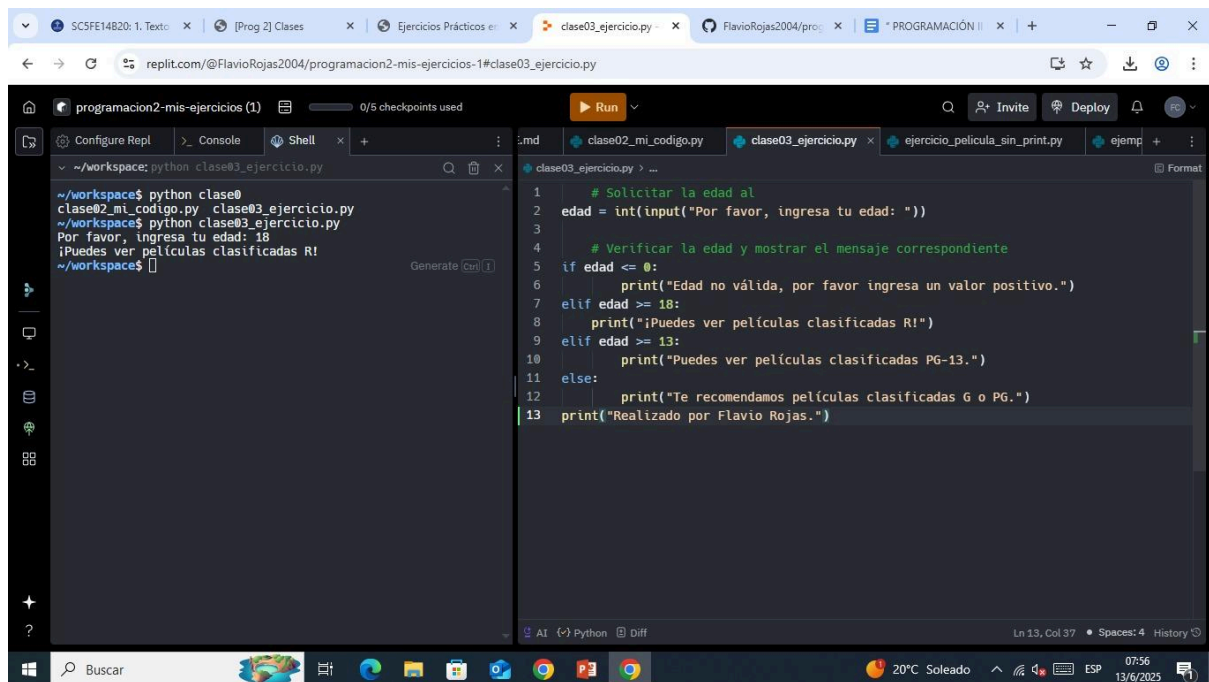
```
~/workspace$ python bucle_while.py
Bucle while:
Contador es: 0
Contador es: 1
Contador es: 2
¡Bucle while terminado!
~/workspace$
```

```
1 #bucle while mientras una condicion sea true
2 contador = 0
3 print("\nBucle while:")
4 while contador < 3:
5     print(f"Contador es: {contador}")
6     contador = contador + 1 # ¡MUY IMPORTANTE! Actualizar la variable de control
7     # para evitar un bucle infinito.
8 print("¡Bucle while terminado!")
9 # ¡Cuidado con los Bucles Infinitos en while!
10 # Asegúrate de que la condición eventualmente se vuelva False.
```

Reseña y aprendizaje:

Este código es útil para hacer bucles como contadores, y con el cual hay que tener un poco de cuidado ya que podría llegar a convertirse en un bucle infinito. El aprendizaje de este código fue poder realizar contadores usando la variable true.

3. Estructuras de control



The screenshot shows a Replit IDE window with a Python script named `clase03_ejercicio.py`. The script prompts the user for their age and uses conditional logic to provide feedback. The console shows the user inputting '18' and receiving the message '¡Puedes ver películas clasificadas R!'.

```
1 # Solicitar la edad al
2 edad = int(input("Por favor, ingresa tu edad: "))
3
4 # Verificar la edad y mostrar el mensaje correspondiente
5 if edad <= 0:
6     print("Edad no válida, por favor ingresa un valor positivo.")
7 elif edad >= 18:
8     print("¡Puedes ver películas clasificadas R!")
9 elif edad >= 13:
10    print("Puedes ver películas clasificadas PG-13.")
11 else:
12    print("Te recomendamos películas clasificadas G o PG.")
13 print("Realizado por Flavio Rojas.")
```

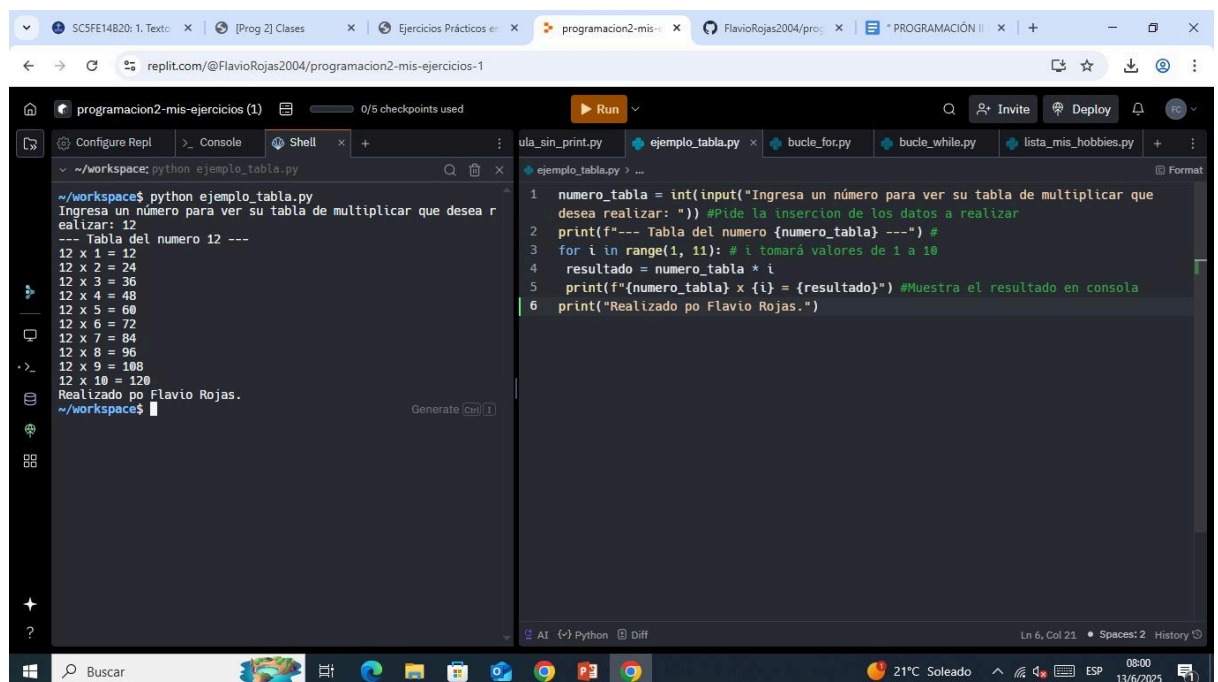
Terminal output:

```
~/workspace$ python clase0
clase02_mi_codigo.py clase03_ejercicio.py
~/workspace$ python clase03_ejercicio.py
Por favor, ingresa tu edad: 18
¡Puedes ver películas clasificadas R!
~/workspace$
```

Reseña y aprendizaje

este código es interesante ya que con el cual podemos verificar valores dados como un límite de edad una incógnita de un número, usando el `if` como la variable condicional complementando el `elif` y el `else`. de el código tuve el conocimiento de cómo hacer una condicional para dar autorizaciones o no

