AI ASSISTANT CODING

ASSIGNMENT – 6.5

NAME : Yashaswini Nagireddi

HT.NO : 2303A51418

BATCH : 21

TASK 1- (AI-Based Code Completion for Conditional Eligibility Check) Task: Use an AI tool to generate eligibility logic.

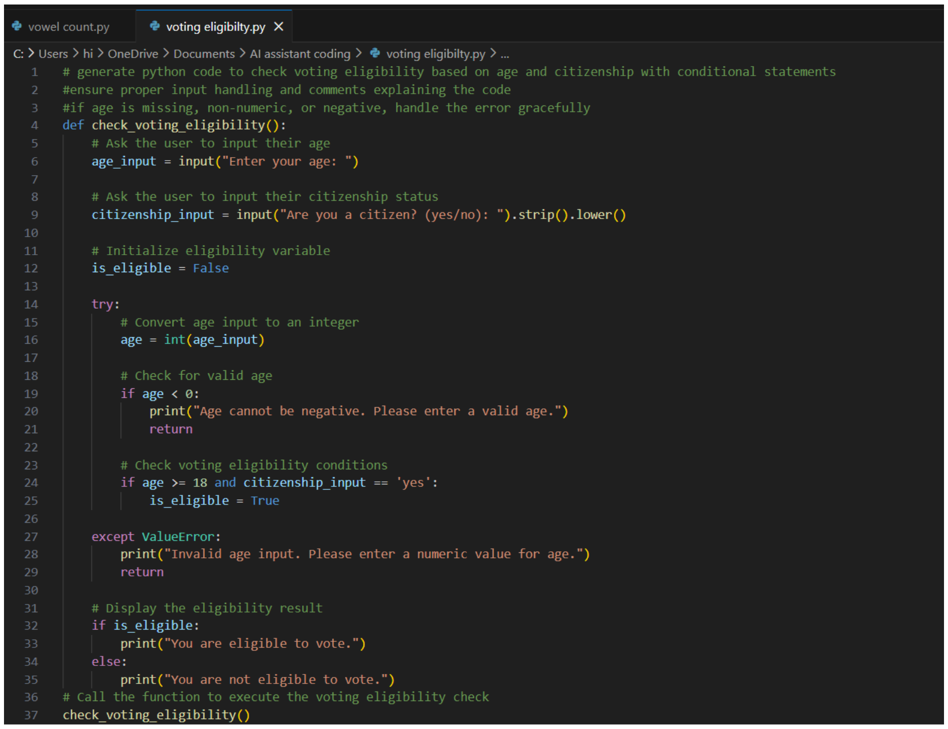
Prompt: “Generate Python code to check voting eligibility based on age and citizenship.” Expected Output:

• AI-generated conditional logic.

• Correct eligibility decisions.

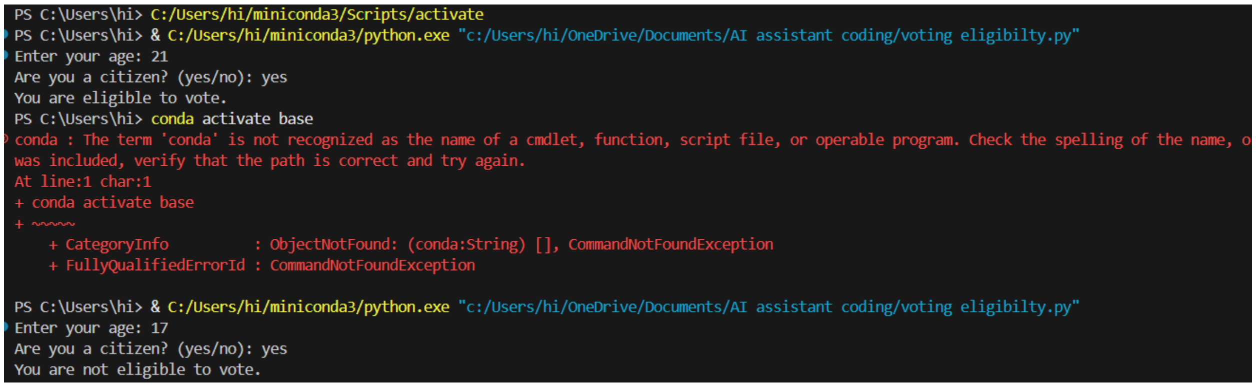
• Explanation of conditions.

CODE :


vowel count.py
voting eligibilty.py ×
C: > Users > hi > OneDrive > Documents > AI assistant coding > voting eligibilty.py > ...

```python
# generate python code to check voting eligibility based on age and citizenship with conditional statements
#ensure proper input handling and comments explaining the code
#if age is missing, non-numeric, or negative, handle the error gracefully
def check_voting_eligibility():
    # Ask the user to input their age
    age_input = input("Enter your age: ")

    # Ask the user to input their citizenship status
    citizenship_input = input("Are you a citizen? (yes/no): ").strip().lower()

    # Initialize eligibility variable
    is_eligible = False

    try:
        # Convert age input to an integer
        age = int(age_input)

        # Check for valid age
        if age < 0:
            print("Age cannot be negative. Please enter a valid age.")
            return

        # Check voting eligibility conditions
        if age >= 18 and citizenship_input == 'yes':
            is_eligible = True

    except ValueError:
        print("Invalid age input. Please enter a numeric value for age.")
        return

    # Display the eligibility result
    if is_eligible:
        print("You are eligible to vote.")
    else:
        print("You are not eligible to vote.")
# Call the function to execute the voting eligibility check
check_voting_eligibility()
```

