

## ASSIGNMENT-6.5

NAME: B. Akhira Nandhini

HTNO:2303A51516

BATCH:22

### AI ASSISTANT CODING

#### Task Description1: (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.

#### INPUT:

The screenshot shows a software interface for AI-assisted coding. On the left, there's a sidebar with project files: .github, 6.5.py, task1.py (selected), task2.py, day1.py, day2.py, day3.py, and day4.py. The main workspace displays a Python script named 6.5.py with code completion suggestions. The code defines a function `voting_eligibility` that checks age and citizenship status. The AI has completed several lines of code, including handling non-integer ages and specific citizenship cases. Below the code editor, the terminal shows command-line interactions where the user enters their age and citizenship status, and the script prints out whether they are eligible to vote or not. The bottom status bar shows system information like battery level, network, and date/time.

```
def voting_eligibility(age,citizenship):
    if age<0:
        return"age can not be less than 0"
    if not isinstance(citizenship,str):
        return"citizenship status must be a string"
    if not isinstance(age,int):
        return"age must be an number"
    if age<18:
        return"you are not eligible to vote"
    elif citizenship!="indian":
        return"you are not eligible to vote"
    else:
        return"you are eligible to vote"
age=int(input("enter your age:"))
citizenship=input("enter your citizenship status:")
```

```

File Edit Selection View Go Run Terminal Help
AI assisted coding
File Edit Selection View Go Run Terminal Help
AI assisted coding
Upgrade to Pro
Search Agents...
New Agent
AI ASSISTED CODING
.github
6.5.py
task1.py
task2.py
day1.py
day2.py
day3.py
day4.py
6.5.py > task1.py > voting_eligibility
def voting_eligibility(age,citizenship):
else:
    return"you are eligible to vote"
age=int(input("enter your age:"))
citizenship=input("enter your citizenship status:")
print(voting_eligibility(age,citizenship))

Problems Output Debug Console Terminal Ports
powershell
PS C:\Users\abhin\OneDrive\Desktop\AI assisted coding> & C:\Users\abhin\AppData\Local\Programs\Python\Python313\python.exe
eDrive\Desktop\AI assisted coding\6.5.py\task1.py
enter your age:15
you are eligible to vote
PS C:\Users\abhin\OneDrive\Desktop\AI assisted coding> & C:\Users\abhin\AppData\Local\Programs\Python\Python313\python.exe
eDrive\Desktop\AI assisted coding\6.5.py\task1.py
enter your age:20
you are not eligible to vote
PS C:\Users\abhin\OneDrive\Desktop\AI assisted coding>

```

30°C Sunny

## EXPLANATION:

- The AI-generated Python program was used to design a voting eligibility check with proper input validation and clear decision logic.
- The code was carefully reviewed line by line to understand how age and citizenship inputs are validated, including handling missing, invalid, and incorrect values.
- Logical flaws and potential errors were identified and addressed by adding appropriate condition checks and exception handling.
- The program was further refined to improve readability and maintainability through meaningful variable names, structured validation, and clear comments.
- Responsible use of AI tools was ensured by verifying the logic, correcting mistakes, and fully understanding the code rather than copying the AI-generated output without evaluation.

## 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.

The screenshot shows a code editor interface with several tabs open. The main tab displays a Python script named `task2.py` containing the following code:

```
def count_vowels_consonants(text):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

    for ch in text:
        if ch.isalpha():          # check only letters
            if ch in vowels:
                vowel_count += 1
            else:
                consonant_count += 1

    return vowel_count, consonant_count

# Example usage
string = "Hello World"
vowels, consonants = count_vowels_consonants(string)

print("String:", string)
print("Vowels:", vowels)
print("Consonants:", consonants)
```

Below the code, the terminal window shows the output of running the script with the string "Hello World". The output is:

```
PS C:\Users\abhin\OneDrive\Desktop\AI assisted coding> & C:\Users\abhin\AppData\Local\Programs\Python\Python313\python.exe "C:/users/abhin/onedrive/Desktop/AI assisted coding/6.5.py/task2.py"
String: Hello world
Vowels: 3
Consonants: 7
PS C:\Users\abhin\OneDrive\Desktop\AI assisted coding>
```

