

UNIVERSIDAD PRIVADA DOMINGO SAVIO

FACULTAD DE INGENIERIA



Actividad

Título: Ejercicios de Replit

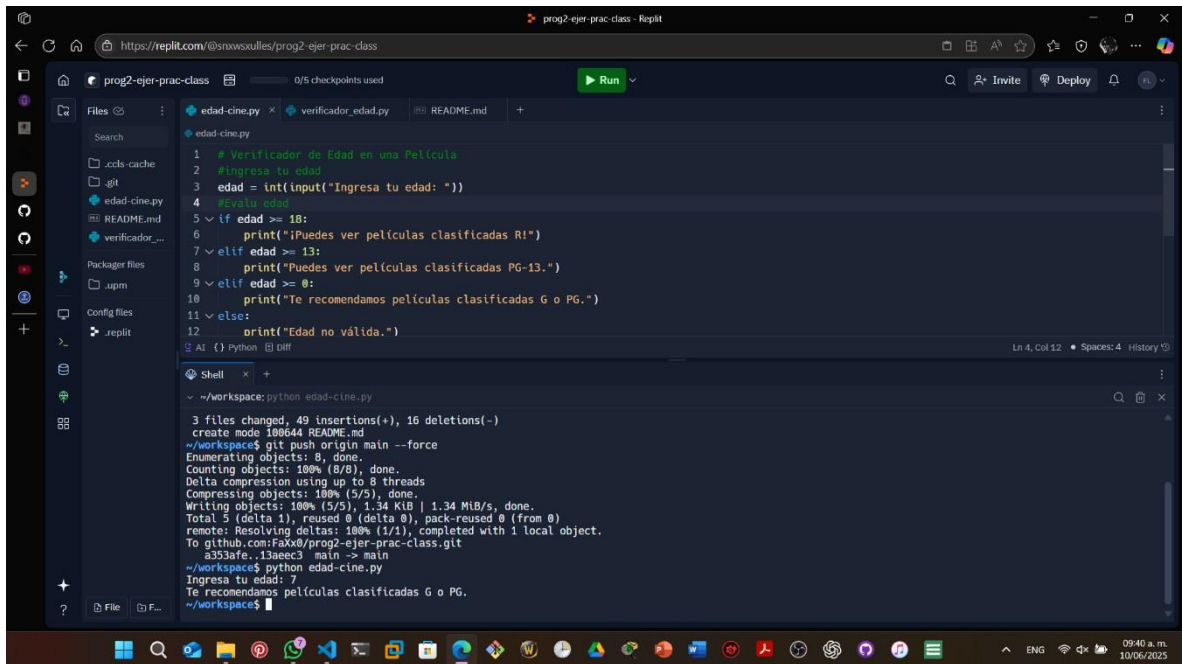
Docente: Ing.Jimmy Nataniel Requena Llorentty

