

Ai Assistant Coding

Week:5.5

Ht.no:2303A52492

Batch:50

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.”

Code:

- Naive approach(basic)

#Naive method

```
import math
```

```
def is_prime_basic(n):
```

```
    if n <= 1:
```

```
        return False
```

```
    for i in range(2, n):
```

```
        if n % i == 0:
```

```
            return False
```

```
    return True
```

- Optimized approach

#Optimized version

```
def is_prime_optimized(n):
```

```
    if n <= 1:
```

```

        return False

    if n == 2:
        return True

    if n % 2 == 0:
        return False

    for i in range(3, int(math.sqrt(n)) + 1, 2):
        if n % i == 0:
            return False

    return True

num = int(input("Enter a number: "))

print("Basic method result:", is_prime_basic(num))
print("Optimized method result:", is_prime_optimized(num))

```

Output:

```

'51810' '--' 'C:\Users\mouni\OneDrive\Desktop\Ai Ass labs\Ai Ass 5.5.py'
Enter a number: 3
Basic method result: True
Optimized method result: True

```

Explanation:

The naive approach code checks whether the given number is prime or non-prime number.

Time Complexity:

O(n) — slow for large numbers.

The optimized version code skips even numbers and stops checking after the square root of the number.

Time Complexity:

O(\sqrt{n}) — much faster.

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.

Prompt:

Generate a python code to generate a recursive function to calculate the Fibonacci numbers.

Code:

```
def fibonacci(n):  
    # Base case 1:  
    # If n is 0, Fibonacci value is 0  
    # This stops further recursive calls  
    if n == 0:  
        return 0  
  
    # Base case 2:  
    # If n is 1, Fibonacci value is 1  
    # This also stops recursion  
    if n == 1:  
        return 1  
  
    # Recursive case:  
    # The function calls itself to find  
    # the Fibonacci values of the two  
    # previous numbers  
    return fibonacci(n - 1) + fibonacci(n - 2)  
  
# Take input from the user  
n = int(input("Enter a number: "))  
  
# Call the recursive function and print the result  
print("Fibonacci number is:", fibonacci(n))
```

Output:

```
C:\...\VSCode\extensions\ms-python.debugpy\2023.10.0-win32-x86\Bundled\11  
'64592' '--' 'C:\Users\mouni\OneDrive\Desktop\Ai Ass labs\Ai Ass 5.5.py  
Enter a number: 4  
Fibonacci number is: 3
```

Explanation:

In base cases 1 and 2 we have given value 0 and 1 so it stops the recursive calls and in the recursive case the function calls itself to find the fibonacci values of the two previous numbers.

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.”

Code:

```
def calculate_average(filename):  
    try:  
        # Step 1: Try to open the file  
        with open(filename, "r") as file:  
            lines = file.readlines()  
        # Step 2: Check if file is empty  
        if not lines:  
            raise ValueError("File is empty. No data to process.")  
        numbers = []  
        # Step 3: Convert each line into an integer  
        for line in lines:  
            numbers.append(int(line.strip()))  
    # Step 4: Calculate average
```

```

average = sum(numbers) / len(numbers)
    print("Average of numbers:", average)
except FileNotFoundError:
    # Triggered when file does not exist
    print("Error: File not found. Please check the file name or path.")
except ValueError as ve:
    # Triggered for:
    # 1) Non-numeric data
    # 2) Empty file (manually raised)
    print("Error:", ve)
except ZeroDivisionError:
    # Triggered if division by zero occurs
    print("Error: Cannot divide by zero. No valid numbers found.")
except Exception as e:
    # Handles any unexpected errors
    print("Unexpected error occurred:", e)
else:
    # Runs only if no exception occurs
    print("File processed successfully.")
finally:
    # Always executes
    print("Program execution completed.")

# ---- Program Execution ----

filename = input("Enter file name: ")
calculate_average(filename)

```

Output:

```

...
'56554' '--' 'C:\Users\mouni\OneDrive\Desktop\Ai Ass labs\Ai Ass 5.5.py'
Enter file name: DevOps
Error: File not found. Please check the file name or path.
Program execution completed.
○ PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>

```

Explanation:

If the file is empty or there is no file with the name we have given, it clearly shows that file not found. If there is the correct file uploaded we will get the average in all cases.

