

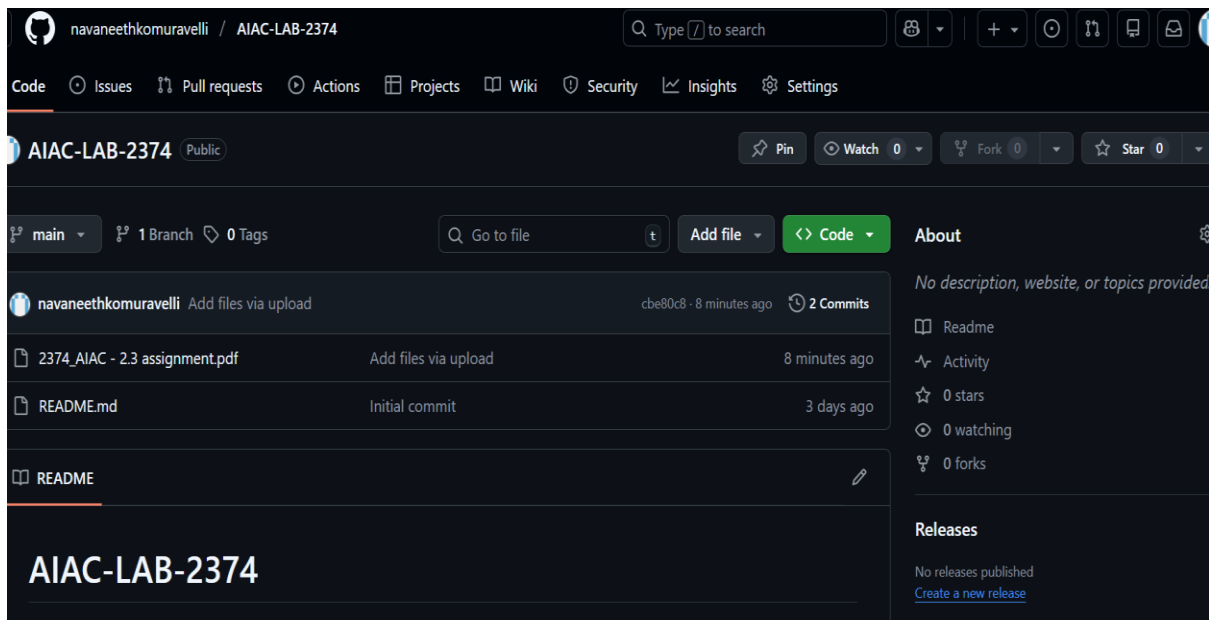
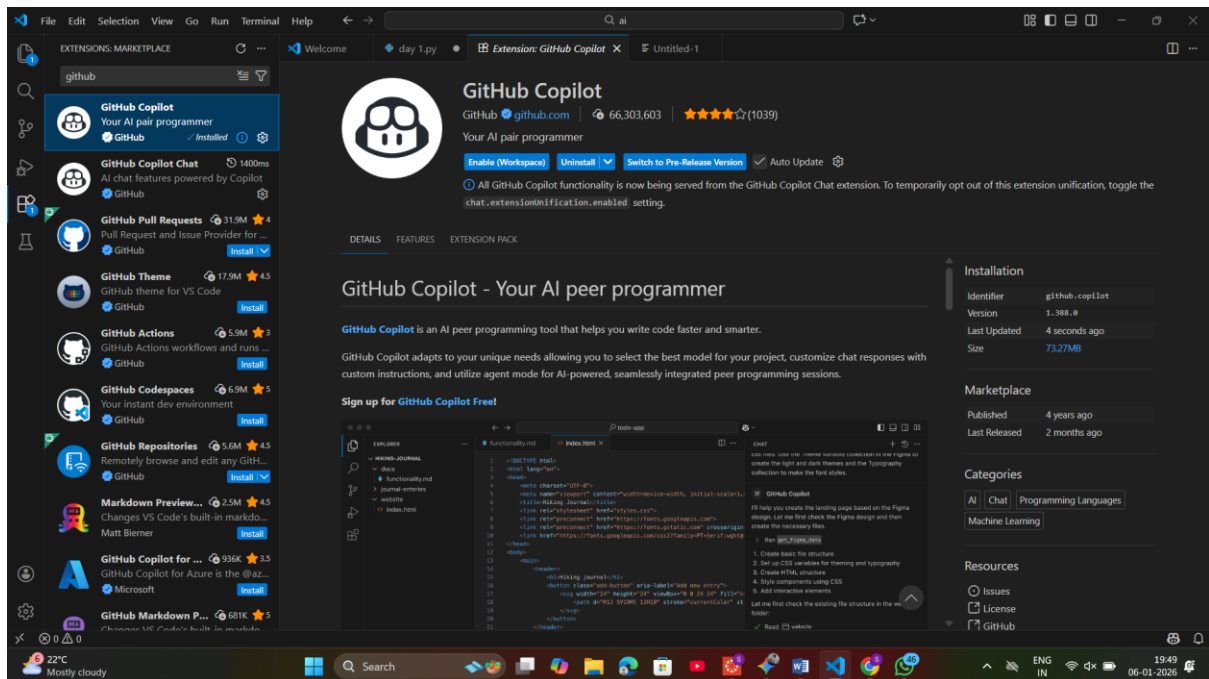
K.Navaneeth

