

ASSIGNMENT-6.5

HT.NO:2303A510I4

Batch.No:30

Task 1: Use an AI tool to generate eligibility logic.

Prompt:

Generate a Python script that validates voter eligibility based on minimum age and citizenship status.

Code:

```
assignment 6.5 > 🗂 Task1.py > ...
1  #2303A510I4
2  def check_voting_eligibility(age,is_citizen):
3      if age >= 18 and is_citizen:
4          return "You are eligible to vote."
5      else:
6          return "You are not eligible to vote."
7  age = int(input("Enter your age: "))
8  citizenship_input = input("Are you a citizen? (yes/no): ").lower()
9  is_citizen = citizenship_input == "yes"
10 result = check_voting_eligibility(age, is_citizen)
11 print(result)
```

Output:

```
gnment 6.5/Task1.py"
Enter your age: 20
Are you a citizen? (yes/no): yes
You are eligible to vote.
○ PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>
```

Observation:

The conditional logic correctly determines voting eligibility based on age and citizenship.

Task 2: Use an AI tool to process strings using loops.

Prompt:

Generate a Python program that iterates through a given string and displays the total number of vowels and consonants.

Code:

```
assignment 6.5 > Task2.py > ...
1  #2303A510I4
2  #generate a python program that iterates through a given string and displays the total number of vowels and consonants
3  def count_vowels_consonants(s):
4      vowels = "aeiouAEIOU"
5      vowel_count = 0
6      consonant_count = 0
7      for char in s:
8          if char.isalpha():
9              if char in vowels:
10                  vowel_count += 1
11              else:
12                  consonant_count += 1
13      return vowel_count, consonant_count
14
15  input_string = input("Enter a string: ")
16  vowels, consonants = count_vowels_consonants(input_string)
17  print(f"Number of vowels: {vowels}")
18  print(f"Number of consonants: {consonants}")
```

Output:

```
● PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> conda activate base
● PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python 6.5/Task2.py"
Enter a string: AI ASSISTED CODING
Number of vowels: 7
Number of consonants: 9
○ PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> Ln 18, Col 45 Sp
```

Observation:

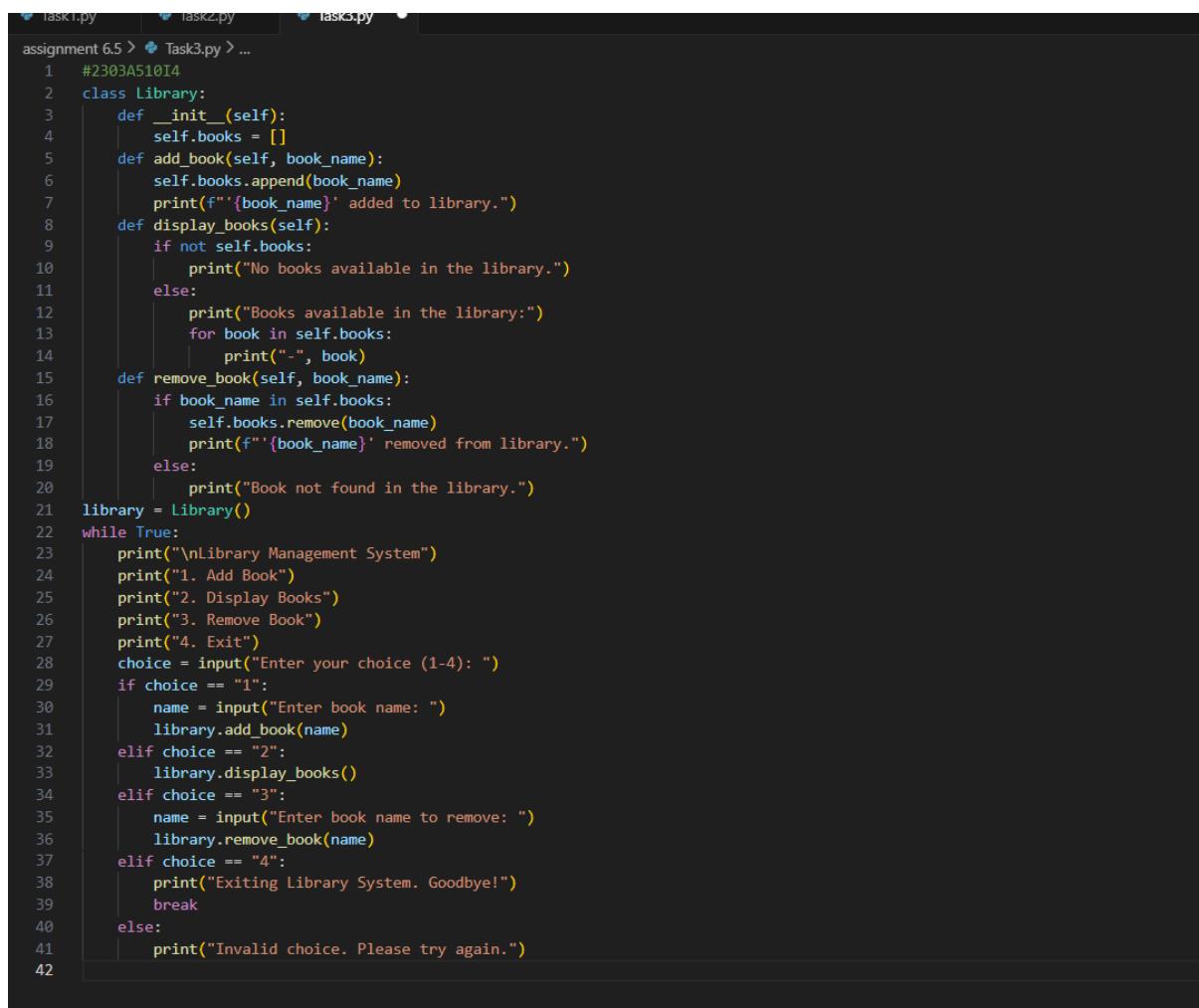
The program accurately processes the string and provides correct vowel and consonant counts.

