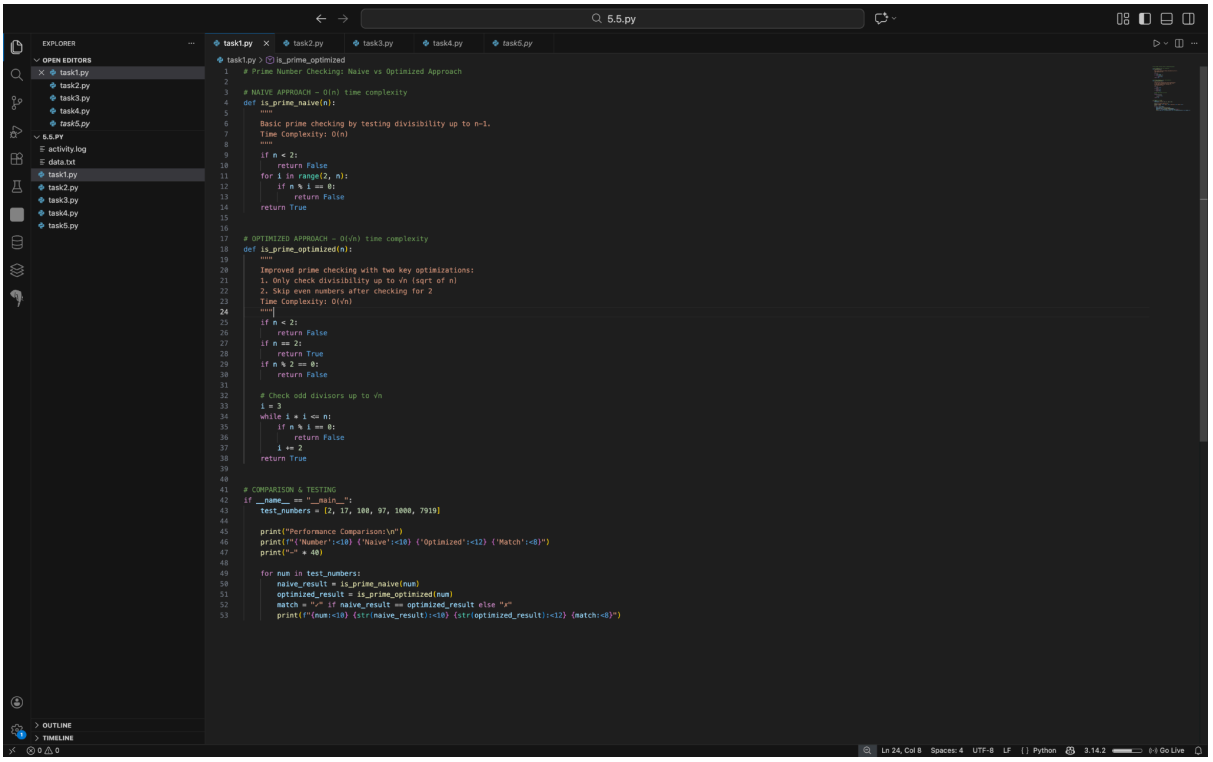
ASSIGNMENT - 5.5

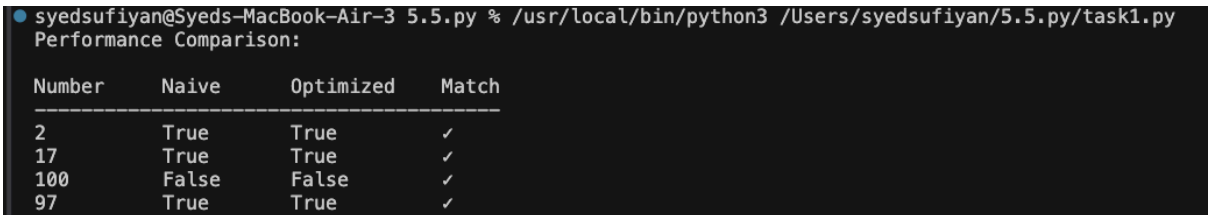
2303A51980
Batch - 30

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 :