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.

Prompt:

Generate a code in python based login system and check whether that the Ai uses secure password handling.

Code:

```
import hashlib
import getpass
import re

user_db = {} # stores username: hashed_password

def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def is_strong_password(password):
    """
    """
```

Password validation rules:

- Minimum 8 characters
 - At least 1 uppercase, 1 lowercase, 1 digit
 - At least 1 special character
- ```
"""

```

```
if len(password) < 8:
 return False

if not re.search(r"[A-Z]", password):
```

```
 return False

if not re.search(r"[a-z]", password):
 return False

if not re.search(r"\d", password):
 return False

if not re.search(r"[@#$%^&*]", password):
 return False

return True

def register():

 username = input("Enter new username: ")

 if username in user_db:

 print("Username already exists. Try login.")

 return

 password = getpass.getpass("Enter password: ")

 if not is_strong_password(password):

 print("Weak password! Must be 8+ chars, mix of
upper/lower/digit/special.")

 return

 confirm = getpass.getpass("Confirm password: ")

 if password != confirm:

 print("Passwords do not match!")

 return

 user_db[username] = hash_password(password)

 print(f'User {username} registered successfully!')

def login():

 username = input("Enter username: ")

 if username not in user_db:

 print("User not found. Register first.")

 return
```

```
password = getpass.getpass("Enter password: ")

if user_db[username] == hash_password(password):
 print(f"Welcome, {username}! Login successful.")

else:
 print("Incorrect password. Try again.")

while True:

 choice = input("\n[register/login/exit]: ").lower()

 if choice == "register":

 register()

 elif choice == "login":

 login()

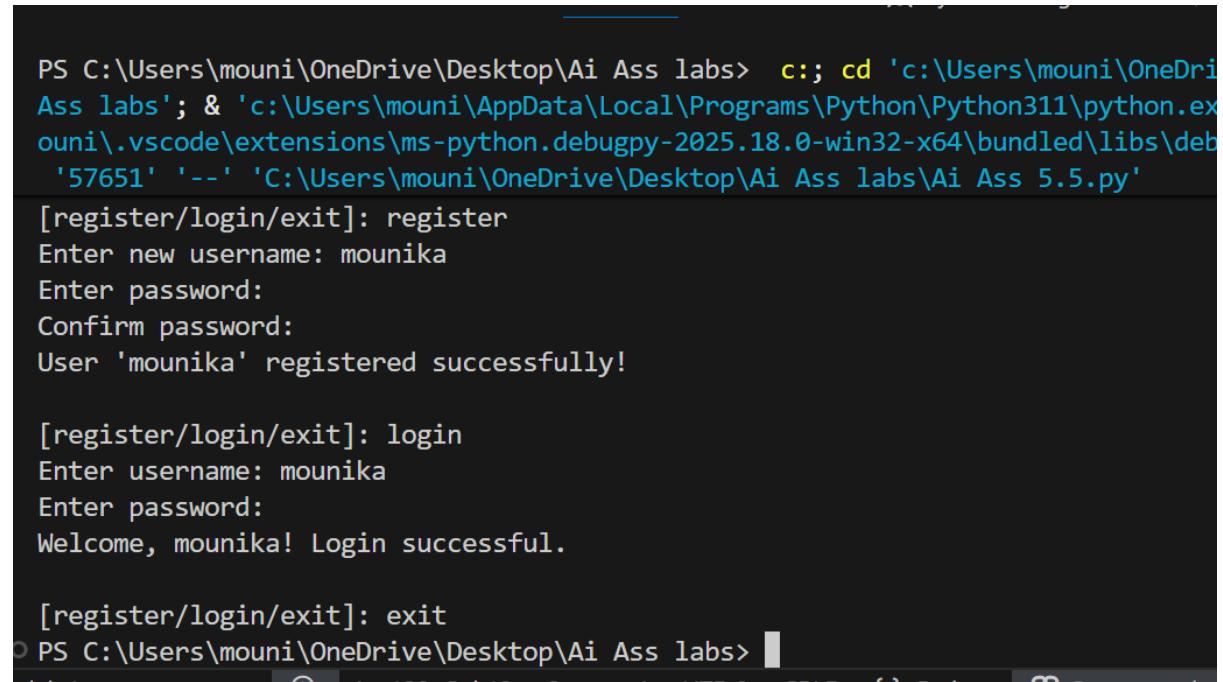
 elif choice == "exit":

 break

 else:

 print("Invalid option.")
```

