

ASSIGNMENT-5.5

ROLL-NO:2303A51932

BATCH-30

TASK-1

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

1) Naive approach 2)

Optimized approach

CODE:

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar displays a file tree with several Python files. The main editor window contains the following Python code:

```
a-5.5.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 import math
12 def is_prime_optimized(n):
13     if n <= 1:
14         return False
15     for i in range(2, int(math.sqrt(n)) + 1):
16         if n % i == 0:
17             return False
18     return True
19
20 num = 29
21 print("Naive:", is_prime_naive(num))
22 print("Optimized:", is_prime_optimized(num))
```

The terminal at the bottom shows the output of running the script:

```
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/a-5.5.py"
Naive: True
Optimized: True
PS C:\Users\Jashwanth\AI coding>
```

OBSERVATION:

The naive method checks divisibility from 2 up to $n-1$, so it performs many unnecessary iterations for large numbers.

The optimized method only checks divisibility up to \sqrt{n} , because any factor larger than \sqrt{n} must have a corresponding smaller factor already checked.

The time complexity of the naive approach is $O(n)$, which makes it slow when n becomes large.

The time complexity of the optimized approach is $O(vn)$, which significantly reduces the number of operations.

Both methods produce the same correct result, but the optimized method reaches the answer much faster.

Thus, the optimized approach improves performance by reducing redundant checks while maintaining correctness.

TASK-2

PROMPT:

Generate Python code for Fibonacci using:

- 1) Recursive method
- 2) Dynamic programming method

Explain time complexity and performance improvement.

CODE:

The screenshot shows the Microsoft Visual Studio Code interface. The Explorer sidebar on the left lists several Python files: ai-5.py, ai-5.5.py, app.log, email_samples.py, programming_prompts.py, prompt_engineering_lab4.py, and social_media_prompts.py. The main editor area contains two snippets of Python code. The first snippet, in ai-5.py, defines a recursive function fib_recursive. The second snippet, in ai-5.5.py, defines a dynamic programming function fib_dp that uses a list dp to store previously computed values. The terminal at the bottom shows the execution of both programs. The output for the recursive method shows it taking a long time for n=50, while the dynamic programming method is instantaneous. The status bar at the bottom right indicates the code is 3.12.10 (Microsoft Store).

```
ai-5.py
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40

# Recursive method
def fib_recursive(n):
    if n <= 1:
        return n
    else:
        return fib_recursive(n-1) + fib_recursive(n-2)

# Dynamic programming method
def fib_dp(n):
    dp = [0, 1]
    for i in range(2, n+1):
        dp.append(dp[i-1] + dp[i-2])
    return dp[n]

n = int(input("Enter position of Fibonacci: "))
print("Recursive:", fib_recursive(n))
print("Dynamic Programming:", fib_dp(n))

ai-5.5.py
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40

# Recursive method
def fib_recursive(n):
    if n <= 1:
        return n
    else:
        return fib_recursive(n-1) + fib_recursive(n-2)

# Dynamic programming method
def fib_dp(n):
    dp = [0, 1]
    for i in range(2, n+1):
        dp.append(dp[i-1] + dp[i-2])
    return dp[n]

n = int(input("Enter position of Fibonacci: "))
print("Recursive:", fib_recursive(n))
print("Dynamic Programming:", fib_dp(n))
```

```
PS C:\Users\Jashwanth\AI coding> & c:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.5.py"
Naive: True
Optimized: True
PS C:\Users\Jashwanth\AI coding> & c:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.py"
Enter position of Fibonacci: 5
Recursive: 5
Dynamic Programming: 5
PS C:\Users\Jashwanth\AI coding>
```

OBSERVATION:

The recursive method recomputes the same values many times. The DP method stores previous results to avoid recomputation. The recursive method has exponential time complexity. The DP method has linear time complexity. Both methods produce the same Fibonacci value. The optimized method performs much faster for large n. TASK-3

PROMPT:

Generate Python code that reads a file and processes data with proper error handling.

Explain each exception clearly using comments.

