Trabajo de IA

Módulo: IA

Actividad: Ejercicios de Python

Curso académico: 2025/26

Alumno: Darel Martínez Caballero

Índice

1. Objetivo del documento.....3

2. Problemas planteados y razonamientos.....3

1. Conversión de tipos y entrada de datos.....3

2. Control de flujo (if/else).....3

3. Bucles.....3

4. Estadísticas.....3

5. Diccionarios y conteo.....4

6. Listas por comprensión.....4

7. Cadenas y slicing.....4

8. Generación de listas.....4

9. Excepciones.....4

10. Tuplas y desempacado.....4

11. Clases y objetos.....5

3. Conclusiones.....5

4. Bibliografía.....5

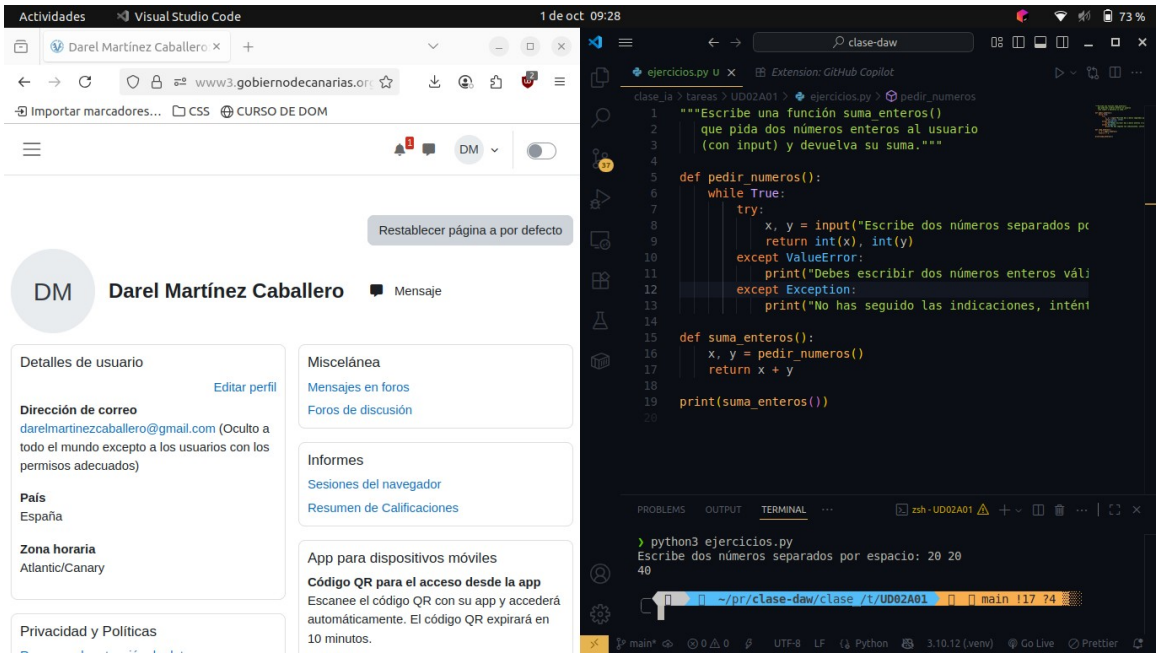
1. Objetivo del documento

El presente documento tiene como finalidad desarrollar una serie de ejercicios de programación en Python. Estos ejercicios abarcan diferentes conceptos fundamentales como estructuras de control, bucles, listas, diccionarios, funciones y clases. El objetivo es afianzar los conocimientos teóricos mediante la práctica.

2. Problemas planteados y razonamientos

1. Conversión de tipos y entrada de datos

Escribe una función suma_enteros() que pida dos números enteros al usuario (con input) y devuelva su suma.



Función pedir_numeros()

- Entra en un **bucle infinito** (while True) hasta que el usuario dé un input válido.
- Pide al usuario que escriba **dos números separados por espacio**.
- Intenta hacer dos cosas:
    1. **Separar el input** usando .split() y guardar los valores en x e y.
    2. **Convertir** x e y a enteros con int(x) y int(y).
- Si **la conversión falla** (no son enteros), muestra un mensaje: "Debes escribir dos números enteros válidos."

- Si ocurre **cualquier otro error**, muestra: "No has seguido las indicaciones, inténtalo otra vez."
- Cuando ambos números son válidos, **los retorna** como una tupla (x, y) y sale del bucle.

