

UNIVERSIDAD PRIVADA DOMINGO SAVIO



Ejercicios de programación II: Actividad 01

Docente:

Jimmy Nataniel Requena Llorentty

Estudiante:

Rebeca Vargas Orellana

Materia:

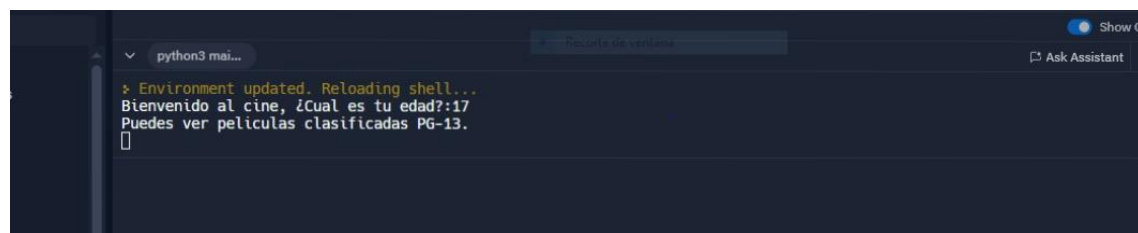
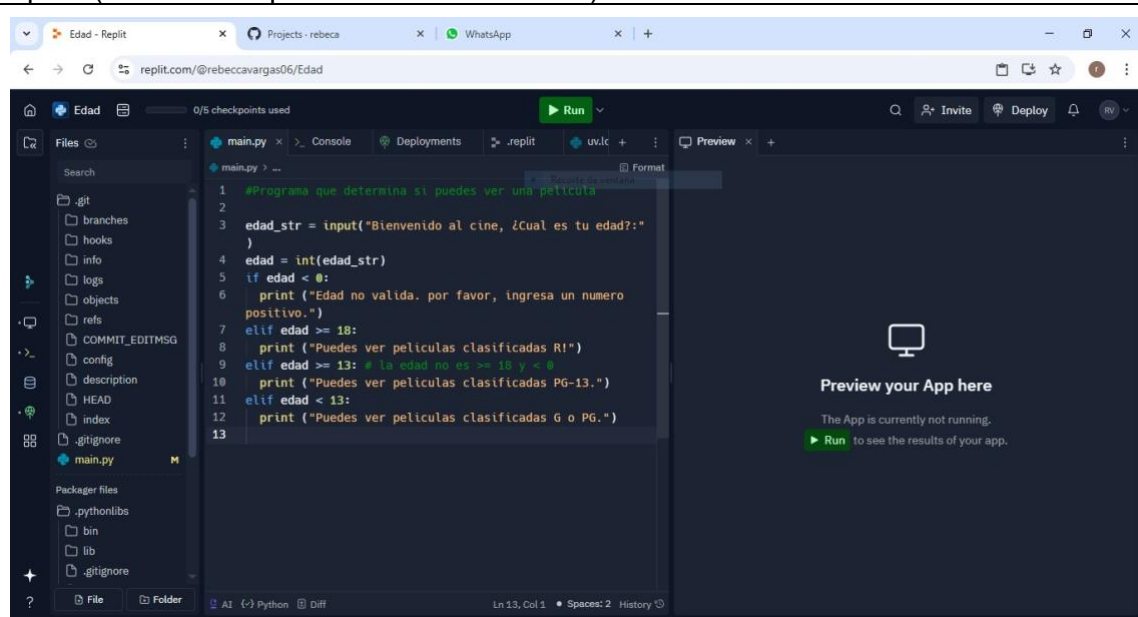
Programación II – Turno mañana

