ASSIGNMENT-5.5

2303A51927

BATCH-30

Task-1 (Transparency in Algorithm Optimization)

Task: Use AI to generate two solutions for checking prime numbers:

• Naive approach(basic)

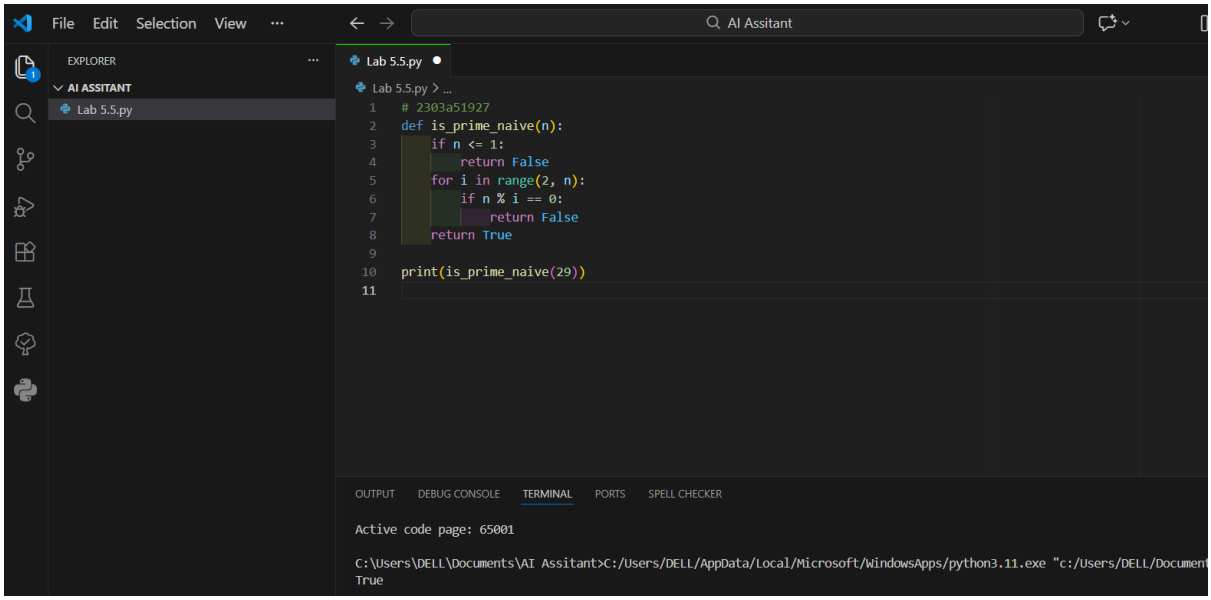
• Optimized approach

Prompt:

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

CODE AND OUTPUT:

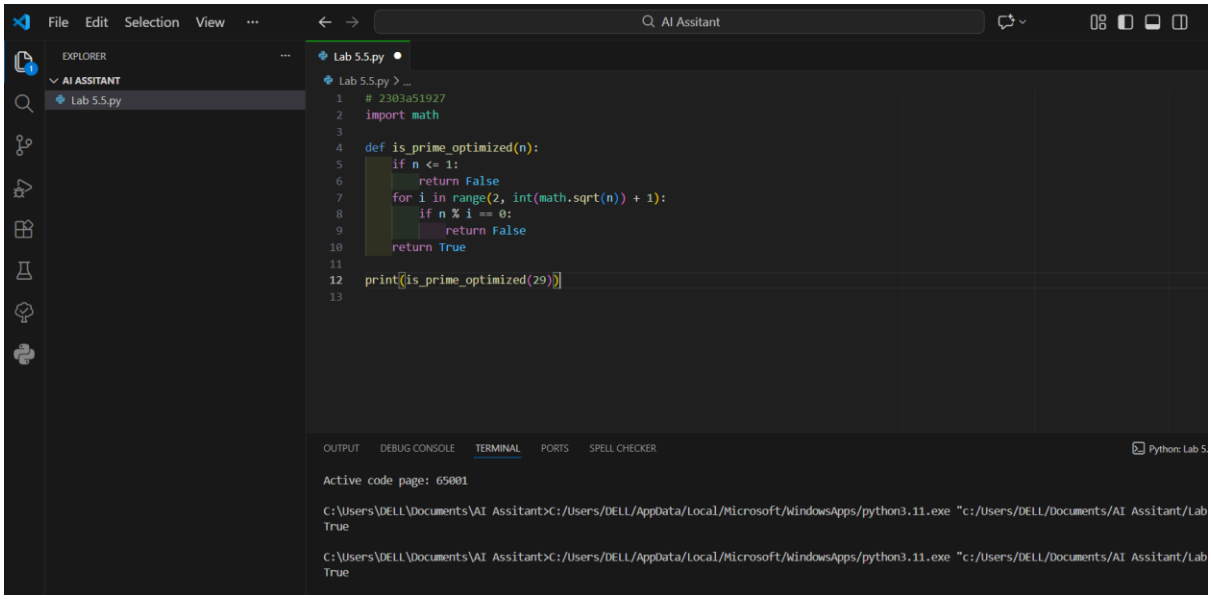
Naive Approach:



Optimized Approach:

Justification

AI generated two different approaches to check prime numbers.
The naive method checks all numbers, while the optimized method reduces unnecessary iterations by checking only up to the square root of the number.
This shows how AI can improve algorithm efficiency and transparency in optimization.