4. Bucle tabla de contar



The screenshot shows a Replit IDE window with a Python script named `ejemplo_tabla.py`. The script prompts the user for a number and uses a `for` loop to display a multiplication table. The console shows the user inputting '12' and receiving a table of products from 1 to 10.

```
1 numero_tabla = int(input("Ingresa un número para ver su tabla de multiplicar que
2 desea realizar: ")) #Pide la insercion de los datos a realizar
3 print(f"--- Tabla del numero {numero_tabla} ---") #
4 for i in range(1, 11): # i tomará valores de 1 a 10
5     resultado = numero_tabla * i
6     print(f"{numero_tabla} x {i} = {resultado}") #Muestra el resultado en consola
7 print("Realizado po Flavio Rojas.")
```

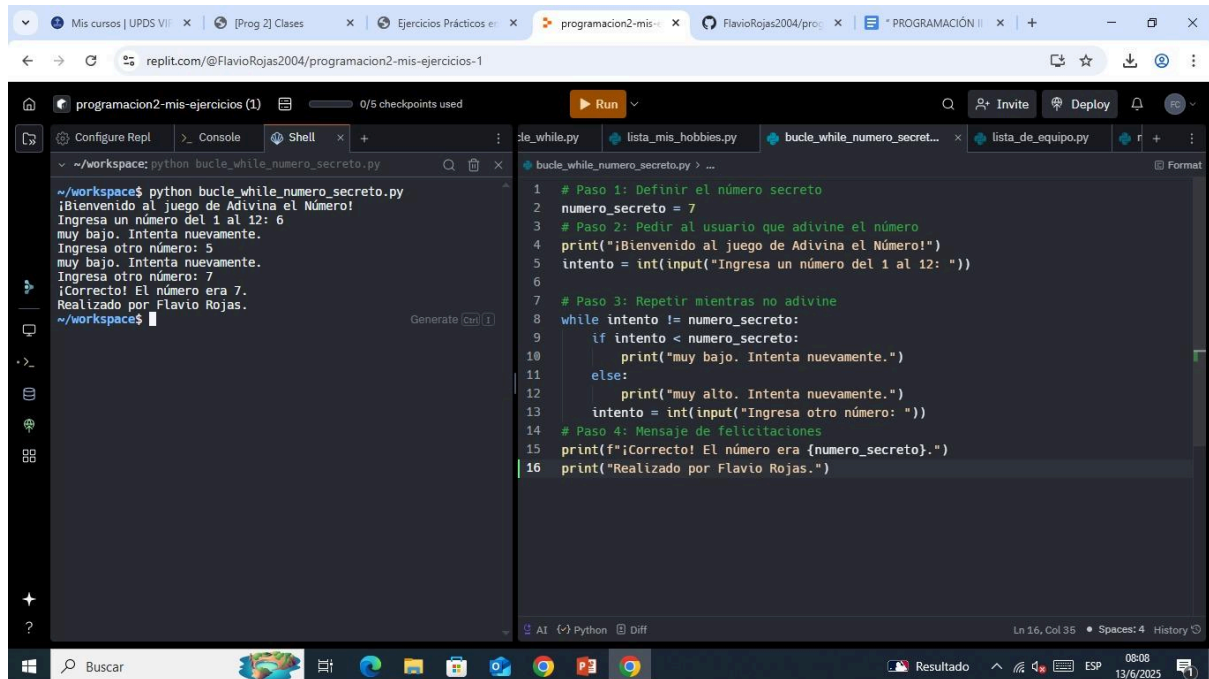
Terminal output:

```
~/workspace$ python ejemplo_tabla.py
Ingresa un número para ver su tabla de multiplicar que desea r
ealizar: 12
--- Tabla del numero 12 ---
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
Realizado po Flavio Rojas.
~/workspace$
```

Reseña y aprendizaje

este es un bucle for con rango 1, 11 dándole los valores a *i* el cual tomará los valores del 1 al 10, este código es una forma de simplificar al momento de hacer líneas de código ya que el *i* toma los valores y realizar la operación hasta el límite dado que es 10.

5. Adivina el número con while



The screenshot shows a Replit workspace with a Python script named `bucle_while_numero_secreto.py`. The script implements a number guessing game. The console output shows the user entering numbers 5 and 7, and the program responding with prompts and feedback. The code uses a `while` loop to repeat the process until the correct number is guessed.

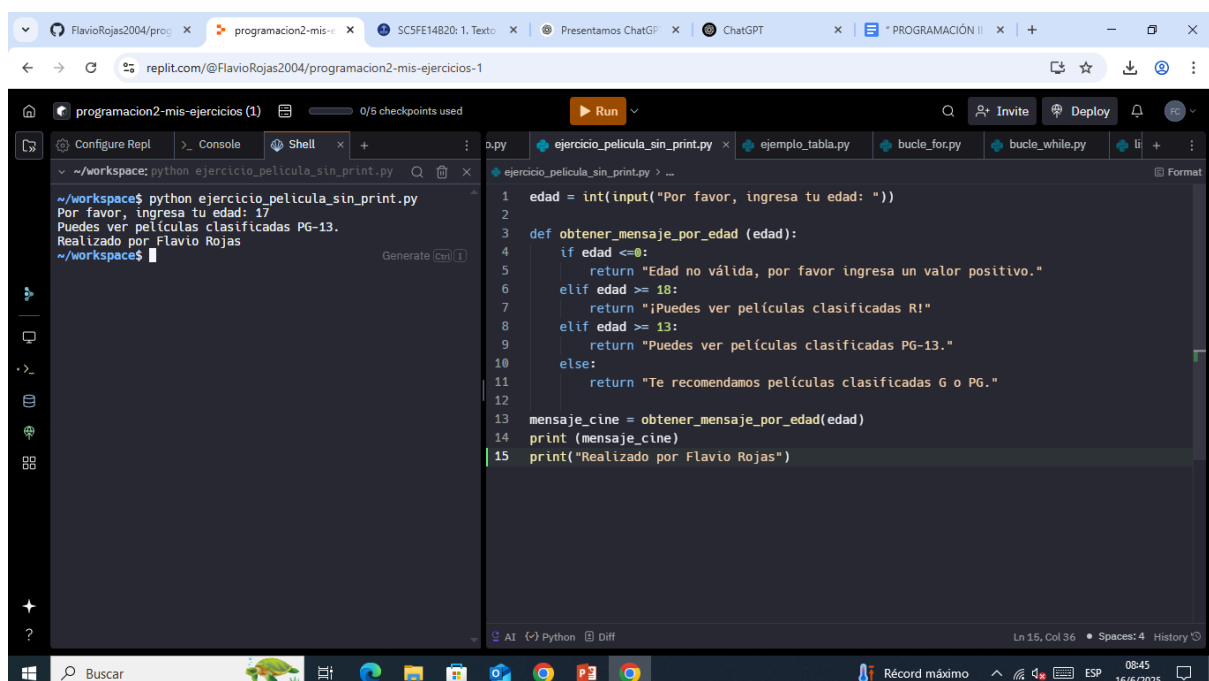
```
~/workspace$ python bucle_while_numero_secreto.py
¡Bienvenido al juego de Adivina el Número!
Ingresa un número del 1 al 12: 6
muy bajo. Intenta nuevamente.
Ingresa otro número: 5
muy bajo. Intenta nuevamente.
Ingresa otro número: 7
¡Correcto! El número era 7.
Realizado por Flavio Rojas.
~/workspace$
```

```
1 # Paso 1: Definir el número secreto
2 numero_secreto = 7
3 # Paso 2: Pedir al usuario que adivine el número
4 print("¡Bienvenido al juego de Adivina el Número!")
5 intento = int(input("Ingresa un número del 1 al 12: "))
6
7 # Paso 3: Repetir mientras no adivine
8 while intento != numero_secreto:
9     if intento < numero_secreto:
10         print("muy bajo. Intenta nuevamente.")
11     else:
12         print("muy alto. Intenta nuevamente.")
13         intento = int(input("Ingresa otro número: "))
14 # Paso 4: Mensaje de felicitaciones
15 print(f"¡Correcto! El número era {numero_secreto}.")
16 print("Realizado por Flavio Rojas.")
```

Reseña y aprendizaje

Este código resalta el aprendizaje de las estructuras repetitivas ya que al falla a encontrar el número secreto se vuelve a pedir que ingrese otra vez un número haciendo secuencialmente hasta que el usuario encuentre el correcto

6. Obtener clasificación de película usando return



The screenshot shows a Replit workspace with a Python script named `ejercicio_pelicula_sin_print.py`. The script defines a function `obtener_mensaje_por_edad` that returns a message based on the user's age. The console output shows the user entering their age (17) and the program returning the appropriate message. The code uses `return` to pass the result back to the caller.