Materia: Programación 2

Estudiante: LORA COLODRO FABRIZZIO

Junio del 2025
Santa Cruz – Bolivia

01-Ejercicio simple-edad-cine



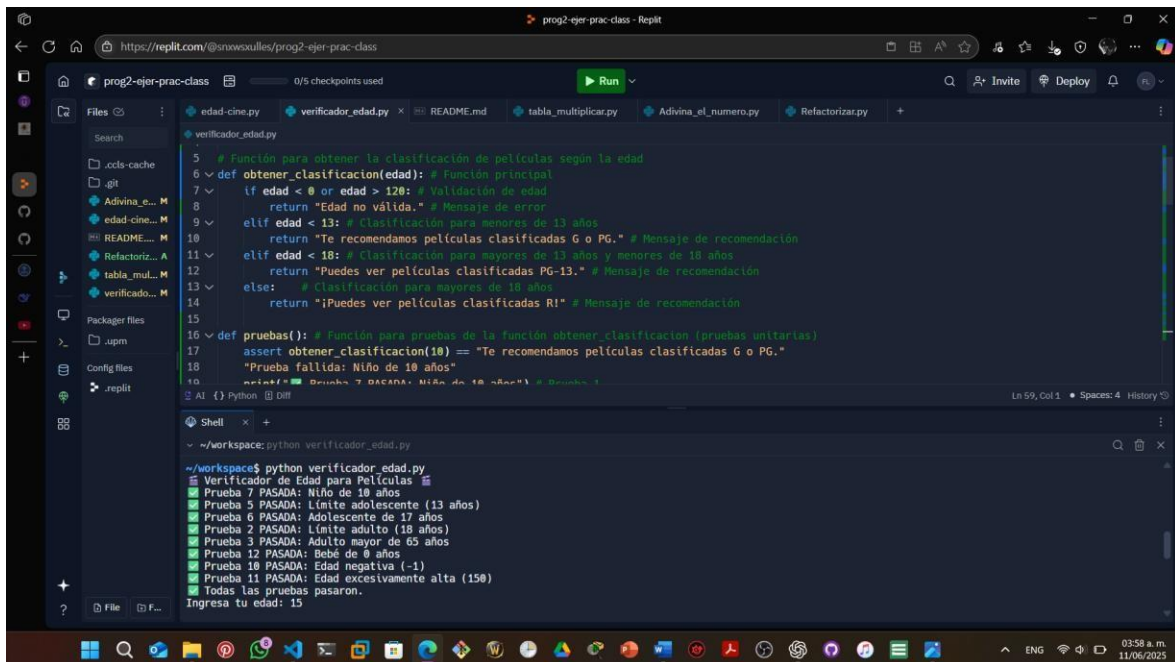
The screenshot shows a Replit workspace for a project named "prog2-ejer-prac-class". The file explorer on the left lists files: "edad-cine.py", "verificador_edad.py", and "README.md". The main editor displays the content of "edad-cine.py", which is a Python script for verifying age and recommending movies. The script prompts the user to enter their age and then checks if they are eligible to watch certain movies based on age ranges.

```
1 # Verificador de Edad en una Película
2 # Ingrese tu edad
3 edad = int(input("Ingrese tu edad: "))
4 # Evalúe 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 >= 0:
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 prompts the user to enter their age, and the user enters 7. The script then prints "Te recomendamos películas clasificadas G o PG.".

```
~/workspace: python edad-cine.py
3 files changed, 49 insertions(+), 16 deletions(-)
create mode 100644 README.md
~/workspace$ git push origin main --force
Enumerating objects: 0, done.
Counting objects: 100% (0/0), 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..13aeeec3 main -> main
~/workspace$ python edad-cine.py
Ingrese tu edad: 7
Te recomendamos películas clasificadas G o PG.
~/workspace$
```

02-Ejer2-edad-cine- completo



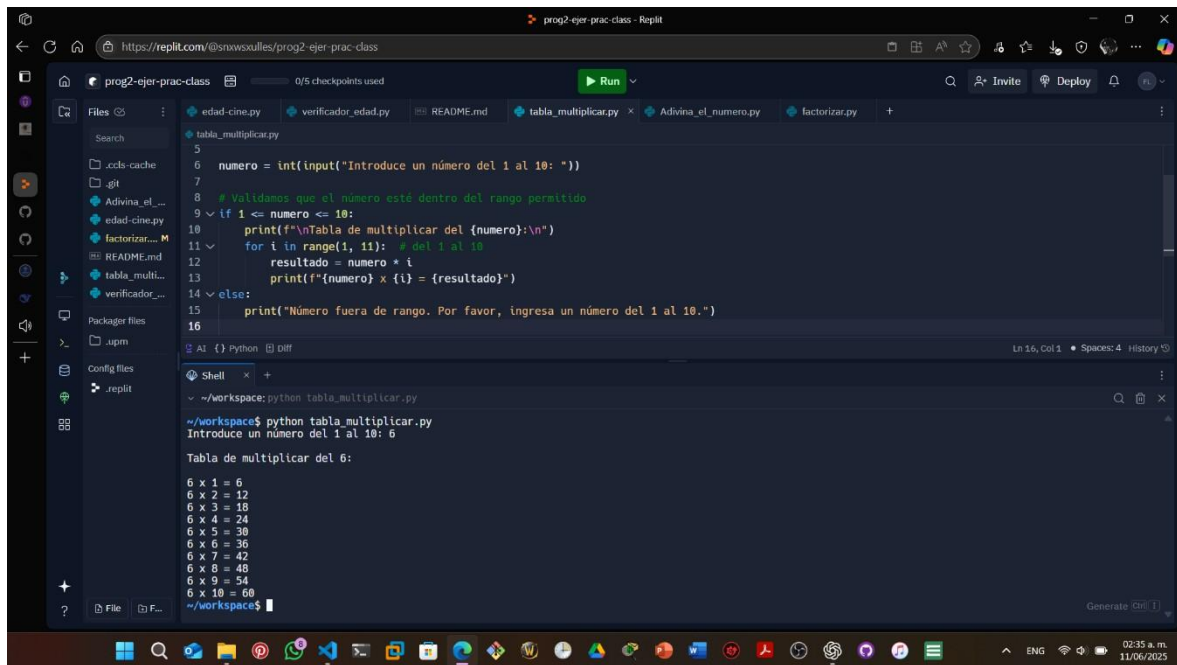
The screenshot shows a Replit workspace for a project named "prog2-ejer-prac-class". The file explorer on the left lists files: "edad-cine.py", "verificador_edad.py", "tabla_multiplicar.py", "Adivina_el_numero.py", and "Refactorizar.py". The main editor displays the content of "verificador_edad.py", which is a more complete Python script for verifying age and recommending movies. The script includes a function to get the classification of movies based on age, a function to test the classification function, and a main function that prompts the user to enter their age and then checks if they are eligible to watch certain movies based on age ranges.

