iLERNA.

Profesora: Geraldin López

Asignatura: MÓDULO OPTATIVO

Actividad práctica: Conociéndome con Python

Objetivo

Aplicar los conocimientos básicos de programación en Python mediante la creación de un programa interactivo que combine:

- Variables y tipos de datos
- Listas, tuplas y conjuntos
- Funciones básicas (print(), len(), type(), input())
- Razonamiento lógico a través de condiciones simples

Instrucciones

1. Abre **Visual Studio Code (VS Code)**.
2. Crea un archivo nuevo con el nombre: **actividad_conociendome.py**
3. Escribe un programa que cumpla con todas las partes que se detallan a continuación.
4. Guarda tu archivo y ejecuta cada parte para observar los resultados.

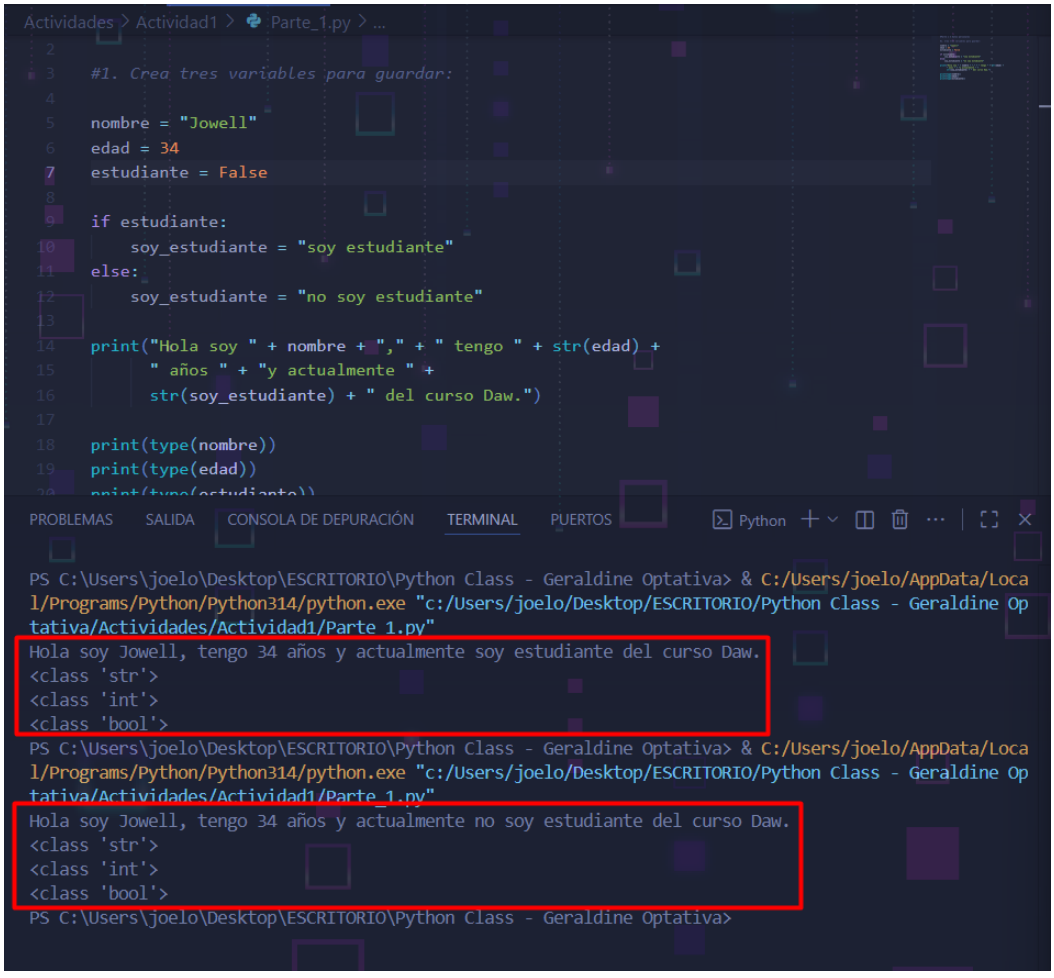
Parte 1 – Datos personales

1. Crea tres variables para guardar:

- Tu **nombre completo** (tipo str)
- Tu **edad** (tipo int)
- Si **eres estudiante o no** (tipo bool)

2. Muestra la información en pantalla con frases personalizadas.
3. Usa la función type() para mostrar el tipo de dato de cada variable.
4. Cambia el valor de una de las variables y muestra nuevamente el resultado.