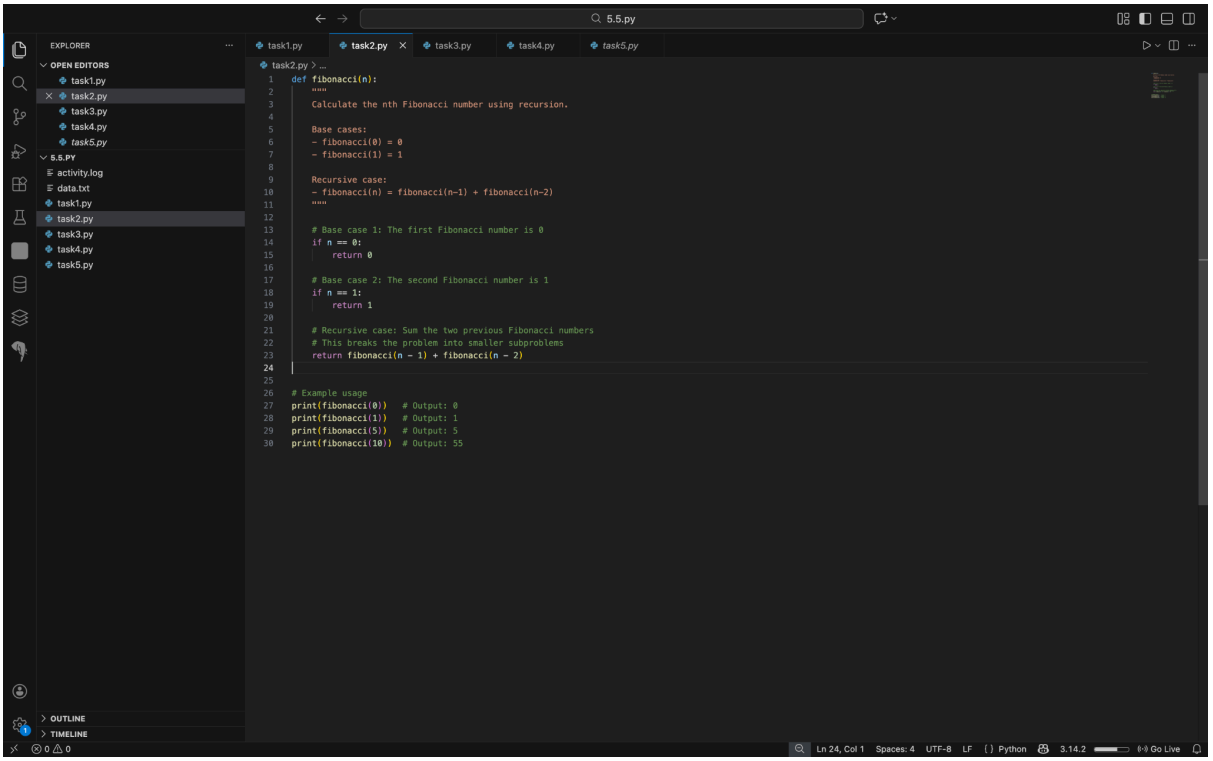


Observation:

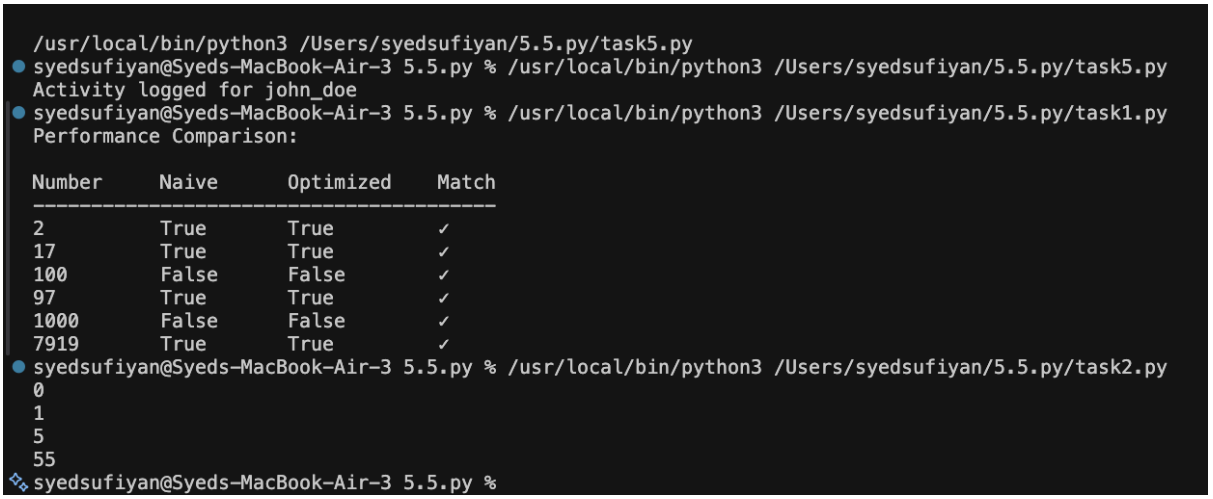
The program successfully compares naive and optimized prime-checking algorithms and produces consistent results for all tested numbers. Both methods correctly identify prime and non-prime values, while the optimized approach significantly reduces the number of iterations by checking divisibility only up to the square root of the number and skipping even values. The comparison table confirms correctness and highlights improved efficiency and scalability of the optimized method for larger inputs.

