AI ASSISTANT CODING

ASSIGNMENT-5.5

Name: B. Akhira Nandhini

Hallticket:2303A51516

Batch:22

Task Description #1 (Transparency in Algorithm Optimization)

Task: Use AI to generate two solutions for checking prime numbers:

• Naive approach(basic)

• Optimized approach

Prompt:

“Generate Python code for two prime-checking methods and explain how the optimized version improves performance.”

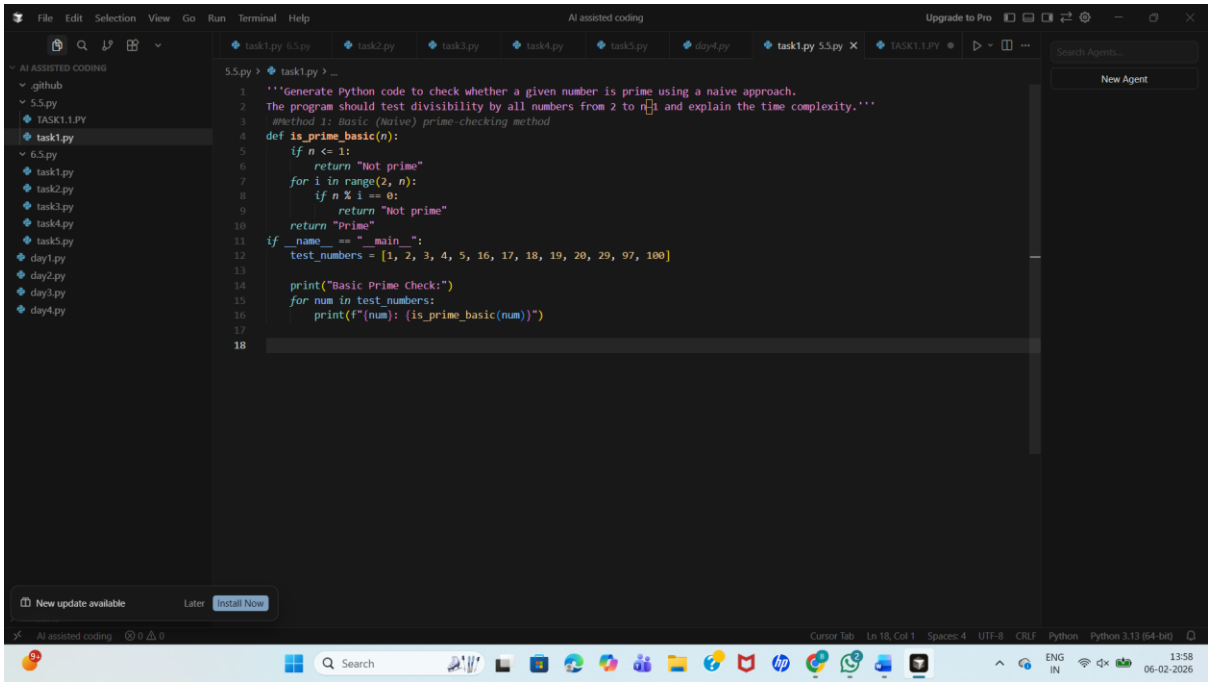
Expected Output:

• Code for both methods.

• Transparent explanation of time complexity.

• Comparison highlighting efficiency improvements.