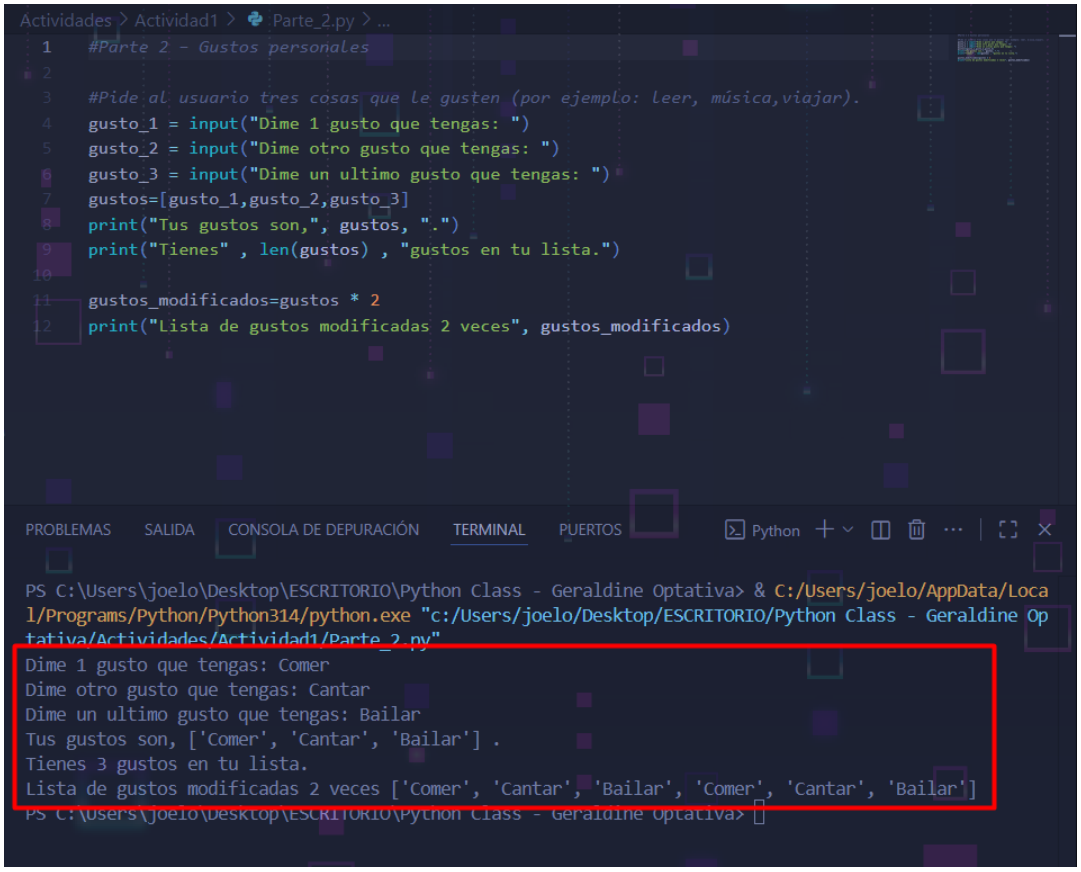
(El primer Run ejecuta TRUE y el segundo FALSE).