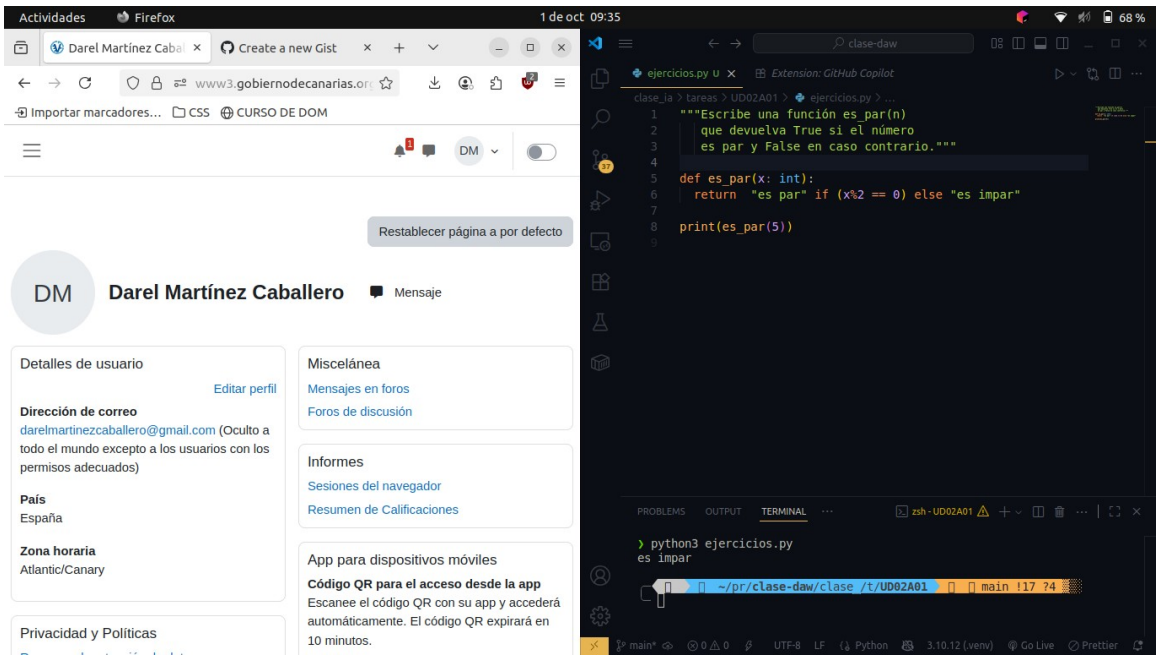
Función suma_enteros()

- Llama a pedir_numeros() para obtener los dos números enteros.
- **Suma** los dos números: x + y.
- **Devuelve** el resultado de la suma.

Código para su ejecución:
https://gist.github.com/DarelShroo/5d9e6c60815457be1a6f2cbce9d1e5a5

2. Control de flujo (if/else)

Escribe una función es_par(n) que devuelva True si el número es par y False en caso contrario.



Función es_par(x: int)

- Recibe un **número entero** x como parámetro.
- Calcula x % 2:
    - % es el **operador módulo**, que devuelve el **resto** de dividir x entre 2.
    - Si x % 2 == 0, significa que x es **divisible entre 2**, por lo tanto **es par**.

    - Si x % 2 != 0, significa que no es divisible entre 2, por lo tanto **es impar**.
- Usa un **operador ternario** (condición en una sola línea)

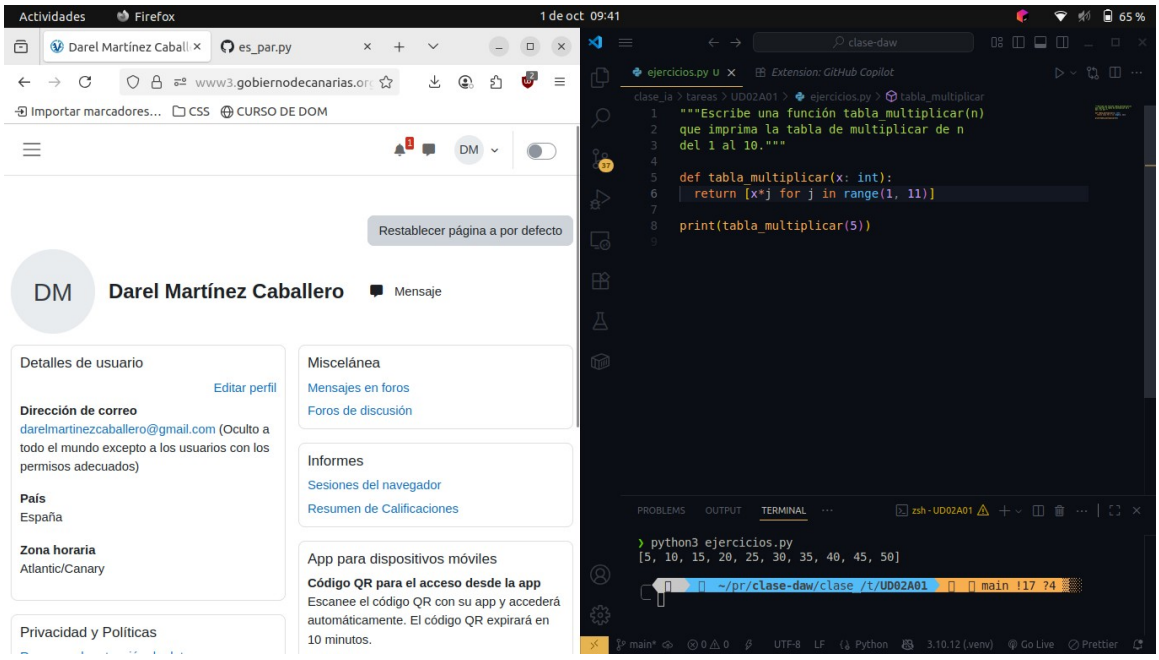
Ejecución final

- print(es_par(5)):
    1. Llama a es_par(5).
    2. Como 5 % 2 da 1, devuelve "es impar".
    3. print() muestra en pantalla "es impar".