Santa Cruz – Bolivia

Ejercicios de Programación

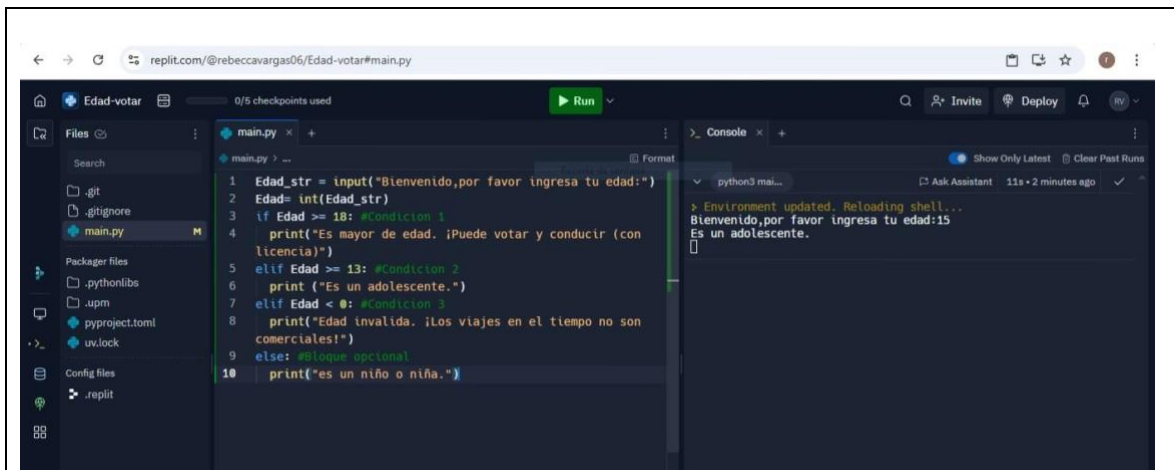
Actividad 01: Clases de Programación II

```
edad_str = input("Bienvenido al cine, ¿Cual es tu edad?:" )
edad = int(edad_str)
if edad < 0:
    print ("Edad no valida. por favor, ingresa un numero positivo.")
elif edad >= 18:
    print ("Puedes ver peliculas clasificadas R!")
elif edad >= 13: # la edad no es >= 18 y < 0
    print ("Puedes ver peliculas clasificadas PG-13.")
elif edad < 13:
    print ("Puedes ver peliculas clasificadas A.")
```



Código: Cine

```
Edad_str = input("Bienvenido,por favor ingresa tu edad:")
Edad= int(Edad_str)
if Edad >= 18: #Condicion 1
    print("Es mayor de edad. ¡Puede votar y conducir (con licencia)")
elif Edad >= 13: #Condicion 2
    print ("Es un adolescente.")
elif Edad < 0: #Condicion 3
    print("Edad invalida. ¡Los viajes en el tiempo no son comerciales!")
else: #Bloque opcional
    print("es un niño o niña.")
```

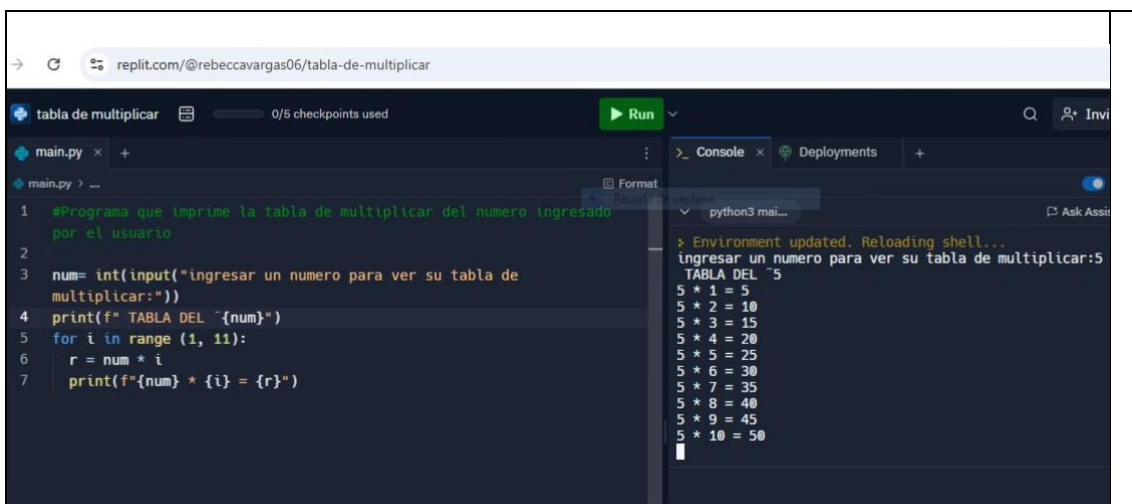


```
1 Edad_str = input("Bienvenido, por favor ingresa tu edad:")
2 Edad = int(Edad_str)
3 if Edad >= 18: #Condicion 1
4     print("Es mayor de edad. ¡Puede votar y conducir (con licencia)!")
5 elif Edad >= 13: #Condicion 2
6     print("Es un adolescente.")
7 elif Edad < 0: #Condicion 3
8     print("Edad invalida. ¡Los viajes en el tiempo no son comerciales!")
9 else: #Bloque opcional
10    print("es un niño o niña.")
```