Parte 2 – Gustos personales

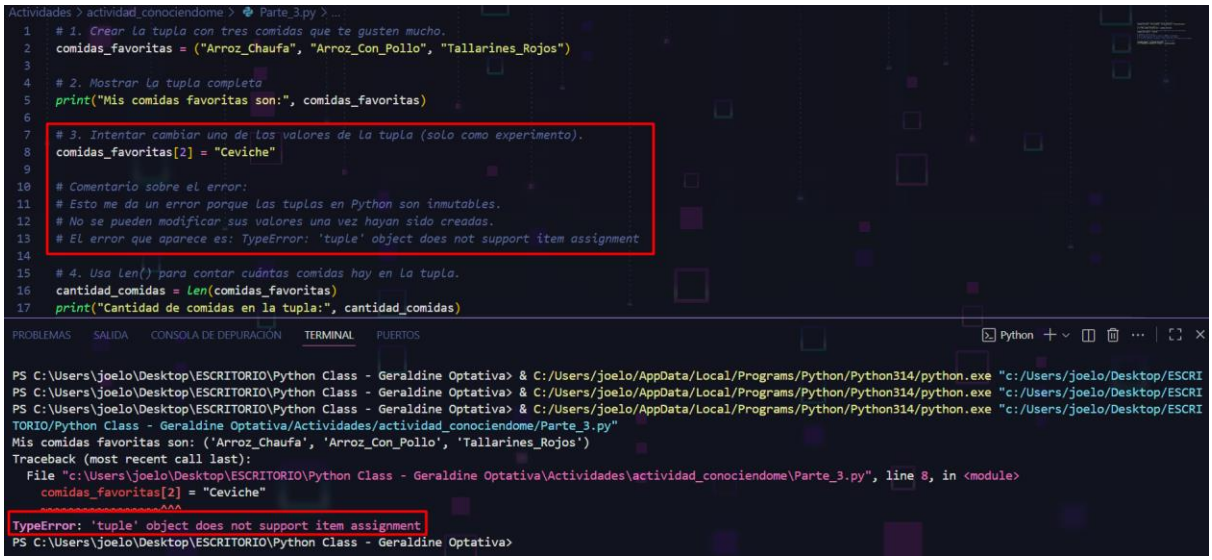
1. Pide al usuario tres cosas que le gusten (por ejemplo: leer, música, viajar).
2. Guarda esas respuestas en una **lista** llamada gustos.
3. Muestra la lista completa en pantalla.
4. Usa la función len() para mostrar cuántos elementos tiene la lista.
5. Crea una nueva lista llamada gustos_modificados que repita los gustos dos veces (usando gustos * 2).
6. Muestra el resultado y explica en un comentario qué observas.

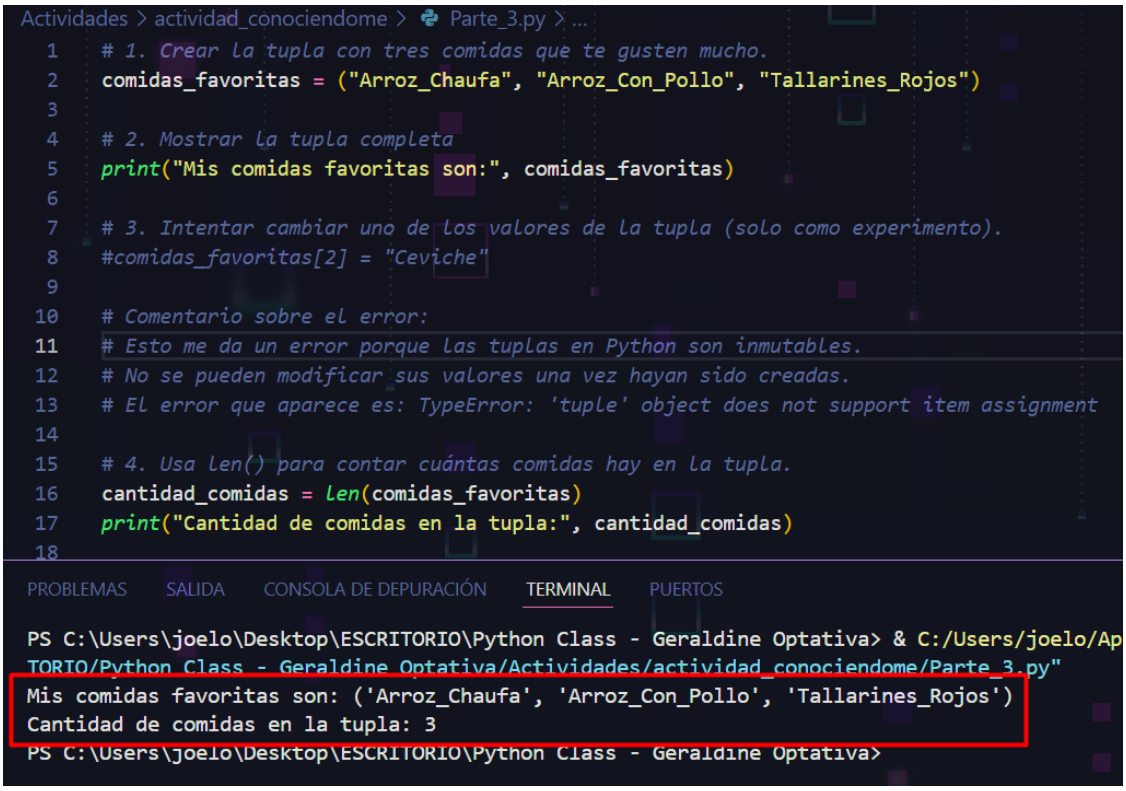
(La lista de gustos_modificados contiene los mismos elementos 2 veces en el mismo orden. Esto se debe a que al multiplicar la lista por 2, se concatenan sus elementos.)



