

AI Assisted Coding

Lab Assignment 6.5

Name : A.Shiva Spandana

Hall Ticket no : 2303A51320

Batch No : 19

Task -1:

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

```
Assignment-6.5.py •
Assignment-6.5.py > check_voting_eligibility
1 # Task -1 : Voting Eligibility Checker
2
3 def check_voting_eligibility(age, is_citizen):
4     if age >= 18 and is_citizen:
5         return "✓ Eligible to vote: You meet all requirements (age 18+ and citizen)"
6     elif age < 18:
7         return "X Not eligible: You must be at least 18 years old"
8     elif not is_citizen:
9         return "X Not eligible: You must be a citizen to vote"
10    else:
11        return "X Not eligible: You do not meet the voting requirements"
12
13 # Test cases
14 print(check_voting_eligibility(25, True)) # Eligible
15 print(check_voting_eligibility(16, True)) # Too young
16 print(check_voting_eligibility(30, False)) # Not a citizen
17 print(check_voting_eligibility(17, False)) # Both conditions fail
```

OUTPUT :

```
PS C:\Users\spand\OneDrive\Desktop\ai assisted coding> & C:/Users/spand/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/spand/OneDrive/Desktop/ai assisted coding/assignment6.5.py"
✓ Eligible to vote: You meet all requirements (age 18+ and citizen)
X Not eligible: You must be at least 18 years old
X Not eligible: You must be a citizen to vote
X Not eligible: You must be at least 18 years old
PS C:\Users\spand\OneDrive\Desktop\ai assisted coding>
```

Justification:

- ✓ This program checks voting eligibility based on **age** and **citizenship** using conditional statements.
- ✓ If the person is **18 or older and a citizen**, they are eligible to vote; otherwise, the program clearly states the reason for ineligibility.
- ✓ It ensures correct decision-making with simple and readable logic.

Task 2:

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

```

18 # Task -2 : Vowel and Consonant Counter
19 def count_vowels_and_consonants(text):
20     vowels = "aeiouAEIOU"
21     vowel_count = 0
22     consonant_count = 0
23     for char in text:
24         if char.isalpha():
25             if char in vowels:
26                 vowel_count += 1
27             else:
28                 consonant_count += 1
29     return vowel_count, consonant_count
30 # Test cases
31 test_string = "Hello World"
32 vowels, consonants = count_vowels_and_consonants(test_string)
33 print(f"String: '{test_string}'")
34 print(f"Vowels: {vowels}")
35 print(f"Consonants: {consonants}")
36 test_string2 = "Python Programming"
37 vowels2, consonants2 = count_vowels_and_consonants(test_string2)
38 print(f"\nString: '{test_string2}'")
39 print(f"Vowels: {vowels2}")
40 print(f"Consonants: {consonants2}")

```

Output:

```

String: 'Hello World'
Vowels: 3
Consonants: 7

String: 'Python Programming'
Vowels: 4
Consonants: 13
PS C:\Users\spand\OneDrive\Desktop\ai assisted coding>

```

Justification:

- ✓ This program counts **vowels and consonants** in a given text by iterating through each character.
- ✓ It checks only **alphabetic characters** and classifies them as vowels or consonants using conditional logic.
- ✓ The function returns accurate counts, ignoring spaces and special characters.

Task 3:

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

```

42 # Task -3 : Simple Library Management System
43 class Book:
44     def __init__(self, book_id, title, author, available=True):
45         self.book_id = book_id
46         self.title = title
47         self.author = author
48         self.available = available
49     def __str__(self):
50         status = "Available" if self.available else "Checked Out"
51         return f"ID: {self.book_id}, Title: {self.title}, Author: {self.author}, Status: {status}"
52 class Library:
53     def __init__(self):
54         self.books = []
55     def add_book(self, book):
56         self.books.append(book)
57         print(f"✓ Book '{book.title}' added to library")
58     def checkout_book(self, book_id):
59         for book in self.books:
60             if book.book_id == book_id:
61                 if book.available:
62                     book.available = False
63                     print(f"✓ '{book.title}' checked out successfully")
64                     return
65                 else:
66                     print(f"X '{book.title}' is already checked out")
67             return
68         print(f"X Book with ID {book_id} not found")
69     def return_book(self, book_id):
70         for book in self.books:
71             if book.book_id == book_id:
72                 if not book.available:
73                     book.available = True
74                     print(f"✓ '{book.title}' returned successfully")
75                     return
76                 else:
77                     print(f"X '{book.title}' is already available")
78                     return
79         print(f"X Book with ID {book_id} not found")
80     def display_all_books(self):
81         if not self.books:
82             print("Library is empty")
83             return
84         print("\n--- Library Books ---")
85         for book in self.books:
86             print(book)
87 # Test the library system
88 library = Library()
89 library.add_book(Book(1, "Python Basics", "John Doe"))
90 library.add_book(Book(2, "Data Science", "Jane Smith"))
91 library.add_book(Book(3, "Web Development", "Mike Johnson"))
92 library.display_all_books()
93 library.checkout_book(1)
94 library.checkout_book(1)
95 library.return_book(1)
96 library.display_all_books()

```

Output :

```
eDrive/Desktop/ai assisted coding/assignment6.5.py"
✓ Book 'Python Basics' added to library
✓ Book 'Data Science' added to library
✓ Book 'Web Development' added to library

--- Library Books ---
ID: 1, Title: Python Basics, Author: John Doe, Status: Available
ID: 2, Title: Data Science, Author: Jane Smith, Status: Available
ID: 3, Title: Web Development, Author: Mike Johnson, Status: Available
✓ 'Python Basics' checked out successfully
X 'Python Basics' is already checked out
✓ 'Python Basics' returned successfully

--- Library Books ---
ID: 1, Title: Python Basics, Author: John Doe, Status: Available
ID: 2, Title: Data Science, Author: Jane Smith, Status: Available
ID: 3, Title: Web Development, Author: Mike Johnson, Status: Available
PS C:\Users\spand\OneDrive\Desktop\ai assisted coding> █
```

Justification:

- ✓ This program implements a simple Library Management System using object-oriented programming concepts.
- ✓ The Book class stores book details and availability, while the Library class manages adding, issuing, returning, and displaying books.
- ✓ It ensures proper tracking of book status with clear messages for each operation.

Task 4 :

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

Expected Output:

- AI-generated attendance logic.
- Correct display of attendance.
- Test cases

```

98 # Task -4 : Student Attendance Tracker
99 class Student:
100     def __init__(self, student_id, name):
101         self.student_id = student_id
102         self.name = name
103         self.attendance = []
104     def mark_attendance(self, date, status):
105         self.attendance.append({"date": date, "status": status})
106         print(f"✓ Attendance marked for {self.name} on {date}: {status}")
107     def get_attendance_percentage(self):
108         if not self.attendance:
109             return 0
110         present = sum(1 for record in self.attendance if record["status"].lower() == "present")
111         return (present / len(self.attendance)) * 100
112 class AttendanceTracker:
113     def __init__(self):
114         self.students = []
115     def add_student(self, student):
116         self.students.append(student)
117         print(f"✓ Student '{student.name}' added to tracker")
118     def display_attendance(self):
119         if not self.students:
120             print("No students in tracker")
121             return
122         print("\n--- Attendance Report ---")
123         for student in self.students:
124             print(f"\nStudent: {student.name} (ID: {student.student_id})")
125             for record in student.attendance:
126                 print(f"    {record['date']}: {record['status']}")
127             print(f"    Attendance: {student.get_attendance_percentage():.1f}%")
128 # Test cases
129 tracker = AttendanceTracker()
130 student1 = Student(101, "Alice")
131 student2 = Student(102, "Bob")
132 tracker.add_student(student1)
133 tracker.add_student(student2)
134 for date in ["2024-01-01", "2024-01-02", "2024-01-03"]:
135     student1.mark_attendance(date, "Present")
136     student2.mark_attendance(date, "Present")
137 student1.mark_attendance("2024-01-04", "Absent")
138 student2.mark_attendance("2024-01-04", "Present")
139 tracker.display_attendance()