Console output:

```
> Environment updated. Reloading shell...
Bienvenido, por favor ingresa tu edad:15
Es un adolescente.
```

Código: Definición Edad



```
1 #Programa que imprime la tabla de multiplicar del numero ingresado por el usuario
2
3 num = int(input("ingresar un numero para ver su tabla de multiplicar:"))
4 print(f" TABLA DEL {num}")
5 for i in range(1, 11):
6     r = num * i
7     print(f"{num} * {i} = {r}")
```

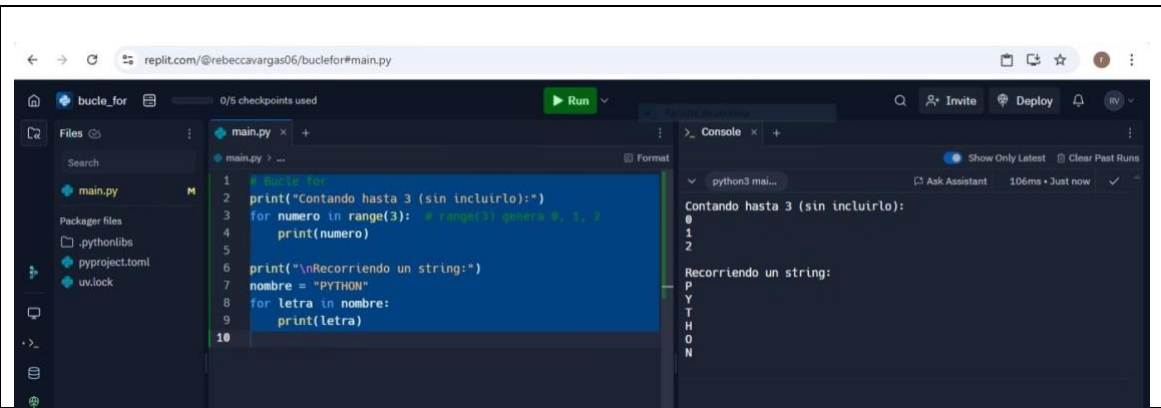
Console output:

```
> Environment updated. Reloading shell...
ingresar un numero para ver su tabla de multiplicar:5
TABLA DEL 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

Código: Tabla de Multiplicar

```
# Bucle for
print("Contando hasta 3 (sin incluirlo):")
for numero in range(3): # range(3) genera 0, 1, 2
    print(numero)

print("\nRecorriendo un string:")
nombre = "PYTHON"
for letra in nombre:
    print(letra)
```



The screenshot shows a Replit IDE window with the URL `replit.com/@rebeccavargas06/buclefor#main.py`. The file `main.py` contains the following Python code:

```
1 #Bucle for
2 print("Contando hasta 3 (sin incluirlo):")
3 for numero in range(3): # range(3) genera 0, 1, 2
4     print(numero)
5
6 print("\nRecorriendo un string:")
7 nombre = "PYTHON"
8 for letra in nombre:
9     print(letra)
10
```

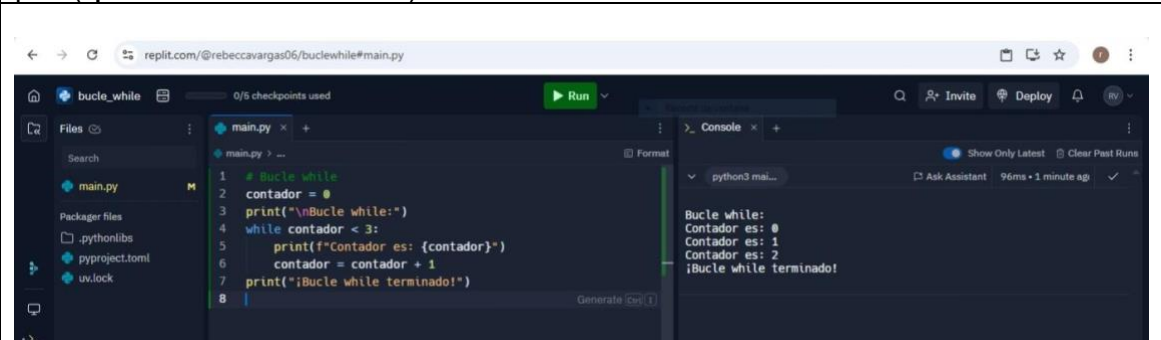
The console output on the right shows the execution results:

```
Contando hasta 3 (sin incluirlo):
0
1
2

Recorriendo un string:
P
Y
T
H
O
N
```

Código: Bucle For

```
# Bucle while
contador = 0
print("\nBucle while:")
while contador < 3:
    print(f"Contador es: {contador}")
    contador = contador + 1
print("¡Bucle while terminado!")
```



The screenshot shows a Replit IDE window with the URL `replit.com/@rebeccavargas06/buclewhile#main.py`. The file `main.py` contains the following Python code:

```
1 # Bucle while
2 contador = 0
3 print("\nBucle while:")
4 while contador < 3:
5     print(f"Contador es: {contador}")
6     contador = contador + 1
7 print("¡Bucle while terminado!")
8
```

The console output on the right shows the execution results:

```
Bucle while:
Contador es: 0
Contador es: 1
Contador es: 2
¡Bucle while terminado!
```

Código: Bucle While

<pre> 1 # Definición de una Función (Crear el "molde") 2 3 def saludar(nombre_persona): 4 """Esta función recibe un nombre y muestra un saludo personalizado.""" 5 mensaje = f"¡Hola, {nombre_persona}! ¡Qué bueno tenerte aquí!" 6 print(mensaje) 7 # Esta función no devuelve un valor explícito (retorna None por defecto) 8 9 def sumar(a, b): 10 """Esta función recibe dos números y devuelve su suma.""" 11 resultado_suma = a + b 12 return resultado_suma # La palabra clave "return" devuelve un valor 13 14 # Llamada a la Función (Usar el "molde") 15 16 saludar("Ana") 17 saludar("Carlos") 18 19 resultado1 = sumar(5, 3) # 5 y 3 son ARGUMENTOS 20 print(f"La suma de 5 y 3 es: {resultado1}") 21 22 resultado2 = sumar(100, 250) 23 print(f"La suma de 100 y 250 es: {resultado2}") </pre>	<pre> ¡Hola, Ana! ¡Qué bueno tenerte aquí! ¡Hola, Carlos! ¡Qué bueno tenerte aquí! La suma de 5 y 3 es: 8 La suma de 100 y 250 es: 350 (Program finished) </pre>
Código: Saludo	

```

def obtener_clasificacion_pelicula(edad):

    if edad < 0:

        return "Edad no válida."

    elif edad >= 18:

        return "Puedes ver películas clasificadas R!"

    elif edad >= 13:

        return "Puedes ver películas clasificadas PG-13."

    else:

        return "Te recomendamos películas clasificadas G o PG."

# Pruebas con assert

print("Ejecutando pruebas...")

assert obtener_clasificacion_pelicula(20) == "Puedes ver películas clasificadas R!",
"Prueba fallida: Adulto"

```

```

assert obtener_clasificacion_pelicula(18) == "Puedes ver películas clasificadas R!",
"Pruueba fallida: Límite Adulto"

assert obtener_clasificacion_pelicula(15) == "Puedes ver películas clasificadas PG-
13.", "Pruueba fallida: Adolescente"

assert obtener_clasificacion_pelicula(13) == "Puedes ver películas clasificadas PG-
13.", "Pruueba fallida: Límite Adolescente"

assert obtener_clasificacion_pelicula(10) == "Te recomendamos películas
clasificadas G o PG.", "Pruueba fallida: Niño"

assert obtener_clasificacion_pelicula(-5) == "Edad no válida.", "Pruueba fallida: Edad
negativa"

print("¡Todas las pruebas pasaron exitosamente! Nuestra función es robusta. ")

```

Código: Películas Assert

The screenshot shows a Replit Python environment with a file named `main.py`. The code defines a function `obtener_clasificacion_pelicula(edad)` that returns a movie rating based on age. The function uses `assert` statements to test its behavior for various ages. The console output shows the function being executed successfully for all test cases.