OUTPUT :



```
PS C:\Users\hi> C:/Users/hi/miniconda3/Scripts/activate
PS C:\Users\hi> & C:/Users/hi/miniconda3/python.exe "c:/Users/hi/OneDrive/Documents/AI assistant coding/voting eligibilty.py"
Enter your age: 21
Are you a citizen? (yes/no): yes
You are eligible to vote.
PS C:\Users\hi> conda activate base
conda : The term 'conda' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, o
was included, verify that the path is correct and try again.
At line:1 char:1
+ conda activate base
+ ~~~~~
    + CategoryInfo          : ObjectNotFound: (conda:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\hi> & C:/Users/hi/miniconda3/python.exe "c:/Users/hi/OneDrive/Documents/AI assistant coding/voting eligibilty.py"
Enter your age: 17
Are you a citizen? (yes/no): yes
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.

TASK2 - (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.”

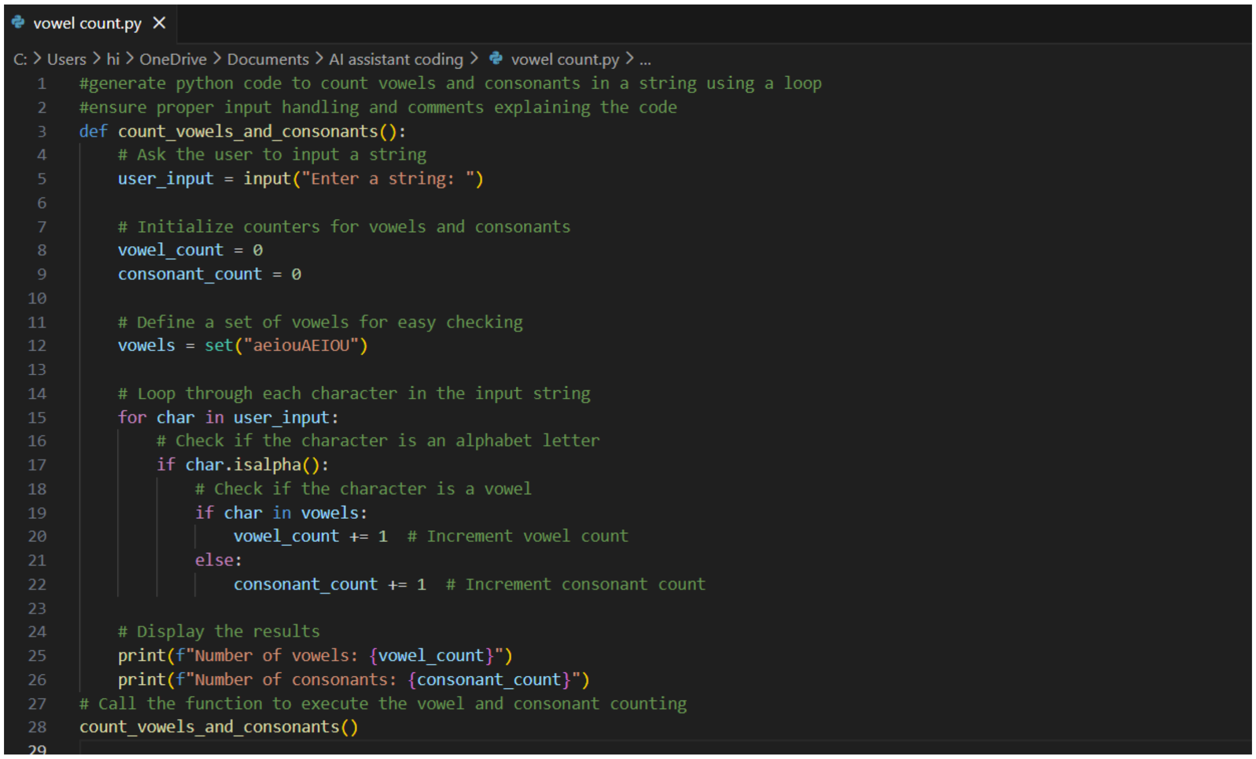
Expected Output:

• AI-generated string processing logic.

• Correct counts.

• Output verification.

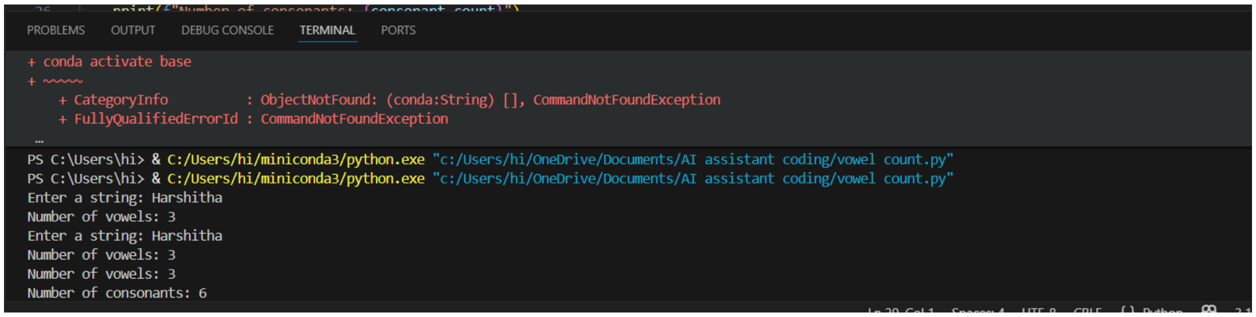
CODE :



