Assignment– 5.5

2303A51655

Batch:-29

Task1:TransparencyinAlgorithmOptimization

PROMPT:

GeneratePythoncodefortwoprime-checkingmethodsandexplainhowthe optimized version improves performance.give two codes one for naive approach and optimized approach

CODEandOUTPUT:

```
AI-5.5.py > ...
#Generate Python code for two prime-checking methods and explain how the optimized version improves performance.give two codes one for naive approach and optim
# Naive Approach to Check for Prime Number
def is_prime_naive(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
# Optimized Approach to Check for Prime Number
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
# Explanation of Optimization:
# The naive approach checks for factors from 2 to n-1, which results in O(n) time complexity.
# The optimized approach reduces the number of checks significantly
# 1. It first eliminates numbers less than or equal to 1, and directly returns
#    True for 2 and 3.
# 2. It eliminates even numbers and multiples of 3 right away.
# 3. It then checks for factors only up to the square root of n, since if n is divisible
#    by any number greater than its square root, it must also be divisible by a number
#    smaller than its square root.
# 4. It checks only numbers of the form 6k ± 1, as all prime numbers greater than 3
#    can be expressed in this form, further reducing the number of checks.
# This results in an average time complexity of O(√n), making it much more efficient for larger numbers.
# Example Usage
number = 29
print(f"Naive: Is {number} prime? {is_prime_naive(number)}")
print(f"Optimized: Is {number} prime? {is_prime_optimized(number)}")
```

```
56856' '--' 'D:\AI\AI-5.5.py'
Naive: Is 29 prime? True
Optimized: Is 29 prime? True
PS D:\AI>
```

Justification:

Weusedtwomethodstocheckprimenumberstounderstandalgorithmtransparencyand optimization.

- Thenaiveapproachchecksallnumbersfrom2ton-1,whichworksbutisslowfor large numbers.
- Theoptimizedapproachchecksonlyuptothesquarerootofthenumber,which reduces unnecessary calculations.

This shows how AI-generated code should be understood and improved, not blindly used. Optimizedalgorithmssavetimeandcomputingresources,whichisanethicalresponsibilityof **developers.**

Task2:TransparencyinRecursiveAlgorithms

PROMPT:

Generatearecursivefunctiontocalculatefibonaccinumbersandaddcommentsexplaining recursion and base case and recursive case.

CODEandOUTPUT:

```
#Generate a recursive function to calculate fibonacci numbers and add comments explaining recursion and base case and recursive case.
def fibonacci(n):
    # Base Case: If n is 0 or 1, return n (the first two Fibonacci numbers)
    if n <= 1:
        return n
    # Recursive Case: Return the sum of the two preceding Fibonacci numbers
    return fibonacci(n - 1) + fibonacci(n - 2)

# Example Usage:
n = int(input("Enter a positive integer to find its Fibonacci number: "))
fib_number = fibonacci(n)
print(f"The {n}th Fibonacci number is: {fib_number}")


#Explain the above code:
# The function 'fibonacci' calculates the nth Fibonacci number using recursion.
# It has a base case that handles the simplest inputs (0 and 1) directly.
# For all other values of n, it calls itself twice to compute the two preceding Fibonacci numbers,
# summing them to get the current Fibonacci number. This continues until the base case is reached.
```

```
PS C:\Users\hp\OneDrive\Desktop\AI_Assited_Coding> & 'c:\Users\hp\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\hp
ndled\libs\debugpy\launcher' '49689' '--' 'c:\Users\hp\OneDrive\Desktop\AI_Assited_Coding\Assignment-5.5.py'
Enter a positive integer to find its Fibonacci number: 20
The 20th Fibonacci number is: 6765
```

Justification:

Thistaskhelpsusunderstand **howrecursionworksclearlyandtransparently**.

- TheAI-generatedcodeincludes**basecases**,whichpreventinfiniterecursion.
- Comments explain when the function stopsandwhen it calls itself again.

Byaddingclearexplanations,thecodebecomeseasiertounderstandanddebug. Thisimproves**trust,learning,andcorrectness**,whichareimportantinethicalAIcoding. Task 3:

Transparency in Error Handling

PROMPT:

GenerateaPythonprogramthatreadsafileandprocessesdata.Generatecodewithproper error handling and clear explanations for each exception.Code with meaningful exception handling.Clear comments explaining each error scenario.Validation that explanations align with runtime behavior.

CODEandOUTPUT:

```python
#Generate a Python program that reads a file and processes data.Generate code with proper error handling and clear explanations for each exception.Code with me
# Python Program to Read a File and Process Data with Error Handling
def read_and_process_file(file_path):
    try:
        # Attempt to open the file
        with open(file_path, 'r') as file:
            data = file.readlines()
            # Process the data (for example, print each line)
            for line in data:
                print(line.strip())
    except FileNotFoundError:
        # This exception is raised when the specified file does not exist
        print(f"Error: The file '{file_path}' was not found. Please check the file path and try again.")
    except PermissionError:
        # This exception is raised when there are insufficient permissions to read the file
        print(f"Error: You do not have permission to read the file '{file_path}'. Please check your permissions.")
    except IsADirectoryError:
        # This exception is raised when a directory is provided instead of a file
        print(f"Error: The path '{file_path}' is a directory, not a file. Please provide a valid file path.")
    except Exception as e:
        # Catch-all for any other exceptions that may occur
        print(f"An unexpected error occurred: {e}")
# Example Usage
file_path = 'example.txt'  # Replace with your file path
read_and_process_file(file_path)
```

```
60320' '--' 'D:\AI\AI-5.5.py'
Error: The file 'example.txt' was not found. Please check the file path and try again.
PS D:\AI>
```

Justification:

Propererrorhandlingisimportantfor**safeandreliablesoftware**.