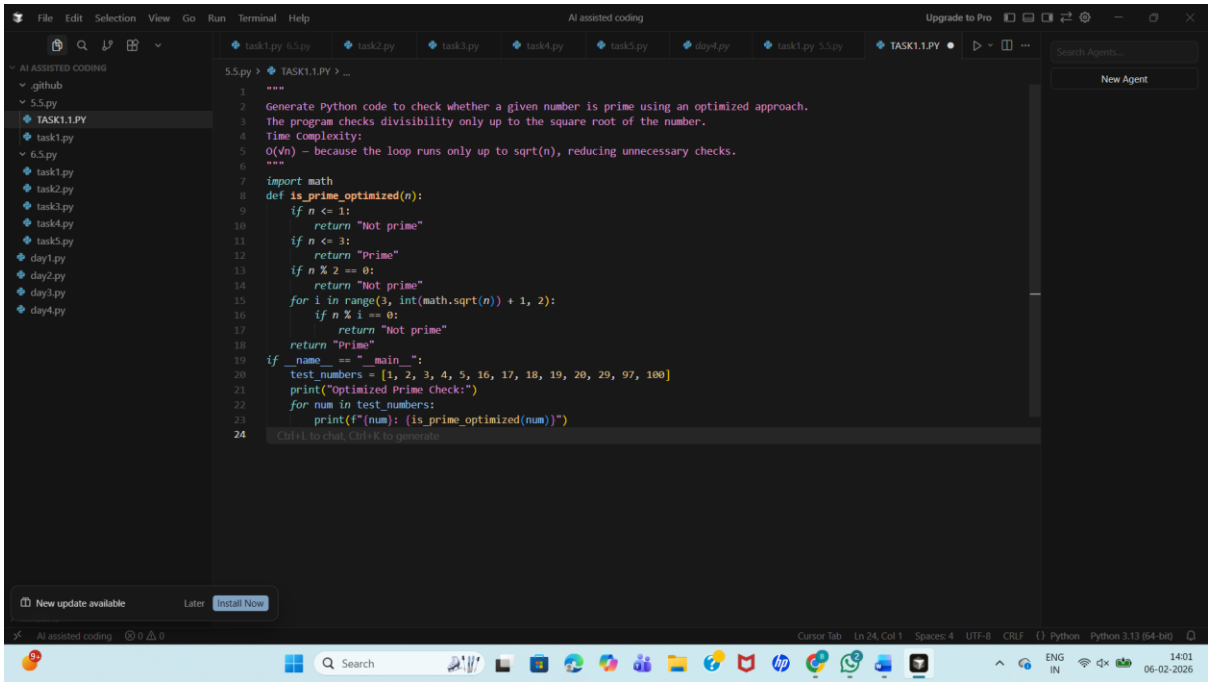
Code:



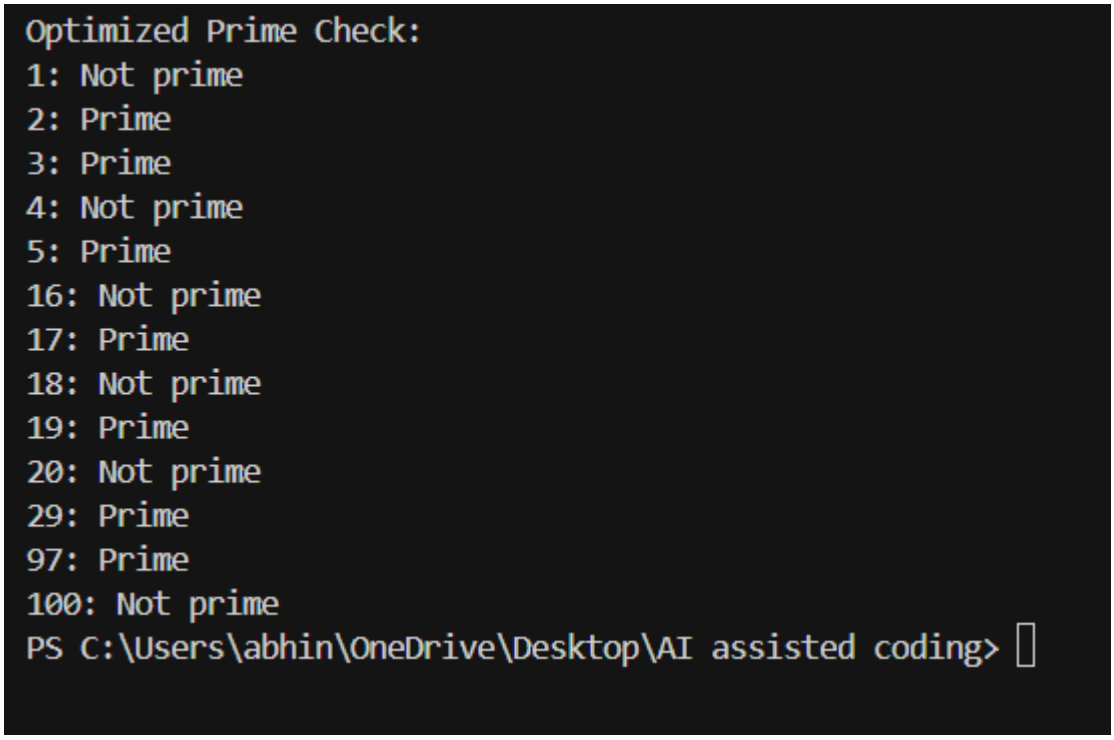
Output:

```
PS C:\Users\abhin\OneDrive\Desktop\AI assisted coding> & C:\Users\abhin\AppData\Local\Programs\Python\Python313\python.e
eDrive/Desktop/AI assisted coding/5.5.py/task1.py"
Basic Prime Check:
1: Not prime
2: Prime
3: Prime
4: Not prime
5: Prime
16: Not prime
17: Prime
18: Not prime
19: Prime
20: Not prime
29: Prime
97: Prime
100: Not prime
```

