

UNIVERSIDAD PRIVADA DOMINGO SAVIO

FACULTAD DE INGENIERIA



Actividad

Título: Ejercicios de Replit

Docente: Ing. Jimmy Nataniel Requena Llorentty

Materia: Programación 2

Estudiante: LORA COLODRO FABRIZZIO

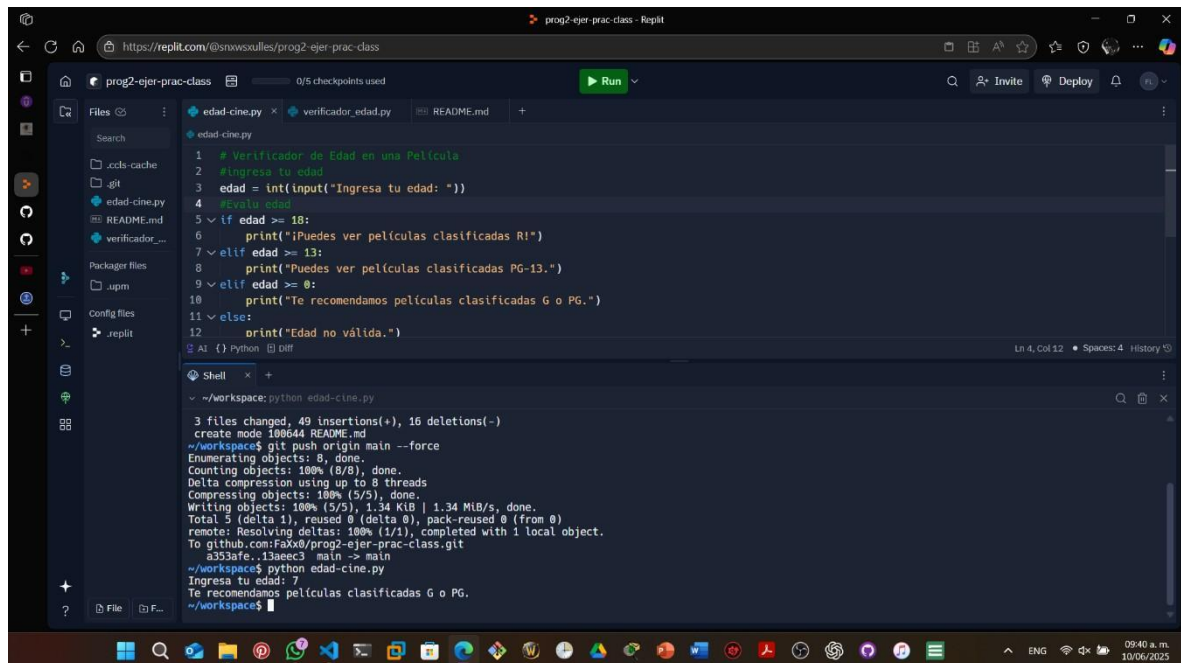
Junio del 2025
Santa Cruz – Bolivia

Fecha y hora actual: 2025-06-17 10:07:13

01-. Ejercicio simple-edad-cine.

Descripción:

Este programa solicita la edad del usuario y le recomienda películas según su clasificación.



The screenshot shows a Replit IDE window titled "prog2-ejer-prac-class - Replit". The browser address bar shows "https://replit.com/@snwxsulles/prog2-ejer-prac-class". The file explorer on the left shows a project named "prog2-ejer-prac-class" with files "edad-cine.py", "verificador_edad.py", and "README.md". The main editor displays the content of "edad-cine.py", which is a Python script for a movie recommendation system based on age. The script prompts the user to enter their age and then prints a recommendation based on the age range.

```
1 # Verificador de Edad en una Pelicula
2 # Ingrese tu edad
3 edad = int(input("Ingresa tu edad: "))
4 # Verifica la edad
5 if edad >= 18:
6     print("¡Puedes ver películas clasificadas R!")
7 elif edad >= 13:
8     print("Puedes ver películas clasificadas PG-13.")
9 elif edad >= 8:
10    print("Te recomendamos películas clasificadas G o PG.")
11 else:
12    print("Edad no válida.")
```

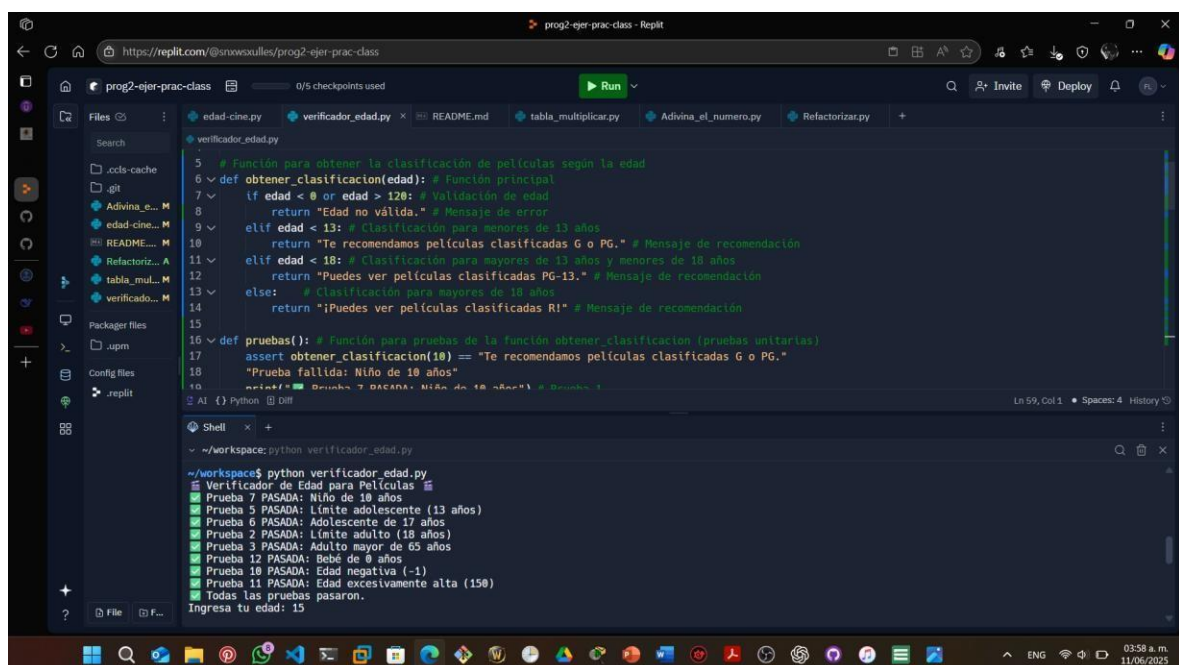
The terminal output shows the execution of the script. It starts with a git push command, followed by running the Python script. The user enters the age "7", and the program outputs "Te recomendamos películas clasificadas G o PG.".

```
~/workspace$ git push origin main --force
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.34 KiB | 1.34 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Faxx0/prog2-ejer-prac-class.git
  a353afe..13aee3  main -> main
~/workspace$ python edad-cine.py
Ingresa tu edad: 7
Te recomendamos películas clasificadas G o PG.
~/workspace$
```

02-. Ejercicio - edad-cine-completo.

