

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: M. Tech/MCA		Assignment Type: Lab	AcademicYear:2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Course Code		Course Title	AI Assisted Problem Solving Using Python
Year/Sem	I/I	Regulation	R24
Date and Day of Assignment	Week1 - TUESDAY	Time(s)	
Duration	2 Hours	Applicable to Batches	M. Tech/MCA
AssignmentNumber:2.3(Present assignment number)/24(Total number of assignments)			
Q.No.	Question		Expected Time to complete
1	<p>Lab 2: Exploring Additional AI Coding Tools – Gemini (Colab) and Cursor AI</p> <p><b>Lab Objectives:</b></p> <ul style="list-style-type: none"> <li>To explore and evaluate the functionality of Google Gemini for AI-assisted coding within Google Colab.</li> <li>To understand and use Cursor AI for code generation, explanation, and refactoring.</li> <li>To compare outputs and usability between Gemini, GitHub Copilot, and Cursor AI.</li> <li>To perform code optimization and documentation using AI tools.</li> </ul> <p><b>Lab Outcomes (LOs):</b> After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> <li>Generate Python code using Google Gemini in Google Colab.</li> <li>Analyze the effectiveness of code explanations and suggestions by Gemini.</li> <li>Set up and use Cursor AI for AI-powered coding assistance.</li> <li>Evaluate and refactor code using Cursor AI features.</li> <li>Compare AI tool behavior and code quality across different platforms.</li> </ul> <p><b>Task Description#1</b></p> <ul style="list-style-type: none"> <li>Use Google Gemini in Colab to write a function that reads a CSV file and calculates mean, min, max.</li> </ul> <p><b>Expected Output#1</b></p>		Week1 - TuesDay

- Functional code with output and screenshot

#### Task Description#2

- Compare Gemini and Copilot outputs for a palindrome check function.

#### Expected Output#2

- Side-by-side comparison and observations

#### Task Description#3

- Ask Gemini to explain a Python function (to calculate area of various shapes) line by line..

#### Expected Output#3

- Detailed explanation with code snippet

#### Task Description#4

- Install and configure Cursor AI. Use it to generate a Python function (e.g., sum of squares).

#### Expected Output#4

- Screenshots of working environments with few prompts to generate python code

#### Task Description#5

- Student need to write code to calculate sum of add number and even numbers in the list

#### Expected Output#5

- Refactored code written by student with improved logic

**Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots**

#### Evaluation Criteria:

Criteria	Max Marks
Successful Use of Gemini in Colab (Task#1 & #2)	2.5
Code Explanation Accuracy (Gemini) (Task#3)	2.5
Cursor AI Setup and Usage (Task#4)	2.5
Refactoring and Improvement Analysis (Task#5)	2.5
<b>Total</b>	<b>10 Marks</b>

## TASK DESCRIPTION -1

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

```
File Edit Selection View Go ... Mohammed Farnas Ali Mudabbir
TASK 1.py import csv Untitled-1 TASK 4.py TASK 2.py TASK 3.py ...TASK 2 TASK 3.py LAB 2

1 import csv
2 import statistics
3 import os
4
5 # Ensure the CSV is saved inside the 'Assignment2' folder
6 folder = "Assignment2"
7 os.makedirs(folder, exist_ok=True) # create folder if it doesn't exist
8 csv_path = os.path.join(folder, "data.csv")
9
10 # Data to write into the CSV file
11 data = [
12     ["Name", "Age", "Score"],
13     ["Raj", 21, 88],
14     ["Priya", 22, 92],
15     ["Amit", 20, 75]
16 ]
17
18 # Create and write to a CSV file inside Assignment2 folder
19 with open(csv_path, mode="w", newline="") as file:
20     writer = csv.writer(file)
21     writer.writerows(data)
22
23 print(f"CSV file created successfully as '{csv_path}'")
24
25 def analyze_csv(path):
26     """
```

```
File Edit Selection View Go ... Mohammed Farnas Ali Mudabbir
TASK 1.py import csv Untitled-1 TASK 4.py TASK 2.py TASK 3.py ...TASK 2 TASK 3.py LAB 2

25 def analyze_csv(path):
26     """
27     Read CSV at path and compute mean, min, max for each numeric column.
28     Returns a dict mapping column -> {'mean':..., 'min':..., 'max':...}
29     """
30     with open(path, newline='') as f:
31         reader = csv.DictReader(f)
32         if not reader.fieldnames:
33             return {}
34         cols = {name: [] for name in reader.fieldnames}
35         for row in reader:
36             for name, value in row.items():
37                 try:
38                     cols[name].append(float(value))
39                 except (TypeError, ValueError):
40                     continue # ignore non-numeric values
41
42     results = {}
43     for name, values in cols.items():
44         if values:
45             results[name] = {
46                 "mean": statistics.mean(values),
47                 "min": min(values),
48                 "max": max(values),
49             }
50     return results
```

```
49         }
50     return results
51
52
53 # Example usage: prints stats for numeric columns in 'Assignment2/data.csv'
54 if __name__ == "__main__":
55     stats = analyze_csv(csv_path)
56     for col, s in stats.items():
57         print(f"{col}: mean={s['mean']}, min={s['min']}, max={s['max']}")
```