Parte 3 – Comidas favoritas

1. Crea una **tupla** llamada comidas_favoritas con tres comidas que te gusten mucho.
2. Muestra la tupla completa.
3. Intenta cambiar uno de los valores de la tupla (solo como experimento).
   - Escribe en un comentario qué sucede y por qué.
4. Usa len() para contar cuántas comidas hay en la tupla.

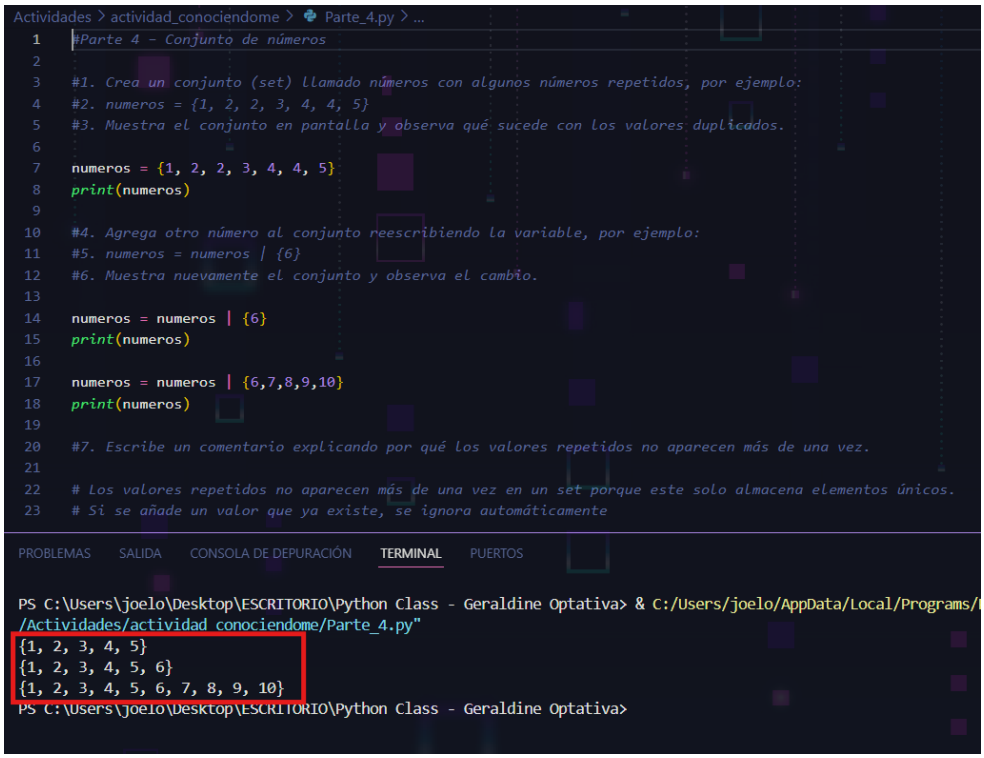




Parte 4 – Conjunto de números

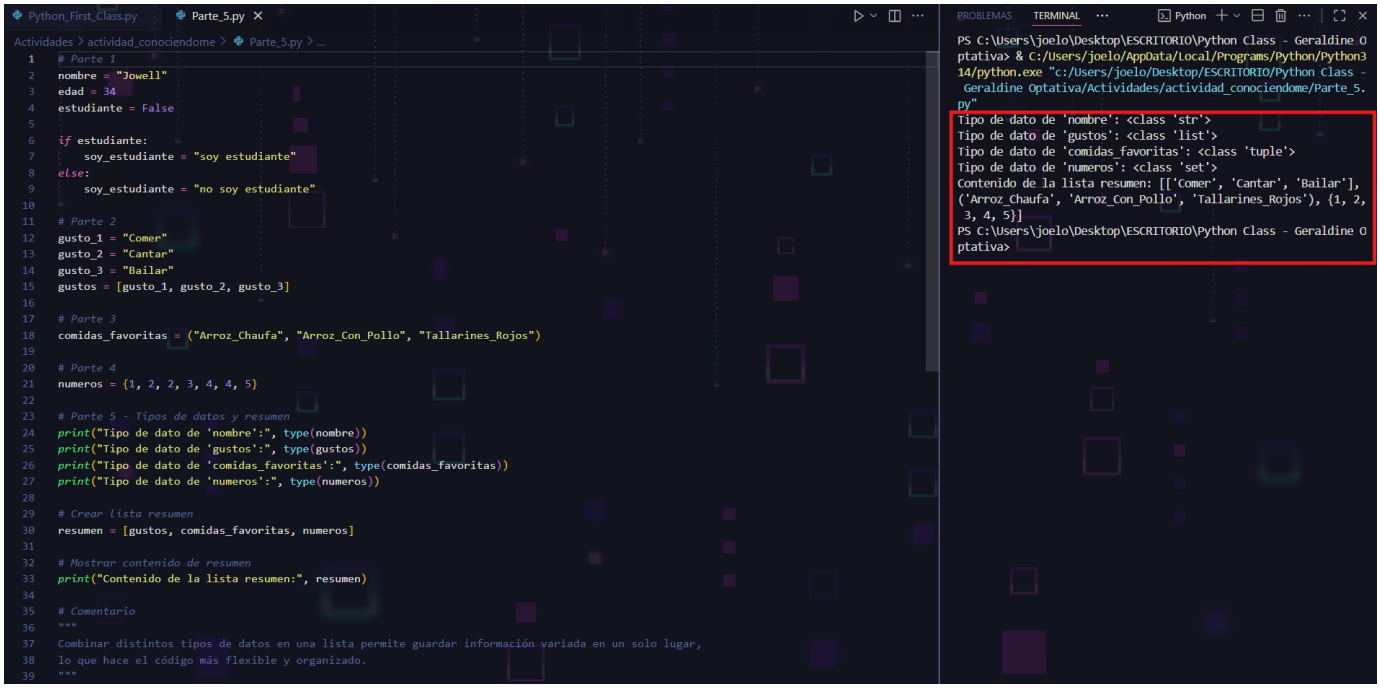
1. Crea un **conjunto (set)** llamado números con algunos números repetidos, por ejemplo:
2. numeros = {1, 2, 2, 3, 4, 4, 5}
3. Muestra el conjunto en pantalla y observa qué sucede con los valores duplicados.
4. Agrega otro número al conjunto reescribiendo la variable, por ejemplo:
5. numeros = numeros | {6}
6. Muestra nuevamente el conjunto y observa el cambio.
7. Escribe un comentario explicando por qué los valores repetidos no aparecen más de una vez.

