

ASSIGNMENT-5.5

NAME:AKSHITHA

HT NO:2303A51360

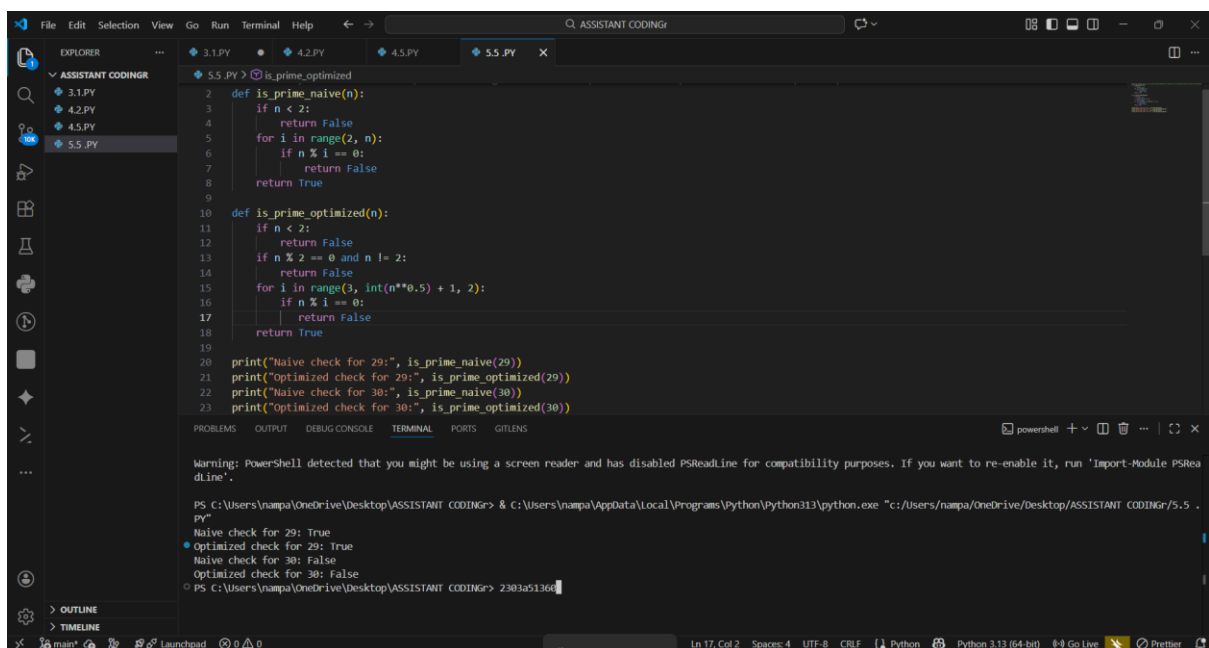
BATCH NO:29

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



```
File Edit Selection View Go Run Terminal Help
ASSISTANT CODING
5.5.PY x
2 def is_prime_naive(n):
3     if n < 2:
4         return False
5     for i in range(2, n):
6         if n % i == 0:
7             return False
8     return True
9
10 def is_prime_optimized(n):
11     if n < 2:
12         return False
13     if n % 2 == 0 and n != 2:
14         return False
15     for i in range(3, int(n**0.5) + 1, 2):
16         if n % i == 0:
17             return False
18     return True
19
20 print("Naive check for 29:", is_prime_naive(29))
21 print("Optimized check for 29:", is_prime_optimized(29))
22 print("Naive check for 30:", is_prime_naive(30))
23 print("Optimized check for 30:", is_prime_optimized(30))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT LENS
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadline for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadline'.

PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> & C:\Users\nampa\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/nampa/OneDrive/Desktop/ASSISTANT CODING/5.5 .py"
Naive check for 29: True
Optimized check for 29: True
Naive check for 30: False
Optimized check for 30: False
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> 2303a51360
```

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 $O(\sqrt{n})$, 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

