Lab Assignent-6.5

Batch-35

HT NO-2303A52462

Task 1: Prompt:

“Generate Python code to check voting eligibility based on age and

citizenship.”

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

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

Output:

```
Enter your age: 21
Are you a citizen? (yes/no): yes
You are eligible to vote!
```

Explanation:

- **Age Condition**
  - age >= 18
  - Ensures the person meets the minimum voting age requirement.
- **Citizenship Condition**
  - citizenship.lower() == "yes"

- Converts input to lowercase so inputs like Yes, YES, or yes are all accepted.

 Logical AND (and)

- Both conditions **must be true** for the person to be eligible.
- If either age is below 18 **or** citizenship is not "yes", voting is not allowed.

Task 2: Generate Python code to count vowels and consonants in a string using loops

Code:

```
text = input("Enter a string: ").lower()
vowels = "aeiou"
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(f"Number of vowels: {vowel_count}")
print(f"Number of consonants: {consonant_count}")
```

Output:

```
Enter a string: Alphabeats
Number of vowels: 4
Number of consonants: 6
```

Task 3: Generate a Python program for a library management system

using classes, loops, and conditional statements.

Code:

```python
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 remove_book(self, book_id):
        for book in self.books:
            if book.book_id == book_id:
                self.books.remove(book)
                print(f"Book removed.")
                return
        print("Book not found.")

    def borrow_book(self, book_id):
        for book in self.books:
            if book.book_id == book_id:
                if book.available:
                    book.available = False
                    print(f"You borrowed '{book.title}'.")
                    return
                else:
                    print("Book is not available.")
                    return
        print("Book not found.")

    def return_book(self, book_id):
        for book in self.books:
            if book.book_id == book_id and not book.available:
                book.available = True
                print(f"You returned '{book.title}'.")
                return
        print("Book not found or not borrowed.")

    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}")

# Test
library = Library()
library.add_book(Book(1, "Python Basics", "John Doe"))
library.add_book(Book(2, "Web Dev", "Jane Smith"))
library.display_books()
library.borrow_book(1)
library.display_books()
library.return_book(1)
library.display_books()
```

Output:

```
ID: 2, Title: Web Dev, Author: Jane Smith, Status: Available
PS C:\python..P> & C:/Users/ARKAN/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/python..P/#task 1.py"
Book 'Python Basics' added successfully.
Book 'Web Dev' added successfully.
ID: 1, Title: Python Basics, Author: John Doe, Status: Available
ID: 2, Title: Web Dev, Author: Jane Smith, Status: Available
You borrowed 'Python Basics'.
ID: 1, Title: Python Basics, Author: John Doe, Status: Borrowed
ID: 2, Title: Web Dev, Author: Jane Smith, Status: Available
You returned 'Python Basics'.
ID: 1, Title: Python Basics, Author: John Doe, Status: Available
ID: 2, Title: Web Dev, Author: Jane Smith, Status: Available
PS C:\python..P>
```

Task 4: Use an AI tool to generate an attendance management class.

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

Code:

```python
class Attendance:
    def __init__(self):
        self.records = {}

    def add_student(self, student_id, name):
        if student_id not in self.records:
            self.records[student_id] = {"name": name, "attendance": []}
            print(f"Student '{name}' added.")

    def mark_attendance(self, student_id, present):
        if student_id in self.records:
            self.records[student_id]["attendance"].append(present)
            status = "present" if present else "absent"
            print(f"{self.records[student_id]['name']} marked {status}.")
        else:
            print("Student not found.")

    def display_attendance(self):
        if not self.records:
            print("No students in records.")
            return
        for student_id, data in self.records.items():
            attendance_list = ["P" if p else "A" for p in data["attendance"]]
            present_count = sum(data["attendance"])
            total = len(data["attendance"])
            percentage = (present_count / total * 100) if total > 0 else 0
            print(f"ID: {student_id}, Name: {data['name']}, Attendance: {' '.join(attendance_list)}, Present: {present_count}/{total} ({percentage:.1f}%)")

# Test
attendance = Attendance()
attendance.add_student(1, "Alice")
attendance.add_student(2, "Bob")
attendance.mark_attendance(1, True)
attendance.mark_attendance(1, True)
attendance.mark_attendance(1, False)
attendance.mark_attendance(2, True)
attendance.mark_attendance(2, False)
attendance.mark_attendance(2, False)
attendance.display_attendance()
```

Output:

```
55  # Test
56  attendance = Attendance()
57  attendance.add_student(1, "Alice")
58  attendance.add_student(2, "Bob")
59  attendance.mark_attendance(1, True)
60  attendance.mark_attendance(1, True)
61  attendance.mark_attendance(1, False)
62  attendance.mark_attendance(2, True)
63  attendance.mark_attendance(2, False)
64  attendance.mark_attendance(2, False)
65  attendance.display_attendance()
66
67
```

Task 5: Generate a Python program using loops and conditionals to simulate an ATM menu.”