METHOD 2:



Output:



Explanation:

- In the naive approach, the program checks whether a number is prime by testing its divisibility with **all integers from 2 to n−1**.
  If the number is divisible by any of these values, it is not a prime number; otherwise, it is considered prime.
- This method is easy to understand and implement, but it performs a large number of unnecessary checks, especially for bigger numbers.
- In the optimized approach, the program checks divisibility **only up to the square root of the number**.
  This is based on the mathematical fact that if a number has a factor greater than √n, it must also have a corresponding factor smaller than √n.
- Additionally, even numbers are skipped after checking divisibility by 2, which further reduces the number of iterations

Task Description #2 (Transparency in Recursive Algorithms)

Objective: Use AI to generate a recursive function to calculate

Fibonacci numbers.

Instructions:

1. Ask AI to add clear comments explaining recursion.

2. Ask AI to explain base cases and recursive calls.

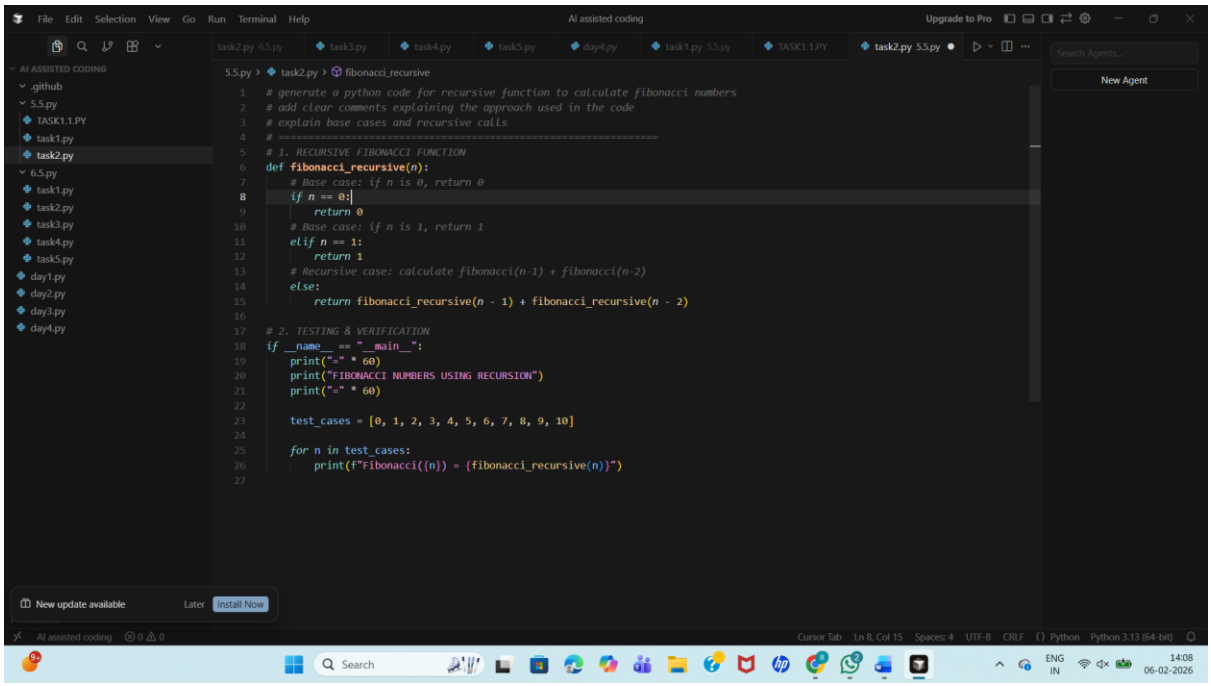
Expected Output:

• Well-commented recursive code.

