

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
<b>Program Name:</b> M. Tech/MCA		<b>Assignment Type:</b> Lab	
<b>Course Coordinator Name</b>		Venkataramana Veeramsetty	
<b>Course Code</b>		<b>Course Title</b>	AI Assisted Problem Solving Using Python
<b>Year/Sem</b>	I/I	<b>Regulation</b>	R24
<b>Date and Day of Assignment</b>	Week1 - TUESDAY	<b>Time(s)</b>	
<b>Duration</b>	2 Hours	<b>Applicable to Batches</b>	M. Tech/MCA
<b>AssignmentNumber:</b> 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></p> <p>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> <ul style="list-style-type: none"> <li>Functional code with output and screenshot</li> </ul>	Week1 - TuesDay

	<p><b>Task Description#2</b></p> <ul style="list-style-type: none"> <li>Compare Gemini and Copilot outputs for a palindrome check function.</li> </ul> <p><b>Expected Output#2</b></p> <ul style="list-style-type: none"> <li>Side-by-side comparison and observations</li> </ul> <p><b>Task Description#3</b></p> <ul style="list-style-type: none"> <li>Ask Gemini to explain a Python function (to calculate area of various shapes) line by line..</li> </ul> <p><b>Expected Output#3</b></p> <ul style="list-style-type: none"> <li>Detailed explanation with code snippet</li> </ul> <p><b>Task Description#4</b></p> <ul style="list-style-type: none"> <li>Install and configure Cursor AI. Use it to generate a Python function (e.g., sum of squares).</li> </ul> <p><b>Expected Output#4</b></p> <ul style="list-style-type: none"> <li>Screenshots of working environments with few prompts to generate python code</li> </ul> <p><b>Task Description#5</b></p> <ul style="list-style-type: none"> <li>Student need to write code to calculate sum of odd numbers and even numbers in the list</li> </ul> <p><b>Expected Output#5</b></p> <ul style="list-style-type: none"> <li>Refactored code written by student with improved logic</li> </ul> <p><b>Note:</b> Report should be submitted a word document for all tasks in a single document with prompts, comments &amp; code explanation, and output and if required, screenshots</p> <p><b>Evaluation Criteria:</b></p> <table border="1"> <thead> <tr> <th>Criteria</th><th>Max Marks</th></tr> </thead> <tbody> <tr> <td>Successful Use of Gemini in Colab (Task#1 &amp; #2)</td><td>2.5</td></tr> <tr> <td>Code Explanation Accuracy (Gemini) (Task#3)</td><td>2.5</td></tr> <tr> <td>Cursor AI Setup and Usage (Task#4)</td><td>2.5</td></tr> <tr> <td>Refactoring and Improvement Analysis (Task#5)</td><td>2.5</td></tr> <tr> <td><b>Total</b></td><td><b>10 Marks</b></td></tr> </tbody> </table>	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>	
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 C:\...\python assessment\lab 2 assessment ● import csv Untitled-2 ● TASK 1.py X ▶ ▷ ...  
C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 2 > TASK 1.py > ...  
● 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     """  
27         Read CSV at path and compute mean, min, max for each numeric column  
28         Returns a dict mapping column -> {'mean':..., 'min':..., 'max':...}  
29     """
```

```
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
51
52
53 # Example usage: prints stats for numeric columns in 'Assignment2/dat
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']}
```

