Assignment–6.5

Name : K.Tejaswini

Batch:21

HtNo:2303A51425

TaskDescription-1

-(AI-BasedCodeCompletionforConditionalEligibilityCheck) Task:

Use an AI tool to generate eligibility logic.

Prompt:

“GeneratePythoncodetocheckvotingeligibilitybasedonageandcitizenship.” Expected

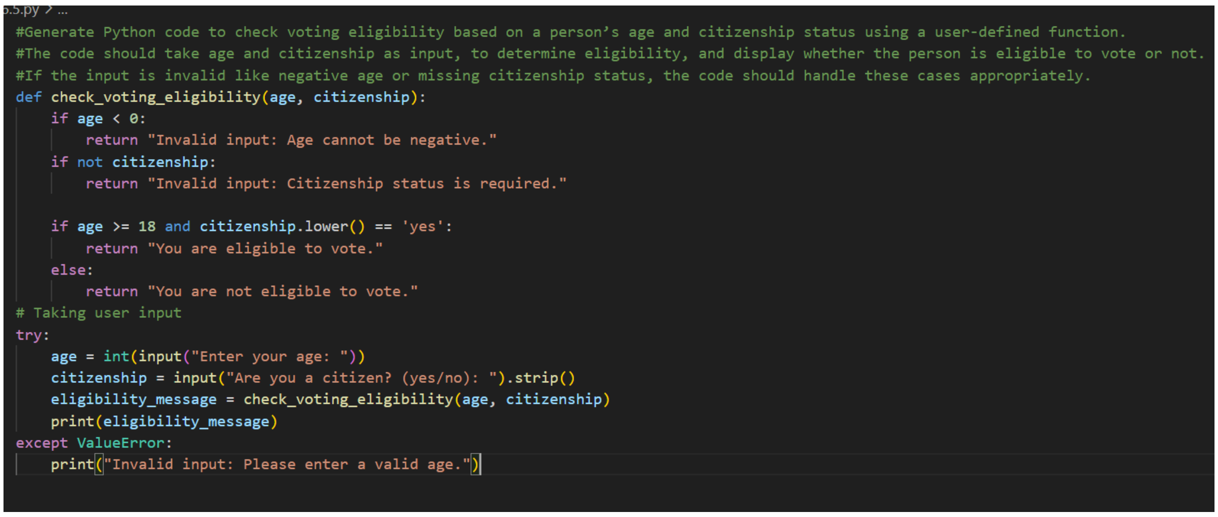
Output:

• AI-generatedconditionallogic.

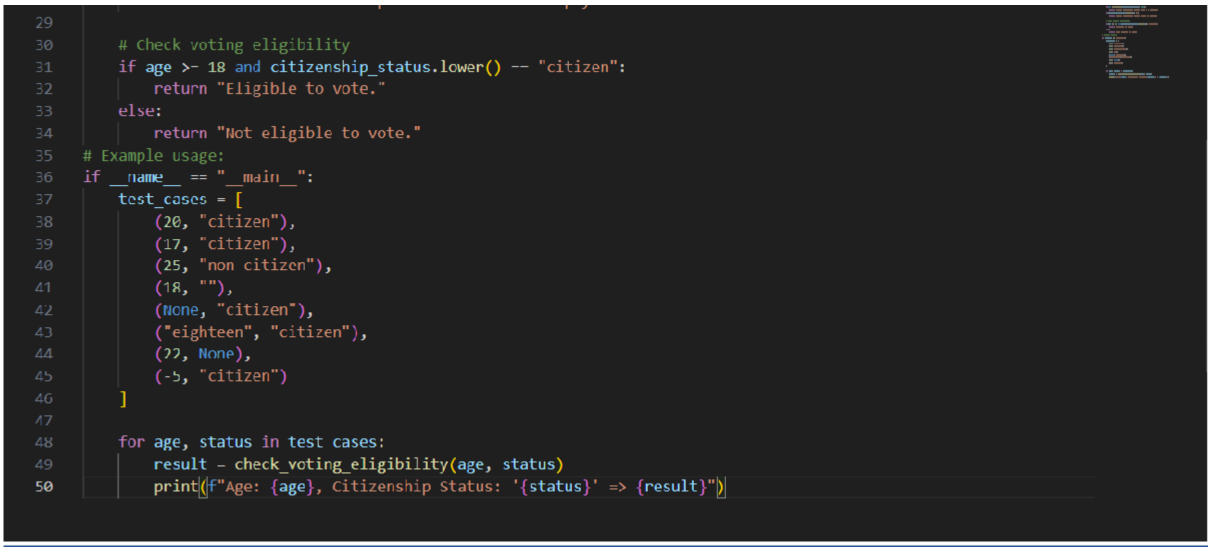
• Correcteligibilitydecisions.

• Explanationofconditions.

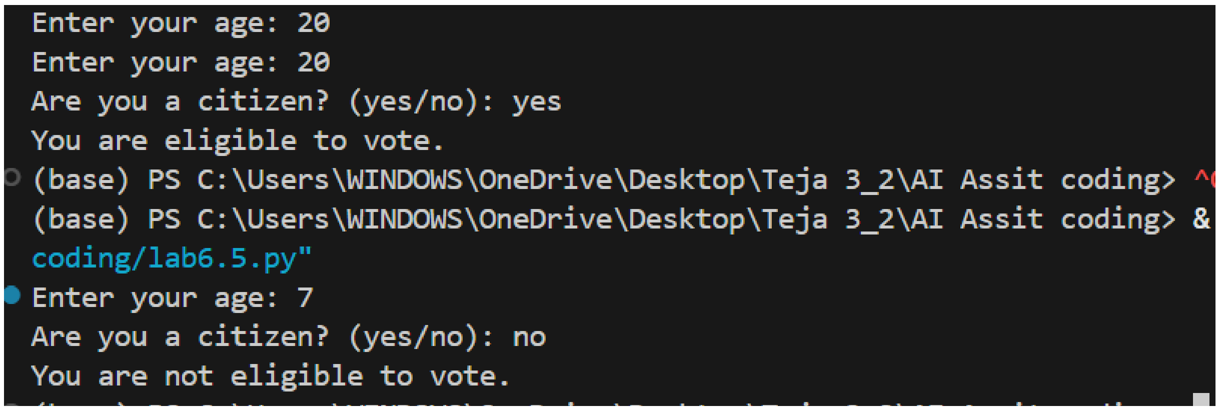
Code:



```
#Generate Python code to check voting eligibility based on a person’s age and citizenship status using a user-defined function.
#The code should take age and citizenship as input, to determine eligibility, and display whether the person is eligible to vote or not.
#If the input is invalid like negative age or missing citizenship status, the code should handle these cases appropriately.
def check_voting_eligibility(age, citizenship):
    if age < 0:
        return "Invalid input: Age cannot be negative."
    if not citizenship:
        return "Invalid input: Citizenship status is required."

    if age >= 18 and citizenship.lower() == 'yes':
        return "You are eligible to vote."
    else:
        return "You are not eligible to vote."
# Taking user input
try:
    age = int(input("Enter your age: "))
    citizenship = input("Are you a citizen? (yes/no): ").strip()
    eligibility_message = check_voting_eligibility(age, citizenship)
    print(eligibility_message)
except ValueError:
    print("Invalid input: Please enter a valid age.")
```


```python

    # Check voting eligibility
    if age >= 18 and citizenship_status.lower() == "citizen":
        return "Eligible to vote."
    else:
        return "Not eligible to vote."
# Example usage:
if __name__ == "__main__":
    test_cases = [
        (20, "citizen"),
        (17, "citizen"),
        (25, "non citizen"),
        (18, ""),
        (None, "citizen"),
        ("eighteen", "citizen"),
        (22, None),
        (-5, "citizen")
    ]

    for age, status in test_cases:
        result = check_voting_eligibility(age, status)
        print(f"Age: {age}, Citizenship Status: '{status}' => {result}")
```

