

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> <ul style="list-style-type: none"> <li>Functional code with output and screenshot</li> </ul>		Week1 - TuesDay

**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.

```

C:\...python assessment\lab 2 assessment • import csv Untitled-2 • TASK 1.py × ▶ ▢ ...
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/data'
54 if __name__ == "__main__":
55     stats = analyze_csv(csv_path)
56     for col, s in stats.items():

```

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/data'
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

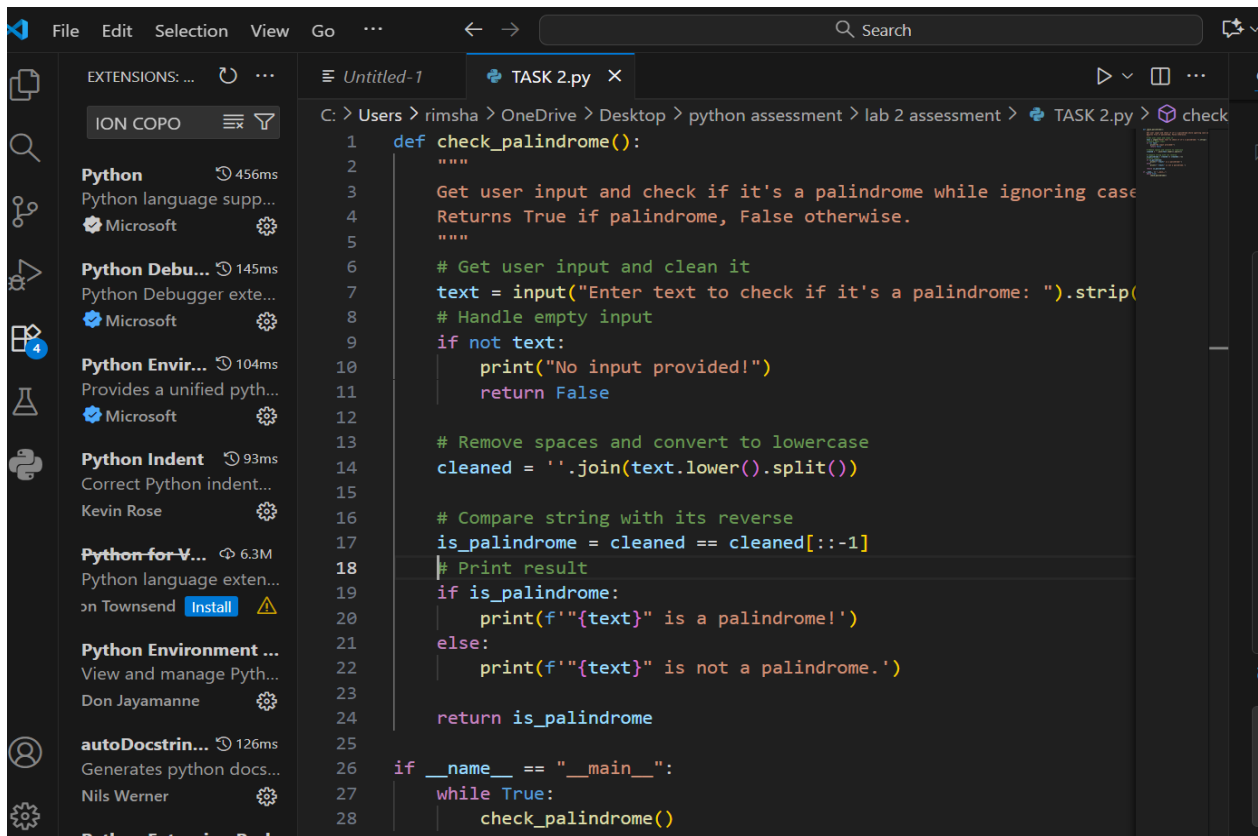
## Practical Output:

```
PS C:\Users\rimsha> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohamed 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



```
File Edit Selection View Go ... Search
EXTENSIONS: ...
ION COPO
Python 456ms
Python language supp...
Microsoft
Python Debu... 145ms
Python Debugger exte...
Microsoft
Python Envir... 104ms
Provides a unified pyth...
Microsoft
Python Indent 93ms
Correct Python indent...
Kevin Rose
Python-for-V... 6.3M
Python language exten...
on Townsend Install
Python Environment ...
View and manage Pyth...
Don Jayamanne
autoDocstrin... 126ms
Generates python docs...
Nils Werner
Python Extension Pack

