

AI ASSISTANT CODING

ASSIGNMENT-6

Name: Kommu Madhupriya

Hallticket:2303A51583

Batch:22

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.

code:

```
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
task1.py > voting_eligibility
6.5.py : 1  '''generate python code to check voting eligibility based on age and citizenship status without using a function
the code should take input from the user.
2  include the case of numbers less than 0.it should return "age can not be less than 0"
3  include the case of citizenship status not being a string.it should return "citizenship status must be a string"
4  include the case of non-numeric for age.it should return "age must be an number"
5  ...
6  ...
7  ...
8  def voting_eligibility(age,citizenship):
9      if age<0:
10          return"age can not be less than 0"
11      if not isinstance(citizenship,str):
12          return"citizenship status must be a string"
13      if not isinstance(age,int):
14          return"age must be an number"
15      if age<18:
16          return"you are not eligible to vote"
17      elif citizenship=="indian":
18          return"you are not eligible to vote"
19      else:
20          return"you are eligible to vote"
21  age=int(input("enter your age:"))
22  citizenship=input("enter your citizenship status:")
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/task1.py"
enter your age:15
enter your citizenship status:indian
you are not eligible to vote
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/task1.py"
enter your age:20
enter your citizenship status:not indian
you are not eligible to vote
PS C:\Users\abhin\OneDrive\Desktop\AI assisted coding>

```

Problems Output Debug Console Terminal Ports

Ctrl+K to generate command

Cursor Tab In 10, Col 43 Spaces: 4 UTF-B CR/LF { } Python Python 3.13 (64-bit) ⌂

ENG IN 14:33 30-01-2026

```

def voting_eligibility(age,citizenship):
    if age >= 18:
        return "you are eligible to vote"
    else:
        return "you are not eligible to vote"
age=int(input("enter your age:"))
citizenship=input("enter your citizenship status:")
print(voting_eligibility(age,citizenship))

```

enter your age:15
enter your citizenship status:indian
you are not eligible to vote
enter your age:20
enter your citizenship status:not indian
you are not eligible to vote

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.

code:

```

1 def count_vowels_consonants(text):
2     vowels = "aeiouAEIOU"
3     vowel_count = 0
4     consonant_count = 0
5
6     for ch in text:
7         if ch.isalpha():          # check only letters
8             if ch in vowels:
9                 vowel_count += 1
10            else:
11                consonant_count += 1
12
13    return vowel_count, consonant_count
14
15
16 # Example usage
17 string = "Hello World"
18 vowels, consonants = count_vowels_consonants(string)
19
20 print("String:", string)
21 print("Vowels:", vowels)
22 print("Consonants:", consonants)

```

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.

code:

```

1  '''Generate a Python program for a simple library management system using classes, loops, and conditional statements.
2  The program should allow the user to add books, remove books, display available books, and exit the system.'''
3  class Library:
4      def __init__(self):
5          self.books = []
6      def add_book(self, book_name):
7          self.books.append(book_name)
8          print(f"Book '{book_name}' added successfully.")
9      def remove_book(self, book_name):
10         if book_name in self.books:
11             self.books.remove(book_name)
12             print(f"Book '{book_name}' removed successfully.")
13         else:
14             print("Book not found in the library.")
15
16     def display_books(self):
17         if len(self.books) == 0:
18             print("No books available in the library.")
19         else:
20             print("Available Books:")
21             for book in self.books:
22                 print("-", book)
23
24 # Main Program
25 library = Library()
26
27 while True:
28     print("\n--- Library Management System ---")
29     print("1. Add Book")
30     print("2. Remove Book")
31     print("3. Display Books")
32     print("4. Exit")
33     choice = input("Enter your choice: ")
34     if choice == "1":
35         name = input("Enter book name: ")
36         library.add_book(name)
37     elif choice == "2":
38         name = input("Enter book name to remove: ")
39         library.remove_book(name)
40     elif choice == "3":
41         library.display_books()
42     elif choice == "4":
43         print("Thank you for using the Library Management System.")
44         break
45     else:
46         print("Invalid choice. Please try again.")

```

output:

The screenshot shows a Python code editor interface with several tabs open. The tabs include 'day3.py', 'task1.py', 'task2.py', and 'task3.py'. The 'task3.py' tab is active, displaying a menu-based library management system. The code uses a class-based approach with methods for adding, removing, and displaying books. It includes validation for user input and handles edge cases like removing a book that doesn't exist.

```

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
task3.py
day1.py
day2.py
day3.py
day4.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.

```

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."