Descripción:

Este programa solicita la edad del usuario y le recomienda películas según su clasificación. El programa también incluye pruebas unitarias para validar su funcionamiento. El programa se ejecuta en un bucle hasta que el usuario ingrese un número válido. El programa termina cuando el usuario ingresa un número válido. El programa muestra un mensaje de error si el usuario ingresa un número inválido. El programa muestra un mensaje de éxito si el usuario ingresa un número válido. El programa muestra un mensaje de despedida al final.



The screenshot shows a Replit IDE window titled "prog2-ejer-prac-class - Replit". The browser address bar shows "https://replit.com/@snuxvulles/prog2-ejer-prac-class". The file explorer on the left shows a project named "prog2-ejer-prac-class" with files: "edad-cine.py", "verificador_edad.py", "README.md", "tabla_multiplicar.py", "Adivina_el_numero.py", and "Refactorizar.py". The main editor displays the code for "verificador_edad.py".

```
5 # Función para obtener la clasificación de películas según la edad
6 def obtener_clasificacion(edad): # Función principal
7     if edad < 0 or edad > 120: # Validación de edad
8         return "Edad no válida." # Mensaje de error
9     elif edad < 13: # Clasificación para menores de 13 años
10        return "Te recomendamos películas clasificadas G o PG." # Mensaje de recomendación
11    elif edad < 18: # Clasificación para mayores de 13 años y menores de 18 años
12        return "Puedes ver películas clasificadas PG-13." # Mensaje de recomendación
13    else: # Clasificación para mayores de 18 años
14        return "¡Puedes ver películas clasificadas R!" # Mensaje de recomendación
15
16 def pruebas(): # Función para pruebas de la función obtener_clasificacion (pruebas unitarias)
17     assert obtener_clasificacion(10) == "Te recomendamos películas clasificadas G o PG."
18     "Prueba fallida: Niño de 10 años"
19
20 # Ejecución de pruebas
21 pruebas()
22
23 # Mensaje de bienvenida
24 print("Bienvenido al verificador de edad para películas.")
25
26 # Bucle para solicitar la edad del usuario
27 while True:
28     edad = input("Ingresa tu edad: ")
29     if edad.isdigit():
30         edad = int(edad)
31         obtener_clasificacion(edad)
32     else:
33         print("Por favor, ingresa un número válido.")
34
35 # Mensaje de despedida
36 print("¡Gracias por usar el verificador de edad para películas!")
```

The terminal output shows the execution of the program:

```
~/workspace$ python verificador_edad.py
Verificador de Edad para Películas
Prueba 7 PASADA: Niño de 10 años
Prueba 5 PASADA: Límite adolescente (13 años)
Prueba 6 PASADA: Adolescente de 17 años
Prueba 2 PASADA: Límite adulto (18 años)
Prueba 3 PASADA: Adulto mayor de 65 años
Prueba 12 PASADA: Bebé de 6 años
Prueba 10 PASADA: Edad negativa (-1)
Prueba 11 PASADA: Edad excesivamente alta (150)
Todas las pruebas pasaron.
Ingresa tu edad: 15
```

The system tray at the bottom shows the time as 03:58 a.m. on 11/06/2023.

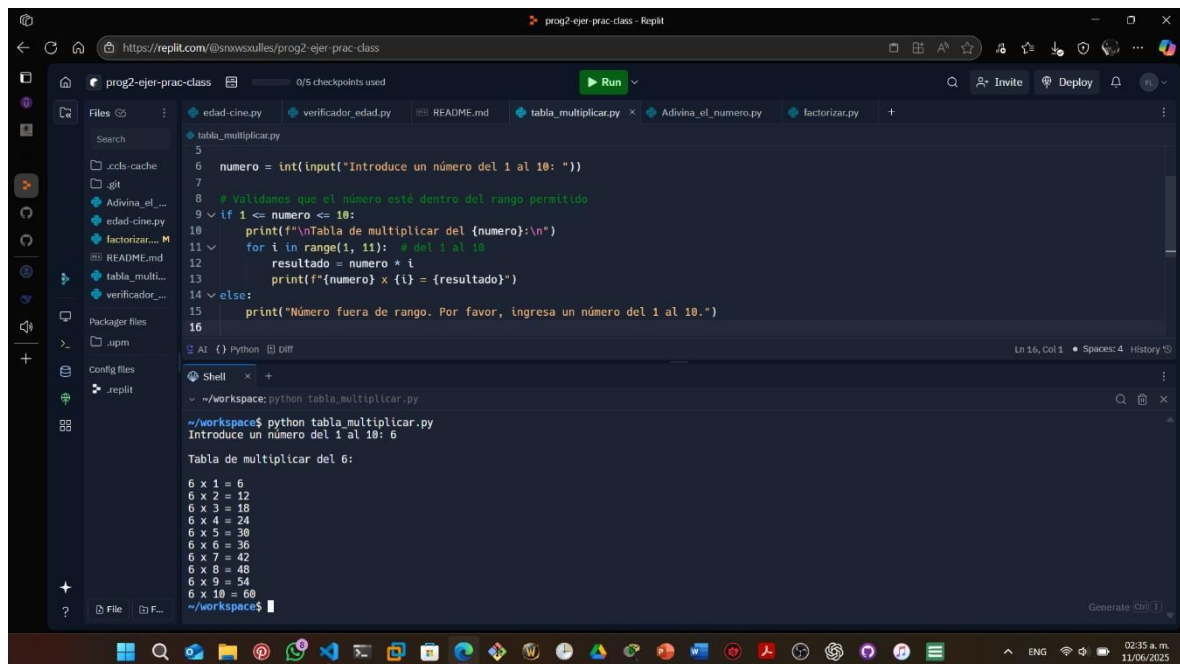
03-. Ejercicio - Tablas-multiplicar.