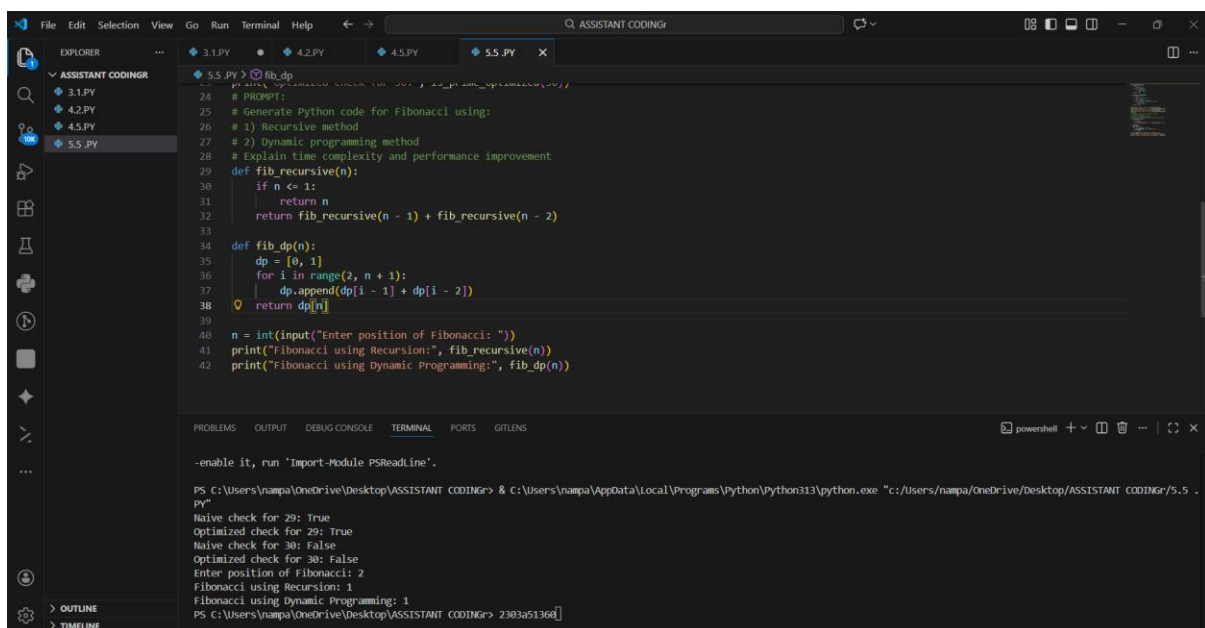
PROMPT:

Generate Python code for Fibonacci using:

- 1) Recursive method
- 2) Dynamic programming method

Explain time complexity and performance improvement

CODE:



```
24 # PROMPT:
25 # Generate Python code for Fibonacci using:
26 # 1) Recursive method
27 # 2) Dynamic programming method
28 # Explain time complexity and performance improvement
29 def fib_recursive(n):
30     if n <= 1:
31         return n
32     return fib_recursive(n - 1) + fib_recursive(n - 2)
33
34 def fib_dp(n):
35     dp = [0, 1]
36     for i in range(2, n + 1):
37         dp.append(dp[i - 1] + dp[i - 2])
38     return dp[n]
39
40 n = int(input("Enter position of Fibonacci: "))
41 print("Fibonacci using Recursion:", fib_recursive(n))
42 print("Fibonacci using Dynamic Programming:", fib_dp(n))
```

```
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr> -enable it, run 'Import-Module PSReadLine'.
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr> & c:\Users\nampa\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/nampa/OneDrive/Desktop/ASSISTANT CODINGr/5.5 .py"
Naive check for 29: True
Optimized check for 29: True
Naive check for 30: False
Optimized check for 30: False
Enter position of Fibonacci: 2
Fibonacci using Recursion: 1
Fibonacci using Dynamic Programming: 1
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr> 2303a5136d]
```