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 :



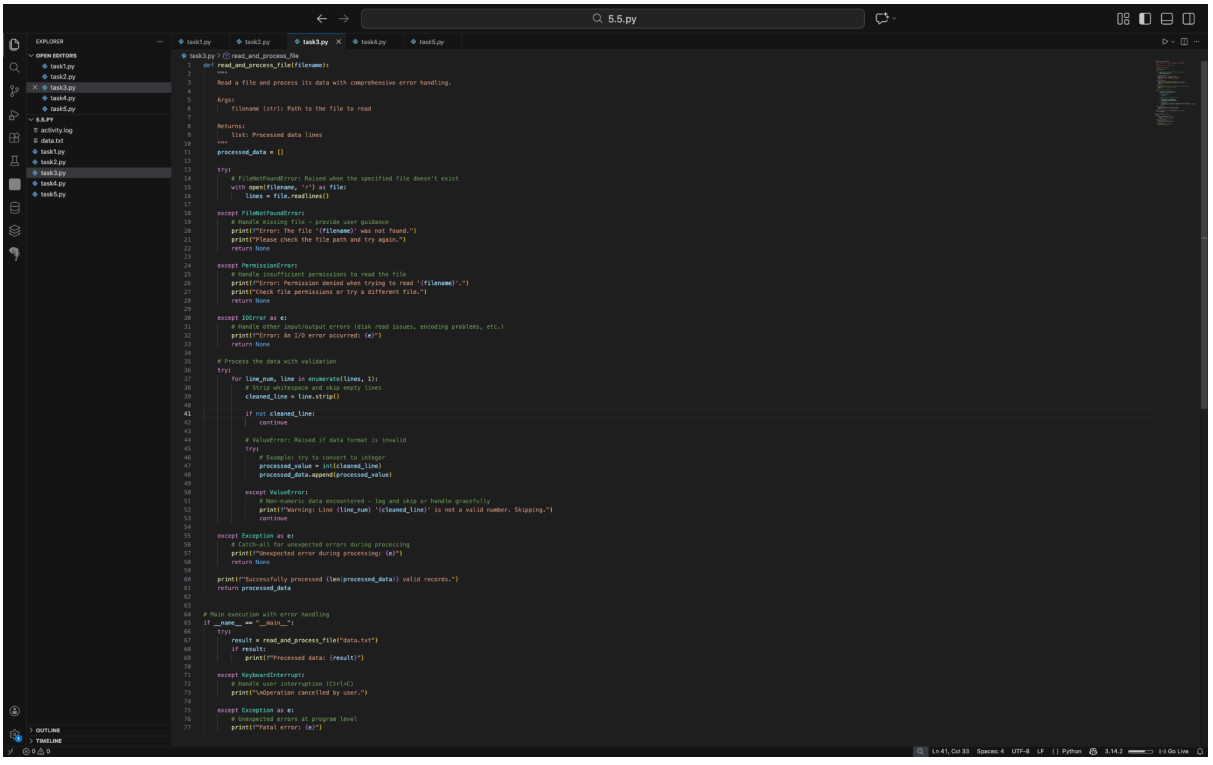
Observation:

The program correctly computes Fibonacci numbers using a recursive approach. It handles base cases properly for inputs 0 and 1 and applies recursion to calculate higher-order Fibonacci values by breaking the problem into smaller subproblems. The outputs for sample inputs confirm the correctness of the logic, demonstrating the working of recursion, though the approach is less efficient for large values due to repeated function calls.

