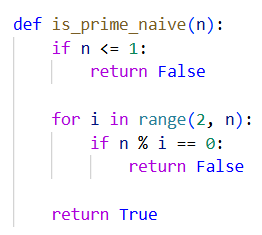
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **Program Name:** B. Tech | | | | **Assignment Type: Lab** | | | **Academic Year:**2025-2026 | | |
| **Course Coordinator Name** | | | | Dr. Rishabh Mittal | | | | | |
| **Instructor(s) Name** | | | | |  | | --- | | Mr. S Naresh Kumar | | Ms. B. Swathi | | Dr. Sasanko Shekhar Gantayat | | Mr. Md Sallauddin | | Dr. Mathivanan | | Mr. Y Srikanth | | Ms. N Shilpa | | Dr. Rishabh Mittal (Coordinator) | | Dr. R. Prashant Kumar | | Mr. Ankushavali MD | | Mr. B Viswanath | | Ms. Sujitha Reddy | | Ms. A. Anitha | | Ms. M.Madhuri | | Ms. Katherashala Swetha | | Ms. Velpula sumalatha | | Mr. Bingi Raju | | | | | | |
| **CourseCode** | | | 23CS002PC304 | **Course Title** | | AI Assisted Coding | | | |
| **Year/Sem** | | | III/II | **Regulation** | | R23 | | | |
| **Date and Day**  **of Assignment** | | | **Week3 –** | **Time(s)** | | 23CSBTB01 To 23CSBTB52 | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | | All batches | | | |
| **Assignment Number: 5.5**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***to complete*** |  |
|  | 1 | **Lab 5: Ethical Foundations – Responsible AI Coding Practices**  **Lab Objectives:**   * To explore the ethical risks associated with AI-generated code. * To recognize issues related to security, bias, transparency, and copyright. * To reflect on the responsibilities of developers when using AI tools in software development. * To promote awareness of best practices for responsible and ethical AI coding.   **Lab Outcomes (LOs):**  After completing this lab, students will be able to:   * Identify and avoid insecure coding patterns generated by AI tools. * Detect and analyze potential bias or discriminatory logic in AI-generated outputs. * Evaluate originality and licensing concerns in reused AI-generated code. * Understand the importance of explainability and transparency in AI-assisted programming. * Reflect on accountability and the human role in ethical AI coding practices.   **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.   **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.   **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.   **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.   **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. | | | | | | Week3 - |  |

**Task 1**:

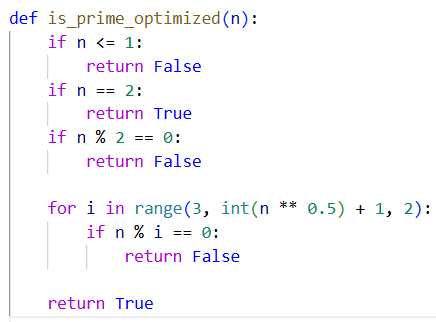
**Prompt** **:**

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

**Suggested code Naive Prime Checking Approach** **:**



**Optimized Prime-Checking Approach :**



**Time Complexity :**

* **O(n)**
* The loop may run up to n-2 iterations in the worst case

**Explanation:**

* The function checks divisibility from 2 to n-1
* If any divisor is found, the number is not prime
* If no divisor is found, the number is prime

**Performance Comparison :**

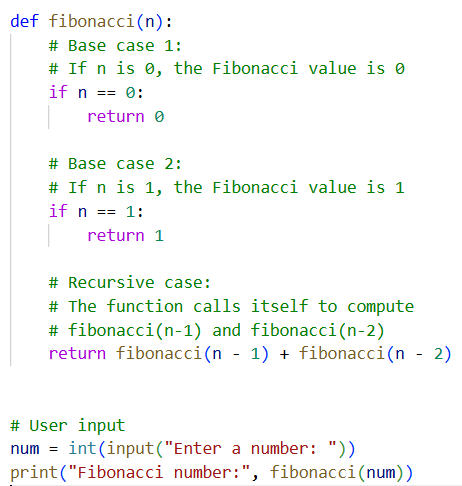
|  |  |  |
| --- | --- | --- |
| **Aspect** | **Naive Approach** | **Optimized Approach** |
| Loop range | 2 to n−1 | 2 to √n |
| Time complexity | O(n) | **O(√n)** |
| Even number handling | Not optimized | Skipped |
| Efficiency for large n | Poor | **High** |
| Practical usability | Low | **High** |

**Task 2** : **Transparency in Recursive Algorithms**

**Prompt :**

Write a recursive Python function to calculate Fibonacci numbers.  
Add clear comments explaining how recursion works.  
Explain the base cases and recursive calls, and ensure the explanation matches the actual execution of the code.

