School of Computer Science and Artificial Intelligence

Lab Assignment # 2.5

Program : B. Tech (CSE)

Specialization :AIML

Course Title :AI Assisted Coding

Course Code:23CS002PC304

Semester : VI

Academic Session : 2025-2026

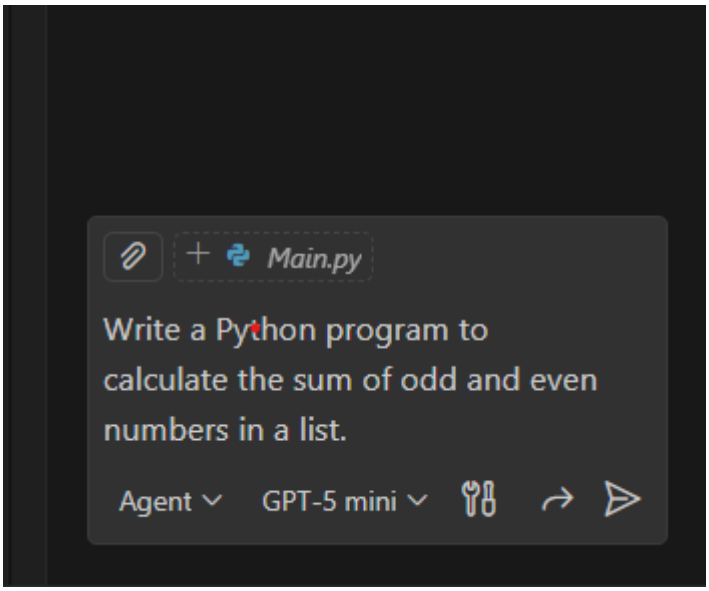
Name of Student :P.Sushathi

Enrollment No. : 2303A52102

Batch No. : 33

Date :09/01/26

TASK-1



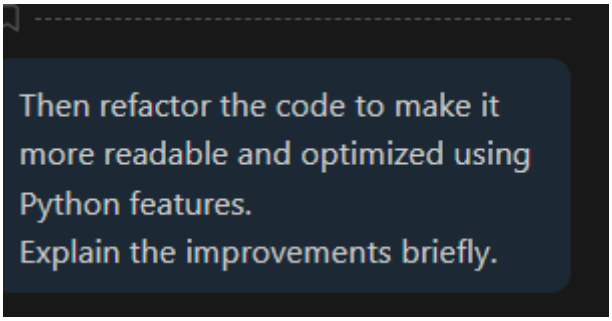
Main.py > ...

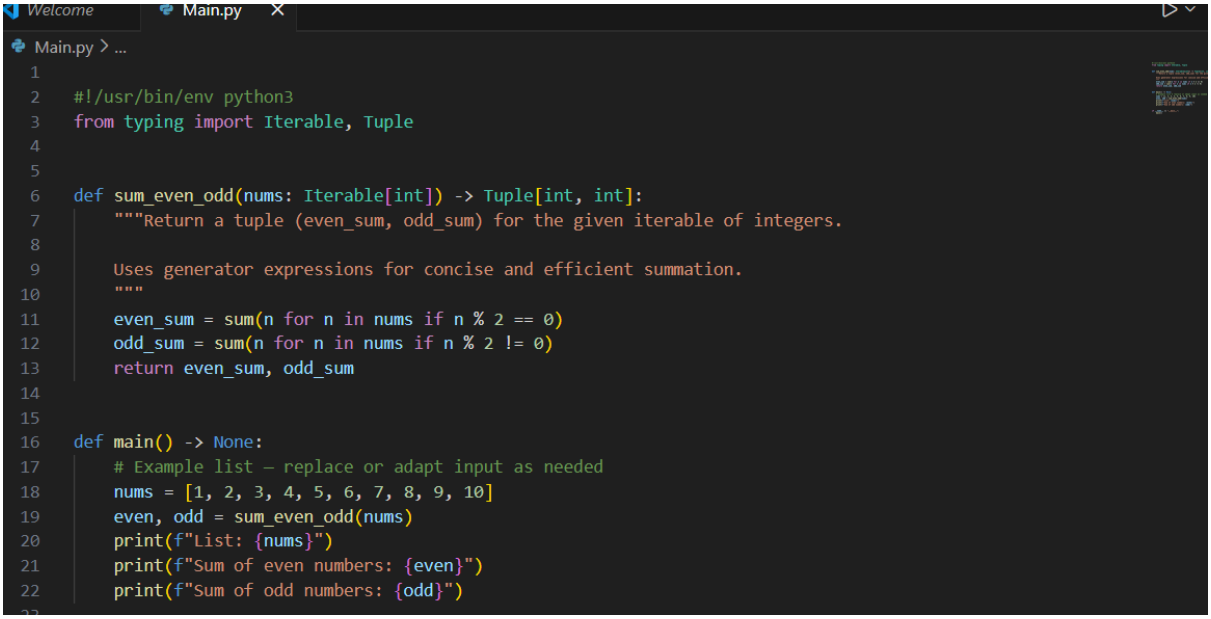
```python
#!/usr/bin/env python3

def sum_even_odd(nums):
    """Return a tuple (even_sum, odd_sum) for the given list of integers."""
    even_sum = 0
    odd_sum = 0
    for n in nums:
        if n % 2 == 0:
            even_sum += n
        else:
            odd_sum += n
    return even_sum, odd_sum


def main():
    # Example list — replace or adapt input as needed
    nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    even, odd = sum_even_odd(nums)
    print("List:", nums)
    print("Sum of even numbers:", even)
    print("Sum of odd numbers:", odd)


if __name__ == "__main__":
    main()

