

AI ASSISTED CODING

HT.NO:2403A52031

LAB-2

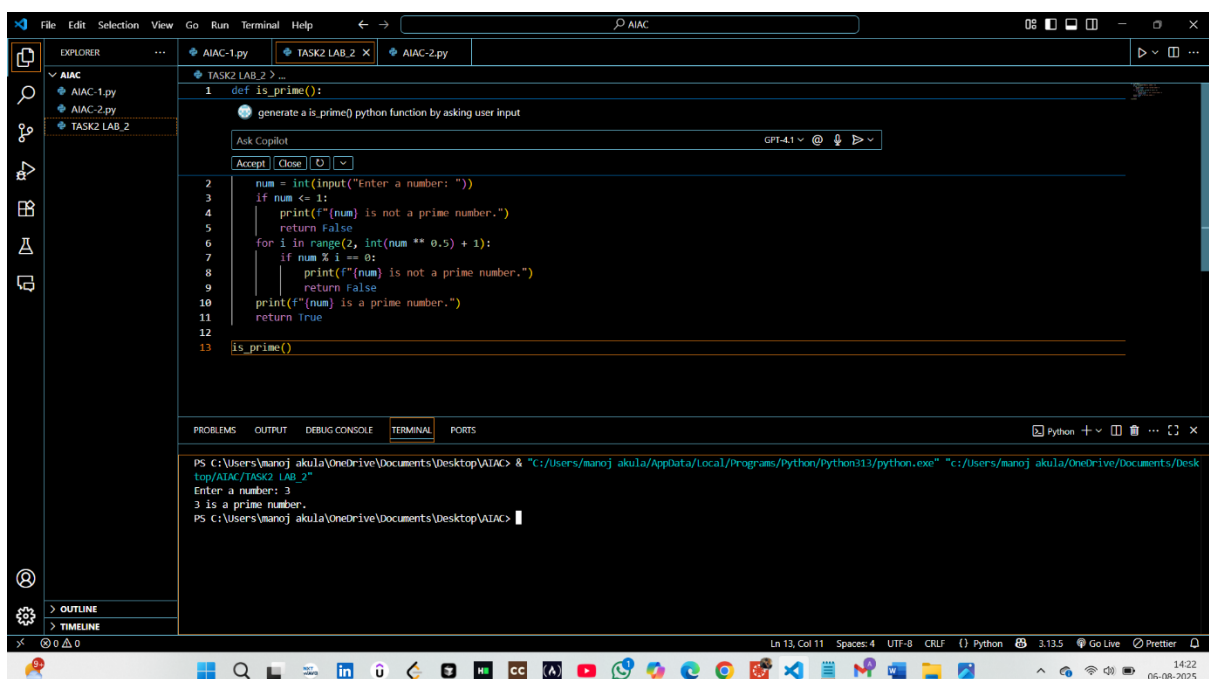
TASK DESCRIPTION 2:

USE THE CO-PILOT TO GENERATE A `is_prime` PYTHON FUNCTION

EXPECTED OUTPUT:

FUNCTION TO CHECK PRIMITY WITH CORRECT LOGIC.

