

# ASSIGNMENT-5.5

ROLL-NO:2306A91001

BATCH-30

# **TASK-1**

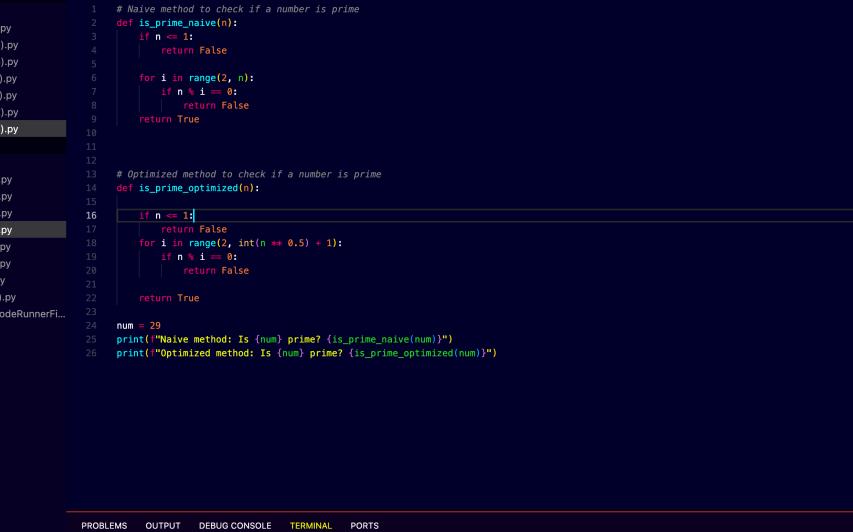
PROMPT: Generate Python code for two prime-checking methods and

explain how the optimised version improves performance

Generate Python code for two prime-checking methods:

- 1) Naive approach
  - 2) Optimised approach

## CODE:



The screenshot shows a Python code editor with the following code in the file `ai(5.5).py`:

```
# Naive method to check if a number is prime
def is_prime_naive(n):
    if n <= 1:
        return False

    for i in range(2, n):
        if n % i == 0:
            return False
    return True

# Optimized method to check if a number is prime
def is_prime_optimized(n):

    if n <= 1:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True

num = 29
print("Naive method: Is {} prime? {}".format(num, is_prime_naive(num)))
print("Optimized method: Is {} prime? {}".format(num, is_prime_optimized(num)))
```

The code editor interface includes a sidebar with project files, a terminal at the bottom showing the execution of the script, and a status bar at the bottom right.

## OBSERVATION:

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

The optimised 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 optimised approach is  $O(\sqrt{n})$ , which significantly reduces the number of operations.

