VEERAGONI MUKESH 2303A51225 BATCH 04

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name	Dr. Rishabh Mittal		
Instructor(s) Name	Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju		
CourseCode	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	**Week1 – Wednesday**	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Applicable to Batches**	All batches
Assignment Number:1.3(Present assignment number)/**24**(Total number of assignments)			

	Question	***Expected Time to co***

		mpl ete
1	Lab 2: Exploring Additional AI Coding Tools beyond Copilot – Gemini (Colab) and Cursor AI **Lab Objectives:** ❖ To explore and evaluate the functionality of Google Gemini for AI-assisted coding within Google Colab. ❖ To understand and use Cursor AI for code generation, explanation, and refactoring. ❖ To compare outputs and usability between Gemini, GitHub Copilot, and Cursor AI. ❖ To perform code optimization and documentation using AI tools. **Lab Outcomes (LOs):** After completing this lab, students will be able to: ❖ Generate Python code using Google Gemini in Google Colab. ❖ Analyze the effectiveness of code explanations and suggestions by Gemini. ❖ Set up and use Cursor AI for AI-powered coding assistance. ❖ Evaluate and refactor code using Cursor AI features. ❖ Compare AI tool behavior and code quality across different platforms.	Wee k1 - Mon day
	Task 1: Refactoring Odd/Even Logic (List Version) ❖ **Scenario:** You are improving legacy code. ❖ **Task:** Write a program to calculate the sum of odd and even numbers in a list, then refactor it using AI. ❖ **Expected Output:** ❖ Original and improved code ❖ **OUTPUT** ❖ **PROMPT :** generate python code Given a list of numbers, calculate the sum of even numbers and the sum of odd numbers separately. ❖ CODE :	

```python
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

even_sum = 0
odd_sum = 0

for num in numbers:
    if num % 2 == 0:
        even_sum += num
    else:
        odd_sum += num

print(f"The sum of even numbers is: {even_sum}")
print(f"The sum of odd numbers is: {odd_sum}")
```

❖ CODE OUPUT

```
The sum of even numbers is: 30
The sum of odd numbers is: 25
```

EXPLANATION

❖ A list named `numbers` is created that contains integers from 1 to 10.

❖ Two variables, `even_sum` and `odd_sum`, are initialized to `0` to store the sums of even and odd numbers.

❖ A `for` loop goes through each number in the list.

❖ If a number is divisible by 2 (`num % 2 == 0`), it is even and added to `even_sum`; otherwise, it is odd and added to `odd_sum`.

❖ Finally, the program prints the sum of even numbers and the sum of odd numbers separately.

Task 2: Area Calculation Explanation

❖ **Scenario:**
You are onboarding a junior developer.

❖ **Task:**
Ask Gemini to explain a function that calculates the area of different shapes.

❖ **Expected Output:**
- ➢ Code
- ➢ Explanation
- ➢ OUTPUT
- ➢ PROMPT : Generate a python function that calculates the area of different shapes.

❖ CODE :

```python
import math

def calculate_area(shape_type, *args):
    shape_type = shape_type.lower()

    if shape_type == 'circle':
        if len(args) == 1:
            radius = args[0]
            return math.pi * radius**2
        else:
            return "Error: Circle requires one argument (radius)."
    elif shape_type == 'square':
        if len(args) == 1:
            side = args[0]
            return side**2
        else:
            return "Error: Square requires one argument (side)."
    elif shape_type == 'rectangle':
        if len(args) == 2:
            length, width = args
            return length * width
        else:
            return "Error: Rectangle requires two arguments (length, width)."
    elif shape_type == 'triangle':
        if len(args) == 2:
            base, height = args
            return 0.5 * base * height
        else:
            return "Error: Triangle requires two arguments (base, height)."
    else:
        return "Error: Unknown shape type. Supported shapes: circle, square, rectangle, triangle."
```