```
C:\> Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 2 > TASK 1.py > ...
25     def analyze_csv(path):
26
27     for name, values in cols.items():
28         if values:
29             results[name] = {
30                 "mean": statistics.mean(values),
31                 "min": min(values),
32                 "max": max(values),
33             }
34     return results
35
36
37 # Example usage: prints stats for numeric columns in 'Assignment2/dat
38 if __name__ == "__main__":
39     stats = analyze_csv(csv_path)
40     for col, s in stats.items():
41         print(f"{col}: mean={s['mean']}, min={s['min']}, max={s['max']}
```

## Expected Output

Functional code with output and screenshot

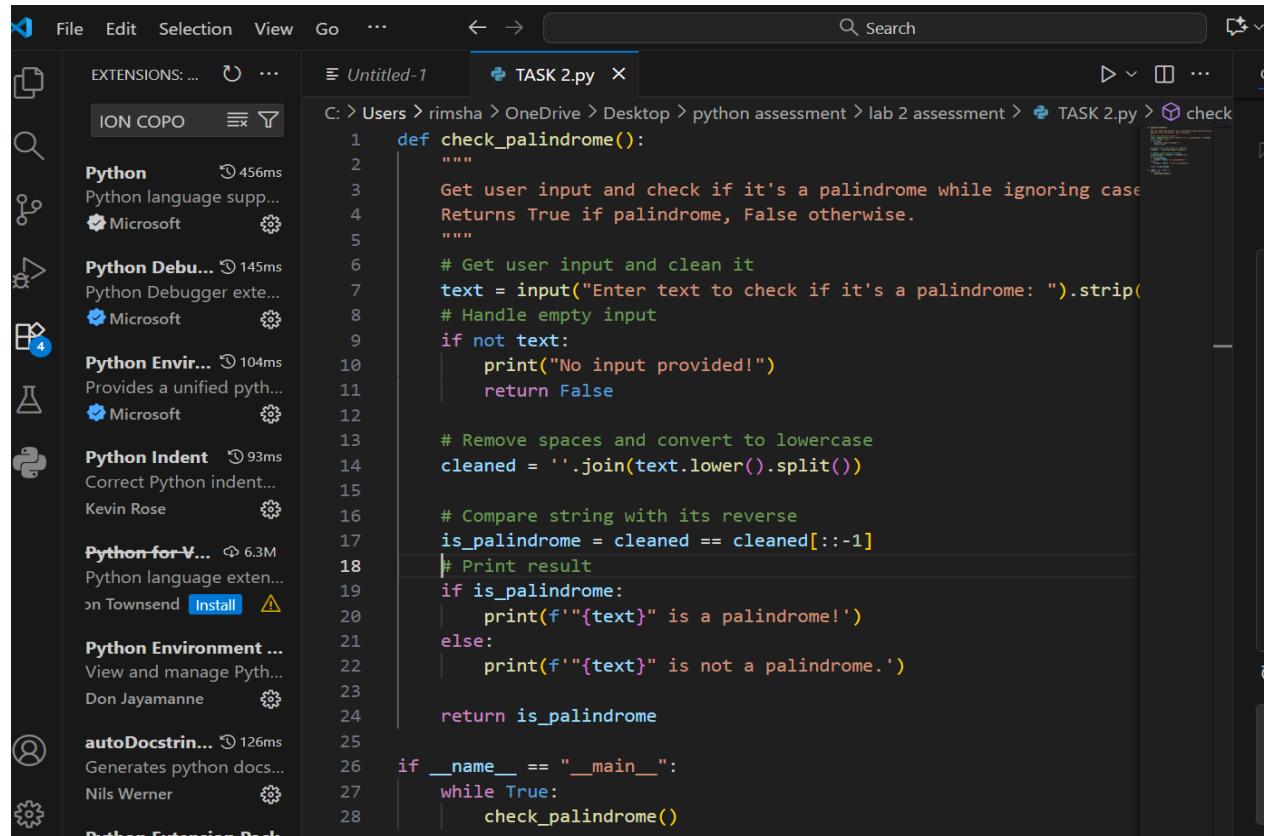
## Practical Output:

```
● PS C:\Users\rimsha> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB 2\TASK 1.py"
    ✓ 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
    ✦ PS C:\Users\rimsha>
```

## TASK DESCRIPTION -2

Compare Gemini and Copilot outputs for a palindrome check function.

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



The screenshot shows the Visual Studio Code interface. On the left is the sidebar with various extensions listed. In the center is the code editor with a file named 'TASK 2.py' open. The code defines a function 'check\_palindrome' that takes user input, removes spaces, converts it to lowercase, and checks if it's a palindrome by comparing it with its reverse. It also handles empty input and prints results.