2303A5234

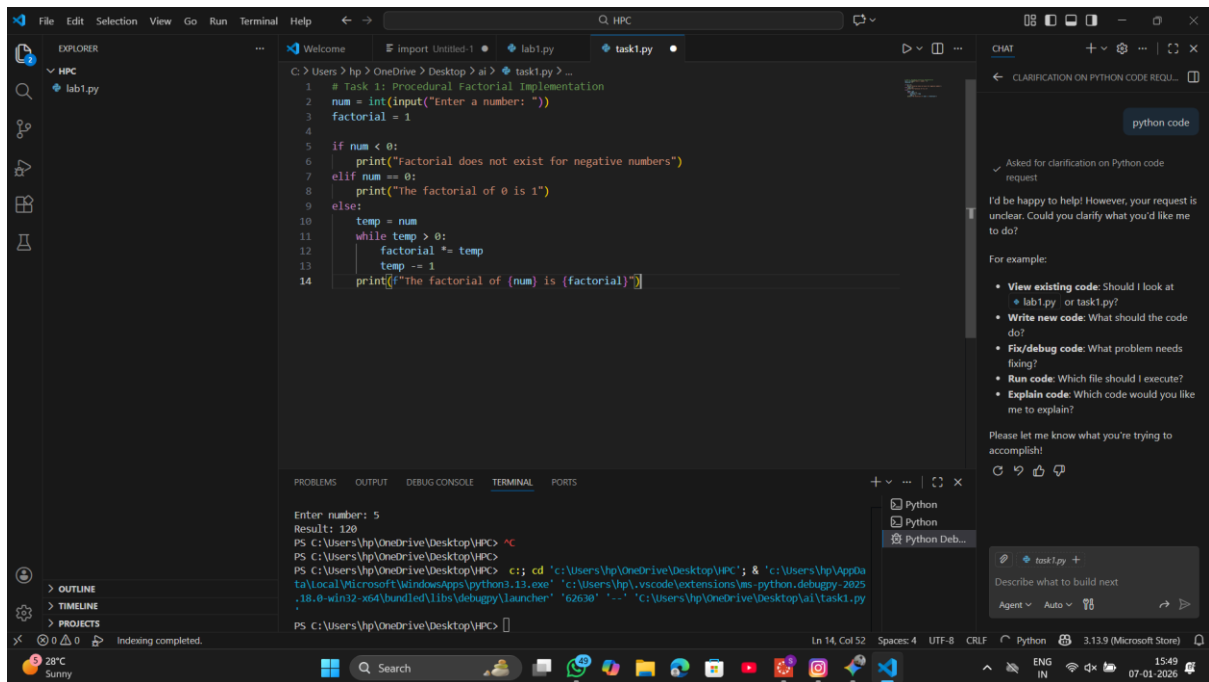
Batch - 32

AI Assisted Coding

Task 0: Environment Setup:-



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

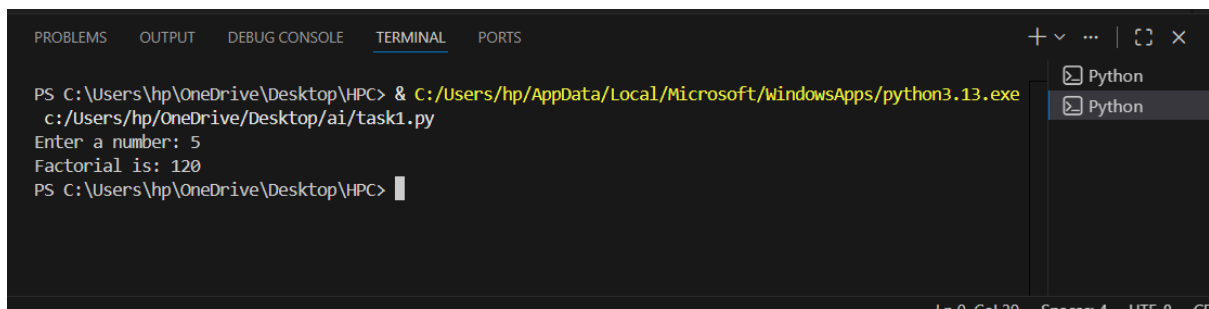


The screenshot shows the Visual Studio Code editor with a file named `task1.py` open. The code implements a factorial function using a while loop. The terminal at the bottom shows the command to run the script and the output for an input of 5.

```
C:\Users\hp> cd "C:\Users\hp\OneDrive\Desktop\ai" & python task1.py