Descripción:

Imprime las tablas de multiplicar del 1 al 10.

Tabla de multiplicar de un número dado por el usuario (entre 1 y 10)

Entrada de datos # Validación de datos # Condicional para validar el rango del número ingresado por el usuario # Bucle para imprimir la tabla de multiplicar del número ingresado por el usuario # Iteración sobre el rango del 1 al 1



```
5
6  numero = int(input("Introduce un número del 1 al 10: "))
7
8  # Validamos que el número esté dentro del rango permitido
9  if 1 <= numero <= 10:
10     print(f"\nTabla de multiplicar del {numero}:\n")
11     for i in range(1, 11): # del 1 al 10
12         resultado = numero * i
13         print(f"{numero} x {i} = {resultado}")
14 else:
15     print("Número fuera de rango. Por favor, ingresa un número del 1 al 10.")
16
```

```
~/workspace$ python tabla_multiplicar.py
Introduce un número del 1 al 10: 6

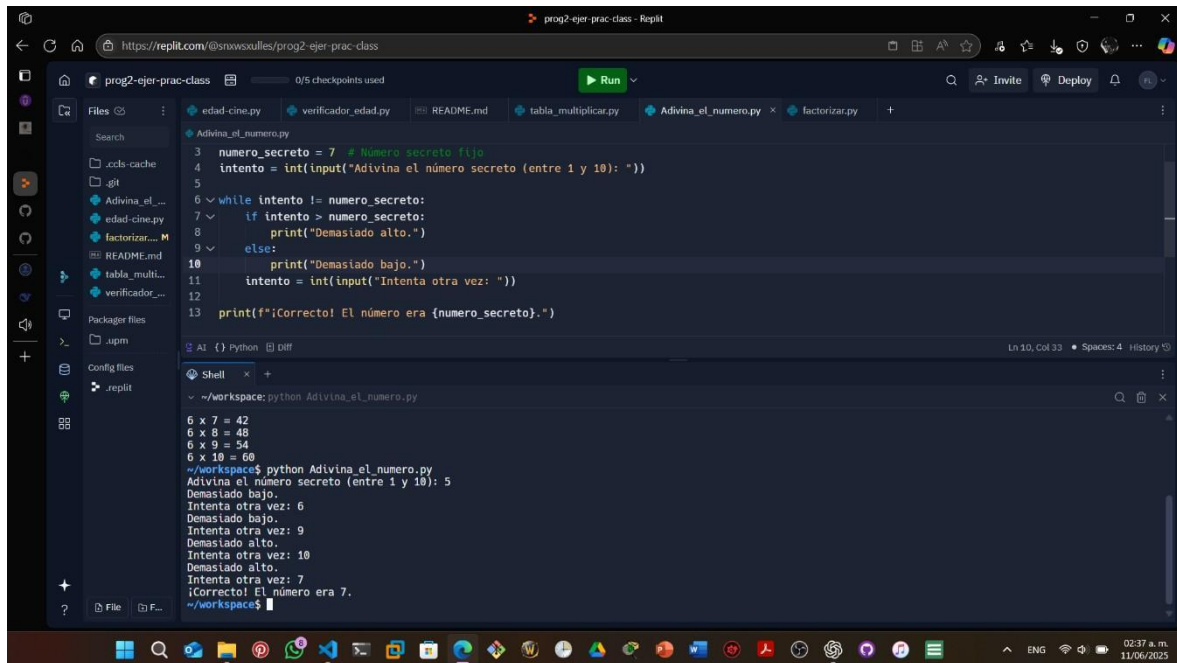
Tabla de multiplicar del 6:

6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
~/workspace$
```

04-. Ejercicio - Adivina el número.

Descripción:

Juego en el que el usuario debe adivinar un número secreto. indica si el número ingresado es demasiado alto o demasiado bajo. continúa hasta que el usuario adivine el número secreto. El número secreto es un número fijo (7) y el usuario tiene que adivinarlo.



```
prog2-ejer-prac-class - Replit
https://replit.com/@snxwsxulles/prog2-ejer-prac-class
0/5 checkpoints used
Run

Files
  Adivina_el_numero.py
  edad_cine.py
  verificador_edad.py
  README.md
  tabla_multiplicar.py
  Adivina_el_numero.py
  factorizar.py
  ...

Search
  Adivina_el...
  edad_cine.py
  factorizar...
  README.md
  tabla_multi...
  verificador...

Package files
  .ipynb
  Config files
  .replit

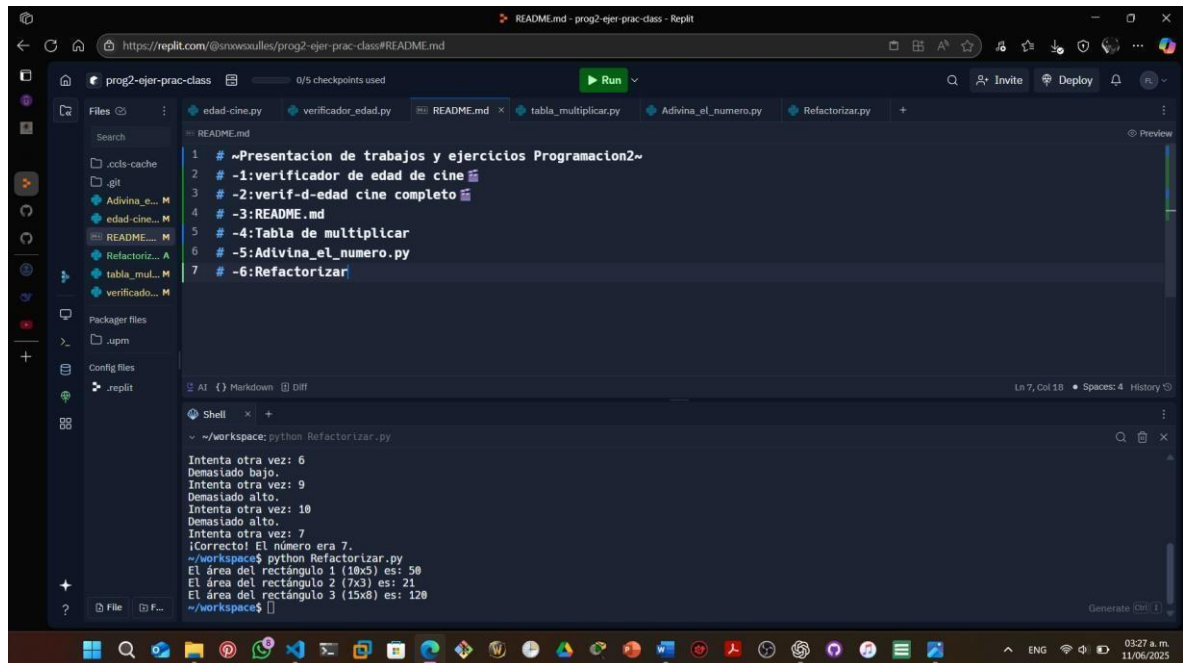
AI {} Python Diff
Ln 10, Col 33 • Spaces: 4 History

~/workspace: python Adivina_el_numero.py
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
~/workspace$ python Adivina_el_numero.py
Adivina el número secreto (entre 1 y 10): 5
Demasiado bajo.
Intenta otra vez: 6
Demasiado bajo.
Intenta otra vez: 9
Demasiado alto.
Intenta otra vez: 10
Demasiado alto.
Intenta otra vez: 7
¡Correcto! El número era 7.
~/workspace$
```

