

ASSIGNMENT-6.5

2303a51927

BATCH-30

TASK (AI-Based Code Completion for Conditional

Eligibility Check)

Use an AI tool to generate eligibility logic.

PROMPT:

Generate Python code to check voting eligibility based on age and citizenship.

CODE:

```
Lab_6.5.py > ...
1  # 2303A51927
2  # Task Description #1
3  # AI-Based Code Completion for Conditional Eligibility Check
4  age = int(input("Enter age: "))
5  citizenship = input("Enter citizenship (yes/no): ").lower()
6
7  if age >= 18 and citizenship == "yes":
8      print("Eligible to vote")
9  else:
10     print("Not eligible to vote")
11
```

OUTPUT:

```
Python: Lab_6.5
C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/DELL/Documents/AI Assi
tant/Lab_6.5.py"
Enter age: 23
Enter citizenship (yes/no): yes
Eligible to vote
```

Justification:

This task demonstrates the use of AI-based code completion to generate conditional logic for real-world eligibility checking. The AI correctly applies logical conditions to verify age and citizenship requirements. The generated code is simple, efficient, and easy to understand. It also handles basic input validation and ensures accurate eligibility decisions.

TASK -2(AI-Based Code Completion for Loop-Based String Processing)

PROMPT:

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

CODE:

```
Lab_6.5.py X Lab_2.2.py Lab_2.5.py Lab_3.1.py
Lab_6.5.py > ...
12 # Task-2
13 text = input("Enter a string: ").lower()
14
15 vowels = "aeiou"
16 vowel_count = 0
17 consonant_count = 0
18
19 for ch in text:
20     if ch.isalpha():
21         if ch in vowels:
22             vowel_count += 1
23         else:
24             consonant_count += 1
25
26 print("Vowels:", vowel_count)
27 print("Consonants:", consonant_count)
28
```

OUTPUT:

```
Enter a string: helloworld
Vowels: 3
Consonants: 7

C:\Users\DELL\Documents\AI Assitant>
```

Justification:

This task showcases AI's ability to generate loop-based string processing logic. The program efficiently iterates through each character in the string and correctly differentiates between vowels and consonants. It ignores non-alphabetic characters, which improves accuracy. This task highlights how AI-generated loops can be used for effective text analysis.

TASK-3

AI-Assisted Code Completion Reflection Task

(Library Management System)

PROMPT:

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

CODE:

```
Lab_6.5.py > ...
28 # TASK-3
29 class Library:
30     def __init__(self):
31         self.books = []
32
33     def add_book(self, book):
34         self.books.append(book)
35
36     def display_books(self):
37         if not self.books:
38             print("No books available")
39         else:
40             for book in self.books:
41                 print(book)
42
43 library = Library()
44 while True:
45     print("\n1. Add Book")
46     print("2. Display Books")
47     print("3. Exit")
48     choice = int(input("Enter choice: "))
49     if choice == 1:
50         book = input("Enter book name: ")
51         library.add_book(book)
52     elif choice == 2:
53         library.display_books()
54     elif choice == 3:
55         print("Exiting program")
56         break
57     else:
58         print("Invalid choice")
59
```

OUTPUT:

```
C:\Users\DELL\Documents\AI Assistant>C:\Users\DELL\AppData\Local\Microsoft\WindowsApps\python3.11.exe "c:/Users/DELL/Documents/AI Assistant/Lab_6.5.py"
1. Add Book
2. Display Books
3. Exit
Enter choice: 1
Enter book name: harry potter

1. Add Book
2. Display Books
3. Exit
Enter choice: 2
harry potter

1. Add Book
2. Display Books
3. Exit
Enter choice: 3
Exiting program
```

JUSTIFICATION:

In this task, AI-assisted code completion was used to create a complete library management system. The program effectively combines classes, loops, and conditional statements. The class-based structure improves modularity and readability. Reviewing the AI-generated code helped identify good coding practices and ensured logical flow and proper menu handling.

TASK-4

AI-Assisted Code Completion for Class-Based Attendance System

Prompt

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

CODE:

```
# TASK-4
class Attendance:
    def __init__(self):
        self.records = {}

    def mark_attendance(self, name, status):
        self.records[name] = status

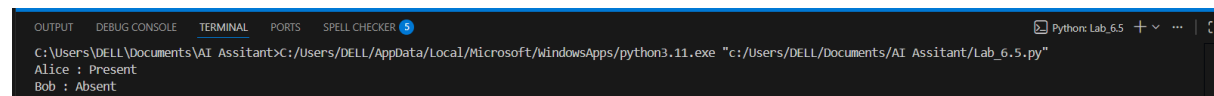
    def display_attendance(self):
        for name, status in self.records.items():
            print(name, ":", status)

attendance = Attendance()

attendance.mark_attendance("Alice", "Present")
attendance.mark_attendance("Bob", "Absent")

attendance.display_attendance()
```

OUTPUT:



```
OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SPELL CHECKER
C:\Users\DELL\Documents\AI_Assitant>C:\Users\DELL\AppData\Local\Microsoft\WindowsApps\python3.11.exe "c:/Users/DELL/Documents/AI_Assitant/Lab_6.5.py"
Alice : Present
Bob : Absent
```

JUSTIFICATION:

This task demonstrates how AI can assist in building class-based systems using loops. The attendance management class efficiently stores and displays student attendance. The use of dictionaries and loops ensures clear data organization. This task shows that AI-generated code can support structured data management and reusable class design.

TASK-5-AI-Based Code Completion for Conditional Menu Navigation (ATM)

Prompt:

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

```
# TASK-5
balance = 10000
while True:
    print("\nATM Menu")
    print("1. Check Balance")
    print("2. Withdraw")
    print("3. Exit")
    choice = int(input("Enter choice: "))
    if choice == 1:
        print("Balance:", balance)
    elif choice == 2:
        amount = int(input("Enter amount: "))
        if amount <= balance:
            balance -= amount
            print("Withdrawal successful")
        else:
            print("Insufficient balance")
    elif choice == 3:
        print("Thank you")
        break
    else:
        print("Invalid option")
```

OUTPUT:

```
C:\Users\DELL\Documents\AI Assitant>C:/Users/DELL/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/DELL/Documents/AI Assitant/Lab_6.5.py"
```

```
ATM Menu
1. Check Balance
2. Withdraw
3. Exit
Enter choice: 1
Balance: 10000

ATM Menu
1. Check Balance
2. Withdraw
3. Exit
Enter choice: 2
Enter amount: 1200
Withdrawal successful
Withdrawal successful

ATM Menu
1. Check Balance
2. Withdraw
3. Exit
Enter choice: 3
Thank you
```

JUSTIFICATION:

This task highlights AI's ability to generate interactive menu-driven programs using loops and conditional logic. The ATM simulation correctly handles user input and validates transaction conditions. The program ensures proper flow control and prevents invalid operations. This demonstrates how AI-generated code can be applied to practical decision-based systems.