Task 3: Use an AI tool to generate a complete program using classes, loops, and conditionals.

Prompt:

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

Code:



```
assignment 6.5 > Task3.py ...  
1 #2303A510I4  
2 class Library:  
3     def __init__(self):  
4         self.books = []  
5     def add_book(self, book_name):  
6         self.books.append(book_name)  
7         print(f'{book_name} added to library.')  
8     def display_books(self):  
9         if not self.books:  
10             print("No books available in the library.")  
11         else:  
12             print("Books available in the library:")  
13             for book in self.books:  
14                 print("-", book)  
15     def remove_book(self, book_name):  
16         if book_name in self.books:  
17             self.books.remove(book_name)  
18             print(f'{book_name} removed from library.')  
19         else:  
20             print("Book not found in the library.")  
21 library = Library()  
22 while True:  
23     print("\nLibrary Management System")  
24     print("1. Add Book")  
25     print("2. Display Books")  
26     print("3. Remove Book")  
27     print("4. Exit")  
28     choice = input("Enter your choice (1-4): ")  
29     if choice == "1":  
30         name = input("Enter book name: ")  
31         library.add_book(name)  
32     elif choice == "2":  
33         library.display_books()  
34     elif choice == "3":  
35         name = input("Enter book name to remove: ")  
36         library.remove_book(name)  
37     elif choice == "4":  
38         print("Exiting Library System. Goodbye!")  
39         break  
40     else:  
41         print("Invalid choice. Please try again.")
```

Output:

```
C:\Users\Shivani Pabba\OneDrive\Desktop\AI> cd ..\..\miniconda3\envs\basic
● PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pabba/OneDrive/Desktop/AI/assignment 6.5/Task3.py"

Library Management System
1. Add Book
2. Display Books
3. Remove Book
4. Exit
Enter your choice (1-4): 1
Enter book name: ai
'ai' added to library.

Library Management System
1. Add Book
2. Display Books
3. Remove Book
4. Exit
Enter your choice (1-4): 2
Books available in the library:
- ai

Library Management System
1. Add Book
2. Display Books
3. Remove Book
4. Exit
Enter your choice (1-4): 3
Enter book name to remove: ai
'ai' removed from library.

Library Management System
1. Add Book
2. Display Books
3. Remove Book
4. Exit
Enter your choice (1-4): 4
Exiting Library System. Goodbye!
○ PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>
```

Observation:

The program uses a class to organize library operations such as adding, displaying, and removing books. A menu-driven loop with conditional statements allows continuous interaction with the user. Overall, the code clearly demonstrates the practical use of classes, loops, and conditionals.

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

Prompt:

Generate a Python class that records and displays student attendance using loops.

Code:

```

assignment 6.5 > Task4.py > ...
1 #2303A510I4
2 #Generate a Python class that records and displays student attendance using loops
3 class Attendance:
4     def __init__(self):
5         self.attendance_record = {}
6     def mark_attendance(self, student_name):
7         self.attendance_record[student_name] = "Present"
8         print(f"{student_name} marked as Present.")
9     def display_attendance(self):
10        if not self.attendance_record:
11            print("No attendance records available.")
12        else:
13            print("Attendance Records:")
14            for student, status in self.attendance_record.items():
15                print(f"- {student}: {status}")
16 attendance = Attendance()
17 while True:
18     print("\nStudent Attendance System")
19     print("1. Mark Attendance")
20     print("2. Display Attendance")
21     print("3. Exit")
22     choice = input("Enter your choice (1-3): ")
23     if choice == "1":
24         name = input("Enter student name: ")
25         attendance.mark_attendance(name)
26     elif choice == "2":
27         attendance.display_attendance()
28     elif choice == "3":
29         print("Exiting Attendance System. Goodbye!")
30         break
31     else:
32         print("Invalid choice. Please try again.")
```

Output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pabba/Assignment 6.5/Task4.py"

Student Attendance System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice (1-3): 1
Enter student name: siri
siri marked as Present.