Output:


```
Enter your age: 20
Enter your age: 20
Are you a citizen? (yes/no): yes
You are eligible to vote.
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> ^
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> &
coding/lab6.5.py"
Enter your age: 7
Are you a citizen? (yes/no): no
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 understandhowageandcitizenshipinputsarevalidated,includinghandling 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 verifyingthelogic,correctingmistakes,andfullyunderstandingthecoderatherthancopying the AI-generated output without evaluation.

TaskDescription-2

-(AI-BasedCodeCompletionforLoop-BasedStringProcessing) Task:

Use an AI tool to process strings using loops.

Prompt:

“GeneratePythoncodetocountvowelsandconsonantsinastringusingaloop.” Expected

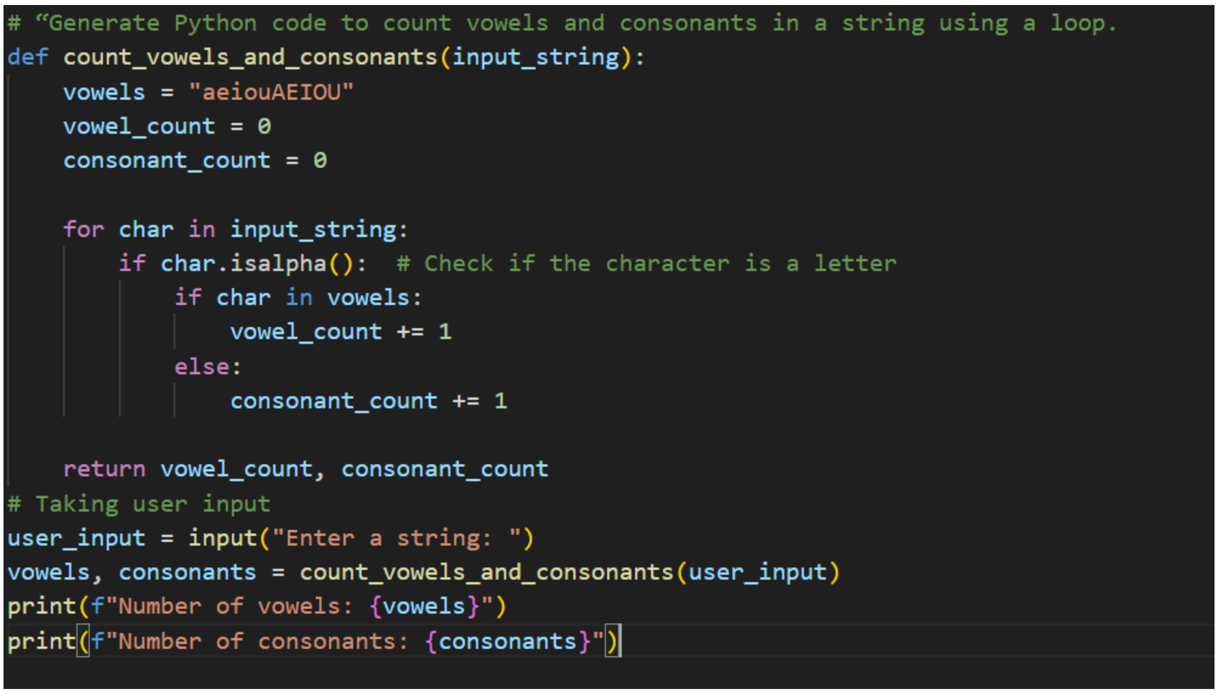
Output:

• AI-generatedstringprocessinglogic.

• Correctcounts.

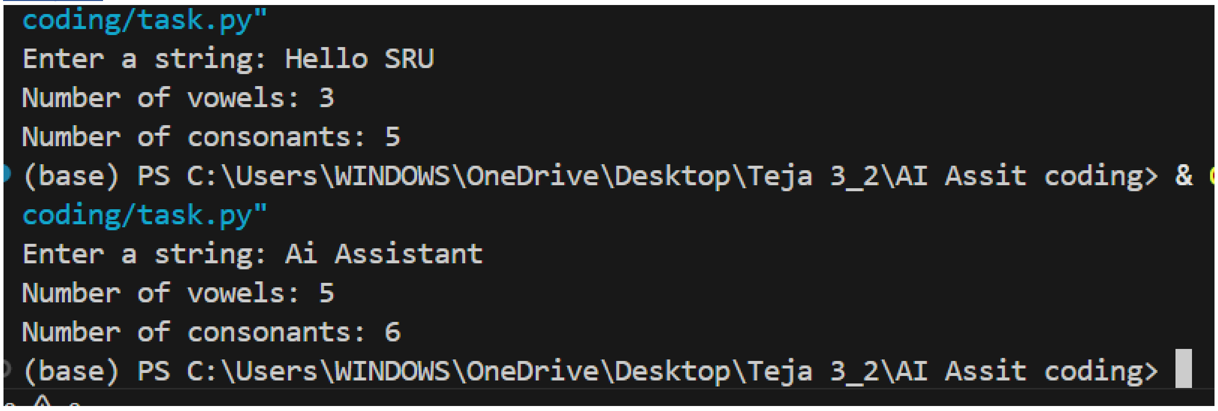
• Outputverification.

Code:



```python
# "Generate Python code to count vowels and consonants in a string using a loop.
def count_vowels_and_consonants(input_string):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

    for char in input_string:
        if char.isalpha():  # Check if the character is a letter
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1

    return vowel_count, consonant_count
# Taking user input
user_input = input("Enter a string: ")
vowels, consonants = count_vowels_and_consonants(user_input)
print(f"Number of vowels: {vowels}")
print(f"Number of consonants: {consonants}")
```

Output:



```
coding/task.py"
Enter a string: Hello SRU
Number of vowels: 3
Number of consonants: 5
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit coding> &
coding/task.py"
Enter a string: Ai Assistant
Number of vowels: 5
Number of consonants: 6
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI Assit 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. Thegeneratedcodewascarefullyexaminedlinebylinetounderstandhowinput validation, characterchecking,andcountinglogicworktogethertoproducecorrectresults.Duringthe review process, potential issues such as empty inputs, non-string values, and invalid characters were identified and handled appropriately to ensure logical correctness. The programwasfurtherrefinedtoimprovereadabilityandefficiencythroughclearcomments, structured conditions, and meaningful variable names. Responsible use of AI tools was demonstrated by validating the generated solution, correcting errors, and ensuring a clear understanding of the logic rather than using the output without evaluation.

TaskDescription-3

-(AI-AssistedCodeCompletionReflectionTask)

Task:UseanAItooltogenerateacompleteprogramusingclasses,loops,andconditionals. Prompt:

“GenerateaPythonprogramforalibrarymanagementsystemusingclasses,loops,and conditional statements.”