05-. Ejercicio - README.md.

Descripción:

Aquí escribes a gusto en este caso en lista puse cada uno de los ejercicios que realicé



The screenshot shows a Replit workspace for a project named "prog2-ejer-prac-class". The main editor displays a README.md file with the following content:

```
1 # ~Presentacion de trabajos y ejercicios Programacion2~
2 # ~1:verificador de edad de cine 🎬
3 # ~2:verif-d-edad cine completo 🎬
4 # ~3:README.md
5 # ~4:Tabla de multiplicar
6 # ~5:Adivina_el_numero.py
7 # ~6:Refactorizar
```

The left sidebar shows a file explorer with the following files: `edad_cine.py`, `verificador_edad.py`, `README.md`, `tabla_multiplicar.py`, `Adivina_el_numero.py`, and `Refactorizar.py`. The bottom panel shows a terminal window with the following output:

```
~/workspace$ python Refactorizar.py
Intenta otra vez: 6
Demasiado bajo.
Intenta otra vez: 9
Demasiado alto.
Intenta otra vez: 10
Demasiado alto.
Intenta otra vez: 7
¡Correcto! El número era 7.
~/workspace$ python Refactorizar.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$
```

The terminal window also shows a "Generate Code" button at the bottom right.

06-. Ejercicio - Refactorizar.

Descripción:

Este código define dos funciones para calcular y mostrar el área de rectángulos:

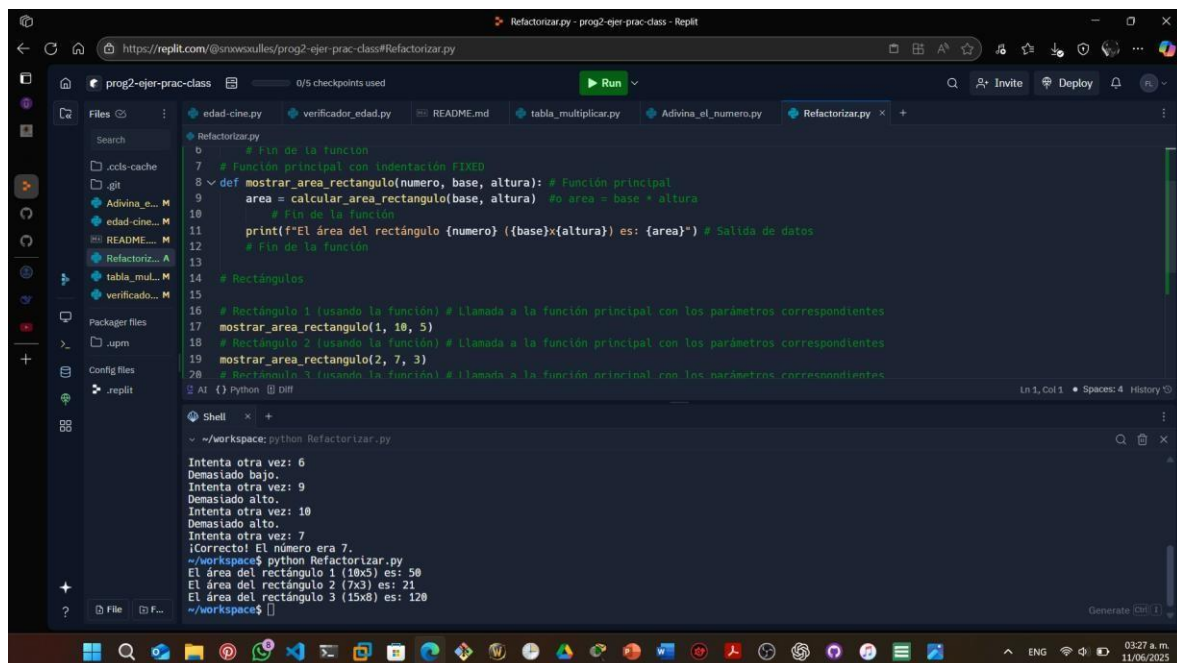
calcular área rectángula (base, altura)

Recibe la base y la altura de un rectángulo, Retorna el área calculada como $\text{base} * \text{altura}$.

mostrar área rectángula (número, base, altura)

Usa la función anterior para calcular el área del rectángulo.