Student Attendance System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice (1-3): 2
Attendance Records:
- siri: Present

Student Attendance System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice (1-3): 3
Exiting Attendance System. Goodbye!
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>
```

Observation:

The program uses a class to store and manage student attendance records efficiently. A menu-driven loop with conditional statements allows marking and displaying attendance. The attendance details are displayed correctly using dictionary traversal.

Task 5: Use an AI tool to complete a navigation menu.

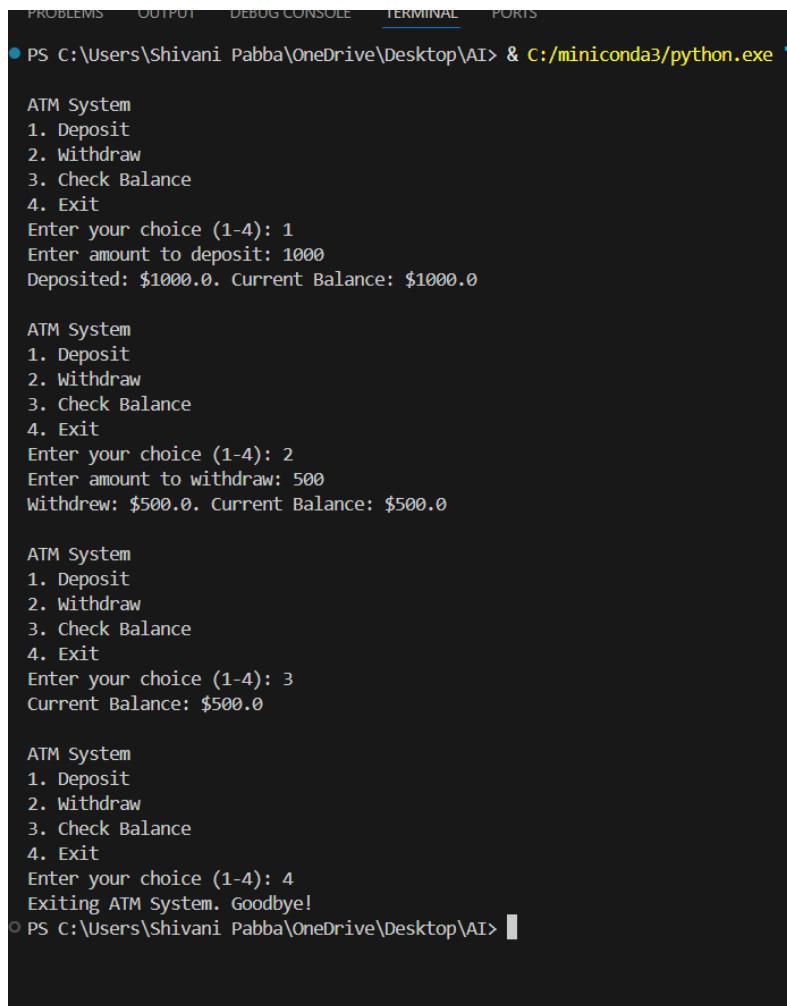
Prompt:

Develop a Python program that simulates an ATM system using loops and conditional statements.

Code:

```
task1.py task2.py task3.py task4.py task5.py
assignment 6.5 > #2303AS1014
1 #Develop a Python program that simulates an ATM system using loops and conditional statements.
2 class ATM:
3     def __init__(self):
4         self.balance = 0
5     def deposit(self, amount):
6         if amount > 0:
7             self.balance += amount
8             print(f"Deposited: ${amount}. Current Balance: ${self.balance}")
9         else:
10            print("Invalid deposit amount.")
11    def withdraw(self, amount):
12        if amount > self.balance:
13            print("Insufficient funds.")
14        elif amount <= 0:
15            print("Invalid withdrawal amount.")
16        else:
17            self.balance -= amount
18            print(f"Withdrew: ${amount}. Current Balance: ${self.balance}")
19    def check_balance(self):
20        print(f"Current Balance: ${self.balance}")
21 atm = ATM()
22 while True:
23     print("\nATM System")
24     print("1. Deposit")
25     print("2. Withdraw")
26     print("3. Check Balance")
27     print("4. Exit")
28     choice = input("Enter your choice (1-4): ")
29     if choice == "1":
30         amount = float(input("Enter amount to deposit: "))
31         atm.deposit(amount)
32     elif choice == "2":
33         amount = float(input("Enter amount to withdraw: "))
34         atm.withdraw(amount)
35     elif choice == "3":
36         atm.check_balance()
37     elif choice == "4":
38         print("Exiting ATM System. Goodbye!")
39         break
40     else:
41         print("Invalid choice. Please try again.")
```

Output:



The screenshot shows a terminal window with the following content:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "ATM System.py"
ATM System
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 1
Enter amount to deposit: 1000
Deposited: $1000.0. Current Balance: $1000.0

ATM System
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 2
Enter amount to withdraw: 500
Withdrew: $500.0. Current Balance: $500.0

ATM System
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 3
Current Balance: $500.0

ATM System
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 4
Exiting ATM System. Goodbye!
○ PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>
```

Observation:

The program uses a class to manage ATM operations such as deposit, withdrawal, and balance enquiry. A menu-driven loop with conditional statements allows continuous user interaction. Input validation is handled properly to prevent invalid transactions and insufficient balance errors.