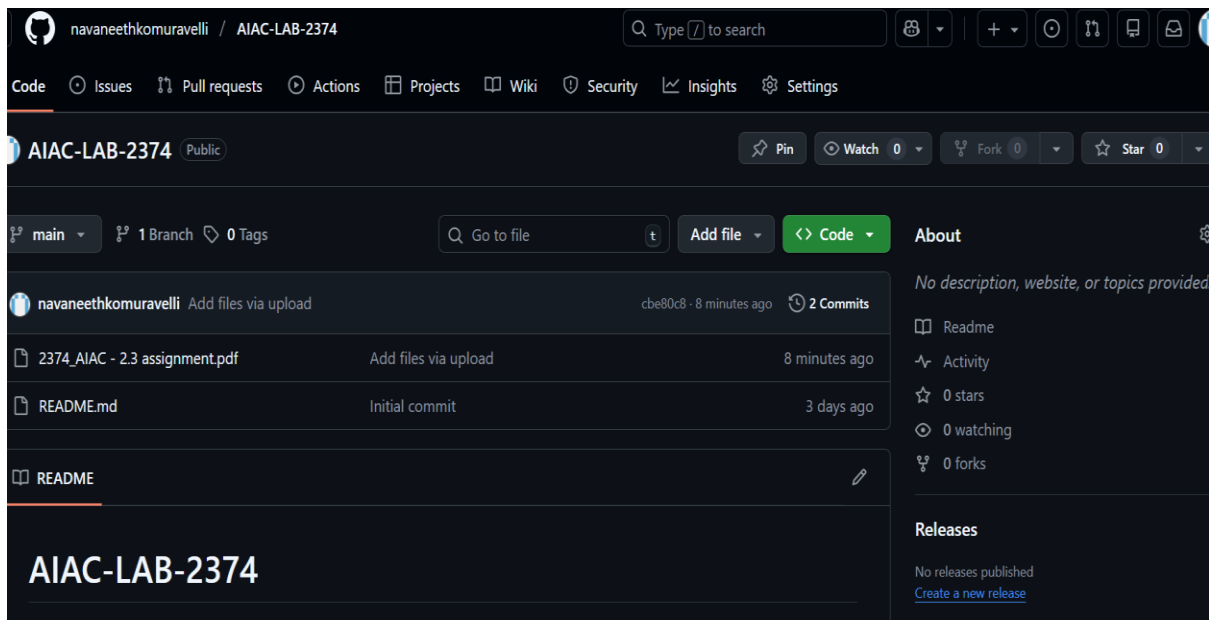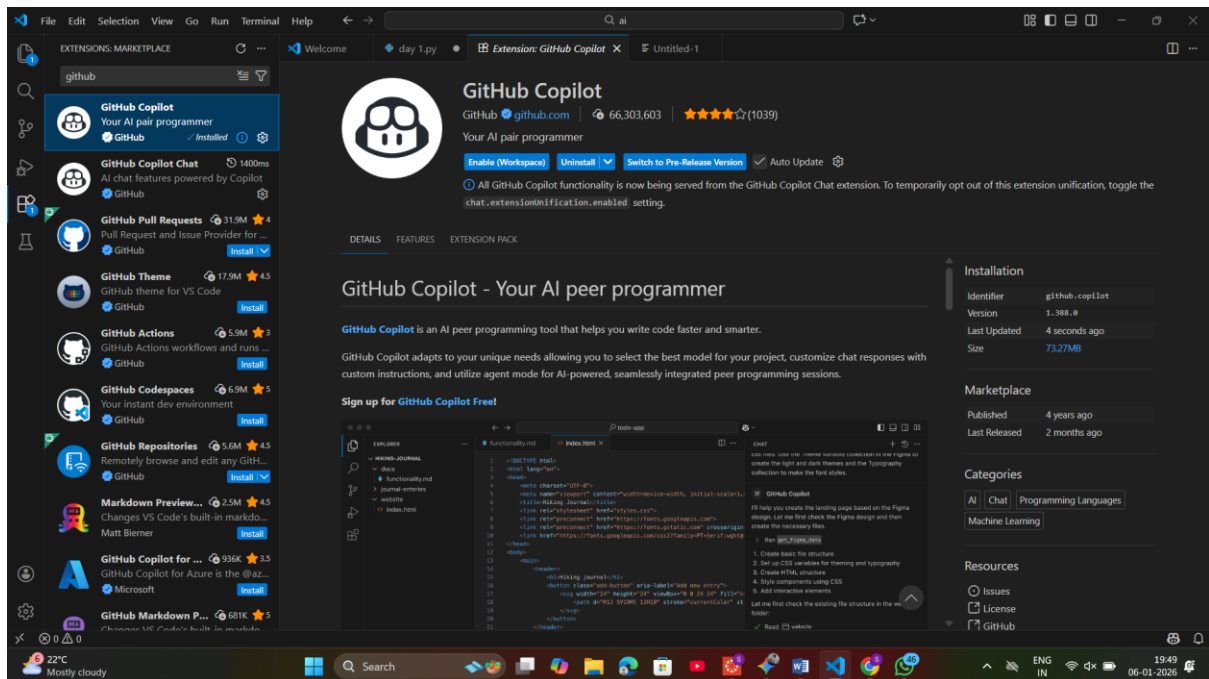K.Navaneeth

