

# Assignment – 5.5

HT No.: 2303A510D9

Bt – 29

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

Use AI to generate two solutions for checking prime numbers:

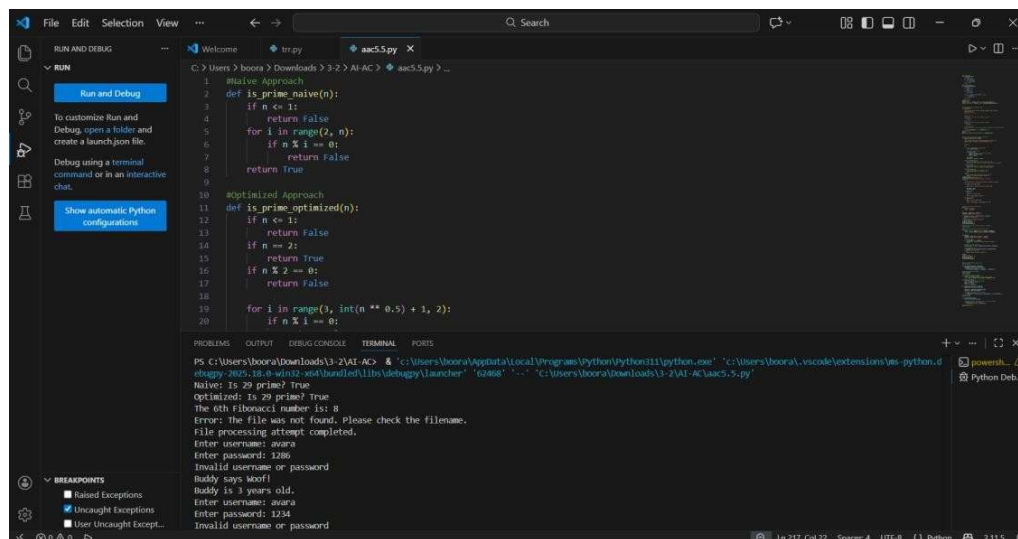
- Naive approach(basic)
- Optimized approach Prompt:

“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 Python IDE with a file named `aac5.py`. The code defines two functions: `naive_approach` and `optimized_approach`. The `naive_approach` function checks for primality by testing divisibility from 2 to `n-1`. The `optimized_approach` function checks for primality by testing divisibility from 2 to  $\sqrt{n}$ . The output window shows the execution of the code, displaying the results for `naive_approach(20)` and `optimized_approach(20)`, both returning `True`. It also shows the 6th Fibonacci number is 8. The output window also displays an error message: "Error: The file was not found. Please check the filename." and a message: "File processing attempt completed. Enter username: avara Enter password: 1234 Invalid username or password".

```
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
```