Imprime un mensaje con el número del rectángulo, sus dimensiones y el área calculada.



```
Refactorizar.py - prog2-ejer-prac-class - Replit
https://replit.com/@snwxsulles/prog2-ejer-prac-class#Refactorizar.py
0/5 checkpoints used
Run

Files
Search
  .cde-cache
  .git
  Adivina_e... M
  edad-cine... M
  README... M
  Refactoriz... A
  tabla_mul... M
  verificado... M
Package files
  .upm
Config files
  .replit

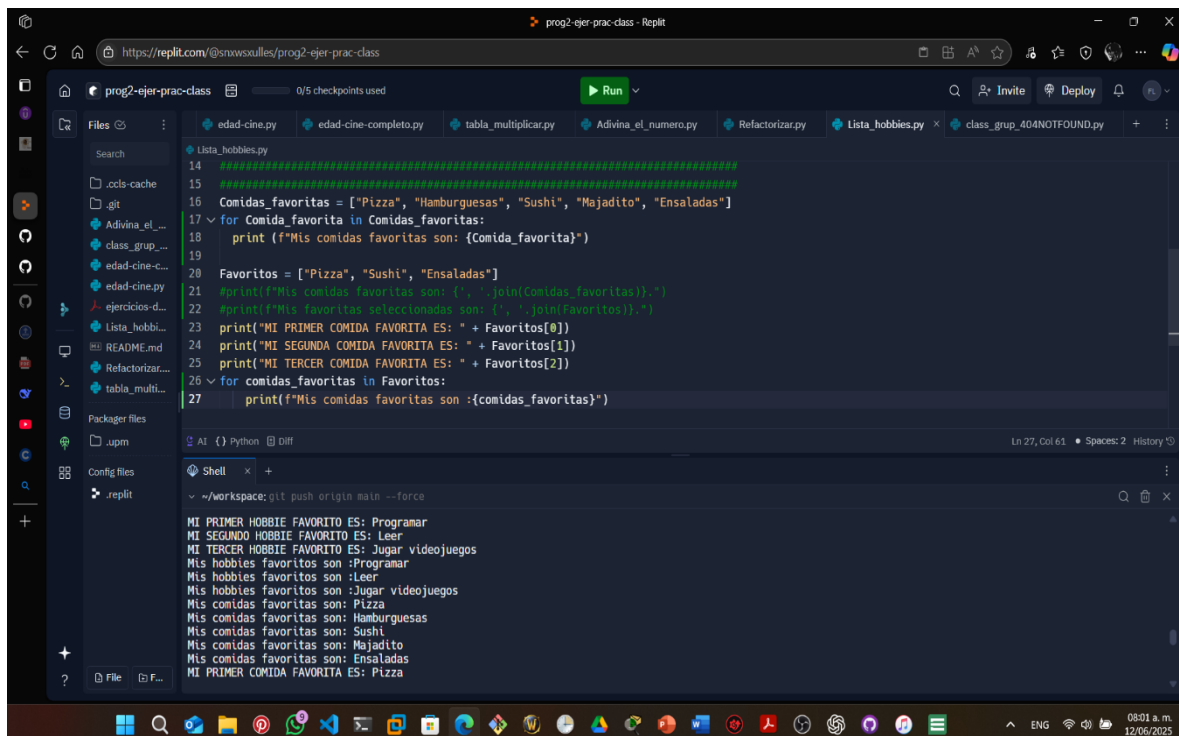
Refactorizar.py
  6 # Fin de la función
  7 # Función principal con indentación FIXED
  8 def mostrar_area_rectangulo(numero, base, altura): # Función principal
  9     area = calcular_area_rectangulo(base, altura) # area = base * altura
 10     # Fin de la función
 11     print(f"El área del rectángulo {numero} ({base}x{altura}) es: {area}") # Salida de datos
 12     # Fin de la función
 13
 14 # Rectángulos
 15
 16 # Rectángulo 1 (usando la función) # Llamada a la función principal con los parámetros correspondientes
 17 mostrar_area_rectangulo(1, 10, 5)
 18 # Rectángulo 2 (usando la función) # Llamada a la función principal con los parámetros correspondientes
 19 mostrar_area_rectangulo(2, 7, 3)
 20 # Rectángulo 3 (usando la función) # Llamada a la función principal con los parámetros correspondientes
 21
 22 Shell
  ~ /workspace: python Refactorizar.py
  Intenta otra vez: 6
  Demasiado bajo.
  Intenta otra vez: 9
  Demasiado alto.
  Intenta otra vez: 10
  Demasiado alto.
  Intenta otra vez: 7
  ¡Correcto! El número era 7.
  ~ /workspace$ python Refactorizar.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$
```


07-. Ejercicio - Lista de Hobbies y comidas favoritas.

Descripción:

Este código define dos listas: una con varios hobbies y otra con los hobbies favoritos del usuario. Primero, recorre la lista completa de hobbies e imprime cada uno con un mensaje indicando que son los hobbies. Luego, en la lista de favoritos, imprime individualmente los tres primeros hobbies usando índices para mostrar cuál es el primer, segundo y tercer hobby favorito. Finalmente, vuelve a recorrer la lista de favoritos con un bucle para imprimir cada hobby favorito en una línea separada. Es importante mencionar que usar el mismo nombre para la variable del bucle y la lista puede generar confusión, aunque el código funciona correctamente.

Este código define dos listas: una con varias comidas favoritas y otra con una selección personal de comidas favoritas del usuario. Primero, recorre la lista completa de comidas favoritas e imprime cada una con un mensaje indicando que son las comidas favoritas. Luego, imprime individualmente las tres primeras comidas favoritas de la lista personal utilizando sus índices para especificar el orden. Finalmente, utiliza un bucle para recorrer esta lista personal y mostrar cada comida favorita en una línea separada, añadiendo además una firma con el nombre "Fabrizzio Lora (FaXx0)" después de cada impresión. Aunque el código funciona correctamente, sería recomendable usar nombres distintos para las variables en los bucles para evitar confusión.



The screenshot shows a Replit IDE window titled "prog2-ejer-prac-class". The file explorer on the left shows a project with several files, including "Lista_hobbies.py". The main editor displays the following Python code:

```
14 # Lista_hobbies.py
15 # =====
16 Comidas_favoritas = ["Pizza", "Hamburguesas", "Sushi", "Majadito", "Ensaladas"]
17 for Comida_favorita in Comidas_favoritas:
18     print(f"mis comidas favoritas son: {Comida_favorita}")
19
20 Favoritos = ["Pizza", "Sushi", "Ensaladas"]
21 # print(f"mis comidas favoritas son: {Comidas_favoritas}")
22 # print(f"mis favoritos seleccionados son: {Favoritos}")
23 print("MI PRIMER COMIDA FAVORITA ES: " + Favoritos[0])
24 print("MI SEGUNDA COMIDA FAVORITA ES: " + Favoritos[1])
25 print("MI TERCER COMIDA FAVORITA ES: " + Favoritos[2])
26 for comidas_favoritas in Favoritos:
27     print(f"mis comidas favoritas son :{comidas_favoritas}")
```

