

UNIVERSIDAD PRIVADA DOMINGO SAVIO

FACULTAD DE INGENIERIA



Actividad

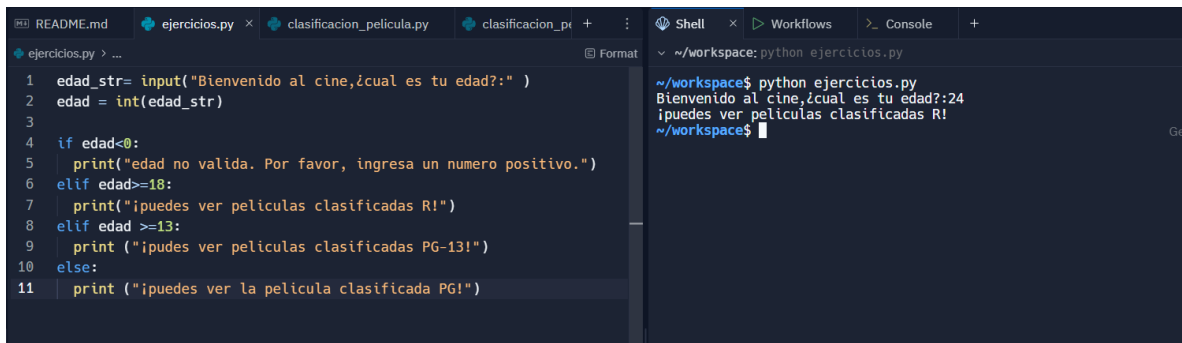
Título: Ejercicios de Replit

Docente: Ing. Jimmy Nataniel Requena Llorentty

Materia: Programación 2

Estudiante: Sarai Alejandra Vidaurre

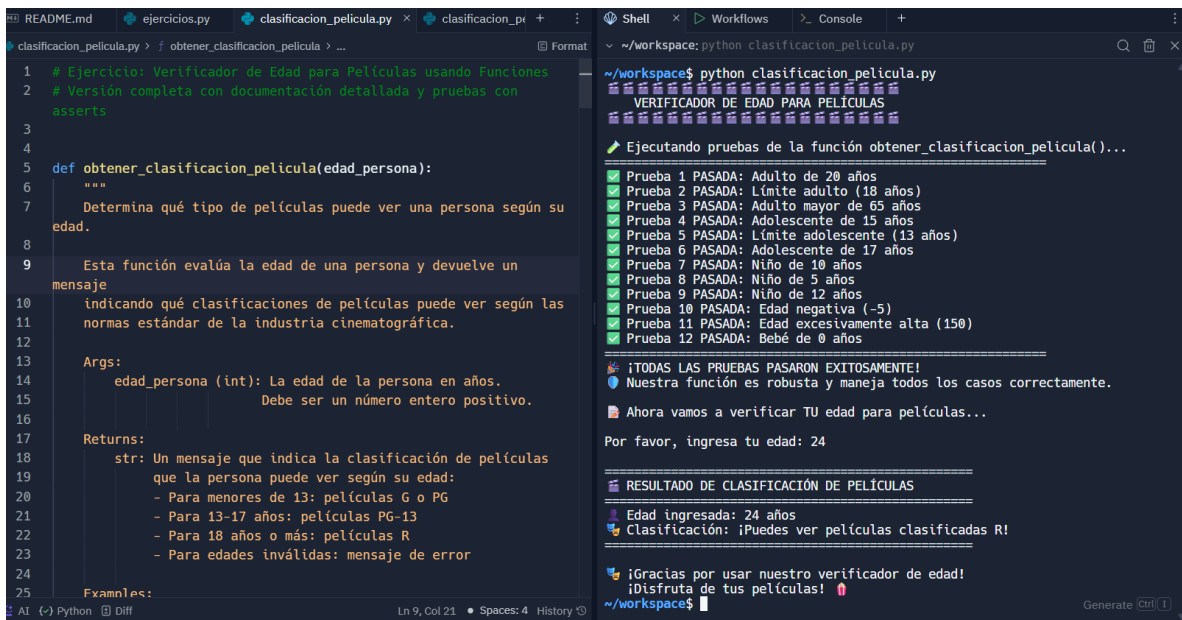
Junio del 2025
Santa Cruz – Bolivia



```
1 edad_str= input("Bienvenido al cine,¿cual es tu edad?:" )
2 edad = int(edad_str)
3
4 if edad<0:
5     print("edad no valida. Por favor, ingresa un numero positivo.")
6 elif edad>=18:
7     print("¡puedes ver peliculas clasificadas R!")
8 elif edad >=13:
9     print ("¡pudes ver peliculas clasificadas PG-13!")
10 else:
11     print ("¡puedes ver la pelicula clasificada PG!")
```

```
~/workspace$ python ejercicios.py
Bienvenido al cine,¿cual es tu edad?:24
¡puedes ver peliculas clasificadas R!
~/workspace$
```