```
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     print("Prueba 7 PASADA: Niño de 10 años")
20
21 # Pruebas de la función obtener_clasificacion (pruebas unitarias)
22 pruebas()
23
24 # Verificador de Edad para Películas
25 # Ingrese tu edad
26 edad = int(input("Ingrese tu edad: "))
27 # Evalúe edad
28 if edad >= 18:
29     print("¡Puedes ver películas clasificadas R!")
30 elif edad >= 13:
31     print("Puedes ver películas clasificadas PG-13.")
32 elif edad >= 0:
33     print("Te recomendamos películas clasificadas G o PG.")
34 else:
35     print("Edad no válida.")
```

The terminal output shows the execution of the script. It prompts the user to enter their age, and the user enters 15. The script then prints "Te recomendamos películas clasificadas G o PG.".

```
~/workspace: python verificador_edad.py
Verificador de Edad para Películas
Prueba 7 PASADA: Niño de 10 años
Prueba 8 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 0 años
Prueba 10 PASADA: Edad negativa (-1)
Prueba 11 PASADA: Edad excesivamente alta (150)
Todas las pruebas pasaron.
Ingrese tu edad: 15
Te recomendamos películas clasificadas G o PG.
```

03-Ejer3-Tablas-multiplicar



```
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
```

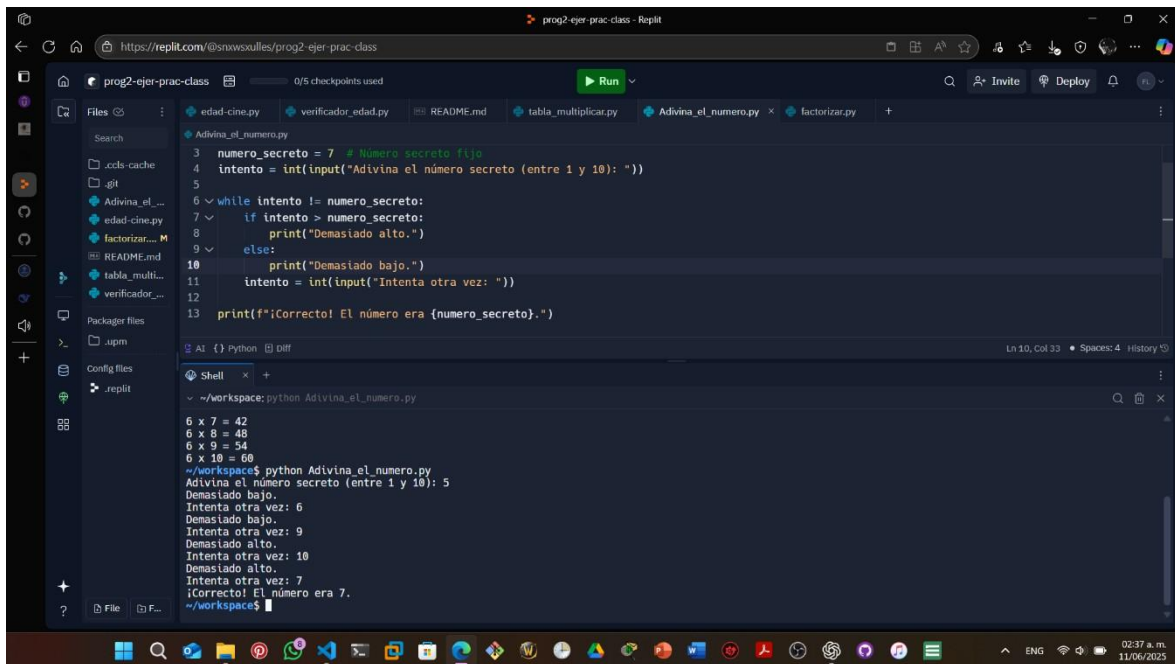
Ln 16, Col 1 • Spaces: 4 History

```
~/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-Ejer4-Adivina el numero



```
3 numero_secreto = 7 # Número secreto fijo
4 intento = int(input("Adivina el número secreto (entre 1 y 10): "))
5
6 while intento != numero_secreto:
7     if intento > numero_secreto:
8         print("Demasiado alto.")
9     else:
10        print("Demasiado bajo.")
11        intento = int(input("Intenta otra vez: "))
12
13 print(f"¡Correcto! El número era {numero_secreto}.")
```

Ln 10, Col 33 • Spaces: 4 History

```
~/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-Ejer-README.md

```
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
```

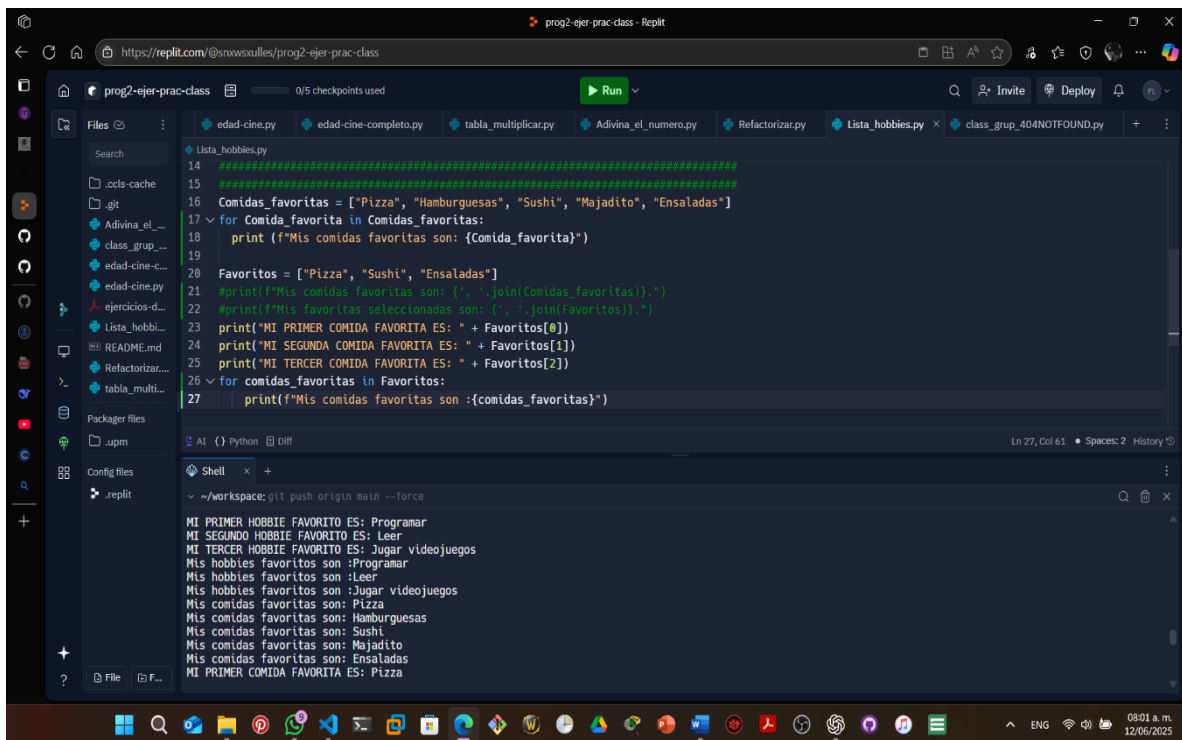
```
~/workspace$ python Refactorizar.py
Intenta otra vez: 6
Demasiado bajo.
Intenta otra vez: 9
Demasiado alto.
Intenta otra vez: 10
Demasiado alto.
Intenta otra vez: 7
¡Correcto! El número era 7.
~/workspace$ python Refactorizar.py
El área del rectángulo 1 (10x5) es: 50
El área del rectángulo 2 (7x3) es: 21
El área del rectángulo 3 (15x8) es: 120
~/workspace$
```

06-Ejer-Refactorizar