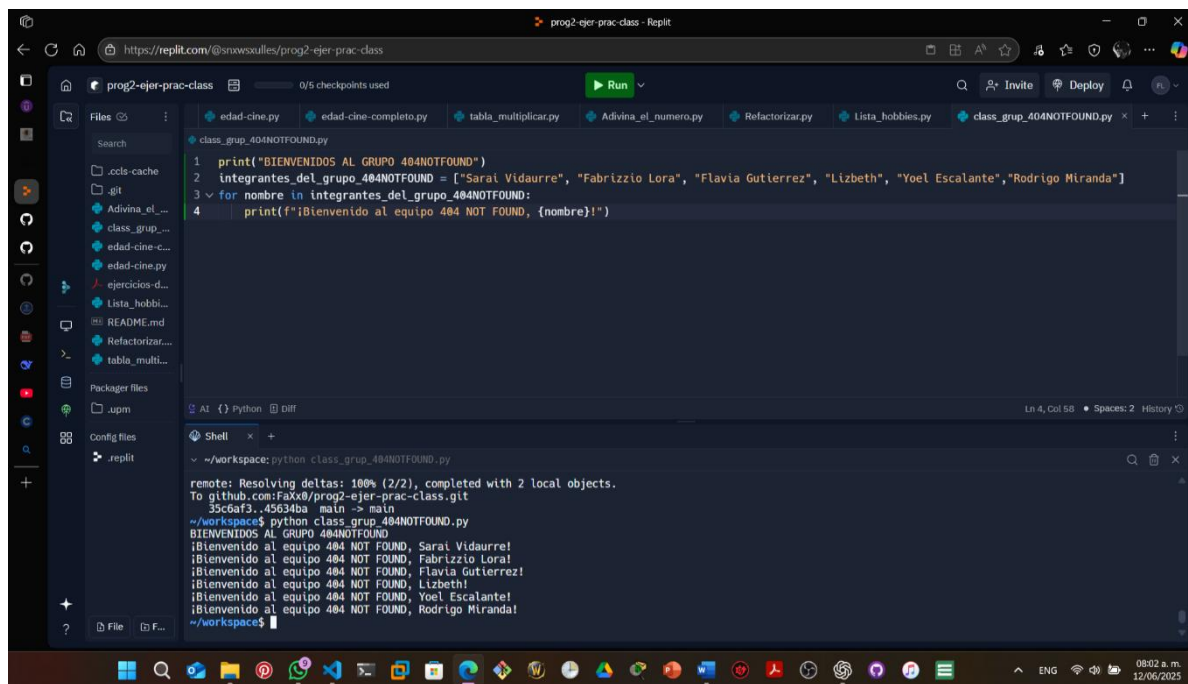
The output window at the bottom shows the following text:

```
MI PRIMER HOBBIE FAVORITO ES: Programar
MI SEGUNDO HOBBIE FAVORITO ES: Leer
MI TERCER HOBBIE FAVORITO ES: Jugar videojuegos
Mis hobbies favoritos son :Programar
Mis hobbies favoritos son :Leer
Mis hobbies favoritos son :Jugar videojuegos
Mis comidas favoritas son: Pizza
Mis comidas favoritas son: Hamburguesas
Mis comidas favoritas son: Sushi
Mis comidas favoritas son: Majadito
Mis comidas favoritas son: Ensaladas
MI PRIMER COMIDA FAVORITA ES: Pizza
```


08-. Ejercicio - Integrantes de mi grupo.

Descripción:

Este programa imprime los nombres de los integrantes del grupo 404NOTFOUND.



The screenshot shows a Replit IDE window titled "prog2-ejer-prac-class - Replit". The browser address bar shows "https://replit.com/@snwswskulles/prog2-ejer-prac-class". The file explorer on the left shows a project named "prog2-ejer-prac-class" with several files, including "class_grup_404NOTFOUND.py". The main editor displays the code for "class_grup_404NOTFOUND.py":

```
1 print("BIENVENIDOS AL GRUPO 404NOTFOUND")
2 integrantes_del_grupo_404NOTFOUND = ["Saraí Vidaurre", "Fabrizzio Lora", "Flavia Gutierrez", "Lizbeth", "Yoel Escalante", "Rodrigo Miranda"]
3 for nombre in integrantes_del_grupo_404NOTFOUND:
4     print(f"¡Bienvenido al equipo 404 NOT FOUND, {nombre}!")
```

The terminal at the bottom shows the execution of the script:

```
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:FaXx0/prog2-ejer-prac-class.git
35c6af3..45634ba main -> main
~/workspace$ python class_grup_404NOTFOUND.py
BIENVENIDOS AL GRUPO 404NOTFOUND
¡Bienvenido al equipo 404 NOT FOUND, Saraí Vidaurre!
¡Bienvenido al equipo 404 NOT FOUND, Fabrizzio Lora!
¡Bienvenido al equipo 404 NOT FOUND, Flavia Gutierrez!
¡Bienvenido al equipo 404 NOT FOUND, Lizbeth!
¡Bienvenido al equipo 404 NOT FOUND, Yoel Escalante!
¡Bienvenido al equipo 404 NOT FOUND, Rodrigo Miranda!
~/workspace$
```

The system clock in the bottom right corner indicates the time is 08:02 a.m. on 12/06/2025.

09-. Ejercicio - Promedio.

