SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment 6.5

Course Title	: AI Assisted Coding
Course Code	: 23CS002PC304
Semester	: III
Academic Session	: 2025-2026
Name of Student	: k.Deepak
Enrollment No.	: 2303A51T02
Batch No.	: 52

Task 1: Conditional Eligibility Check (Voting) Prompt:
"Generate Python code to check voting eligibility based on age and citizenship." Code:

```
#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:

```
s\91939\Desktop\3-2\AI Assisted coding\Assignment6.5.py'
Enter your age: 20
Are you a citizen? (yes/no): yes
You are eligible to vote.
```

Explanation :
This code checks if a person is eligible to vote based on their age and citizenship status.

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:

```python
#Generate Python code to count vowels and co
def count_vowels_consonants(input_string):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

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

    print("Vowels:", vowel_count)
    print("Consonants:", consonant_count)
# Example usage
sample_string = "Hello World"
count_vowels_consonants(sample_string)
```

Output:

```
x64\bundled\libs\debu
Vowels: 3
Consonants: 7
PS C:\Users\91939\Des
```

Explanation:
The function count_vowels_consonants takes a string and counts the number of vowels and consonants.
It iterates through each character in the string, checks if it's an alphabetic character, and then determines whether it's a vowel or consonant.
Finally, it prints the counts of vowels and consonants

Task 3: AI-Assisted Code Completion Reflection 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.” Code:

```python
#Generate a Python program for a library management system using clas
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.is_available = True

    def __str__(self):
        status = "Available" if self.is_available else "Checked Out"
        return f'"{self.title}" by {self.author} - {status}'
class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book):
        self.books.append(book)
        print(f'Added: {book}')

    def display_books(self):
        print("Library Catalog:")
        for book in self.books:
            print(book)

    def check_out_book(self, title):
        for book in self.books:
            if book.title == title:
                if book.is_available:
                    book.is_available = False
                    print(f'You have checked out: {book}')
                    return
                else:
                    print(f'Sorry, "{title}" is currently checked out.')
                    return
        print(f'Sorry, "{title}" is not found in the library.')

    def return_book(self, title):
        for book in self.books:
            if book.title == title:
                if not book.is_available:
                    book.is_available = True
                    print(f'You have returned: {book}')
                    return
                else:
                    print(f'"{title}" was not checked out.')
                    return
        print(f'Sorry, "{title}" is not found in the library.')
# Example usage
library = Library()
book1 = Book("1984", "George Orwell")
book2 = Book("To Kill a Mockingbird", "Harper Lee")
library.add_book(book1)
library.add_book(book2)
library.display_books()
library.check_out_book("1984")
library.display_books()
library.return_book("1984")
library.display_books()
```

Output:

```
Added: "1984" by George Orwell - Available
Added: "To Kill a Mockingbird" by Harper Lee - Available
Library Catalog:
"1984" by George Orwell - Available
"To Kill a Mockingbird" by Harper Lee - Available
You have checked out: "1984" by George Orwell - Checked Out
Library Catalog:
"1984" by George Orwell - Checked Out
"To Kill a Mockingbird" by Harper Lee - Available
You have returned: "1984" by George Orwell - Available
Library Catalog:
"1984" by George Orwell - Available
"To Kill a Mockingbird" by Harper Lee - Available
```

Explanation:
This code defines a simple library management system using classes.
The Book class represents a book with attributes for title, author, and availability status.

Task 4: AI-Assisted Code Completion for Class Based Attendance System)

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
#Generate a Python class to mark and display student attendance usin
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 for {self.name}:')
        for day, status in enumerate(self.attendance, start=1):
            print(f'Day {day}: {"Present" if status else "Absent"}')
# Example usage
student = Student("Alice")
student.mark_attendance(True)
student.mark_attendance(False)
student.mark_attendance(True)
student.display_attendance()
```

Output:

```
x64\bundled\libs\debugpy\launch
Attendance for Alice:
Day 1: Present
Day 2: Absent
Day 3: Present
```

Explanation:
This code defines a Student class that allows marking and displaying attendance. The mark_attendance method records whether the student was present (True) or absent (False).
The display_attendance method prints the attendance record for the student.

Task 5: AI-Based Code Completion for Conditional Menu Navigation
Use an AI tool to complete a navigation menu.
Prompt: “Generate a Python program using loops and conditionals to simulate an ATM menu.” Code:

```
#Generate a Python program using loops and conditionals to simulate an ATM
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("Select an option (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 number.")
        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 or insufficient funds.")
```

```
            if 0 < amount <= balance:
                balance -= amount
                print(f'${amount} withdrawn successfully.')
            else:
                print("Invalid amount or insufficient funds.")
        elif choice == '4':
            print("Thank you for using the ATM. Goodbye!")
            break
        else:
            print("Invalid selection. Please choose a valid option.")
# Start the ATM menu
atm_menu()
```

Output:

```
ATM Menu:
ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Select an option (1-4): 1
Your current balance is: $1000

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Select an option (1-4): 2
Enter amount to deposit: $200
$200.0 deposited successfully.

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Select an option (1-4): 3
Enter amount to withdraw: $100
$100.0 withdrawn successfully.
```

```
ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Select an option (1-4): 1
Your current balance is: $1100.0

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

Explanation:
This code simulates an ATM menu using a while loop and conditional statements.
The user can check their balance, deposit money, withdraw money, or exit the program.
Generate a python function that removes all negative numbers from a list and show before and after output.