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

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

## TASK-2

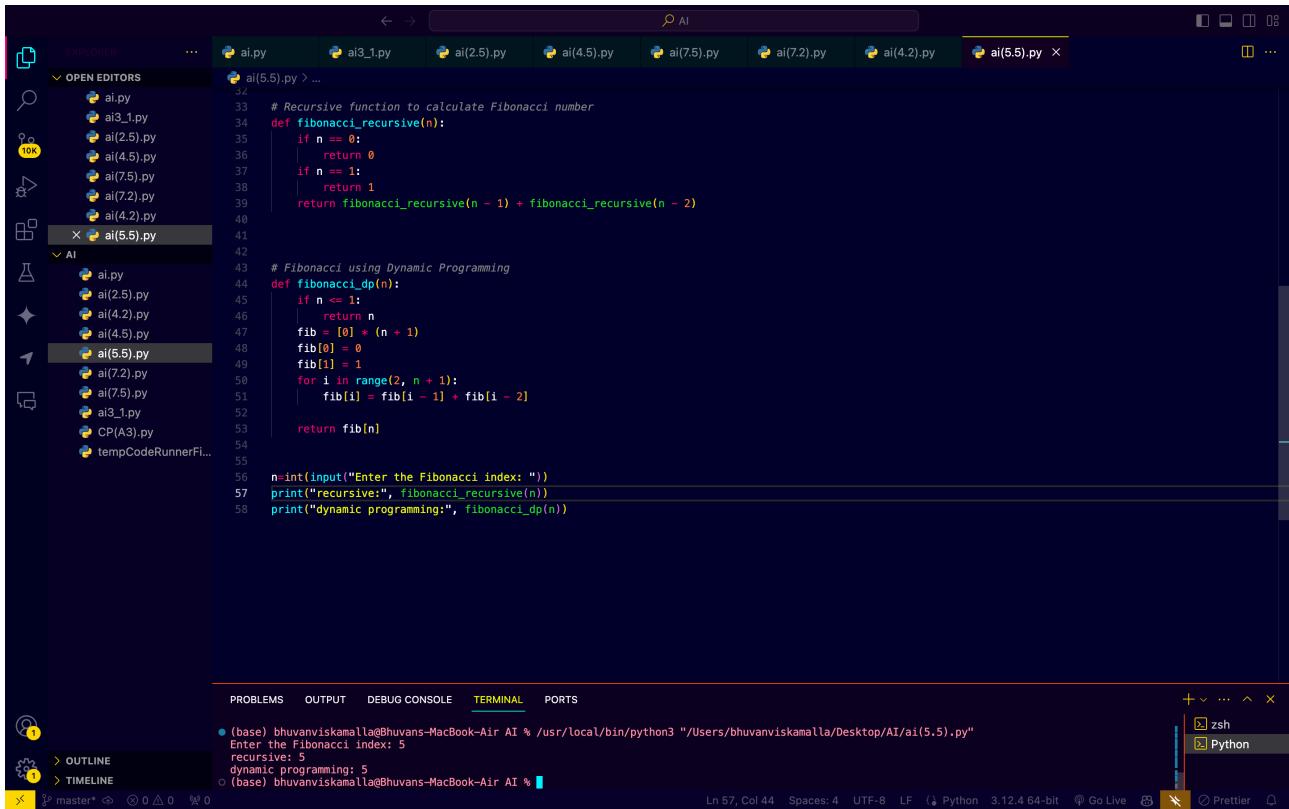
### PROMPT:

Generate Python code for Fibonacci using:

- 1) Recursive method
- 2) Dynamic programming method

# Explain time complexity and performance improvement.

## CODE:



```
32 # Recursive function to calculate Fibonacci number
33 def fibonacci_recursive(n):
34     if n == 0:
35         return 0
36     if n == 1:
37         return 1
38     return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2)
39
40
41
42
43 # Fibonacci using Dynamic Programming
44 def fibonacci_dp(n):
45     if n <= 1:
46         return n
47     fib = [0] * (n + 1)
48     fib[0] = 0
49     fib[1] = 1
50     for i in range(2, n + 1):
51         fib[i] = fib[i - 1] + fib[i - 2]
52
53     return fib[n]
54
55
56 n=int(input("Enter the Fibonacci index: "))
57 print("recursive:", fibonacci_recursive(n))
58 print("dynamic programming:", fibonacci_dp(n))
```