```

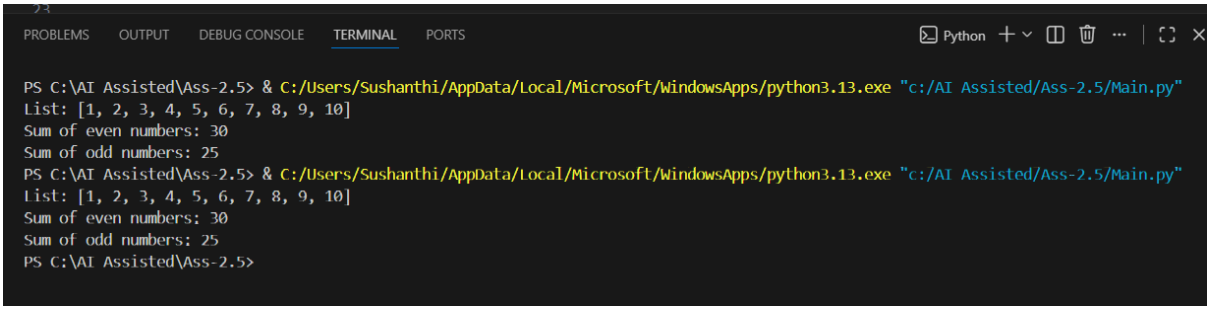
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
PS C:\AI Assisted\Ass-2.5> & C:/Users/Sushanthi/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/AI Assisted/Ass-2.5/Main.py"
List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Sum of even numbers: 30
Sum of odd numbers: 25
PS C:\AI Assisted\Ass-2.5>
```

Refactoring Prompt

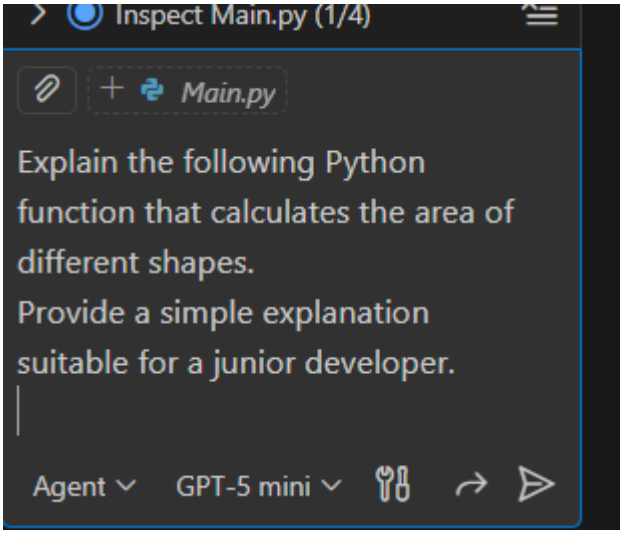






AI reduced the code size, removed loops and improved readability.

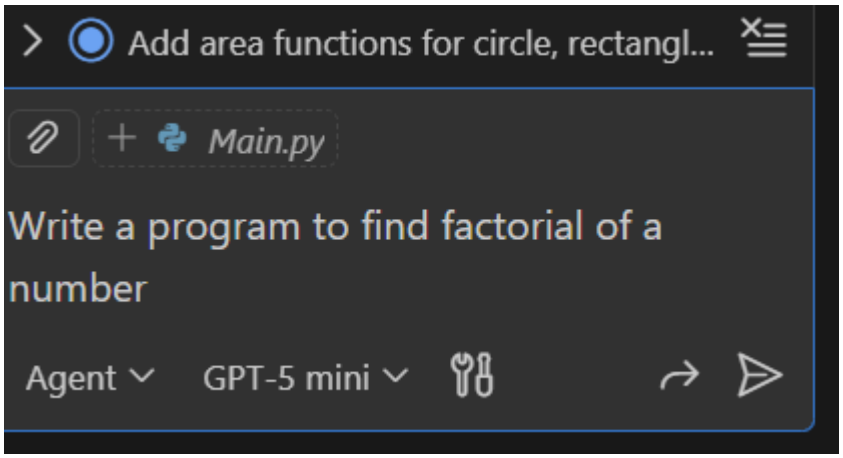
TASK-2




```python
from math import pi
def area_circle(radius):
    """Return area of a circle given its radius.