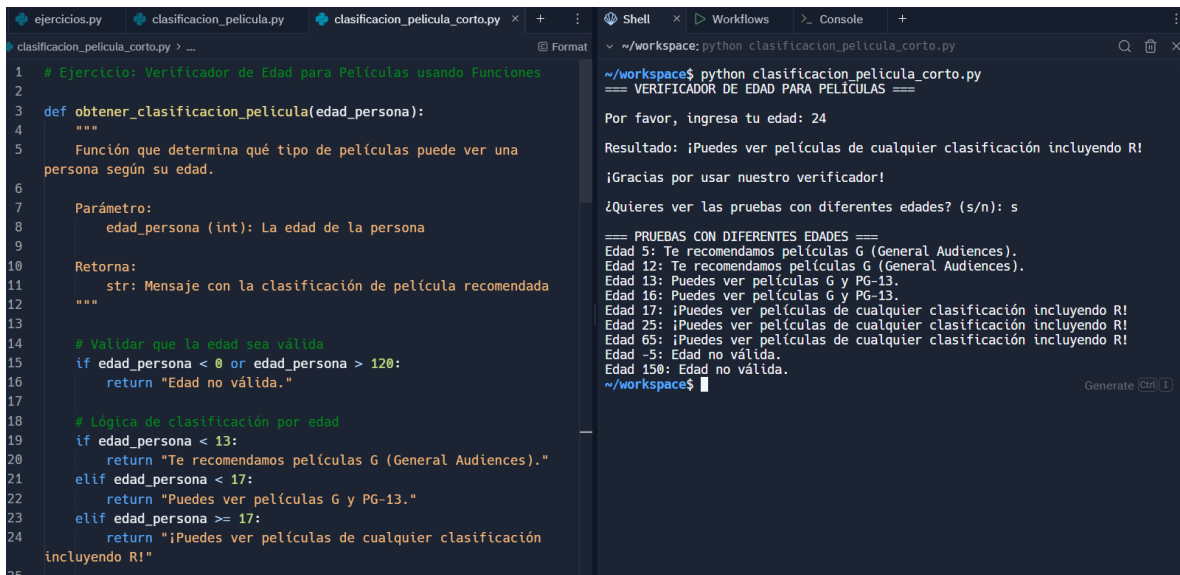
Ilustración 1 ejercicio #1



```
1 # Ejercicio: Verificador de Edad para Películas usando funciones
2 # Versión completa con documentación detallada y pruebas con
3 # asserts
4
5 def obtener_clasificacion_pelicula(edad_persona):
6     """
7     Determina qué tipo de películas puede ver una persona según su
8     edad.
9
10    Esta función evalúa la edad de una persona y devuelve un
11    mensaje
12    indicando qué clasificaciones de películas puede ver según las
13    normas estándar de la industria cinematográfica.
14
15    Args:
16        edad_persona (int): La edad de la persona en años.
17        Debe ser un número entero positivo.
18
19    Returns:
20        str: Un mensaje que indica la clasificación de películas
21        que la persona puede ver según su edad:
22        - Para menores de 13: películas G o PG
23        - Para 13-17 años: películas PG-13
24        - Para 18 años o más: películas R
25        - Para edades inválidas: mensaje de error
26
27    Examples:
28        obtener_clasificacion_pelicula(24)
29        """
```

```
~/workspace$ python clasificacion_pelicula.py
VERIFICADOR DE EDAD PARA PELÍCULAS
Ejecutando pruebas de la función obtener_clasificacion_pelicula()...
✓ Prueba 1 PASADA: Adulto de 20 años
✓ Prueba 2 PASADA: Límite adulto (18 años)
✓ Prueba 3 PASADA: Adulto mayor de 65 años
✓ Prueba 4 PASADA: Adolescente de 15 años
✓ Prueba 5 PASADA: Límite adolescente (13 años)
✓ Prueba 6 PASADA: Adolescente de 17 años
✓ Prueba 7 PASADA: Niño de 10 años
✓ Prueba 8 PASADA: Niño de 5 años
✓ Prueba 9 PASADA: Niño de 12 años
✓ Prueba 10 PASADA: Edad negativa (-5)
✓ Prueba 11 PASADA: Edad excesivamente alta (150)
✓ Prueba 12 PASADA: Bebé de 0 años
¡TODAS LAS PRUEBAS PASARON EXITOSAMENTE!
Nuestra función es robusta y maneja todos los casos correctamente.
Ahora vamos a verificar TU edad para películas...
Por favor, ingresa tu edad: 24
RESULTADO DE CLASIFICACIÓN DE PELÍCULAS
Edad ingresada: 24 años
Clasificación: ¡Puedes ver películas clasificadas R!
¡Gracias por usar nuestro verificador de edad!
¡Disfruta de tus películas!
```

