

AI ASSISTED CODING

ASSIGNMENT-6.5

NAME: S.Ravikiran

H.T.NO:2303A52261

BATCH:36

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.”

Expected Output:

- AI-generated conditional logic.
- Correct eligibility decisions.
- Explanation of conditions.

Code:

```
▶ age = int(input("Enter your age: "))
  citizenship = input("Are you a citizen? (yes/no): ")

  if age >= 18 and citizenship.lower() == "yes":
      print("You are eligible to vote.")
  else:
      print("You are not eligible to vote.")
```

OutPut :

```
Enter your age: 20
Are you a citizen? (yes/no): yes
You are eligible to vote.
```

Explanation :

- `age = int(input(...))`: Takes the user's age as input.
- `citizenship = input(...)`: Takes citizenship status.
- `if age >= 18 and citizenship.lower() == "yes":`
Checks both conditions:
 - Age must be 18 or above
 - Citizenship must be “yes”
- Prints eligibility result based on conditions.

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.”

Expected Output:

- AI-generated string processing logic.

- Correct counts.
- Output verification.

Code :

```
text = input("Enter a string: ")
vowels = "aeiouAEIOU"
vowel_count = 0
consonant_count = 0

for char in text:
    if char.isalpha():
        if char in vowels:
            vowel_count += 1
        else:
            consonant_count += 1

print("Vowels:", vowel_count)
print("Consonants:", consonant_count)
```

Output :

```
--> Enter a string: hello world
Vowels: 3
Consonants: 7
```

Explanation :

- Loops through each character in the string.
- `isalpha()` ensures only letters are counted.
- Vowels are checked using a predefined string.
- Counts vowels and consonants separately.

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 :

```
▶ class Library:  
    def __init__(self):  
        self.books = []  
  
    def add_book(self, book):  
        self.books.append(book)  
        print(book, "added to library.")  
  
    def display_books(self):  
        if not self.books:  
            print("No books available.")  
        else:  
            for book in self.books:  
                print(book)  
  
library = Library()  
  
while True:  
    print("\n1. Add Book\n2. Display Books\n3. Exit")  
    try:  
        choice = int(input("Enter choice: "))
```

```
while True:
    print("\n1. Add Book\n2. Display Books\n3. Exit")
    try:
        choice = int(input("Enter choice: "))
    except ValueError:
        print("Invalid input. Please enter a number (1, 2, or 3).")
        continue

    if choice == 1:
        book_name = input("Enter book name: ")
        library.add_book(book_name)
    elif choice == 2:
        library.display_books()
    elif choice == 3:
        print("Exiting...")
        break
    else:
        print("Invalid choice. Please enter 1, 2, or 3.")
```

OutPut :

```
...
1. Add Book
2. Display Books
3. Exit
Enter choice: 1
Enter book name: Rich dad and Poor Dad
Rich dad and Poor Dad added to library.

1. Add Book
2. Display Books
3. Exit
Enter choice: 2
Rich dad and Poor Dad

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

Explanation :

This program uses **classes, loops, and conditional statements** to manage a small library.

- A **class Library** is created to store and manage books.
- The `_init_()` method initializes an empty list to store book names.
- The `add_book()` method adds a new book to the library.
- The `display_books()` method shows all available books using a loop.
- A **while loop** displays a menu repeatedly until the user chooses to exit.
- **Conditional statements (if-elif-else)** handle user choices like adding or displaying books.

Task Description #4

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

Code :

```
class Attendance:  
    def __init__(self):  
        self.students = {}  
  
    def mark_attendance(self, name, status):  
        self.students[name] = status  
  
    def display_attendance(self):  
        for name, status in self.students.items():  
            print(name, ":", status)  
  
attendance = Attendance()  
  
attendance.mark_attendance("Alice", "Present")  
attendance.mark_attendance("Bob", "Absent")  
  
attendance.display_attendance()
```

OutPut :

```
Alice : Present  
Bob : Absent
```

Explanation :

- Uses a dictionary to store student names and attendance.
- Loop displays attendance for all students.

Task Description #5 (AI-Based Code Completion for Conditional Menu Navigation)

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

Code :

```
| balance = 1000  
  
| while True:  
|     print("\nATM Menu")  
|     print("1. Check Balance")  
|     print("2. Deposit")  
|     print("3. Withdraw")  
|     print("4. Exit")  
  
|     choice = int(input("Enter choice: "))  
  
|     if choice == 1:  
|         print("Balance:", balance)  
|     elif choice == 2:  
|         amount = int(input("Enter deposit amount: "))  
|         balance += amount  
|         print("Amount deposited.")  
|     elif choice == 3:  
|         amount = int(input("Enter withdrawal amount: "))  
|         if amount <= balance:  
|             balance -= amount  
|             print("Please collect cash.")  
|         else:
```

```
        print("Please collect cash.")
    else:
        print("Insufficient balance.")
elif choice == 4:
    print("Thank you!")
    break
else:
    print("Invalid choice")
```

OutPut :

```
ATM Menu
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 1
Balance: 1000
```

```
ATM Menu
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 2
Enter deposit amount: 1000
Amount deposited.
```

```
ATM Menu
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 3
Enter withdrawal amo
Please collect cash.
```

```
ATM Menu
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter choice: 4
Thank you!
```

Explanation :

This program simulates an **ATM system** using **loops and conditionals**.

- A variable `balance` stores the current account balance.
- A **while loop** keeps the ATM menu running until the user exits.
- The menu offers four options:
 1. Check Balance
 2. Deposit Money
 3. Withdraw Money
 4. Exit
- **Conditional statements** process the selected option.
- Withdrawal checks if sufficient balance is available before allowing the transaction.