2303A5234

Batch - 32

AI Assisted Coding

Task 0: Environment Setup:-

Task 1: Non-Modular Logic (Factorial):-


```python
# Task 1: Procedural Factorial Implementation
num = int(input("Enter a number: "))
factorial = 1

if num < 0:
    print("Factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    temp = num
    while temp > 0:
        factorial *= temp
        temp -= 1
    print(f"The factorial of {num} is {factorial}")
```
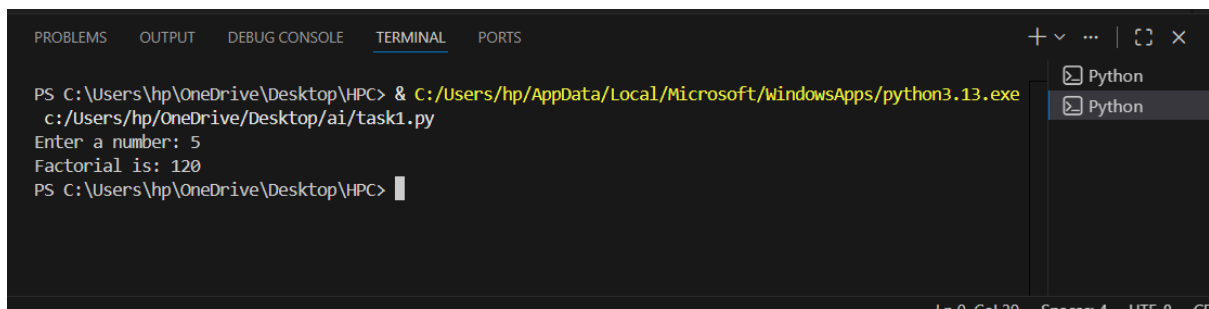


```
PS C:\Users\hp\OneDrive\Desktop\HPC> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.13.exe
 c:/Users/hp/OneDrive/Desktop/ai/task1.py
Enter a number: 5
Factorial is: 120
PS C:\Users\hp\OneDrive\Desktop\HPC>
```
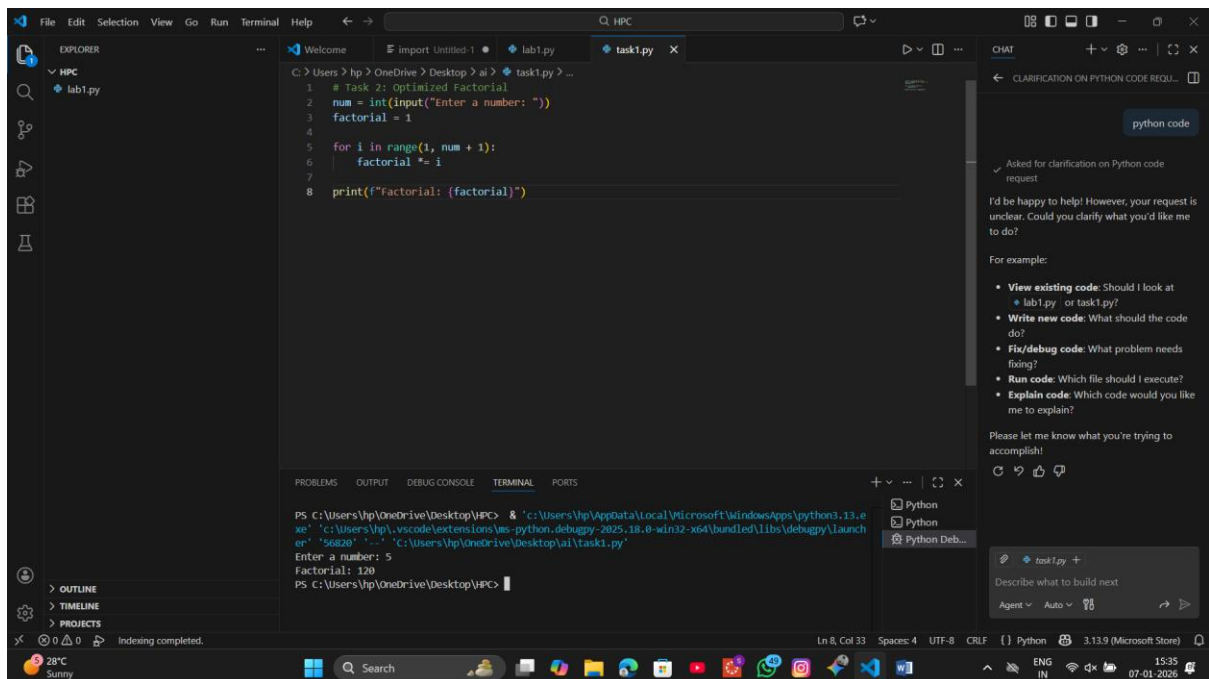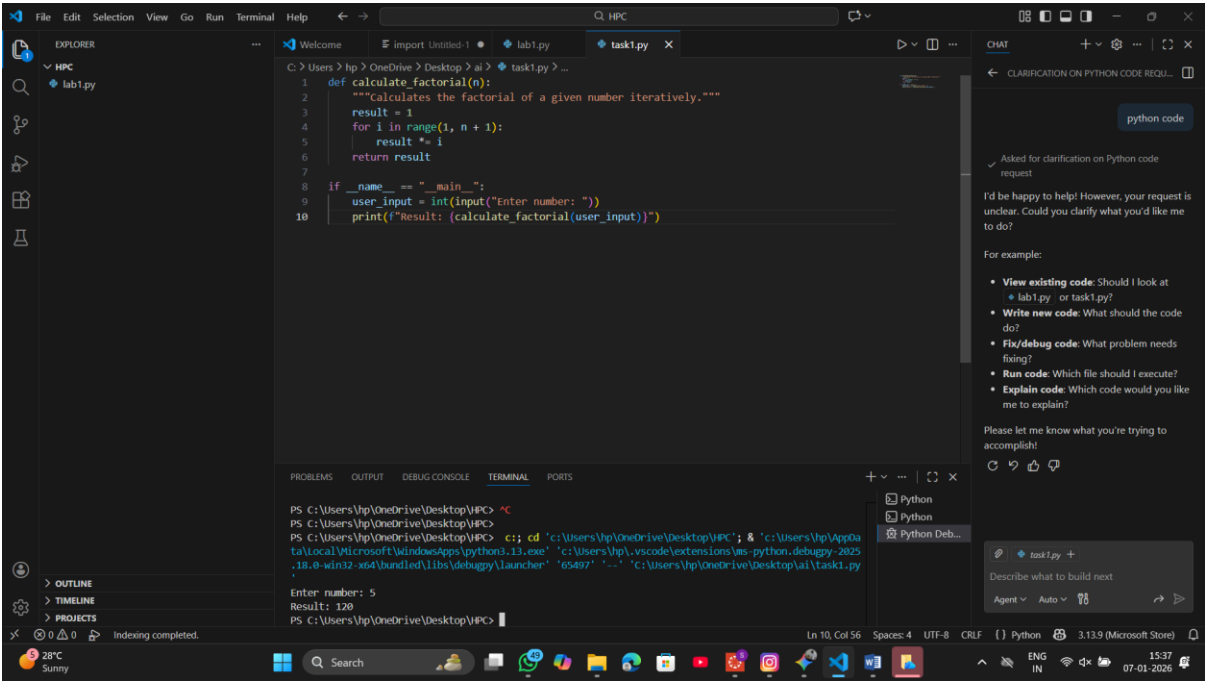
Task 2: AI Code Optimization:-


```python
# Task 2: Optimized Factorial
num = int(input("Enter a number: "))
factorial = 1

for i in range(1, num + 1):
    factorial *= i

print(f"Factorial: {factorial}")
```

Task 3: Modular Design:-

Task 4: Comparative Analysis:-

Criteria	Procedural (Task 1 & 2)	Modular (Task 3)
Logic Clarity	Linear and straightforward for very small tasks but becomes "spaghetti code" as complexity grows.	High clarity; the mathematical logic is isolated from the input/output logic.
Reusability	None. To use the logic elsewhere, the code must be manually copied and pasted.	High. The function can be imported into other Python files or called multiple times in one script.
Debugging Ease	Difficult. Errors in logic are mixed with errors in user input handling.	Simple. You can test the function with specific values (Unit Testing) to ensure the math is correct.
Project Suitability	Suitable only for small, one-off scripts or prototypes.	Essential for enterprise-level, large-scale software development.
AI Dependency Risk	High. AI might generate redundant variables or inefficient loops in long scripts.	Low. AI is highly specialized and accurate when asked to write specific, single-purpose functions.

Task 5: Iterative vs Recursive Thinking:-