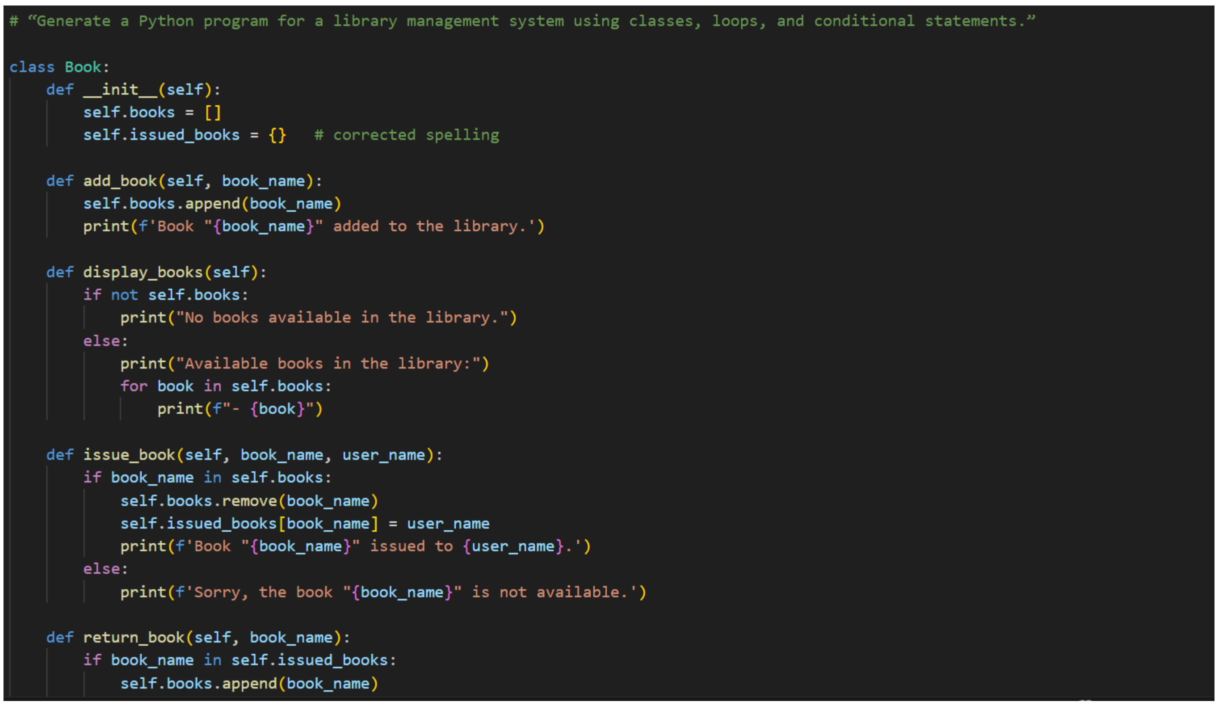
ExpectedOutput:

• CompleteAI-generatedprogram.

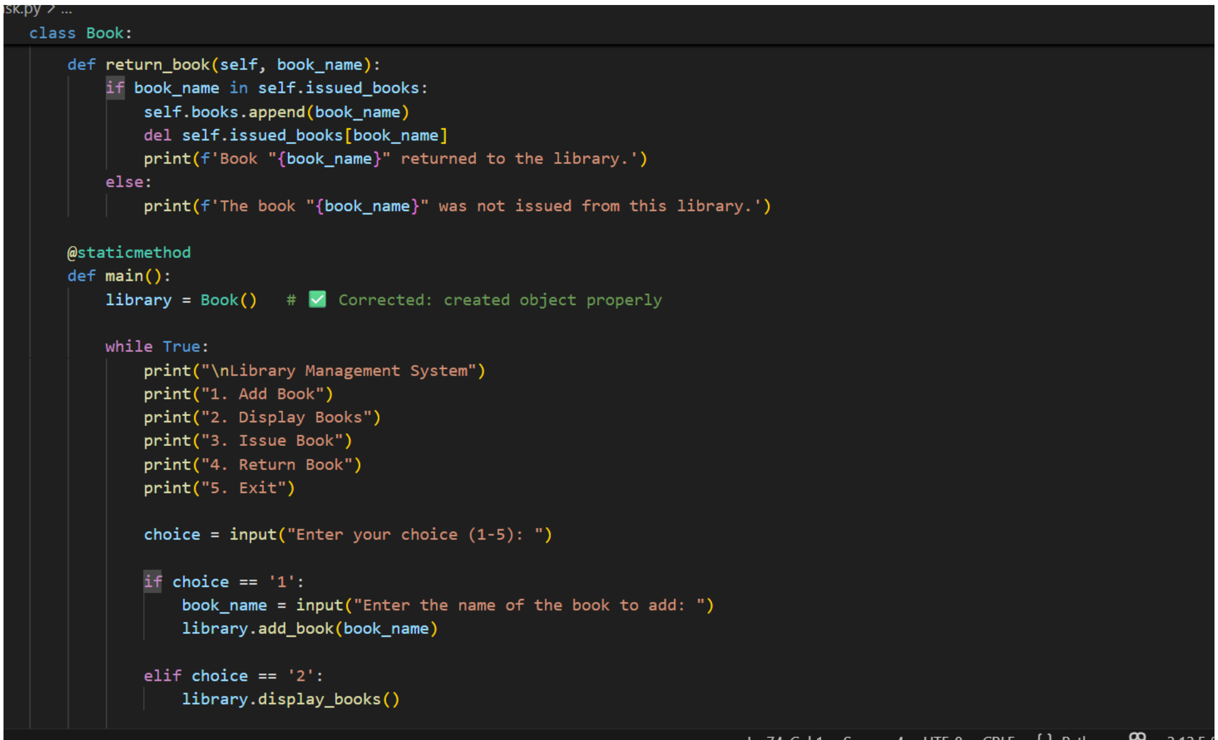
• ReviewofAIsuggestionsquality.

• ShortreflectiononAI-assistedcodingexperience

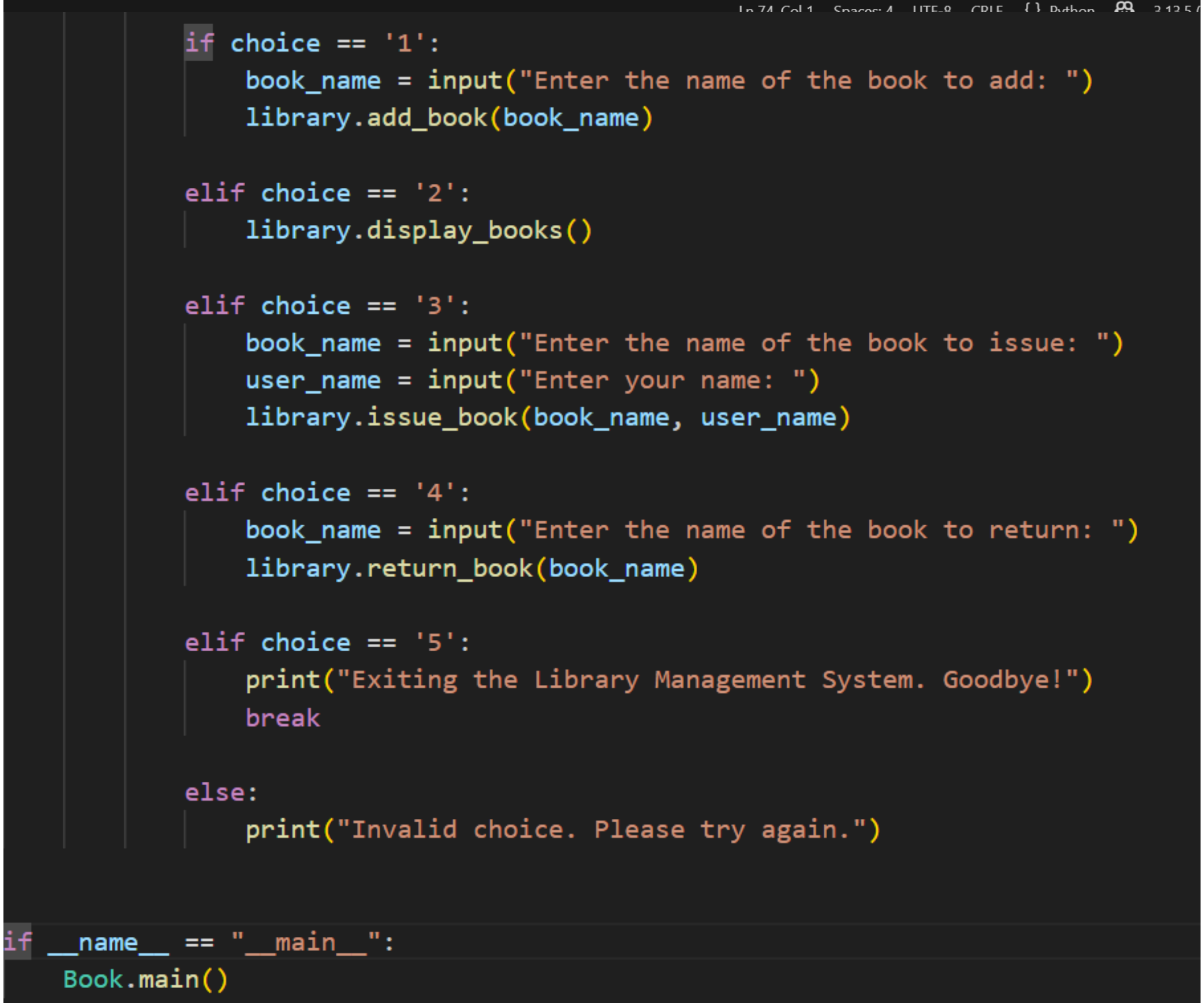
Code:



```
# “Generate a Python program for a library management system using classes, loops, and conditional statements.”

class Book:
    def __init__(self):
        self.books = []
        self.issued_books = {}   # corrected spelling

    def add_book(self, book_name):
        self.books.append(book_name)
        print(f'Book "{book_name}" added to the library.')

    def display_books(self):
        if not self.books:
            print("No books available in the library.")
        else:
            print("Available books in the library:")
            for book in self.books:
                print(f"- {book}")

    def issue_book(self, book_name, user_name):
        if book_name in self.books:
            self.books.remove(book_name)
            self.issued_books[book_name] = user_name
            print(f'Book "{book_name}" issued to {user_name}.')
        else:
            print(f'Sorry, the book "{book_name}" is not available.')

    def return_book(self, book_name):
        if book_name in self.issued_books:
            self.books.append(book_name)
```