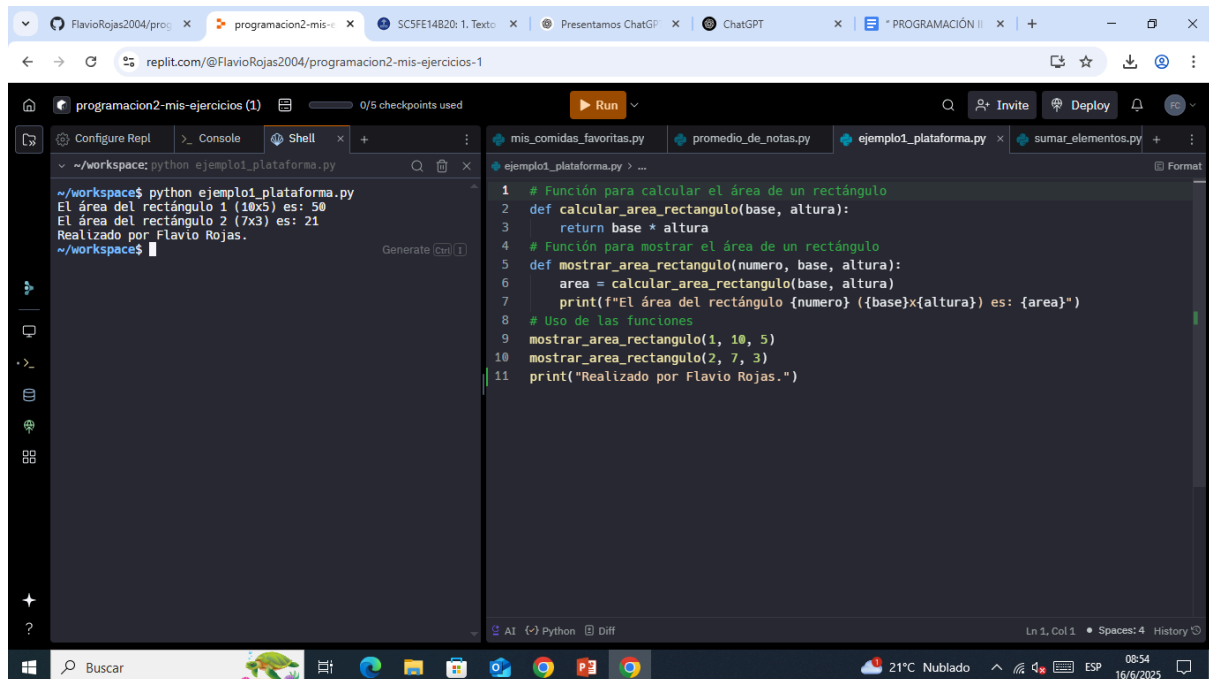
```
~/workspace$ python ejercicio_pelicula_sin_print.py
Por favor, ingresa tu edad: 17
Puedes ver películas clasificadas PG-13.
Realizado por Flavio Rojas
~/workspace$
```

```
1 edad = int(input("Por favor, ingresa tu edad: "))
2
3 def obtener_mensaje_por_edad(edad):
4     if edad <= 0:
5         return "Edad no válida, por favor ingresa un valor positivo."
6     elif edad >= 18:
7         return "¡Puedes ver películas clasificadas R!"
8     elif edad >= 13:
9         return "Puedes ver películas clasificadas PG-13."
10    else:
11        return "Te recomendamos películas clasificadas G o PG."
12
13 mensaje_cine = obtener_mensaje_por_edad(edad)
14 print(mensaje_cine)
15 print("Realizado por Flavio Rojas")
```

Reseña y aprendizaje

al igual que el ejemplo de estructuras de control este ejemplo busca lo mismo que es hacer una clase de parámetros para el ingreso a ver películas pero en este caso en particular lo hacemos utilizando el comando return para devolver el mensaje, este fue útil para aprender a hacer lo mismo pero añadiendo diferente variable

7. Ejemplo de refactorización



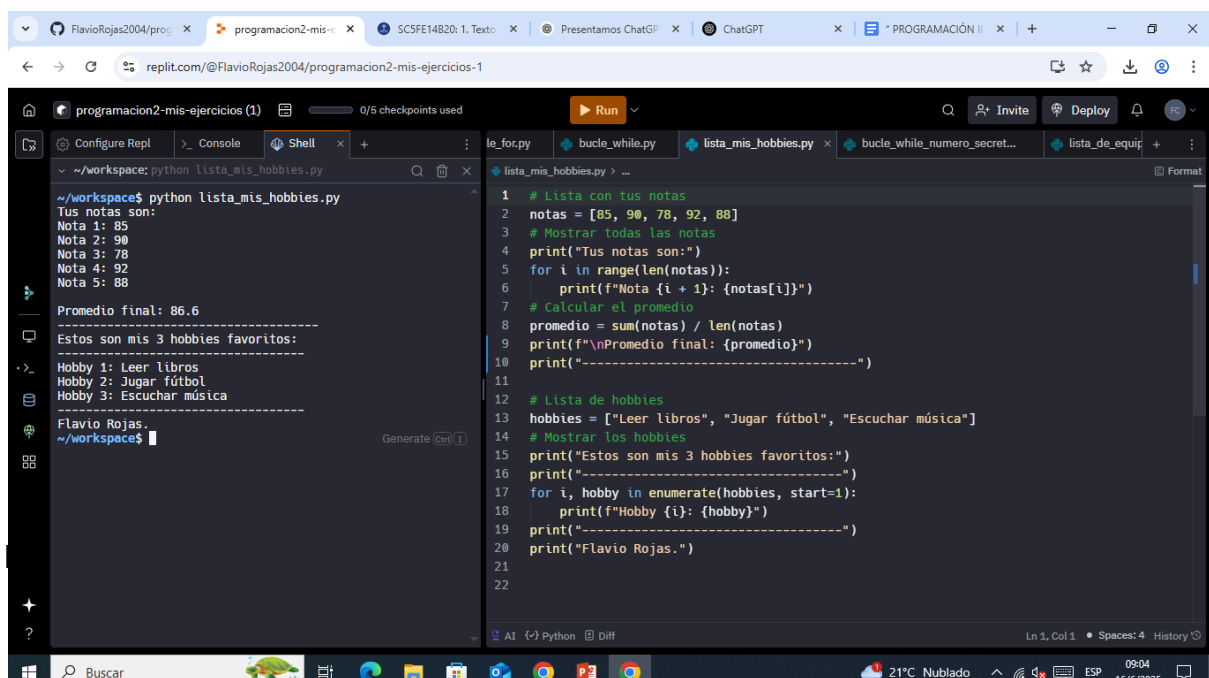
```
~/workspace$ python ejemplo1_plataforma.py
El área del rectángulo 1 (10x5) es: 50
El área del rectángulo 2 (7x3) es: 21
Realizado por Flavio Rojas.
~/workspace$
```

```
1 # Función para calcular el área de un rectángulo
2 def calcular_area_rectangulo(base, altura):
3     return base * altura
4 # Función para mostrar el área de un rectángulo
5 def mostrar_area_rectangulo(numero, base, altura):
6     area = calcular_area_rectangulo(base, altura)
7     print(f"El área del rectángulo {numero} ({base}x{altura}) es: {area}")
8 # Uso de las funciones
9 mostrar_area_rectangulo(1, 10, 5)
10 mostrar_area_rectangulo(2, 7, 3)
11 print("Realizado por Flavio Rojas.")
```