vowel count.py
C: > Users > hi > OneDrive > Documents > AI assistant coding > vowel count.py > ...

```python
#generate python code to count vowels and consonants in a string using a loop
#ensure proper input handling and comments explaining the code
def count_vowels_and_consonants():
    # Ask the user to input a string
    user_input = input("Enter a string: ")

    # Initialize counters for vowels and consonants
    vowel_count = 0
    consonant_count = 0

    # Define a set of vowels for easy checking
    vowels = set("aeiouAEIOU")

    # Loop through each character in the input string
    for char in user_input:
        # Check if the character is an alphabet letter
        if char.isalpha():
            # Check if the character is a vowel
            if char in vowels:
                vowel_count += 1  # Increment vowel count
            else:
                consonant_count += 1  # Increment consonant count

    # Display the results
    print(f"Number of vowels: {vowel_count}")
    print(f"Number of consonants: {consonant_count}")
# Call the function to execute the vowel and consonant counting
count_vowels_and_consonants()
```

OUTPUT :



```
+ conda activate base
+ ~~~~~
    + CategoryInfo          : ObjectNotFound: (conda:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
 
PS C:\Users\hi> & C:/Users/hi/miniconda3/python.exe "c:/Users/hi/OneDrive/Documents/AI assistant coding/vowel count.py"
PS C:\Users\hi> & C:/Users/hi/miniconda3/python.exe "c:/Users/hi/OneDrive/Documents/AI assistant coding/vowel count.py"
Enter a string: Harshitha
Number of vowels: 3
Enter a string: Harshitha
Number of vowels: 3
Number of vowels: 3
Number of consonants: 6
```

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. 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.

TASK 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.”

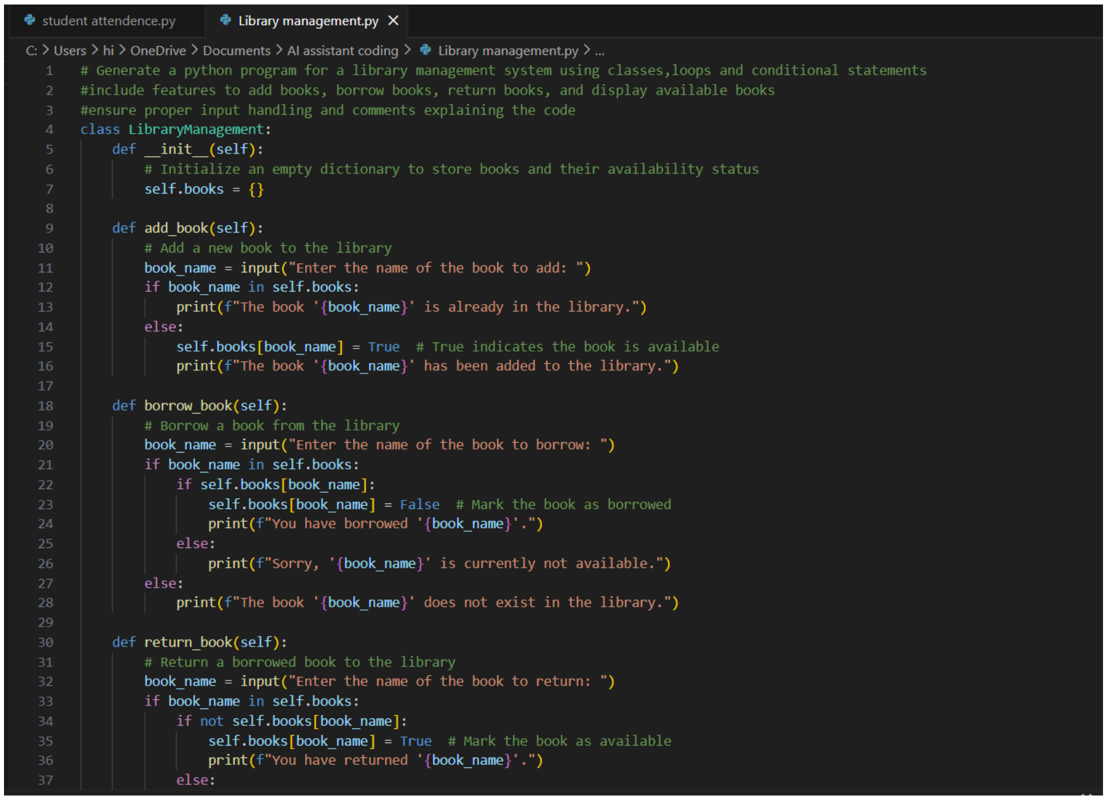
Expected Output:

• Complete AI-generated program.

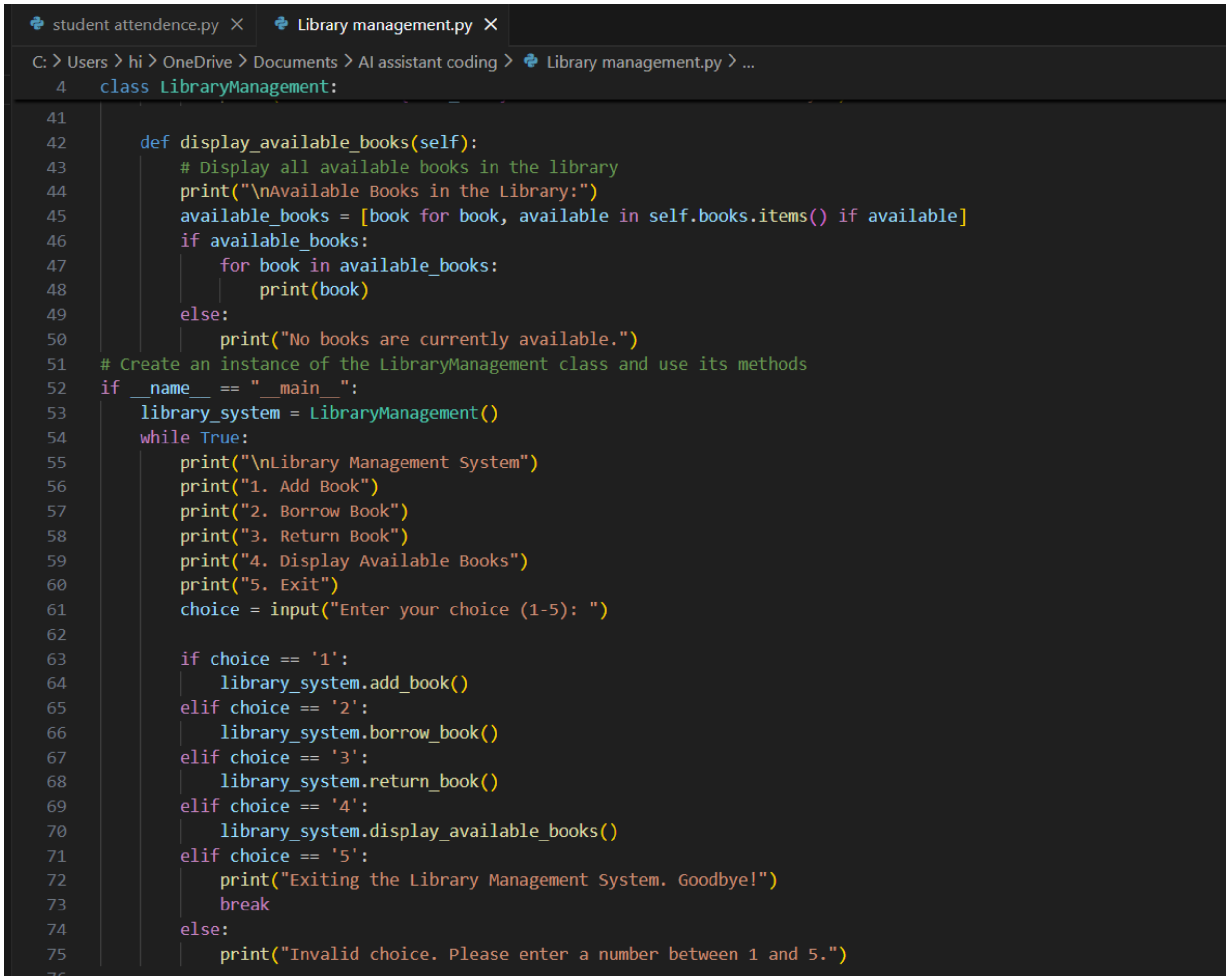
• Review of AI suggestions quality.

• Short reflection on AI-assisted coding experience.

CODE :

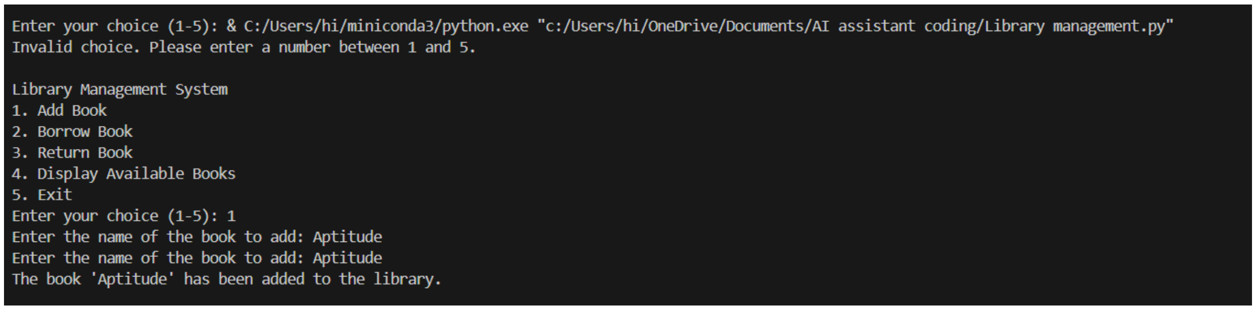

student attendence.py
Library management.py
C: > Users > hi > OneDrive > Documents > AI assistant coding > Library management.py > ...