Ilustración 2 ejercicio clasificación de película con asserts



```
ejercicios.py clasificacion_pelicula.py clasificacion_pelicula_corto.py + : Shell x Workflows >_ Console +
clasificacion_pelicula_corto.py > ... Format ~ /workspace: python clasificacion_pelicula_corto.py

1 # Ejercicio: Verificador de Edad para Películas usando Funciones
2
3 def obtener_clasificacion_pelicula(edad_persona):
4     """
5     Función que determina qué tipo de películas puede ver una
6     persona según su edad.
7
8     Parámetro:
9         edad_persona (int): La edad de la persona
10
11     Retorna:
12         str: Mensaje con la clasificación de película recomendada
13     """
14
15     # Validar que la edad sea válida
16     if edad_persona < 0 or edad_persona > 120:
17         return "Edad no válida."
18
19     # Lógica de clasificación por edad
20     if edad_persona < 13:
21         return "Te recomendamos películas G (General Audiences)."
```

```
~/workspace$ python clasificacion_pelicula_corto.py
=== VERIFICADOR DE EDAD PARA PELÍCULAS ===

Por favor, ingresa tu edad: 24

Resultado: ¡Puedes ver películas de cualquier clasificación incluyendo R!

¡Gracias por usar nuestro verificador!

¿Quieres ver las pruebas con diferentes edades? (s/n): s

=== PRUEBAS CON DIFERENTES EDADES ===
Edad 5: Te recomendamos películas G (General Audiences).
Edad 12: Te recomendamos películas G (General Audiences).
Edad 13: Puedes ver películas G y PG-13.
Edad 16: Puedes ver películas G y PG-13.
Edad 17: ¡Puedes ver películas de cualquier clasificación incluyendo R!
Edad 25: ¡Puedes ver películas de cualquier clasificación incluyendo R!
Edad 65: ¡Puedes ver películas de cualquier clasificación incluyendo R!
Edad -5: Edad no válida.
Edad 150: Edad no válida.
~/workspace$
```