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

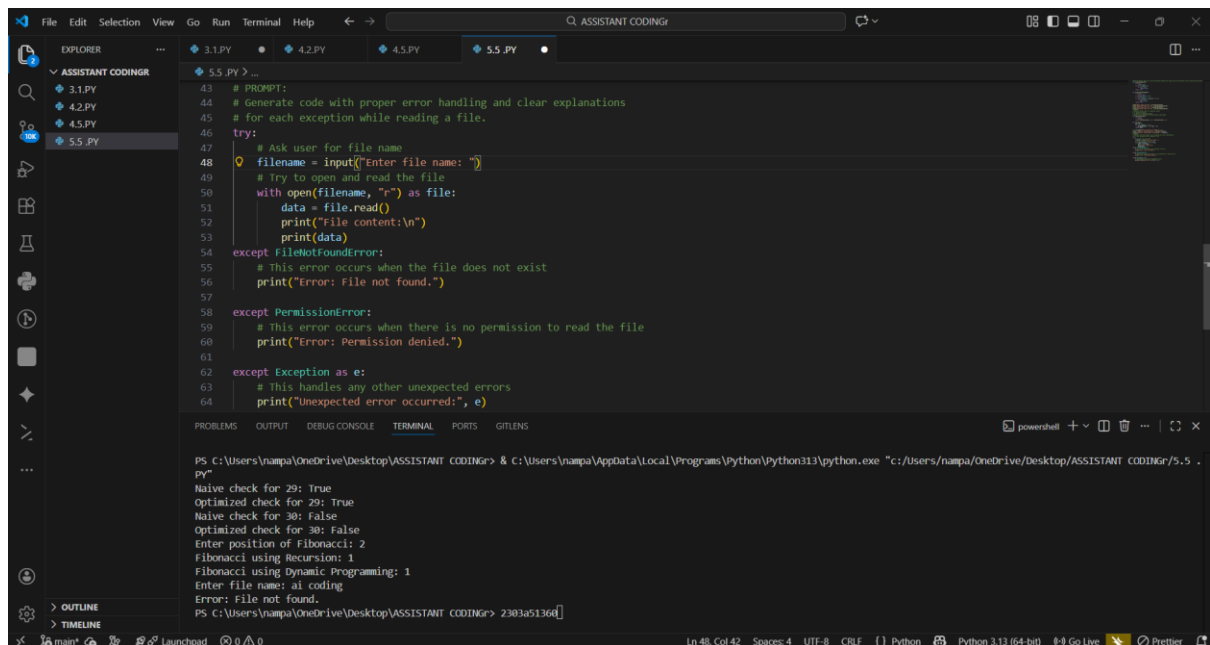
TASK-3

PROMPT:

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

Explain each exception clearly using comments

CODE:



```
43 # PROMPT:
44 # Generate code with proper error handling and clear explanations
45 # for each exception while reading a file.
46 try:
47     # Ask user for file name
48     filename = input("Enter file name: ")
49     # Try to open and read the file
50     with open(filename, "r") as file:
51         data = file.read()
52         print("File content:\n")
53         print(data)
54 except FileNotFoundError:
55     # This error occurs when the file does not exist
56     print("Error: File not found.")
57
58 except PermissionError:
59     # This error occurs when there is no permission to read the file
60     print("Error: Permission denied.")
61
62 except Exception as e:
63     # This handles any other unexpected errors
64     print("Unexpected error occurred:", e)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT LENS

```
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> & C:\Users\nampa\AppData\Local\Programs\Python\Python313\python.exe "C:/Users/nampa/OneDrive/Desktop/ASSISTANT CODING/5.5 .
py"
Naive check for 29: True
Optimized check for 29: True
Naive check for 30: False
Optimized check for 30: False
Enter position of Fibonacci: 2
Fibonacci using Recursion: 1
Fibonacci using Dynamic Programming: 1
Enter file name: ai coding
Error: File not found.
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODING> 2303a5136d
```

OBSERVATION:

The program clearly separates different types of errors.

Each exception is handled with a meaningful message.

FileNotFoundError explains missing file issues.

PermissionError explains access-related problems.

A general exception block handles unknown runtime errors.

The explanations match the behavior seen during execution.

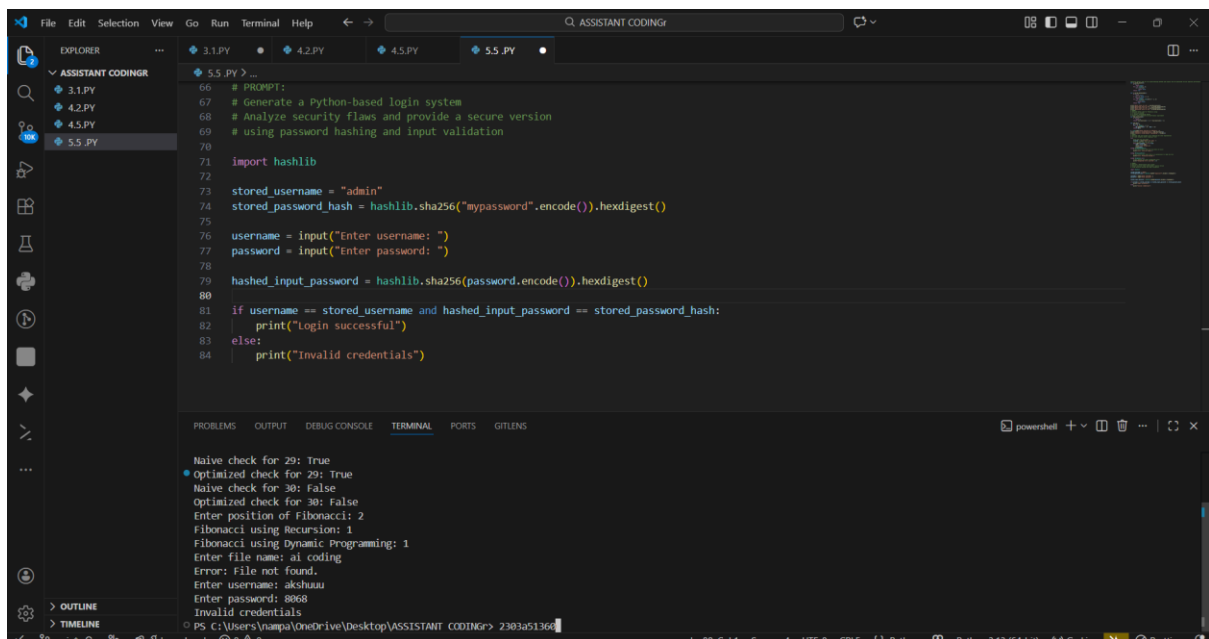
TASK-4

PROMPT:

Generate a Python-based login system.

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

CODE:



```
66 # PROMPT:
67 # Generate a Python-based login system
68 # Analyze security flaws and provide a secure version
69 # using password hashing and input validation
70
71 import hashlib
72
73 stored_username = "admin"
74 stored_password_hash = hashlib.sha256("mypassword".encode()).hexdigest()
75
76 username = input("Enter username: ")
77 password = input("Enter password: ")
78
79 hashed_input_password = hashlib.sha256(password.encode()).hexdigest()
80
81 if username == stored_username and hashed_input_password == stored_password_hash:
82     print("Login successful")
83 else:
84     print("Invalid credentials")
```

Naive check for 29: True
Optimized check for 29: True
Naive check for 30: False
Optimized check for 30: False
Enter position of Fibonacci: 2
Fibonacci using Recursion: 1
Fibonacci using Dynamic Programming: 1
Enter file name: ai coding
Error: File not found.
Enter username: akshuuu
Enter password: 8068
Invalid credentials

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.

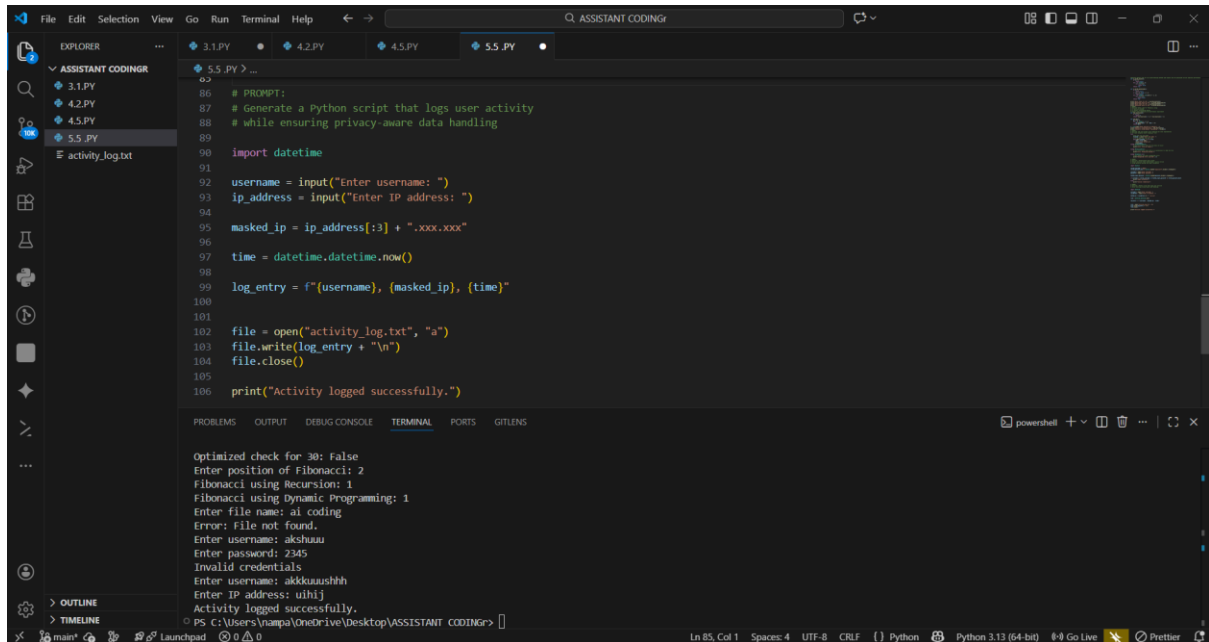
TASK-5

PROMPT:

Generate a Python script that logs user activity.

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

CODE:



```
86 # PROMPT:
87 # Generate a Python script that logs user activity
88 # while ensuring privacy-aware data handling
89
90 import datetime
91
92 username = input("Enter username: ")
93 ip_address = input("Enter IP address: ")
94
95 masked_ip = ip_address[:3] + ".xxx.xxx"
96
97 time = datetime.datetime.now()
98
99 log_entry = f"{username}, ({masked_ip}), {time}"
100
101
102 file = open("activity_log.txt", "a")
103 file.write(log_entry + "\n")
104 file.close()
105
106 print("Activity logged successfully.")
```

```
Optimized check for 30: False
Enter position of Fibonacci: 2
Fibonacci using Recursion: 1
Fibonacci using Dynamic Programming: 1
Enter file name: ai coding
Error: File not found.
Enter username: akshuuu
Enter password: 2345
Invalid credentials
Enter username: akdkuuushhh
Enter IP address: uhlj
Activity logged successfully.
PS C:\Users\Nampa\OneDrive\Desktop\ASSISTANT 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.