## EXPLANATION:

- The AI tool was used to generate a Python program that applies a user-defined function, loop-based logic, and conditional statements to count vowels and consonants in a string.
- The generated code was carefully examined line by line to understand how input validation, character checking, and counting logic work together to produce correct results.
- During the review process, potential issues such as empty inputs, non-string values, and invalid characters were identified and handled appropriately to ensure logical correctness.
- The program was further refined to improve readability and efficiency through clear comments, structured conditions, and meaningful variable names.

## 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.

Input:

The screenshot shows a software interface for AI-assisted coding. The top menu includes File, Edit, Selection, View, Go, Run, Terminal, and Help. A toolbar with icons for file operations is visible. The main window has tabs for day3.py, task1.py, task2.py, and task3.py, with task3.py currently selected. The code editor displays the following Python script:

```
'''Generate a Python program for a simple library management system using classes, loops, and conditional statements.
The program should allow the user to add books, remove books, display available books, and exit the system.'''
class Library:
    def __init__(self):
        self.books = []
    def add_book(self, book_name):
        self.books.append(book_name)
        print(f"Book '{book_name}' added successfully.")
    def remove_book(self, book_name):
        if book_name in self.books:
            self.books.remove(book_name)
            print(f"Book '{book_name}' removed successfully.")
        else:
            print("Book not found in the library.")
    def display_books(self):
        if len(self.books) == 0:
            print("No books available in the library.")
        else:
            print("Available Books:")
            for book in self.books:
                print("-", book)
# Main Program
library = Library()
while True:
    print("\n-- Library Management System --")
    print("1. Add Book")
    print("2. Remove Book")
    print("3. Display Books")
    print("4. Exit")
    choice = input("Enter your choice: ")
    if choice == "1":
        name = input("Enter book name: ")
        library.add_book(name)
    elif choice == "2":
        name = input("Enter book name: ")
        library.remove_book(name)
    elif choice == "3":
        library.display_books()
    elif choice == "4":
        break
print("Thank you for using the Library Management System!")
```

The bottom status bar shows the cursor tab, line 46, column 1, spaces: 4, UTT-B, CRLF, Python, Python 3.10 (64-bit), and a date/time stamp of 31-01-2026.

The screenshot shows a dark-themed IDE window titled "AI assisted coding". The left sidebar lists files under "AI ASSISTED CODING" including .github, 6.5.py, task1.py, task2.py, task3.py, day1.py, day2.py, day3.py, and day4.py. The main editor area contains the following Python code:

```
6.5.py : task3.py > ...
24     library = Library()
25
26     while True:
27         print("\n--- Library Management System ---")
28         print("1. Add Book")
29         print("2. Remove Book")
30         print("3. Display Books")
31         print("4. Exit")
32         choice = input("Enter your choice: ")
33         if choice == "1":
34             name = input("Enter book name: ")
35             library.add_book(name)
36         elif choice == "2":
37             name = input("Enter book name to remove: ")
38             library.remove_book(name)
39         elif choice == "3":
40             library.display_books()
41         elif choice == "4":
42             print("Thank you for using the Library Management System.")
43             break
44         else:
45             print("Invalid choice. Please try again.")

46 Ctrl+L to chat, Ctrl+K to generate
```

The status bar at the bottom shows "Cursor Tab Ln 46, Col 1 Spaces: 4 UTF-8 CRLF Python Python 3.13 (64-bit)" and the date "31-01-2026".