TASK 2.py
C: > Users > rimsha > OneDrive > Desktop > python assessment > lab 2 assessment > TASK 2.py > check
1 def check_palindrome():
2     """
3     Get user input and check if it's a palindrome while ignoring case
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:

```
▼ TERMINAL
Code + ▾ □ □ □ ...

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:

```
Apps  Welcome to Colab  Cannot save changes  Share
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text ▶ Run all Copy to Drive RAM Disk

Table of contents
Welcome to Colab!
Getting started
Data science
Machine learning
More resources
+ Section

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: ")

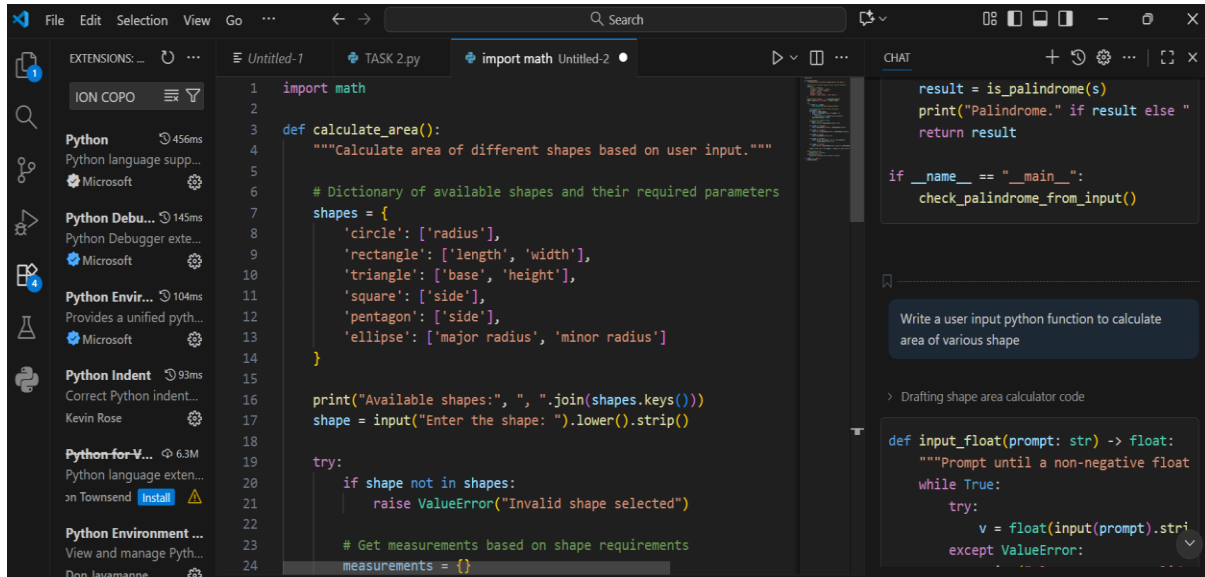
# Check if it's a palindrome and print the result
if is_palindrome(user_input):
    print(f'{user_input} is a palindrome!')
else:
    print(f'{user_input} is not a palindrome.')