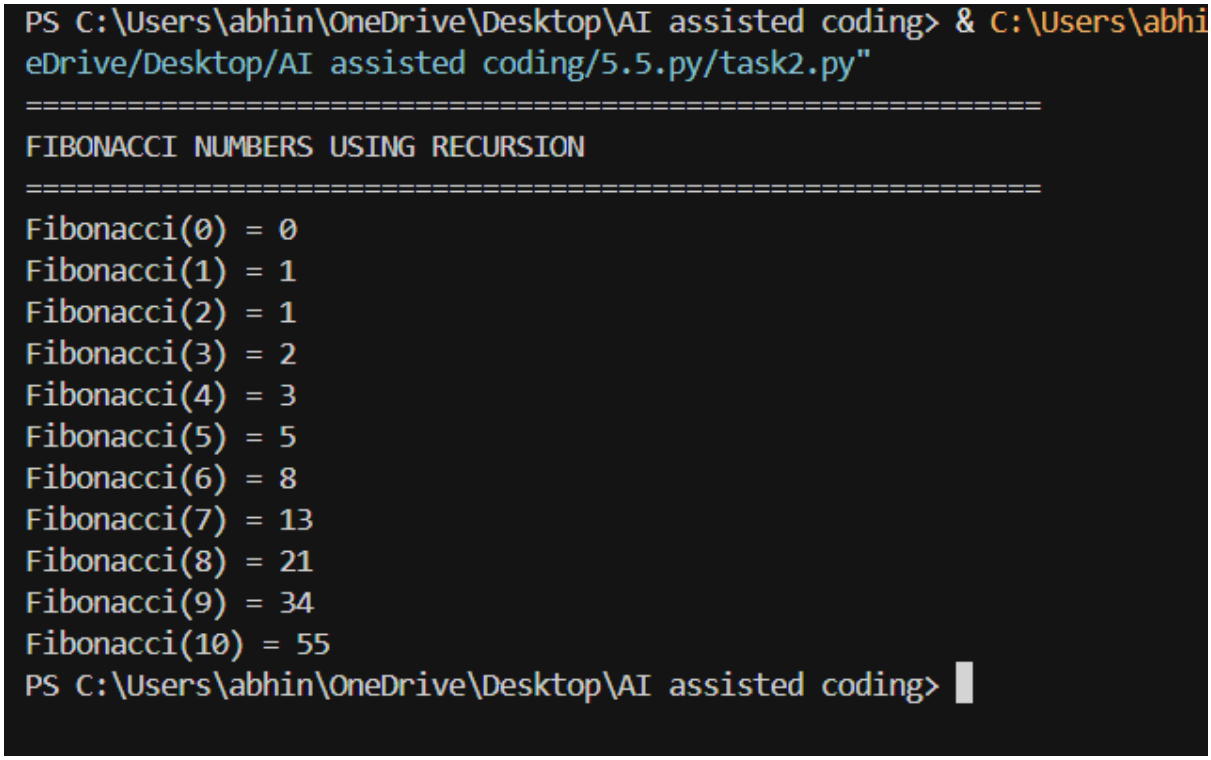
• Clear explanation of how recursion works.

• Verification that explanation matches actual execution.

Code:



Output:



Explanation:

The Fibonacci series is a sequence of numbers in which each number is the sum of the previous two numbers.
The sequence starts as: **0, 1, 1, 2, 3, 5, 8, …**

This program uses **recursion**, where a function calls itself to solve a problem by breaking it into smaller subproblems.

Task Description #3 (Transparency in Error Handling)

Task: Use AI to generate a Python program that reads a file and

processes data.

Prompt:

“Generate code with proper error handling and clear explanations

for each exception.”