Ilustración 3 Clasificación de película con funciones

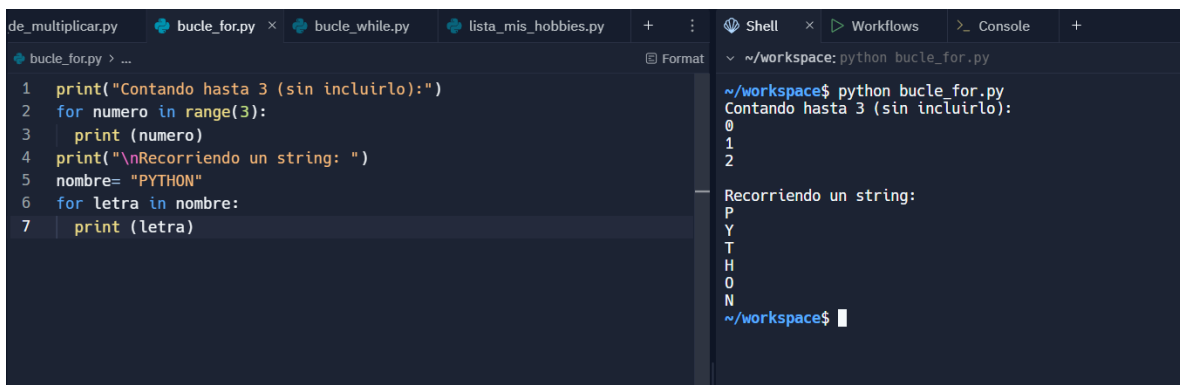


```
clasificacion_pelicula_corto.py tabla_de_multiplicar.py x bucle_for.py + : Shell x Workflows >_ Console +
tabla_de_multiplicar.py > ... Format ~ /workspace: python tabla_de_multiplicar.py

1 num_tabla = int(input( "introduce el numero de la tabla: " ))
2 print(f"--- Tabla del {num_tabla} ---")
3 for i in range(1, 11):
4     resultado = num_tabla * i
5     print(f"{num_tabla} x {i} = {resultado}")

~/workspace$ python tabla_de_multiplicar.py
introduce 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
~/workspace$
```

Ilustración 4 Tabla de multiplicar



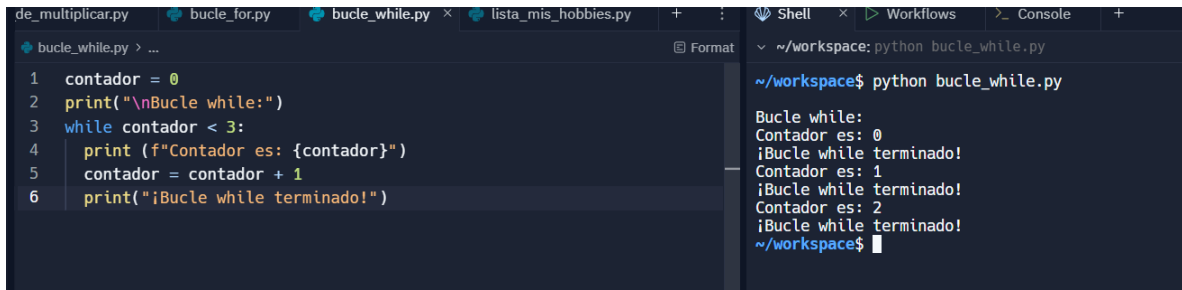
```
de_multiplicar.py bucle_for.py x bucle_while.py lista_mis_hobbies.py + : Shell x Workflows >_ Console +
bucle_for.py > ... Format ~ /workspace: python bucle_for.py

1 print("Contando hasta 3 (sin incluirlo):")
2 for numero in range(3):
3     print(numero)
4 print("\nRecorriendo un string: ")
5 nombre= "PYTHON"
6 for letra in nombre:
7     print(letra)

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

Recorriendo un string:
P
Y
T
H
O
N
~/workspace$
```