Reseña y aprendizaje:

Este código de refactorización para hallar las medidas de un rectángulo utiliza la valorización de multiplicar la base por la altura las cuales ya se declaran en el mismo código. este código fue por así decirlo sencillo ya que solo se hace lo básico y solo se le añadiría la función de multiplicar

8. lista de notas y lista de hobbies



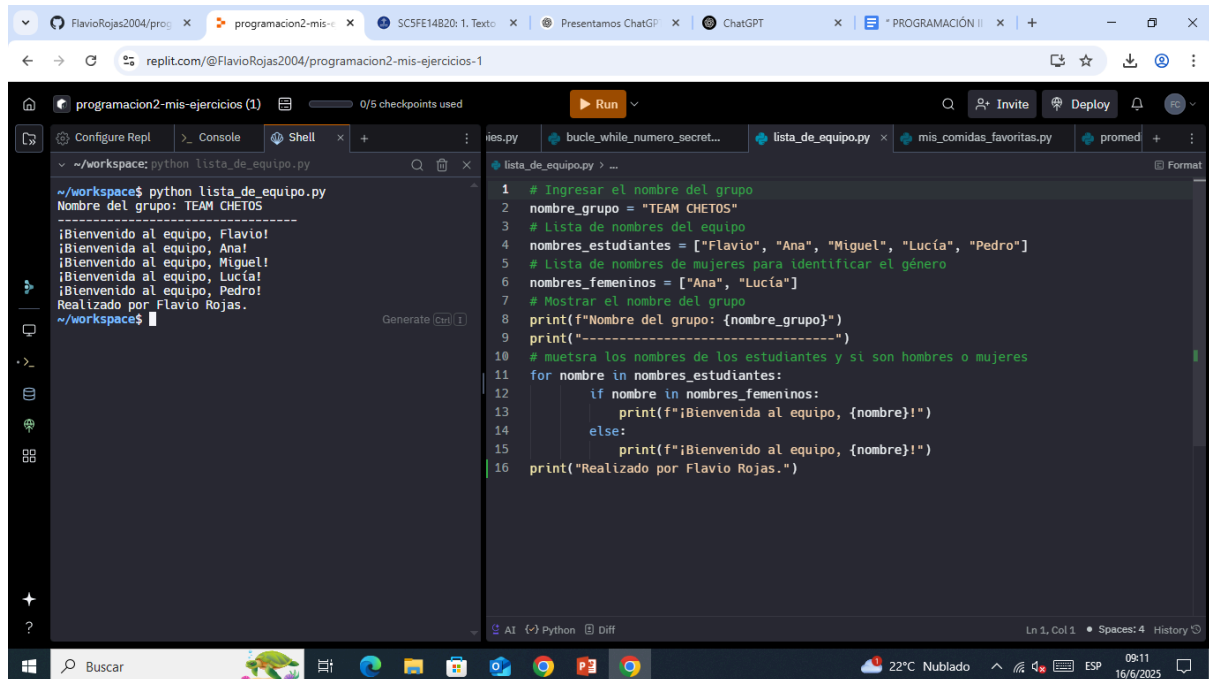
```
~/workspace$ python lista_mis_hobbies.py
Tus notas son:
Nota 1: 85
Nota 2: 90
Nota 3: 78
Nota 4: 92
Nota 5: 88

Promedio final: 86.6
-----
Estos son mis 3 hobbies favoritos:
Hobby 1: Leer libros
Hobby 2: Jugar fútbol
Hobby 3: Escuchar música
-----
Flavio Rojas.
~/workspace$
```

```
1 # Lista con tus notas
2 notas = [85, 90, 78, 92, 88]
3 # Mostrar todas las notas
4 print("Tus notas son:")
5 for i in range(len(notas)):
6     print(f"Nota {i + 1}: {notas[i]}")
7 # Calcular el promedio
8 promedio = sum(notas) / len(notas)
9 print(f"\nPromedio final: {promedio}")
10 print("-----")
11
12 # Lista de hobbies
13 hobbies = ["Leer libros", "Jugar fútbol", "Escuchar música"]
14 # Mostrar los hobbies
15 print("Estos son mis 3 hobbies favoritos:")
16 print("-----")
17 for i, hobby in enumerate(hobbies, start=1):
18     print(f"Hobby {i}: {hobby}")
19 print("-----")
20 print("Flavio Rojas.")
21
22
```

Estos códigos son similares en estructuras ya que ambos se enmarcan en un modelo de lista para mostrar lo requerido en cada caso siendo muy interesante e importante al momento de querer mostrar algo ordenado y eficiente. Este código es bastante útil para hacer los trabajos más limpios y legibles al momento de programar.

9. Lista grupal



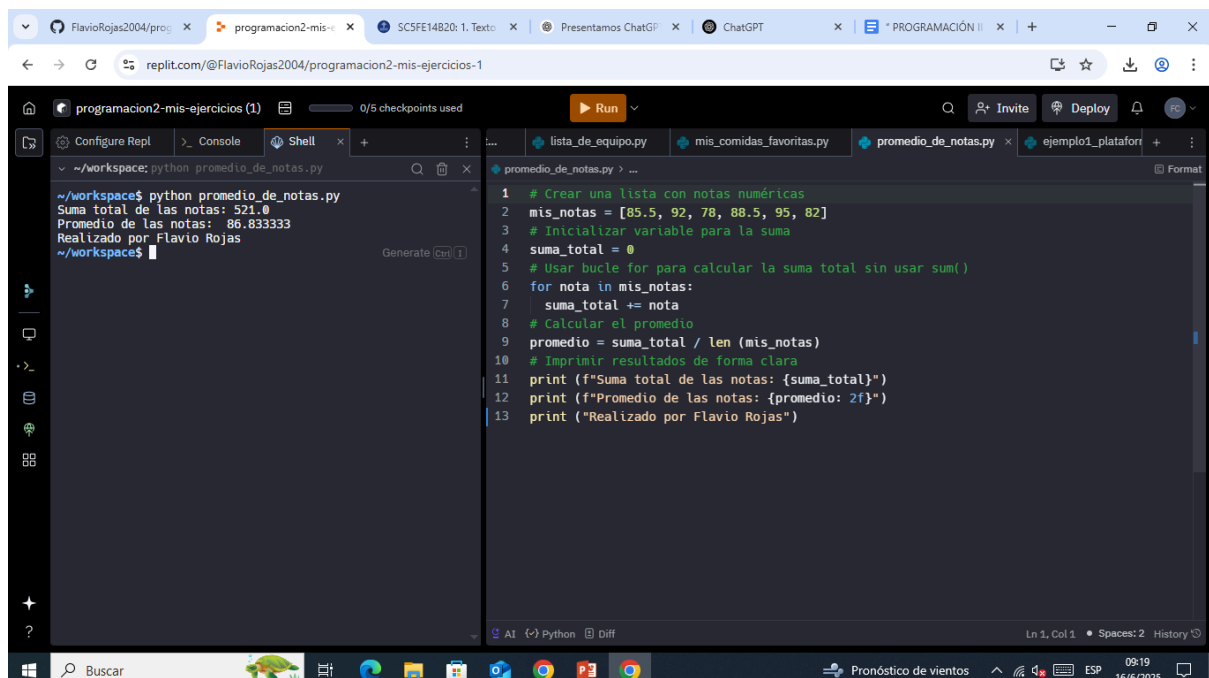
```
~/workspace$ python lista_de_equipo.py
Nombre del grupo: TEAM CHETOS
-----
¡Bienvenido al equipo, Flavio!
¡Bienvenida al equipo, Ana!
¡Bienvenido al equipo, Miguel!
¡Bienvenida al equipo, Lucía!
¡Bienvenido al equipo, Pedro!
Realizado por Flavio Rojas.
~/workspace$
```

```
1 # Ingresar el nombre del grupo
2 nombre_grupo = "TEAM CHETOS"
3 # Lista de nombres del equipo
4 nombres_estudiantes = ["Flavio", "Ana", "Miguel", "Lucía", "Pedro"]
5 # Lista de nombres de mujeres para identificar el género
6 nombres_femeninos = ["Ana", "Lucía"]
7 # Mostrar el nombre del grupo
8 print(f"Nombre del grupo: {nombre_grupo}")
9 print("-----")
10 # muestra los nombres de los estudiantes y si son hombres o mujeres
11 for nombre in nombres_estudiantes:
12     if nombre in nombres_femeninos:
13         print(f"¡Bienvenida al equipo, {nombre}!")
14     else:
15         print(f"¡Bienvenido al equipo, {nombre}!")
16 print("Realizado por Flavio Rojas.")
```