(Los números duplicados son ignorados, ya que cuando se agregan elementos repetidos, el conjunto SET los elimina automáticamente, manteniendo solo un valor único.)

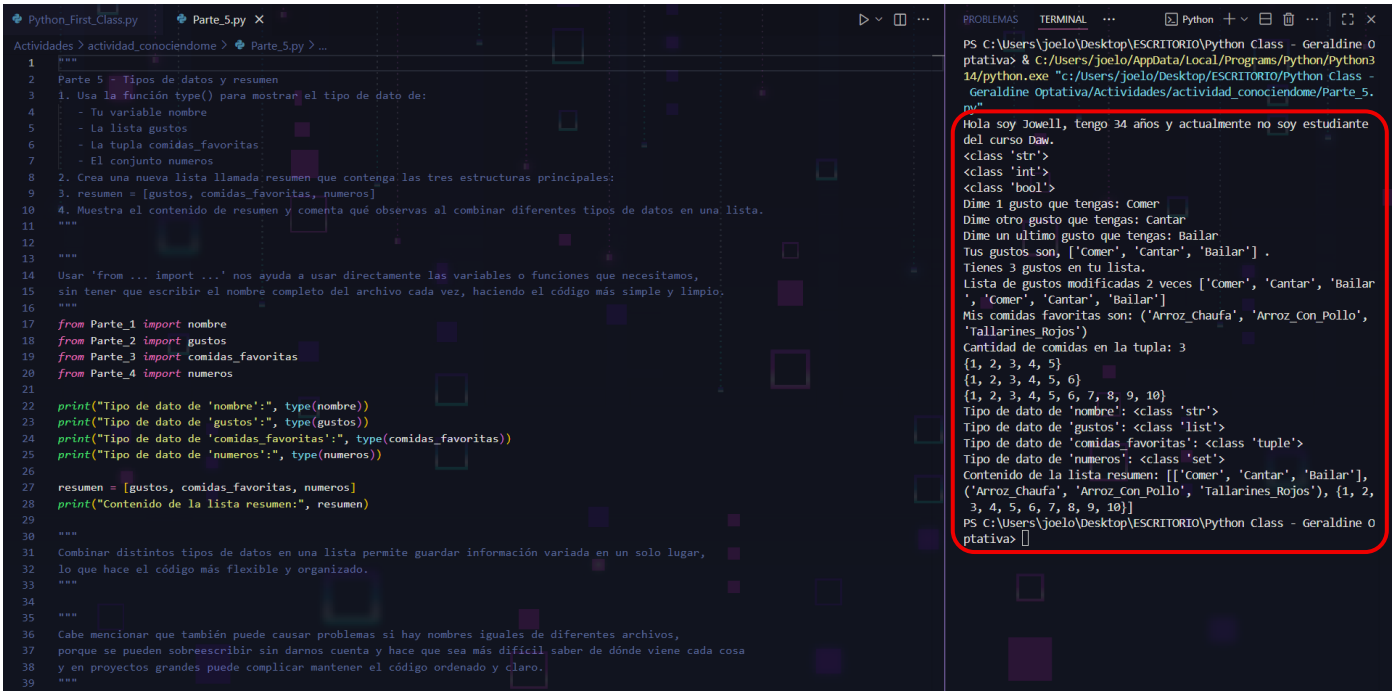


Parte 5 – Tipos de datos y resumen

1. Usa la función type() para mostrar el tipo de dato de:
   - Tu variable nombre
   - La lista gustos
   - La tupla comidas_favoritas
   - El conjunto numeros
2. Crea una **nueva lista** llamada resumen que contenga las tres estructuras principales:
3. resumen = [gustos, comidas_favoritas, numeros]
4. Muestra el contenido de resumen y comenta qué observas al combinar diferentes tipos de datos en una lista.



