AI ASSISTANT CODING

ASSIGNMENT 2.5

Name:Kashaboina.Archana

HT.No: 2303A51329

Batch: 20

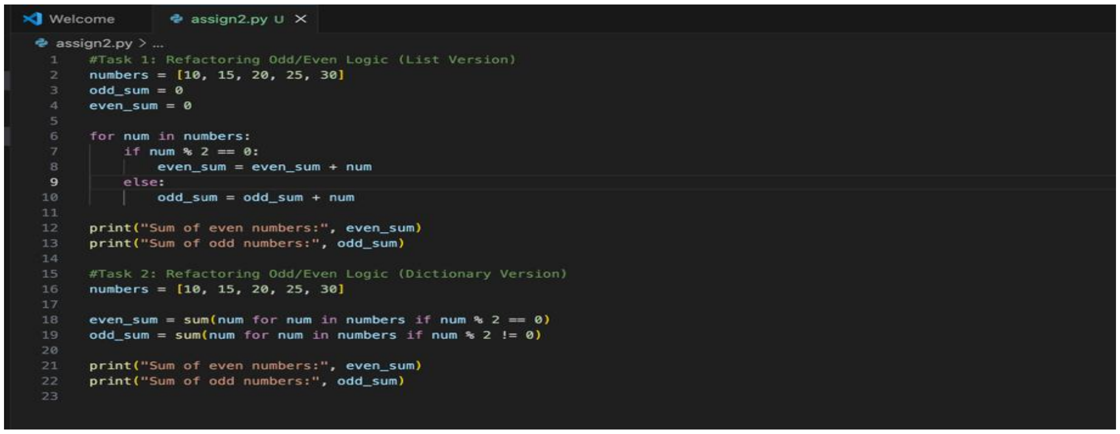
Task 1: Task 1: Refactoring Odd/Even Logic (List Version) Write a program to calculate the sum of odd and even numbers in a list, then refactor it using AI.

Scenario

You are improving legacy code.

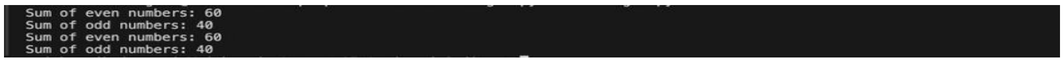
Prompt: calculate the sum of odd and even numbers and refactor it using AI.

Code:



```python
#Task 1: Refactoring Odd/Even Logic (List Version)
numbers = [10, 15, 20, 25, 30]
odd_sum = 0
even_sum = 0

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

print("Sum of even numbers:", even_sum)
print("Sum of odd numbers:", odd_sum)

#Task 2: Refactoring Odd/Even Logic (Dictionary Version)
numbers = [10, 15, 20, 25, 30]

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("Sum of even numbers:", even_sum)
print("Sum of odd numbers:", odd_sum)
```

Result:



```
Sum of even numbers: 60
Sum of odd numbers: 40
Sum of even numbers: 60
Sum of odd numbers: 40
```

Observation:

The refactored version removes manual looping and conditional accumulation, making the code shorter, more readable, and efficient while producing the same output.

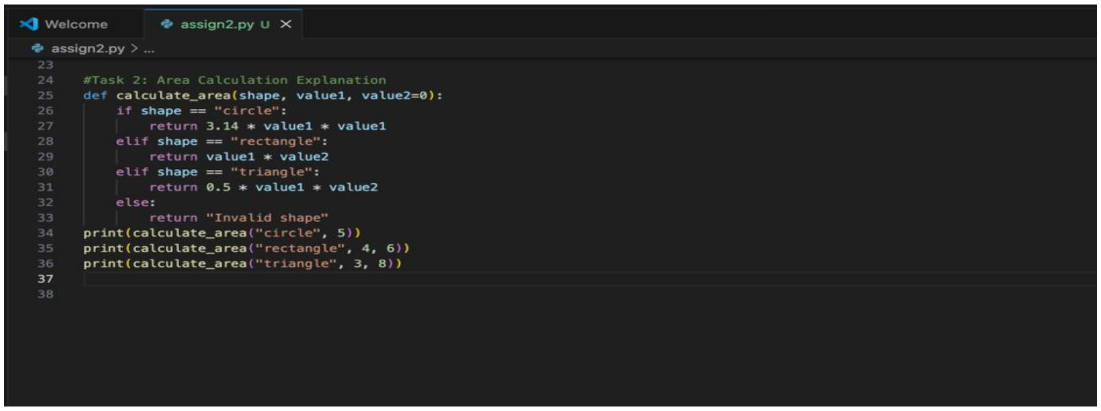
Task 2: Area Calculation Explanation. Ask Gemini to explain a function that calculates the area of different shapes.

Scenario

You are onboarding a junior developer.

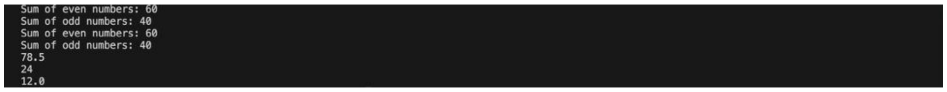
Prompt: give a function that calculates the area of different shapes

Code:



```python
#Task 2: Area Calculation Explanation
def calculate_area(shape, value1, value2=0):
    if shape == "circle":
        return 3.14 * value1 * value1
    elif shape == "rectangle":
        return value1 * value2
    elif shape == "triangle":
        return 0.5 * value1 * value2
    else:
        return "Invalid shape"
print(calculate_area("circle", 5))
print(calculate_area("rectangle", 4, 6))
print(calculate_area("triangle", 3, 8))
```

Result:



```
Sum of even numbers: 60
Sum of odd numbers: 40
Sum of even numbers: 60
Sum of odd numbers: 40
78.5
24
12.0
```

Observation:

Gemini effectively explains both the logic and mathematical reasoning in a clear and structured way, making it suitable for junior developers and beginners.

