

## Lab Assignment – 5.5

Hall Ticket No.: 2303A510E6

Name – Anushka Boora

Batch – 29

### Task Description #1 (Transparency in Algorithm Optimization)

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

- Naive approach(basic)
- Optimized approach

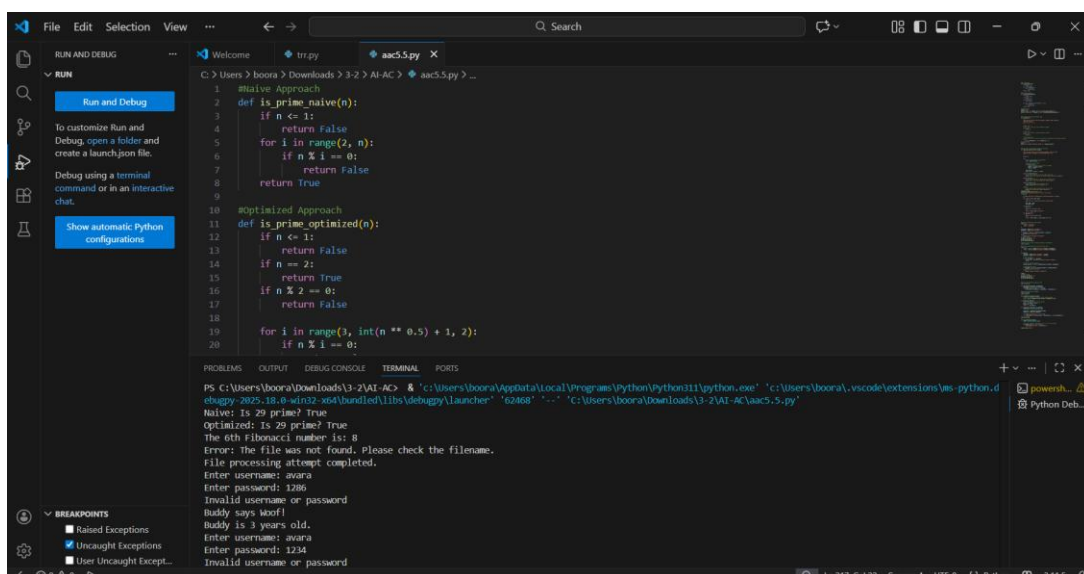
Prompt:

“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

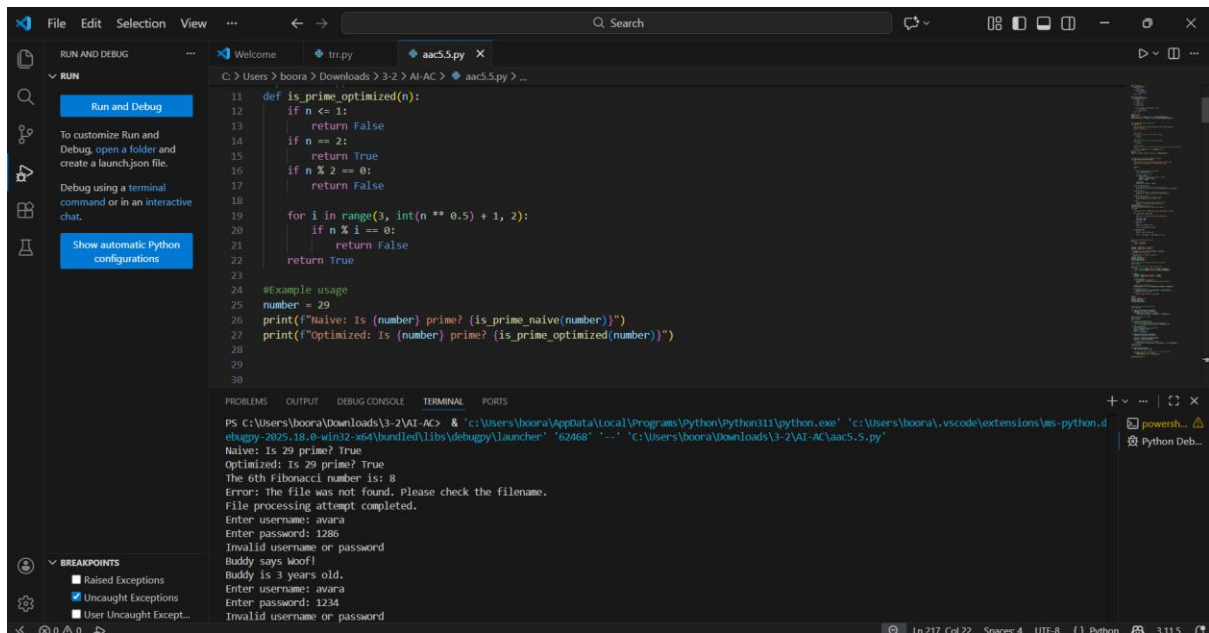


The screenshot shows a VS Code editor with a Python file named `aac5.5.py`. The code defines two functions: `is_prime_naive(n)` and `is_prime_optimized(n)`. The naive function checks divisibility from 2 to  $n-1$ , while the optimized function checks from 2 to  $\sqrt{n}$ . The terminal output shows the execution of the code, including a prompt for a number (29), the result (True), and a comparison of the two methods. The optimized method is shown to be faster.

```
1 #Naive Approach
2 def is_prime_naive(n):
3     if n <= 1:
4         return False
5     for i in range(2, n):
6         if n % i == 0:
7             return False
8     return True
9
10 #Optimized Approach
11 def is_prime_optimized(n):
12     if n <= 1:
13         return False
14     if n == 2:
15         return True
16     if n % 2 == 0:
17         return False
18     for i in range(3, int(n ** 0.5) + 1, 2):
19         if n % i == 0:
20             return False
21     return True
```

Terminal Output:

```
PS C:\Users\boora\Downloads\3-2\AI-AC> & "c:\Users\boora\AppData\Local\Programs\Python\Python311\python.exe" "c:\Users\boora\.vscode\extensions\ms-python.d
ebbuggy-2022.12.0-win32-x64\bundle\libs\debugpy\launcher" "false" "-." "c:\Users\boora\Downloads\3-2\AI-AC\aac5.5.py"
Naive: Is 29 prime? True
Optimized: Is 29 prime? True
The 0th Fibonacci number is: 0
Error: The file was not found. Please check the filename.
File processing attempt completed.
Enter username: avara
Enter password: 1286
Invalid username or password
Buddy says Woof!
Buddy is 3 years old.
Enter username: avara
Enter password: 1234
Invalid username or password
```



## Task Description #2 (Transparency in Recursive Algorithms)

Objective: Use AI to generate a recursive function to calculate Fibonacci numbers.

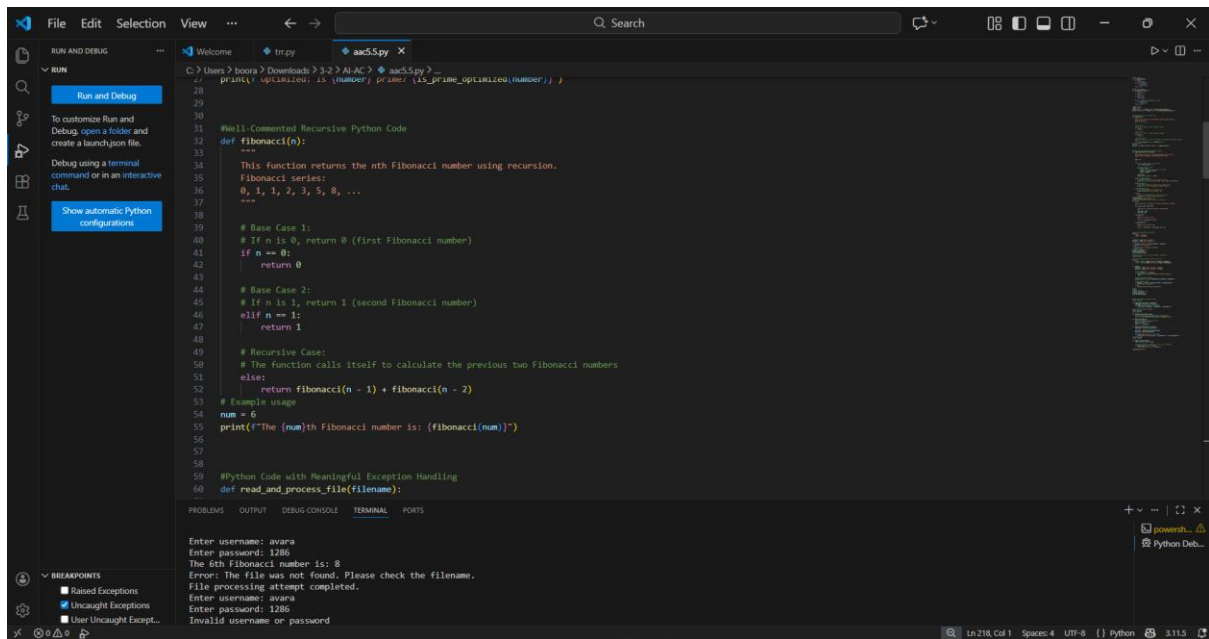
Instructions:

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



The screenshot shows a Visual Studio Code editor with a Python file named `aa55.py`. The code includes a recursive Fibonacci function, an example usage, and a function for reading and processing a file with exception handling. The terminal output shows the execution of the program, including user input for a file name and the resulting Fibonacci number.

```
File Edit Selection View ... Search
C:\Users\boora> Downloads> 3-2> AI-AC> aa55.py
print(f'optimized: {is_optimized} prime: {is_prime_optimized}')

#Well-Commented Recursive Python Code
def fibonacci(n):
    """
    This function returns the nth Fibonacci number using recursion.
    Fibonacci series:
    0, 1, 1, 2, 3, 5, 8, ...
    """
    # Base Case 1:
    # If n is 0, return 0 (first Fibonacci number)
    if n == 0:
        return 0
    # Base Case 2:
    # If n is 1, return 1 (second Fibonacci number)
    elif n == 1:
        return 1
    # Recursive Case:
    # The function calls itself to calculate the previous two Fibonacci numbers
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
# Example usage
num = 6
print(f"The {num}th Fibonacci number is: {fibonacci(num)}")

#Python Code with Meaningful Exception Handling
def read_and_process_file(filename):
    """
    This function reads a file and processes its content.
    """
    try:
        with open(filename, 'r') as file:
            content = file.read()
            # Process the content here
    except FileNotFoundError:
        print(f"Error: The file was not found. Please check the filename.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
    finally:
        print("File processing attempt completed.")
# Example usage
filename = "data.txt"
read_and_process_file(filename)
```

Enter username: avara  
Enter password: 1286  
The 6th Fibonacci number is: 8  
Error: The file was not found. Please check the filename.  
File processing attempt completed.  
Enter username: avara  
Enter password: 1286  
Invalid username or password

### 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

```
58
59 #Python Code with Meaningful Exception Handling
60 def read_and_process_file(filename):
61     """
62     This function reads a file containing numbers (one per line),
63     converts them to integers, and calculates their sum.
64     """
65     total = 0
66
67     try:
68         # Try to open the file in read mode
69         file = open(filename, "r")
70
71         # Read file line by line
72         for line in file:
73             # Remove whitespace and convert to integer
74             number = int(line.strip())
75             total += number
76
77         file.close()
78         print("Sum of numbers:", total)
79
80     except FileNotFoundError:
81         # This block executes if the file does not exist
82         print("Error: The file was not found. Please check the filename.")
83
84     except ValueError:
85         # This block executes if conversion to integer fails
86         print("Error: The file contains non-numeric data.")
87
88     except Exception as e:
89         # This block handles any unexpected error
90         print("Unexpected error occurred:", e)
91
```

Enter username: avara  
Enter password: 1286  
The 6th Fibonacci number is: 8  
Error: The file was not found. Please check the filename.  
File processing attempt completed.  
Enter username: avara  
Enter password: 1286  
Invalid username or password

```
89 except Exception as e:
90     # This block handles any unexpected error
91     print("Unexpected error occurred:", e)
92
93 finally:
94     # This block always executes (error or no error)
95     print("File processing attempt completed.")
96
97 # Example usage
98 read_and_process_file("numbers.txt")
99
100 #Python Code Demonstrating Use of Classes and Objects
101 class Dog:
102     """
103     A simple Dog class to demonstrate classes and objects in Python.
104     """
105     def __init__(self, name, age):
106         """
107         Constructor to initialize the dog's name and age.
108         """
109         self.name = name
110         self.age = age
111
112     def bark(self):
113         """
114         Method for the dog to bark.
115         """
116         return f"{self.name} says Woof!"
117
118     def get_age(self):
119         """
120         Method to get the dog's age.
121         """
122         return f"{self.name} is {self.age} years old."
```

Enter username: avara  
Enter password: 1286  
The 6th Fibonacci number is: 8  
Error: The file was not found. Please check the filename.  
File processing attempt completed.  
Enter username: avara  
Enter password: 1286  
Invalid username or password

## 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

```

123
124 #Insecure AI-Generated Login System
125 users = {
126     "admin": "admin123",
127     "user1": "password"
128 }
129
130 username = input("Enter username: ")
131 password = input("Enter password: ")
132
133 if username in users and users[username] == password:
134     print("Login successful")
135 else:
136     print("Invalid username or password")
137
138 # Example usage
139 my_dog = Dog("Buddy", 3)
140 print(my_dog.bark())
141 print(my_dog.get_age())
142
143 #Revised Secure Version (Password Hashing + Validation)
144 import hashlib
145
146 # Storing hashed passwords instead of plain text
147 users = {
148     "admin": hashlib.sha256("admin123".encode()).hexdigest(),
149     "user1": hashlib.sha256("securepass".encode()).hexdigest()
150 }
151
152 def login():
153     username = input("Enter username: ").strip()
154     password = input("Enter password: ").strip()
155
156     # Input validation
157     if not username or not password:
158         print("Invalid username or password")
159
160
161 Enter username: admin
162 Enter password: 1286
163 The 6th Fibonacci number is: 8
164 Error: The file was not found. Please check the filename.
165 File processing attempt completed.
166 Enter username: admin
167 Enter password: 1286
168 Invalid username or password
  
```

## 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