```
def check_palindrome():
    """
    Get user input and check if it's a palindrome while ignoring case.
    Returns True if palindrome, False otherwise.
    """

    # Get user input and clean it
    text = input("Enter text to check if it's a palindrome: ").strip()
    # Handle empty input
    if not text:
        print("No input provided!")
        return False

    # Remove spaces and convert to lowercase
    cleaned = ''.join(text.lower().split())

    # Compare string with its reverse
    is_palindrome = cleaned == cleaned[::-1]
    # Print result
    if is_palindrome:
        print(f'{text} is a palindrome!')
    else:
        print(f'{text} is not a palindrome.')

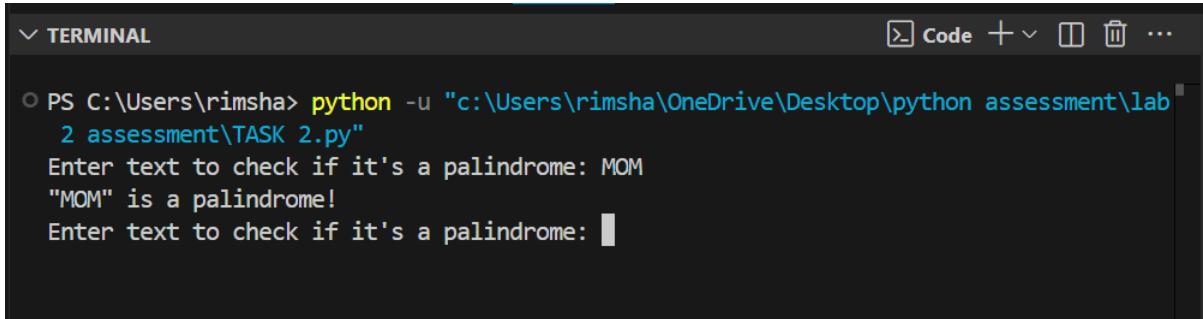
    return is_palindrome

if __name__ == "__main__":
    check_palindrome()
```

## Expected Output:

Side-by-side comparison and observations

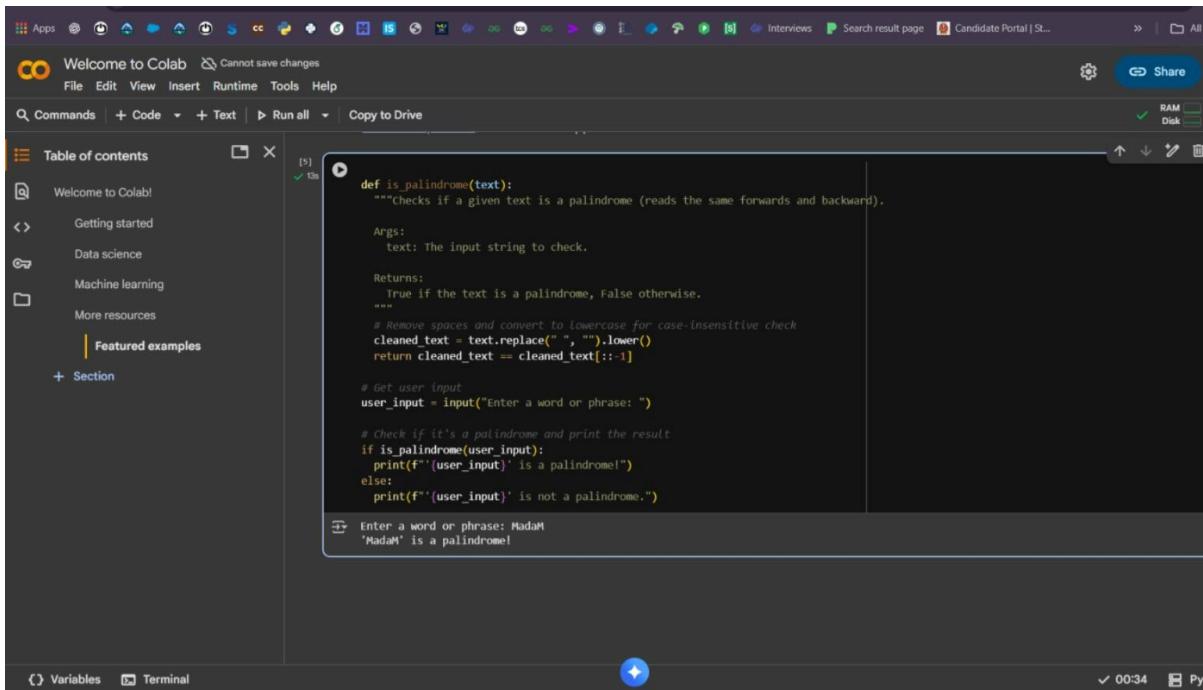
## Practical Output:



The screenshot shows a terminal window with the following text:

```
PS C:\Users\rimsha> python -u "c:\Users\rimsha\OneDrive\Desktop\python assessment\lab 2 assessment\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 a Google Colab interface with the following code in a code cell:

```
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

```
1 import math
2
3 def calculate_area():
4     """Calculate area of different shapes based on user input."""
5
6     # Dictionary of available shapes and their required parameters
7     shapes = {
8         'circle': ['radius'],
9         'rectangle': ['length', 'width'],
10        'triangle': ['base', 'height'],
11        'square': ['side'],
12        'pentagon': ['side'],
13        'ellipse': ['major radius', 'minor radius']
14    }
15
16    print("Available shapes: ", ", ".join(shapes.keys()))
17    shape = input("Enter the shape: ").lower().strip()
18
19    try:
20        if shape not in shapes:
21            raise ValueError("Invalid shape selected")
22
23        # Get measurements based on shape requirements
24        measurements = {}
```

```
result = is_palindrome(s)
print("Palindrome." if result else "Not a palindrome")
return result

if __name__ == "__main__":
    check_palindrome_from_input()
```

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

```
3 def calculate_area():
4     for param in shapes[shape]:
5         value = float(input(f"Enter the {param}: "))
6         if value <= 0:
7             raise ValueError(f"{param} must be positive")
8         measurements[param] = value
9
10    # Calculate area based on shape
11    if shape == "circle":
12        area = math.pi * measurements['radius'] ** 2
13
14    elif shape == "rectangle":
15        area = measurements['length'] * measurements['width']
16
17    elif shape == "triangle":
18        area = 0.5 * measurements['base'] * measurements['height']
19
20    elif shape == "square":
21        area = measurements['side'] ** 2
22
23    elif shape == "pentagon":
24        area = (0.25 * math.sqrt(5 * (5 + 2 * math.sqrt(5)))) *
25            measurements['side'] ** 2
```

```
result = is_palindrome(s)
print("Palindrome." if result else "Not a palindrome")
return result

if __name__ == "__main__":
    check_palindrome_from_input()
```

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

The screenshot shows a code editor interface with several windows open. On the left, there's a sidebar titled 'EXTENSIONS:' listing various Python-related extensions: 'ION COPO', 'Python', 'Python language supp...', 'Python Debug...', 'Python Envir...', 'Python Indent', 'Python-for-V...', 'Python Environment ...', and 'Python for V...'. The main window has tabs for 'Untitled-1', 'TASK 2.py', and 'import math Untitled-2'. The 'TASK 2.py' tab contains the following Python code:

```
3 def calculate_area():
4     if shape == "square":
5         area = measurements['side'] ** 2
6
7     elif shape == "pentagon":
8         area = (0.25 * math.sqrt(5 * (5 + 2 * math.sqrt(5))) *
9                 * measurements['side'] ** 2)
10
11    elif shape == "ellipse":
12        area = math.pi * measurements['major radius'] * measurements['minor radius']
13
14    print(f"\nThe area of the {shape} is: {area:.2f} square units")
15
16    except ValueError as e:
17        print(f"Error: {str(e)}")
18    except Exception as e:
19        print(f"An unexpected error occurred: {str(e)}")
20
21 if __name__ == "__main__":
22     calculate_area()
```

To the right of the code editor is a 'CHAT' window showing a conversation about a palindrome checker. The user asks for a function to calculate the area of various shapes, and the AI responds with a draft of the code.

## Expected Output:

Detailed explanation with code snippet

## Practical Output:

The terminal window shows the execution of the 'python' command followed by the 'TASK 2.py' script. The output is as follows:

```
python
Available shapes: circle, rectangle, triangle, square, pentagon, ellipse
Available shapes: circle, rectangle, triangle, square, pentagon, ellipse
Enter the shape: CIRCLE
Enter the radius: 3
Enter the shape: CIRCLE
Enter the radius: 3