Código para su ejecución:
https://gist.github.com/DarelShroo/61493c171e9eaf88e591fad6a8de9f00

3. Bucles

Escribe una función tabla_multiplicar(n) que imprima la tabla de multiplicar de n del 1 al 10.



Función tabla_multiplicar(x: int)

- Recibe un **número entero** x como parámetro.
- Crea una **lista por comprensión**:
    - j recorre los valores del **1 al 10** (range(1, 11)).
    - Cada elemento de la lista es x * j.

- Devuelve la **lista completa** de resultados de la multiplicación

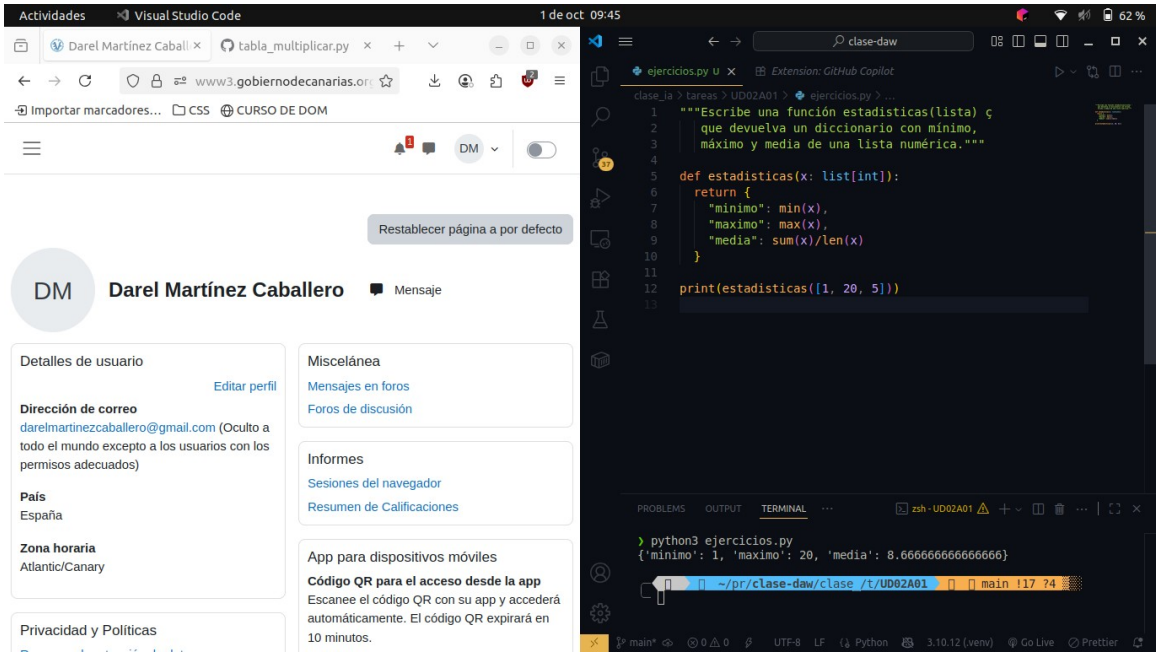
Ejecución final

- print(tabla_multiplicar(5)):
    1. Llama a tabla_multiplicar(5).
    2. La lista resultante será: [5, 10, 15, 20, 25, 30, 35, 40, 45, 50].
    3. print() muestra esta lista en pantalla.

Código para su ejecución:
https://gist.github.com/DarelShroo/9d29407d8f6eba8e6327b140a410992e

4. Estadísticas

Escribe una función estadisticas(lista) que devuelva un diccionario con mínimo, máximo y media de una lista numérica.



Función estadisticas(x: list[int])

- Recibe una **lista de números enteros o flotantes** x.
- Devuelve un **diccionario** con tres claves:
    1. "minimo": utiliza min(x) para obtener el **valor más pequeño** de la lista.
    2. "maximo": utiliza max(x) para obtener el **valor más grande** de la lista.
    3. "media": calcula la **media aritmética** con sum(x)/len(x):

- sum(x) suma todos los elementos de la lista.
- len(x) devuelve la cantidad de elementos.
- La división da el promedio.