## Output:

The screenshot shows the same IDE window with the "Terminal" tab selected. The terminal output displays the execution of the Python script and its interaction with the user:

```
PS C:\Users\abhin\OneDrive\Desktop\AI assisted coding & C:\Users\abhin\AppData\Local\Programs\Python\Python313\python.exe "c:/users/abhin/onedrive/desktop/ai assisted coding/6.5.py/task3.py"
--- Library Management System ---
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice: 1
Enter book name: python
Book 'python' added successfully.

--- Library Management System ---
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice: 2
Enter book name to remove: python
Book 'python' removed successfully.

--- Library Management System ---
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice: 3
No books available in the library.

--- Library Management System ---
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Enter your choice: 4
Thank you for using the Library Management System.
```

The status bar at the bottom shows "Cursor Tab Ln 36, Col 24 (320 selected) Spaces: 4 UTF-8 CRLF Python Python 3.13 (64-bit)" and the date "31-01-2026".

## Explanation:

- The AI-generated Python program was used to design a simple library management system using a class-based approach along with loops and conditional statements for menu-driven operations.

- The code was carefully reviewed line by line to understand how object-oriented concepts, such as classes, methods, and instance variables, manage book records and availability status.
- Logical conditions were examined to ensure correct handling of operations like adding, displaying, issuing, and returning books, while also preventing invalid actions such as issuing an already issued book.
- The program was refined for better readability and maintainability through meaningful method names, clear comments, and structured input validation within the menu loop.
- Responsible use of AI tools was demonstrated by verifying the generated logic, handling user input errors properly, and ensuring a clear understanding of the program's functionality rather than relying on the AI output without evaluation

**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.

Input:

```

File Edit Selection View Go Run Terminal Help
AI assisted coding
Upgrade to Pro Search Agents... New Agent
AI ASSISTED CODING
.github
6.5.py
task1.py task2.py task3.py task4.py
day1.py day2.py day3.py day4.py
6.5.py : task4.py > ...
1  '''Generate a python class to manage student attendance.
2  The class should allow marking attendance for multiple students using a loop, and display the attendance record.
3  Include options to mark present and view attendance.'''
4  class AttendanceSystem:
5      def __init__(self):
6          self.attendance = {} # Dictionary to store student attendance
7
8      def mark_attendance(self, students):
9          print("Mark attendance: P for present, A for absent")
10         for student in students:
11             status = input(f"Is {student} present? (P/A): ").upper()
12             if status == 'P':
13                 self.attendance[student] = "Present"
14             else:
15                 self.attendance[student] = "Absent"
16
17      def display_attendance(self):
18          print("\n--- Attendance Record ---")
19          if not self.attendance:
20              print("No attendance marked yet.")
21          else:
22              for student, status in self.attendance.items():
23                  print(f"{student}: {status}")
24
25
26      # Example usage
27      students_list = ["Alice", "Bob", "Charlie"]
28      attendance_system = AttendanceSystem()
29
30      # Mark attendance for students
31      attendance_system.mark_attendance(students_list)
32
33      # Display the attendance record
34      attendance_system.display_attendance()
35

```

DIRECTORY: AI assisted coding

Timeline: 31-01-2026

Cursor Tab: Ln 35, Col 1 Spaces: 4 UTF-8 CRLF Python Python 3.13 (64-bit)

ENG IN 14:46

Output:

```

:/Users/abhin/OneDrive/Desktop/AI assisted coding/6.5.py/task4.py"
Mark attendance: P for present, A for absent
Is Alice present? (P/A): p
Is Bob present? (P/A): a
Is Charlie present? (P/A): p

--- Attendance Record ---
Alice: Present
Bob: Absent
Charlie: Present
PS C:\Users\abhin\OneDrive\Desktop\AI assisted coding>

```

Explanation:

- The AI-generated Python program was used to develop an attendance management system using a class-based structure along with loops and conditional statements for menu-driven operations.
- The code was reviewed in detail to understand how methods are used to add students, mark attendance as present or absent, and display attendance records using dictionary-based storage.
- Conditional checks were analysed to ensure proper handling of invalid inputs such as empty names, incorrect attendance status, and non-existing students. The implementation was refined to improve readability.

and maintainability by using meaningful method names, clear comments, and structured control flow within the loop.

- Responsible use of AI tools was demonstrated by validating the generated logic, handling edge cases correctly, and ensuring a clear understanding of the program's functionality rather than relying on the AI output without verification.

