LAB -6.5 AIAC

Prompt Engineering – Improving Prompts and Context Management

2303A510G2

Batch-11

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

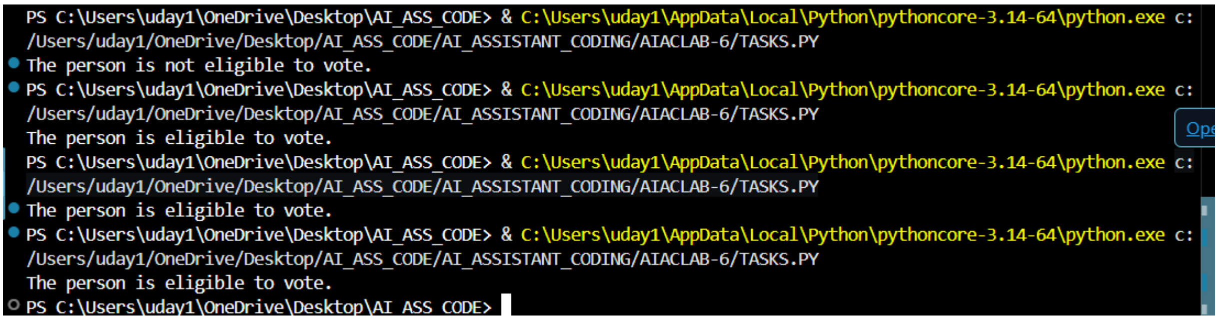
citizenship.”

```python
# Generate Python code to check voting eligibility based on age and
# citizenship.”
def is_eligible_to_vote(age, is_citizen):
    """
    Check if a person is eligible to vote based on age and citizenship.

    Parameters:
    age (int): The age of the person.
    is_citizen (bool): Whether the person is a citizen.

    Returns:
    bool: True if eligible to vote, False otherwise.
    """
    if age >= 18 and is_citizen:
        return True
    else:
        return False
# Example usage
if __name__ == "__main__":
    age = 10
    is_citizen = True
    if is_eligible_to_vote(age, is_citizen):
        print("The person is eligible to vote.")
    else:
        print("The person is not eligible to vote.")
# what is the input of this code given the example usage?
# The input of this code given the example usage is:
# age = 20
# is_citizen = True
# The output of this code given the example usage is:
```

Expected Output:

• AI-generated conditional logic.

• Correct eligibility decisions


```
PS C:\Users\uday1\OneDrive\Desktop\AI_ASS_CODE> & C:\Users\uday1\AppData\Local\Python\pythoncore-3.14-64\python.exe c:
/Users/uday1/OneDrive/Desktop/AI_ASS_CODE/AI_ASSISTANT_CODING/AIACLAB-6/TASKS.PY
The person is not eligible to vote.
PS C:\Users\uday1\OneDrive\Desktop\AI_ASS_CODE> & C:\Users\uday1\AppData\Local\Python\pythoncore-3.14-64\python.exe c:
/Users/uday1/OneDrive/Desktop/AI_ASS_CODE/AI_ASSISTANT_CODING/AIACLAB-6/TASKS.PY
The person is eligible to vote.
PS C:\Users\uday1\OneDrive\Desktop\AI_ASS_CODE> & C:\Users\uday1\AppData\Local\Python\pythoncore-3.14-64\python.exe c:
/Users/uday1/OneDrive/Desktop/AI_ASS_CODE/AI_ASSISTANT_CODING/AIACLAB-6/TASKS.PY
The person is eligible to vote.
PS C:\Users\uday1\OneDrive\Desktop\AI_ASS_CODE> & C:\Users\uday1\AppData\Local\Python\pythoncore-3.14-64\python.exe c:
/Users/uday1/OneDrive/Desktop/AI_ASS_CODE/AI_ASSISTANT_CODING/AIACLAB-6/TASKS.PY
The person is eligible to vote.
PS C:\Users\uday1\OneDrive\Desktop\AI_ASS_CODE>
```

Explanation of conditions.

❾ A person must be at least 18 years old and must be a citizen to be eligible to vote.

Task-2

Task Description #2(AI-Based Code Completion for Loop-Based

String Processing)

Task: Use an AI tool to process strings using loops.