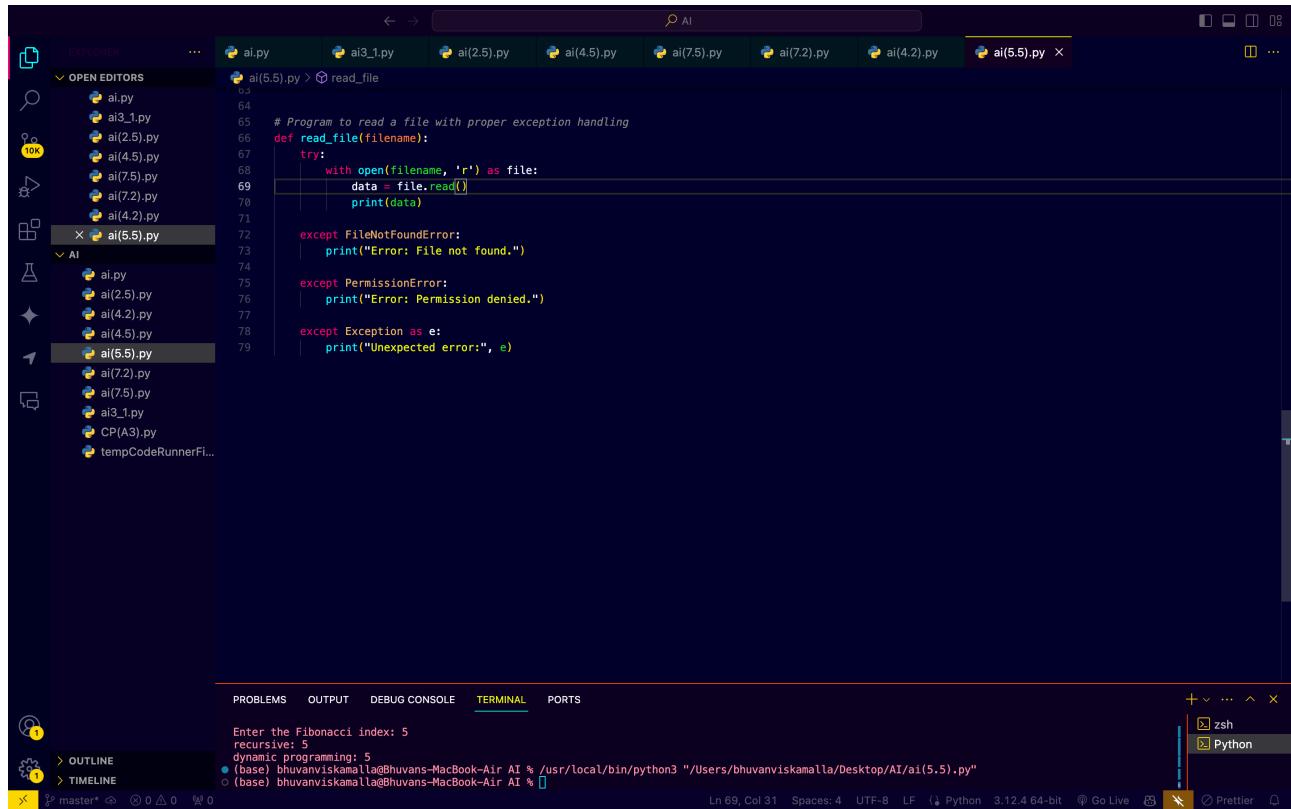
## OBSERVATION:

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 optimised 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 VS Code interface with the following details:

- EXPLORER:** Shows a list of files in the AI folder, including ai.py, ai3\_1.py, ai(2.5).py, ai(4.5).py, ai(7.5).py, ai(7.2).py, ai(4.2).py, and ai(5.5).py. The file ai(5.5).py is currently open.
- OPEN EDITORS:** The file ai(5.5).py is the active editor, displaying the following Python code:

```
# Program to read a file with proper exception handling
def read_file(filename):
    try:
        with open(filename, 'r') as file:
            data = file.read()
            print(data)

    except FileNotFoundError:
        print("Error: File not found.")

    except PermissionError:
        print("Error: Permission denied.")

    except Exception as e:
        print("Unexpected error:", e)
```

- TERMINAL:** Shows the command entered and its output:

```
Enter the Fibonacci index: 5
recursive: 5
dynamic programming: 5
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvanviskamalla/Desktop/AI/ai(5.5).py"
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI %
```

OBSERVATION:

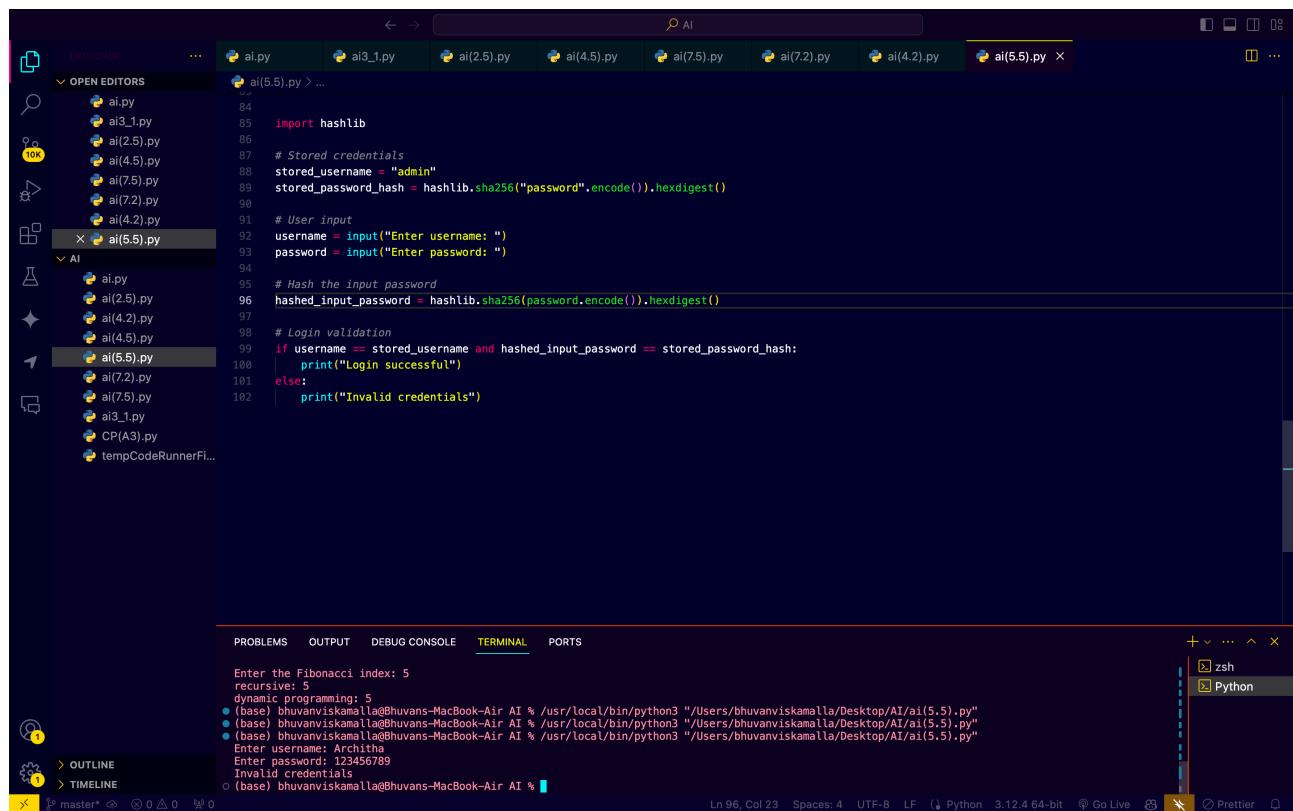
The program clearly separates different types of errors. Each exception is handled with a meaningful message. FileNotFoundError explains missing file issues. PermissionError explains access-related problems. A general exception block handles unknown runtime errors. The explanations match the behaviour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

# TASK-4

## PROMPT:

Generate a Python-based login system.  
Analyse security flaws and provide a revised secure version using password hashing and input validation.

## CODE:



The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows multiple Python files in the AI folder, including ai.py, ai3\_1.py, ai(2.5).py, ai(4.5).py, ai(7.5).py, ai(7.2).py, ai(4.2).py, and ai(5.5).py. The file ai(5.5).py is currently open.
- Code Editor:** Displays the content of ai(5.5).py. The code is as follows:

```
84 import hashlib
85
86 # Stored credentials
87 stored_username = "admin"
88 stored_password_hash = hashlib.sha256("password".encode()).hexdigest()
89
90 # User input
91 username = input("Enter username: ")
92 password = input("Enter password: ")
93
94 # Hash the input password
95 hashed_input_password = hashlib.sha256(password.encode()).hexdigest()
96
97 # Login validation
98 if username == stored_username and hashed_input_password == stored_password_hash:
99     print("Login successful")
100 else:
101     print("Invalid credentials")
```

- Terminal:** Shows a terminal session with the following output:

```
Enter the Fibonacci index: 5
dynamic programming: 5
● (base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvanviskamalla/Desktop/AI/ai(5.5).py"
● (base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvanviskamalla/Desktop/AI/ai(5.5).py"
● (base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvanviskamalla/Desktop/AI/ai(5.5).py"
Enter username: Architha
Enter password: 123456789
Invalid credentials
● (base) bhuvanviskamalla@Bhuvans-MacBook-Air AI %
```