Reseña y aprendizaje:

este código como el nombre dice fue utilizado para realizar una lista de integrantes de un grupo de proyecto el cual utiliza la estructura for al cual le añadí dos listas una de mujeres y otra de hombres esto para que al momento de asignar el mensaje el programa lo haga de acuerdo al sexo. de este código en particular resalta la automatización al momento de mostrar el saludo con el nombre de cada persona.

10. Promedio de notas



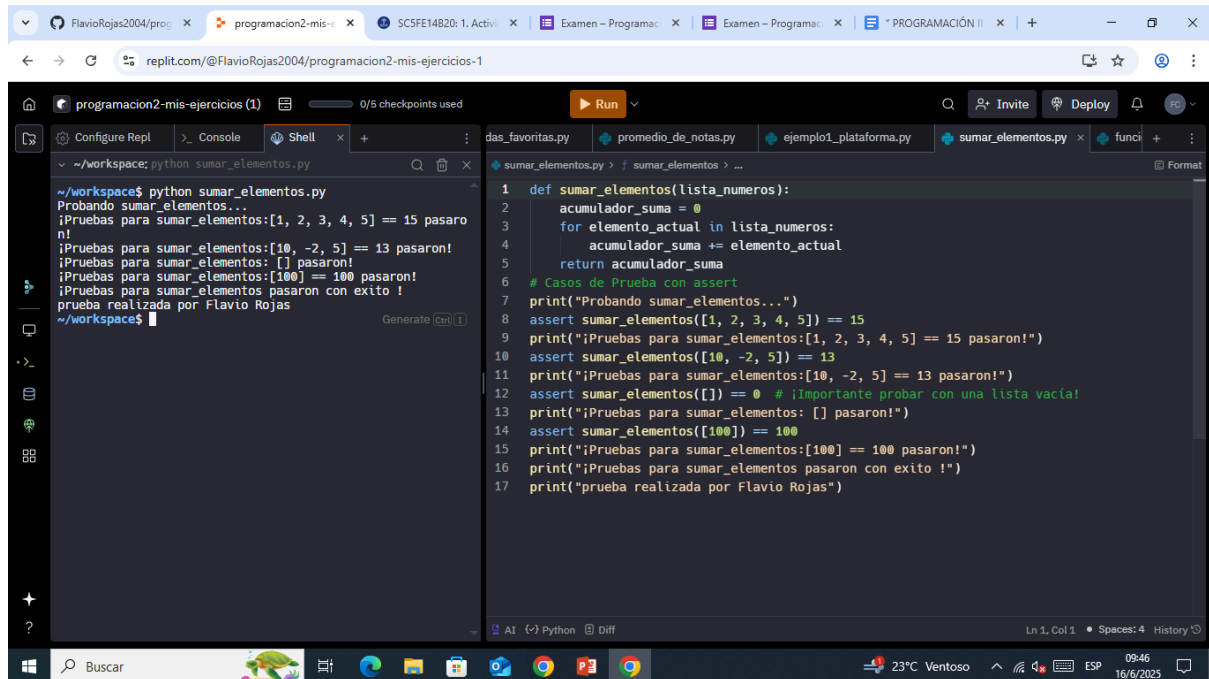
```
~/workspace$ python promedio_de_notas.py
Suma total de las notas: 521.0
Promedio de las notas: 86.833333
Realizado por Flavio Rojas
~/workspace$
```

```
1 # Crear una lista con notas numéricas
2 mis_notas = [85.5, 92, 78, 88.5, 95, 82]
3 # Inicializar variable para la suma
4 suma_total = 0
5 # Usar bucle for para calcular la suma total sin usar sum()
6 for nota in mis_notas:
7     suma_total += nota
8 # Calcular el promedio
9 promedio = suma_total / len(mis_notas)
10 # Imprimir resultados de forma clara
11 print(f"Suma total de las notas: {suma_total}")
12 print(f"Promedio de las notas: {promedio: 2f}")
13 print("Realizado por Flavio Rojas")
```


Reseña y aprendizaje:

este código de lista solo le añadimos el comando para sacar el promedio el cual es bastante fácil de atender ya que solo se utiliza de fórmula de suma total dividido por todas las notas. de el código recalcar la importancia que tiene para realizar actividades escolares ya que solo se ingresan las notas y el programa se encargará de todo lo demás

11. Función sumar elementos



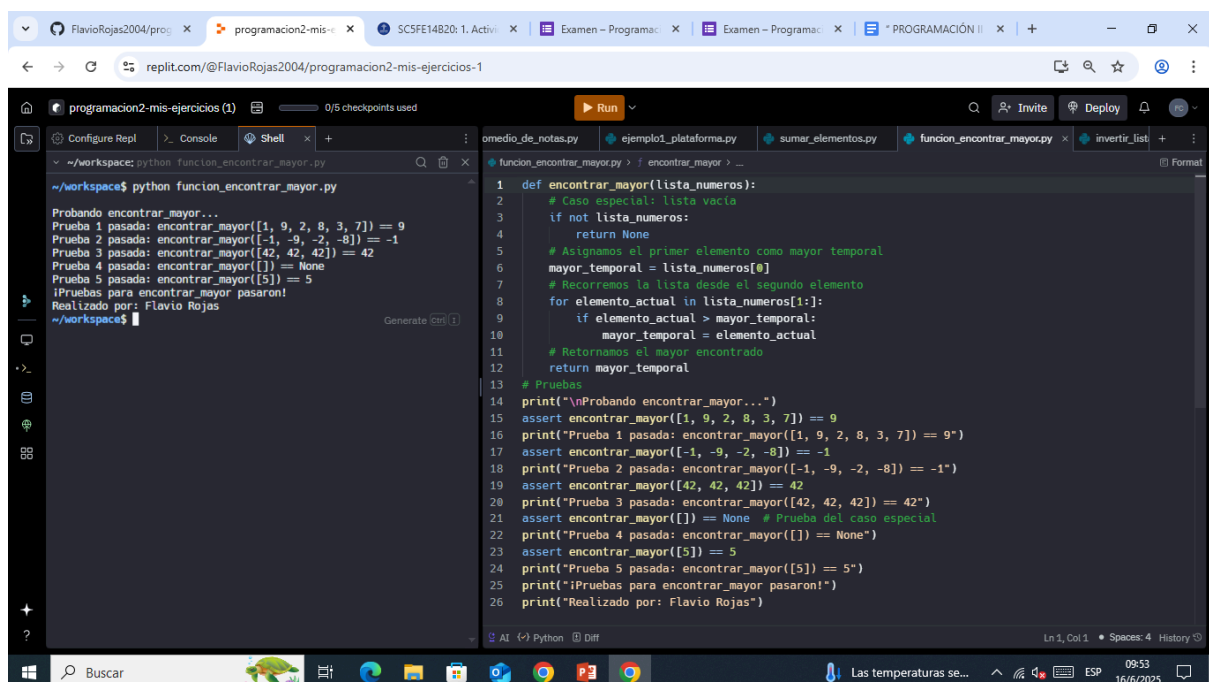
```
def sumar_elementos(lista_numeros):
    acumulador_suma = 0
    for elemento_actual in lista_numeros:
        acumulador_suma += elemento_actual
    return acumulador_suma

# Casos de Prueba con assert
print("Probando sumar_elementos...")
assert sumar_elementos([1, 2, 3, 4, 5]) == 15
print("¡Pruebas para sumar_elementos:[1, 2, 3, 4, 5] == 15 pasaron!")
assert sumar_elementos([10, -2, 5]) == 13
print("¡Pruebas para sumar_elementos:[10, -2, 5] == 13 pasaron!")
assert sumar_elementos([]) == 0 # ¡importante probar con una lista vacía!
print("¡Pruebas para sumar_elementos: [] pasaron!")
assert sumar_elementos([100]) == 100
print("¡Pruebas para sumar_elementos:[100] == 100 pasaron!")
print("¡Pruebas para sumar_elementos pasaron con éxito!")
print("prueba realizada por Flavio Rojas")
```

Reseña y aprendizaje:

Este ejemplo es para hallar la suma total de los elementos y luego verificar con el assert y constatar que el código funcione correctamente. este código aprendí a constatar resultados con el assert.