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...  
naive: 20 prime? True  
Optimized: 20 prime? True  
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: 1234  
Invalid username or password

```
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
19     for i in range(3, int(n ** 0.5) + 1, 2):
20         if n % i == 0:
21             return False
22     return True
23
24 #Example usage
25 number = 29
26 print(f"Naive: Is {number} prime? {is_prime_naive(number)}")
27 print(f"Optimized: Is {number} prime? {is_prime_optimized(number)}")
28
29
30
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
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
ebugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62468' '-' 'c:\Users\boora\Downloads\3-2\AI-AC\aac5.5.py'
Naive: Is 29 prime? True
Optimized: Is 29 prime? True
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
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.

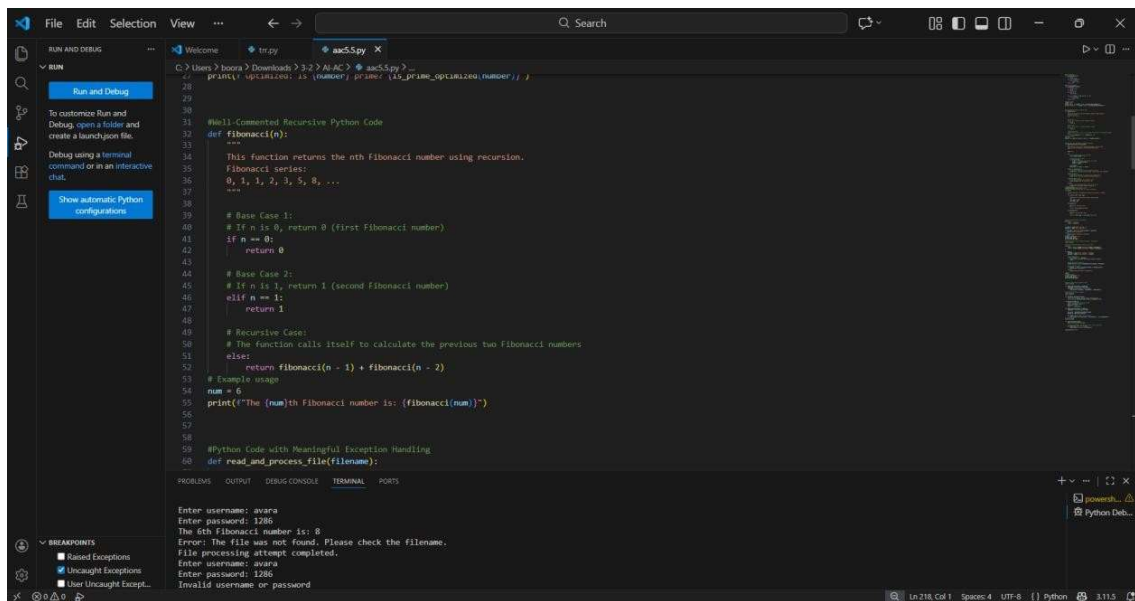
Instructions:

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



### 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

```
File Edit Selection View ... Search
C:\Users\boora> Downloads > 3-2 > AI AC > aad5py > ...

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
92 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ - - | x
Python Deb...

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
```

```
File Edit Selection View ... Search
C:\Users\boora> Downloads > 3-2 > AI AC > aad5py > ...

90
91 except Exception as e:
92     # This block handles any unexpected error
93     print("Unexpected error occurred:", e)
94
95 finally:
96     # This block always executes (error or no error)
97     print("File processing attempt completed.")
98
99 # Example usage
100 read_and_process_file("numbers.txt")
101
102 #Python Code Demonstrating Use of Classes and Objects
103 class Dog:
104     """
105     A simple Dog class to demonstrate classes and objects in Python.
106     """
107     def __init__(self, name, age):
108         """
109         Constructor to initialize the dog's name and age.
110         """
111         self.name = name
112         self.age = age
113
114     def bark(self):
115         """
116         Method for the dog to bark.
117         """
118         return f"{self.name} says Woof!"
119
120     def get_age(self):
121         """
122         Method to get the dog's age.
123         """
124         return f"{self.name} is {self.age} years old."
125
126 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ - - | x
Python Deb...

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

The screenshot shows a Python IDE with a file named `aac5.py`. The code implements a login system with two versions: an insecure one and a revised secure one using password hashing. The terminal output shows the execution of the script, including user input and error messages.

```

123 #Insecure AI-Generated Login System
124
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     if username in users and users[username] == password:
161         print("Login successful")
162     else:
163         print("Invalid username or password")
164
165 if __name__ == "__main__":
166     login()
167
168 # Fibonacci function
169 def fibonacci(n):
170     if n < 0:
171         return 0
172     elif n == 0:
173         return 0
174     elif n == 1:
175         return 1
176     else:
177         return fibonacci(n-1) + fibonacci(n-2)
178
179 # Example usage
180 print(fibonacci(8))
181
182 # File processing
183 def process_file(filename):
184     try:
185         with open(filename, 'r') as file:
186             content = file.read()
187             print(f"File content: {content}")
188     except FileNotFoundError:
189         print(f"Error: The file was not found. Please check the filename.")
190     except Exception as e:
191         print(f"An error occurred: {e}")
192
193 # Example usage
194 process_file("example.txt")
195
196 # Main execution
197 if __name__ == "__main__":
198     login()
199     fibonacci(8)
200     process_file("example.txt")
  
```

Terminal Output:

```

Enter username: avara
Enter password: 1286
The 8th 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 #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