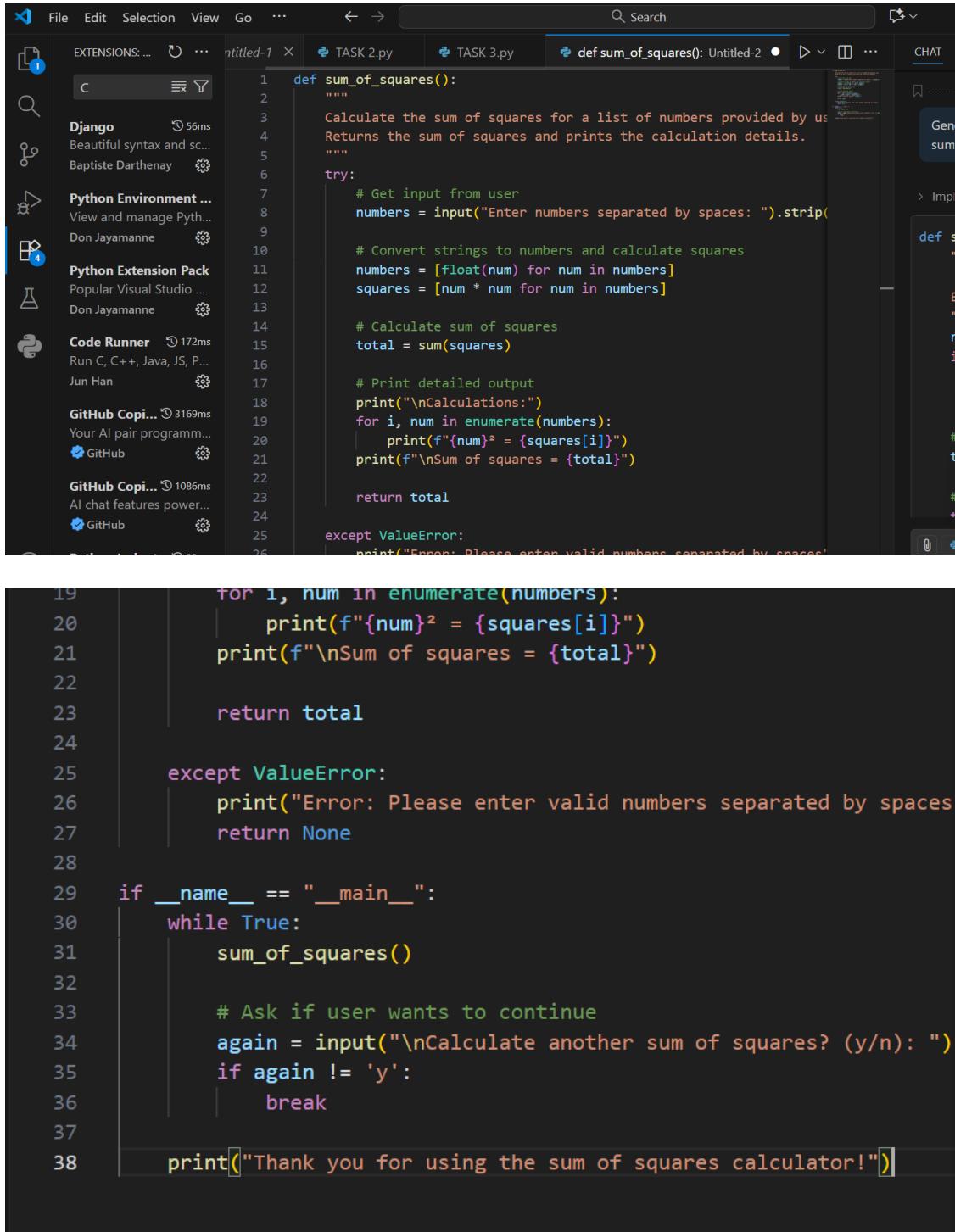
Enter the radius: 3

The area of the circle is: 28.27 square units
The area of the circle is: 28.27 square units
```

## TASK DESCRIPTION -4

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



The screenshot shows a Visual Studio Code interface. On the left, the Extensions sidebar lists several extensions: Django, Python Environment, Python Extension Pack, Code Runner, GitHub Copilot, and GitHub Copilot. The main code editor window displays the following Python code:

```
def sum_of_squares():
    """
    Calculate the sum of squares for a list of numbers provided by user.
    Returns the sum of squares and prints the calculation details.
    """
    try:
        # Get input from user
        numbers = input("Enter numbers separated by spaces: ").strip()

        # Convert strings to numbers and calculate squares
        numbers = [float(num) for num in numbers]
        squares = [num * num for num in numbers]

        # Calculate sum of squares
        total = sum(squares)

        # Print detailed output
        print("\nCalculations:")
        for i, num in enumerate(numbers):
            print(f"{num}^2 = {squares[i]}")
        print(f"\nSum of squares = {total}")

        return total
    except ValueError:
        print("Error: Please enter valid numbers separated by spaces")
        return None

if __name__ == "__main__":
    while True:
        sum_of_squares()

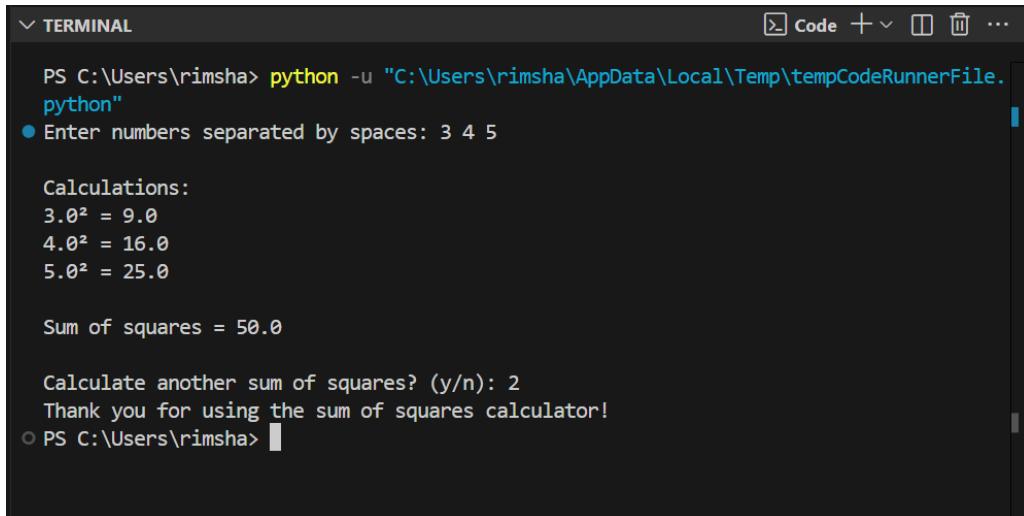
        # Ask if user wants to continue
        again = input("\nCalculate another sum of squares? (y/n): ")
        if again != 'y':
            break