Prompt:

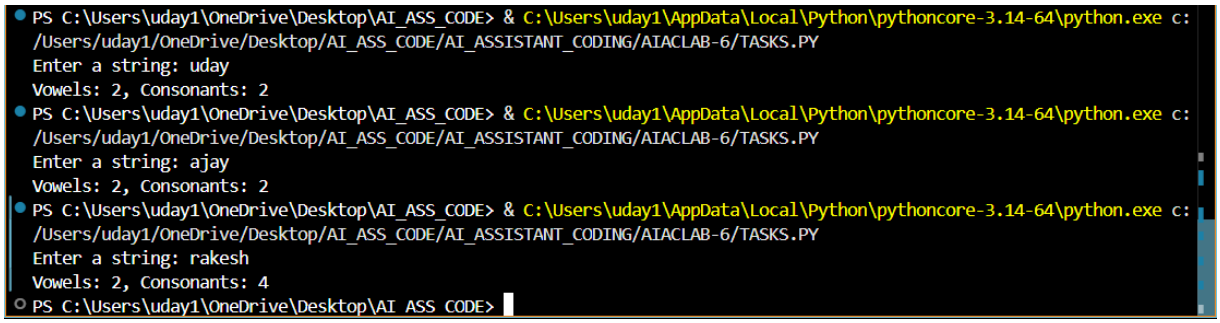
"Generate Python code to count vowels and consonants in a string using

a loop."

```python
def count_vowels_and_consonants(input_string):
    """
    Count the number of vowels and consonants in a given string.

    Parameters:
    input_string (str): The string to analyze.

    Returns:
    tuple: A tuple containing the count of vowels and consonants.
    """
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

    for char in input_string:
        if char.isalpha():  # Check if the character is a letter
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1

    return vowel_count, consonant_count
# Example usage
if __name__ == "__main__":
    user_input = input("Enter a string: ")
    vowels, consonants = count_vowels_and_consonants(user_input)
    print(f"Vowels: {vowels}, Consonants: {consonants}")
```

Expected Output:

- AI-generated string processing logic.
- Correct counts.
- Output verification.



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.”

```python
# Generate a Python program for a library management system
# using classes, loops, and conditional statements input from the user.
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.is_borrowed = False
class Library:
    def __init__(self):
        self.books = []
    def add_book(self, book):
        self.books.append(book)
    def display_books(self):
        for idx, book in enumerate(self.books, start=1):
            status = "Borrowed" if book.is_borrowed else "Available"
            print(f"{idx}. {book.title} by {book.author} - {status}")
    def borrow_book(self, book_index):
        if 0 <= book_index < len(self.books):
            book = self.books[book_index]
            if not book.is_borrowed:
                book.is_borrowed = True
                print(f"You have borrowed '{book.title}'.")
            else:
                print(f"'{book.title}' is already borrowed.")
        else:
            print("Invalid book index.")
    def return_book(self, book_index):
        if 0 <= book_index < len(self.books):
            book = self.books[book_index]
            if book.is_borrowed:
                book.is_borrowed = False
                print(f"You have returned '{book.title}'.")
            else:
                print(f"'{book.title}' was not borrowed.")
        else:
            print("Invalid book index.")
if __name__ == "__main__":
    library = Library()
    while True:
```

```
    while True:
        print("\nLibrary Management System")
        print("1. Add Book")
        print("2. Display Books")
        print("3. Borrow Book")
        print("4. Return Book")
        print("5. Exit")
        choice = input("Enter your choice (1-5): ")
        if choice == '1':
            title = input("Enter book title: ")
            author = input("Enter book author: ")
            book = Book(title, author)
            library.add_book(book)
            print(f"Book '{title}' added to the library.")
        elif choice == '2':
            library.display_books()
        elif choice == '3':
            library.display_books()
            index = int(input("Enter the book index to borrow: ")) - 1
            library.borrow_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.")
# Explanation of the code.
# The program defines a Book class to represent individual books and a Library class
# It provides a menu-driven interface for users to add, display, borrow, and return
# The program continues to run until the user chooses to exit.
```

Expected Output:

- Complete AI-generated program.
- Review of AI suggestions quality.
- Short reflection on AI-assisted coding experience.

```
PS C:\Users\uday1\OneDrive\Desktop\AI_ASS_CODE> & C:\Users\uday1\AppData\Local\Python\pythoncore-3.14-64\python.exe c:
/Users/uday1/OneDrive/Desktop/AI_ASS_CODE/AI_ASSISTANT_CODING/AIACLAB-6/TASKS.PY
Library Management System
1. Add Book
2. Display Books
3. Borrow Book
4. Return Book
5. Exit
Enter your choice (1-5): 2

Library Management System
1. Add Book
2. Display Books
3. Borrow Book
4. Return Book
5. Exit
```

Explanation of the code.