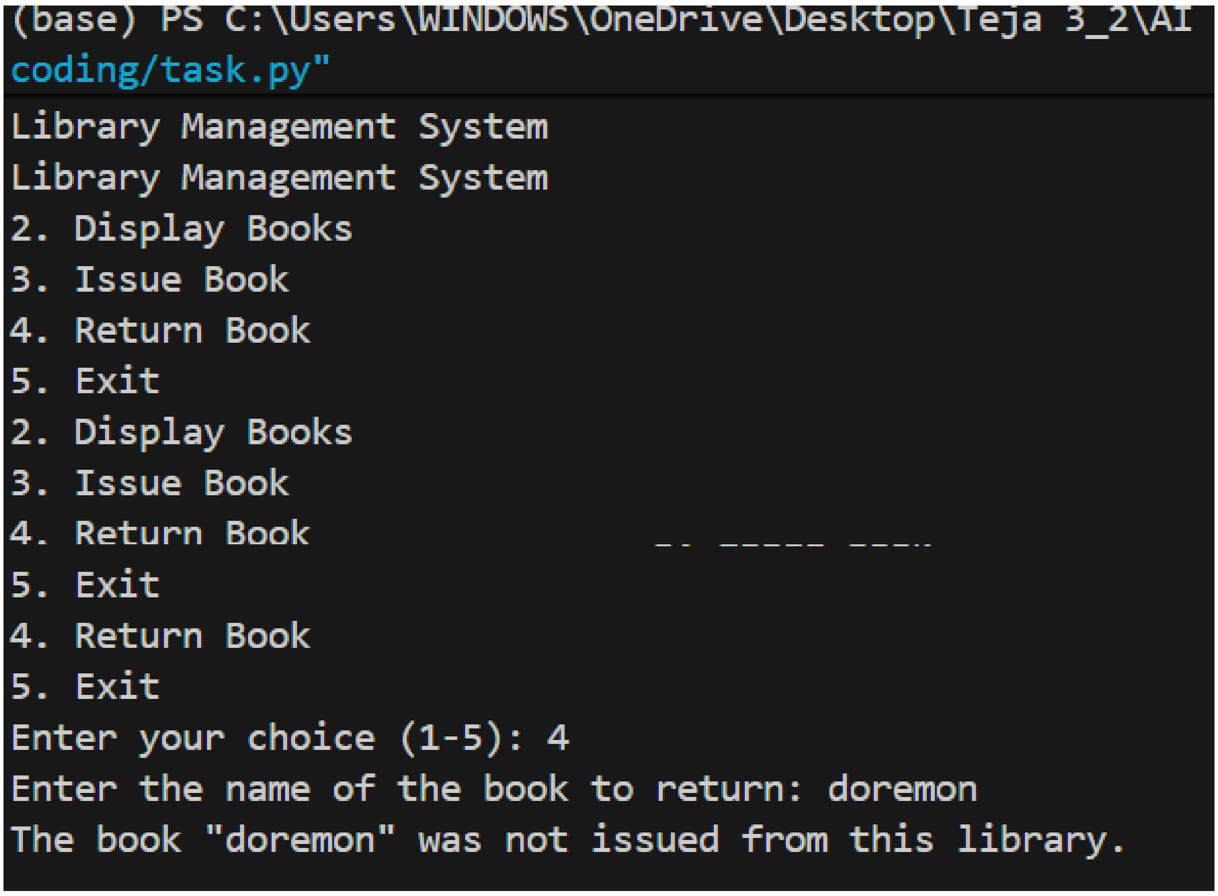


```python
class Book:
    def return_book(self, book_name):
        if book_name in self.issued_books:
            self.books.append(book_name)
            del self.issued_books[book_name]
            print(f'Book "{book_name}" returned to the library.')
        else:
            print(f'The book "{book_name}" was not issued from this library.')

    @staticmethod
    def main():
        library = Book()   # ✅ Corrected: created object properly

        while True:
            print("\nLibrary Management System")
            print("1. Add Book")
            print("2. Display Books")
            print("3. Issue Book")
            print("4. Return Book")
            print("5. Exit")

            choice = input("Enter your choice (1-5): ")

            if choice == '1':
                book_name = input("Enter the name of the book to add: ")
                library.add_book(book_name)

            elif choice == '2':
                library.display_books()
```



```python
            if choice == '1':
                book_name = input("Enter the name of the book to add: ")
                library.add_book(book_name)

            elif choice == '2':
                library.display_books()

            elif choice == '3':
                book_name = input("Enter the name of the book to issue: ")
                user_name = input("Enter your name: ")
                library.issue_book(book_name, user_name)

            elif choice == '4':
                book_name = input("Enter the name of the book to return: ")
                library.return_book(book_name)

            elif choice == '5':
                print("Exiting the Library Management System. Goodbye!")
                break

            else:
                print("Invalid choice. Please try again.")


if __name__ == "__main__":
    Book.main()
```

Output:



```
(base) PS C:\Users\WINDOWS\OneDrive\Desktop\Teja 3_2\AI
coding/task.py"
Library Management System
Library Management System
2. Display Books
3. Issue Book
4. Return Book
5. Exit
2. Display Books
3. Issue Book
4. Return Book
5. Exit
4. Return Book
5. Exit
Enter your choice (1-5): 4
Enter the name of the book to return: doremon
The book "doremon" was not issued from this library.
```

Explanation:

TheAI-generated Pythonprogramwasusedtodesignasimplelibrarymanagementsystem using aclass-based approach along with loopsand conditional statementsfor menu-driven operations.Thecodewascarefullyreviewedlinebylinetounderstandhowobject-oriented concepts, such as classes, methods, and instance variables, manage book records and availability status. Logical conditions were examined to ensure correct handling of operationslikeadding,displaying,issuing,andreturningbooks,whilealsopreventinginvalid 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 ensuringaclearunderstandingoftheprogram'sfunctionalityratherthanrelyingontheAI output without evaluation.

