

Task Description #1 (Transparency in Algorithm Optimization)

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

Prompt: 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 :



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\SAIAKSHITH\OneDrive\Documents\AI-PROGRAMMING> & C:/Python314/python.exe "c:/Users/SAIAKSHITH/OneDrive/Documents/AI-PROGRAMMING/exp 5.py"
True
True
PS C:\Users\SAIAKSHITH\OneDrive\Documents\AI-PROGRAMMING> |
```

Task Description #2 (Transparency in Recursive Algorithms)

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

Code :-

generate 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:-

```
exp 5.py > fibonacci
25 # print(is_prime_optimized(31))
26
27 # generate a recursive code for calculating fibonacci series and add comments to explain how recursion works and also include base c
28 def fibonacci(n):
29     # Base case: if n is 0 or 1, return n
30     if n <= 0:
31         return 0
32     elif n == 1:
33         return 1
34     # Recursive case: return the sum of the two preceding numbers
35     else:
36         return fibonacci(n - 1) + fibonacci(n - 2)
37 # Example usage:
38 num_terms = 10
39 print("Fibonacci series:")
40 for i in range(num_terms):
41     print(fibonacci(i), end=" ")
42
43 # generate a Python program that reads a file and processes data
44 # def read_file(file_path):
45 #     try:
46 #         with open(file_path, 'r') as file:
47 #             data = file.readlines()
```

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:

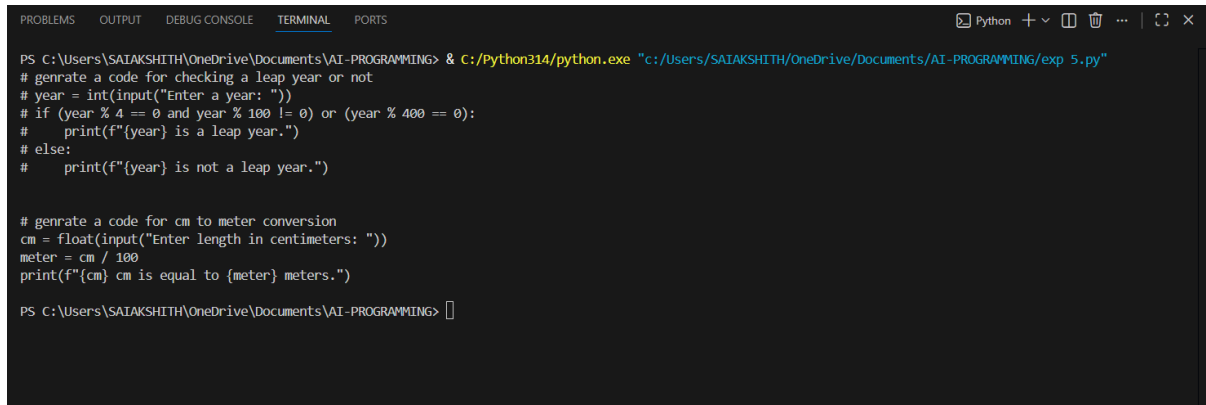
print(f"Error: The file at {file_path} was not found.")

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:-

A screenshot of a Python terminal window. The window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing a command prompt where a Python script is being executed. The script contains two functions: one for checking if a year is a leap year and another for converting centimeters to meters. The output of the script is visible in the terminal.

```
PS C:\Users\SAIAKSHITH\OneDrive\Documents\AI-PROGRAMMING> & c:/Python314/python.exe "c:/Users/SAIAKSHITH/OneDrive/Documents/AI-PROGRAMMING/exp 5.py"
# genrate a code for checking a leap year or not
# year = int(input("Enter a year: "))
# if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
#     print(f"{year} is a leap year.")
# else:
#     print(f"{year} is not a leap year.")

# genrate a code for cm to meter conversion
cm = float(input("Enter length in centimeters: "))
meter = cm / 100
print(f"{cm} cm is equal to {meter} meters.")

PS C:\Users\SAIAKSHITH\OneDrive\Documents\AI-PROGRAMMING> 
```

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:-

```
58 # generate a code for a login system for password handling
59 def login_system(stored_username, stored_password):
60     username = input("Enter username: ")
61     password = input("Enter password: ")
62     if username == stored_username and password == stored_password:
63         print("Login successful!")
64     else:
65         print("Login failed. Incorrect username or password.")
66
67 login_system("admin", "password123")
68
69 # # Generate a Python script that logs user activity (username, IP address, timestamp), identify privacy risks in the logging, and
70 # import logging
71 # from datetime import datetime
72 # # Basic logging of user activity (privacy risks: storing IP address and exact timestamp)
73 # def log_user_activity(username, ip_address):
74
75 S C:\Users\SAIAKSHITH\OneDrive\Documents\AI-PROGRAMMING> & C:/Python314/python.exe "c:/Users/SAIAKSHITH/OneDrive/Documents/AI-PROGRAMMING/exp 5.py"
Enter username: admin
Enter password: pass1234
Login failed. Incorrect username or password.
S C:\Users\SAIAKSHITH\OneDrive\Documents\AI-PROGRAMMING> |
```

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('.')[0:2]) + '.xxx.xxx' # Masking last two octets

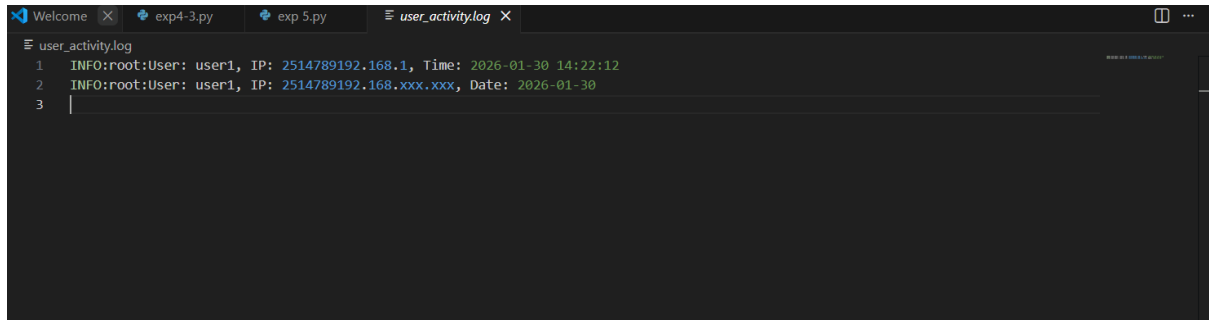
logging.info(f"User: {username}, IP: {anonymized_ip}, Date: {date}")

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:-



```
user_activity.log
1 INFO:root:User: user1, IP: 2514789192.168.1, Time: 2026-01-30 14:22:12
2 INFO:root:User: user1, IP: 2514789192.168.xxx.xxx, Date: 2026-01-30
3 |
```