1 # Task 1: Procedural Factorial Implementation
2 num = int(input("Enter a number: "))
3 factorial = 1
4
5 if num < 0:
6     print("Factorial does not exist for negative numbers")
7 elif num == 0:
8     print("The factorial of 0 is 1")
9 else:
10    temp = num
11    while temp > 0:
12        factorial *= temp
13        temp -= 1
14    print(f"The factorial of {num} is {factorial}")
```

Terminal Output:

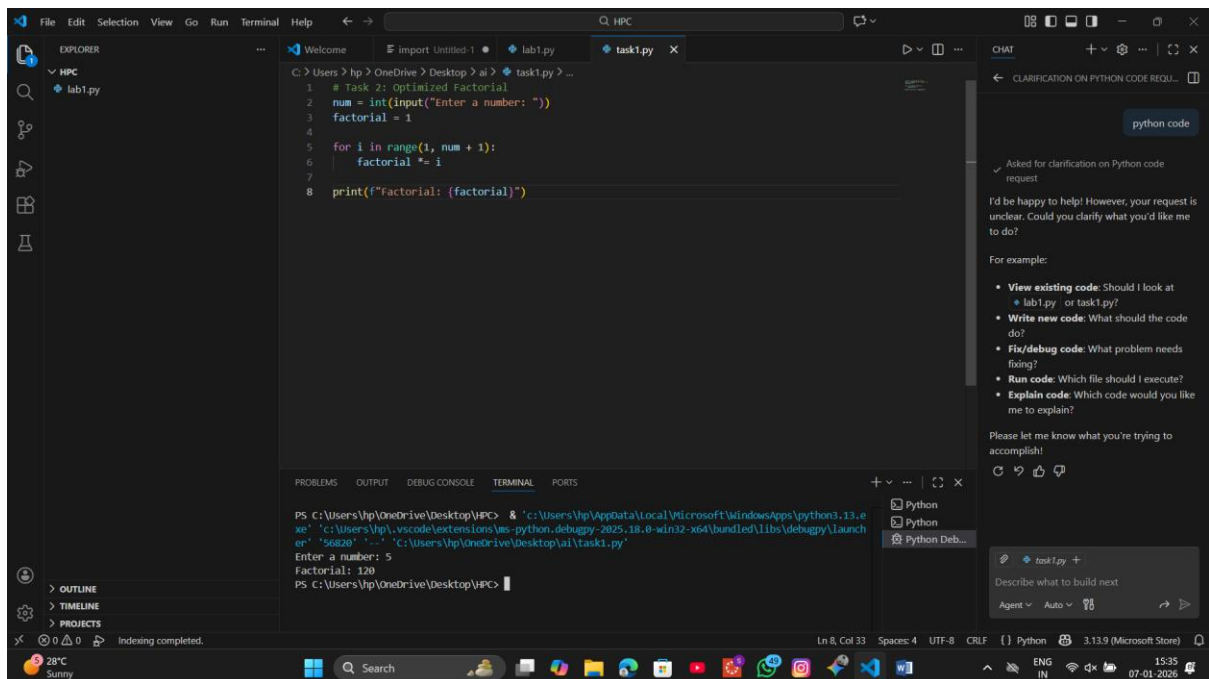
```
PS C:\Users\hp\OneDrive\Desktop\ai> python task1.py
Enter a number: 5
Result: 120
PS C:\Users\hp\OneDrive\Desktop\ai>
```



