

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 the Visual Studio Code interface with two files open: 'aac5.5.py' and 'tr.py'. The 'aac5.5.py' file contains two functions: 'is_prime_naive(n)' and 'is_prime_optimized(n)'. The 'is_prime_naive' function uses a naive approach with nested loops. The 'is_prime_optimized' function uses an optimized approach with a range of square root plus one. The terminal below shows the execution of 'aac5.5.py' and its output, which includes a comparison of the two methods.

```
C:\> Users> boora> Downloads> 3-2> AI-AC> aac5.5.py > ...
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
19     for i in range(1, int(n ** 0.5) + 1, 2):
20         if n % i == 0:
21             return False
22
23
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.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 60th Fibonacci number is: 18
Enter the file was not found. Please check the filename.
File processing attempt completed.
Enter username: avara
Enter password: 1236
Invalid username or password
Buddy says woof!
Buddy is 3 years old.
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("Naive: Is {} prime? {}".format(number, is_prime_naive(number)))
27 print("Optimized: Is {} prime? {}".format(number, is_prime_optimized(number)))
28
29
30
PS C:\Users\boora\Downloads\3-2\AI-AC> & 'c:/Users/boora/AppData/Local/Programs/Python/3.11/python.exe' 'c:/Users/boora/.vscode/extensions/ms-python.debugpy-2025.18.0-win32-x64/bundle/lib/python/debugpy/launcher' '62468' '--' 'C:/Users/boora/Downloads/3-2/AI-AC/aac5.py'
Debugpy: Is 29 prime? True
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: 1234
Invalid username or password
Buddy says Noof!
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 Python code editor interface with a dark theme. The main window displays a file named `aas5.py` containing the following code:

```
C:\> Users > hoora > Downloads > 3-2 > AI-AC > aas5.py > ...
30
31 #Well-Commented Recursive Python Code
32 def fibonacci(n):
33     """
34     This function returns the nth Fibonacci number using recursion.
35     Fibonacci series:
36     0, 1, 1, 2, 3, 5, 8, ...
37     """
38
39     # Base Case 1:
40     # If n is 0, return 0 (first Fibonacci number)
41     if n == 0:
42         return 0
43
44     # Base Case 2:
45     # If n is 1, return 1 (second Fibonacci number)
46     elif n == 1:
47         return 1
48
49     # Recursive Case:
50     # The function calls itself to calculate the previous two Fibonacci numbers
51     else:
52         return fibonacci(n - 1) + fibonacci(n - 2)
53
54 # Example usage
55 num = 6
56 print(f"The {num}th Fibonacci number is: {fibonacci(num)}")
57
58
59 #Python Code with Meaningful Exception Handling
60 def read_and_process_file(filename):
61     """
62     This function reads a file and processes its contents.
63     """
64     try:
65         with open(filename, 'r') as file:
66             data = file.read()
67             print(data)
68     except FileNotFoundError:
69         print(f"Error: The file was not found. Please check the filename.")
70     except IOError:
71         print(f"File processing attempt completed.")
72     finally:
73         print("File processing attempt completed.")
```

The bottom status bar indicates the code is in Python mode, version 3.11.5.

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 ... < > Q Search
RUN AND DEBUG ... Welcome tr.py aac5.5.py
C:\Users\bora\Downloads\3-2>AI-AC>aac5.5.py ...
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
66     total = 0
67
68     try:
69         # Try to open the file in read mode
70         file = open(filename, "r")
71
72         # Read file line by line
73         for line in file:
74             # Remove whitespace and convert to integer
75             number = int(line.strip())
76             total += number
77
78         file.close()
79         print("Sum of numbers:", total)
80
81     except FileNotFoundError:
82         # This block executes if the file does not exist
83         print("Error: The file was not found. Please check the filename.")
84
85     except ValueError:
86         # This block executes if conversion to integer fails
87         print("Error: The file contains non-numeric data.")
88
89     except Exception as e:
90         # This block handles any unexpected error
91         print("Unexpected error occurred:", e)
```

The screenshot shows a Python code editor interface with the following details:

- File Explorer:** Shows a folder structure: C:\Users\bora\Downloads\3-2\AI-AC\aa5.py
- Run and Debug:** A sidebar with instructions to customize run and debug settings.
- Code Editor:** The main area displays the Python script 'aa5.py'. The code includes:
 - A function `read_and_process_file(filename)` that handles exceptions and prints file processing attempts.
 - An example usage of the function.
 - A class `Dog` with methods `bark()` and `get_age()`.
- Terminal:** A tab showing command-line output:

```
Enter username: avara
Enter password: 1286
The 6th Fibonacci number is: 8
Error: File was not found. Please check the filename.
File processing attempt completed.
Enter username: avara
Enter password: 1286
Invalid username or password
```
- Breakpoints:** A sidebar showing breakpoints for raised exceptions and uncaught exceptions.
- Bottom Status Bar:** Shows line 218, column 1, spaces 4, UTF-8, Python, and line 3115.

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 code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Back, Forward, Search.
- Sidebar:** RUN, BREAKPOINTS, powerhell, Python Deb.
- Code Area:** The file 'aac5.5.py' contains the following code:

```
1 # Welcome
2 #!/usr/bin/python
3 # insecure AI-generated Login System
4 users = {
5     "admin": "admin123",
6     "user1": "password"
7 }
8
9 username = input("Enter username: ")
10 password = input("Enter password: ")
11
12 if username in users and users[username] == password:
13     print("Login successful")
14 else:
15     print("Invalid username or password")
16 # Example usage
17 my_dog = Dog("Buddy", 3)
18 print(my_dog bark())
19 print(my_dog.get_age())
20
21 # Revised Secure Version (Password Hashing + Validation)
22 import hashlib
23
24 # Storing hashed passwords instead of plain text
25 users = {
26     "admin": hashlib.sha256("admin123".encode()).hexdigest(),
27     "user1": hashlib.sha256("securepass".encode()).hexdigest()
28 }
29
30 def login():
31     username = input("Enter username: ").strip()
32     password = input("Enter password: ").strip()
33
34     # Input validation
35     if not username or not password:
36         print("Enter username and password")
```
- Terminal:** Shows the execution of the script:

```
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
```
- Status Bar:** Ln 218, Col 1 | Spaces: 4 | UTF-8 | Python | 3.11.5

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

The screenshot shows a Python IDE interface with the following details:

- File Bar:** File, Edit, Selection, View, ..., Search.
- Run Bar:** RUN AND DEBUG, RUN, Run and Debug (highlighted).
- Code Editor:** A file named `tmp.py` containing Python code for an "Insecure AI-Generated Logging Script". The code includes functions for logging activity, anonymizing usernames, and masking IP addresses.
- Terminal:** Shows command-line interactions with the script, including entering a username and password, and attempting to log activity.
- Breakpoints:** A sidebar showing breakpoints categorized by type: Raised Exceptions, Uncaught Exceptions, and User Uncaught Exceptions.
- Output:** A terminal window showing the execution of the script and its output.

```
C:\Users\bora>Downloads>3-2>AI-AC>tmp.py & aas55.py > ...
177 #Insecure AI-Generated Logging Script
178
179 import datetime
180
181 def log_activity(username, ip_address):
182     timestamp = datetime.datetime.now()
183     with open("activity.log", "a") as file:
184         file.write(f"{username}, {ip_address}, {timestamp}\n")
185
186 #Improved Privacy-Aware Logging
187 import hashlib
188
189 def anonymize_username(username):
190     # Hash the username to prevent direct identification
191     return hashlib.sha256(username.encode()).hexdigest()[:10]
192
193 def mask_ip(ip_address):
194     # Mask the last part of the IP address
195     parts = ip_address.split(".")
196     parts[-1] = "xxx"
197     return ".".join(parts)
198
199 def log_activity(username, ip_address):
200     timestamp = datetime.datetime.now()
201
202     anon_user = anonymize_username(username)
203     masked_ip = mask_ip(ip_address)
204
205     with open("activity.log", "a") as file:
206         file.write(f"user={anon_user}, ip={masked_ip}, time={timestamp}\n")
207
208 # Example usage
209 import datetime
210
211 Enter username: avara
Enter password: 1286
The 6th Fibonacci number is: 8
try_fibonacci file not found. Please check the filename.
file processing attempt completed.
Enter username: avara
Enter password: 1286
Invalid username or password
```