```

def obtener_clasificacion_pelicula(edad):
    if edad < 0:
        return "Edad no válida."
    elif edad >= 18:
        return "Puedes ver películas clasificadas R!"
    elif edad >= 13:
        return "Puedes ver películas clasificadas PG-13."
    else:
        return "Te recomendamos películas clasificadas G o PG."

# Pruebas con assert
print("Ejecutando pruebas...")

assert obtener_clasificacion_pelicula(20) == "Puedes ver películas
clasificadas R!", "Pruueba fallida: Adulto"
assert obtener_clasificacion_pelicula(18) == "Puedes ver películas
clasificadas R!", "Pruueba fallida: Límite Adulto"
assert obtener_clasificacion_pelicula(15) == "Puedes ver películas
clasificadas PG-13.", "Pruueba fallida: Adolescente"
assert obtener_clasificacion_pelicula(13) == "Puedes ver películas
clasificadas PG-13.", "Pruueba fallida: Límite Adolescente"
assert obtener_clasificacion_pelicula(10) == "Te recomendamos
películas clasificadas G o PG.", "Pruueba fallida: Niño"
assert obtener_clasificacion_pelicula(-5) == "Edad no válida.", "Pruueba fallida: Edad
negativa"

print("¡Todas las pruebas pasaron exitosamente! Nuestra función es robusta.")

```

Código: Clasificación películas Assert

```

Def invertir_lista(lista_original):
    # Crear una nueva lista vacía
    Lista_invertida = []

    # Recorrer la lista original desde el final hacia el inicio
    For i in range(len(lista_original) - 1, -1, -1):
        Lista_invertida.append(lista_original[i])

```

```

# Retornar la nueva lista invertida
Return lista_invertida

# Pruebas
Print("\nProbando invertir_lista...")
Lista_prueba = [1, 2, 3, 4, 5]
Lista_resultante = invertir_lista(lista_prueba)
Assert lista_resultante == [5, 4, 3, 2, 1]
Assert lista_prueba == [1, 2, 3, 4, 5] # Verifica que la original no cambi6
Assert invertir_lista(["a", "b", "c"]) == ["c", "b", "a"]
Assert invertir_lista([]) == []
Print("¡Pruebas para invertir_lista pasaron! ✓")

```

The screenshot shows a Replit Python environment with a file named 'main.py'. The code in the file is as follows:

```

def invertir_lista(lista_original):
    # Crear una nueva lista vacía
    lista_invertida = []

    # Recorrer la lista original desde el final hacia el inicio
    for i in range(len(lista_original) - 1, -1, -1):
        lista_invertida.append(lista_original[i])

    # Retornar la nueva lista invertida
    return lista_invertida

# Pruebas
print("\nProbando invertir_lista...")
lista_prueba = [1, 2, 3, 4, 5]
lista_resultante = invertir_lista(lista_prueba)
assert lista_resultante == [5, 4, 3, 2, 1]
assert lista_prueba == [1, 2, 3, 4, 5] # Verifica que la original
no cambie
assert invertir_lista(["a", "b", "c"]) == ["c", "b", "a"]
assert invertir_lista([]) == []
print("¡Pruebas para invertir_lista pasaron!")

```

The console output shows the following:

```

File "/home/runner/workspace/main.py", line 20
print("¡Pruebas para invertir_lista pasaron! ✓")
      ^
SyntaxError: unterminated string literal (detected at line 20)

```

Código: Lista Invertida

```

def encontrar_mayor(lista_numeros):
    # Caso especial: lista vacía
    if not lista_numeros:
        return None

    # Asignar el primer elemento como el mayor temporal
    mayor_temporal = lista_numeros[0]

    # Recorrer el resto de la lista
    for elemento_actual in lista_numeros[1:]:
        if elemento_actual > mayor_temporal:
            mayor_temporal = elemento_actual