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

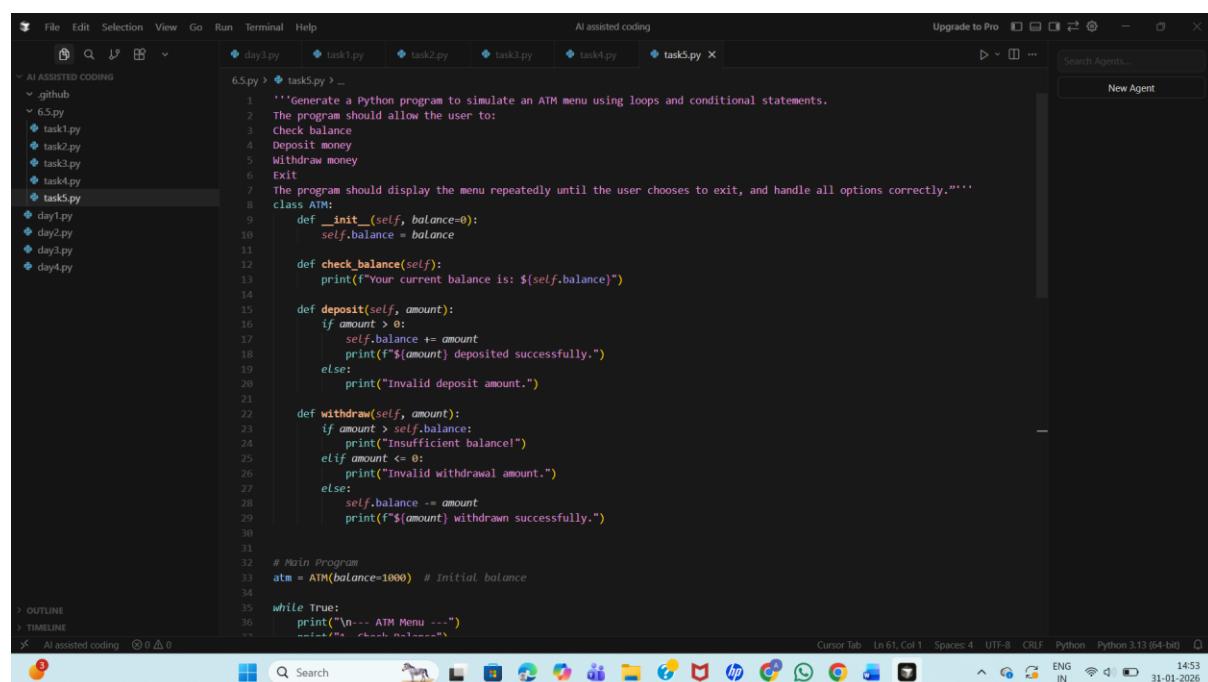
Task: Use an AI tool to complete a navigation menu.

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.

Input:



The screenshot shows a software interface for AI-assisted coding. The main window displays a Python script named `task5.py` which generates an ATM menu. The code includes docstrings explaining the purpose of the program and its menu options. It defines a class `ATM` with methods for checking balance, depositing money, and withdrawing money. The program uses a loop to repeatedly display the menu until the user chooses to exit. The interface has a toolbar at the top with various icons, a file tree on the left showing other files like `day1.py` through `day4.py`, and a status bar at the bottom with system information.

```
'''Generate a Python program to simulate an ATM menu using loops and conditional statements.
The program should allow the user to:
Check balance
Deposit money
Withdraw money
Exit
The program should display the menu repeatedly until the user chooses to exit, and handle all options correctly.'''
class ATM:
    def __init__(self, balance=0):
        self.balance = balance
    def check_balance(self):
        print(f"Your current balance is: ${self.balance}")
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"${amount} deposited successfully.")
        else:
            print("Invalid deposit amount.")
    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance!")
        elif amount <= 0:
            print("Invalid withdrawal amount.")
        else:
            self.balance -= amount
            print(f"${amount} withdrawn successfully.")
# Main Program
atm = ATM(balance=1000) # Initial balance
while True:
    print("\n--- ATM Menu ---")
    print("1. Check Balance")
    print("2. Deposit Money")
    print("3. Withdraw Money")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")
    if choice == "1":
        atm.check_balance()
    elif choice == "2":
        amount = float(input("Enter deposit amount: "))
        atm.deposit(amount)
    elif choice == "3":
        amount = float(input("Enter withdrawal amount: "))
        atm.withdraw(amount)
    elif choice == "4":
        print("Thank you for using the ATM. Goodbye!")
        break
    else:
        print("Invalid choice. Please enter 1, 2, 3, or 4.")
```

```

File Edit Selection View Go Run Terminal Help
AI assisted coding
Upgrade to Pro Search Agents... New Agent
AI ASSISTED CODING
github
6.5.py
task1.py
task2.py
task3.py
task4.py
task5.py
day1.py
day2.py
day3.py
day4.py
day5.py
6.5.py > task5.py > ...
32 # Main Program
33 atm = ATM(balance=1000) # Initial balance
34
35 while True:
36     print("\n--- ATM Menu ---")
37     print("1. Check Balance")
38     print("2. Deposit Money")
39     print("3. Withdraw Money")
40     print("4. Exit")
41
42     choice = input("Enter your choice: ")
43
44     if choice == "1":
45         atm.check_balance()
46
47     elif choice == "2":
48         amount = float(input("Enter amount to deposit: "))
49         atm.deposit(amount)
50
51     elif choice == "3":
52         amount = float(input("Enter amount to withdraw: "))
53         atm.withdraw(amount)
54
55     elif choice == "4":
56         print("Thank you for using the ATM. Goodbye!")
57         break
58
59     else:
60         print("Invalid choice. Please try again.")
61

```

DIALOGS: OUTLINE TIMELINE AI assisted coding

CURSOR TAB: Ln 61, Col 1 SPACES: 4 UTF-8 CR/LF: Python Python 3.13 (64-bit)

ENGLISH IN 14:54 31-01-2026

## Output:

```

File Edit Selection View Go Run Terminal Help
AI assisted coding
Upgrade to Pro Search Agents... New Agent
AI ASSISTED CODING
github
6.5.py
task1.py
task2.py
task3.py
task4.py
task5.py
day1.py
day2.py
day3.py
day4.py
day5.py
6.5.py > task5.py > ...
Problems Output Debug Console Terminal Ports
PS C:\Users\abhin\Desktop\AI assisted coding & C:\Users\abhin\AppData\Local\Programs\Python\Python313\python.exe "c:/users/abhin/onedevice/Desktop/AI assisted coding/6.5.py/tasks.py"
--- ATM Menu ---
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 1
Your current balance is: $1000
--- ATM Menu ---
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 2
Enter amount to deposit: 300
$300.0 deposited successfully.
--- ATM Menu ---
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 3
Enter amount to withdraw: 50000
Insufficient balance!
--- ATM Menu ---
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 4
Thank you for using the ATM. Goodbye!

```

CHOOSE TO GENERATE COMMAND

CURSOR TAB: Ln 61, Col 1 SPACES: 4 UTF-8 CR/LF: Python Python 3.13 (64-bit)

ENGLISH IN 14:53 31-01-2026

## Explanation:

- The AI-generated Python program was used to simulate an ATM system using loops and conditional statements to provide a menu-driven interface.
- The code was carefully examined to understand how the loop keeps the menu running until the user chooses to exit and how conditional

branches handle balance inquiry, deposit, and withdrawal operations. Input validation was analysed to ensure that invalid menu selections, non-numeric inputs, negative amounts, and insufficient balance cases are handled correctly.

- The program structure was reviewed to identify and prevent logical errors, while clear comments and meaningful variable names were used to improve readability and maintainability.
- Responsible use of AI tools was demonstrated by verifying the generated logic, testing different transaction scenarios, and ensuring correct behaviour rather than relying on the AI output without evaluation.