Assignment-6.5

NAME:A.JEEVAN SAI

HALL NO:2303A51420

BACTH NO:21

Task Description #1

(AI-Based Code Completion for Conditional Eligibility Check)

Task: Use an AI tool to generate eligibility logic.

Prompt: "Generate Python code to check voting eligibility based on age and citizenship." Code:

```python
'''# Lab 06: Eligibility Logic Generator
# Generate Python code to check voting eligibility based on age and citizenship.
# Expected Output:
• AI-generated conditional logic.
• Correct eligibility decisions.
• Explanation of conditions.'''

def generate_eligibility_code():
    code = '''
def check_voting_eligibility(age, is_citizen):
    if age >= 18 and is_citizen:
        return "Eligible to vote"
    else:
        return "Not eligible to vote"
'''
    return code
def explain_conditions():
    explanation = '''
The function `check_voting_eligibility` checks two conditions:
1. Age: The individual must be 18 years or older.
2. Citizenship: The individual must be a citizen.
Both conditions must be true for the individual to be eligible to vote.
'''
    return explanation
```

```python
25    # Generate the eligibility code
26    eligibility_code = generate_eligibility_code()
27    print("Generated Eligibility Code:")
28    print(eligibility_code)
29
30    # Explain the conditions
31    conditions_explanation = explain_conditions()
32    print("Explanation of Conditions:")
33    print(conditions_explanation)
34
35    # Define the generated code as a function
36    exec(eligibility_code)
37
38    # Test the generated function
39    test_cases = [
40        (20, True),
41        (17, True),
42        (19, False),
43        (16, False)
44    ]
45    for age, is_citizen in test_cases:
46        result = check_voting_eligibility(age, is_citizen)
47        print(f"Age: {age}, Citizen: {is_citizen} => {result}")
48
```

Output:

```
PS C:\Users\akash\AppData\Local\Programs\Microsoft VS Code> & C:\Users\akash\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/akash/OneDrive/Desktop/betech_3_2/AI_Assisted_Coding/
Generated Eligibility Code:

def check_voting_eligibility(age, is_citizen):
    if age >= 18 and is_citizen:
        return "Eligible to vote"
    else:
        return "Not eligible to vote"

Explanation of Conditions:

The function `check_voting_eligibility` checks two conditions:
1. Age: The individual must be 18 years or older.
2. Citizenship: The individual must be a citizen.
Both conditions must be true for the individual to be eligible to vote.

Age: 20, Citizen: True => Eligible to vote
Age: 17, Citizen: True => Not eligible to vote
Age: 19, Citizen: False => Not eligible to vote
Age: 16, Citizen: False => Not eligible to vote
PS C:\Users\akash\AppData\Local\Programs\Microsoft VS Code> 
```

Task Description #2

(AI-Based Code Completion for Loop-Based String Processing)
Task: Use an AI tool to process strings using loops.

Prompt:

"Generate Python code to count vowels and consonants in a string using a loop." **Code:**

```python
# lab 06.py > ...
1   '''
2   Task:   process strings using loops.
3   Prompt:
4   "Generate Python code to count vowels and consonants in a string
5   using a loop."
6   Expected Output:
7   • AI-generated string processing logic.
8   • Correct counts.
9   • Output verification.'''
10
11  def count_vowels_consonants(input_string):
12      vowels = "aeiouAEIOU"
13      vowel_count = 0
14      consonant_count = 0
15
16      for char in input_string:
17          if char.isalpha():  # Check if the character is a letter
18              if char in vowels:
19                  vowel_count += 1
20              else:
21                  consonant_count += 1
22
23      return vowel_count, consonant_count
24
25  # Example usage
26  input_string = "Hello, World!"
27  vowels, consonants = count_vowels_consonants(input_string)
28  print(f"Input String: {input_string}")
29  print(f"Vowels: {vowels}, Consonants: {consonants}")
30
31  # Output verification
32  assert vowels == 3, "Vowel count is incorrect"
33  assert consonants == 7, "Consonant count is incorrect"
34
35  # Output verification
36  print("Output verification passed.")
37
38
39
```

Output:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   SQL HISTORY   TASK MONITOR

PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding> & C:\Users\akash\AppData\Local\Programs\Python\Python313\python.exe
Input String: Hello, World!
Vowels: 3, Consonants: 7
Output verification passed.
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding>


Output verification passed.
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding>

Output verification passed.
Output verification passed.
Output verification passed.
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding> []
```

Task Description #3

(AI-Assisted Code Completion Reflection Task)

Task: Use an AI tool to generate a complete program using classes, loops, and conditionals.

Prompt:

"Generate a Python program for a library management system using classes, loops, and conditional statements."

Expected Output:

• Complete AI-generated program.

• Review of AI suggestions quality.

• Short reflection on AI-assisted coding experience.
Code:

```python
lab 06.py > ...
1    '''
2    Task: to generate a complete program using classes,
3    loops, and conditionals.
4    Prompt:
5    "Generate a Python program for a library management system
6    using classes, loops, and conditional statements."
7    Expected Output:
8    • Complete AI-generated program.
9    • Review of AI suggestions quality.
10   • Short reflection on AI-assisted coding experience.'''
11
12   class Book:
13       def __init__(self, title, author):
14           self.title = title
15           self.author = author
16           self.is_available = True
17
18       def __str__(self):
19           status = "Available" if self.is_available else "Checked Out"
20           return f"{self.title} by {self.author} - {status}"
21   class Library:
22       def __init__(self):
23           self.books = []
24
25       def add_book(self, book):
26           self.books.append(book)
27           print(f'Book "{book.title}" added to the library.')
28
29       def display_books(self):
30           if not self.books:
31               print("No books in the library.")
32               return
33           for idx, book in enumerate(self.books, start=1):
34               print(f"{idx}. {book}")
35
```