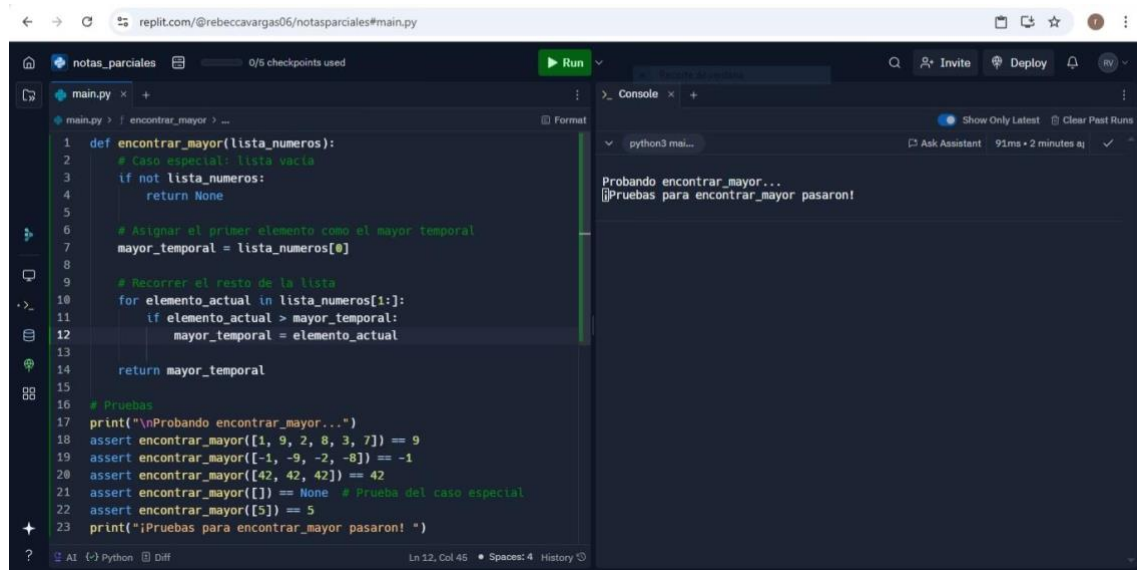
    return mayor_temporal

# Pruebas
print("\nProbando encontrar_mayor...")
assert encontrar_mayor([1, 9, 2, 8, 3, 7]) == 9
assert encontrar_mayor([-1, -9, -2, -8]) == -1
assert encontrar_mayor([42, 42, 42]) == 42
assert encontrar_mayor([]) == None # Prueba del caso especial
assert encontrar_mayor([5]) == 5

```



```
print("¡Pruebas para encontrar mayor pasaron!")
```



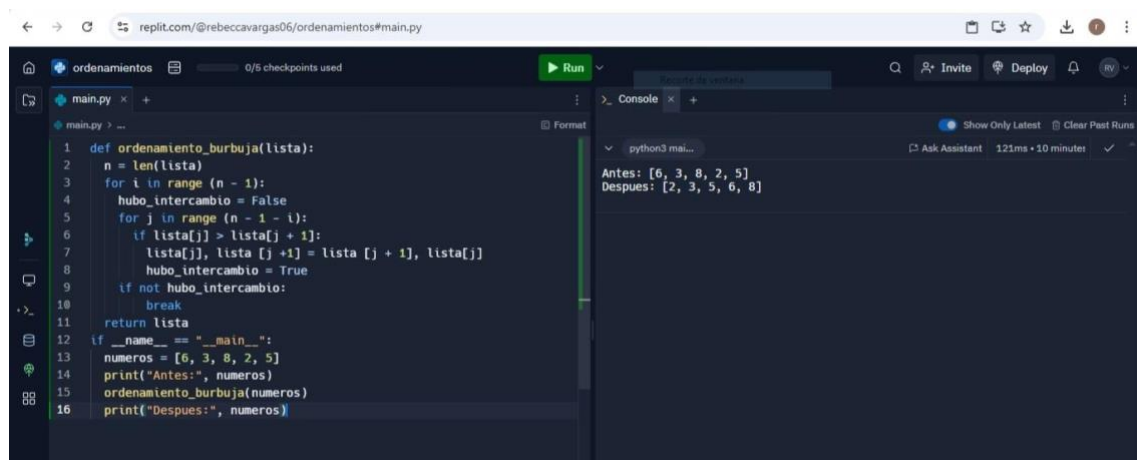
The screenshot shows a Replit Python environment with a file named `main.py`. The code defines a function `encontrar_mayor` that finds the maximum value in a list. It includes several test cases using `assert` statements and a final print statement indicating that the tests passed. The console output shows the function being tested and the tests passing.

