HALL TICKET NUMBER: 2403A51365

NAME: RAYAPUDI HARITHA

BATCH: 24BTCAICSB14

**Lab 2: Exploring Additional AI Coding Tools – Gemini (Collab) and Cursor AI**

**ASSIGNMENT: 2.3**

**TASK 1:**

Use Google Gemini in Collab to write a function that reads a CSV file and calculates mean, min, max.

Enter the prompt

**Prompt**: generate a python code for displaying all the elements from .csv file and calculating mean , max, min.

**Step 1:** create a file with .csv extension

A screenshot of a computer

AI-generated content may be incorrect.

**Step 2:** Open Google Collab and upload the file.

A screenshot of a computer

AI-generated content may be incorrect.

* Upload the file into files section.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3:** enter the prompt

**Prompt**: generate a python code for displaying all the elements from .csv file and calculating mean , max, min.

**Code:** A screenshot of a computer

AI-generated content may be incorrect.

Output:

A white background with black and white clouds

AI-generated content may be incorrect.

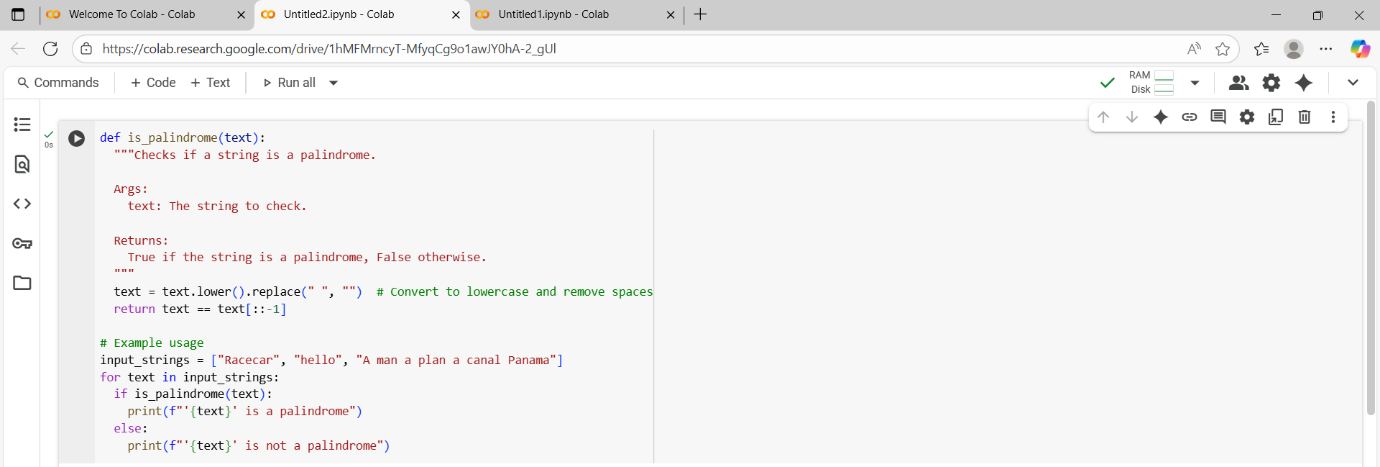
TASK 2: Compare Gemini and Copilot outputs for a palindrome check function.

**Prompt:**

Write a python function for checking palindrome.

GOOGLE COLLAB VERSION:

**Code:**



Output:

A screenshot of a computer

AI-generated content may be incorrect.

COPILOT – VS CODE VERSION:

**Code:** A screen shot of a computer

AI-generated content may be incorrect.

Output:

A screen shot of a computer

AI-generated content may be incorrect.

**Comparison table of Gemini and Copilot output:-**

| **Feature** | **Gemini Output** | **Copilot Output** |
| --- | --- | --- |
| Function Name | is\_palindrome | is\_palindrome |
| Input Cleaning | Lowercase + removes spaces | Lowercase + removes non-alphanumeric chars |
| Case Sensitivity | Not sensitive | Not sensitive |
| Spaces Handling | Yes | Yes |
| Special Characters | Not handled | Handles special characters |
| Code Length | More lines, includes docstring and examples | Shorter, direct function usage |
| Readability | Beginner-friendly with docstring | Concise and practical |
| Robustness | Handles spaces and case, but not special symbols | Handles spaces, case, and special symbols |