12. Encontrar al mayor



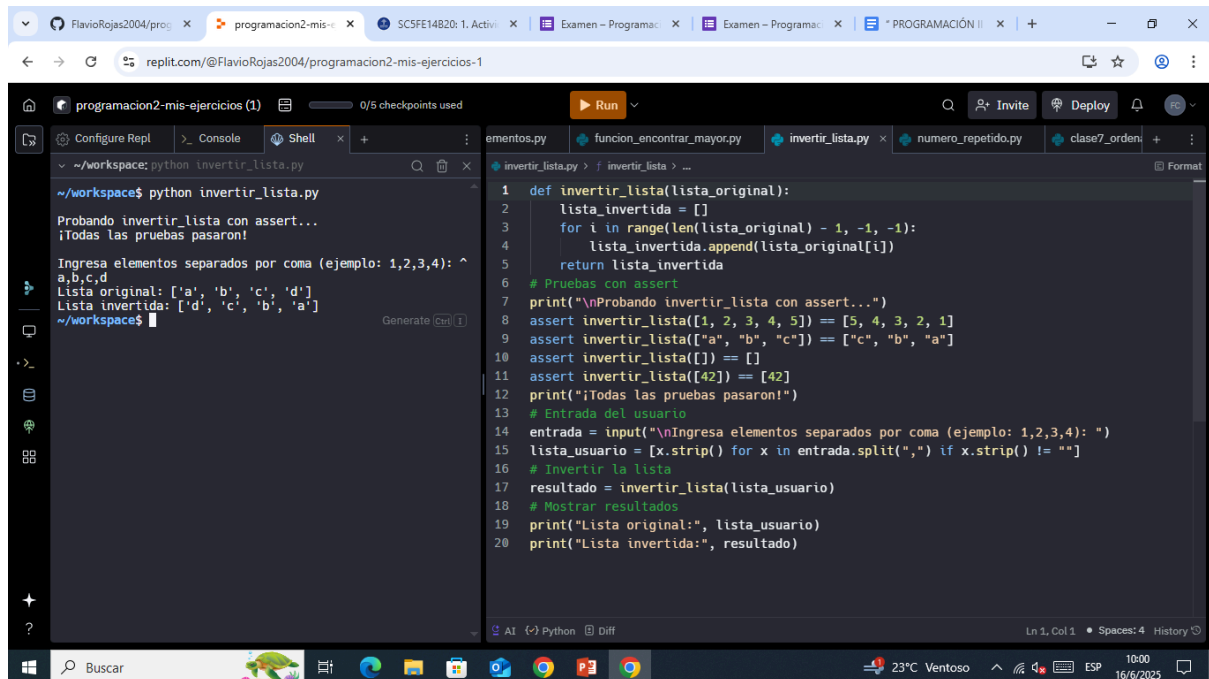
```
def encontrar_mayor(lista_numeros):
    # Caso especial: lista vacía
    if not lista_numeros:
        return None
    # Asignamos el primer elemento como mayor temporal
    mayor_temporal = lista_numeros[0]
    # Recorremos la lista desde el segundo elemento
    for elemento_actual in lista_numeros[1:]:
        if elemento_actual > mayor_temporal:
            mayor_temporal = elemento_actual
    # Retornamos el mayor encontrado
    return mayor_temporal

# Pruebas
print("\nProbando encontrar_mayor...")
assert encontrar_mayor([1, 9, 2, 8, 3, 7]) == 9
print("Prueba 1 pasada: encontrar_mayor([1, 9, 2, 8, 3, 7]) == 9")
assert encontrar_mayor([-1, -9, -2, -8]) == -1
print("Prueba 2 pasada: encontrar_mayor([-1, -9, -2, -8]) == -1")
assert encontrar_mayor([42, 42, 42]) == 42
print("Prueba 3 pasada: encontrar_mayor([42, 42, 42]) == 42")
assert encontrar_mayor([]) == None # Prueba del caso especial
print("Prueba 4 pasada: encontrar_mayor([]) == None")
assert encontrar_mayor([5]) == 5
print("Prueba 5 pasada: encontrar_mayor([5]) == 5")
print("¡Pruebas para encontrar_mayor pasaron!")
print("Realizado por: Flavio Rojas")
```

Reseña y aprendizaje:

en este código aprendí a utilizar un bucle con return para que el programa se encargue de mostrar el mayor en una lista desordenada ya que utilizando este bucle y añadiendo el assert para validar los datos.

13. Invertir lista

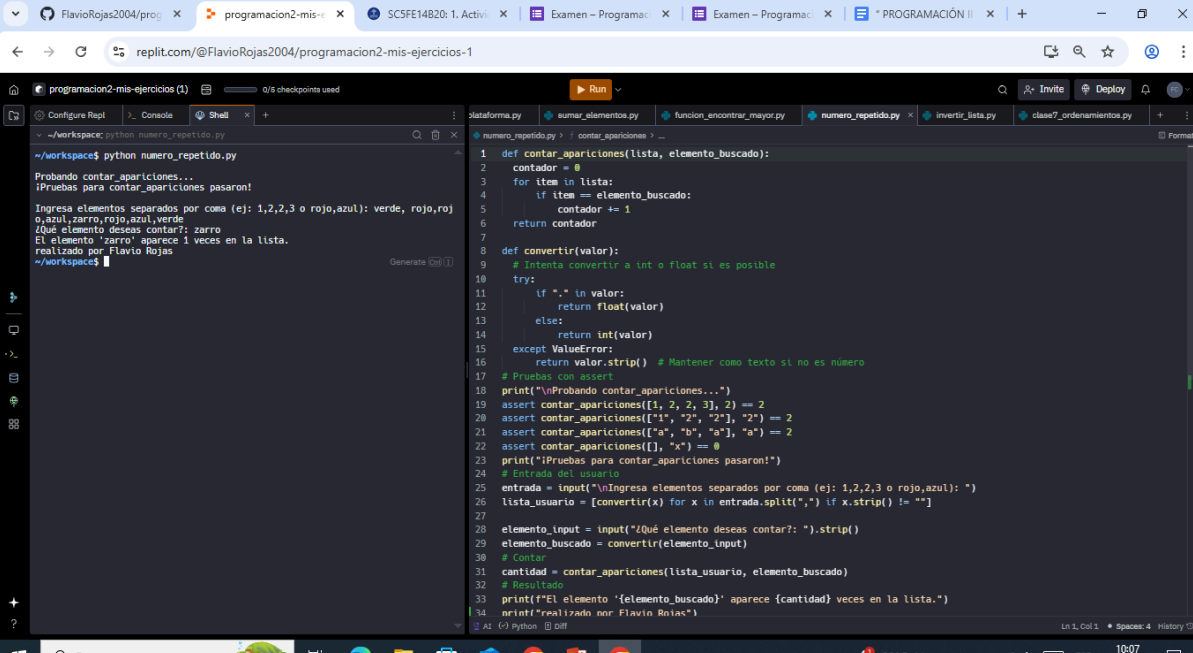


```
FlavioRojas2004/prog... programacion2-mis-ejerc... SCSFE14B20: 1. Activ... Examen - Programac... Examen - Programac... PROGRAMACIÓN II...  
replit.com/@FlavioRojas2004/programacion2-mis-ejercicios-1  
programacion2-mis-ejercicios (1) 0/5 checkpoints used Run Invite Deploy  
~/workspace:python invertir_lista.py  
~/workspace$ python invertir_lista.py  
Probando invertir_lista con assert...  
¡Todas las pruebas pasaron!  
Ingresa elementos separados por coma (ejemplo: 1,2,3,4): ^  
a,b,c,d  
Lista original: ['a', 'b', 'c', 'd']  
Lista invertida: ['d', 'c', 'b', 'a']  
~/workspace$  
1 def invertir_lista(lista_original):  
2     lista_invertida = []  
3     for i in range(len(lista_original) - 1, -1, -1):  
4         lista_invertida.append(lista_original[i])  
5     return lista_invertida  
6 # Pruebas con assert  
7 print("\nProbando invertir_lista con assert...")  
8 assert invertir_lista([1, 2, 3, 4, 5]) == [5, 4, 3, 2, 1]  
9 assert invertir_lista(["a", "b", "c"]) == ["c", "b", "a"]  
10 assert invertir_lista([]) == []  
11 assert invertir_lista([42]) == [42]  
12 print("¡Todas las pruebas pasaron!")  
13 # Entrada del usuario  
14 entrada = input("\nIngresa elementos separados por coma (ejemplo: 1,2,3,4): ")  
15 lista_usuario = [x.strip() for x in entrada.split(",") if x.strip() != ""]  
16 # Invertir la lista  
17 resultado = invertir_lista(lista_usuario)  
18 # Mostrar resultados  
19 print("Lista original:", lista_usuario)  
20 print("Lista invertida:", resultado)
```