- Theprogramhandlescommonerrorslikemissingfilesandpermissionissues.
- Eachexceptionisexplainedsousersknowwhatwentwrong.

ThistaskshowsthatAI-generatedcodemustbe**checkedandimproved**toavoidcrashesand confusion.
Clearerrormessagesincreasetransparencyanduserconfidence. Task 4:

Security in User Authentication

PROMPT:

GenerateaPython-basedloginsystem.CheckwhethertheAIusessecurepasswordhandling practices.Identification of security flaws (plain-text passwords, weak validation).Revised versionusingpasswordhashingandinputvalidation.Shortnoteonbestpracticesforsecure authentication.

CODEandOUTPUT:

```python
#Generate a Python-based login system.Check whether the AI uses secure password handling practices.Identification of security flaws (plain-text passwords, weak
import hashlib
import re
# Simple Login System with Secure Password Handling
class LoginSystem:
    def __init__(self):
        # In a real application, user data would be stored in a database
        self.users = {}

    def hash_password(self, password):
        # Hash the password using SHA-256 for secure storage
        return hashlib.sha256(password.encode()).hexdigest()

    def validate_password(self, password):
        # Validate password strength: at least 8 characters, including letters and numbers
        if len(password) < 8:
            return False
        if not re.search(r"[a-zA-Z]", password):
            return False
        if not re.search(r"[0-9]", password):
            return False
        return True

    def register(self, username, password):
        if username in self.users:
            print("Error: Username already exists.")
            return
        if not self.validate_password(password):
            print("Error: Password must be at least 8 characters long and include both letters and numbers.")
            return
        hashed_password = self.hash_password(password)
        self.users[username] = hashed_password
        print("User registered successfully.")

    def login(self, username, password):
        if username not in self.users:
            print("Error: Username does not exist.")
            return
        hashed_password = self.hash_password(password)
        if self.users[username] == hashed_password:
            print("Login successful.")
        else:
            print("Error: Incorrect password.")
# Example Usage
login_system = LoginSystem()
login_system.register("user1", "Password123")
login_system.login("user1", "Password123")
```

```python
            print("Error: Username already exists.")
            return
        if not self.validate_password(password):
            print("Error: Password must be at least 8 characters long and include both letters and numbers.")
            return
        hashed_password = self.hash_password(password)
        self.users[username] = hashed_password
        print("User registered successfully.")

    def login(self, username, password):
        if username not in self.users:
            print("Error: Username does not exist.")
            return
        hashed_password = self.hash_password(password)
        if self.users[username] == hashed_password:
            print("Login successful.")
        else:
            print("Error: Incorrect password.")
# Example Usage
login_system = LoginSystem()
login_system.register("user1", "Password123")
login_system.login("user1", "Password123")
# Best Practices for Secure Authentication:
# 1. Always hash passwords before storing them to prevent plain-text password storage.
# 2. Use strong password policies to ensure users create secure passwords.
# 3. Implement account lockout mechanisms to prevent brute-force attacks.
# 4. Use secure protocols (like HTTPS) to protect data in transit.
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\AI> & 'c:\Users\satiq\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\satiq\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-
51163' '--' 'D:\AI\AI-5.5.py'
User registered successfully.
Login successful.
PS D:\AI>
```

Justification:

Thistaskhighlights**securityrisksinAI-generatedcode**.

- Theinsecureversionstorespasswordsinplaintext,whichisunsafe.

- Theimprovedversionuses**passwordhashing**,makingitdifficultforattackerstosteal passwords.

DevelopersmustreviewAIcodecarefullytoensureitfollows**securecodingpractices**. Security is a key ethical responsibility when handling user data.

Task5:PrivacyinDataLogging

PROMPT:

GenerateaPythonscriptthatlogsuseractivity(username,IPaddress,timestamp).Examine whether sensitive data is logged unnecessarily or insecurely.Identified privacy risks in logging.Improvedversionwithminimal,anonymized,ormaskedlogging.Explanationof privacy-aware logging principles.

CODEandOUTPUT:

```
#Generate a Python script that logs user Activity (username, IP address, timestamp).Examine whether sensitive data is logged unnecessarily or insecurely.Identi
import logging

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(message)s'
)

def log_user_activity(username, ip_address):
    anonymized_ip = '.'.join(ip_address.split('.')[:-1] + ['xxx'])
    logging.info(f'User: {username}, IP: {anonymized_ip}')

log_user_activity('user1', '192.168.1.100')
# Explanation of Privacy-Aware Logging Principles:
# 1. Minimize Data Collection: Only log information that is necessary for the intended purpose
#    (e.g., username and anonymized IP address) to reduce privacy risks.
# 2. Anonymization: Mask or anonymize sensitive data (like IP addresses) to protect user privacy.
# 3. Secure Storage: Ensure that log files are stored securely with appropriate access controls
#    to prevent unauthorized access.
# 4. Compliance: Adhere to data protection regulations and best practices regarding user data logging.
```

```
2026-01-30 10:24:05,795 - User: user1, IP: 192.168.1.xxx
PS D:\AI>
```

Justification:

Thistaskfocuseson**userprivacyanddataprotection**.

- LoggingfullusernamesandIPaddressescanexposesensitiveinformation.
- The improved version masks user data and logs only what is necessary.

Thisfollowstheprincipleof**dataminimization**,whichhelpsprotectuserprivacy. EthicalAIcodingrequiresrespectingpersonaldataandavoidingunnecessarydatacollection.