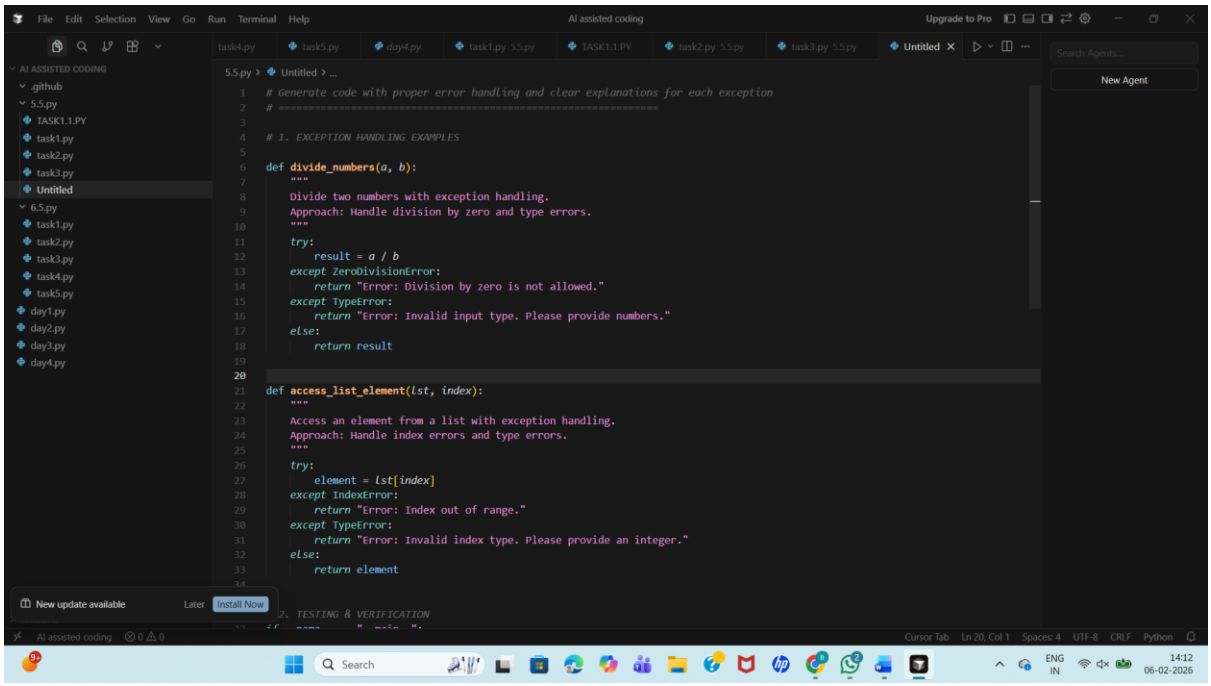
Expected Output:

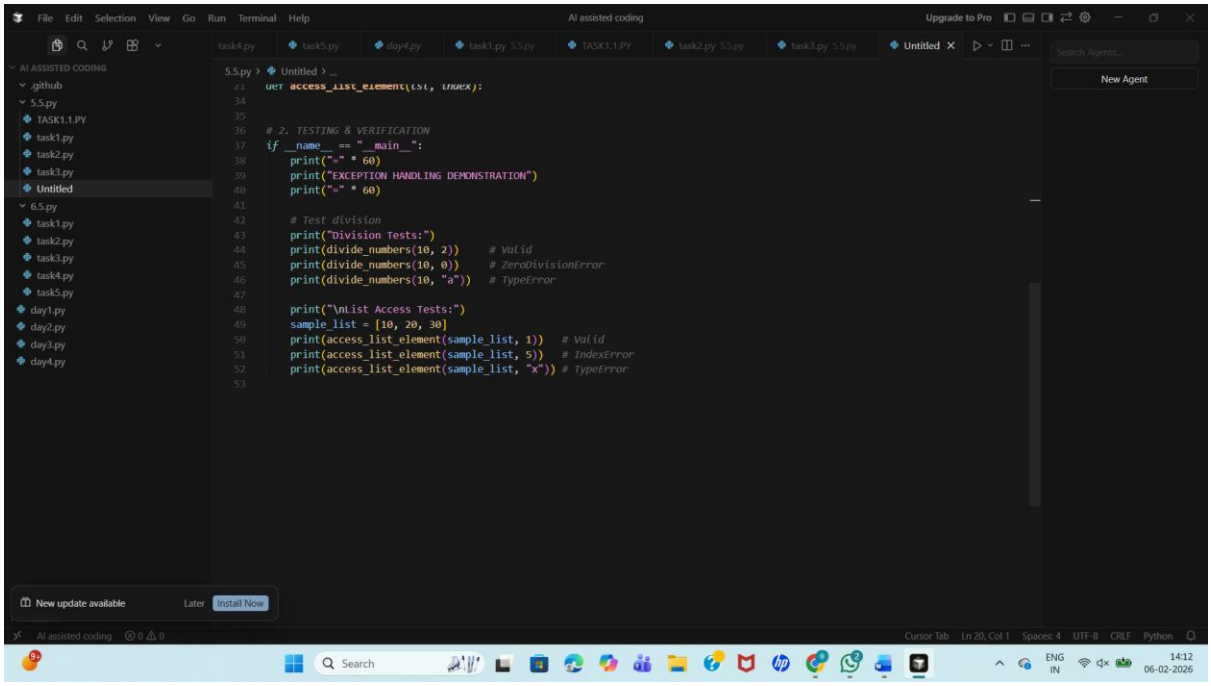
• Code with meaningful exception handling.