Ilustración 5 bucle for

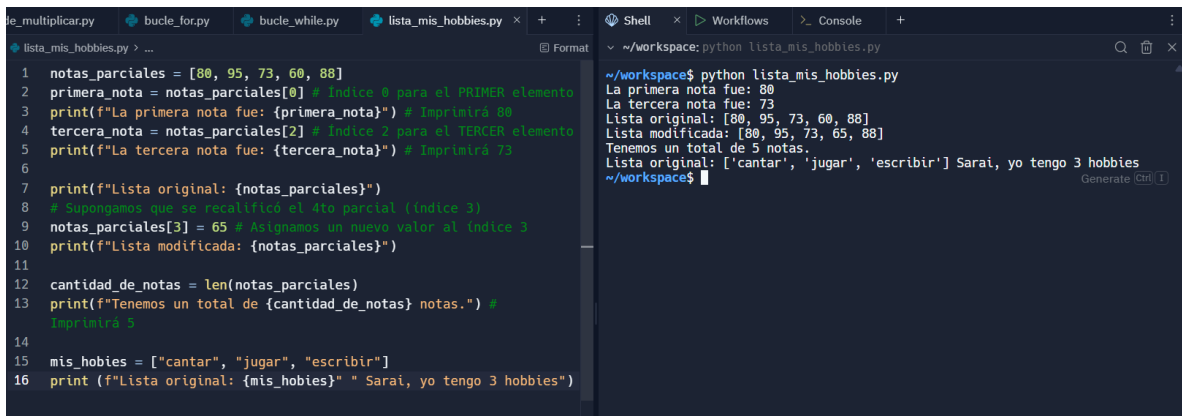


The screenshot shows a code editor with a file named `bucle_while.py`. The code is a Python while loop that counts from 0 to 3. The output in the console shows the loop executing three times, printing the counter value and a termination message.