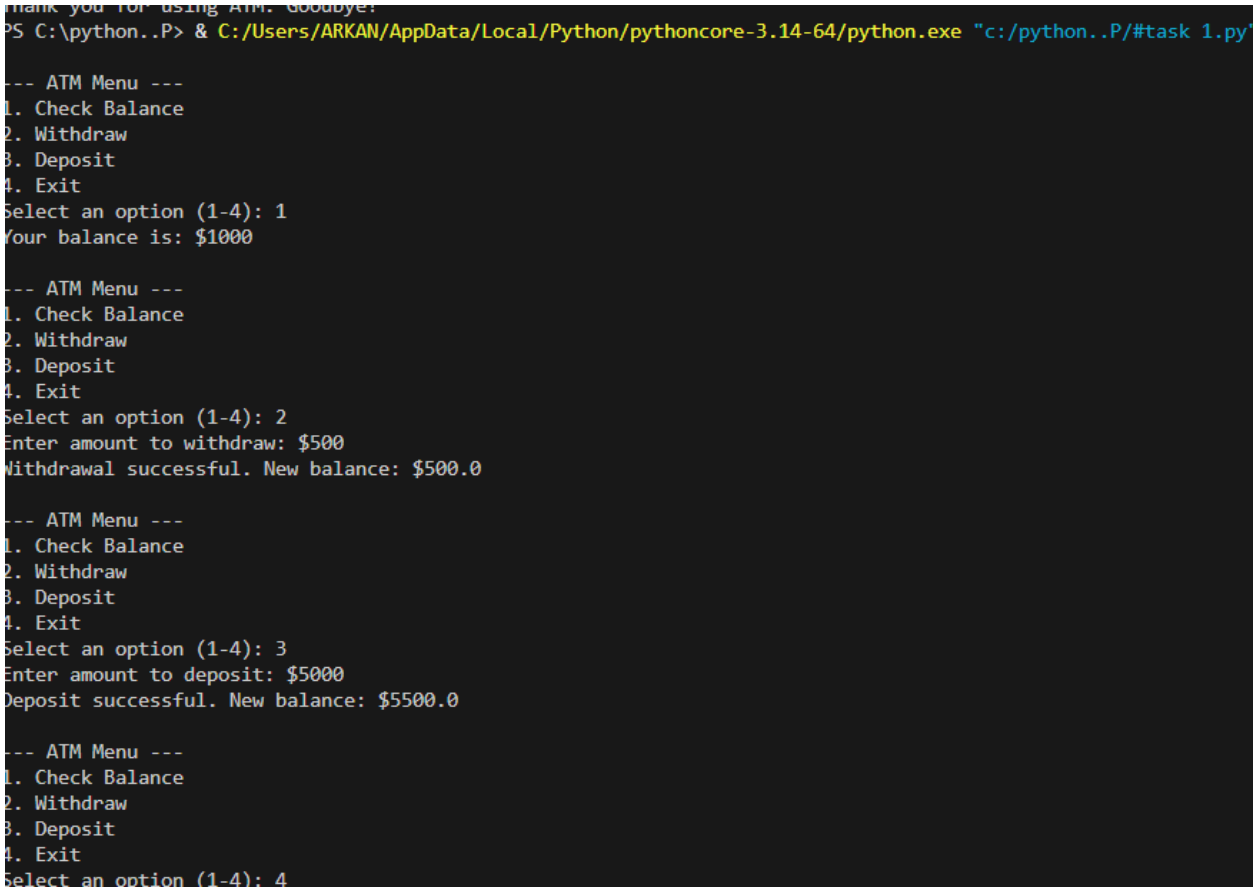
Code:

```
def atm_menu():
    balance = 1000
    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): ")

        if choice == "1":
            print(f"Your balance is: ${balance}")
        elif choice == "2":
            amount = float(input("Enter amount to withdraw: $"))
            if amount > balance:
                print("Insufficient funds!")
            elif amount <= 0:
                print("Invalid amount!")
            else:
                balance -= amount
                print(f"Withdrawal successful. New balance: ${balance}")
        elif choice == "3":
            amount = float(input("Enter amount to deposit: $"))
            if amount <= 0:
                print("Invalid amount!")
            else:
                balance += amount
                print(f"Deposit successful. New balance: ${balance}")
        elif choice == "4":
            print("Thank you for using ATM. Goodbye!")
            break
        else:
            print("Invalid option. Please try again.")

atm_menu()
```

Output:


Thank you for using ATM. Goodbye!
PS C:\python..P> & C:/Users/ARKAN/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/python..P/#task 1.py"
--- ATM Menu ---
1. Check Balance
2. Withdraw
3. Deposit
4. Exit
Select an option (1-4): 1
Your balance is: $1000
--- ATM Menu ---
1. Check Balance
2. Withdraw
3. Deposit
4. Exit
Select an option (1-4): 2
Enter amount to withdraw: $500
Withdrawal successful. New balance: $500.0
--- ATM Menu ---
1. Check Balance
2. Withdraw
3. Deposit
4. Exit
Select an option (1-4): 3
Enter amount to deposit: $5000
Deposit successful. New balance: $5500.0
--- ATM Menu ---
1. Check Balance
2. Withdraw
3. Deposit
4. Exit
Select an option (1-4): 4