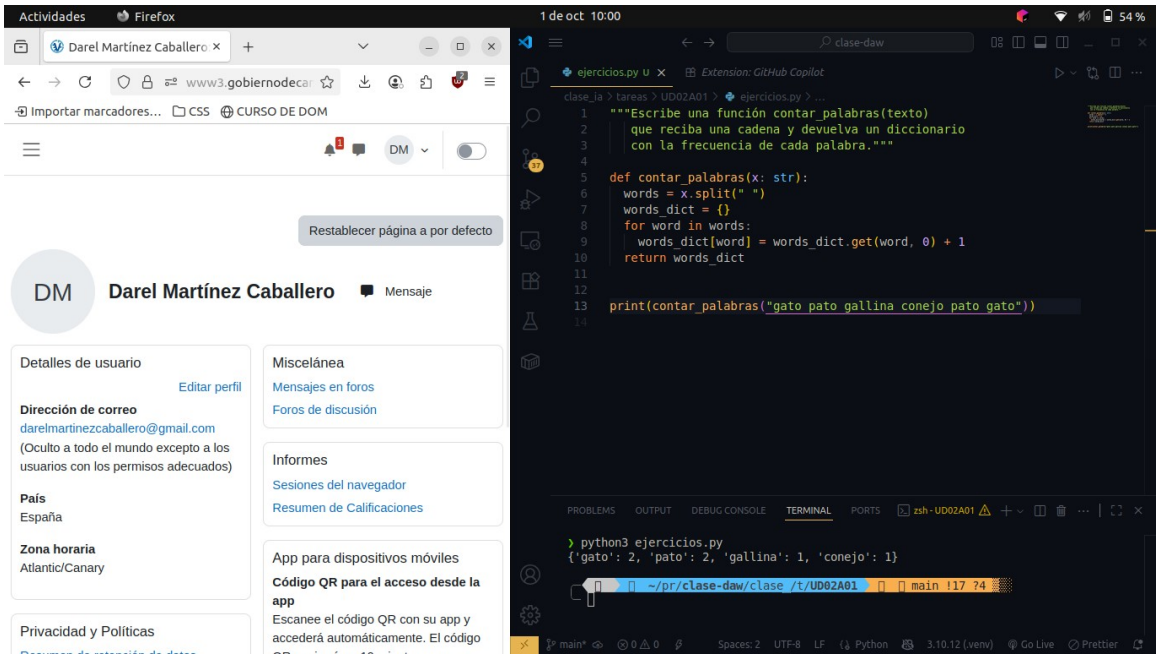
Ejecución final

- print(estadisticas([1, 20, 5])):
    1. Llama a estadisticas([1, 20, 5]).
    2. Calcula:
        - minimo = 1
        - maximo = 20
        - media = (1 + 20 + 5) / 3 = 26 / 3 ≈ 8.6667
    3. Devuelve el diccionario
    4. print() muestra el diccionario en pantalla.

Código para su ejecución:
https://gist.github.com/DarelShroo/7c931faf8e1ccb1f3bb2f5f8845f2ac6

5. Diccionarios y conteo

Escribe una función contar_palabras(texto) que reciba una cadena y devuelva un diccionario con la frecuencia de cada palabra.



Función contar_palabras(x: str)

- Recibe una **cadena de texto** x.
- Divide el texto en **palabras** usando split(" "):
    - Esto genera una **lista de palabras**, separadas por espacios.
- Crea un **diccionario vacío** words_dict para almacenar la frecuencia de cada palabra.
- Recorre cada palabra en la lista
    - words_dict.get(word, 0) devuelve el valor actual de la palabra en el diccionario, o 0 si no existe.
    - Luego le suma 1, contando así cada aparición de la palabra.
- Devuelve el **diccionario final**, donde cada clave es una palabra y su valor es la cantidad de veces que aparece.

