

AI Assisted Coding Assignment- 6.5

2303A51543 BT: 29 VELDI.HRUTHIKA 06.02.2026



Task Description #1 (AI-Based Code Completion for Conditional Eligibility Check)

Task: Use an AI tool to generate eligibility logic.

Prompt:

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

Expected Output:

- AI-generated conditional logic.
- Correct eligibility decisions.
- Explanation of conditions.

```
#2303A51543 BT-29
```

```
#Task -1
```

```
def check_voting_eligibility(age, citizenship=True):
    """
    Check if a person is eligible to vote.

    Args:
        age: The person's age (int)
        citizenship: Whether the person is a citizen (bool, default=True)

    Returns:
        bool: True if eligible to vote, False otherwise
    """
    if age >= 18 and citizenship:
        return True
    return False

# Example usage
if __name__ == "__main__":
    print(check_voting_eligibility(25)) # True
    print(check_voting_eligibility(16)) # False
    print(check_voting_eligibility(20, citizenship=False)) # False
```

```

1 def check_voting_eligibility(age, citizenship=True):
2     """
3     Check if a person is eligible to vote.
4
5     Args:
6         age: The person's age (int)
7         citizenship: Whether the person is a citizen (bool, default=True)
8
9     Returns:
10        bool: True if eligible to vote, False otherwise
11    """
12    if age >= 18 and citizenship:
13        return True
14    return False
15
16 # Example usage
17 if __name__ == "__main__":
18     # PS C:\Users\hruth\OneDrive\Desktop\A.I.AC> & C:\Users\hruth\AppData\Local\Programs\Python\Python312\python.exe c:\Users\hruth\OneDrive\Desktop\A.I.AC\Ass_6.5.py
19     True
20     False
21     False
22     PS C:\Users\hruth\OneDrive\Desktop\A.I.AC>

```

The function checks if a person is eligible to vote based on two conditions:

1. **Age:** Must be 18 or older.
2. **Citizenship:** Must be a citizen (default `True`).

It returns `True` if both conditions are met, otherwise `False`.

Example:

- `check_voting_eligibility(25)` → `True`
- `check_voting_eligibility(16)` → `False`
- `check_voting_eligibility(20, citizenship=False)` → `False`



Task Description #2(AI-Based Code Completion for Loop-Based String Processing)

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

Prompt:

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

Expected Output:

- AI-generated string processing logic.
- Correct counts.
- Output verification.

#Task -2

```
def count_vowels_consonants(input_string):
    """
```

Count the number of vowels and consonants in a given string.

Args:

input_string: The string to analyze (str)

Returns:

tuple: A tuple containing the count of vowels and consonants (vowels_count, consonants_count)

```
"""
vowels = 'aeiouAEIOU'
vowels_count = 0
consonants_count = 0

for char in input_string:
    if char.isalpha(): # Check if the character is an alphabet
        if char in vowels:
            vowels_count += 1
        else:
            consonants_count += 1

return vowels_count, consonants_count
# Example usage
if __name__ == "__main__":
    test_string = "Hello, World!"
    vowels, consonants = count_vowels_consonants(test_string)
    print(f"Vowels: {vowels}, Consonants: {consonants}") # Vowels: 3, Consonants: 7
```

```
23 | print(check_voting_eligibility(20, citizenship=False)) # False
24 |
25 | #Task -2
26 | def count_vowels_consonants(input_string):
    """
    Count the number of vowels and consonants in a given string.

    Args:
        input_string: The string to analyze (str)
```