    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:



```
PS C:\Users\rimsha> python -u "C:\Users\rimsha\AppData\Local\Temp\tempCodeRunnerFile.py"
● Enter numbers separated by spaces: 3 4 5

Calculations:
3.02 = 9.0
4.02 = 16.0
5.02 = 25.0

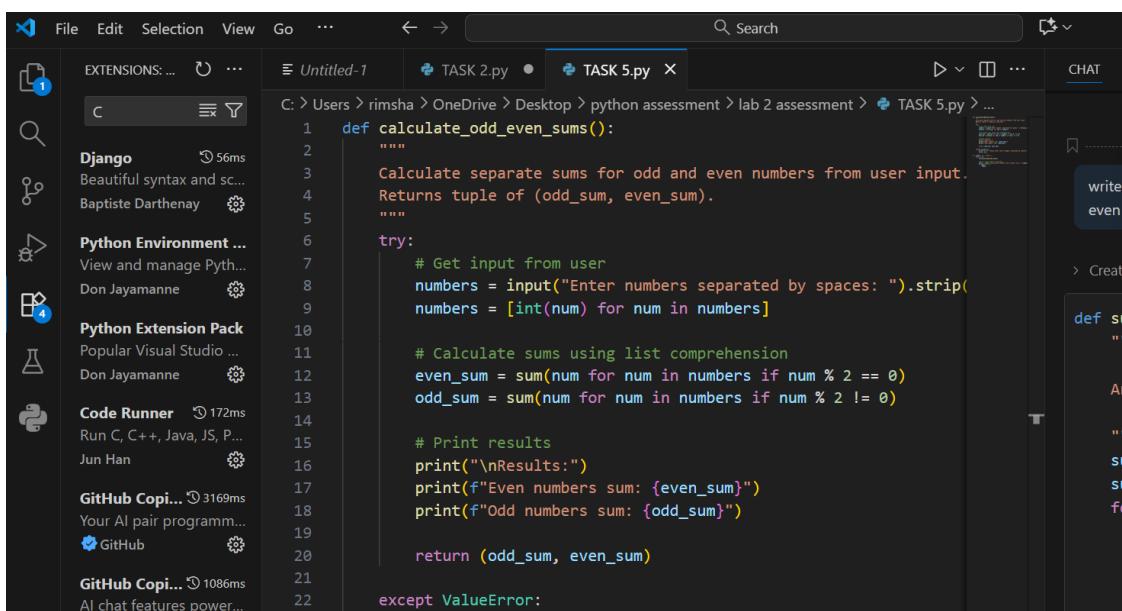
Sum of squares = 50.0

Calculate another sum of squares? (y/n): 2
Thank you for using the sum of squares calculator!
○ PS C:\Users\rimsha>
```