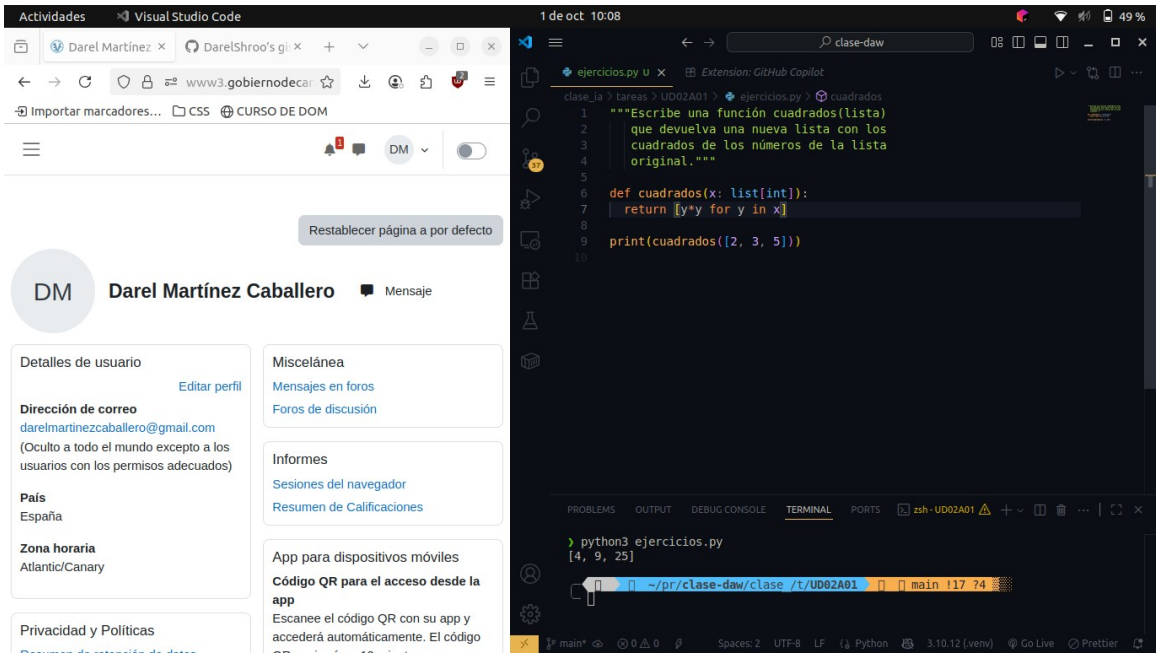
Ejecución final

- print(contar_palabras("gato pato gallina conejo pato gato")):
    1. Divide el texto en palabras: ['gato', 'pato', 'gallina', 'conejo', 'pato', 'gato'].
    2. Cuenta las ocurrencias
    3. Devuelve el diccionario
    4. print() muestra el diccionario en pantalla.

Código para su ejecución:
https://gist.github.com/DarelShroo/5432843f8f6e76b6f00bf5a50dc6f98b

6. Listas por comprensión

Escribe una función cuadrados(lista) que devuelva una nueva lista con los cuadrados de los números de la lista original.



Función cuadrados(x: list[int])

- Recibe una **lista de números enteros** x.
- Crea una **nueva lista** usando **list comprehension**
  - Recorre cada elemento y de la lista x.
  - Calcula el **cuadrado** de y (y*y).
  - Añade ese cuadrado a la nueva lista.
- Devuelve la **lista de cuadrados**.

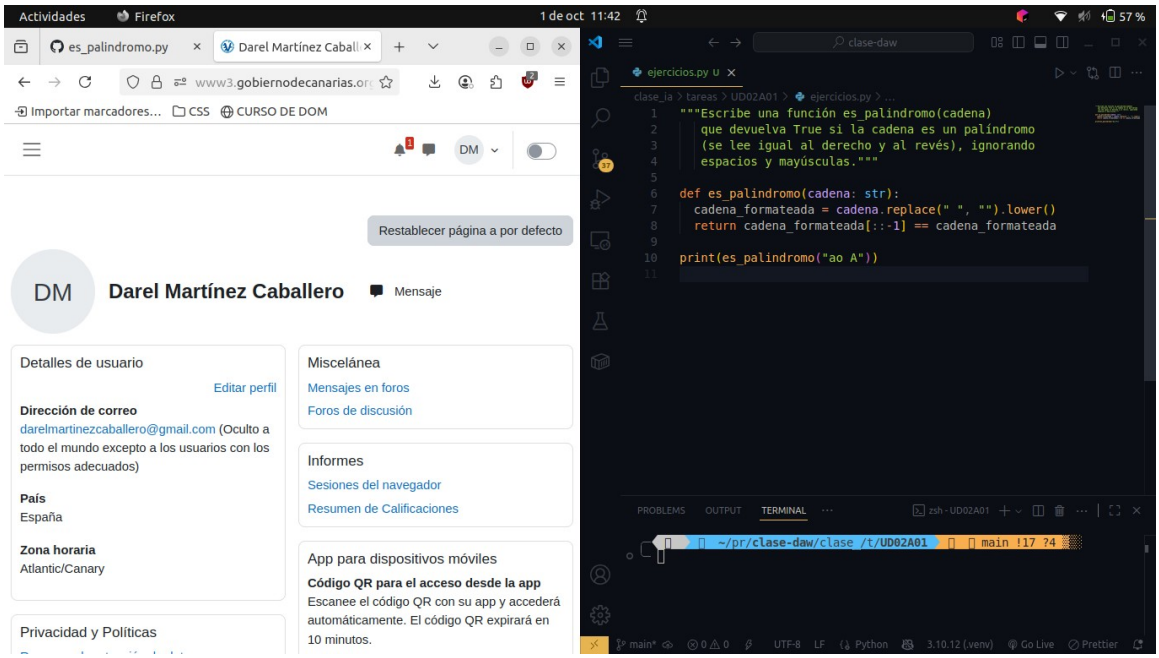
Ejecución final

- print(cuadrados([2, 3, 5])):
  1. Toma la lista [2, 3, 5].
  2. Calcula los cuadrados
  3. Devuelve la lista [4, 9, 25].
  4. print() muestra [4, 9, 25] en pantalla.