Reseña y aprendizaje:

En este código se utiliza un strip para invertir los datos ingresados y validando con el assert para mayor eficacia, ya que al pedir que ingrese la lista a invertir se debe separar con una coma como lo indica en el comando split.

14. Contar elementos

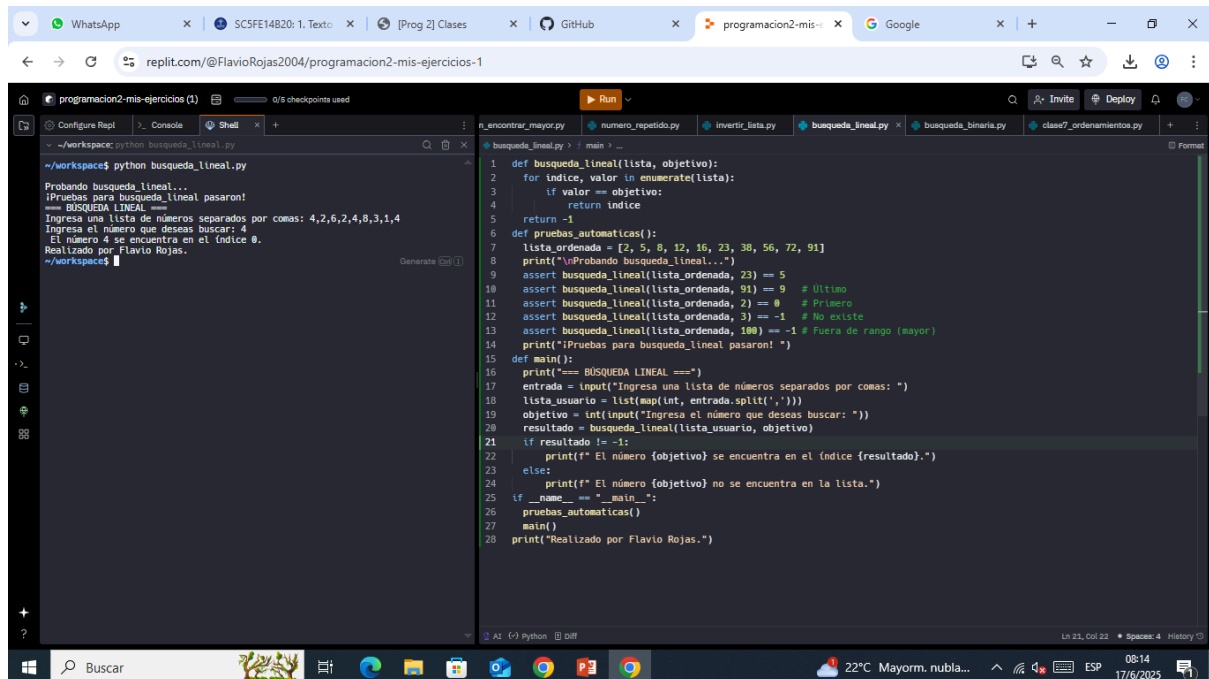


```
FlavioRojas2004/prog... programacion2-mis-ejerc... SCSFE14B20: 1. Activ... Examen - Programac... Examen - Programac... PROGRAMACIÓN II...  
replit.com/@FlavioRojas2004/programacion2-mis-ejercicios-1  
programacion2-mis-ejercicios (1) 0/5 checkpoints used Run Invite Deploy  
~/workspace:python numero_repetido.py  
~/workspace$ python numero_repetido.py  
Probando contar_apariciones...  
¡Pruebas para contar_apariciones pasaron!  
Ingresa elementos separados por coma (ej: 1,2,2,3 o rojo,azul): verde, rojo,rojo  
o,azul,zarzo,rojo,azul,verde  
¿Qué elemento deseas contar?: zarzo  
El elemento 'zarzo' aparece 1 veces en la lista.  
realizado por Flavio Rojas  
~/workspace$  
1 def contar_apariciones(lista, elemento_buscado):  
2     contador = 0  
3     for item in lista:  
4         if item == elemento_buscado:  
5             contador += 1  
6     return contador  
7  
8 def convertir(valor):  
9     # Intenta convertir a int o float si es posible  
10    try:  
11        if "." in valor:  
12            return float(valor)  
13        else:  
14            return int(valor)  
15    except ValueError:  
16        return valor.strip() # Mantener como texto si no es número  
17 # Pruebas con assert  
18 print("\nProbando contar_apariciones...")  
19 assert contar_apariciones([1, 2, 2, 3], 2) == 2  
20 assert contar_apariciones(["1", "2", "2"], "2") == 2  
21 assert contar_apariciones(["a", "b", "a"], "a") == 2  
22 assert contar_apariciones([], "x") == 0  
23 print("¡Pruebas para contar_apariciones pasaron!")  
24 # Entrada del usuario  
25 entrada = input("\nIngresa elementos separados por coma (ej: 1,2,2,3 o rojo,azul): ")  
26 lista_usuario = [convertir(x) for x in entrada.split(",") if x.strip() != ""]  
27  
28 elemento_input = input("\n¿Qué elemento deseas contar?: ").strip()  
29 elemento_buscado = convertir(elemento_input)  
30 # Contar  
31 cantidad = contar_apariciones(lista_usuario, elemento_buscado)  
32 # Mostrar resultado  
33 print(f"El elemento '{elemento_buscado}' aparece {cantidad} veces en la lista.")  
34 print("realizado por Flavio Rojas")
```


Reseña y aprendizaje:

este código es interesante ya que es un buscador de palabras las cuales muestra cuántas veces llega a repetirse, útil para redacción de documentos. de código se sostuvo a la validación de los asserts los cuales se añaden y los cuales se hace la verificación.