Task 3: Prompt Sensitivity Experiment Use Cursor AI with different prompts for the same problem and observe code changes.

Scenario

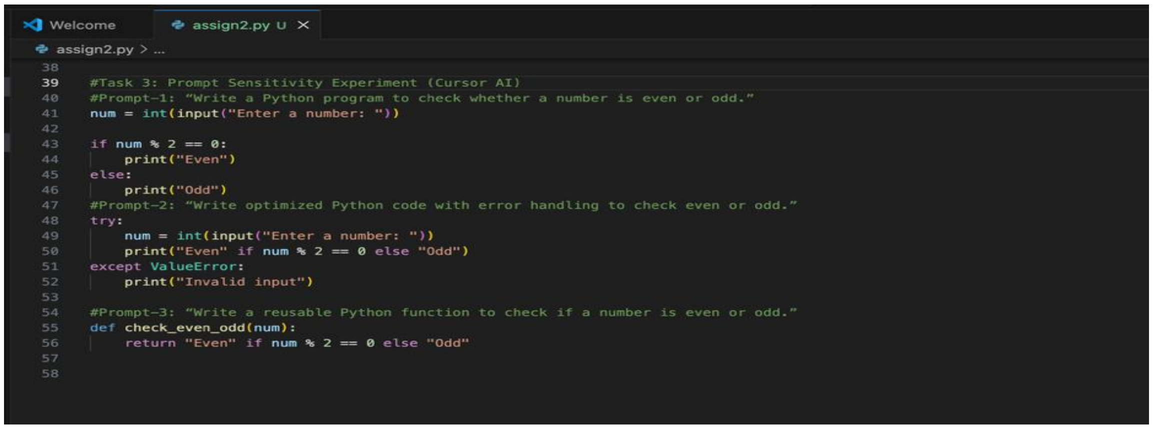
You are testing how AI responds to different prompts.

Prompt-1: Write a Python program to check whether a number is even or odd

Prompt-2: Write optimized Python code with error handling to check even or odd

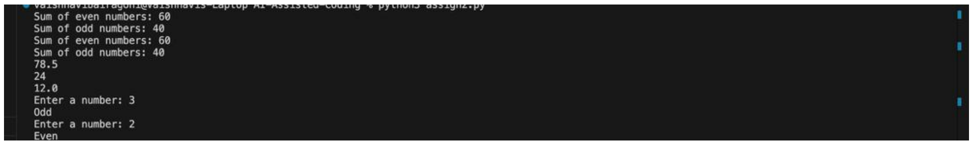
Prompt-3: Write a reusable Python function to check if a number is even or odd

Code:



```python
#Task 3: Prompt Sensitivity Experiment (Cursor AI)
#Prompt-1: "Write a Python program to check whether a number is even or odd."
num = int(input("Enter a number: "))

if num % 2 == 0:
    print("Even")
else:
    print("Odd")
#Prompt-2: "Write optimized Python code with error handling to check even or odd."
try:
    num = int(input("Enter a number: "))
    print("Even" if num % 2 == 0 else "Odd")
except ValueError:
    print("Invalid input")

#Prompt-3: "Write a reusable Python function to check if a number is even or odd."
def check_even_odd(num):
    return "Even" if num % 2 == 0 else "Odd"
```

Result:



```
vaishnavibairagoni@Vaishnavis-Laptop AI-Assisted-Coding % python3 assign2.py
Sum of even numbers: 60
Sum of odd numbers: 40
Sum of even numbers: 60
Sum of odd numbers: 40
78.5
24
12.0
Enter a number: 3
Odd
Enter a number: 2
Even
```

Observation:

Changing the prompt directly affected the **structure and quality** of the generated code. Simple prompts resulted in basic logic. Optimization prompts introduced **error handling** and concise syntax. Reusability prompts led to **function-based design**.

Task 4: Tool Comparison Reflection. Based on your work in this topic, compare Gemini, Copilot, and Cursor AI for usability and code quality.

Scenario

You must recommend an AI coding tool.

Prompt: Compare Google Gemini, GitHub Copilot, and Cursor AI in terms of usability and code quality. Highlight their strengths and limitations, and recommend the most suitable tool for regular software development

Observation:

Tool Comparison

Google Gemini (Google Colab)
Google Gemini is highly effective for explaining programming concepts and providing detailed code explanations. It is particularly useful for beginners and junior developers who require conceptual clarity. Gemini performs well in educational scenarios such as onboarding and documentation. However, it lacks direct integration with development environments, making it less suitable for real-time coding.

GitHub Copilot
GitHub Copilot offers seamless integration with Visual Studio Code and provides real-time code suggestions as developers write code. It significantly improves coding speed and productivity while maintaining good code quality. Copilot is best suited for regular development work and supports modern programming practices. Its IDE-level assistance makes it the most practical tool for professional software development.

Cursor AI
Cursor AI is effective for refactoring, code optimization, and prompt-based experimentation. It responds well to detailed prompts and helps improve code readability and structure. Cursor AI is particularly useful during code reviews and when working with existing codebases. However, it may not be as beginner-friendly as Gemini.

Recommendation

Based on usability and code quality, GitHub Copilot is recommended as the primary AI-assisted coding tool. It provides the best balance between productivity, integration, and code generation quality. Gemini and Cursor AI serve as valuable supplementary tools for learning and optimization purposes respectively.

Conclusion

Each AI coding tool has distinct strengths. Selecting the appropriate tool depends on the development context. For learning and explanation, Gemini is preferred; for active development, Copilot is ideal; and for refactoring and analysis, Cursor AI is effective.