-(AI-AssistedCodeCompletionforClass-BasedAttendanceSystem) Task:

Use an AI tool to generate an attendance management class.

Prompt:"GenerateaPythonclasstomarkanddisplaystudentattendanceusingloops." Expected

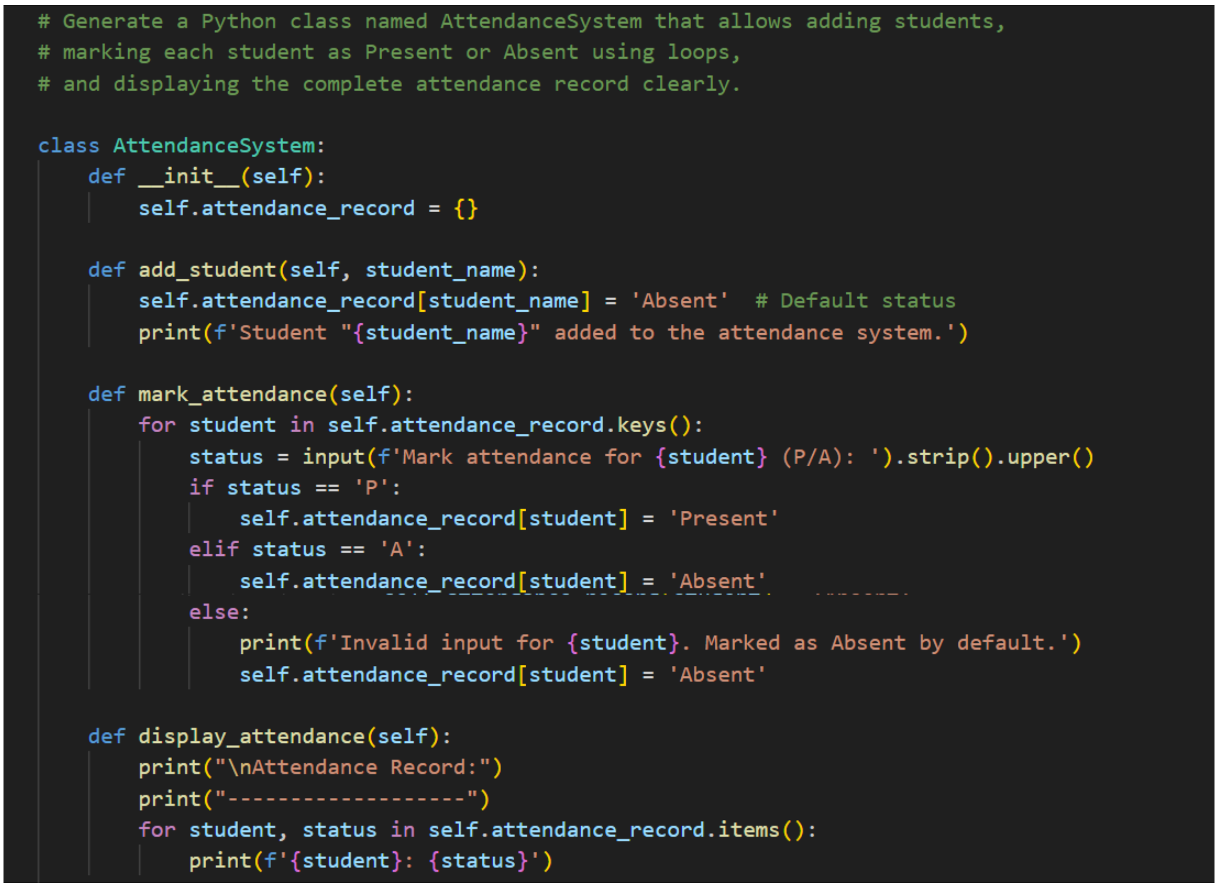
Output:

• AI-generatedattendancelogic.

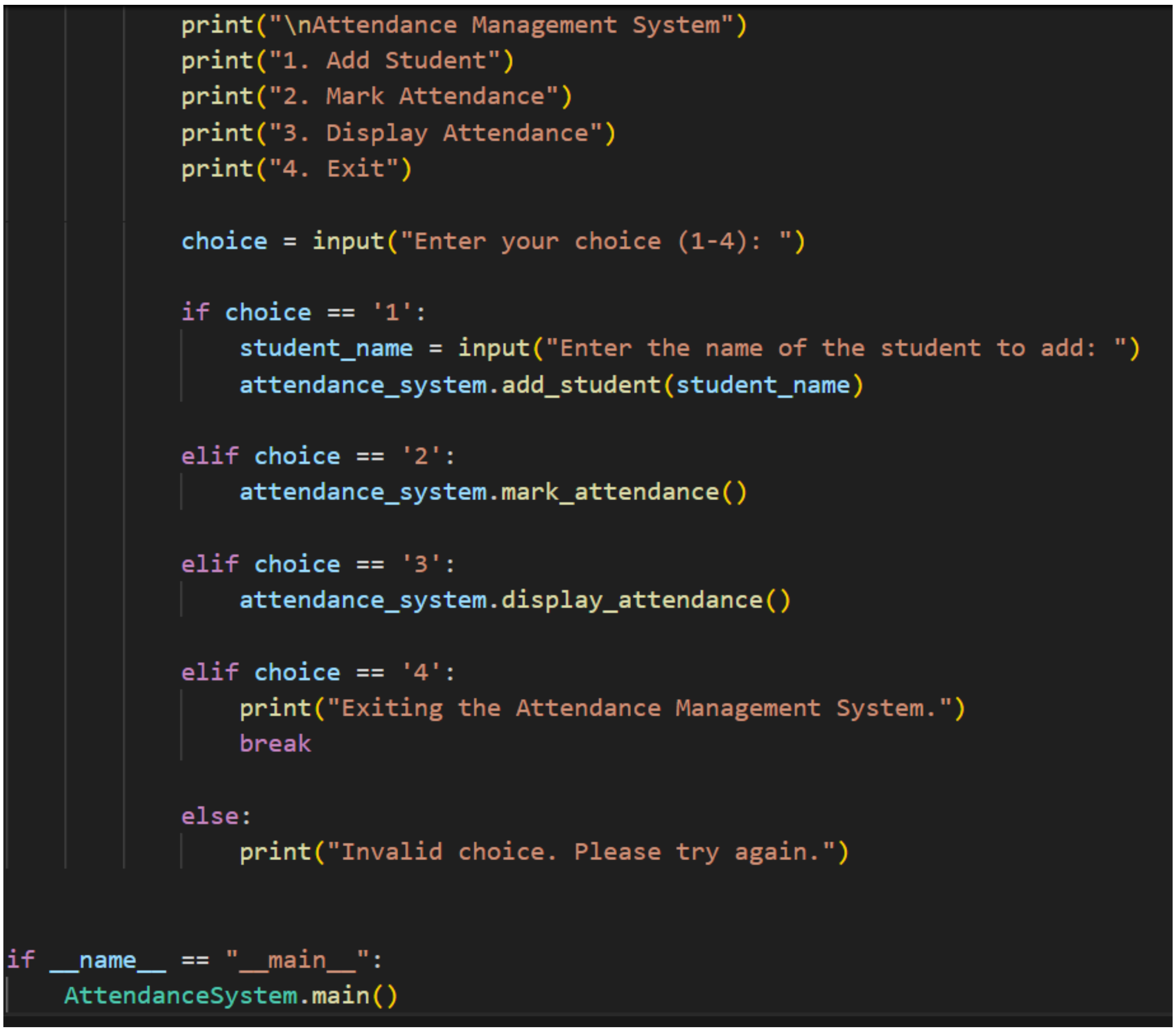
• Correctdisplayofattendance.