Otra Forma que mire de hacer el ejercicio, es importando las variables desde los archivos anteriores usando el **from archivo import variable - funcion – clase** lo que fuere y pues esto imprime en la terminal los datos a mayor detalle.



```
"""
Parte 5 - Tipos de datos y resumen
1. Usa la función type() para mostrar el tipo de dato de:
   - Tu variable nombre
   - La lista gustos
   - La tupla comidas_favoritas
   - El conjunto numeros
2. Crea una nueva lista llamada resumen que contenga las tres estructuras principales:
3. resumen = [gustos, comidas_favoritas, numeros]
4. Muestra el contenido de resumen y comenta qué observas al combinar diferentes tipos de datos en una lista.
"""

"""
Usar 'from ... import ...' nos ayuda a usar directamente las variables o funciones que necesitamos,
sin tener que escribir el nombre completo del archivo cada vez, haciendo el código más simple y limpio.
"""
from Parte_1 import nombre
from Parte_2 import gustos
from Parte_3 import comidas_favoritas
from Parte_4 import numeros

print("Tipo de dato de 'nombre':", type(nombre))
print("Tipo de dato de 'gustos':", type(gustos))
print("Tipo de dato de 'comidas_favoritas':", type(comidas_favoritas))
print("Tipo de dato de 'numeros':", type(numeros))

resumen = [gustos, comidas_favoritas, numeros]
print("Contenido de la lista resumen:", resumen)

"""
Combinar distintos tipos de datos en una lista permite guardar información variada en un solo lugar,
lo que hace el código más flexible y organizado.
"""

"""
Cabe mencionar que también puede causar problemas si hay nombres iguales de diferentes archivos,
porque se pueden sobreescribir sin darnos cuenta y hace que sea más difícil saber de dónde viene cada cosa
y en proyectos grandes puede complicar mantener el código ordenado y claro.
"""
```

```
PS C:\Users\joelo\Desktop\ESCRITORIO\Python Class - Geraldine Optativa> & C:/Users/joelo/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/joelo/Desktop/ESCRITORIO/Python Class - Geraldine Optativa/Actividades/actividad_conociendome/Parte_5.py"
Hola soy Jowell, tengo 34 años y actualmente no soy estudiante del curso Daw.
<class 'str'>
<class 'int'>
<class 'bool'>
Dime 1 gusto que tengas: Comer
Dime otro gusto que tengas: Cantar
Dime un ultimo gusto que tengas: Bailar
Tus gustos son, ['Comer', 'Cantar', 'Bailar'] .
Tienes 3 gustos en tu lista.
Lista de gustos modificadas 2 veces ['Comer', 'Cantar', 'Bailar', 'Comer', 'Cantar', 'Bailar']
Mis comidas favoritas son: ('Arroz_Chaufa', 'Arroz_Con_Pollo', 'Tallarines_Rojos')
Cantidad de comidas en la tupla: 3
{1, 2, 3, 4, 5}
{1, 2, 3, 4, 5, 6}
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
Tipo de dato de 'nombre': <class 'str'>
Tipo de dato de 'gustos': <class 'list'>
Tipo de dato de 'comidas_favoritas': <class 'tuple'>
Tipo de dato de 'numeros': <class 'set'>
Contenido de la lista resumen: [['Comer', 'Cantar', 'Bailar'], ('Arroz_Chaufa', 'Arroz_Con_Pollo', 'Tallarines_Rojos'), {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}]
PS C:\Users\joelo\Desktop\ESCRITORIO\Python Class - Geraldine Optativa>
```