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

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

Ilustración 6 bucle while

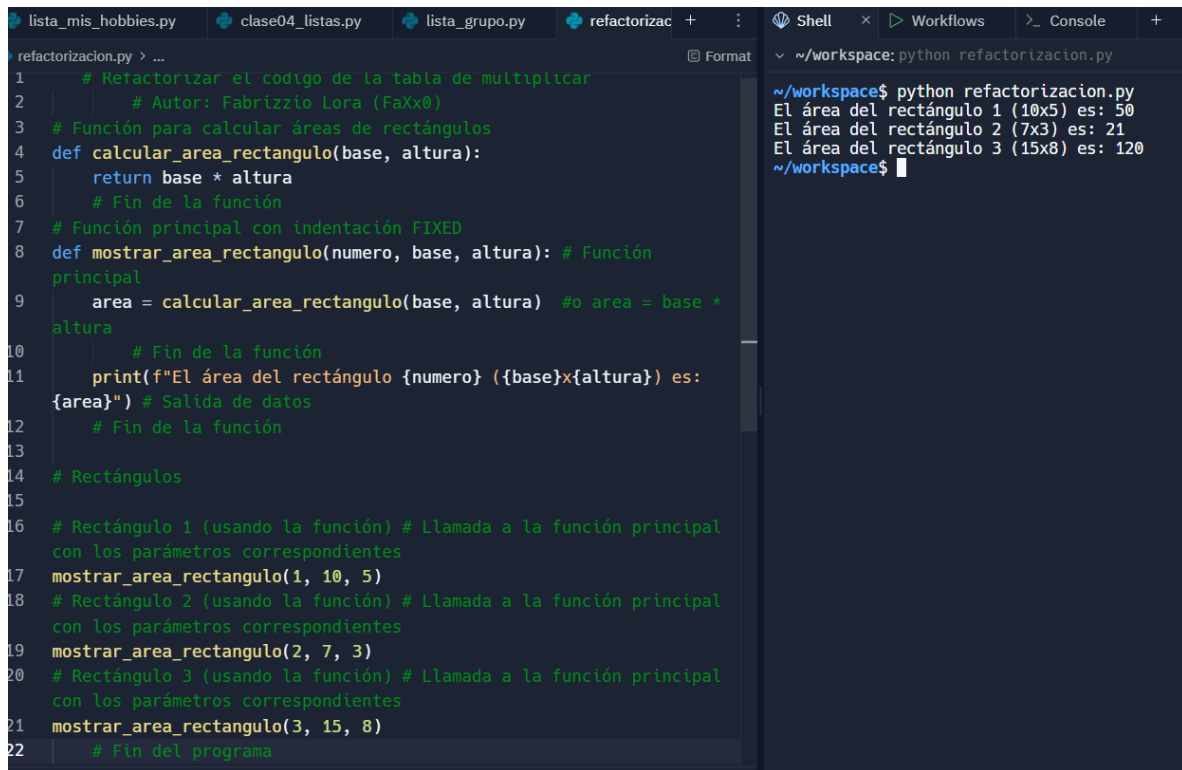


The screenshot shows a code editor with a file named `lista_mis_hobbies.py`. The code is a Python script that creates a list of partial notes, prints the first and third elements, modifies the fourth element, and prints the modified list. It also prints the length of the list and a message about the total number of notes. Finally, it prints a list of hobbies and a message about the user's hobbies.

```
1 notas_parciales = [80, 95, 73, 60, 88]
2 primera_not = notas_parciales[0] # Índice 0 para el PRIMER elemento
3 print(f"La primera nota fue: {primera_not}") # Imprimirá 80
4 tercera_not = notas_parciales[2] # Índice 2 para el TERCER elemento
5 print(f"La tercera nota fue: {tercera_not}") # Imprimirá 73
6
7 print(f"Lista original: {notas_parciales}")
8 # Supongamos que se recalificó el 4to parcial (índice 3)
9 notas_parciales[3] = 65 # Asignamos un nuevo valor al índice 3
10 print(f"Lista modificada: {notas_parciales}")
11
12 cantidad_de_notas = len(notas_parciales)
13 print(f"Tenemos un total de {cantidad_de_notas} notas.") #
14     Imprimirá 5
15
16 mis_hobbies = ["cantar", "jugar", "escribir"]
17 print (f"Lista original: {mis_hobbies}" " Sarai, yo tengo 3 hobbies")
```

```
~/workspace$ python lista_mis_hobbies.py
La primera nota fue: 80
La tercera nota fue: 73
Lista original: [80, 95, 73, 60, 88]
Lista modificada: [80, 95, 73, 65, 88]
Tenemos un total de 5 notas.
Lista original: ['cantar', 'jugar', 'escribir'] Sarai, yo tengo 3 hobbies
~/workspace$
```