## 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

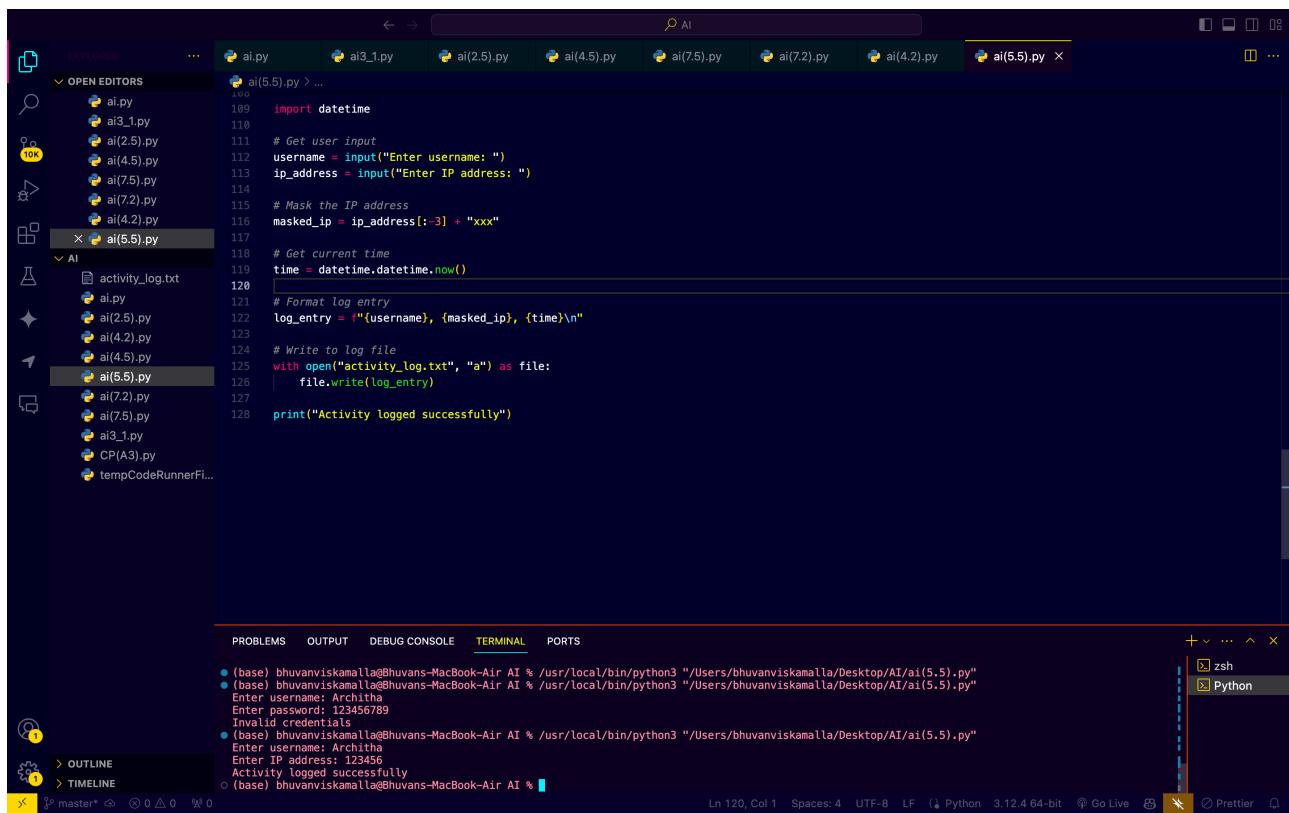
compares hashed values instead of raw passwords. This reduces the risk of password leakage. Secure authentication improves protection against attacks.

## TASK-5

### PROMPT:

Generate a Python script that logs user activity. Analyse privacy risks and provide an improved version using masked or minimal logging.

### CODE:



```
100 import datetime
101
102 # Get user input
103 username = input("Enter username: ")
104 ip_address = input("Enter IP address: ")
105
106 # Mask the IP address
107 masked_ip = ip_address[:-3] + "xxx"
108
109 # Get current time
110 time = datetime.datetime.now()
111
112 # Format log entry
113 log_entry = f"{username}, {masked_ip}, {time}\n"
114
115 # Write to log file
116 with open("activity_log.txt", "a") as file:
117     file.write(log_entry)
118
119 print("Activity logged successfully")
```

The terminal output shows:

```
(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvaniskamalla/Desktop/AI/ai(5.5).py"
Enter username: Architha
Enter password: 123456789
Invalid credentials
(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvaniskamalla/Desktop/AI/ai(5.5).py"
Enter username: Architha
Enter IP address: 123456
Activity logged successfully
(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI %
```

## 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.