• Testcases

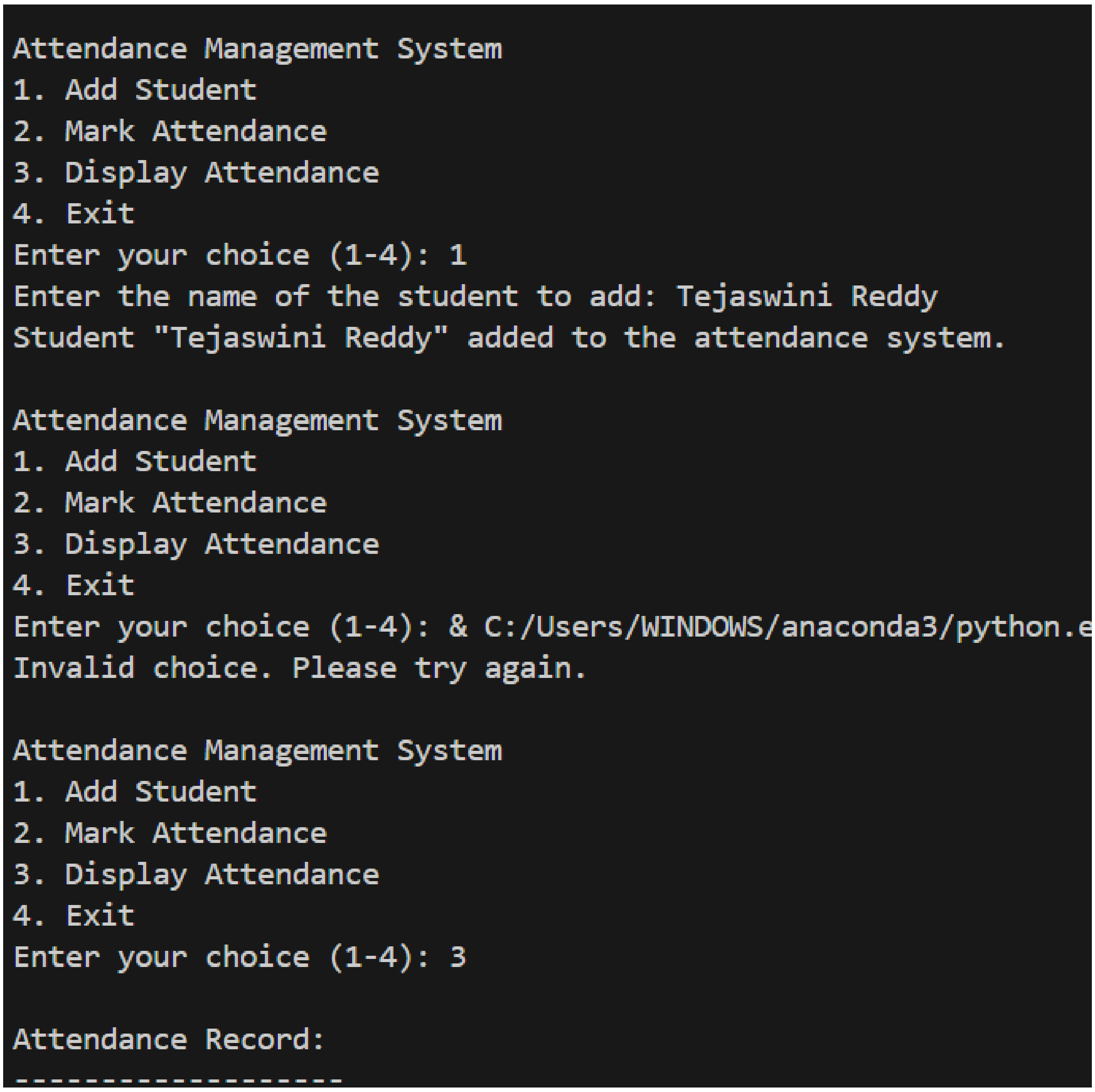
Code:



```python
# Generate a Python class named AttendanceSystem that allows adding students,
# marking each student as Present or Absent using loops,
# and displaying the complete attendance record clearly.

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

    def add_student(self, student_name):
        self.attendance_record[student_name] = 'Absent'  # Default status
        print(f'Student "{student_name}" added to the attendance system.')

    def mark_attendance(self):
        for student in self.attendance_record.keys():
            status = input(f'Mark attendance for {student} (P/A): ').strip().upper()
            if status == 'P':
                self.attendance_record[student] = 'Present'
            elif status == 'A':
                self.attendance_record[student] = 'Absent'
            else:
                print(f'Invalid input for {student}. Marked as Absent by default.')
                self.attendance_record[student] = 'Absent'

    def display_attendance(self):
        print("\nAttendance Record:")
        print("--------------------")
        for student, status in self.attendance_record.items():
            print(f'{student}: {status}')
```


```
            print("\nAttendance Management System")
            print("1. Add Student")
            print("2. Mark Attendance")
            print("3. Display Attendance")
            print("4. Exit")

            choice = input("Enter your choice (1-4): ")

            if choice == '1':
                student_name = input("Enter the name of the student to add: ")
                attendance_system.add_student(student_name)

            elif choice == '2':
                attendance_system.mark_attendance()

            elif choice == '3':
                attendance_system.display_attendance()

            elif choice == '4':
                print("Exiting the Attendance Management System.")
                break

            else:
                print("Invalid choice. Please try again.")


if __name__ == "__main__":
    AttendanceSystem.main()
```

Output:



```
Attendance Management System
1. Add Student
2. Mark Attendance
3. Display Attendance
4. Exit
Enter your choice (1-4): 1
Enter the name of the student to add: Tejaswini Reddy
Student "Tejaswini Reddy" added to the attendance system.

Attendance Management System
1. Add Student
2. Mark Attendance
3. Display Attendance
4. Exit
Enter your choice (1-4): & C:/Users/WINDOWS/anaconda3/python.e
Invalid choice. Please try again.

Attendance Management System
1. Add Student
2. Mark Attendance
3. Display Attendance
4. Exit
Enter your choice (1-4): 3

Attendance Record:
-------------------
```

Explanation:

TheAI-generatedPythonprogramwasusedtodevelopanattendancemanagementsystem using aclass-based structure along with loopsand conditional statementsfor 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 invalidinputssuchasemptynames,incorrectattendancestatus,andnon-existingstudents. The implementation was refined to improve readability and maintainability by using meaningful method names, clear comments, and structured control flow within the loop. ResponsibleuseofAItoolswasdemonstratedbyvalidatingthegeneratedlogic,handling edge cases correctly, and ensuring a clear understanding of the program's functionality rather than relying on the AI output without verification.

TaskDescription-5

-(AI-BasedCodeCompletionforConditionalMenuNavigation) Task:

Use an AI tool to complete a navigation menu.

Prompt:"GenerateaPythonprogramusingloopsandconditionalstosimulateanATM menu."