PROMPT:generate a `is_prime()` python function by asking user input



```
File Edit Selection View Go Run Terminal Help
AIAC
TASK2 LAB_2
AIAC-1.py AIAC-2.py TASK2 LAB_2
TASK2 LAB_2 > ...
1 def is_prime():
    generate a is_prime() python function by asking user input
    Ask Copilot GPT-4.1
    Accept Close
2 num = int(input("Enter a number: "))
3 if num <= 1:
4     print(f"{num} is not a prime number.")
5     return False
6 for i in range(2, int(num ** 0.5) + 1):
7     if num % i == 0:
8         print(f"{num} is not a prime number.")
9         return False
10    print(f"{num} is a prime number.")
11    return True
12
13 is_prime()

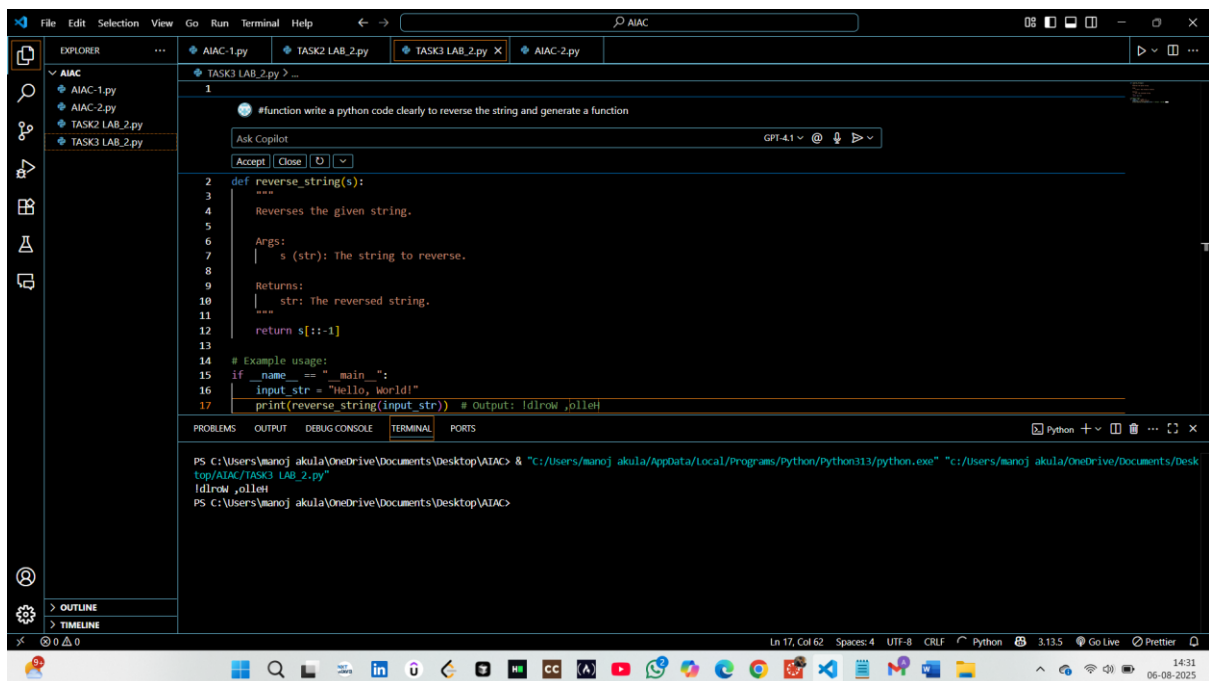
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
PS C:\Users\manoj_aku\OneDrive\Documents\Desktop\AIAC> & "C:\Users\manoj_aku\AppData\Local\Programs\Python\Python313\python.exe" "C:\Users\manoj_aku\OneDrive\Documents\Desktop\AIAC\TASK2 LAB_2"
Enter a number: 3
3 is a prime number.
PS C:\Users\manoj_aku\OneDrive\Documents\Desktop\AIAC>
```

TASK DESCRIPTION 3:

Write a comment like `#function` to reverse the string and use copilot to generate the function

EXPECTED OUTPUT 3: Auto-completed reverse function

PROMPT: `#function` write a python code clearly to reverse the string and generate a function



The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a terminal at the bottom. The main editor area displays a Python file named `TASK3 LAB_2.py`. The code defines a function `reverse_string(s)` that reverses a given string. The function is called in the `__main__` block with the input string "Hello, World!". The output of the function is displayed in the terminal as "dlrow ,olleH".

```
1 #function write a python code clearly to reverse the string and generate a function
2 def reverse_string(s):
3     """
4     Reverses the given string.
5
6     Args:
7         s (str): The string to reverse.
8
9     Returns:
10        str: The reversed string.
11    """
12    return s[::-1]
13
14 # Example usage:
15 if __name__ == "__main__":
16     input_str = "Hello, World!"
17     print(reverse_string(input_str)) # Output: dlrow ,olleH
```

The terminal output shows the command prompt running the script and the resulting reversed string:

```
PS C:\Users\manoj akula\OneDrive\Documents\Desktop\AIAC> & "C:/Users/manoj akula/AppData/Local/Programs/Python/Python313/python.exe" "C:/Users/manoj akula/OneDrive/Documents/Desktop/AIAC/TASK3 LAB_2.py"
dlrow ,olleH
PS C:\Users\manoj akula\OneDrive\Documents\Desktop\AIAC>
```

