

## AI Assisted coding

### LAB Assignment 6.5

J.Sujith  
2303A51327  
Batch 20

#### Task-1: Voting Eligibility

##### Prompt

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

```
[3] ✓ 13s #task 1
def check_voting_eligibility(age, is_citizen):
    """
    Check if a person is eligible to vote.

    Args:
        age (int): The person's age
        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
    return False

# Example usage
if __name__ == "__main__":
    age = int(input("Enter your age: "))
    is_citizen = input("Are you Indian citizen? (yes/no): ").lower() == "yes"

    if check_voting_eligibility(age, is_citizen):
        print("You are eligible to vote!")
    else:
        print("You are not eligible to vote.")

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

##### Observation

The program successfully checks whether a person is eligible to vote by verifying two conditions: age and citizenship. If the age is 18 or above and the user is a citizen, the program displays that the person is eligible to vote. Otherwise, it correctly reports that the person is not eligible. This task demonstrates the use of conditional statements and user input handling.

## Task-2: Count Vowels and Consonants

### Prompt

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

```
[5]
✓ 26s #task 2
def count_vowels_and_consonants(text):
    """
    Count vowels and consonants in a string.

    Args:
        text (str): The input string

    Returns:
        tuple: (vowel_count, consonant_count)
    """
    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

    return vowel_count, consonant_count

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

... Enter a string: sar
Vowels: 1
Consonants: 2
```

## **Observation**

The program iterates through each character in the given string and checks whether it is a vowel or a consonant. It maintains separate counters for vowels and consonants and displays the final count. This task demonstrates the use of loops, string traversal, and basic conditional logic.

## **Task–3: Library Management System**

### **Prompt**

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

```
[6]
✓ Os
▶ #TASK-3
class Book:
    def __init__(self, book_id, title, author, available=True):
        self.book_id = book_id
        self.title = title
        self.author = author
        self.available = available

class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book):
        self.books.append(book)
        print(f"Book '{book.title}' added successfully.")

    def search_book(self, title):
        for book in self.books:
            if book.title.lower() == title.lower():
                return book
        return None

    def borrow_book(self, title):
        book = self.search_book(title)
        if book and book.available:
            book.available = False
            print(f"You borrowed '{book.title}'.")
        elif book:
            print(f"'{book.title}' is not available.")
        else:
            print("Book not found.")

    def return_book(self, title):
        book = self.search_book(title)
        if book and not book.available:
            book.available = True
            print(f"You returned '{book.title}'.")
```

```
        print(f"You returned '{book.title}'.")
    else:
        print("Invalid return.")

    def display_books(self):
        if not self.books:
            print("No books in library.")
            return
        for book in self.books:
            status = "Available" if book.available else "Borrowed"
            print(f"ID: {book.book_id}, Title: {book.title}, Author: {book.author}, Status: {status}")

# Example usage
if __name__ == "__main__":
    library = Library()
    library.add_book(Book(1, "Python Basics", "John Doe"))
    library.add_book(Book(2, "Data Science", "Jane Smith"))
    library.display_books()
    library.borrow_book("Python Basics")
    library.display_books()
    library.return_book("Python Basics")
```

```

... Book 'Python Basics' added successfully.
Book 'Data Science' added successfully.
ID: 1, Title: Python Basics, Author: John Doe, Status: Available
ID: 2, Title: Data Science, Author: Jane Smith, Status: Available
You borrowed 'Python Basics'.
ID: 1, Title: Python Basics, Author: John Doe, Status: Borrowed
ID: 2, Title: Data Science, Author: Jane Smith, Status: Available
You returned 'Python Basics'.
```

## Observation

The program uses classes to represent books and the library system. It allows users to add books, issue books, return books, and display available books. The system uses loops for continuous interaction and conditional statements to validate operations. This task demonstrates object-oriented programming concepts and real-world problem modeling.

## Task-4: Student Attendance System

### Prompt

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