CODE:

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows files like `ai-5.5.py`, `vscode`, `ai-5.5.py`, `applog`, `email_samples.py`, `programming_prompts.py`, `prompt_engineering_lab4.py`, and `social_media_prompts.py`.
- Code Editor:** Displays the `ai-5.5.py` file with Python code for calculating Fibonacci numbers and handling file operations.
- Terminal:** Shows command-line output for running the script and interacting with the AI coding feature.
- Status Bar:** Shows the current file is `ai-5.5.py`, line 61, column 43, with 4 spaces, and the Python extension is active.

OBSERVATION:

The program clearly separates different types of errors. Each exception is handled with a meaningful message. `FileNotFoundException` explains missing file issues. `PermissionError` explains access-related problems. A general exception block handles unknown runtime errors. The explanations match the behaviour seen during execution.

TASK-4

PROMPT:

Generate a Python-based login system.

Analyze security flaws and provide a revised secure version using password hashing and input validation.

CODE:

```
File Edit Selection View Go Run Terminal Help
EXPLORER AI CODING > vscode
    a-5.5.py
    app.log
    email_samples.py
    programming_prompts.py
    prompt_engineering_lab4.py
    social_media_prompts.py
    ...
    ai-5.5.py <-
    a-5.5.py >-
    68     # Stored credentials
    69     stored_username = "admin"
    70     stored_password_hash = hashlib.sha256("mypassword".encode()).hexdigest()
    71
    72     # User input
    73     username = input("Enter username: ")
    74     password = input("Enter password: ")
    75
    76     # Hash the input password
    77     hashed_input_password = hashlib.sha256(password.encode()).hexdigest()
    78
    79     # Login validation
    80     if username == stored_username and hashed_input_password == stored_password_hash:
    81         print("Login successful")
    82     else:
    83         print("Invalid credentials")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.5.py"
Name: True
Optimized: True
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.5.py"
Enter position of Fibonacci: 5
Recursive: 5
Dynamic Programming: 5
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.5.py"
Error: File not found.
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.5.py"
Enter file name: AI Assisted Coding
Error: File not found.
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.5.py"
Enter username: Swarnali
Enter password: 1234567890
Invalid credentials
PS C:\Users\Jashwanth\AI coding> []
Ln 83, Col 33  Spaces: 4  UTF-8  CRLF  () Python  3.12.10 (Microsoft Store)  
```

OBSERVATION:

storing passwords in plain text is a serious security risk. Hashing ensures passwords are not stored in readable form. User input is validated before authentication. The system compares hashed values instead of raw passwords. This reduces the risk of password leakage. Secure authentication improves protection against attacks.

TASK-5

PROMPT:

Generate a Python script that logs user activity.

Analyze privacy risks and provide an improved version using masked or minimal logging.

CODE:

```
ai-5.5.py

91 # Get user input
92 username = input("Enter username: ")
93 ip_address = input("Enter IP address: ")
94
95 # Mask the IP address
96 masked_ip = ip_address[:3] + ".xxx.xxx"
97
98 # Get current time
99 time = datetime.datetime.now()
100
101 # Format log entry
102 log_entry = f"{username}, {masked_ip}, {time}"
103
104 # Write to log file
105 with open("activity.log.txt", "a") as file:
106     file.write(log_entry + "\n")
107
108 print("Activity logged successfully.")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.5.py"
Naive: True
Optimized: True
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.5.py"
Enter position of Fibonaccii: 5
Recursive: 5
Dynamic Programming: 5
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.5.py"
Error: File not found.
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.5.py"
Enter file name: AI Assisted Coding
AI Assisted Coding
Enter file name: AI Assisted Coding
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.5.py"
Enter username: Sewan
Enter password: 1234567890
Input error: Invalid
PS C:\Users\Jashwanth\AI coding> & C:/Users/Jashwanth/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Jashwanth/AI coding/ai-5.5.py"
Enter username: sewan
Enter IP address: 123456
Activity logged successfully.
PS C:\Users\Jashwanth\AI coding>
```

OBSERVATION:

Logging full IP addresses can expose user identity. Masking the IP reduces the risk of tracking users. Only necessary information is stored in logs. Sensitive data is not written in raw form. Minimal logging supports user privacy. Privacy-aware logging prevents misuse of stored data.