**Suggested Code Recursive Fibonacci Function :**



**Sample output:**



**Explanation of Recursive Calls :**

For values of n greater than 1:

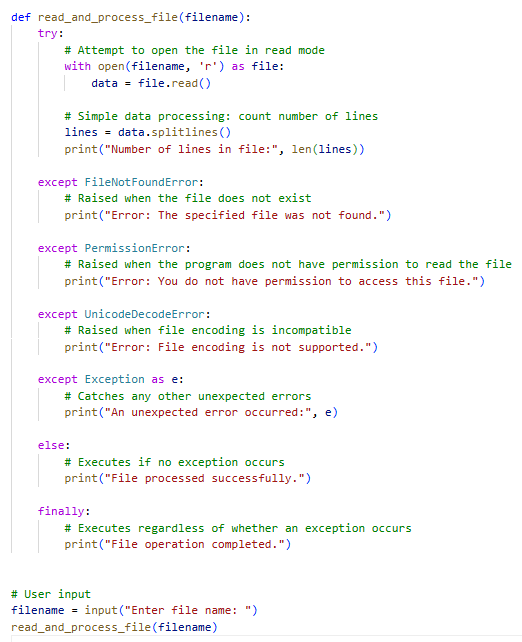
* The function computes  
  **Fibonacci(n) = Fibonacci(n−1) + Fibonacci(n−2)**
* Each call reduces the problem size until a base case is reached
* Results are combined while returning from recursive calls

**Task 3 : Transparency in Error Handling**

**Prompt**:

Generate code with proper error handling and clear explanations for each exception.

**Suggested Code** :



**Sample Output** **:**



**Validation: Explanation vs Runtime Behavior:**

|  |  |  |
| --- | --- | --- |
| **Scenario** | **Runtime Behavior** | **Explanation Match** |
| File does not exist | “File not found” message printed | Yes |
| No read permission | Permission error message | Yes |
| Unsupported encoding | Encoding error message | Yes |
| Valid file | Line count + success message | Yes |
| Any unexpected issue | Generic error message | Yes |

**Task 4** **: Security in User Authentication**

**Prompt :**

Generate a Python-based user login system.  
Store and verify passwords securely using hashing and include basic input validation

**Suggested Code AI-Generated Login System (Initial / Insecure Version) :**



**Sample output** :



**Revised Secure Version (With Hashing & Validation):**



**Sample output :**



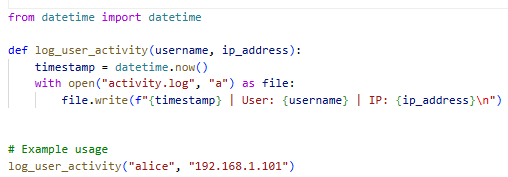
**Identification of Security Flaws :**

|  |  |
| --- | --- |
| **Security Issue** | **Explanation** |
| Plain-text password storage | Password is stored and compared in readable form |
| No password hashing | Easily compromised if code or database is leaked |
| No input validation | Accepts empty or malformed inputs |
| No protection against brute force | Unlimited login attempts |
| Password visible while typing | input() exposes password |

**Task Description 5 : (Privacy in Data Logging)**

**Prompt:**

**Suggested code** **AI-Generated Logging Script (Initial / Privacy-Risky Version) :**



**Identified Privacy Risks in Logging**

|  |  |
| --- | --- |
| Privacy Risk | **Explanation** |
| Logging full IP address | Can be used to identify user location |
| Logging real usernames | Directly identifies individuals |
| No data minimization | Logs more data than required |
| Plain-text storage | Log file can be read if accessed |
| No retention policy | Data stored indefinitely |

**Improved Privacy-Aware Logging Version :**



**How Privacy Is Improved:**

|  |  |
| --- | --- |
| **Improvement** | **Description** |
| Username anonymization | Prevents direct user identification |
| IP masking | Reduces location precision |
| Data minimization | Logs only necessary information |
| Reduced exposure | Limits sensitive data in plain text |

**Privacy-Aware Logging Principles:**

* Log only what is strictly necessary (data minimization)
* Avoid storing personally identifiable information (PII)
* Mask or anonymize sensitive fields
* Protect log files from unauthorized access
* Define data retention and deletion policies
* Comply with privacy regulations (GDPR, etc.)