### **Output:**



```
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs> c:; cd 'c:\Users\mouni\OneDri
Ass labs'; & 'c:\Users\mouni\AppData\Local\Programs\Python\Python311\python.ex
ouini\.vscode\extensions\ms-python.python.debugpy-2025.18.0-win32-x64\bundled\libs\deb
'57651' '--' 'C:\Users\mouni\OneDrive\Desktop\Ai Ass labs\Ai Ass 5.5.py'
[register/login/exit]: register
Enter new username: mounika
Enter password:
Confirm password:
User 'mounika' registered successfully!

[register/login/exit]: login
Enter username: mounika
Enter password:
Welcome, mounika! Login successful.

[register/login/exit]: exit
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>
```

### **Explanation:**

When registering, the program checks if the password is strong and hides input; it shows success only if all checks pass.

During login, it verifies the hashed password and welcomes the user if correct, or shows an error if the username or password is wrong.

### **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.

#### **Prompt:**

Generate a code in python script that logs user activity (username, IP address, timestamp).Examine whether sensitive data is logged unnecessarily or insecurely.

#### **Code:**

```
import logging
from datetime import datetime
import socket
import hashlib
Configure logging
logging.basicConfig(
 filename='user_activity_safe.log',
 level=logging.INFO,
 format='%(asctime)s - %(user_id)s - %(ip_masked)s - %(action)s')
def get_ip_masked():
 try:
 hostname = socket.gethostname()
 ip = socket.gethostbyname(hostname)
 # Mask last octet for privacy
 parts = ip.split('.')
 parts[-1] = 'xxx'
```

```

return '.'.join(parts)

except:
 return "Unknown IP"

def anonymize_username(username):
 # Hash the username to avoid storing real name
 return hashlib.sha256(username.encode()).hexdigest()[:8] # first 8 chars

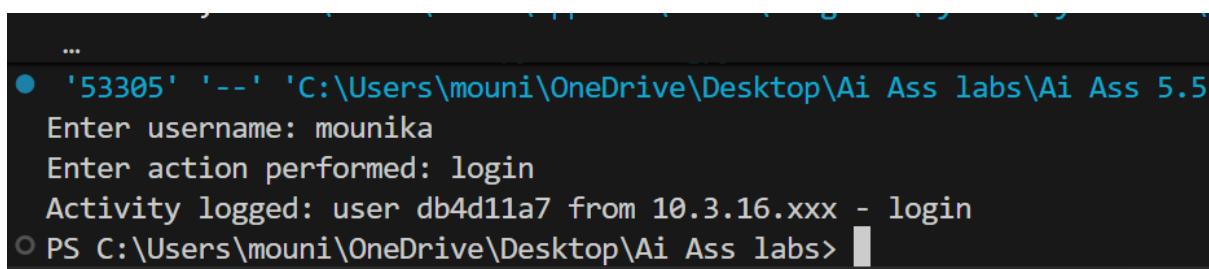
def log_activity(username, action):
 user_id = anonymize_username(username)
 ip_masked = get_ip_masked()
 # Use 'action' instead of 'message' to avoid conflict
 logging.info("", extra={'user_id': user_id, 'ip_masked': ip_masked, 'action': action})
 print(f'Activity logged: user {user_id} from {ip_masked} - {action}')

Example usage

username = input("Enter username: ")
action = input("Enter action performed: ")
log_activity(username, action)

```

### **Output:**



```

...
● '53305' '--' 'C:\Users\mouni\OneDrive\Desktop\Ai Ass labs\Ai Ass 5.5
Enter username: mounika
Enter action performed: login
Activity logged: user db4d11a7 from 10.3.16.xxx - login
○ PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs> █

```

### **Explanation:**

The code logs user activity safely by asking usernames and masking IP addresses, avoiding any direct personal identification. It also fixes a logging error by using a custom key action instead of message.