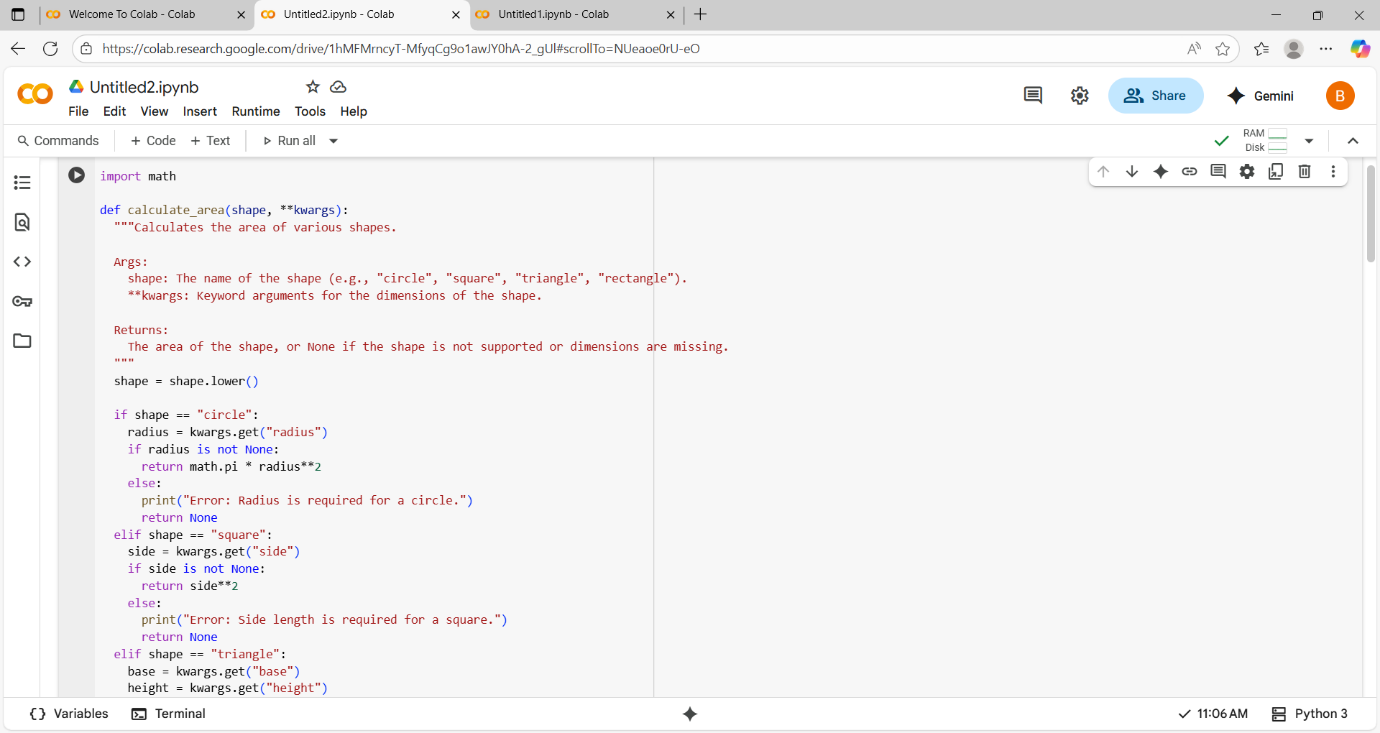
**OBSERVATION: -**

Gemini’s version is way more suitable for beginners whereas Copilot version is more input-safe as it allows users to enter their required input to check. Both are efficient but the copilot code is shorter, but Gemini code is for learners because it has more lines and more examples.

TASK 3: Ask Gemini to explain a Python function (to calculate area of various shapes) line by line

**Prompt:** Explain a python function to calculate are of various shapes line by line.

**CODE:**



A screenshot of a computer

AI-generated content may be incorrect.