```python
# Generate a python program for a library management system using classes,loops and conditional statements
#include features to add books, borrow books, return books, and display available books
#ensure proper input handling and comments explaining the code
class LibraryManagement:
    def __init__(self):
        # Initialize an empty dictionary to store books and their availability status
        self.books = {}

    def add_book(self):
        # Add a new book to the library
        book_name = input("Enter the name of the book to add: ")
        if book_name in self.books:
            print(f"The book '{book_name}' is already in the library.")
        else:
            self.books[book_name] = True  # True indicates the book is available
            print(f"The book '{book_name}' has been added to the library.")

    def borrow_book(self):
        # Borrow a book from the library
        book_name = input("Enter the name of the book to borrow: ")
        if book_name in self.books:
            if self.books[book_name]:
                self.books[book_name] = False  # Mark the book as borrowed
                print(f"You have borrowed '{book_name}'.")
            else:
                print(f"Sorry, '{book_name}' is currently not available.")
        else:
            print(f"The book '{book_name}' does not exist in the library.")

    def return_book(self):
        # Return a borrowed book to the library
        book_name = input("Enter the name of the book to return: ")
        if book_name in self.books:
            if not self.books[book_name]:
                self.books[book_name] = True  # Mark the book as available
                print(f"You have returned '{book_name}'.")
            else:
```



```
student attendence.py    Library management.py
C: > Users > hi > OneDrive > Documents > AI assistant coding > Library management.py > ...
class LibraryManagement:

    def display_available_books(self):
        # Display all available books in the library
        print("\nAvailable Books in the Library:")
        available_books = [book for book, available in self.books.items() if available]
        if available_books:
            for book in available_books:
                print(book)
        else:
            print("No books are currently available.")
# Create an instance of the LibraryManagement class and use its methods
if __name__ == "__main__":
    library_system = LibraryManagement()
    while True:
        print("\nLibrary Management System")
        print("1. Add Book")
        print("2. Borrow Book")
        print("3. Return Book")
        print("4. Display Available Books")
        print("5. Exit")
        choice = input("Enter your choice (1-5): ")

        if choice == '1':
            library_system.add_book()
        elif choice == '2':
            library_system.borrow_book()
        elif choice == '3':
            library_system.return_book()
        elif choice == '4':
            library_system.display_available_books()
        elif choice == '5':
            print("Exiting the Library Management System. Goodbye!")
            break
        else:
            print("Invalid choice. Please enter a number between 1 and 5.")
```

OUTPUT :



```
Enter your choice (1-5): & C:/Users/hi/miniconda3/python.exe "c:/Users/hi/OneDrive/Documents/AI assistant coding/Library management.py"
Invalid choice. Please enter a number between 1 and 5.