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

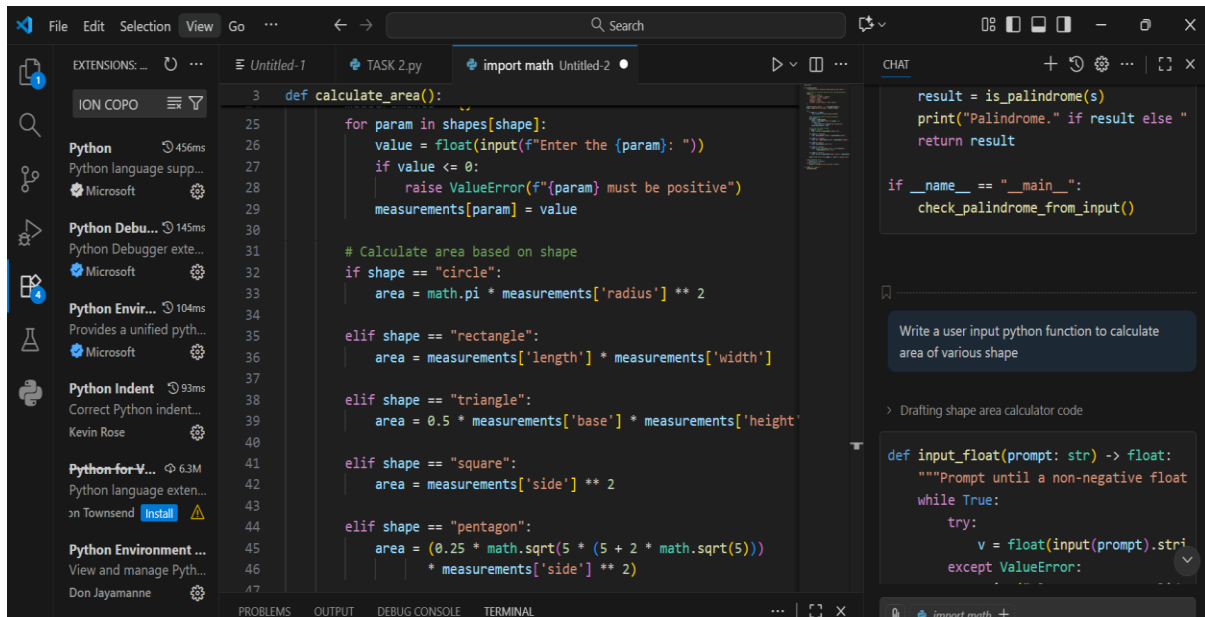
## TASK DESCRIPTION -3

Detailed explanation with code snippet

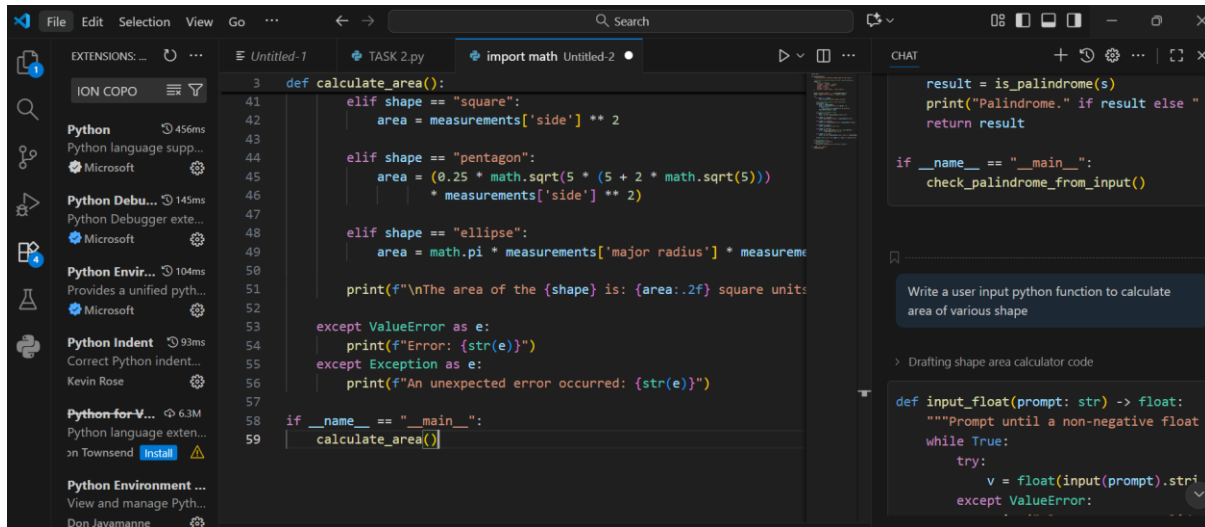
**Prompt:** Write a user input python function to calculate area of various shape



```
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 = {}
```



```
25    for param in shapes[shape]:
26        value = float(input(f"Enter the {param}: "))
27        if value <= 0:
28            raise ValueError(f"{param} must be positive")
29        measurements[param] = value
30
31    # Calculate area based on shape
32    if shape == "circle":
33        area = math.pi * measurements['radius'] ** 2
34
35    elif shape == "rectangle":
36        area = measurements['length'] * measurements['width']
37
38    elif shape == "triangle":
39        area = 0.5 * measurements['base'] * measurements['height']
40
41    elif shape == "square":
42        area = measurements['side'] ** 2
43
44    elif shape == "pentagon":
45        area = (0.25 * math.sqrt(5 * (5 + 2 * math.sqrt(5))))
46        * measurements['side'] ** 2
47
```