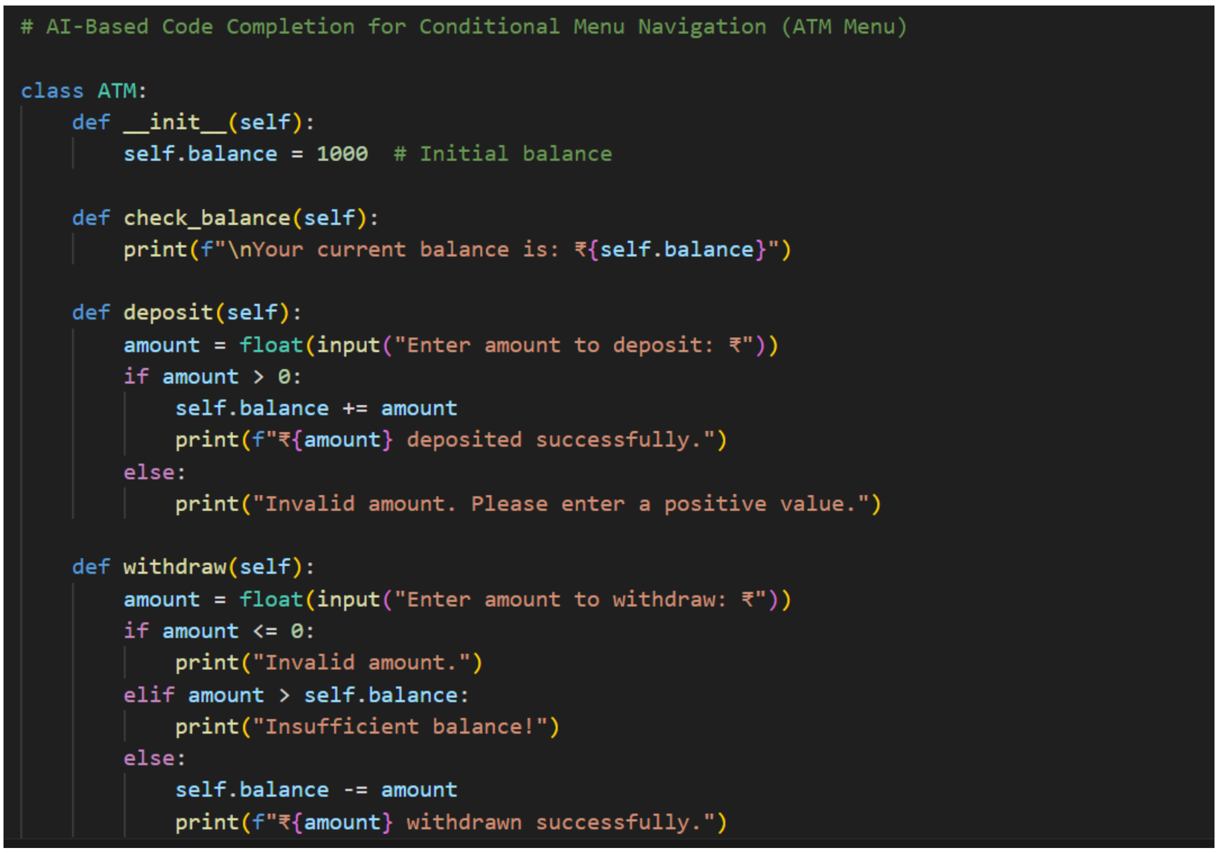
ExpectedOutput:

• AI-generatedmenulogic.

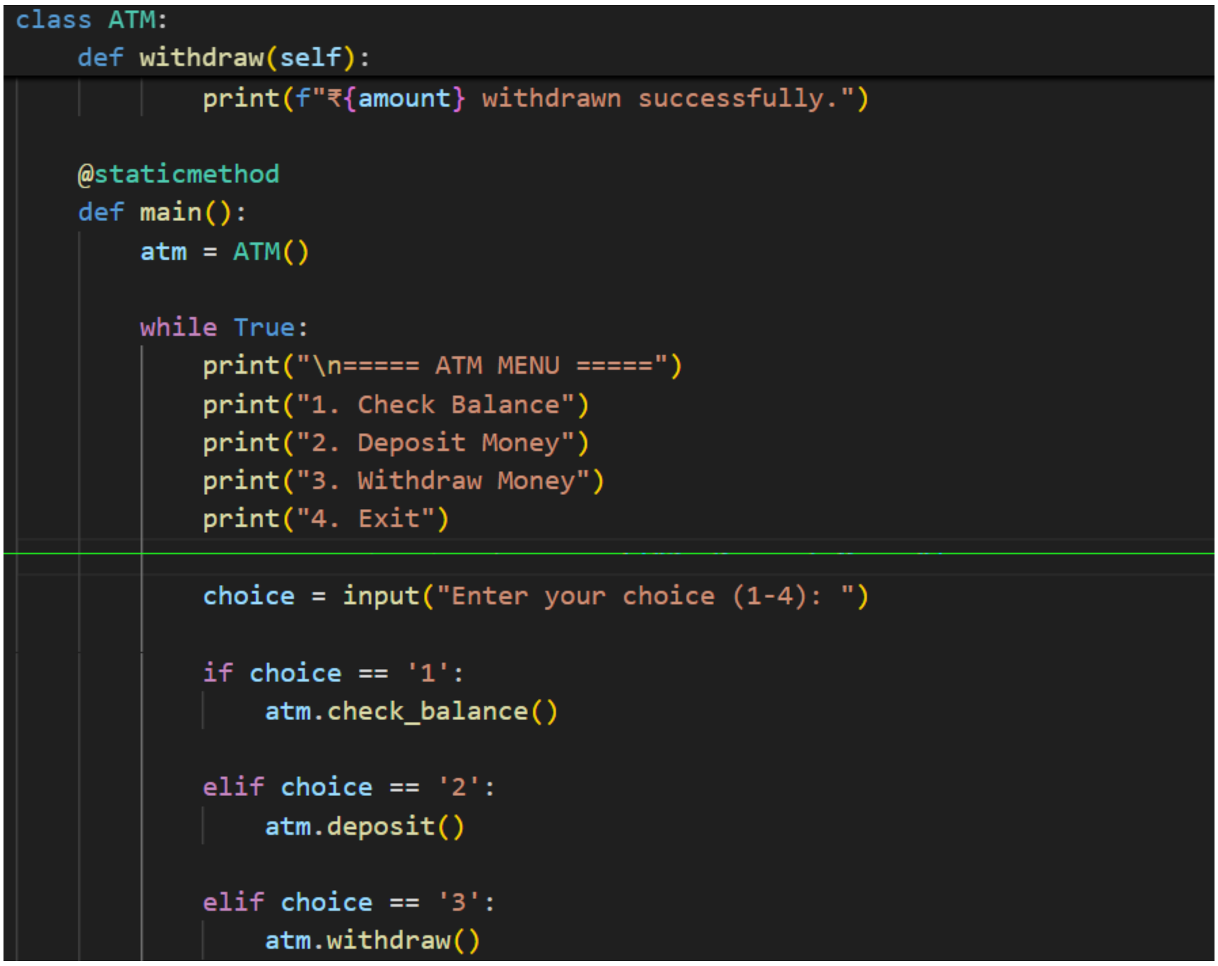
• Correctoptionhandling.

• Outputverification.