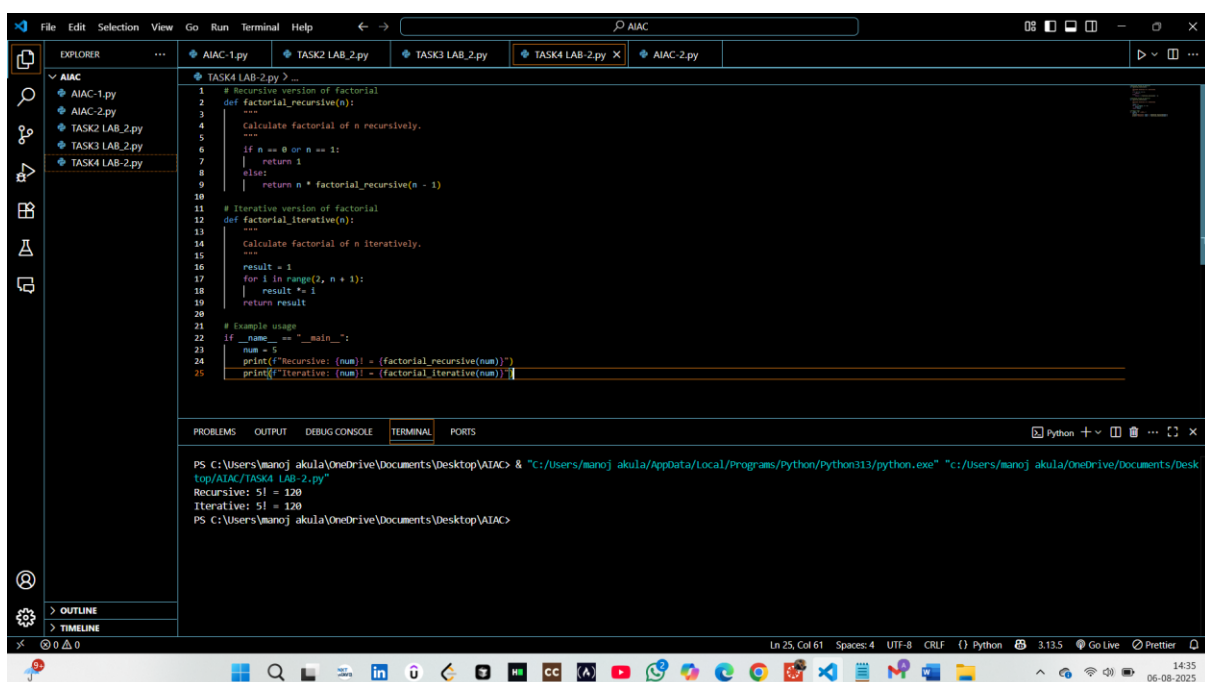
TASK DESCRIPTION 4:

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

EXPECTED OUTPUT 4:

Two working factorial implements

PROMPT: Generate a code in python that should both recursive and iterative versions of a factorial function using comments. Two working factorial implements.



The screenshot shows a Visual Studio Code editor window with a Python file named `TASK4 LAB-2.py`. The code defines two functions: `factorial_recursive(n)` and `factorial_iterative(n)`. The recursive function uses a base case of `n == 0 or n == 1` to return 1, and otherwise returns `n * factorial_recursive(n - 1)`. The iterative function uses a loop to calculate the factorial by multiplying the result by each number from 1 to `n`. The code also includes a main block that prompts the user for a number and prints the results of both functions.

```
1 # Recursive version of factorial
2 def factorial_recursive(n):
3     """
4     Calculate factorial of n recursively.
5     """
6     if n == 0 or n == 1:
7         return 1
8     else:
9         return n * factorial_recursive(n - 1)
10
11 # Iterative version of factorial
12 def factorial_iterative(n):
13     """
14     Calculate factorial of n iteratively.
15     """
16     result = 1
17     for i in range(2, n + 1):
18         result *= i
19     return result
20
21 # Example usage
22 if __name__ == "__main__":
23     num = 5
24     print(f'Recursive: {num}! = {factorial_recursive(num)}')
25     print(f'Iterative: {num}! = {factorial_iterative(num)}')
```

The terminal output shows the execution of the script, displaying the factorial of 5 using both recursive and iterative methods, both resulting in 120.