Código para su ejecución:
https://gist.github.com/DarelShroo/d10fd7bc8ccb85fba7c2b26fa7339123

7. Cadenas y slicing

Escribe una función es_palindromo(cadena) que devuelva True si la cadena es un palíndromo (se lee igual al derecho y al revés), ignorando espacios y mayúsculas.



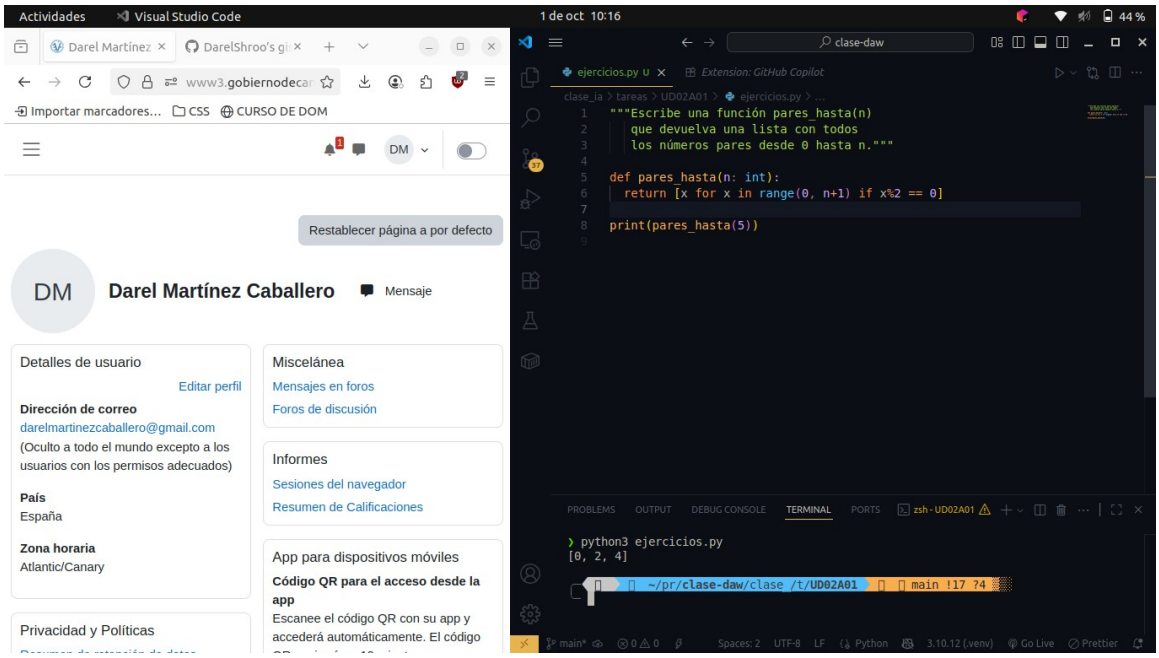
Función es_palindromo(cadena: str)

- Recibe una **cadena de texto** cadena.
- **Formatea la cadena** para compararla correctamente
    - replace(" ", ""): elimina todos los espacios.
    - lower(): convierte todas las letras a minúsculas.
- Comprueba si la cadena es un **palíndromo**
    - [::-1] invierte la cadena.
    - Compara la cadena invertida con la original formateada.
    - Devuelve True si son iguales (es palíndromo), o False si no lo son.

Código para su ejecución:
https://gist.github.com/DarelShroo/3400fadc6b3c82f1d76fe5633f1c4bdc

8. Generación de listas

Escribe una función pares_hasta(n) que devuelva una lista con todos los números pares desde 0 hasta n.



Función pares_hasta(n: int)

- Recibe un **número entero** n.
- Crea una **lista por comprensión**:
    - range(0, n+1) genera todos los números desde 0 hasta n inclusive.
    - if x % 2 == 0 filtra solo los números pares.
    - Cada número par se añade a la nueva lista.
- Devuelve la **lista de números pares**.