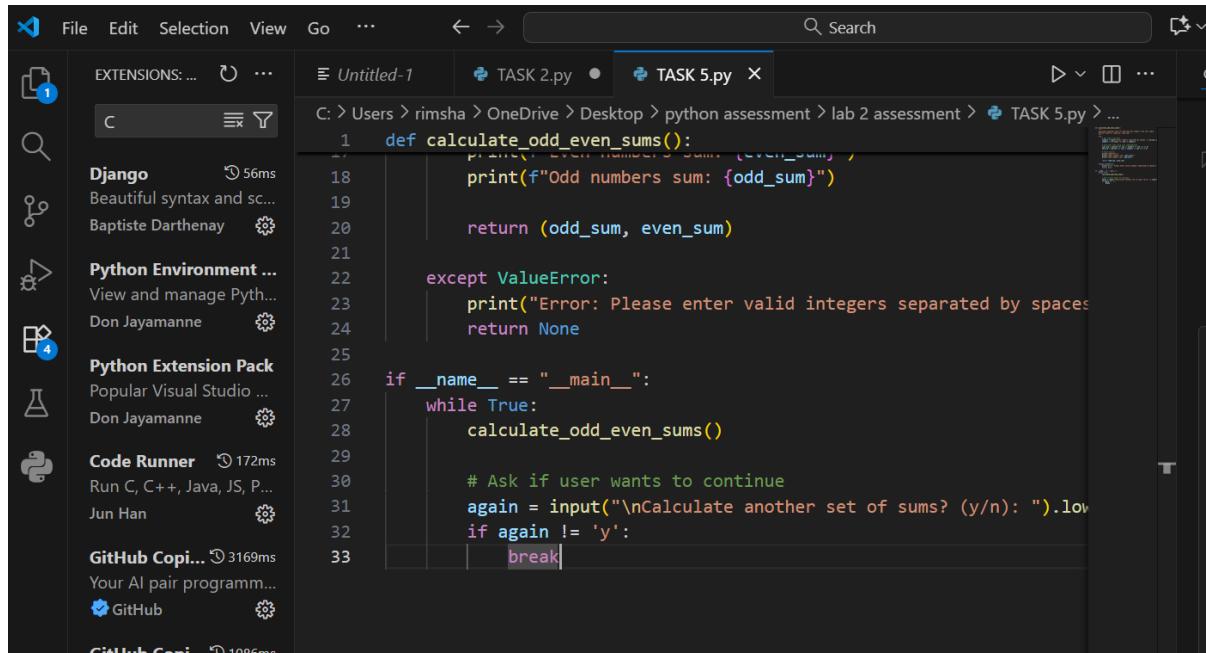
## TASK DESCRIPTION -5

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

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



```
File Edit Selection View Go ... ← → Search CHAT
EXTENSIONS: ... ○ ...
C Untitled-1 TASK 2.py ● TASK 5.py ...
C: > Users > rimsha > OneDrive > Desktop > python assessment > lab 2 assessment > TASK 5.py > ...
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()
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:
```



A screenshot of the Visual Studio Code interface. The left sidebar shows the 'EXTENSIONS' view with several extensions listed:

- Django (56ms)
- Python Environment ...
- Python Extension Pack
- Code Runner (172ms)
- GitHub Copi...
- GitHub

The main editor area has tabs for 'Untitled-1', 'TASK 2.py', and 'TASK 5.py'. The 'TASK 5.py' tab is active, displaying the following Python code:

```
1 def calculate_odd_even_sums():
2     odd_sum = 0
3     even_sum = 0
4     print("Enter numbers separated by spaces: ")
5     numbers = input()
6     for number in numbers.split():
7         if int(number) % 2 == 0:
8             even_sum += int(number)
9         else:
10            odd_sum += int(number)
11    print(f"Even numbers sum: {even_sum}")
12    print(f"Odd numbers sum: {odd_sum}")
13
14    return (odd_sum, even_sum)
15
16 except ValueError:
17     print("Error: Please enter valid integers separated by spaces")
18     return None
19
20
21 if __name__ == "__main__":
22     while True:
23         calculate_odd_even_sums()
24
25         # Ask if user wants to continue
26         again = input("\nCalculate another set of sums? (y/n): ").lower()
27         if again != 'y':
28             break
```

## Expected Output:

Refactored code written by student with improved logic

## Practical Output:

```
python
Enter numbers separated by spaces: 4 7 2
```

Results:

```
Even numbers sum: 6
Odd numbers sum: 7
```

Results:

```
Even numbers sum: 6
Even numbers sum: 6
Odd numbers sum: 7
```