```
0 # 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 * 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
```

```
~/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-lista de Hobbies y comidas favoritas



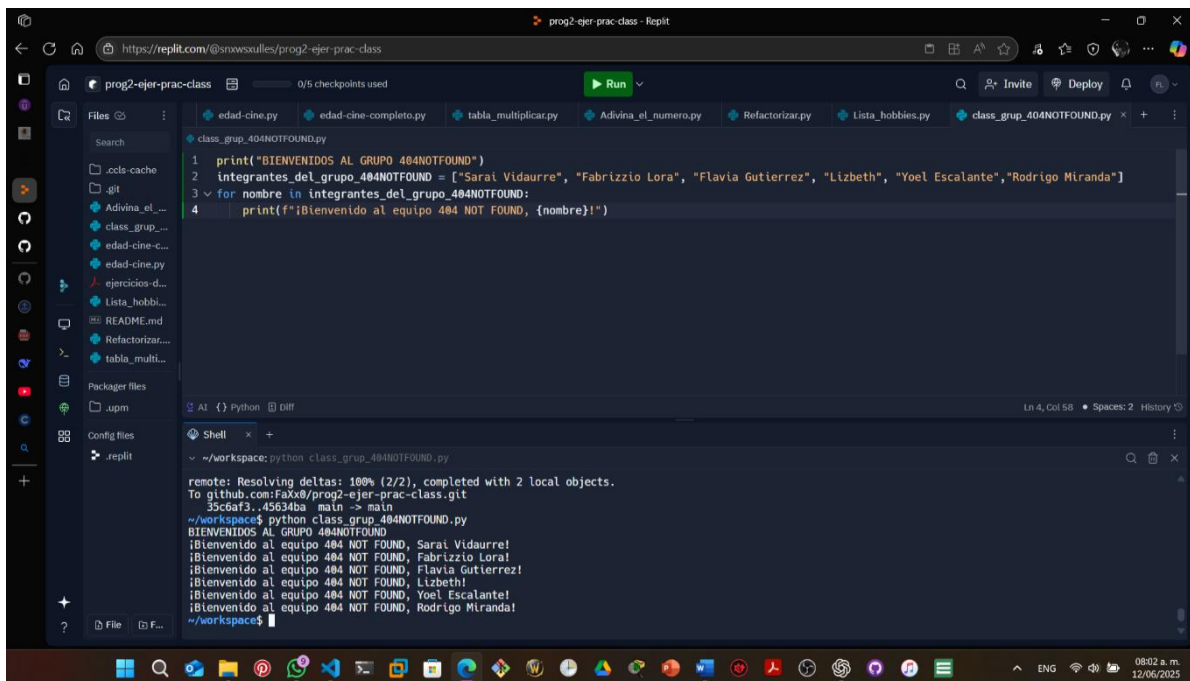
The screenshot shows a Replit environment with a Python script named `Lista_hobbies.py`. The script defines two lists: `Comidas_favoritas` and `Favoritos`. It then prints out the first three items from each list. The output in the terminal shows the first three hobbies and the first three favorite foods.

```
14 # Lista_hobbies.py
15
16 Comidas_favoritas = ["Pizza", "Hamburguesas", "Sushi", "Majadito", "Ensaladas"]
17 for Comida_favorita in Comidas_favoritas:
18     print(f"Mi comida favorita son: {Comida_favorita}")
19
20 Favoritos = ["Pizza", "Sushi", "Ensaladas"]
21 #print(f"Mi comida favorita son: {' '.join(Comidas_favoritas)}")
22 #print(f"Mi favoritos seleccionados son: {' '.join(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"Mi comida favorita son :{comidas_favoritas}")
```