This screenshot shows a terminal window with the command to run the Python script. The output shows the user entering 5 and the script returning 120 as the factorial.

```
PS C:\Users\hp\OneDrive\Desktop\ai> python task1.py
Enter a number: 5
Result: 120
PS C:\Users\hp\OneDrive\Desktop\ai>
```

Task 2: AI Code Optimization:-



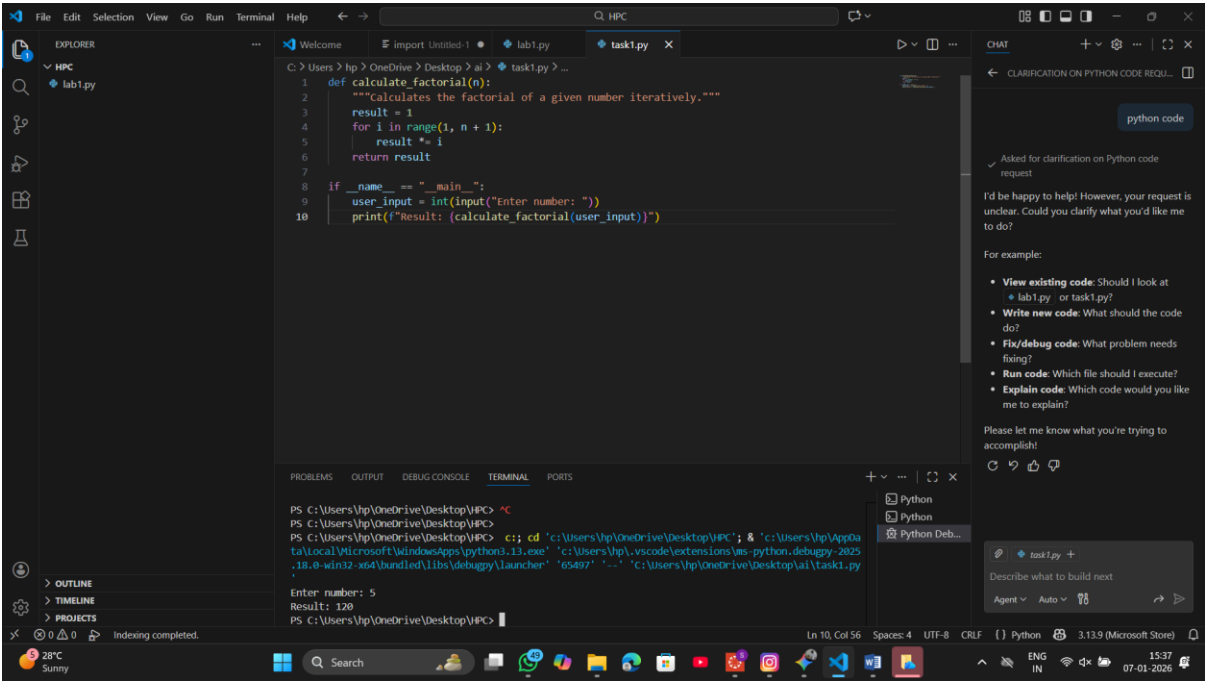
The screenshot shows the Visual Studio Code editor with a file named `task1.py` open. The code implements a factorial function using a for loop, which is more concise than the while loop in Task 1. The terminal at the bottom shows the command to run the script and the output for an input of 5.

```
C:\Users\hp> cd "C:\Users\hp\OneDrive\Desktop\ai" & python task1.py
1 # Task 2: Optimized Factorial
2 num = int(input("Enter a number: "))
3 factorial = 1
4
5 for i in range(1, num + 1):
6     factorial *= i
7
8 print(f"Factorial: {factorial}")
```

Terminal Output:

```
PS C:\Users\hp\OneDrive\Desktop\ai> python task1.py
Enter a number: 5
Factorial: 120
PS C:\Users\hp\OneDrive\Desktop\ai>
```

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:-