❖ CODE OUTPUT:

```python
# Test cases
print(f"Area of a circle with radius 5: {calculate_area('circle', 5)}")
print(f"Area of a square with side 4: {calculate_area('square', 4)}")
print(f"Area of a rectangle with length 6 and width 3: {calculate_area('rectangle', 6, 3)}")
print(f"Area of a triangle with base 10 and height 5: {calculate_area('triangle', 10, 5)}")

# Error cases
print(f"Unknown shape: {calculate_area('pentagon', 5)}")
print(f"Insufficient arguments for circle: {calculate_area('circle')}")
print(f"Too many arguments for square: {calculate_area('square', 4, 5)}")
```

```
Area of a circle with radius 5: 78.53981633974483
Area of a square with side 4: 16
Area of a rectangle with length 6 and width 3: 18
Area of a triangle with base 10 and height 5: 25.0
Unknown shape: Error: Unknown shape type. Supported shapes: circle, square, rectangle, triangl
Insufficient arguments for circle: Error: Circle requires one argument (radius).
Too many arguments for square: Error: Square requires one argument (side).
```

❖ EXPLANATION

The `math` module is imported to use the value of π (`math.pi`) for circle area calculation.

- ❖ A function `calculate_area()` is defined that takes a `shape_type` and a variable number of arguments using `*args`.
- ❖ The `shape_type` is converted to lowercase so the function works even if the user enters uppercase or mixed-case input.
- ❖ Based on the shape name (`circle`, `square`, `rectangle`, `triangle`), the function checks whether the correct number of arguments is provided.
- ❖ If the arguments are correct, it calculates and returns the area using the appropriate formula.
- ❖ If the arguments are incorrect or the shape is unknown, the function returns a clear error message explaining what is wrong.

❖

Task 3: Prompt Sensitivity Experiment

❖ **Scenario:**
You are testing how AI responds to different prompts.

❖ **Task:**
Use Cursor AI with different prompts for the same problem and observe code changes.

❖ **Expected Output:**

- **Prompt list**
- **Code variations**
- **OUTPUT**
- **PROMPT :** generate python code Given a list of numbers, calculate the sum of even numbers and the sum of odd numbers separately.(USING CURSOR AI)
- CODE

```
def calculate_sums(numbers):
    """
    Calculate the sum of even numbers and the sum of odd numbers separately.

    Args:
        numbers: A list of numbers

    Returns:
        A tuple containing (sum_of_evens, sum_of_odds)
    """
    sum_even = 0
    sum_odd = 0

    for num in numbers:
        if num % 2 == 0:
            sum_even += num
        else:
            sum_odd += num

    return sum_even, sum_odd


def main():
    # Example usage
    numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    sum_even, sum_odd = calculate_sums(numbers)

    print(f"List of numbers: {numbers}")
    print(f"Sum of even numbers: {sum_even}")
    print(f"Sum of odd numbers: {sum_odd}")


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

- CODE OUTPUT :

```
$ python sum_even_odd.py
List of numbers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Sum of even numbers: 30
Sum of odd numbers: 25
```

- EXPLANATION :
- This program calculates the **sum of even numbers** and **sum of odd numbers** from a given list.

- The list used is:
  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- The program checks **each number** in the list:
  - If the number is divisible by 2, it is treated as **even** and added to the even sum.
  - Otherwise, it is treated as **odd** and added to the odd sum.
- **Even numbers** in the list are: 2, 4, 6, 8, 10
  Their sum is: **30**
- **Odd numbers** in the list are: 1, 3, 5, 7, 9
  Their sum is: **25**
  - Finally, the program prints the list along with the calculated sums of even and odd numbers as shown in the output.

Task 4: Tool Comparison Reflection

- **Scenario:**
  You must recommend an AI coding tool.

- **Task:**
  Based on your work in this topic, compare Gemini, Copilot, and Cursor AI for usability and code quality.

- **Expected Output:**
  Short written reflection

OUTPUT
PROMPT : COMPARING THE CODE (USING GEMINI AND CURSOR AI)

COMPARISON TABLE (USING GEMINI AND CURSOR AI)

Feature	Gemini (Google Colab)	Cursor AI
Code Understanding	Clearly understands the logic and purpose of the program	Quickly understands the intent from exist code
Refactoring Style	Focuses on readability and simplicity	Focuses on optimization and concise synt
Suggested Improvement	Uses Pythonic constructs like generator expressions with `sum()`	Suggests optimized one-liners or functio style code
Code Explanation	Provides detailed, beginner-friendly explanations	Provides short and technical explanations
Ease of Use	Easy to use via prompts in Google Colab	Integrated directly into the code editor
Output Accuracy	Produces correct and clean output	Produces correct and efficient output
Best For	Beginners and learning concepts	Developers focusing on speed and refactoring

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