Output in terminal:

```
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-INTEGRANTES DE MI GRUPO



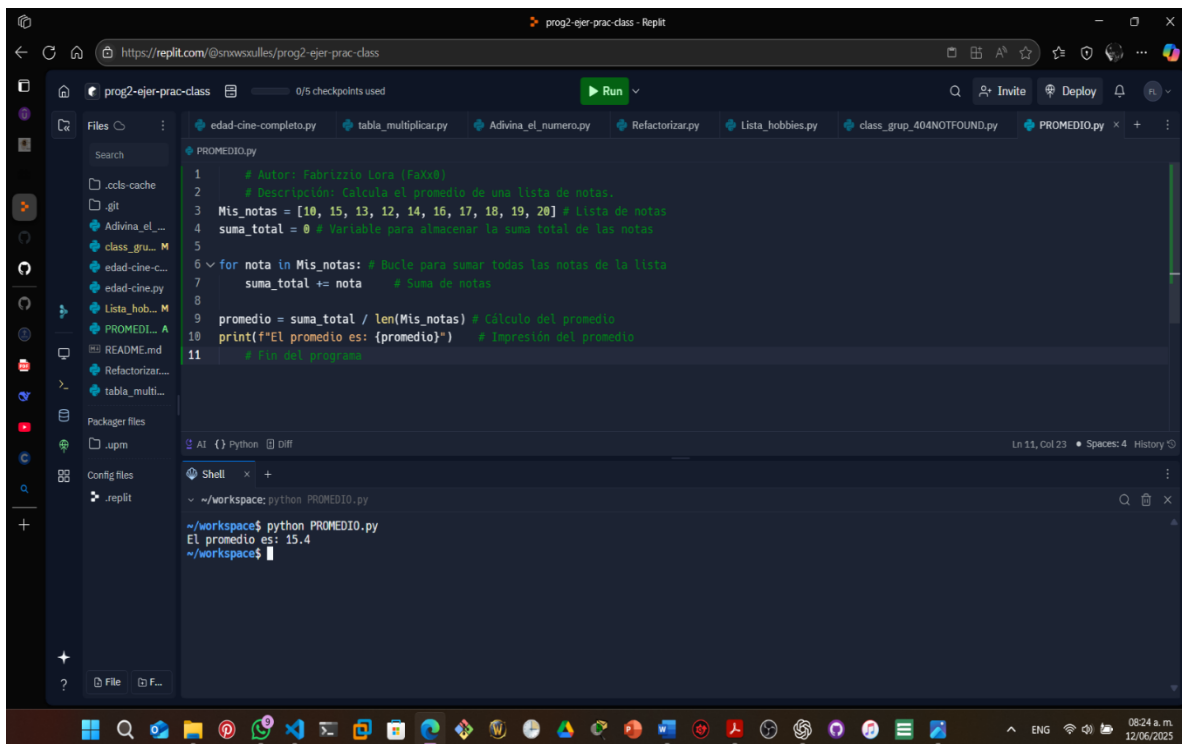
The screenshot shows a Replit environment with a Python script named `class_grup_404NOTFOUND.py`. The script defines a list of group members and prints out each name. The output in the terminal shows the list of group members.