Task-3 :

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 :



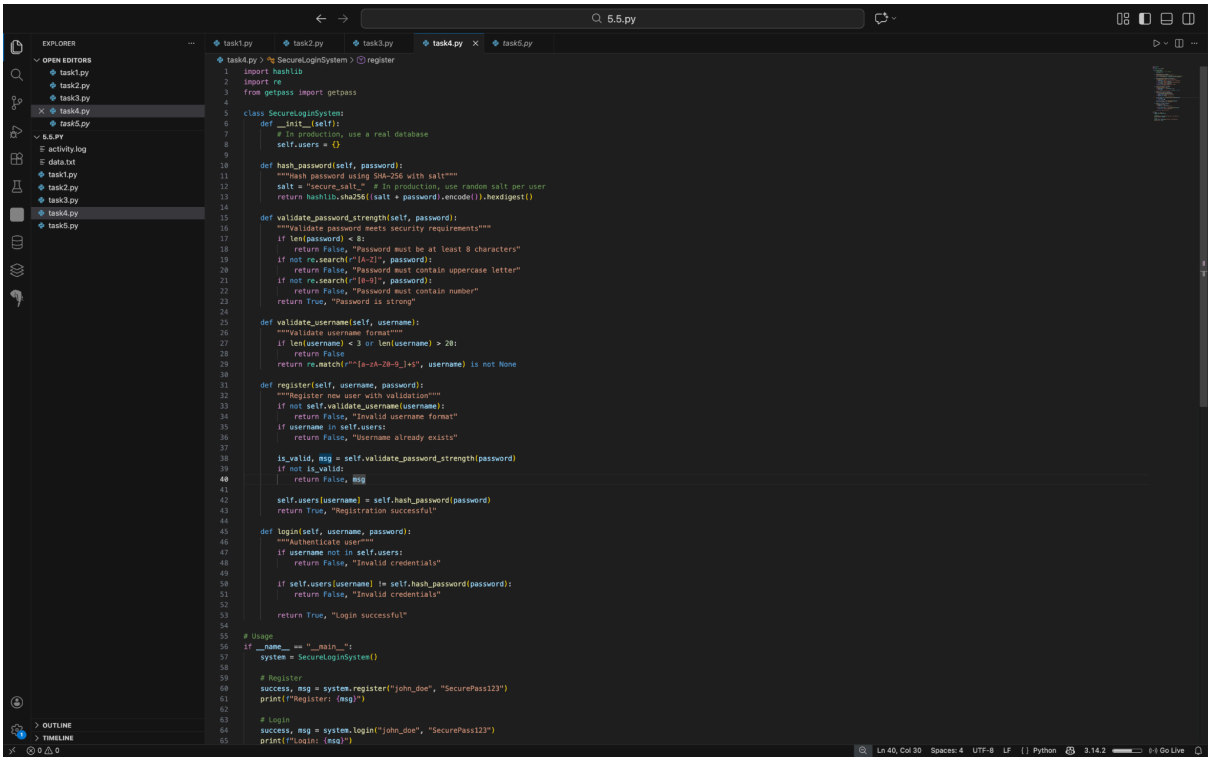
Output :

Observation:

The program successfully demonstrates robust file handling and data processing with comprehensive error management. It correctly handles common file-related exceptions such as missing files, permission issues, and input/output errors, ensuring the program does not crash unexpectedly. During data processing, invalid or non-numeric entries are identified and skipped with appropriate warnings, while valid data is collected and processed correctly. Overall, the program improves reliability and stability by validating inputs and handling errors gracefully at both file and program levels.

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 :