```
1 def encontrar_mayor(lista_numeros):
2     # Caso especial: lista vacía
3     if not lista_numeros:
4         return None
5
6     # Asignar el primer elemento como el mayor temporal
7     mayor_temporal = lista_numeros[0]
8
9     # Recorrer el resto de la lista
10    for elemento_actual in lista_numeros[1:]:
11        if elemento_actual > mayor_temporal:
12            mayor_temporal = elemento_actual
13
14    return mayor_temporal
15
16 # Pruebas
17 print("\nProbando encontrar_mayor...")
18 assert encontrar_mayor([1, 9, 2, 8, 3, 7]) == 9
19 assert encontrar_mayor([-1, -9, -2, -8]) == -1
20 assert encontrar_mayor([42, 42, 42]) == 42
21 assert encontrar_mayor([]) == None # Prueba del caso especial
22 assert encontrar_mayor([5]) == 5
23 print("¡Pruebas para encontrar_mayor pasaron!")
```

Código: Encontrar número mayor

```
def ordenamiento_burbuja(lista):
    n = len(lista)
    for i in range (n - 1):
        hubo_intercambio = False
        for j in range (n - 1 - i):
            if lista[j] > lista[j + 1]:
                lista[j], lista[j + 1] = lista[j + 1], lista[j]
                hubo_intercambio = True
        if not hubo_intercambio:
            break
    return lista

if __name__ == "__main__":
    numeros = [6, 3, 8, 2, 5]
    print("Antes:", numeros)
    ordenamiento_burbuja(numeros)
    print("Despues:", numeros)
```



The screenshot shows a Replit Python environment with a file named `main.py`. The code implements a bubble sort algorithm. It includes a main block that initializes a list of numbers, prints it, sorts it using the `ordenamiento_burbuja` function, and prints the result. The console output shows the list before and after sorting.

```
1 def ordenamiento_burbuja(lista):
2     n = len(lista)
3     for i in range (n - 1):
4         hubo_intercambio = False
5         for j in range (n - 1 - i):
6             if lista[j] > lista[j + 1]:
7                 lista[j], lista[j + 1] = lista[j + 1], lista[j]
8                 hubo_intercambio = True
9         if not hubo_intercambio:
10            break
11    return lista
12
13 if __name__ == "__main__":
14     numeros = [6, 3, 8, 2, 5]
15     print("Antes:", numeros)
16     ordenamiento_burbuja(numeros)
17     print("Despues:", numeros)
```

Código: Ordenamiento burbuja


```

# Definición de la función de ordenamiento burbuja
def ordenamiento_burbuja(lista):
    n = len(lista)
    for i in range(n):
        for j in range(0, n - i - 1):
            if lista[j] > lista[j + 1]:
                lista[j], lista[j + 1] = lista[j + 1], lista[j]

#llamar a la función:
lista_a_ordenar = [64, 34, 25, 12, 22, 11, 90]
print(f"Lista original: {lista_a_ordenar}")

ordenamiento_burbuja(lista_a_ordenar) # Llamamos a la función
print(f"Lista ordenada: {lista_a_ordenar}")

# Prueba con assert:

# Caso 1: Lista desordenada
lista1 = [6, 3, 8, 2, 5]
ordenamiento_burbuja(lista1)
assert lista1 == [2, 3, 5, 6, 8], "Falló en Caso 1"

# Caso 2: Lista ya ordenada
lista2 = [1, 2, 3, 4, 5]
ordenamiento_burbuja(lista2)
assert lista2 == [1, 2, 3, 4, 5], "Falló en Caso 2"