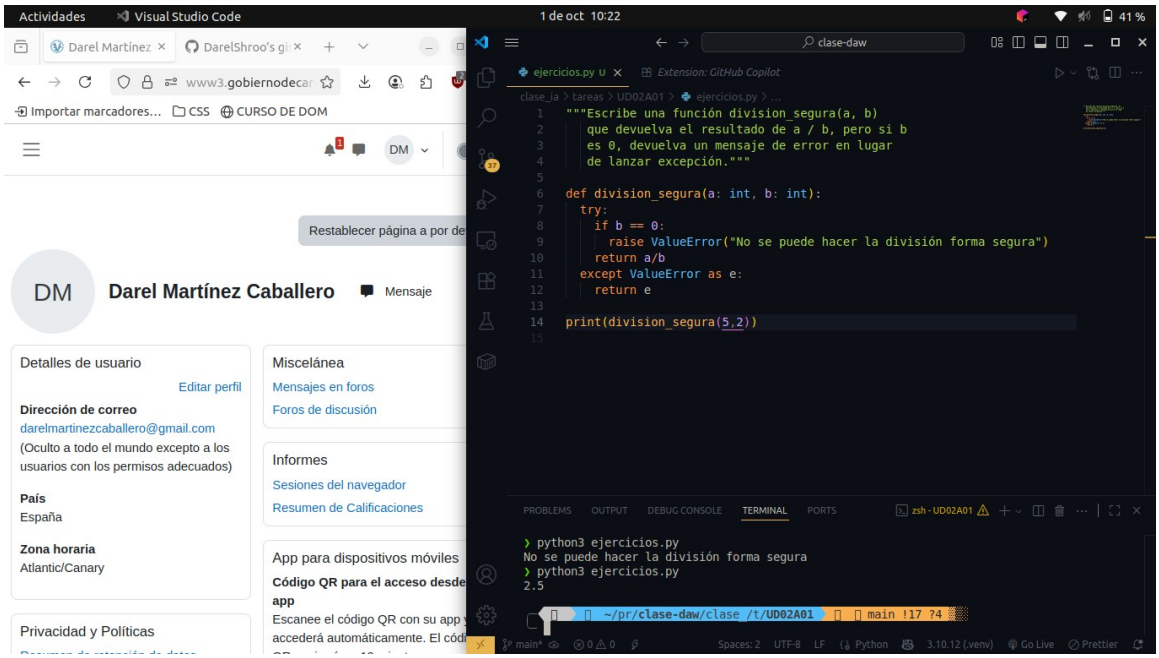
Ejecución final

- print(pares_hasta(5)):
    1. Genera los números del 0 al 5: [0, 1, 2, 3, 4, 5].
    2. Filtra los pares: [0, 2, 4].
    3. print() muestra [0, 2, 4] en pantalla.

Código para su ejecución:
https://gist.github.com/DarelShroo/1711bc8df95f36131889e2a034d83a47

9. Excepciones

Escribe una función division_segura(a, b) que devuelva el resultado de a / b, pero si b es 0, devuelva un mensaje de error en lugar de lanzar excepción.



Función division_segura(a: int, b: int)

- Recibe **dos números enteros** a y b.
- Intenta hacer la división de manera **segura** usando try/except.
- Dentro del try
    - Si b es 0, **lanza una excepción** ValueError con un mensaje personalizado.
    - Esto evita que Python lance el error ZeroDivisionError.

Ejecución final

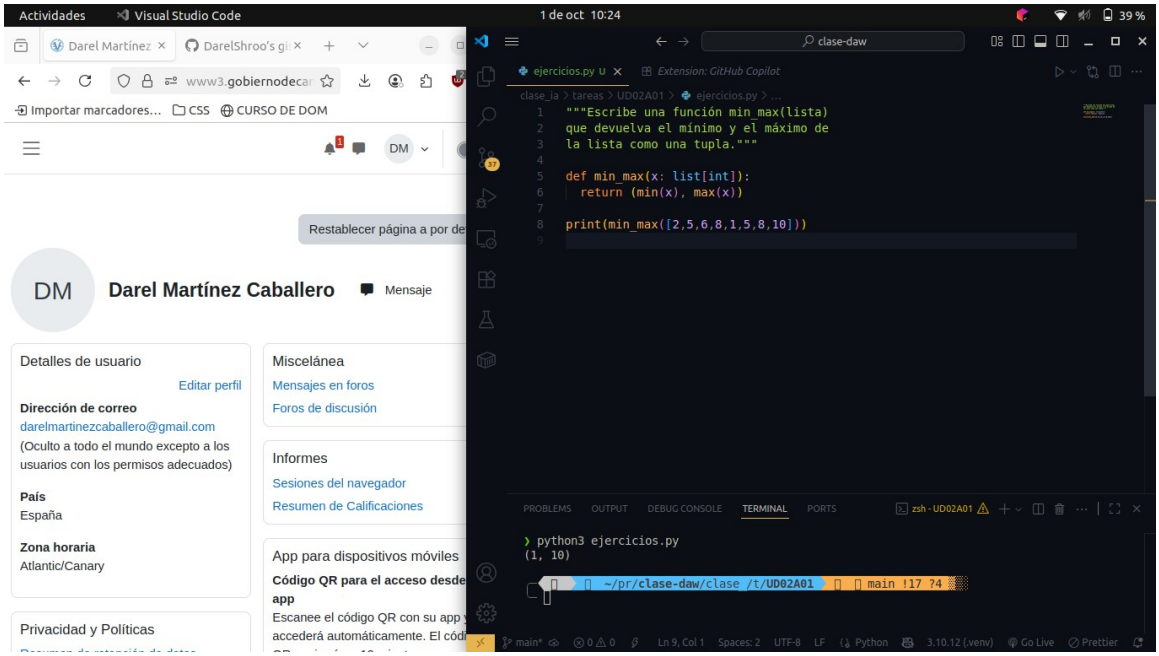
- print(division_segura(5, 2)):
    1. b = 2, no es 0 → hace la división 5 / 2 = 2.5.
    2. Devuelve 2.5.
    3. print() muestra 2.5 en pantalla.
- Si hubieras hecho division_segura(5, 0):
    1. Detecta que b == 0 y lanza ValueError.
    2. El except captura la excepción y devuelve el mensaje: "No se puede hacer la división forma segura".