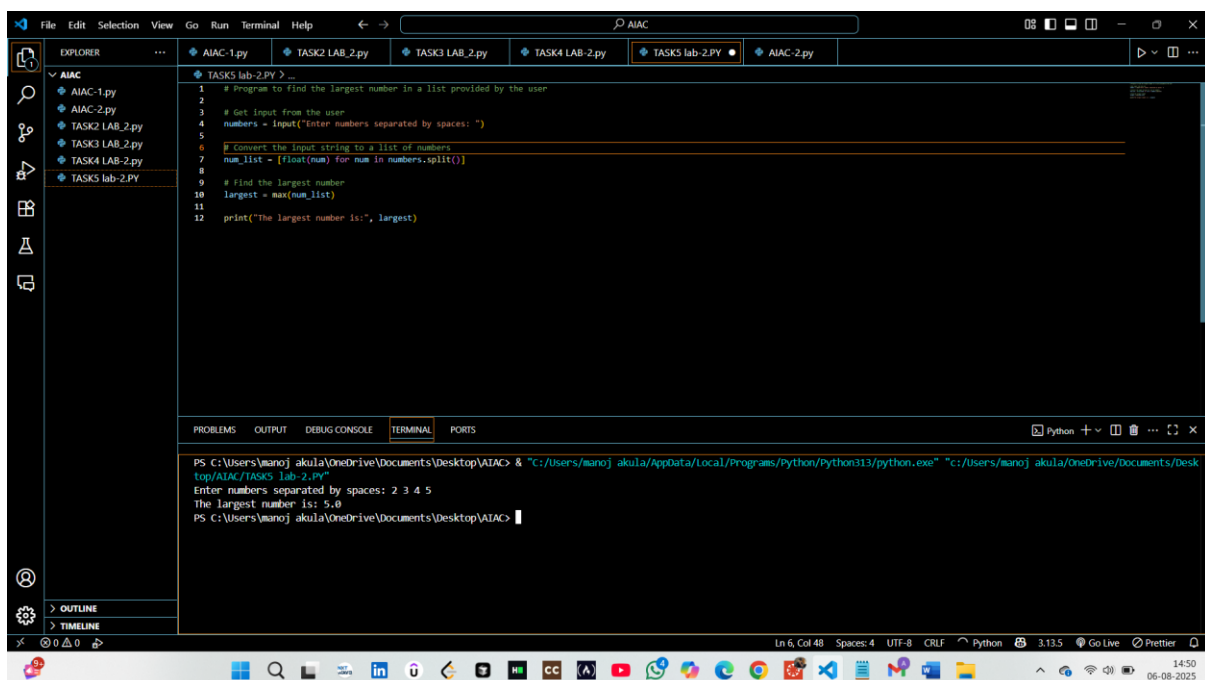
```
PS C:\Users\manoj_akula\OneDrive\Documents\Desktop\AIAC> & "c:/Users/manoj_akula/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/manoj_akula/OneDrive/Documents/Desktop/AIAC/TASK4 LAB-2.py"
Recursive: 5! = 120
Iterative: 5! = 120
PS C:\Users\manoj_akula\OneDrive\Documents\Desktop\AIAC>
```

TASK DESCRIPTION 5: Use copilot to find the largest number in a list .Access code quality and efficiency.

EXPECTED OUTPUT 5:

A valid function with your review

PROMPT: generate a python code to find the largest number among the given list as an input given by user.



The screenshot shows a Visual Studio Code editor window with a Python file named 'TASK5 lab-2.py'. The code is as follows:

```
1 # Program to find the largest number in a list provided by the user
2
3 # Get input from the user
4 numbers = input("Enter numbers separated by spaces: ")
5
6 # Convert the input string to a list of numbers
7 num_list = [float(num) for num in numbers.split()]
8
9 # Find the largest number
10 largest = max(num_list)
11
12 print("The largest number is:", largest)
```

The terminal output shows the execution of the script:

```
PS C:\Users\manoj akula\OneDrive\Documents\Desktop\AIAC> & "c:/Users/manoj akula/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/manoj akula/OneDrive/Documents/Desktop/AIAC/TASK5 lab-2.py"
Enter numbers separated by spaces: 2 3 4 5
The largest number is: 5.0
PS C:\Users\manoj akula\OneDrive\Documents\Desktop\AIAC>
```

CONCLUSION : I have learnt a lot about proper way of using github copilot and generating expected code through our prompts

THANK YOU