## Expected Output:

Detailed explanation with code snippet

## Practical Output:

```
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

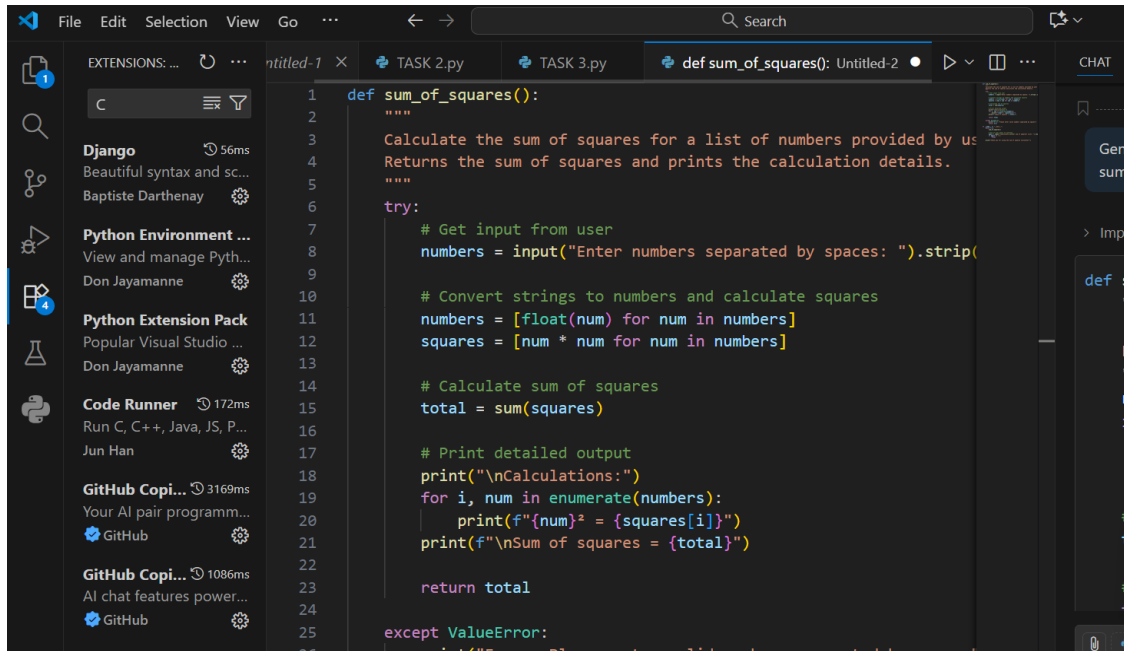
The area of the circle is: 28.27 square units
The area of the circle is: 28.27 square units
```