Task 2: Transparency in Recursive Algorithms (Fibonacci)

Prompt Used:

Generate a recursive Python function to calculate Fibonacci numbers with clear comments and explanation of base cases and recursive calls.

CODE AND OUTPUT:

Justification:

AI-generated recursive code clearly shows base cases and recursive calls.
This helps in understanding how recursion works step-by-step.
Transparency in recursive logic ensures that the program behavior matches the explanation provided by AI.

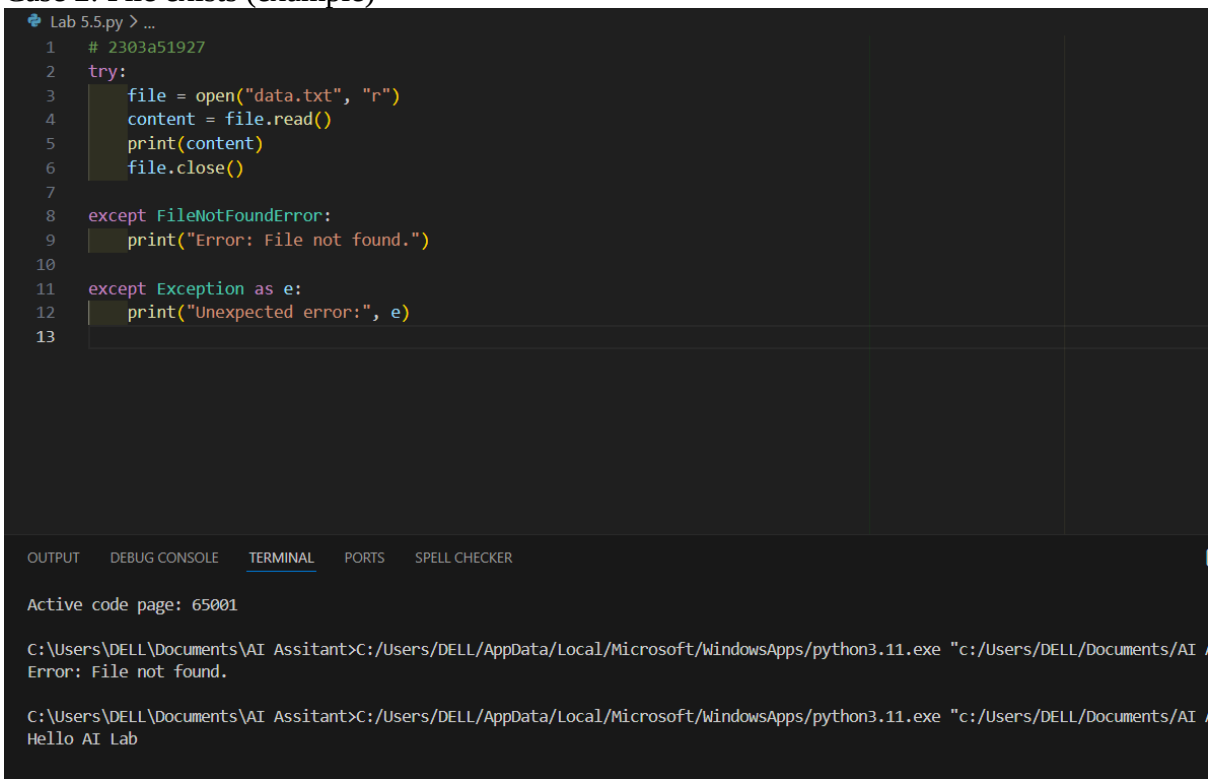
Task 3: Transparency in Error Handling (File Handling)

Prompt Used: Generate a Python program that reads a file and includes proper error handling with explanations for each exception.

CODE AND OUTPUT:

Case 1: File not found

Case 2: File exists (example)



Justification

AI generated code with meaningful exception handling.
Different types of errors are handled separately, which improves program reliability and transparency.
This ensures that the program behaves safely during runtime errors.