    Uses formula: area = pi * r^2
    """
    return pi * (radius ** 2)


def area_rectangle(width, height):
    """Return area of a rectangle (width * height)."""
    return width * height


def area_triangle(base, height):
    """Return area of a triangle using (base * height) / 2."""
    return 0.5 * base * height


def _demo():
    c = area_circle(3)
    r = area_rectangle(4, 5)
    t = area_triangle(6, 7)
    print(f"Circle area (r=3): {c}")
    print(f"Rectangle area (4x5): {r}")
    print(f"Triangle area (base=6,height=7): {t}")


if __name__ == "__main__":
```

TASK-3

Prompt 1



Main.py > main

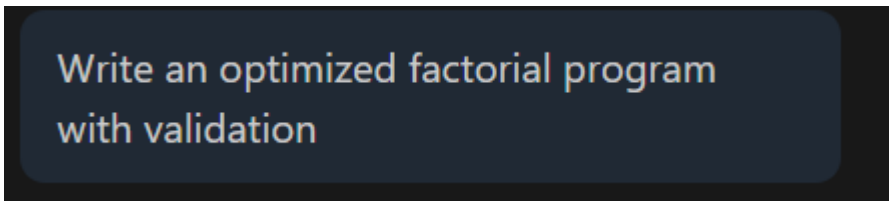
```
def factorial(n: int) -> int:
    if n < 0:
        raise ValueError("factorial() not defined for negative values")
    return math.factorial(n)


def main() -> None:
    # Example list — replace or adapt input as needed
    nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    even, odd = sum_even_odd(nums)
    print(f"List: {nums}")
    print(f"Sum of even numbers: {even}")
    print(f"Sum of odd numbers: {odd}")

    # Factorial: if a number is passed as a CLI argument, compute that.
    # Otherwise, show a small demo of factorials.
    if len(sys.argv) > 1:
        try:
            n = int(sys.argv[1])
        except ValueError:
            print("Please provide an integer as the first argument.")
            return
        try:
            print(f"\nFactorial of {n}: {factorial(n)}")
        except ValueError as e:
            print(e)
    else:
        sample = 5
        print("\nSample factorials (no CLI arg provided):")
        for i in range(0, sample + 1):
            print(f"{i}! = {factorial(i)}")
        print("\nTo compute a specific factorial, run:")
        print("  python Main.py 7")


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

Ln 50, Col 19 Space

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

Sample factorials (no CLI arg provided):
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120

To compute a specific factorial, run:
  python Main.py 7
PS C:\AI Assisted\Ass-2.5>
```

Prompt 2




```python
import math
def factorial(n):
    """Return n! for a non-negative integer n.

    Optimization: delegates computation to `math.factorial` (C-optimized).

    Validation/acceptance rules:
    - Accepts `int`, `float` that is an integer (e.g. 5.0), or numeric `str` like '5' or '5.0'.
    - Raises `ValueError` for non-integer, negative, or otherwise invalid inputs.
    """
    # Try to coerce common input types to an integer, with validation
    try:
        if isinstance(n, str):
            s = n.strip()
            if s == "":
                raise ValueError
            # allow '5', '5.0', '  5 ' and scientific notation like '1e2'
            f = float(s)
            if not f.is_integer():
                raise ValueError("Factorial requires an integer input")
            n_int = int(f)
        elif isinstance(n, float):
            if not n.is_integer():
                raise ValueError("Factorial requires an integer input")
            n_int = int(n)
        elif isinstance(n, int):
```

```
            raise ValueError("Factorial requires an integer input")
        n_int = int(n)
    elif isinstance(n, int):
        n_int = n
    else:
        # try a best-effort conversion for numeric-like objects
        n_int = int(n)
        if float(n) != float(n_int):
            raise ValueError("Factorial requires an integer input")
except (ValueError, TypeError):
    raise ValueError("Factorial requires an integer-like non-negative input")

if n_int < 0:
    raise ValueError("Factorial is not defined for negative numbers")

# Use math.factorial for best performance and big-integer support
return math.factorial(n_int)


if __name__ == "__main__":
    try:
        s = input("Enter a non-negative integer to compute factorial (or press Enter to show exa
        if not s.strip():
            print("5! =", factorial(5))
        else:
            print(f"{s.strip()}! =", factorial(s.strip()))
    except ValueError as e:
        print("Invalid input:", e)
```

```
Enter a non-negative integer to compute factorial (or press Enter to show example 5!): 8
8! = 40320
PS S:\3 2 sem\AIAssisted_2026>
```

Observation

Different prompts produced different code structures, logic and features.

Task 4: Tool Comparison Reflection

Scenario:

You must recommend an AI coding tool.

Task:

Based on the work done in this lab, Gemini, Copilot, and Cursor AI were compared in terms of usability and code quality.

Short Written Reflection:

After using all three AI tools, it was observed that each tool serves a different purpose.
Google Gemini is very useful for learning and understanding concepts because it provides detailed explanations and well-structured code. It is best suited for beginners and documentation work.

GitHub Copilot is excellent for fast coding. It suggests code directly inside the IDE, which saves time and improves productivity. However, it does not provide much explanation, so it is better suited for experienced programmers.

Cursor AI performs very well in code refactoring and optimization. It allows users to experiment with different prompts and observe changes in the code, making it useful for improving code quality and testing different approaches.

Final Recommendation:

For learning and concept clarity — **Gemini**
For fast development — **Copilot**
For code improvement and experimentation — **Cursor AI**