## TASK DESCRIPTION -4

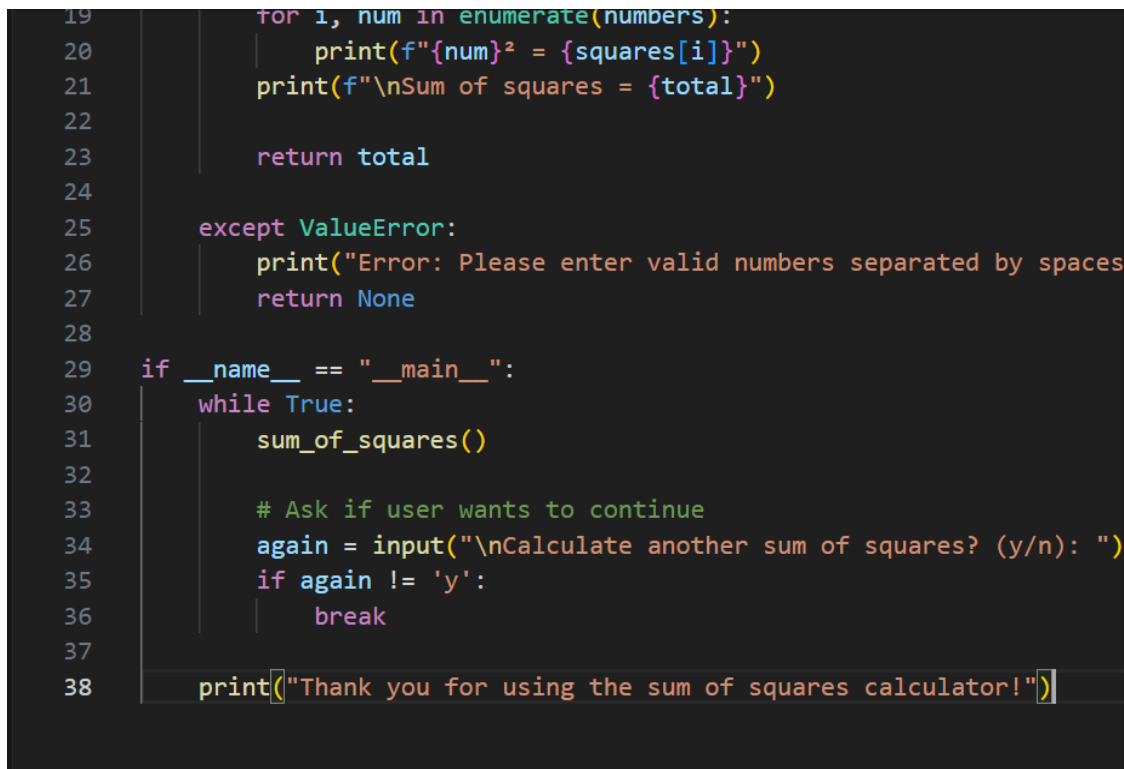
Install and configure Cursor AI. Use it to generate a Python function (e.g., sum of squares).

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



The screenshot shows the Cursor AI IDE interface. The left sidebar displays a list of extensions including Django, Python Environment, Python Extension Pack, Code Runner, and GitHub Copilot. The main editor window shows a Python file named 'def sum\_of\_squares(): Untitled-2'. The code defines a function 'sum\_of\_squares()' that takes a list of numbers as input, calculates the sum of their squares, and prints the result. The function includes a try-except block to handle ValueError exceptions.

```
1 def sum_of_squares():
2     """
3     Calculate the sum of squares for a list of numbers provided by user.
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
```



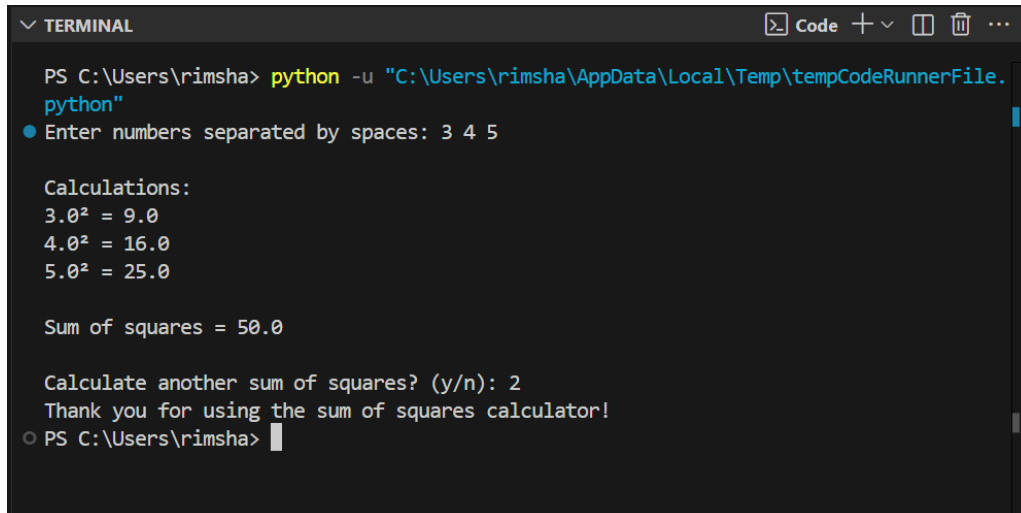
This block shows the continuation of the Python code from the previous screenshot. It includes the main execution logic, which uses a while loop to repeatedly call the 'sum\_of\_squares()' function until the user chooses to stop. The code also includes a final print statement to thank the user for using the calculator.

```
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:
31         sum_of_squares()
32
33         # Ask if user wants to continue
34         again = input("\nCalculate another sum of squares? (y/n): ")
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:



```
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

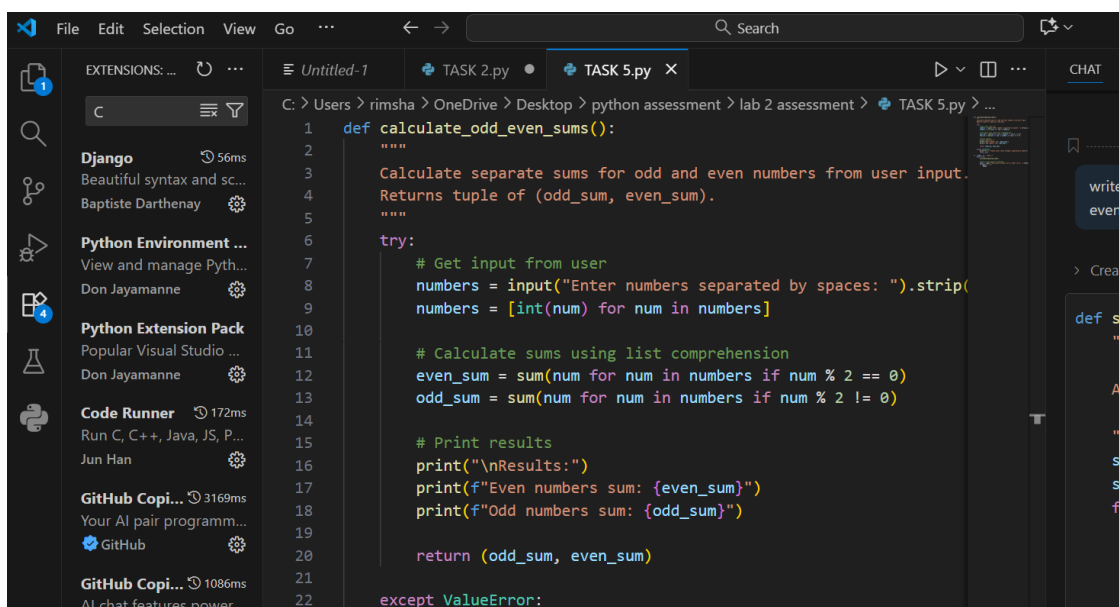
Sum of squares = 50.0

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 add number and even numbers in the list

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



```
def calculate_odd_even_sums():
    """
    Calculate separate sums for odd and even numbers from user input.
    Returns tuple of (odd_sum, even_sum).
    """
    try:
        # Get input from user
        numbers = input("Enter numbers separated by spaces: ").strip().split()
        numbers = [int(num) for num in numbers]

        # Calculate sums using list comprehension
        even_sum = sum(num for num in numbers if num % 2 == 0)
        odd_sum = sum(num for num in numbers if num % 2 != 0)

        # Print results
        print("\nResults:")
        print(f"Even numbers sum: {even_sum}")
        print(f"Odd numbers sum: {odd_sum}")

        return (odd_sum, even_sum)

    except ValueError:
```

The screenshot shows the Visual Studio Code interface with a Python file named 'TASK 5.py' open. The code defines a function 'calculate\_odd\_even\_sums()' that takes a string of numbers separated by spaces as input. It splits the string into a list of integers, calculates the sum of even and odd numbers, and prints the results. The function also includes an exception handler for 'ValueError' to handle invalid input. The main block of the script uses a 'while True' loop to repeatedly prompt the user for input and calculate the sums until the user chooses to stop.

```
1 def calculate_odd_even_sums():
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
31         again = input("\nCalculate another set of sums? (y/n): ").lower()
32         if again != 'y':
33             break
```

### Expected Output:

Refactored code written by student with improved logic

### Practical Output:

The screenshot shows a terminal window with the following 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
```