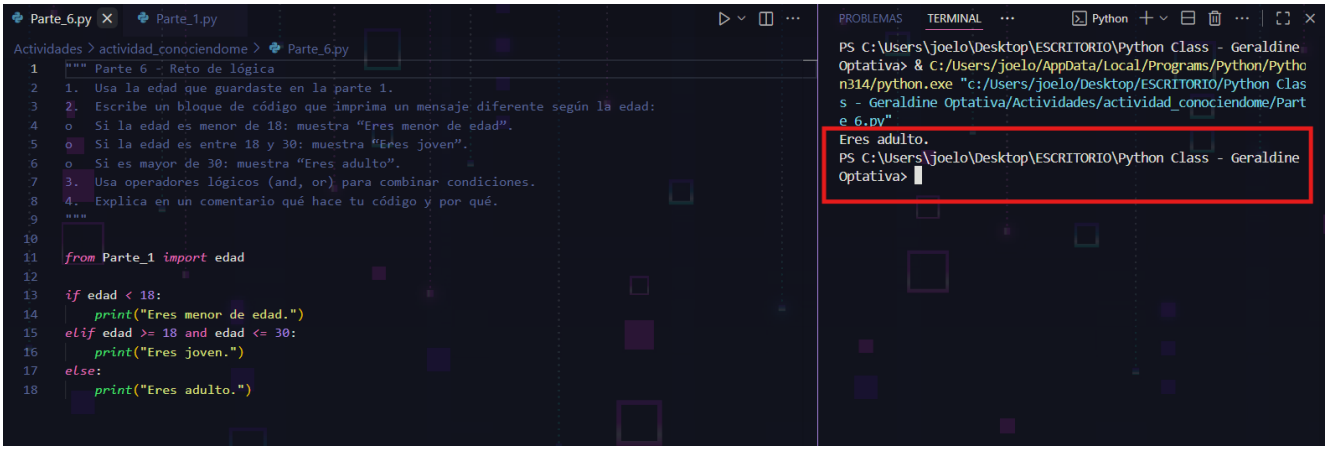
Parte 6 – Reto de lógica

1. Usa la edad que guardaste en la parte 1.
2. Escribe un bloque de código que imprima un mensaje diferente según la edad:
   - Si la edad es menor de 18: muestra “Eres menor de edad”.
   - Si la edad es entre 18 y 30: muestra “Eres joven”.
   - Si es mayor de 30: muestra “Eres adulto”.
3. Usa operadores lógicos (and, or) para combinar condiciones.
4. Explica en un comentario qué hace tu código y por qué.

Pista: Recuerda que puedes usar la estructura if, elif y else.

Ejemplo :

```
if edad < 18:
    print("Eres menor de edad.")
elif edad >= 18 and edad <= 30:
    print("Eres joven.")
else:
    print("Eres adulto.")
```



Parte 7 – Reflexiona

Responde en tu documento:

1. ¿Qué tipo de dato devuelve la función input()?

   - La función input() devuelve un dato de tipo cadena de texto (string).

2. ¿Por qué el conjunto no muestra los valores duplicados?

   - Un conjunto no muestra valores duplicados porque almacena solo elementos únicos.

3. ¿Cuál es la diferencia entre una lista y una tupla?

   - La diferencia entre una lista y una tupla es que la lista es mutable (puede modificarse) y la tupla es inmutable (no se puede modificar).

4. ¿Qué sucede si cambias el orden de los elementos en un conjunto?

   - Cambiar el orden de los elementos en un conjunto no afecta porque los conjuntos son estructuras no ordenadas.

5. ¿Por qué las listas permiten modificaciones y las tuplas no?

   - Las listas permiten modificaciones porque están diseñadas para ser mutables, mientras que las tuplas son inmutables para garantizar seguridad y eficiencia.

6. ¿Qué ventajas tiene usar conjuntos en un programa?

   - Las ventajas de usar conjuntos son eliminar duplicados automáticamente, realizar búsquedas rápidas y facilitar operaciones como unión, intersección y diferencia.

7. En el bloque de lógica, ¿por qué crees que se usan condiciones en orden (if, elif, else) y no todas a la vez?

   - En un bloque de lógica, se usan condiciones en orden (if, elif, else) porque el programa evalúa secuencialmente y se detiene al cumplirse una condición, evitando evaluaciones innecesarias.

Entrega

Envía una **captura o archivo** mostrando:

- Tu código completo y comentado.
- El resultado en la consola al ejecutar el programa.
- Tus respuestas a las preguntas de reflexión.