```

Output :

```
Drive/Desktop/ai assisted coding/assignment0.5.py
/ Student 'Alice' added to tracker
/ Student 'Bob' added to tracker
/ Attendance marked for Alice on 2024-01-01: Present
/ Attendance marked for Bob on 2024-01-01: Present
/ Attendance marked for Alice on 2024-01-02: Present
/ Attendance marked for Bob on 2024-01-02: Present
/ Attendance marked for Alice on 2024-01-03: Present
/ Attendance marked for Bob on 2024-01-03: Present
/ Attendance marked for Alice on 2024-01-04: Absent
/ Attendance marked for Bob on 2024-01-04: Present
```

```
--- Attendance Report ---
```

```
Student: Alice (ID: 101)
```

```
2024-01-01: Present
```

```
2024-01-02: Present
```

```
2024-01-03: Present
```

```
2024-01-04: Absent
```

```
Attendance: 75.0%
```

```
Student: Bob (ID: 102)
```

```
2024-01-01: Present
```

```
2024-01-02: Present
```

```
2024-01-03: Present
```

```
2024-01-04: Present
```

```
Attendance: 100.0%
```

```
PS C:\Users\spand\OneDrive\Desktop\ai assisted coding> █
```

Justification:

- ✓ This program tracks **student attendance** using object-oriented principles.
- ✓ The Student class records daily attendance and calculates attendance percentage, while the Attendance Tracker class manages multiple students and generates reports.
- ✓ It provides a clear and structured way to monitor attendance efficiently

Task 5 :

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

```

141 # Task - 5 : ATM Simulation
142 class ATMSimulation:
143     def __init__(self, balance=1000):
144         self.balance = balance
145     def display_menu(self):
146         print("\n--- ATM Menu ---")
147         print("1. Check Balance")
148         print("2. Withdraw Money")
149         print("3. Deposit Money")
150         print("4. Exit")
151     def check_balance(self):
152         print(f"✓ Current Balance: ${self.balance:.2f}")
153     def withdraw_money(self):
154         try:
155             amount = float(input("Enter amount to withdraw: $"))
156             if amount <= 0:
157                 print("X Amount must be greater than zero")
158             elif amount > self.balance:
159                 print(f"X Insufficient funds. Available balance: ${self.balance:.2f}")
160             else:
161                 self.balance -= amount
162                 print(f"✓ Successfully withdrawn ${amount:.2f}")
163                 print(f"✓ Remaining balance: ${self.balance:.2f}")
164         except ValueError:
165             print("X Invalid input. Please enter a valid number")
166     def deposit_money(self):
167         try:
168             amount = float(input("Enter amount to deposit: $"))
169             if amount <= 0:
170                 print("X Amount must be greater than zero")
171             else:
172                 self.balance += amount
173                 print(f"✓ Successfully deposited ${amount:.2f}")
174                 print(f"✓ New balance: ${self.balance:.2f}")
175         except ValueError:
176             print("X Invalid input. Please enter a valid number")
177     def run(self):
178         print("✓ Welcome to ATM Simulation")
179         while True:
180             self.display_menu()
181             choice = input("Select an option (1-4): ")
182             if choice == "1":
183                 self.check_balance()
184             elif choice == "2":
185                 self.withdraw_money()
186             elif choice == "3":
187                 self.deposit_money()
188             elif choice == "4":
189                 print("✓ Thank you for using ATM. Goodbye!")
190                 break
191             else:
192                 print("X Invalid option. Please select 1-4")
193 # Test the ATM system
194 atm = ATMSimulation(1000)
195 atm.run()

```

Output :

```
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
Select an option (1-4): 1
✓ Current Balance: $1000.00

--- ATM Menu ---
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
Select an option (1-4): 4
✓ Thank you for using ATM. Goodbye!
PS C:\Users\spand\OneDrive\Desktop\ai assisted coding> |
```

Justification:

- ✓ This program simulates an **ATM system** that allows users to check balance, withdraw, and deposit money.
- ✓ It uses a menu-driven approach with input validation to handle invalid entries and insufficient funds.
- ✓ The system ensures secure and user-friendly banking operations through clear prompts and messages.