PROBLEMS 1 OUTPUT TERMINAL PORTS DEBUG CONSOLE

```
File Edit Selection View ... Q ALIAC
EXPLORER
  ALIAC
    ASSIGNMENT-1(0.1)
      ASS-1(1543).pdf
      Ass_4.5.py
      Ass_4.5(2).py
      Ass_4.5(3).py
      Ass_4.5(4).py
      Ass_5.5.py
      Ass_6.5.py
      ASS-4.5(1543).pdf
      ASS-5.5(1543).pdf
      data.txt
      exam.py
      String_Ass1.py
      String.py
      user_activity.log
  OUTLINE
  TIMELINE

Ass_6.5.py
26 def count_vowels_consonants(input_string):
27     """
28     Count the number of vowels and consonants in a given string.
29
30     Args:
31         input_string: The string to analyze (str)
32     Returns:
33         tuple: A tuple containing the count of vowels and consonants (vowels_count, consonants_count)
34     """
35     vowels = 'aeiouAEIOU'
36     vowels_count = 0
37     consonants_count = 0
38
39     for char in input_string:
40         if char.isalpha(): # Check if the character is an alphabet
41             if char in vowels:
42                 vowels_count += 1
43             else:
44                 consonants_count += 1
45
46     return vowels_count, consonants_count
47
48 # Example usage
49 if __name__ == "__main__":
50     test_string = "Hello, World!"
51     vowels, consonants = count_vowels_consonants(test_string)
52     print(f"Vowels: {vowels}, Consonants: {consonants}") # Vowels: 3, Consonants: 7
53
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
Python 3.12.3
PS C:\Users\hruth\OneDrive\Desktop\A.I.AC> & C:/Users/hruth/AppData/Local/Programs/Python/Python312/python.exe c:/Users/hruth/OneDrive/Desktop/A.I.AC/Ass_6.5.py
True
False
False
PS C:\Users\hruth\OneDrive\Desktop\A.I.AC>
```

```
24
25 #Task -2
26 def count_vowels_consonants(input_string):
27     """
28     Count the number of vowels and consonants in a given string.
29
30     Args:
31         input_string: The string to analyze (str)
32     Returns:
33         tuple: A tuple containing the count of vowels and consonants (vowels_count, consonants_count)
34     """
35     vowels = 'aeiouAEIOU'
36     vowels_count = 0
37     consonants_count = 0
38
39     for char in input_string:
40         if char.isalpha(): # Check if the character is an alphabet
41             if char in vowels:
42                 vowels_count += 1
43             else:
44                 consonants_count += 1
45
46     return vowels_count, consonants_count
47 # Example usage
48 if __name__ == "__main__":
49     test_string = "Hello, World!"
50     vowels, consonants = count_vowels_consonants(test_string)
51     print(f"Vowels: {vowels}, Consonants: {consonants}") # Vowels: 3, Consonants: 7
52
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
PS C:\Users\hruth\OneDrive\Desktop\A.I.AC> & C:/Users/hruth/AppData/Local/Programs/Python/Python312/python.exe c:/Users/hruth/OneDrive/Desktop/A.I.AC/Ass_6.5.py
Vowels: 3, Consonants: 7
PS C:\Users\hruth\OneDrive\Desktop\A.I.AC>
```



Task Description #3 (AI-Assisted Code Completion Reflection Task)

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."

Expected Output:

- Complete AI-generated program.
- Review of AI suggestions quality.
- Short reflection on AI-assisted coding experience

```
#Task -3
def library_management_system():
    #Using classes, loops, conditional statements, functions, and data structures
    class Book:
        def __init__(self, title, author):
            self.title = title
            self.author = author
    class Library:
        def __init__(self):
            self.books = []
        def add_book(self, book):
            self.books.append(book)
        def display_books(self):
            for book in self.books:
                print(f"Title: {book.title}, Author: {book.author}")
    # Create a library instance
    library = Library()
    # Add books to the library
    library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
    library.add_book(Book("1984", "George Orwell"))
    library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald"))
    # Display the books in the library
    library.display_books()
if __name__ == "__main__":
    library_management_system()
```

```

62
63 #Task -3
64 def library_management_system():
65     #Using classes,loops, conditional statements, functions, and data structures
66     class Book:
67         def __init__(self, title, author):
68             self.title = title
69             self.author = author
70     class Library:
71         def __init__(self):
72             self.books = []
73         def add_book(self, book):
74             self.books.append(book)
75         def display_books(self):
76             for book in self.books:
77                 print(f"Title: {book.title}, Author: {book.author}")
78     # Create a library instance
79     library = Library()
80     # Add books to the library
81     library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
82     library.add_book(Book("1984", "George Orwell"))
83     library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald"))
84     # Display the books in the library
85     library.display_books()
86 if __name__ == "__main__":    library_management_system()

```

```

51 print(' Vowels: {vowels}, consonants: {consonants} ' .format(vowels=vowels, consonants=consonants))
52
53 #Task -3
54 def library_management_system():
55     #Using classes,loops, conditional statements, functions, and data structures
56     class Book:
57         def __init__(self, title, author):
58             self.title = title
59             self.author = author
60     class Library:
61         def __init__(self):
62             self.books = []
63         def add_book(self, book):
64             self.books.append(book)
65         def display_books(self):
66             for book in self.books:
67                 print(f"Title: {book.title}, Author: {book.author}")
68     # Create a library instance
69     library = Library()
70     # Add books to the library
71     library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
72     library.add_book(Book("1984", "George Orwell"))
73     library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald"))
74     # Display the books in the library
75     library.display_books()
76 if __name__ == "__main__":
77     library_management_system()
78

```

PROBLEMS

OUTPUT

TERMINAL

PORTS

DEBUG CONSOLE

```

PS C:\Users\hruth\OneDrive\Desktop\A.I.AC> & C:/Users/hruth/AppData/Local/Programs/Python/Python312/python.exe c:/Users/h
Title: To Kill a Mockingbird, Author: Harper Lee
Title: 1984, Author: George Orwell
Title: The Great Gatsby, Author: F. Scott Fitzgerald
PS C:\Users\hruth\OneDrive\Desktop\A.I.AC>

```



Task Description #4 (AI-Assisted Code Completion for Class-Based Attendance System)

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

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

#Task -4

```
def generate_attendance_system():
```

```
    """
```

Function to request AI to generate a Python class for managing student attendance.