## Expected Output

Functional code with output and screenshot

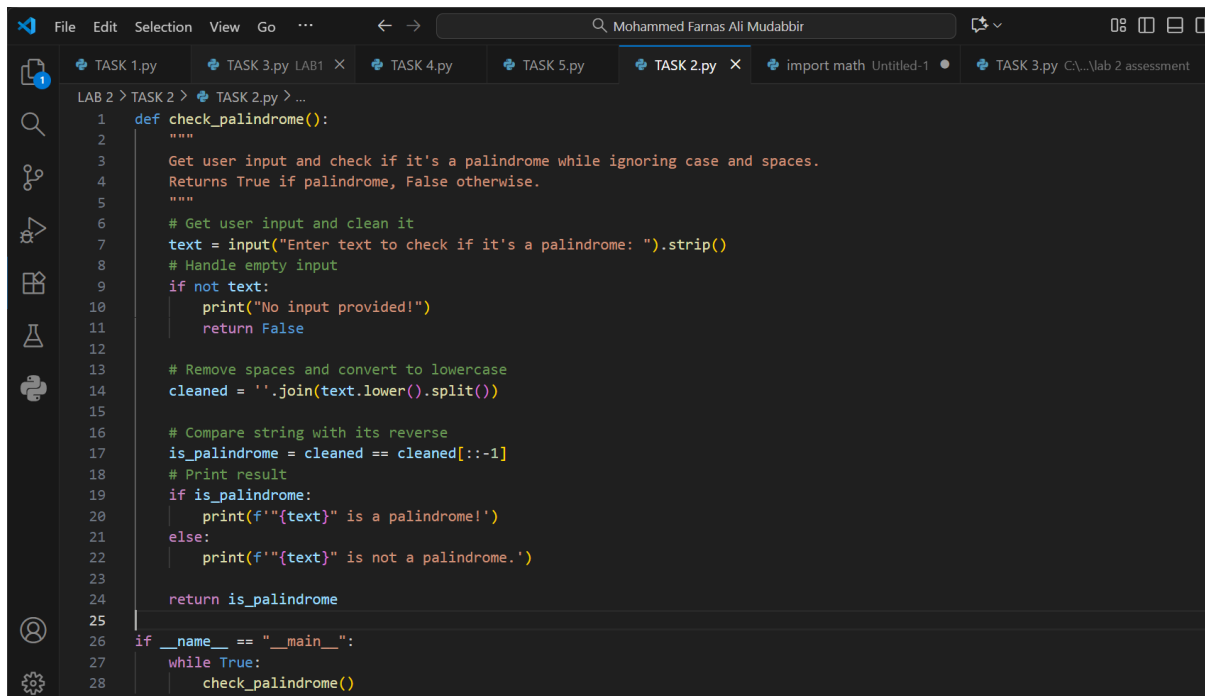
## Practical output:

```
dabbir\tempCodeRunnerFile.python"
dabbir\tempCodeRunnerFile.python"
✅ CSV file created successfully as 'Assignment2\data.csv'
✅ CSV file created successfully as 'Assignment2\data.csv'
Age: mean=21.0, min=20.0, max=22.0
Score: mean=85.0, min=75.0, max=92.0
Score: mean=85.0, min=75.0, max=92.0
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> 
```

## TASK DESCRIPTION -2

Compare Gemini and Copilot outputs for a palindrome check function.