```
[12] # Task-4: Student Attendance System
✓ 0s

class Student:
    def __init__(self, student_id, name):
        self.student_id = student_id
        self.name = name
        self.attendance = []

    def mark_attendance(self, present):
        """Mark attendance for a day (True for present, False for absent)"""
        self.attendance.append(present)

    def get_attendance_percentage(self):
        """Calculate attendance percentage"""
        if not self.attendance:
            return 0
        present_days = sum(1 for day in self.attendance if day)
        return (present_days / len(self.attendance)) * 100

class AttendanceSystem:
    def __init__(self):
        self.students = []

    def add_student(self, student):
        self.students.append(student)

    def display_attendance(self):
        if not self.students:
            print("No students in the system.")
            return

        for student in self.students:
            attendance_str = "".join(["P" if day else "A" for day in student.attendance])
            percentage = student.get_attendance_percentage()
            print(f"ID: {student.student_id}, Name: {student.name}, "
                  f"Attendance: {attendance_str}, Percentage: {percentage:.2f}%")
```

```
# Main program
if __name__ == "__main__":
    system = AttendanceSystem()
    system.add_student(Student(1, "Alice"))
    system.add_student(Student(2, "Bob"))

    # Mark attendance for 5 days
    for student in system.students:
        for day in range(5):
            student.mark_attendance(day % 2 == 0)

    system.display_attendance()
```

✓ ID: 1, Name: Alice, Attendance: PAPAP, Percentage: 60.00%  
ID: 2, Name: Bob, Attendance: PAPAP, Percentage: 60.00%

## Observation

The attendance system allows adding students and marking their attendance as present or absent. It calculates the attendance percentage for each student and displays the results in a structured format. This task demonstrates the use of classes, lists, loops, and percentage calculations effectively.

## Task–5: ATM Simulation

### Prompt

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

```
[14]
✓ 9s # Task-5: ATM System

class ATM:

    def __init__(self, balance=1000):
        self.balance = balance

    def check_balance(self):
        print(f"Your current balance: ${self.balance}")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient funds.")
        elif amount <= 0:
            print("Invalid amount.")
        else:
            self.balance -= amount
            print(f"Withdrawal successful. New balance: ${self.balance}")

    def deposit(self, amount):
        if amount <= 0:
            print("Invalid amount.")
        else:
            self.balance += amount
            print(f"Deposit successful. New balance: ${self.balance}")

    def run(self):
        while True:
            print("\n--- ATM Menu ---")
            print("1. Check Balance")
            print("2. Withdraw")
            print("3. Deposit")
            print("4. Exit")

            choice = input("Select an option (1-4): ")
```

```
[14]
✓ 9s

        if choice == "1":
            self.check_balance()
        elif choice == "2":
            amount = float(input("Enter withdrawal amount: "))
            self.withdraw(amount)
        elif choice == "3":
            amount = float(input("Enter deposit amount: "))
            self.deposit(amount)
        elif choice == "4":
            print("Thank you for using ATM. Goodbye!")
            break
        else:
            print("Invalid option. Please try again.")

if __name__ == "__main__":
    atm = ATM()
    atm.run()
```

```
...
--- ATM Menu ---
1. Check Balance
2. Withdraw
3. Deposit
4. Exit
Select an option (1-4): 2
Enter withdrawal amount: 200
Withdrawal successful. New balance: $800.0

--- ATM Menu ---
1. Check Balance
2. Withdraw
3. Deposit
4. Exit
Select an option (1-4): 4
Thank you for using ATM. Goodbye!
```

## **Observation**

The ATM simulation provides a simple command-line interface for users to interact with an ATM system. It includes functionalities to check balance, withdraw money, deposit money, and exit the system. The program handles invalid inputs gracefully and updates the balance correctly. This task demonstrates basic programming concepts such as classes, methods, loops, and user input handling.