Task Description #1 (Transparency in Algorithm Optimization)

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

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

Code:-

```
def is_prime_basic(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
# Method 2: Optimized prime-checking method
def is_prime_optimized(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True
```

output :

Task Description #2 (Transparency in Recursive Algorithms)

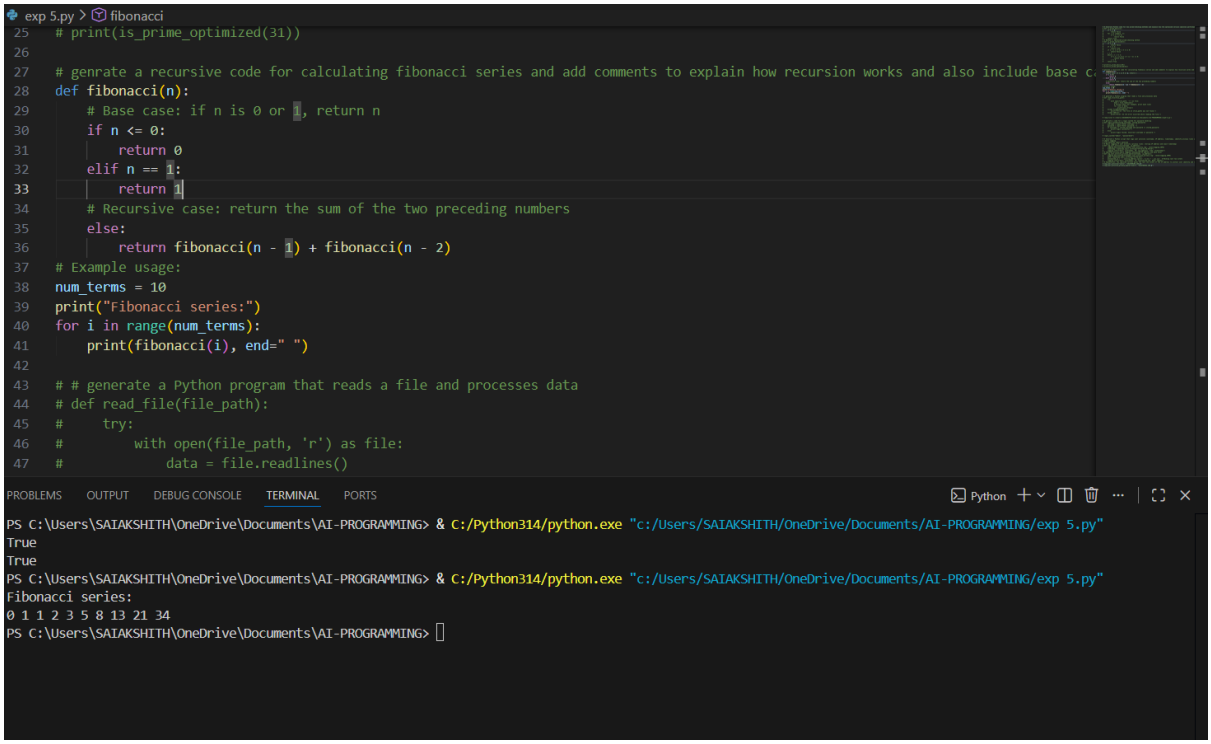
Objective: Use AI to generate a recursive function to calculate Fibonacci numbers.

Code :-

genrate a recursive code for calculating fibonacci series and add comments to explain how recursion works and also include base case and recursive case in comments.

```
def fibonacci(n):
    # Base case: if n is 0 or 1, return n
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    # Recursive case: return the sum of the two preceding numbers
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
# Example usage:
num_terms = 10
print("Fibonacci series:")
for i in range(num_terms):
    print(fibonacci(i), end=" ")
```

output:-



Task Description #3 (Transparency in Error Handling)

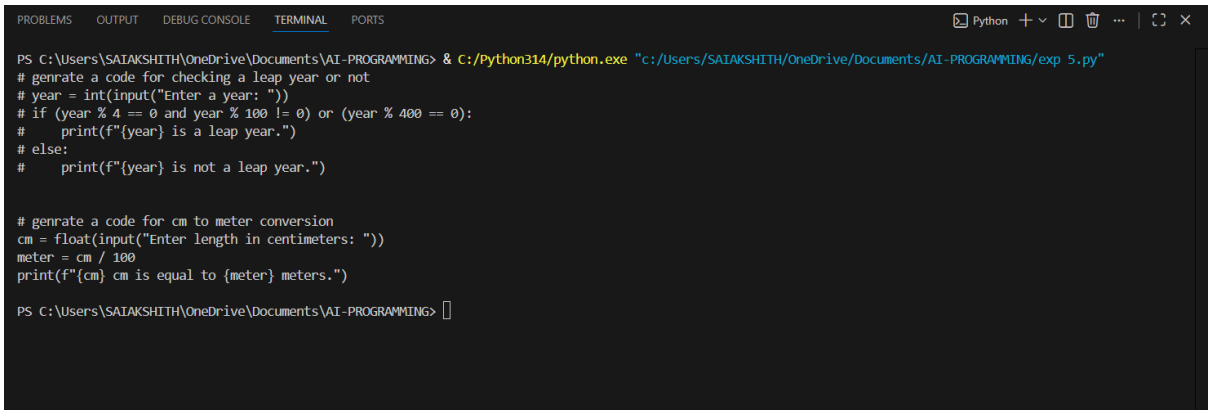
Task: Use AI to generate a Python program that reads a file and processes data.