3. print() mostraría ese mensaje.

Código para su ejecución:
https://gist.github.com/DarelShroo/8b27e0f0810d1d65c498a29506a5259e

10. Tuplas y desempacado

Escribe una función min_max(lista) que devuelva el mínimo y el máximo de la lista como una tupla.



Función min_max(x: list[int])

- Recibe una **lista de números enteros** x.
- Calcula el **mínimo** con min(x).
- Calcula el **máximo** con max(x).
- Devuelve una **tupla** (mínimo, máximo) con esos valores.

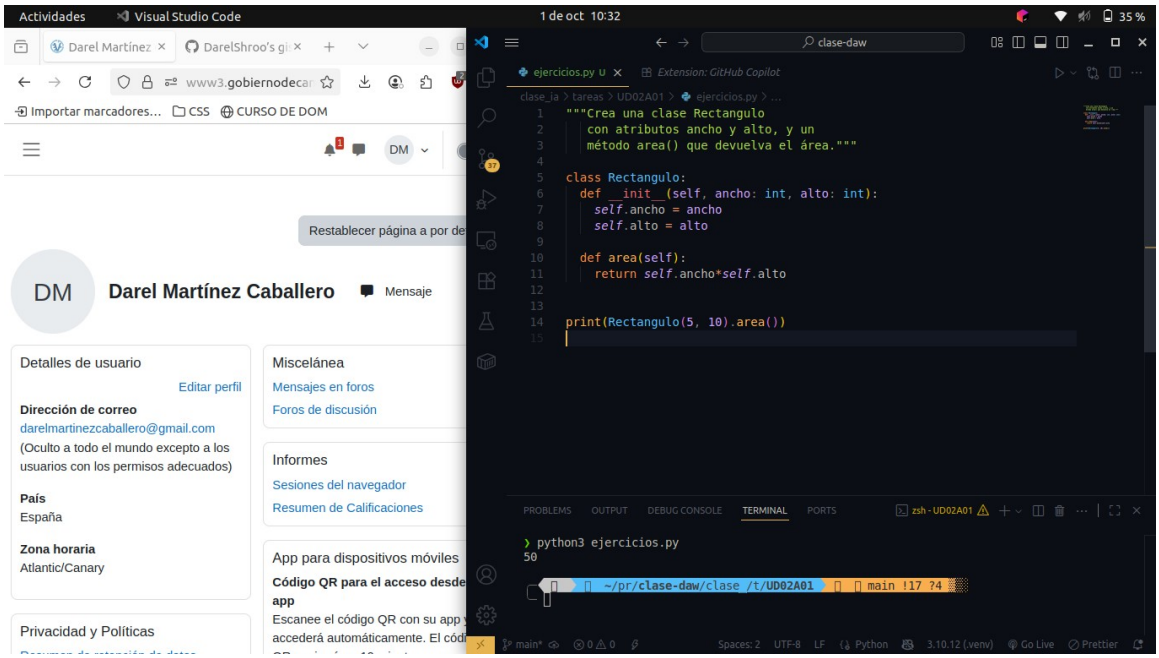
Ejecución final

- print(min_max([2,5,6,8,1,5,8,10])):
    1. Calcula min([2,5,6,8,1,5,8,10]) = 1.
    2. Calcula max([2,5,6,8,1,5,8,10]) = 10.
    3. Devuelve la tupla (1, 10).
    4. print() muestra (1, 10) en pantalla.

Código para su ejecución:
https://gist.github.com/DarelShroo/d579a584f7fa94061295da160e32b318

11. Clases y objetos

Crea una clase Rectangulo con atributos ancho y alto, y un método area() que devuelva el área.



Clase Rectangulo

- Define una **clase** llamada Rectangulo para representar un rectángulo con **ancho** y **alto**.

a) Método __init__

- Se ejecuta cuando se crea un objeto de la clase.
- Recibe dos parámetros: ancho y alto.
- Los almacena como atributos del objeto: self.ancho y self.alto.

b) Método area

- Calcula el **área del rectángulo** multiplicando ancho * alto.
- Devuelve el resultado.

Ejecución final

- Rectangulo(5, 10).area():
    1. Crea un **objeto** Rectangulo con ancho=5 y alto=10.
    2. Llama al método area() del objeto.

3. Calcula 5 * 10 = 50.
4. print() muestra 50 en pantalla.

Código para su ejecución:
https://gist.github.com/DarelShroo/ebf40e2131eb58e93e128cc590c3687d

3. Conclusiones

A través del desarrollo de estos ejercicios se pretende consolidar los fundamentos de la programación en Python. La práctica permitirá comprender la utilidad de cada estructura y su aplicación en la resolución de problemas reales.

4. Bibliografía

• Documentación oficial de Python: https://docs.python.org/3/