• Clear comments explaining each error scenario.

• Validation that explanations align with runtime behavior.

code:





Output:

```
eDrive/Desktop/AI assisted coding/5.5.py/Untitled"
============================================================
EXCEPTION HANDLING DEMONSTRATION
============================================================
Division Tests:
5.0
Error: Division by zero is not allowed.
Error: Invalid input type. Please provide numbers.

List Access Tests:
20
Error: Index out of range.
Error: Invalid index type. Please provide an integer.
PS C:\Users\abhin\OneDrive\Desktop\AI assisted coding>
```

Explanation:

Exception handling is a mechanism in Python used to handle runtime errors so that the program does not crash and can continue executing smoothly. It helps in writing reliable and user-friendly programs.

Task Description #4 (Security in User Authentication)

Task: Use an AI tool to generate a Python-based login system.

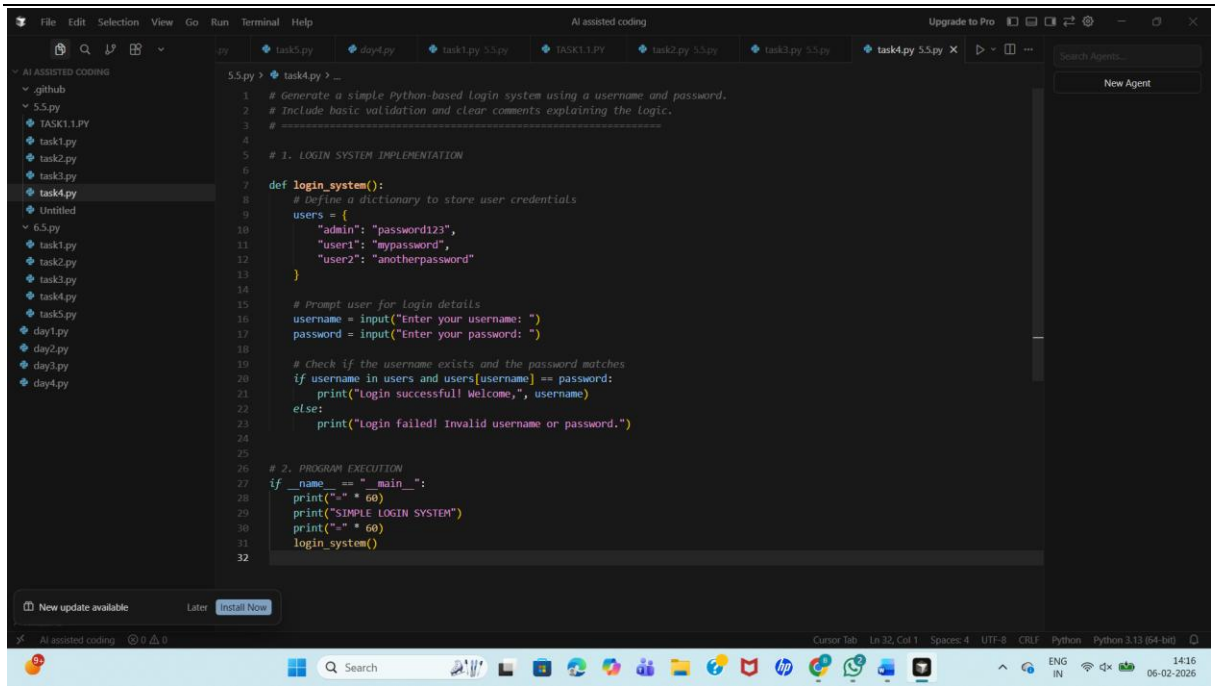
Analyze: Check whether the AI uses secure password handling