Code:-

```
# generate a Python program that reads a file and processes data
def read_file(file_path):
    try:
        with open(file_path, 'r') as file:
            data = file.readlines()
            # Process data (for example, print each line)
            for line in data:
                print(line.strip())
    except FileNotFoundError:
        print(f"Error: The file at {file_path} was not found.")
    except IOError:
        print("Error: An I/O error occurred while reading the file.")
```

read_file('C:\\Users\\SAIAKSHITH\\OneDrive\\Documents\\AI-PROGRAMMING\\exp4-3.py')

output:-



Task Description #4 (Security in User Authentication)

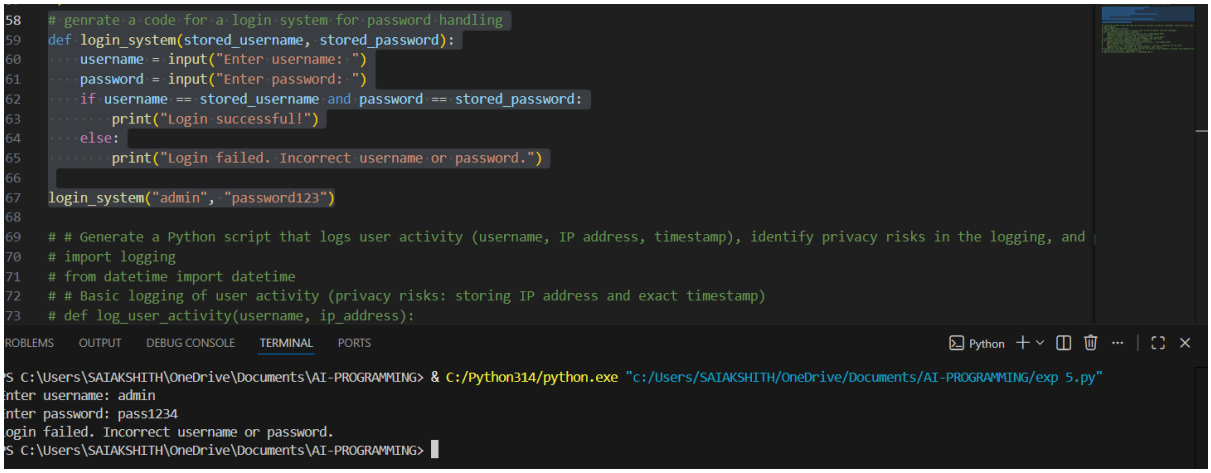
Task: Use an AI tool to generate a Python-based login system.

Code:-

```
# genrate a code for a login system for password handling
def login_system(stored_username, stored_password):
    username = input("Enter username: ")
    password = input("Enter password: ")
    if username == stored_username and password == stored_password:
        print("Login successful!")
    else:
        print("Login failed. Incorrect username or password.")

login_system("admin", "password123")
```

output:-



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).

Code:-

Generate a Python script that logs user activity (username, IP address, timestamp), identify privacy risks in the logging, and provide a privacy-aware improved version using minimal, anonymized, or masked data with a brief explanation.

```
import logging

from datetime import datetime

# Basic logging of user activity (privacy risks: storing IP address and exact timestamp)

def log_user_activity(username, ip_address):

    logging.basicConfig(filename='user_activity.log', level=logging.INFO)

    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    logging.info(f"User: {username}, IP: {ip_address}, Time: {timestamp}")

# Improved privacy-aware logging (anonymized IP address and date only)

def log_user_activity_privacy_aware(username, ip_address):

    logging.basicConfig(filename='user_activity_privacy.log', level=logging.INFO)

    date = datetime.now().strftime("%Y-%m-%d")

    anonymized_ip = '.'.join(ip_address.split('.')[:2]) + '.xxx.xxx'  # Masking last two octets

    logging.info(f"User: {username}, IP: {anonymized_ip}, Date: {date}")
```

Explanation: The improved version masks the last two octets of the IP address to protect user identity and logs only the date instead of the exact timestamp to reduce traceability.

log_user_activity("user1", "2514789192.168.1")

log_user_activity_privacy_aware("user1", "2514789192.168.1")

output:-