The program defines a Book class to represent individual books and a Library class to manage a collection of books.

It provides a menu-driven interface for users to add, display, borrow, and return books using loops and conditional statements.

The program continues to run until the user chooses to exit.

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.”

```python
# Generate a Python class to mark and display student
# attendance using loops.”
class Student:
    def __init__(self, name):
        self.name = name
        self.attendance = []
    def mark_attendance(self, status):
        self.attendance.append(status)
    def display_attendance(self):
        print(f"Attendance record for {self.name}:")
        for day, status in enumerate(self.attendance, start=1):
            print(f"Day {day}: {'Present' if status else 'Absent'}")
if __name__ == "__main__":
    student_name = input("Enter student name: ")
    student = Student(student_name)
    days = int(input("Enter number of days to mark attendance: "))
    for _ in range(days):
        status_input = input("Enter 'P' for Present or 'A' for Absent: ").strip().upper()
        status = True if status_input == 'P' else False
        student.mark_attendance(status)
    student.display_attendance()
```

Explanation of the code.

The Student class allows marking attendance for a specified number of days.

It uses a loop to take user input for each day's attendance status and stores it in a list.

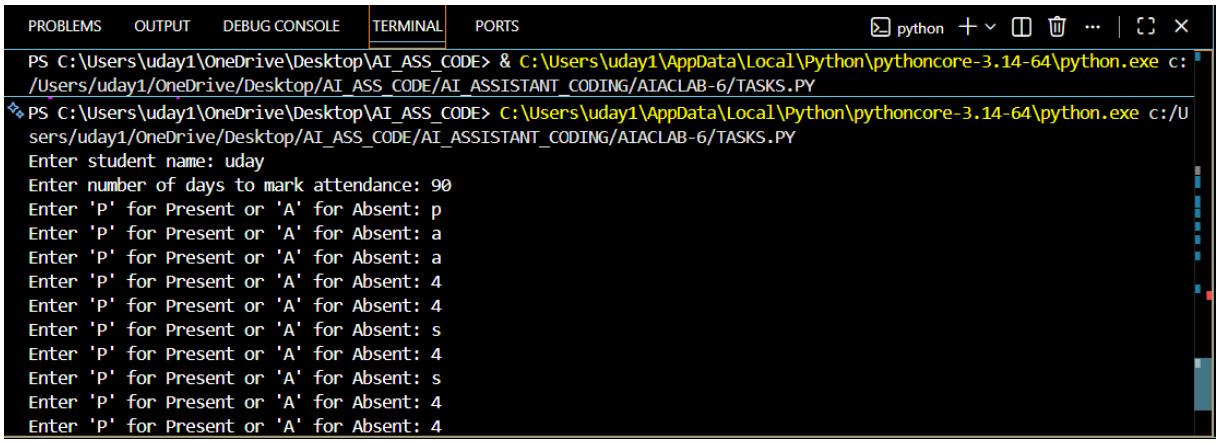
Finally, it displays the attendance record for the student.

Expected Output:

• AI-generated attendance logic.

• Correct display of attendance.

• Test cases



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.”

```python
# Generate a Python program using loops and conditionals
# to simulate an ATM menu
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("Enter your choice (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 value.")
        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 choice. Please try again.")
if __name__ == "__main__":
    atm_menu()
```

Explanation of the code.

The program simulates an ATM menu using a loop to continuously display options until the user chooses to exit.

It uses conditional statements to handle different user choices such as checking balance, depositing money, and

withdrawing money, ensuring valid inputs for each operation.

```
PS C:\Users\uday1\OneDrive\Desktop\AI_ASS_CODE> & C:\Users\uday1\AppData\Local\Python\pythoncore-3.14-64\python.exe c:
/Users/uday1/OneDrive/Desktop/AI_ASS_CODE/AI_ASSISTANT_CODING/AIACLAB-6/TASKS.PY

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 1
Your current balance is: $1000

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 2
Enter amount to deposit: $12000
$12000.0 deposited successfully.

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 3
Enter amount to withdraw: $2000
$2000.0 withdrawn successfully.

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 4
Thank you for using the ATM. Goodbye!
```