15. Búsqueda lineal



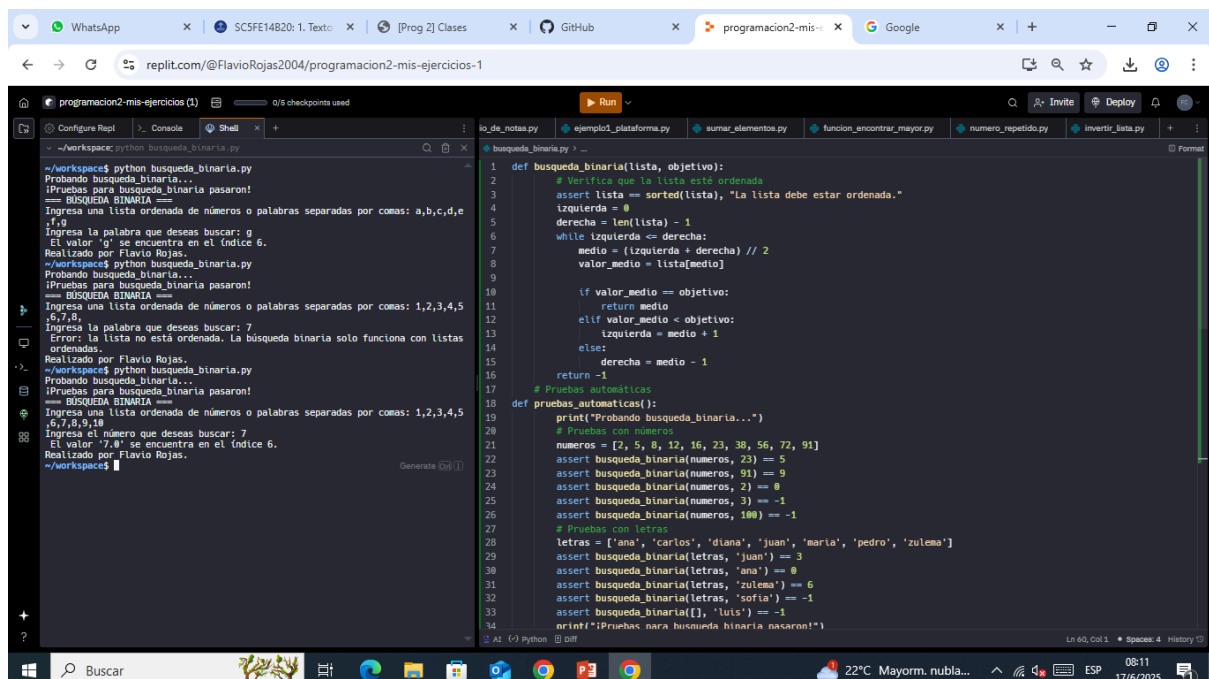
The screenshot shows a Replit workspace for a project named 'programacion2-mis-ejercicios-1'. The file explorer on the left shows several Python files, including 'busqueda_lineal.py'. The main editor displays the code for 'busqueda_lineal.py'. The code defines a function 'busqueda_lineal(lista, objetivo)' that iterates through a list to find a target value. It also includes a 'pruebas_automáticas()' function with various assertions and a 'main()' function that takes user input and calls the search function. The terminal on the left shows the execution of the script, including test results and user input prompts.

```
1 def busqueda_lineal(lista, objetivo):
2     for indice, valor in enumerate(lista):
3         if valor == objetivo:
4             return indice
5     return -1
6
7 def pruebas_automáticas():
8     lista_ordenada = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
9     print("Probando busqueda_lineal...")
10    assert busqueda_lineal(lista_ordenada, 23) == 5
11    assert busqueda_lineal(lista_ordenada, 91) == 9 # Último
12    assert busqueda_lineal(lista_ordenada, 2) == 0 # Primero
13    assert busqueda_lineal(lista_ordenada, 3) == -1 # No existe
14    assert busqueda_lineal(lista_ordenada, 100) == -1 # Fuera de rango (mayor)
15    print("¡Pruebas para busqueda_lineal pasaron!")
16
17 def main():
18    print("== BÚSQUEDA LINEAL ==")
19    entrada = input("Ingresa una lista de números separados por comas: ")
20    lista_usuario = list(map(int, entrada.split(',')))
21    objetivo = int(input("Ingresa el número que deseas buscar: "))
22    resultado = busqueda_lineal(lista_usuario, objetivo)
23
24    if resultado != -1:
25        print(f"El número {objetivo} se encuentra en el índice {resultado}.")
26    else:
27        print(f"El número {objetivo} no se encuentra en la lista.")
28
29 if __name__ == "__main__":
30     pruebas_automáticas()
31     main()
32
33 print("Realizado por Flavio Rojas.")
```

Reseña y aprendizaje:

Con este código se desarrolló la búsqueda lineal para hallar elementos como números o palabras haciendo las validaciones con los asserts para su correcto funcionamiento. este código realza que al igual que los algunos anteriores la estructura es similar pero realiza la búsqueda más optimizada.

16. Búsqueda binaria



The screenshot shows a Replit workspace for a project named 'programacion2-mis-ejercicios-1'. The file explorer on the left shows several Python files, including 'busqueda_binaria.py'. The main editor displays the code for 'busqueda_binaria.py'. The code defines a function 'busqueda_binaria(lista, objetivo)' that implements a binary search algorithm on a sorted list. It includes a 'pruebas_automáticas()' function with assertions for both numbers and letters, and a 'main()' function that takes user input and calls the search function. The terminal on the left shows the execution of the script, including test results and user input prompts.

```
1 def busqueda_binaria(lista, objetivo):
2     # Verificar que la lista está ordenada
3     assert lista == sorted(lista), "La lista debe estar ordenada."
4     izquierda = 0
5     derecha = len(lista) - 1
6
7     while izquierda <= derecha:
8         medio = (izquierda + derecha) // 2
9         valor_medio = lista[medio]
10
11        if valor_medio == objetivo:
12            return medio
13        elif valor_medio < objetivo:
14            izquierda = medio + 1
15        else:
16            derecha = medio - 1
17    return -1
18
19 # Pruebas automáticas
20 def pruebas_automáticas():
21    print("Probando busqueda_binaria...")
22    # Pruebas con números
23    numeros = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
24    assert busqueda_binaria(numeros, 23) == 5
25    assert busqueda_binaria(numeros, 91) == 9
26    assert busqueda_binaria(numeros, 2) == 0
27    assert busqueda_binaria(numeros, 3) == -1
28    assert busqueda_binaria(numeros, 100) == -1
29
30    # Pruebas con letras
31    letras = ['ana', 'carlos', 'diana', 'juan', 'maria', 'pedro', 'zulema']
32    assert busqueda_binaria(letras, 'juan') == 3
33    assert busqueda_binaria(letras, 'ana') == 0
34    assert busqueda_binaria(letras, 'zulema') == 6
35    assert busqueda_binaria(letras, 'sofia') == -1
36    assert busqueda_binaria(letras, 'luis') == -1
37
38    print("¡Pruebas para busqueda_binaria pasaron!")
39
40 def main():
41    print("== BÚSQUEDA BINARIA ==")
42    entrada = input("Ingresa una lista ordenada de números o palabras separadas por comas: ")
43    lista_usuario = list(map(int, entrada.split(',')))
44    objetivo = input("Ingresa la palabra que deseas buscar: ")
45
46    resultado = busqueda_binaria(lista_usuario, objetivo)
47
48    if resultado != -1:
49        print(f"El valor '{objetivo}' se encuentra en el índice {resultado}.")
50    else:
51        print("Error: la lista no está ordenada. La búsqueda binaria solo funciona con listas ordenadas.")
52
53 if __name__ == "__main__":
54     pruebas_automáticas()
55     main()
56
57 print("Realizado por Flavio Rojas.")
```

Reseña y aprendizaje:

Este código utiliza búsqueda binaria la haciendo el funcionamiento correcto y eficaz con los asserts pero tiene un pequeño detalle al contrario de la búsqueda al momento de que no se inserta la lista de manera ordenada tiende al error ya que es necesario para que este de esta forma para que pueda realizar la búsqueda.