Library Management System
1. Add Book
2. Borrow Book
3. Return Book
4. Display Available Books
5. Exit
Enter your choice (1-5): 1
Enter the name of the book to add: Aptitude
Enter the name of the book to add: Aptitude
The book 'Aptitude' has been added to the library.
```

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

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

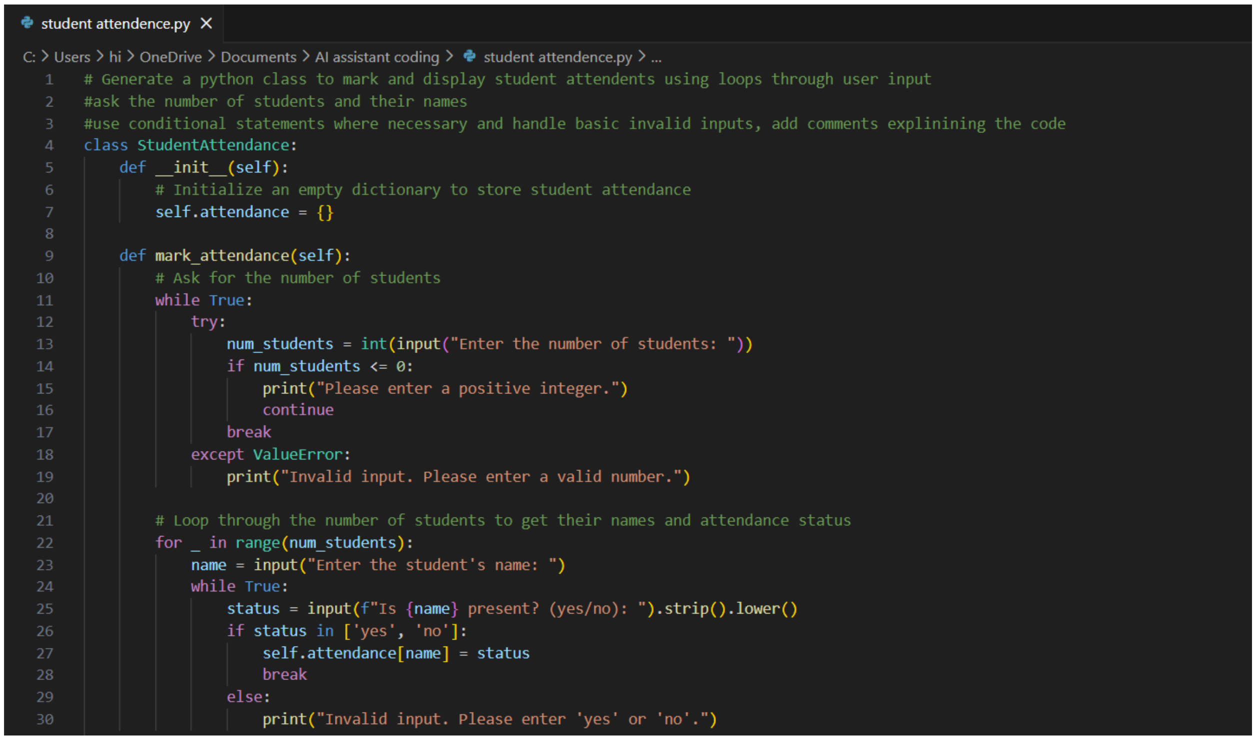
Prompt: "Generate a Python class to mark and display student attendance using loops."
Expected Output:

• AI-generated attendance logic.

• Correct display of attendance.

• Test cases.

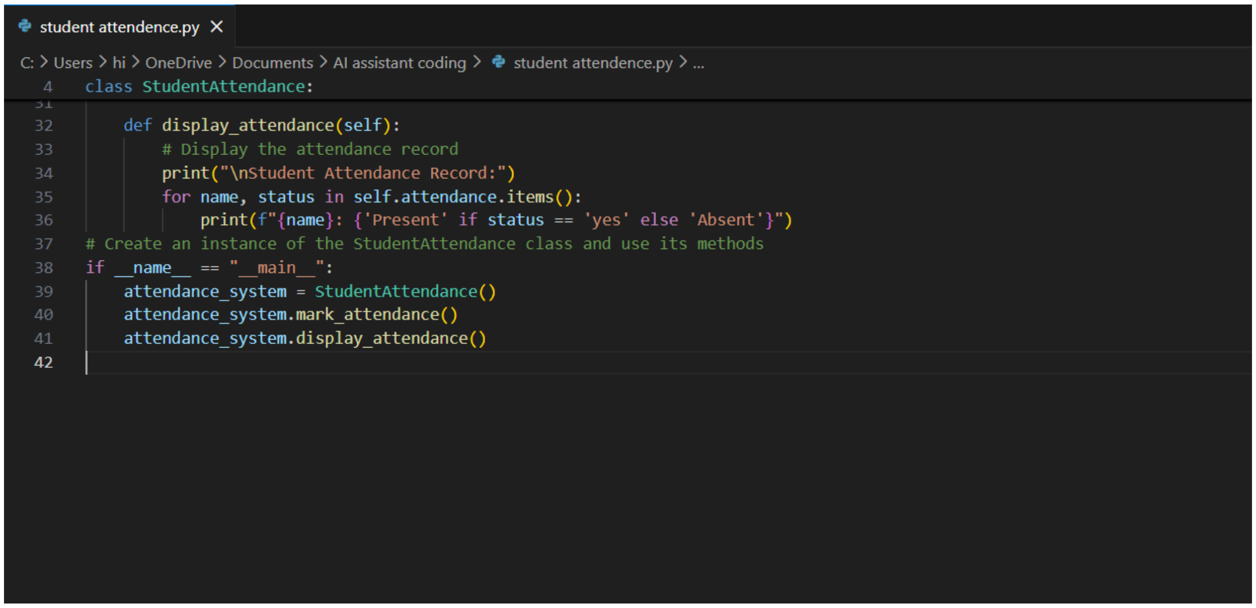
CODE :



student attendence.py

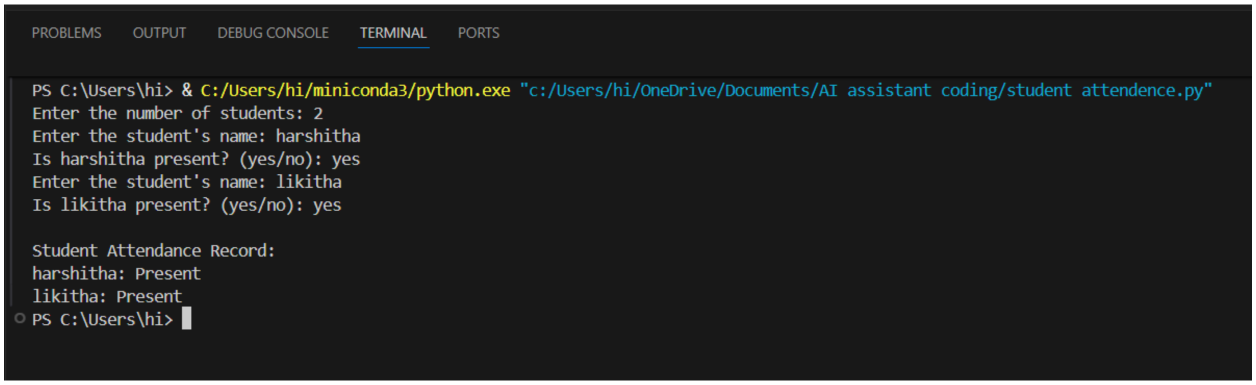
C: > Users > hi > OneDrive > Documents > AI assistant coding > student attendence.py > ...

```python
# Generate a python class to mark and display student attendents using loops through user input
#ask the number of students and their names
#use conditional statements where necessary and handle basic invalid inputs, add comments explinining the code
class StudentAttendance:
    def __init__(self):
        # Initialize an empty dictionary to store student attendance
        self.attendance = {}

    def mark_attendance(self):
        # Ask for the number of students
        while True:
            try:
                num_students = int(input("Enter the number of students: "))
                if num_students <= 0:
                    print("Please enter a positive integer.")
                    continue
                break
            except ValueError:
                print("Invalid input. Please enter a valid number.")

