SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment # 5

Name of Student : GINNE SAI TEJA
Enrollment No. : 2303A51526
Batch No. 22

Task Description #1 (Transparency in Algorithm Optimization)
Task: Use AI to generate two solutions for checking prime numbers:-

- **Naive approach(basic),**
- **Optimized approach**

Prompt:-
write a python program of a prime number checking in two methods navie and optimised approach. and add when in input letters it gives invalid input

Code:-

ASSIGNMENT-5py.py

AIAC-LAB ASSIGNMENT-1 > ASSIGNMENT-5py.py > ...

```
#task-1
#write a python program of a prime number checking in two methods navie and optimised approach. and add when in input letters
def is_prime_naive(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
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
def check_prime():
    user_input = input("Enter a number to check if it's prime: ")
    try:
        n = int(user_input)
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
        return
    if is_prime_naive(n):
        print(f"{n} is a prime number (Naive Method).")
    else:
        print(f"{n} is not a prime number (Naive Method).")
    if is_prime_optimized(n):
        print(f"{n} is a prime number (Optimized Method).")
    else:
        print(f"{n} is not a prime number (Optimized Method).")
check_prime()
```

SR UNIVERSITY

Output:-

```
Enter a number to check if it's prime: 5
5 is a prime number (Naive Method).
5 is a prime number (Optimized Method).
PS C:\AIAC-LAB ASSIGNMENT-1> ^C
PS C:\AIAC-LAB ASSIGNMENT-1>
PS C:\AIAC-LAB ASSIGNMENT-1>  c:; cd 'c:\AIAC-LAB
ns\ms-python.debugpy-2025.19.2PS C:\AIAC-LAB ASSIG
Users\SaiTeja\.vscode\extensions\ms-python.debugpy
Enter a number to check if it's prime: -9
-9 is not a prime number (Naive Method).
```

Justification:-

The program implements two methods to check whether a number is prime. The naïve approach determines primality by checking divisibility from 2 to $n - 1$. The optimized approach improves efficiency by first eliminating even numbers and then checking divisibility using the $6k \pm 1$ rule, which significantly reduces the number of required iterations. Additionally, the program handles invalid input by catching ValueError exceptions when the user enters non-integer values. This enhances the robustness of the program and ensures a user-friendly experience.

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:-

Write a Python program to calculate and print the Fibonacci series using a recursive function.

Code:-

```
#task-2
#Write a Python program to calculate and print the Fibonacci series using a recursive function.
def fibonacci(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]
    else:
        fib_series = fibonacci(n - 1)
        fib_series.append(fib_series[-1] + fib_series[-2])
        return fib_series
num_terms = int(input("Enter the number of terms in the Fibonacci series: "))
fib_series = fibonacci(num_terms)
```

Output:-

```
Enter the number of terms in the Fibonacci series: 5
Fibonacci series: [0, 1, 1, 2, 3]
PS C:\AIAC-LAB ASSIGNMENT-1>
```

Justification:-

Recursion is a technique where a function calls itself to solve a problem. In this program, the fibonacci function uses recursive calls with smaller values of n until it reaches the base cases ($n = 0$ or $n = 1$), which stop the recursion. This approach breaks the problem into smaller parts and computes the Fibonacci series step by step.

Task Description #3 (Transparency in Error Handling)

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

Prompt:-

write a python program to read file and processes data using error handling and clear explanations for each exception.

Code:-

```python
#Task-3
# write a python program to read file and  processes data using error handling and clear e
try:
    file_name = input("Enter the file name to read: ")
    with open(file_name, 'r') as file:
        data = file.read()
        print("File content:")
        print(data)
except FileNotFoundError:
    print("Error: The file was not found. Please check the file name and try again.")
except IOError:
    print("Error: An I/O error occurred while trying to read the file.")
except Exception as e:
    print(f"An unexpected error occurred: {e}")

```

Output:-

```
Enter the file name to read: AIAC_Lab Assignment_1.docx
Error: The file was not found. Please check the file name and try again.
```

Justification:-

The code handles various exceptions that might occur during file reading:
FileNotFoundError: Raised when the specified file does not exist.
IOError: Raised when an I/O error occurs while trying to read the file.
Exception: Catches any other unexpected errors.

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:-

write a python program to build a login sytem and secure password handling practices.

Code:-

```python
#task-4
#write a python program to build a login sytem and secure password handling practices.
import hashlib
def hash_password(password):
    # Hash the password using SHA-256
    return hashlib.sha256(password.encode()).hexdigest()
def verify_password(stored_password_hash, provided_password):
    # Verify the provided password against the stored hash
    return stored_password_hash == hash_password(provided_password)
# Simulated user database
user_db = {
    "sai teja": hash_password("sai@456"),
    "bunny": hash_password("SwordArtOnline"),
}
username = input("Enter your username: ")
password = input("Enter your password: ")
if username in user_db:
    if verify_password(user_db[username], password):
        print("Login successful!")
    else:
        print("Invalid password.")
else:
    print("Username not found.")

```

Output:-

```
Enter your username: sai teja
Enter your password: sai@456
Login successful!
PS C:\AIAC-LAB ASSIGNMENT-1>
```

Justification:-
Secure password handling is crucial to protect user data. By hashing passwords, we ensure that even if the database is compromised, the actual passwords remain secure. Using a strong hashing algorithm like SHA-256 adds an additional layer of security. This approach prevents storing plain-text passwords, reducing the risk of unauthorized access.

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:-
write a python program to script the logs user activities Example: username, time stamp, IP address etc Examine sensitive data is logged unnecessarily or insecurely and Improved version with minimal, anonymized, or masked logging system.

Code:-

```python
# Task-5
#write a python program to script the logs user activities Example: username,
import logging
from datetime import datetime
import socket
# Configure logging
logging.basicConfig(
    filename='user_activities.log',
    level=logging.INFO,
    format='%(asctime)s - %(message)s',
)
def log_user_activity(username):
    # Get the current timestamp
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    # Get the user's IP address
    ip_address = socket.gethostbyname(socket.gethostname())
    # Log the activity with anonymized data
    logging.info(f'User: {username}, IP: {ip_address}, Time: {timestamp}')
# Example usage
username = input("Enter your username: ")
log_user_activity(username)
print("User activity logged successfully.")
```

Output:-

```
Enter your username: sai teja
User activity logged successfully.
PS C:\AIAC-LAB ASSIGNMENT-1>
```

Justification:-

The code handles various exceptions that might occur during file reading:
FileNotFoundError: Raised when the specified file does not exist.
IOError: Raised when an I/O error occurs while trying to read the file.
Exception: Catches any other unexpected errors.