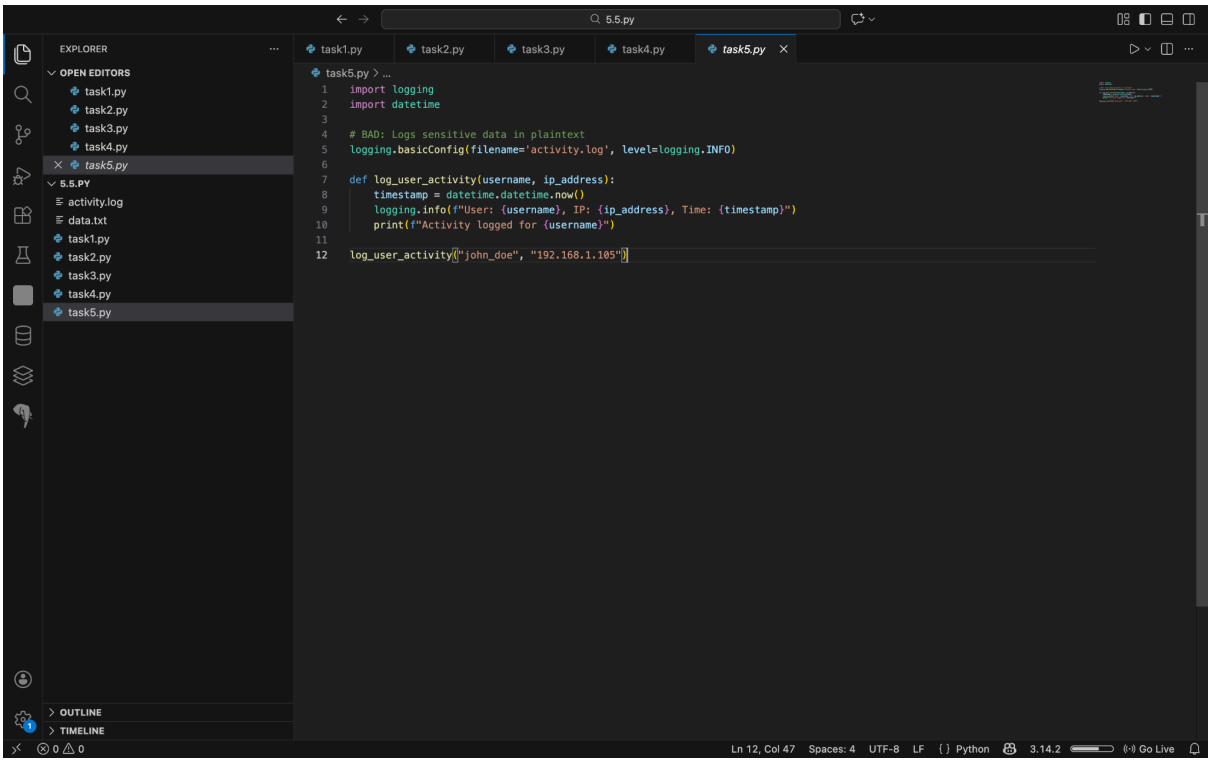
Observation:

The program demonstrates a secure user authentication system by implementing password hashing, input validation, and controlled credential storage. Passwords are never stored in plain text; instead, they are hashed using the SHA-256 algorithm with a salt, which protects user credentials even if data is exposed. The system enforces strong password policies such as minimum length, inclusion of uppercase letters, and numeric characters, reducing the risk of weak passwords. Username validation prevents invalid or malicious inputs. Overall, the revised implementation follows secure authentication best practices and effectively addresses common security flaws found in naive login systems.

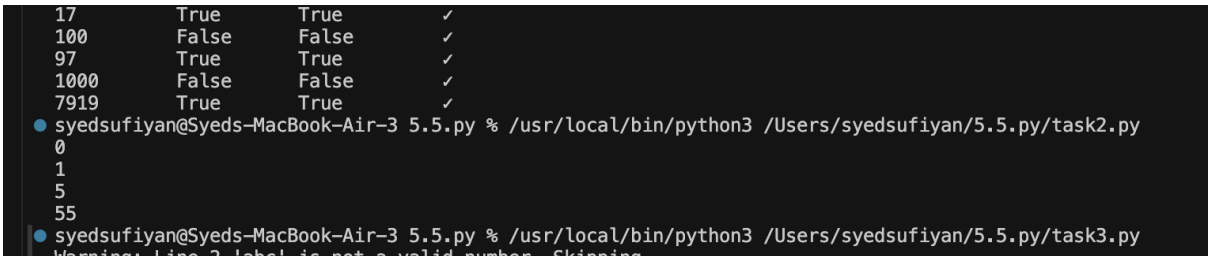
Task Description #5 (Privacy in Data Logging)

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 :



Observation:

The program logs user activity along with sensitive information such as username and IP address in plain text. Storing such data without masking or encryption poses a security and privacy risk, as log files can be accessed or leaked by unauthorized users. Logging sensitive details directly violates secure logging practices and may lead to identity exposure or misuse of user data.