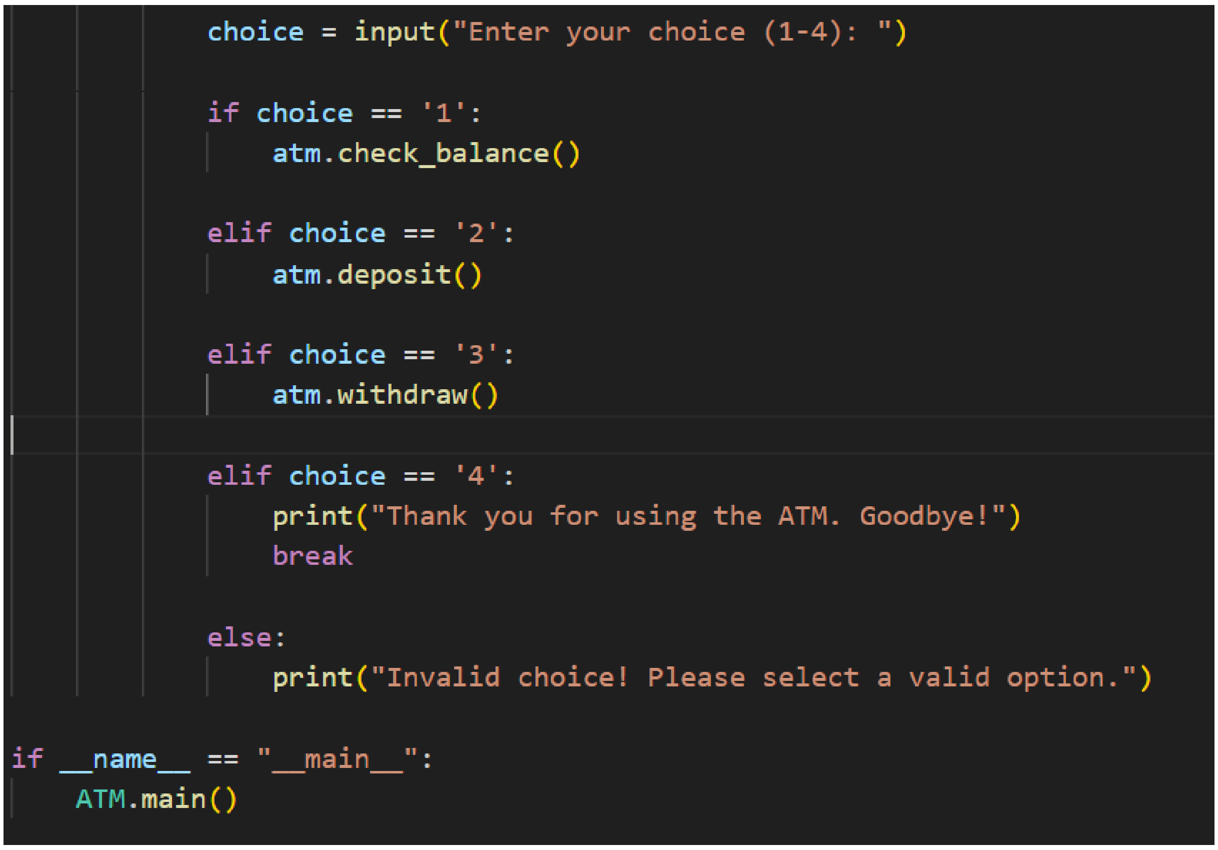
Code:



```python
# AI-Based Code Completion for Conditional Menu Navigation (ATM Menu)

class ATM:
    def __init__(self):
        self.balance = 1000  # Initial balance

    def check_balance(self):
        print(f"\nYour current balance is: ₹{self.balance}")

    def deposit(self):
        amount = float(input("Enter amount to deposit: ₹"))
        if amount > 0:
            self.balance += amount
            print(f"₹{amount} deposited successfully.")
        else:
            print("Invalid amount. Please enter a positive value.")

    def withdraw(self):
        amount = float(input("Enter amount to withdraw: ₹"))
        if amount <= 0:
            print("Invalid amount.")
        elif amount > self.balance:
            print("Insufficient balance!")
        else:
            self.balance -= amount
            print(f"₹{amount} withdrawn successfully.")
```

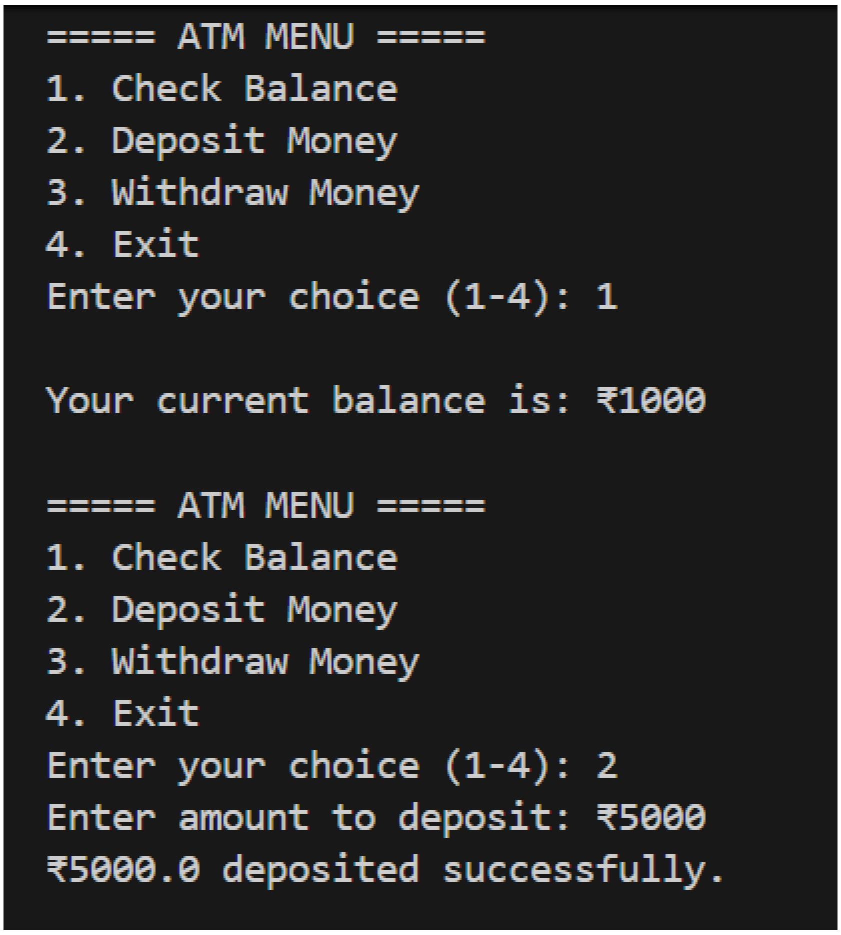


```python
class ATM:
    def withdraw(self):
            print(f"₹{amount} withdrawn successfully.")

    @staticmethod
    def main():
        atm = ATM()

        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':
                atm.deposit()

            elif choice == '3':
                atm.withdraw()
```



```python
            choice = input("Enter your choice (1-4): ")

            if choice == '1':
                atm.check_balance()

            elif choice == '2':
                atm.deposit()

            elif choice == '3':
                atm.withdraw()

            elif choice == '4':
                print("Thank you for using the ATM. Goodbye!")
                break

            else:
                print("Invalid choice! Please select a valid option.")

if __name__ == "__main__":
    ATM.main()
```

Output:



```
===== ATM MENU =====
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 1

Your current balance is: ₹1000

===== ATM MENU =====
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 2
Enter amount to deposit: ₹5000
₹5000.0 deposited successfully.
```

Explanation:

The AI-generated Python program was used to simulate an ATM system using loops and conditionalstatementstoprovideamenu-driveninterface.Thecodewascarefullyexamined 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 structurewasreviewedtoidentifyandpreventlogicalerrors,whileclearcommentsand meaningful variable names were used to improve readability and maintainability. ResponsibleuseofAItoolswasdemonstratedbyverifyingthegeneratedlogic,testing differenttransactionscenarios,andensuringcorrectbehaviourratherthanrelyingontheAI output without evaluation.