```
1 print("BIENVENIDOS AL GRUPO 404NOTFOUND")
2 integrantes_del_grupo_404NOTFOUND = ["Sara 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}!")
```

Output in terminal:

```
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:snxwskulles/prog2-ejer-prac-class
35c6af3..45634ba main -> main
~/workspace$ python class_grup_404NOTFOUND.py
BIENVENIDOS AL GRUPO 404NOTFOUND
¡Bienvenido al equipo 404 NOT FOUND, Sara 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$
```


09-PROMEDIO

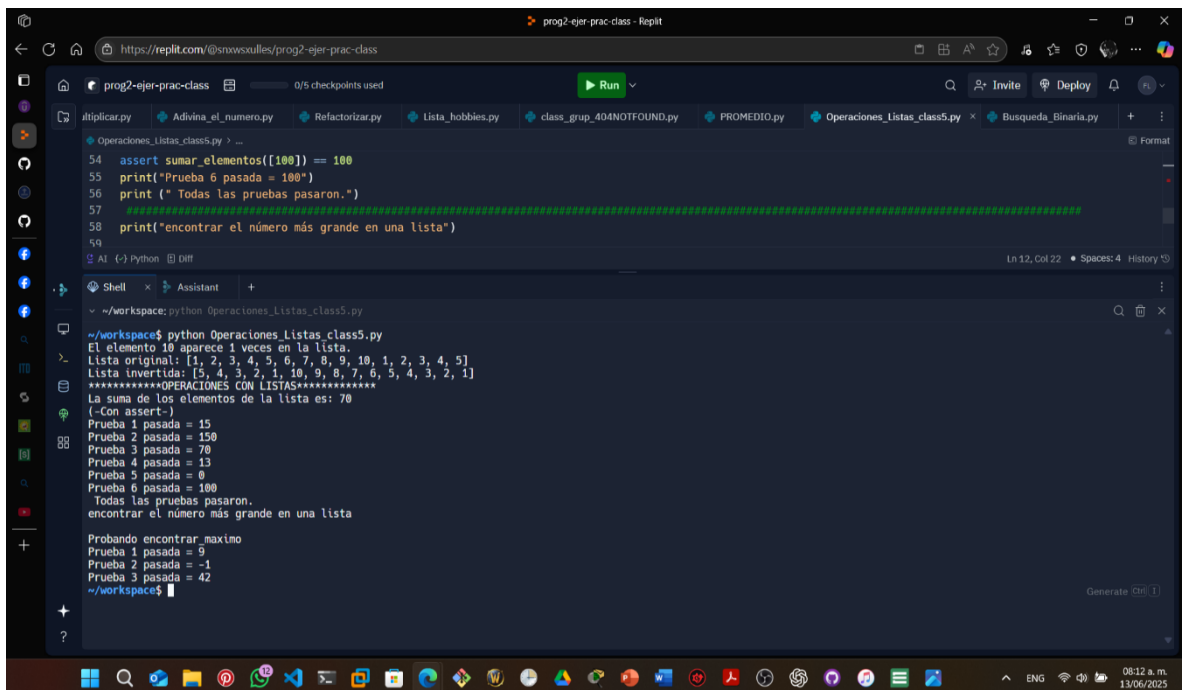


The screenshot shows a Replit IDE window titled "prog2-ejer-prac-class". The file explorer on the left lists several files, including "PROMEDIO.py". The main editor displays the following Python code:

```
1 # Autor: Fabrizio Iora (Faxx8)
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
```

The terminal at the bottom shows the command `python PROMEDIO.py` being executed, resulting in the output: `El promedio es: 15.4`.

10-OPERACIONES CON LISTAS



The screenshot shows a Replit IDE window titled "prog2-ejer-prac-class". The file explorer on the left lists several files, including "Operaciones_Listas_class5.py". The main editor displays the following Python code:

```
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
```

The terminal at the bottom shows the command `python Operaciones_Listas_class5.py` being executed, resulting in the following output:

```
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- Ejer-bisqueda binaria y lineal desordenada

```
69     return -1
70
71 lista = [50, 10, 70, 30, 90, 20] # Desordenada
72 print(busqueda_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$
```