Descripción:

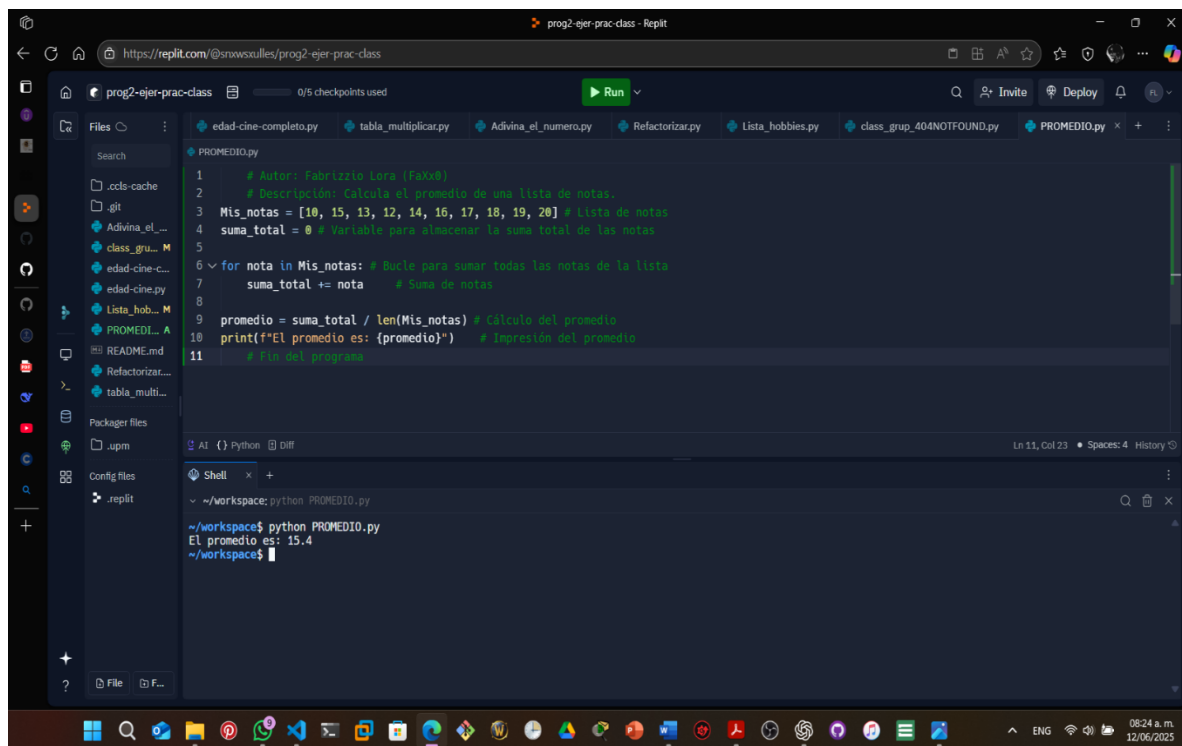
Este código crea una lista llamada `Mis notas` que contiene varias calificaciones.

Luego, utiliza un bucle para recorrer cada nota en la lista y sumar todas ellas en la variable

`suma total`.

Después, calcula el promedio dividiendo la suma total entre la cantidad de notas que hay en la lista.

Finalmente, imprime el resultado del promedio junto con una firma que dice "Fabrizzio Lora (FaXx0)". El programa termina después de mostrar esta información.



The screenshot shows a Replit IDE window titled "prog2-ejer-prac-class - Replit". The browser address bar shows "https://replit.com/@srowsulles/prog2-ejer-prac-class". The file explorer on the left shows a project named "prog2-ejer-prac-class" with several files, including "PROMEDIO.py". The main editor displays the code for "PROMEDIO.py":

```
1 # Autor: Fabrizzio Lora (FaXx0)
2 # Descripción: Calcula el promedio de una lista de notas.
3 Mis_notas = [10, 15, 13, 12, 14, 16, 17, 18, 19, 20] # Lista de notas
4 suma_total = 0 # Variable para almacenar la suma total de las notas
5
6 for nota in Mis_notas: # Bucle para sumar todas las notas de la lista
7     suma_total += nota # Suma de notas
8
9 promedio = suma_total / len(Mis_notas) # Cálculo del promedio
10 print(f"El promedio es: {promedio}") # Impresión del promedio
11 # Fin del programa
```

Below the code editor, a terminal window shows the execution of the script:

```
~/workspace$ python PROMEDIO.py
El promedio es: 15.4
~/workspace$
```

The bottom of the image shows a Windows taskbar with various application icons and a system clock indicating 08:24 a.m. on 12/06/2025.

10-. Ejercicio - Operaciones con listas.

Descripción:

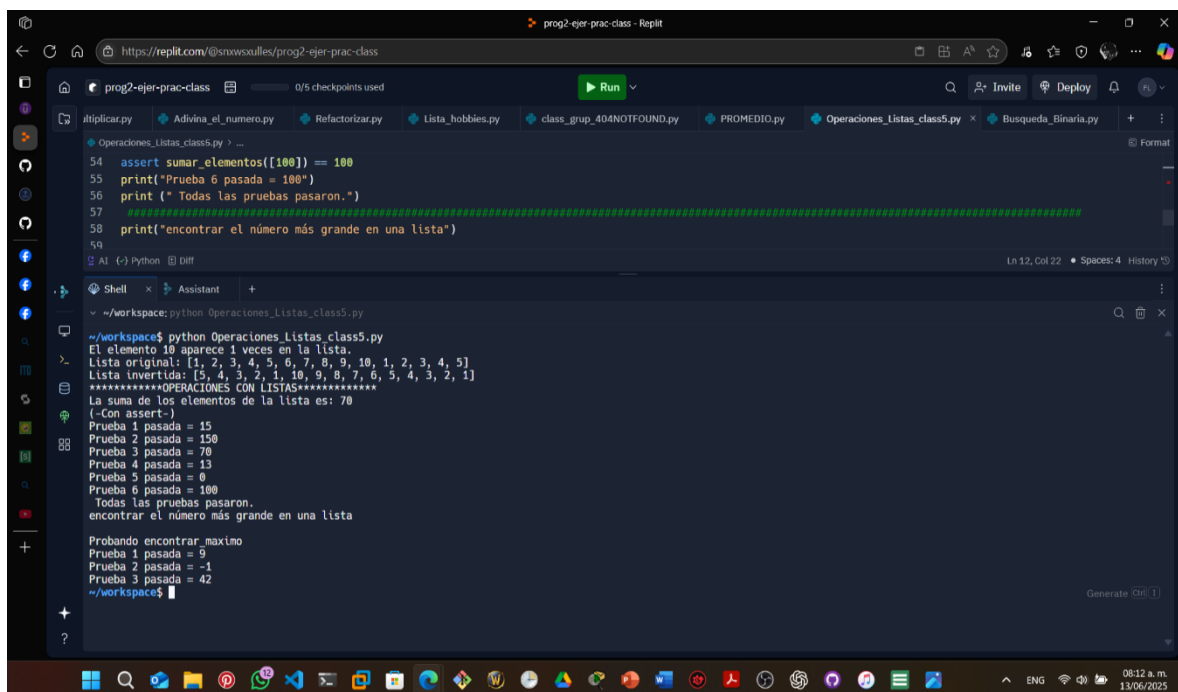
Este código contiene varias funciones y operaciones relacionadas con listas en Python, todas orientadas a realizar tareas comunes como contar elementos, invertir listas, sumar valores y encontrar el máximo.