        # Loop through the number of students to get their names and attendance status
        for _ in range(num_students):
            name = input("Enter the student's name: ")
            while True:
                status = input(f"Is {name} present? (yes/no): ").strip().lower()
                if status in ['yes', 'no']:
                    self.attendance[name] = status
                    break
                else:
                    print("Invalid input. Please enter 'yes' or 'no'.")
```



```
student attendence.py ×
C: > Users > hi > OneDrive > Documents > AI assistant coding > student attendence.py > ...
class StudentAttendance:

    def display_attendance(self):
        # Display the attendance record
        print("\nStudent Attendance Record:")
        for name, status in self.attendance.items():
            print(f"{name}: {'Present' if status == 'yes' else 'Absent'}")
# Create an instance of the StudentAttendance class and use its methods
if __name__ == "__main__":
    attendance_system = StudentAttendance()
    attendance_system.mark_attendance()
    attendance_system.display_attendance()
```

OUTPUT :



```
PS C:\Users\hi> & C:/Users/hi/miniconda3/python.exe "c:/Users/hi/OneDrive/Documents/AI assistant coding/student attendence.py"
Enter the number of students: 2
Enter the student's name: harshitha
Is harshitha present? (yes/no): yes
Enter the student's name: likitha
Is likitha present? (yes/no): yes

Student Attendance Record:
harshitha: Present
likitha: Present
PS C:\Users\hi>
```

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 5- (AI-Based Code Completion for Conditional Menu Navigation) Task: Use an AI tool to complete a navigation menu.