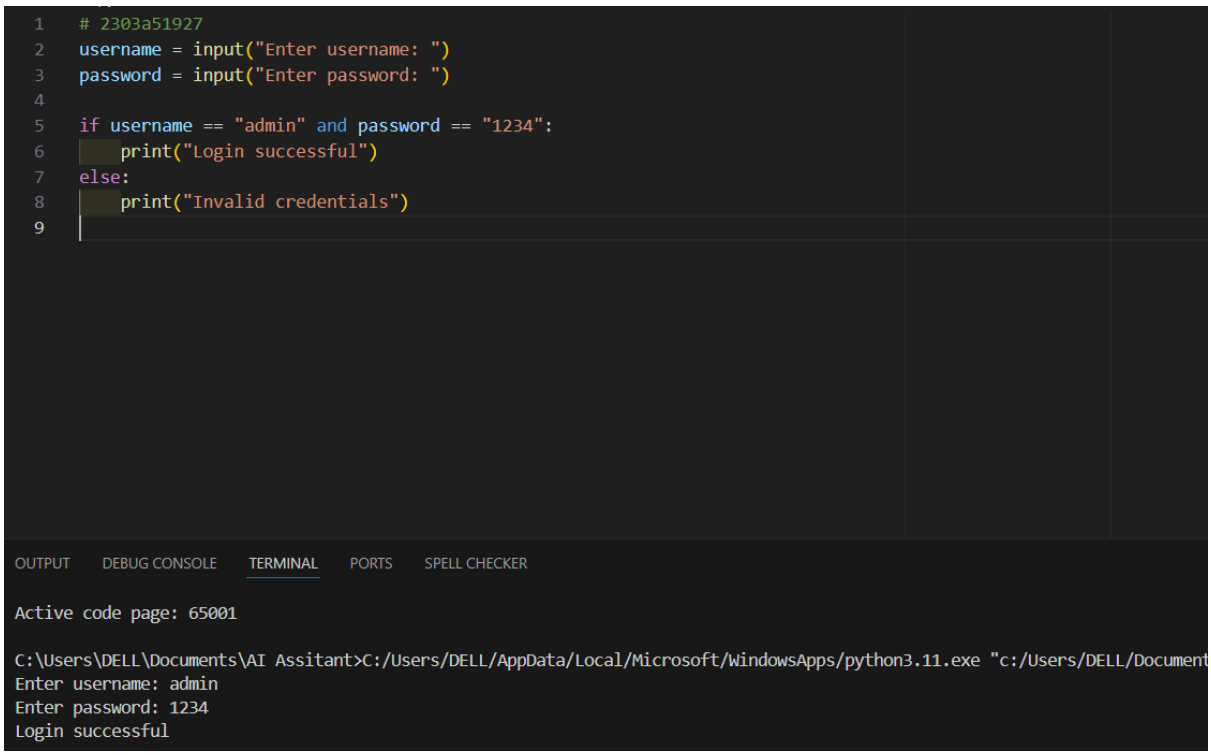
Task 4: Security in User Authentication

Prompt Used

Generate a Python-based login system and analyze security issues. Then improve it using secure password handling techniques.

CODE AND OUTPUT:

Insecure Login System



Secure Login System:

```python
# 2303a51927
import hashlib

stored_password = hashlib.sha256("1234".encode()).hexdigest()

username = input("Enter username: ")
password = input("Enter password: ")

hashed_password = hashlib.sha256(password.encode()).hexdigest()

if username == "admin" and hashed_password == stored_password:
    print("Login successful")
else:
    print("Invalid credentials")

```

```
OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SPELL CHECKER

Active code page: 65001

C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe
Enter username: admin
Enter password: 1234
Login successful
```

Justification

AI initially generated a simple login system that stores passwords in plain text, which is insecure.
By improving the code with password hashing, security risks are reduced.
This demonstrates the importance of reviewing AI-generated code for security vulnerabilities.

Task 5: Privacy in Data Logging

Prompt Used

Generate a Python script that logs user activity and analyze privacy risks. Then improve it using privacy-aware techniques.

CODE AND OUTPUT:

Code 1: Normal Logging (Privacy Risk)

```python
# 2303a51927
import datetime

username = input("Enter username: ")
ip_address = "192.168.1.10"   # Example IP address

with open("log.txt", "a") as file:
    file.write(f"{username}, {ip_address}, {datetime.datetime.now()}\n")

print("User activity logged successfully.")
```

```
OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SPELL CHECKER

Active code page: 65001

C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/DELL/Docu
Enter username: varsha
User activity logged successfully.
```

Code 2: Improved Privacy-Safe Logging

```python
# 2303a51927 Lab 5.5: Privacy-Safe Logging
import datetime
import hashlib

username = input("Enter username: ")

# Masking username using hashing
masked_user = hashlib.sha256(username.encode()).hexdigest()[:8]

with open("log.txt", "a") as file:
    file.write(f"{masked_user}, {datetime.datetime.now()}\n")

print("Privacy-safe logging completed.")
```

```
OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SPELL CHECKER        Python: Lab 5.5

Active code page: 65001

C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/DELL/Documents/AI Assitant/Lab 5.5.py"
Enter username: varsha
User activity logged successfully.

C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/DELL/Documents/AI Assitant/Lab 5.5.py"
Enter username: varsha
Privacy-safe logging completed.
```

Lab 5.5.py ● | log.txt | data.txt

log.txt

```
1  varsha, 192.168.1.10, 2026-01-30 20:39:21.669395
2  17f2f99c, 2026-01-30 20:40:58.447041
3
```

Justification

AI-generated logging code initially stored sensitive data like username and IP address, which can violate privacy.
By masking user information, privacy risks are minimized.
This highlights the importance of ethical and privacy-aware coding practices.