Returns:

AI-generated code for an attendance management system using loops.

```
    """
```

```
class AttendanceSystem:
```

```
    def __init__(self):
```

```
        self.attendance = {}
```

```
    def mark_attendance(self, student_name):
```

```
        self.attendance[student_name] = "Present"
```

```
    def display_attendance(self):
```

```
        for student, status in self.attendance.items():
            print(f"{student}: {status}")
```

```
# Example usage
```

```
if __name__ == "__main__":
```

```
    attendance_system = AttendanceSystem()
```

```
    attendance_system.mark_attendance("Alice")
```

```
    attendance_system.mark_attendance("Bob")
```

```
    attendance_system.display_attendance()
```

```
generate_attendance_system()
```

```

#Task -4
def generate_attendance_system():
    """
    Function to request AI to generate a Python class for managing student attendance.

    Returns:
    AI-generated code for an attendance management system using loops.
    """

    class AttendanceSystem:
        def __init__(self):
            self.attendance = {}

        def mark_attendance(self, student_name):
            self.attendance[student_name] = "Present"

        def display_attendance(self):
            for student, status in self.attendance.items():
                print(f"{student}: {status}")

    # Example usage
    if __name__ == "__main__":
        attendance_system = AttendanceSystem()
        attendance_system.mark_attendance("Alice")
        attendance_system.mark_attendance("Bob")
        attendance_system.display_attendance()

```

```
77     library_management_system()
78
79 #Task -4
80 def generate_attendance_system():
81     """
82     Function to request AI to generate a Python class for managing student attendance.
83
84     Returns:
85     AI-generated code for an attendance management system using loops.
86     """
87     class AttendanceSystem:
88         def __init__(self):
89             self.attendance = {}
90
91         def mark_attendance(self, student_name):
92             self.attendance[student_name] = "Present"
93
94         def display_attendance(self):
95             for student, status in self.attendance.items():
96                 print(f"{student}: {status}")
97
98     # Example usage
99     if __name__ == "__main__":
100         attendance_system = AttendanceSystem()
101         attendance_system.mark_attendance("Alice")
102         attendance_system.mark_attendance("Bob")
103         attendance_system.display_attendance()
104     generate_attendance_system()
105
```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

Alice: Present
Bob: Present

The **Library Management System** is a solid start, using classes for books and the library. It's simple, clear, and easy to expand.

Good: Well-structured OOP design, easy to understand.

Needs Improvement: Could use error handling, search functionality, and data persistence for practicality.



Task Description #5 (AI-Based Code Completion for Conditional Menu Navigation)

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

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

Expected Output:

- AI-generated menu logic.
- Correct option handling.
- Output verification

```

#Task-5
def generate_atm_menu():
    """
    Function to request AI to generate a Python program that simulates an ATM
    menu.

    Returns:
    AI-generated code for an ATM menu with loops and conditionals.
    """
    class ATM:
        def __init__(self, balance=0):
            self.balance = balance

        def display_menu(self):
            print("ATM Menu:")
            print("1. Check Balance")
            print("2. Deposit")
            print("3. Withdraw")
            print("4. Exit")

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

        def deposit(self, amount):
            self.balance += amount
            print(f"${amount} deposited successfully.")

        def withdraw(self, amount):
            if amount > self.balance:
                print("Insufficient funds.")
            else:
                self.balance -= amount
                print(f"${amount} withdrawn successfully.")

    # Example usage
    if __name__ == "__main__":
        atm = ATM(1000) # Initial balance of $1000
        while True:
            atm.display_menu()
            choice = input("Enter your choice: ")
            if choice == '1':
                atm.check_balance()
            elif choice == '2':
                amount = float(input("Enter amount to deposit: "))

```

```

        atm.deposit(amount)
    elif choice == '3':
        amount = float(input("Enter amount to withdraw: "))
        atm.withdraw(amount)
    elif choice == '4':
        print("Thank you for using the ATM. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.")
generate_atm_menu()

```

```

Ass_6.5.py > generate_atm_menu
106 #Task-5
107 def generate_atm_menu():
108     """
109     Function to request AI to generate a Python program that simulates an ATM menu.
110
111     Returns:
112     AI-generated code for an ATM menu with loops and conditionals.
113     """
    class ATM:
        def __init__(self, balance=0):
            self.balance = balance

        def display_menu(self):
            print("ATM Menu:")
            print("1. Check Balance")
            print("2. Deposit")
            print("3. Withdraw")
            print("4. Exit")

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

        def deposit(self, amount):
            self.balance += amount
            print(f"${amount} deposited successfully.")

        def withdraw(self, amount):
            if amount > self.balance:
                print("Insufficient funds.")
            else:
                self.balance -= amount
                print(f"${amount} withdrawn successfully.")

```

```
Bob: Present
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 1
Your current balance is: $1000
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 2
Enter amount to deposit: 200
$200.0 deposited successfully.
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 3
Enter amount to withdraw: 100
$100.0 withdrawn successfully.
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 1
Your current balance is: $1100.0
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 4
Thank you for using the ATM. Goodbye!
PS C:\Users\hruth\OneDrive\Desktop\A.I.AC>
```