**Prompt:** write a user input palindrome function



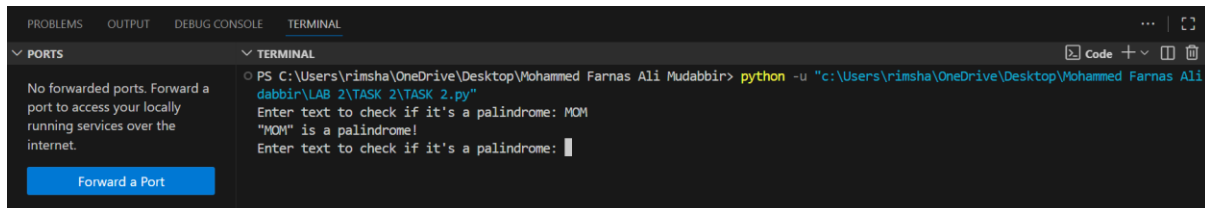
The screenshot shows a VS Code editor window with the file explorer on the left and the editor area on the right. The editor area displays a Python script named `check_palindrome.py`. The script defines a function `check_palindrome()` that takes user input, checks if it's a palindrome (ignoring case and spaces), and prints the result. The script also includes a `__main__` block that runs the function in a loop.

```
1 def check_palindrome():
2     """
3     Get user input and check if it's a palindrome while ignoring case and spaces.
4     Returns True if palindrome, False otherwise.
5     """
6     # Get user input and clean it
7     text = input("Enter text to check if it's a palindrome: ").strip()
8     # Handle empty input
9     if not text:
10        print("No input provided!")
11        return False
12
13    # Remove spaces and convert to lowercase
14    cleaned = ''.join(text.lower().split())
15
16    # Compare string with its reverse
17    is_palindrome = cleaned == cleaned[::-1]
18    # Print result
19    if is_palindrome:
20        print(f"{text} is a palindrome!")
21    else:
22        print(f"{text} is not a palindrome.")
23
24    return is_palindrome
25
26 if __name__ == "__main__":
27     while True:
28         check_palindrome()
```

## Expected Output:

Side-by-side comparison and observations

## Practical output:



The screenshot shows a VS Code terminal window with the command prompt. The command `python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB 2\TASK 2\TASK 2.py"` has been executed. The output shows the program prompting for input, receiving "MOM", and printing "MOM is a palindrome!".

```
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB 2\TASK 2\TASK 2.py> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB 2\TASK 2\TASK 2.py"
Enter text to check if it's a palindrome: MOM
"MOM" is a palindrome!
Enter text to check if it's a palindrome: 
```

## Google colab:

The screenshot shows the Google Colab interface. On the left, there's a 'Table of contents' sidebar with links like 'Welcome to Colab!', 'Getting started', 'Data science', 'Machine learning', and 'More resources'. The main editor area contains a Python function `def is_palindrome(text):` with a docstring explaining it checks if text is a palindrome. The function takes a string `text` and returns a boolean. It removes spaces and converts to lowercase for a case-insensitive check. Below the code, there's a prompt 'Enter a word or phrase: Madam' and the output 'Madam is a palindrome!'. The bottom status bar shows '00:34' and 'Pyt'.

```
def is_palindrome(text):
    """Checks if a given text is a palindrome (reads the same forwards and backward).

    Args:
        text: The input string to check.

    Returns:
        True if the text is a palindrome, False otherwise.
    """
    # Remove spaces and convert to lowercase for case-insensitive check
    cleaned_text = text.replace(" ", "").lower()
    return cleaned_text == cleaned_text[::-1]

# Get user input
user_input = input("Enter a word or phrase: ")

# Check if it's a palindrome and print the result
if is_palindrome(user_input):
    print(f'{user_input} is a palindrome!')
else:
    print(f'{user_input} is not a palindrome.')

Enter a word or phrase: Madam
'Madam' is a palindrome!
```