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Explanation**

**Here is a detailed explanation of the calculate\_area function:**

* import math: This line imports the math module, which is needed to use math.pi for calculating the area of a circle.
* def calculate\_area (shape, \*\*kwargs): This line defines a function named calculate\_area.
  + shape: This parameter takes a string representing the type of shape (e.g., "circle", "square").
  + \*\*kwargs: This allows the function to accept an arbitrary number of keyword arguments. These arguments will be stored in a dictionary called kwargs and are used to provide the dimensions of the shape (like radius, side, base, height, length, width).
* Docstring ("""Docstring goes here"""): This block explains the purpose of the function, its arguments (Args), and what it returns (Returns).
* shape = shape.lower(): This line converts the input shape string to lowercase. This makes the function case-insensitive, so "Circle" and "circle" are treated the same.
* if shape == "circle": This starts a conditional block. If the lowercase shape is "circle", the code inside this block is executed.
  + radius = kwargs.get("radius"): This line tries to get the value associated with the key "radius" from the kwargs dictionary. If "radius" is not found, it returns None.
  + if radius is not None: This checks if a value for radius was found.
    - `return math.pi \* radius2: If a radius` was provided, this line calculates the area of a circle using the formula π \* radius² and returns the result.
  + else: If no radius was provided.
    - print("Error: Radius is required for a circle."): An error message is printed to the console.
    - return None: The function returns None to indicate that the area could not be calculated.
* elif shape == "square": This block is executed if the shape is not "circle", but is "square".
  + side = kwargs.get("side"): Gets the value for "side" from kwargs.
  + if side is not None: Checks if a side was provided.
    - `return side2: Calculates and returns the area of a square (side²).
  + else: If no side was provided.
    - print("Error: Side length is required for a square."): Prints an error message.
    - return None: Returns None.
* elif shape == "triangle": This block is executed if the shape is not "circle" or "square", but is "triangle".
  + base = kwargs.get("base"): Gets the value for "base" from kwargs.
  + height = kwargs.get("height"): Gets the value for "height" from kwargs.
  + if base is not None and height is not None: Checks if both base and height were provided.
    - return 0.5 \* base \* height: Calculates and returns the area of a triangle (0.5 \* base \* height).
  + else: If either base or height was not provided.
    - print("Error: Base and height are required for a triangle."): Prints an error message.
    - return None: Returns None.
* elif shape == "rectangle": This block is executed if the shape is not "circle", "square", or "triangle", but is "rectangle".
  + length = kwargs.get("length"): Gets the value for "length" from kwargs.
  + width = kwargs.get("width"): Gets the value for "width" from kwargs.
  + if length is not None and width is not None: Checks if both length and width were provided.
    - return length \* width: Calculates and returns the area of a rectangle (length \* width).
  + else: If either length or width was not provided.
    - print("Error: Length and width are required for a rectangle."): Prints an error message.
    - return None: Returns None.
* else: This block is executed if the shape does not match any of the supported shapes.
  + print(f"Error: Shape '{shape}' is not supported."): Prints an error message indicating the unsupported shape.
  + return None: Returns None.
* # Example usage: This is a comment indicating the start of example code that demonstrates how to use the function. The lines following this show how to call the calculate\_area function with different shapes and arguments and print the results**.**

TASK 4: INSTALLATION OF CURSOR AI

Here, I am installing Cursor AI for windows

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a login form

AI-generated content may be incorrect.

PROMPT: Write a python function to calculate sum of squares of a number.

A screenshot of a computer

AI-generated content may be incorrect.

**CODE:**

A computer screen shot of a black screen with colorful text

AI-generated content may be incorrect.

OUTPUT:-

A screenshot of a computer

AI-generated content may be incorrect.

**TASK 5: to write code to calculate sum of add number and even numbers in the list**

**And refactoring the code.**

ORIGINAL CODE:  
A screenshot of a computer

AI-generated content may be incorrect.

OUTPUT:-

A screenshot of a computer

AI-generated content may be incorrect.

REFACTORED CODE:-

A screenshot of a computer program

AI-generated content may be incorrect.

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect. COMPARISION TABLE

| **Feature** | **Original Code** | **Refactored Code** |
| --- | --- | --- |
| Structure | Simple loop | Function + List Comprehension |
| Length | 10 lines | 8 lines |
| Readability | Easier for beginners | Compact and concise |
| Reusability | No | Yes (function can be reused) |
| Efficiency | O(n) | O(n) (same, but cleaner syntax) |

**OBSERVATIONS:**

* The **original version** is more beginner-friendly but longer.
* The **refactored version** is more concise, reusable, and Pythonic.
* Both give the same output, but AI’s approach is better for professional coding standards.