practices.

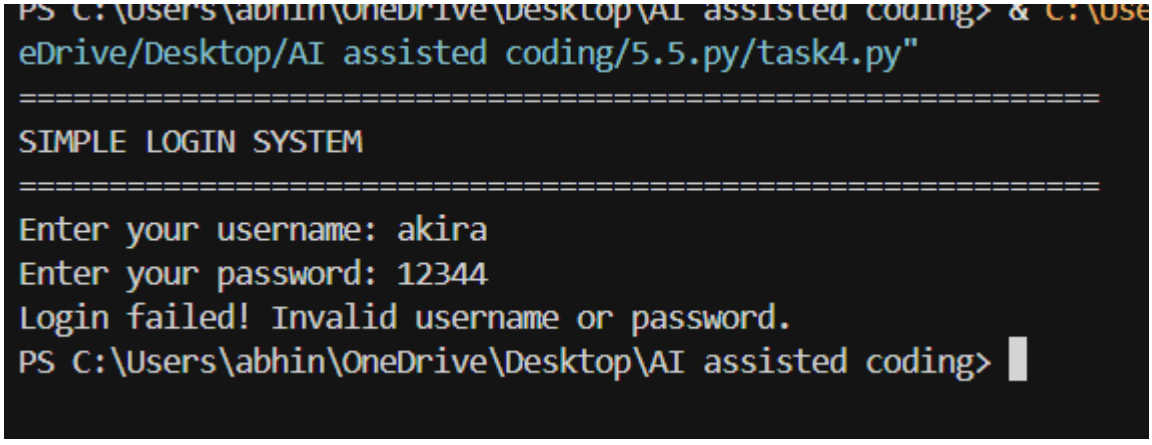
Expected Output:

- Identification of security flaws (plain-text passwords, weak
- validation).
- Revised version using password hashing and input validation.
- Short note on best practices for secure authentication.

Code:



output:



Explanation: This program implements a basic login system using Python. It verifies a user by checking the entered username and password against stored credentials.

Task Description #5 (Privacy in Data Logging)

Task: Use an AI tool to generate a Python script that logs user

activity (username, IP address, timestamp).

Analyze: Examine whether sensitive data is logged unnecessarily

or insecurely.

Expected Output:

• Identified privacy risks in logging.

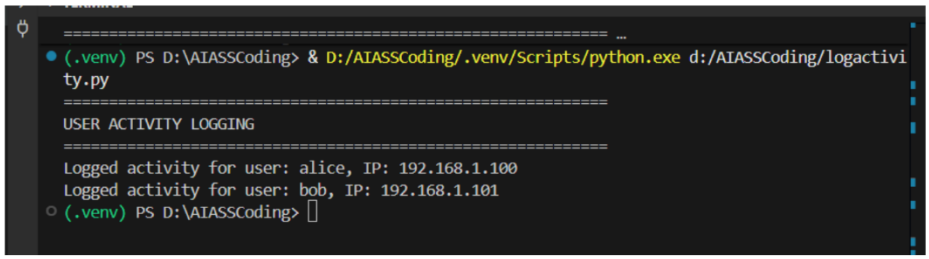
• Improved version with minimal, anonymized, or masked

logging.

• Explanation of privacy-aware logging principles.

CODE :

OUTPUT :



FINAL DESCRIPTION :

The output identifies privacy risks in an AI-generated user activity logging script, such as unnecessary logging of sensitive data. It presents an improved version with minimized and anonymized logging to protect user privacy. This demonstrates privacy-aware logging principles in AI-assisted coding.