## TASK DESCRIPTION -3

Detailed explanation with code snippet

**Prompt:** Write a user input python function to calculate area of various shape

The screenshot shows a code editor with a file explorer on the left. The main editor area contains a Python function `def calculate_area():` with a docstring explaining it calculates the area of different shapes based on user input. The function uses a dictionary `shapes` to map shape names to their required parameters. It prompts the user to enter a shape and then calculates the area based on the provided parameters. The code is as follows:

```
import math

def calculate_area():
    """Calculate area of different shapes based on user input."""

    # Dictionary of available shapes and their required parameters
    shapes = {
        'circle': ['radius'],
        'rectangle': ['length', 'width'],
        'triangle': ['base', 'height'],
        'square': ['side'],
        'pentagon': ['side'],
        'ellipse': ['major radius', 'minor radius']
    }

    print("Available shapes:", ", ".join(shapes.keys()))
    shape = input("Enter the shape: ").lower().strip()

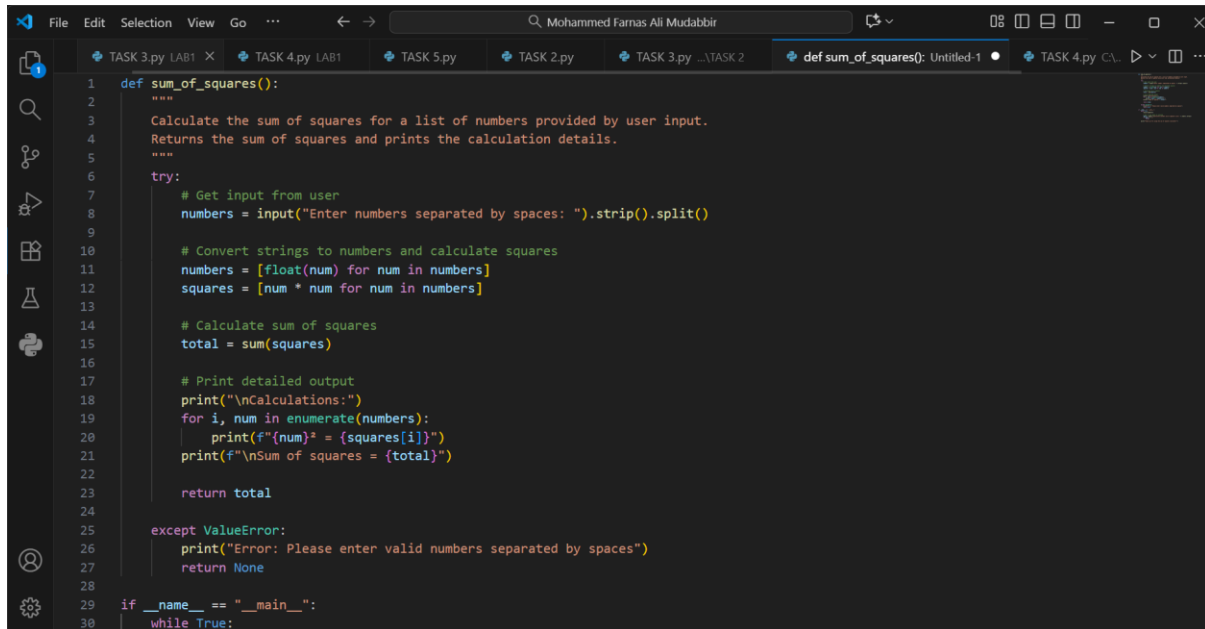
    try:
        if shape not in shapes:
            raise ValueError("Invalid shape selected")

        # Get measurements based on shape requirements
        measurements = {}
        for param in shapes[shape]:
            value = float(input(f"Enter the {param}: "))
```