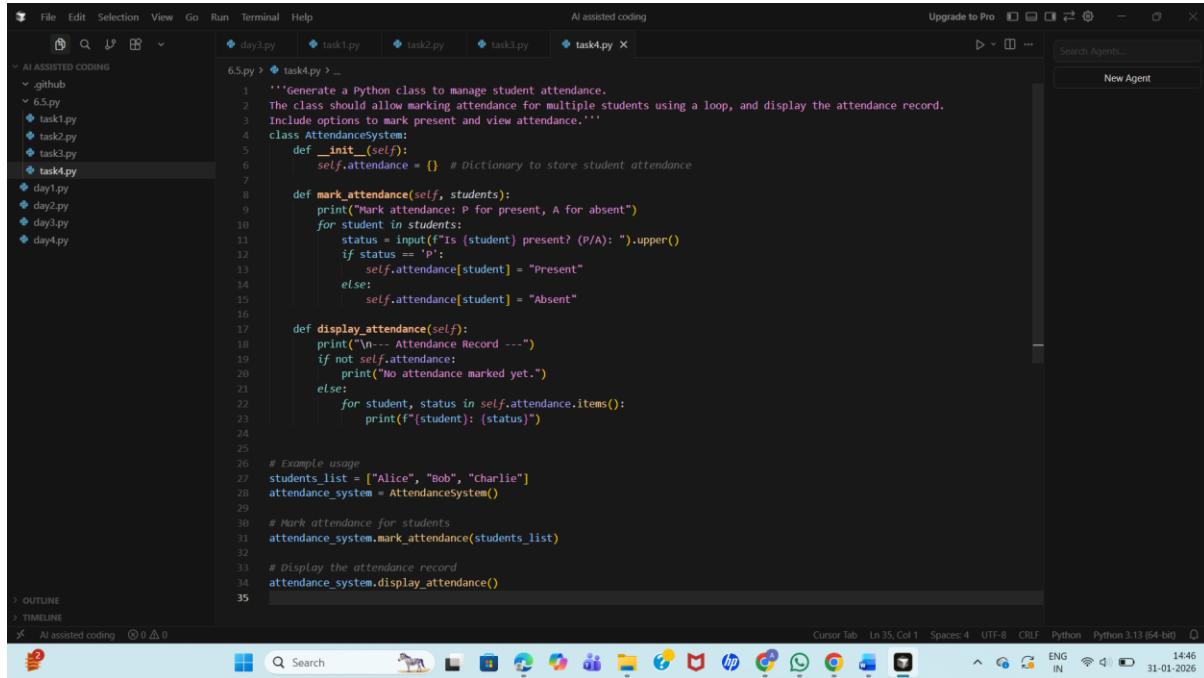
Expected Output:

- AI-generated attendance logic.

- Correct display of attendance.

- Test cases.

code:



```

File Edit Selection View Go Run Terminal Help
AI assisted coding
Upgrade to Pro Search Agents... New Agent
AI ASSISTED CODING
6.5.py task1.py task2.py task3.py task4.py
AI assisted coding
6.5.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

```

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.

Code:

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Includes icons for file operations like Open, Save, Find, and Run.
- Search Bar:** Search Agents... and New Agent.
- Code Editor Area:** Shows a Python script named `task5.py`. The code is as follows:

```
6.5.py > task5.py > ...
1  """Generate a Python program to simulate an ATM menu using loops and conditional statements.
2  The program should allow the user to:
3  Check balance
4  Deposit money
5  Withdraw money
6  Exit
7  The program should display the menu repeatedly until the user chooses to exit, and handle all options correctly."""
8 class ATM:
9     def __init__(self, balance=0):
10         self.balance = balance
11
12     def check_balance(self):
13         print(f"Your current balance is: ${self.balance}")
14
15     def deposit(self, amount):
16         if amount > 0:
17             self.balance += amount
18             print(f"${amount} deposited successfully.")
19         else:
20             print("Invalid deposit amount.")
21
22     def withdraw(self, amount):
23         if amount > self.balance:
24             print("Insufficient balance!")
25         elif amount <= 0:
26             print("Invalid withdrawal amount.")
27         else:
28             self.balance -= amount
29             print(f"${amount} withdrawn successfully.")
30
31
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 (1-4): ")
43
44     if choice == "1":
45         atm.check_balance()
46     elif choice == "2":
47         amount = float(input("Enter deposit amount: "))
48         atm.deposit(amount)
49     elif choice == "3":
50         amount = float(input("Enter withdrawal amount: "))
51         atm.withdraw(amount)
52     elif choice == "4":
53         print("Thank you for using the ATM. Goodbye!")
54         break
55     else:
56         print("Invalid choice. Please enter 1, 2, 3, or 4.")
```

The code is a Python program to simulate an ATM menu. It defines a `ATM` class with methods for checking balance, depositing, withdrawing, and exiting. The main program runs a loop where it prints the menu, gets user input, and calls the appropriate methods. The code is annotated with comments explaining its purpose and structure.

```

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 CRLF: Python: Python 3.13 (64-bit)

ENG 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\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/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: 5000
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!

```

DIALOGS: OUTLINE TIMELINE AI assisted coding

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

ENG 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.