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 - Monday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	M. Tech/MCA
AssignmentNumber:1.3(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	Lab 1: Environment Setup – GitHub Copilot and VS Code Integration **Lab Objectives:** ● To install and configure GitHub Copilot in Visual Studio Code. ● To explore AI-assisted code generation using GitHub Copilot. ● To analyze the accuracy and effectiveness of Copilot's code suggestions. ● To understand prompt-based programming using comments and code context **Lab Outcomes (LOs):** After completing this lab, students will be able to: ● Set up GitHub Copilot in VS Code successfully. ● Use inline comments and context to generate code with Copilot. ● Evaluate AI-generated code for correctness and readability. ● Compare code suggestions based on different prompts and programming styles. **Task Description#1** ● Install and configure GitHub Copilot in VS Code. Take screenshots of each step. **Expected Output#1** ● Install and configure GitHub Copilot in VS Code. Take screenshots of each step.	Week1 - Wednesday

Task Description#2

- Use Copilot to generate a is_prime() Python function**.**

Prompt : write a python function to check whether the user input number is prime or not

Expected Output#2

- Function to check primality with correct logic.

```python
import math

def is_prime(n: int) -> bool:
    """Return True if n is a prime number, otherwise False."""
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0:
        return False
    limit = math.isqrt(n)
    for i in range(3, limit + 1, 2):
        if n % i == 0:
            return False
    return True

if __name__ == "__main__":
    try:
        s = input("Enter an integer (or 'q' to quit): ").strip()
        if s.lower() == 'q':
            print("Exiting.")
        else:
            num = int(s)
            print(f"{num} is {'a prime' if is_prime(num) else 'not a prime'} number.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Largest (iterative): 876.0
Largest (builtin)   : 876.0
PS C:\Users\91832\OneDrive\Documents\Desktop\AI Assignments> cd "c:/
PS C:\Users\91832\OneDrive\Documents\Desktop\AI Assignments> & "C:/P
/AI Assignments/task1.py"
Ent                                                          76
76
PS C:\Users\91832\OneDrive\Documents\Desktop\AI Assignments>
```

Focus folder in explorer (ctrl + click)

Task Description#3

- Write a comment like # Function to reverse a string and use Copilot to generate the function.

Prompt : write a python function to reverse the user input string

```python
def reverse_string(s: str) -> str:
    """Return the reversed version of the input string."""
    return s[::-1]

if __name__ == "__main__":
    try:
        user_input = input("Enter a string: ")
    except (EOFError, KeyboardInterrupt):
        print("\nNo input provided.")
    else:
        print(reverse_string(user_input))
```

Expected Output#3

- Auto-completed reverse function

```
 Assignments/task2.py"
Enter a string: fathima
amihtaf
PS C:\Users\91832\OneDrive\Documents\Desktop\AI Assignments>
```

Task Description#4

- Generate both recursive and iterative versions of a factorial function using comments

Prompt : **write a python code for user input recursive and iterative function of a factorial function using comment**

..

```python
def factorial_recursive(n: int) -> int:
    """Recursive factorial.
    - Base case: 0! = 1 and 1! = 1
    - Recursive step: n! = n * (n-1)!
    Raises ValueError for negative inputs.
    """
    if n < 0:
        raise ValueError("Negative input not allowed")
    if n <= 1:  # base case
        return 1
    return n * factorial_recursive(n - 1)  # recursive call

def factorial_iterative(n: int) -> int:
    """Iterative factorial using a simple loop.
    - Handles 0! = 1 naturally by starting result = 1.
    Raises ValueError for negative inputs.
    """
    if n < 0:
        raise ValueError("Negative input not allowed")
    result = 1
    # multiply result by each integer from 2 to n
    for i in range(2, n + 1):
        result *= i
    return result

if __name__ == "__main__":
    try:
        s = input("Enter a non-negative integer (or 'q' to quit): ").strip()
        if s.lower() == 'q':
            print("Exiting.")
        else:
            n = int(s)
            if n < 0:
                raise ValueError
            print(f"Iterative factorial of {n} is: {factorial_iterative(n)}")
            print(f"Recursive factorial of {n} is: {factorial_recursive(n)}")
    except ValueError:
        print("Invalid input. Please enter a non-negative integer.")
```

Expected Output#4

- Two working factorial implementations

```
 Assignments/Task3.py"
Enter a non-negative integer (or 'q' to quit): 56
Iterative factorial of 56 is: 710998587804863451854045647463724949736497978881168458687447040000000000000
Recursive factorial of 56 is: 710998587804863451854045647463724949736497978881168458687447040000000000000
PS C:\Users\91832\OneDrive\Documents\Desktop\AI Assignments>
```

Task Description#5

- Use Copilot to find the largest number in a list. Assess code quality and efficiency

Prompt : **write a user input python code to find largest number in the list and assess the code quality and efficiency**

```python
# Simple CLI that reads a list of numbers from the user and prints the largest value.
from typing import List

def max_in_list_iterative(nums: List[float]) -> float:
    """Return the largest number using an O(n) iterative scan. Raises ValueError for empty list."""
    if not nums:
        raise ValueError("empty list")
    max_val = nums[0]
    for x in nums[1:]:
        if x > max_val:
            max_val = x
    return max_val

def max_in_list_builtin(nums: List[float]) -> float:
    """Return the largest number using Python's built-in max (O(n))."""
    if not nums:
        raise ValueError("empty list")
    return max(nums)

def parse_numbers(s: str) -> List[float]:
    """Parse comma/space-separated numbers into a list of floats. Raises ValueError on invalid tokens."""
    if not s:
        return []
    tokens = s.replace(",", " ").split()
    return [float(t) for t in tokens]

if __name__ == "__main__":
    try:
        s = input("Enter numbers separated by spaces or commas (or 'q' to quit): ").strip()
        if s.lower() == "q":
            print("Exiting.")
        else:
            nums = parse_numbers(s)
            if not nums:
                print("No numbers provided.")
            else:
                print("Largest (iterative):", max_in_list_iterative(nums))
                print("Largest (builtin)   :", max_in_list_builtin(nums))
    except ValueError:
        print("Invalid input. Please enter only numbers separated by spaces or commas.")
```

Expected Output#5

- A valid function with your review

```
PS C:\Users\91832\OneDrive\Documents\Desktop\AI Assignments> & "C:/Program Fi
/91832/OneDrive/Documents/Desktop/AI Assignments/Task4.py"
Enter numbers separated by spaces or commas (or 'q' to quit): 1 4 6 8 0
Largest (iterative): 8.0
Largest (builtin)   : 8.0
PS C:\Users\91832\OneDrive\Documents\Desktop\AI Assignments>
```

Assessment (brief)

- **Correctness: Handles typical numeric input, empty input, and invalid tokens (raises/handles ValueError).**
- **Time complexity: O(n) for both implementations (optimal for finding max).**
- **Space complexity: O(1) extra space for iterative version; parsing requires O(n) to store values.**
- **Readability: Clear functions with single responsibilities; parse helper improves robustness.**
- **Suggestions: Add unit tests, support integers exclusively if required, or accept streamed input for very large data to avoid storing the whole list.**

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 Setup of Copilot (Task #1)	2
is_prime() Python function (Task #2)	2
Reverse a string function (Task #3)	2

	Factorial Function (Task #4)	2		
	Find the largest number (Task #5)	2		
	Total	**10 Marks**		