Ilustración 7 lista mis hobbies

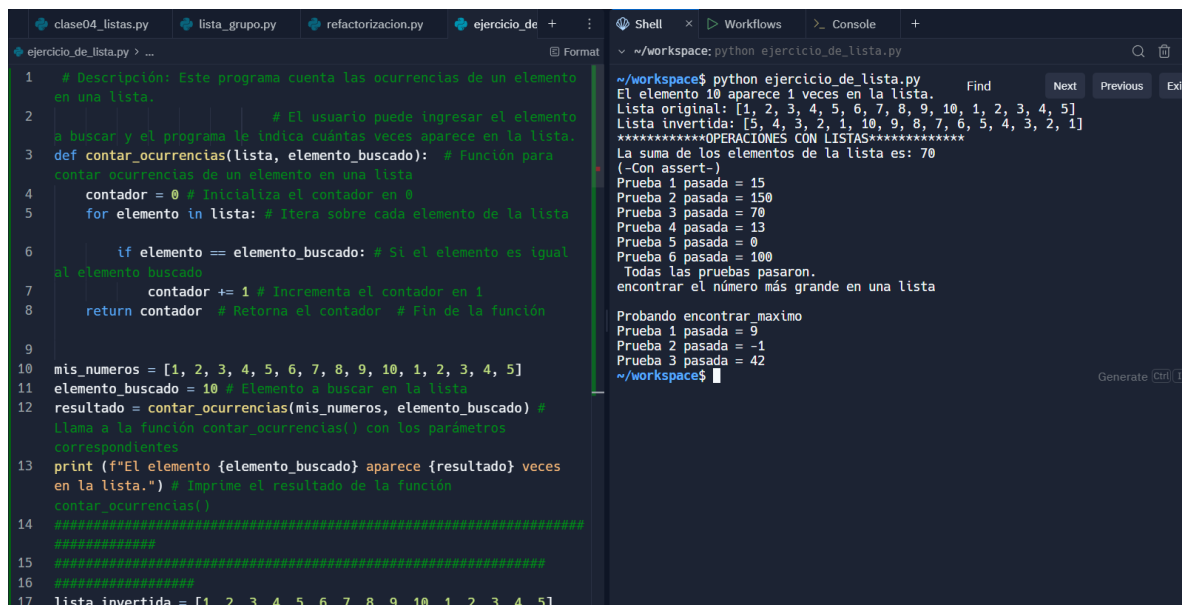


The screenshot shows a code editor with a file named `refactorizacion.py`. The code defines a function `calcular_area_rectangulo` and a main function `mostrar_area_rectangulo`. It then calls `mostrar_area_rectangulo` for three rectangles with dimensions (1, 10), (2, 7), and (3, 15). The output in the console shows the calculated areas: 50, 21, and 120.

```
1 # Refactorizar el código de la tabla de multiplicar
2 # Autor: Fabrizio Lora (FaXx0)
3 # Función para calcular áreas de rectángulos
4 def calcular_area_rectangulo(base, altura):
5     return base * altura
6     # Fin de la función
7 # Función principal con indentación FIXED
8 def mostrar_area_rectangulo(numero, base, altura): # Función principal
9     area = calcular_area_rectangulo(base, altura) #o area = base *
10    altura
11    # Fin de la función
12    print(f"El área del rectángulo {numero} ({base}x{altura}) es:
13    {area}") # Salida de datos
14    # Fin de la función
15
16 # Rectángulos
17
18 # Rectángulo 1 (usando la función) # Llamada a la función principal
19 # con los parámetros correspondientes
20 mostrar_area_rectangulo(1, 10, 5)
21 # Rectángulo 2 (usando la función) # Llamada a la función principal
22 # con los parámetros correspondientes
23 mostrar_area_rectangulo(2, 7, 3)
24 # Rectángulo 3 (usando la función) # Llamada a la función principal
25 # con los parámetros correspondientes
26 mostrar_area_rectangulo(3, 15, 8)
27 # Fin del programa
```

```
~/workspace$ python refactorizacion.py
El área del rectángulo 1 (10x5) es: 50
El área del rectángulo 2 (7x3) es: 21
El área del rectángulo 3 (15x8) es: 120
~/workspace$
```

Ilustración 8 ejercicio de refactorización



The screenshot shows a code editor with a file named `ejercicio_de_lista.py`. The code defines a function `contar_ocurrencias` that counts the occurrences of an element in a list. It then uses this function to count the occurrences of the number 10 in a list of numbers. The output in the console shows the result and some test cases.

```
1 # Descripción: Este programa cuenta las ocurrencias de un elemento
2 # en una lista.
3 # El usuario puede ingresar el elemento
4 # a buscar y el programa le indica cuántas veces aparece en la lista.
5 def contar_ocurrencias(lista, elemento_buscado): # Función para
6     contar ocurrencias de un elemento en una lista
7     contador = 0 # Inicializa el contador en 0
8     for elemento in lista: # Itera sobre cada elemento de la lista
9
10        if elemento == elemento_buscado: # Si el elemento es igual
11        al elemento buscado
12            contador += 1 # Incrementa el contador en 1
13        return contador # Retorna el contador # Fin de la función
14
15 mis_numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5]
16 elemento_buscado = 10 # Elemento a buscar en la lista
17 resultado = contar_ocurrencias(mis_numeros, elemento_buscado) #
18 llama a la función contar_ocurrencias() con los parámetros
19 correspondientes
20 print(f"El elemento {elemento_buscado} aparece {resultado} veces
21 en la lista.") # Imprime el resultado de la función
22 contar_ocurrencias()
23
24 *****
25 *****
26 *****
27 lista invertida = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5]
```

```
~/workspace$ python ejercicio_de_lista.py
El elemento 10 aparece 1 veces en la lista.
Lista original: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5]
Lista invertida: [5, 4, 3, 2, 1, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
*****OPERACIONES CON LISTAS*****
La suma de los elementos de la lista es: 70
(-Con assert-)
Prueba 1 pasada = 15
Prueba 2 pasada = 150
Prueba 3 pasada = 70
Prueba 4 pasada = 13
Prueba 5 pasada = 0
Prueba 6 pasada = 100
Todas las pruebas pasaron.
encontrar el número más grande en una lista

Probando encontrar_maximo
Prueba 1 pasada = 9
Prueba 2 pasada = -1
Prueba 3 pasada = 42
~/workspace$
```

Ilustración 9 ejercicio de listas clase 05