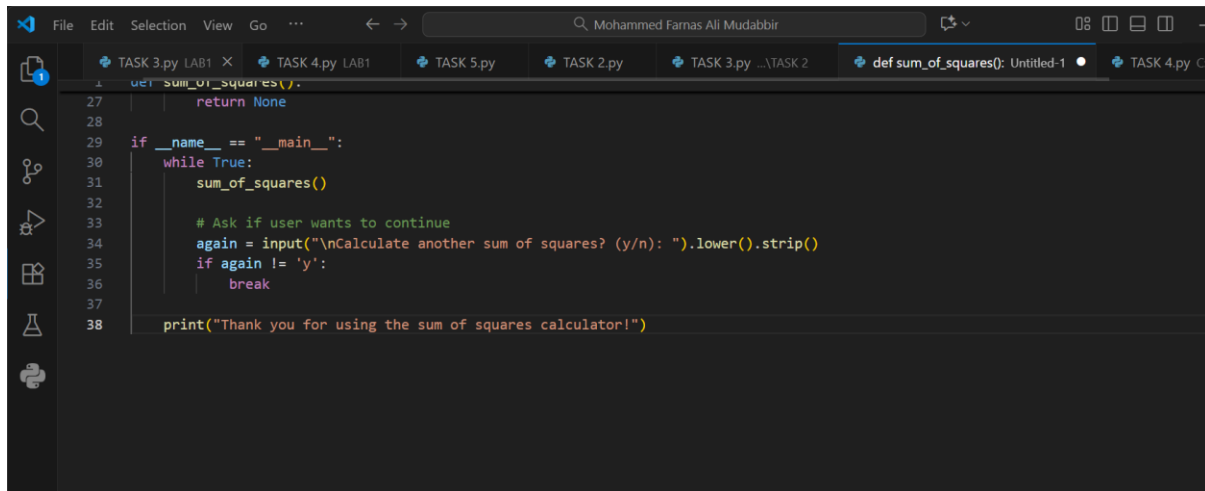


Install and configure Cursor AI. Use it to generate a Python function (e.g., sum of squares).

**Prompt:** Generate a user input python function to calculate sum of square



```
1 def sum_of_squares():
2     """
3     Calculate the sum of squares for a list of numbers provided by user input.
4     Returns the sum of squares and prints the calculation details.
5     """
6     try:
7         # Get input from user
8         numbers = input("Enter numbers separated by spaces: ").strip().split()
9
10        # Convert strings to numbers and calculate squares
11        numbers = [float(num) for num in numbers]
12        squares = [num * num for num in numbers]
13
14        # Calculate sum of squares
15        total = sum(squares)
16
17        # Print detailed output
18        print("\nCalculations:")
19        for i, num in enumerate(numbers):
20            print(f"{num}^2 = {squares[i]}")
21        print(f"\nSum of squares = {total}")
22
23        return total
24
25    except ValueError:
26        print("Error: Please enter valid numbers separated by spaces")
27        return None
28
29 if __name__ == "__main__":
30     while True:
```



```
1 def sum_of_squares():
2     return None
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29 if __name__ == "__main__":
30     while True:
31         sum_of_squares()
32
33         # Ask if user wants to continue
34         again = input("\nCalculate another sum of squares? (y/n): ").lower().strip()
35         if again != 'y':
36             break
37
38     print("Thank you for using the sum of squares calculator!")
```

## Expected Output

Screenshots of working environments with few prompts to generate python code

## Practical output:



```
TERMINAL
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mu
dabbir\tempCodeRunnerFile.python"
Enter numbers separated by spaces: 7 5 4

Calculations:
7.02 = 49.0
5.02 = 25.0
4.02 = 16.0

Sum of squares = 90.0

Calculate another sum of squares? (y/n): 2
Thank you for using the sum of squares calculator!
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir>
```

## TASK DESCRIPTION -5

Student need to write code to calculate sum of add number and even numbers in the list

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

```
File Edit Selection View Go ... Mohammed Farnas Ali Mudabbir
TASK 3.py LAB1 TASK 4.py TASK 5.py LAB1 TASK 2.py TASK 3.py ...\TASK 2 def calculate_odd_even_sums(): Untitled-1 TASK
1 def calculate_odd_even_sums():
2     """
3     Calculate separate sums for odd and even numbers from user input.
4     Returns tuple of (odd_sum, even_sum).
5     """
6     try:
7         # Get input from user
8         numbers = input("Enter numbers separated by spaces: ").strip().split()
9         numbers = [int(num) for num in numbers]
10
11        # Calculate sums using list comprehension
12        even_sum = sum(num for num in numbers if num % 2 == 0)
13        odd_sum = sum(num for num in numbers if num % 2 != 0)
14
15        # Print results
16        print("\nResults:")
17        print(f"Even numbers sum: {even_sum}")
18        print(f"Odd numbers sum: {odd_sum}")
19
20        return (odd_sum, even_sum)
21
22    except ValueError:
23        print("Error: Please enter valid integers separated by spaces")
24        return None
25
26 if __name__ == "__main__":
27     while True:
28         calculate_odd_even_sums()
29
30     # Ask if user wants to continue
```

```
23         print("Error: Please enter valid integers separated by spaces")
24         return None
25
26 if __name__ == "__main__":
27     while True:
28         calculate_odd_even_sums()
29
30     # Ask if user wants to continue
31     again = input("\nCalculate another set of sums? (y/n): ").lower().strip()
32     if again != 'y':
33         break
```

**Expected Output:** Refactored code written by student with improved logic  
**Practical Output:**

```
Enter numbers separated by spaces: 7 5 9

Results:
Even numbers sum: 0
Odd numbers sum: 21
```