Primero, define una función para contar cuántas veces aparece un elemento específico en una lista, luego invierte una lista original y muestra ambas versiones.

Después, hay una función que suma todos los elementos de una lista y su resultado se imprime, incluyendo una firma.

Además, utiliza pruebas con `assert` para validar que la función de suma funciona correctamente en diferentes casos, mostrando mensajes cuando las pruebas son exitosas.

Finalmente, define una función para encontrar el número más grande en una lista, con comprobaciones para listas vacías, y también verifica su correcto funcionamiento con pruebas. Todo el código está diseñado para ser claro y didáctico, mostrando paso a paso cómo trabajar con listas en Python.



```
54 assert sumar_elementos([100]) == 100
55 print("Prueba 6 pasada = 100")
56 print (" Todas las pruebas pasaron.")
57
58 print("encontrar el número más grande en una lista")
59
```

Ln 12, Col 22 • Spaces: 4 History ↻

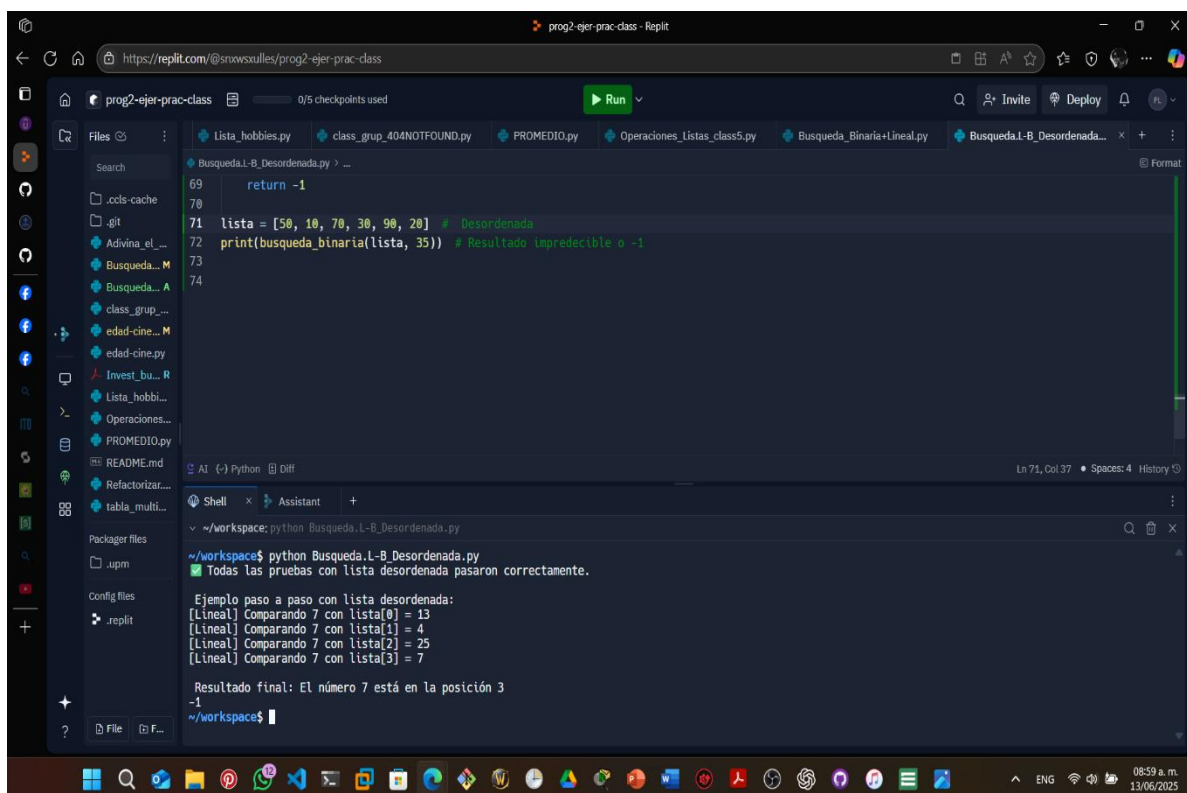
```
~/workspace: python Operaciones_Listas_class5.py
El elemento 10 aparece 1 veces en la lista.
Lista original: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5]
Lista invertida: [5, 4, 3, 2, 1, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
*****OPERACIONES CON LISTAS*****
La suma de los elementos de la lista es: 70
(-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$
```

11-. Ejercicio - Búsqueda binaria y lineal desordenada

Descripción:

Este código implementa dos algoritmos clásicos de búsqueda en listas: la búsqueda lineal y la búsqueda binaria. La función de búsqueda lineal recorre cada elemento de la lista uno por uno hasta encontrar el objetivo o llegar al final, y tiene la opción de mostrar cada comparación que realiza para mayor claridad. Además, incluye pruebas automáticas para verificar que la búsqueda lineal funcione correctamente con una lista desordenada, asegurando que encuentra los elementos cuando están y devuelve -1 cuando no. Por otro lado, la función de búsqueda binaria busca un elemento en una lista ordenada dividiendo repetidamente la lista en mitades para reducir el rango de búsqueda, aunque en el ejemplo se prueba con una lista desordenada, por lo que el resultado puede ser incorrecto o -1. El código finaliza imprimiendo el resultado de la búsqueda binaria y mostrando una firma. En conjunto, el código sirve para comparar y entender cómo funcionan estas dos técnicas de búsqueda.



```
69     return -1
70
71 lista = [50, 10, 70, 30, 90, 20] # Desordenado
72 print(búsqueda_binaria(lista, 35)) # Resultado impredecible o -1
73
74
```

```
~/workspace$ python Busqueda.L-B_Desordenada.py
Todas las pruebas con lista desordenada pasaron correctamente.

Ejemplo paso a paso con lista desordenada:
[Lineal] Comparando 7 con lista[0] = 13
[Lineal] Comparando 7 con lista[1] = 4
[Lineal] Comparando 7 con lista[2] = 25
[Lineal] Comparando 7 con lista[3] = 7

Resultado final: El número 7 está en la posición 3
-1
~/workspace$
```