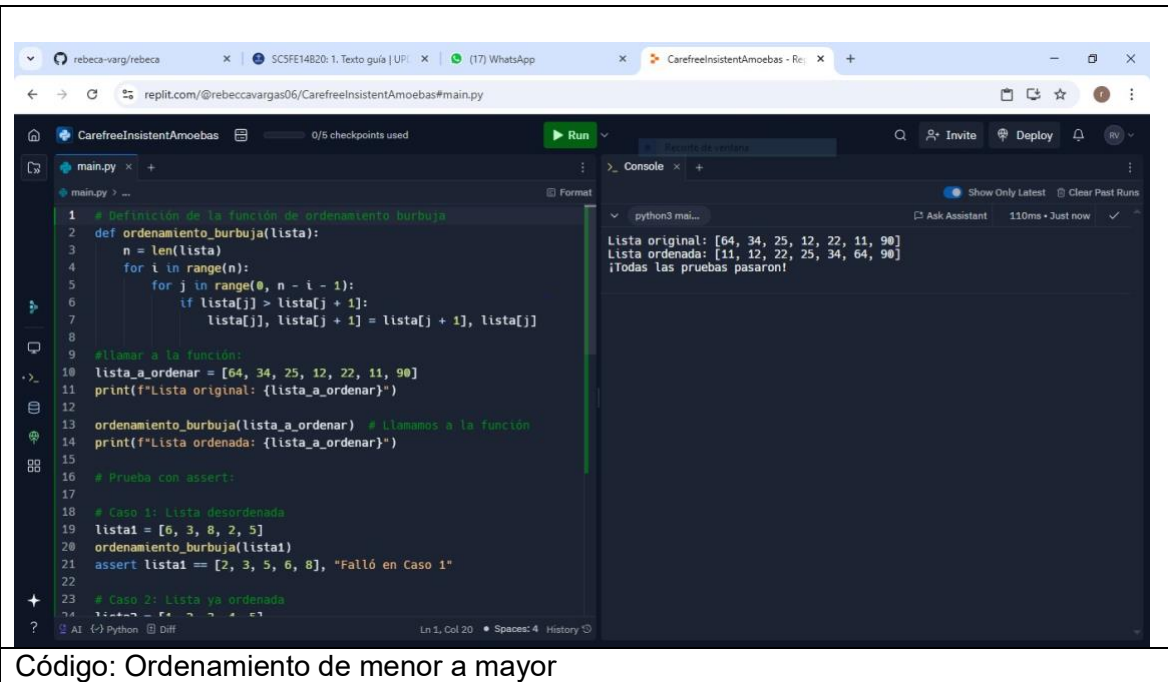
# Caso 3: Lista ordenada a la inversa (peor caso)
lista3 = [5, 4, 3, 2, 1]
ordenamiento_burbuja(lista3)
assert lista3 == [1, 2, 3, 4, 5], "Falló en Caso 3"

# Caso 4: Lista con elementos duplicados
lista4 = [5, 1, 4, 2, 5, 5, 2]
ordenamiento_burbuja(lista4)
assert lista4 == [1, 2, 2, 4, 5, 5, 5], "Falló en Caso 4"

# Casos borde
assert ordenamiento_burbuja([]) == None
assert ordenamiento_burbuja([42]) == None, "Fallo en caso borde"

print("¡Todas las pruebas pasaron!")

```



```
1 # Definición de la función de ordenamiento burbuja
2 def ordenamiento_burbuja(lista):
3     n = len(lista)
4     for i in range(n):
5         for j in range(0, n - i - 1):
6             if lista[j] > lista[j + 1]:
7                 lista[j], lista[j + 1] = lista[j + 1], lista[j]
8
9 #llamar a la función:
10 lista_a_ordenar = [64, 34, 25, 12, 22, 11, 90]
11 print(f'Lista original: {lista_a_ordenar}')
12
13 ordenamiento_burbuja(lista_a_ordenar) # llamamos a la función
14 print(f'Lista ordenada: {lista_a_ordenar}')
15
16 # Prueba con assert:
17
18 # Caso 1: Lista desordenada
19 lista1 = [6, 3, 8, 2, 5]
20 ordenamiento_burbuja(lista1)
21 assert lista1 == [2, 3, 5, 6, 8], "Falló en Caso 1"
22
23 # Caso 2: Lista ya ordenada
24 lista2 = [1, 2, 3, 4, 5]
```

Lista original: [64, 34, 25, 12, 22, 11, 90]
Lista ordenada: [11, 12, 22, 25, 34, 64, 90]
¡Todas las pruebas pasaron!

Código: Ordenamiento de menor a mayor

+