```python
    def check_out_book(self, book_index):
        if 0 <= book_index < len(self.books):
            book = self.books[book_index]
            if book.is_available:
                book.is_available = False
                print(f'You have checked out "{book.title}".')
            else:
                print(f'Sorry, "{book.title}" is already checked out.')
        else:
            print("Invalid book index.")

    def return_book(self, book_index):
        if 0 <= book_index < len(self.books):
            book = self.books[book_index]
            if not book.is_available:
                book.is_available = True
                print(f'You have returned "{book.title}".')
            else:
                print(f'"{book.title}" was not checked out.')
        else:
            print("Invalid book index.")
def main():
    library = Library()
    while True:
        print("\nLibrary Management System")
        print("1. Add Book")
        print("2. Display Books")
        print("3. Check Out Book")
        print("4. Return Book")
        print("5. Exit")
        choice = input("Enter your choice: ")
```

```python
        if choice == '1':
            title = input("Enter book title: ")
            author = input("Enter book author: ")
            book = Book(title, author)
            library.add_book(book)
        elif choice == '2':
            library.display_books()
        elif choice == '3':
            library.display_books()
            index = int(input("Enter the book index to check out: ")) - 1
            library.check_out_book(index)
        elif choice == '4':
            library.display_books()
            index = int(input("Enter the book index to return: ")) - 1
            library.return_book(index)
        elif choice == '5':
            print("Exiting the system. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")
if __name__ == "__main__":
    main()
# Review of AI suggestions quality:
```

Output:

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   SQL HISTORY   TASK MONITOR

PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding> & C:\Users\akash\AppData\Local\Programs\Python\Python313
g/lab 06.py"

Library Management System
1. Add Book
2. Display Books
3. Check Out Book
4. Return Book
5. Exit
Enter your choice: 1
Enter book title: llb
Enter book author: prabhas

Task Description #4

(AI-Assisted Code Completion for Class-Based Attendance System)

Task: Use an AI tool to generate an attendance management class.

Prompt: "Generate a Python class to mark and display student attendance using loops."

Expected Output:

• AI-generated attendance logic.

• Correct display of attendance.

• Test cases.

Code:

```python
'''
Task:To generate an attendance management class.
Prompt: "Generate a Python class to mark and display student
attendance using loops."
Expected Output:
• AI-generated attendance logic.
• Correct display of attendance.
• Test cases'''

class AttendanceManager:
    def __init__(self):
        self.attendance = {}

    def mark_attendance(self, student_name, present=True):
        """Marks attendance for a student."""
        self.attendance[student_name] = 'Present' if present else 'Absent'

    def display_attendance(self):
        """Displays the attendance of all students."""
        print("Attendance Record:")
        for student, status in self.attendance.items():
            print(f"{student}: {status}")
# Test cases
if __name__ == "__main__":
    manager = AttendanceManager()
    # Mark attendance for students
    manager.mark_attendance("Alice", True)
    manager.mark_attendance("Bob", False)
    manager.mark_attendance("Charlie", True)
    # Display attendance
    manager.display_attendance()
```

Output:

```
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\Devpos and fullstack> & C:\Users\akash\AppData\Local\Programs\Python\Python313\python.exe "
tech_3_2/Devpos and fullStack/temp.py"
Attendance Record:
Alice: Present
Bob: Absent
Charlie: Present
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\Devpos and fullStack>
```

Task Description #5

(AI-Based Code Completion for Conditional Menu Navigation)

Task: Use an AI tool to complete a navigation menu.

Prompt: "Generate a Python program using loops and conditionals to simulate an ATM menu."

Expected Output:

• AI-generated menu logic.

• Correct option handling.
• Output verification

Code:

```python
'''
Task:To complete a navigation menu.
Prompt: "Generate a Python program using loops and conditionals
to simulate an ATM menu."
Expected Output:
• AI-generated menu logic.
• Correct option handling.
• Output verification'''

def atm_menu():
    balance = 1000  # Initial balance
    while True:
        print("\nATM Menu:")
        print("1. Check Balance")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Exit")

        choice = input("Please select an option (1-4): ")

        if choice == '1':
            print(f"Your current balance is: ${balance}")

        elif choice == '2':
            amount = float(input("Enter amount to deposit: $"))
            if amount > 0:
                balance += amount
                print(f"${amount} deposited successfully.")
            else:
                print("Invalid amount. Please enter a positive number.")

        elif choice == '3':
            amount = float(input("Enter amount to withdraw: $"))
            if 0 < amount <= balance:
                balance -= amount
                print(f"${amount} withdrawn successfully.")
            else:
                print("Invalid amount. Please check your balance and try again.")

        elif choice == '4':
            print("Thank you for using the ATM. Goodbye!")
            break

        else:
            print("Invalid option. Please select a valid option (1-4).")
if __name__ == "__main__":
    atm_menu()
```

Output:


```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    SQL HISTORY    TASK MONITOR

PS C:\Users\akash\OneDrive\Desktop\betech_3_2\Devpos and fullStack> & C:\Users\akash\AppData\Local\Programs\Python\Python313\python.exe
tech_3_2/Devpos and fullStack/temp.py"

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Please select an option (1-4): 2
Enter amount to deposit: $100000
$100000.0 deposited successfully.
```