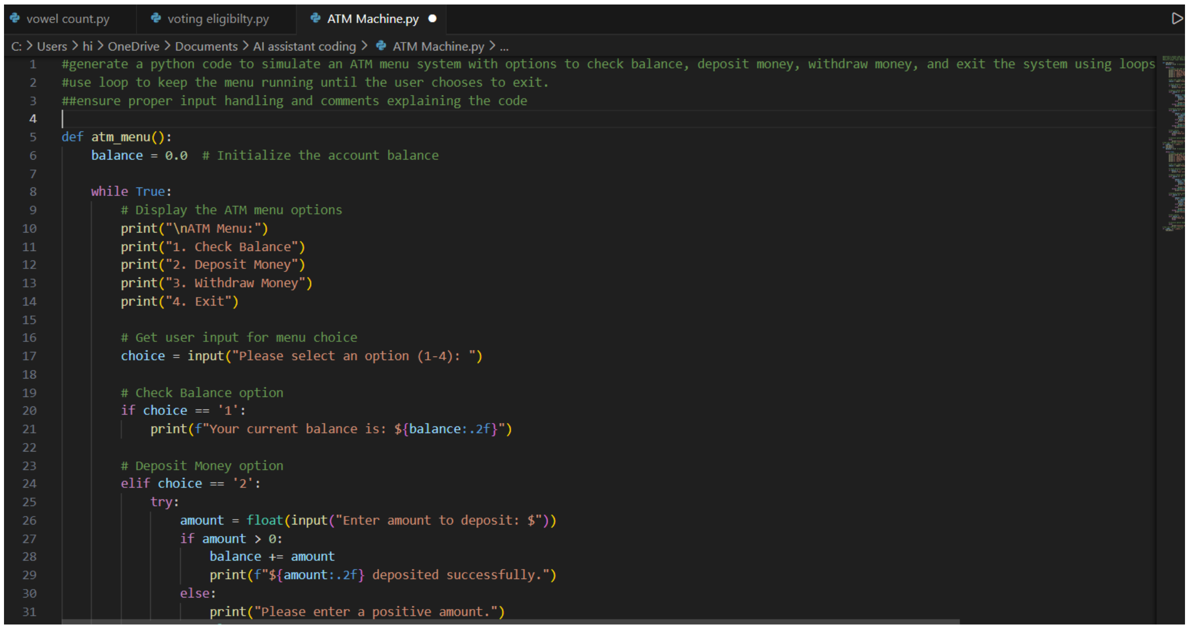
Prompt: “Generate a Python program using loops and conditionals to simulate an ATM menu.” Expected Output:

• AI-generated menu logic.

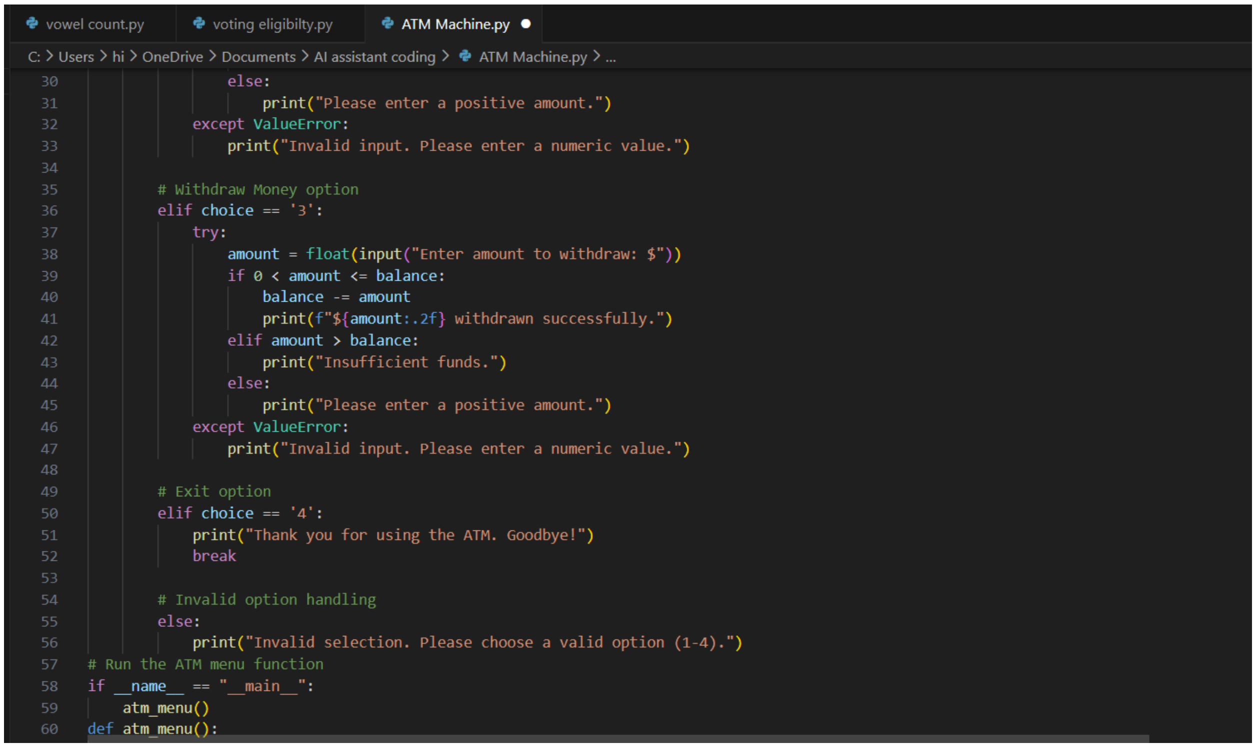
• Correct option handling.

• Output verification.

CODE :


vowel count.py
voting eligibilty.py
ATM Machine.py
C: > Users > hi > OneDrive > Documents > AI assistant coding > ATM Machine.py > ...

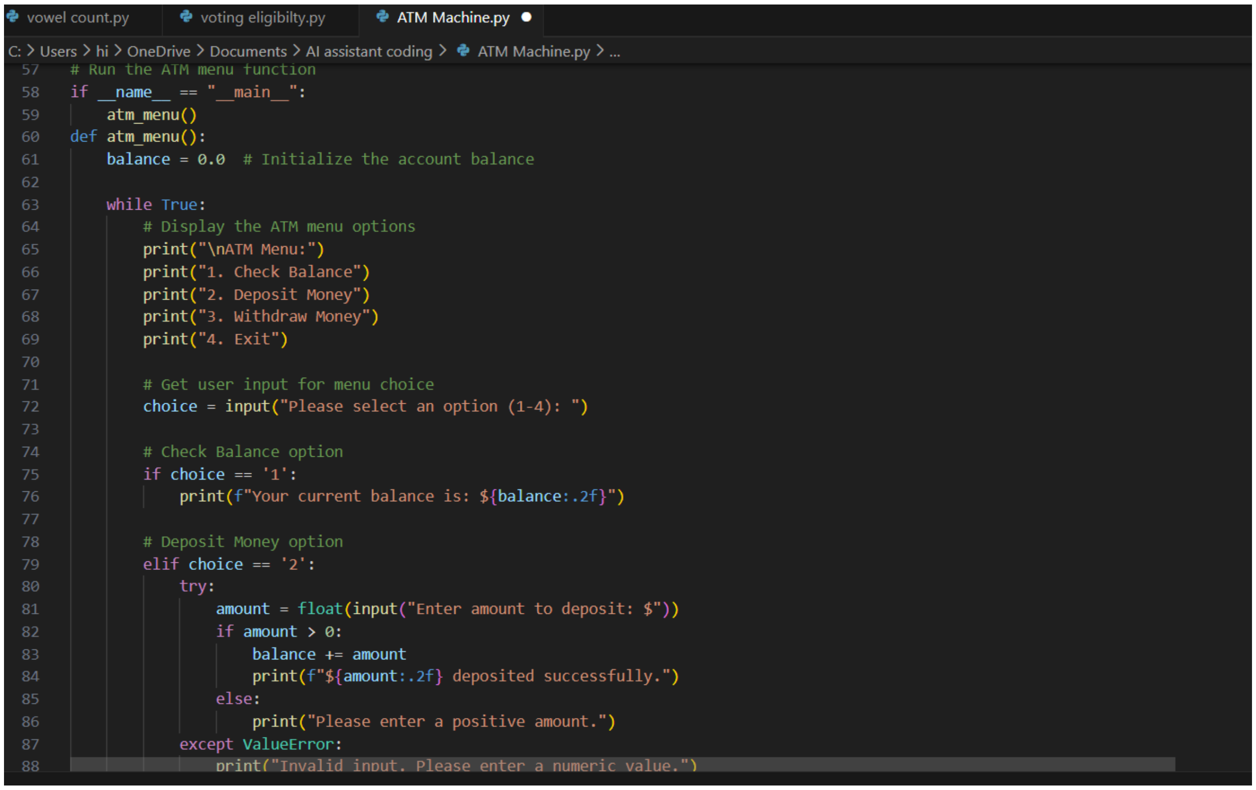
```python
#generate a python code to simulate an ATM menu system with options to check balance, deposit money, withdraw money, and exit the system using loops
#use loop to keep the menu running until the user chooses to exit.
##ensure proper input handling and comments explaining the code

def atm_menu():
    balance = 0.0  # Initialize the account balance

    while True:
        # Display the ATM menu options
        print("\nATM Menu:")
        print("1. Check Balance")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Exit")

        # Get user input for menu choice
        choice = input("Please select an option (1-4): ")

        # Check Balance option
        if choice == '1':
            print(f"Your current balance is: ${balance:.2f}")

        # Deposit Money option
        elif choice == '2':
            try:
                amount = float(input("Enter amount to deposit: $"))
                if amount > 0:
                    balance += amount
                    print(f"${amount:.2f} deposited successfully.")
                else:
                    print("Please enter a positive amount.")
```



vowel count.py | voting eligibilty.py | ATM Machine.py

C: > Users > hi > OneDrive > Documents > AI assistant coding > ATM Machine.py > ...

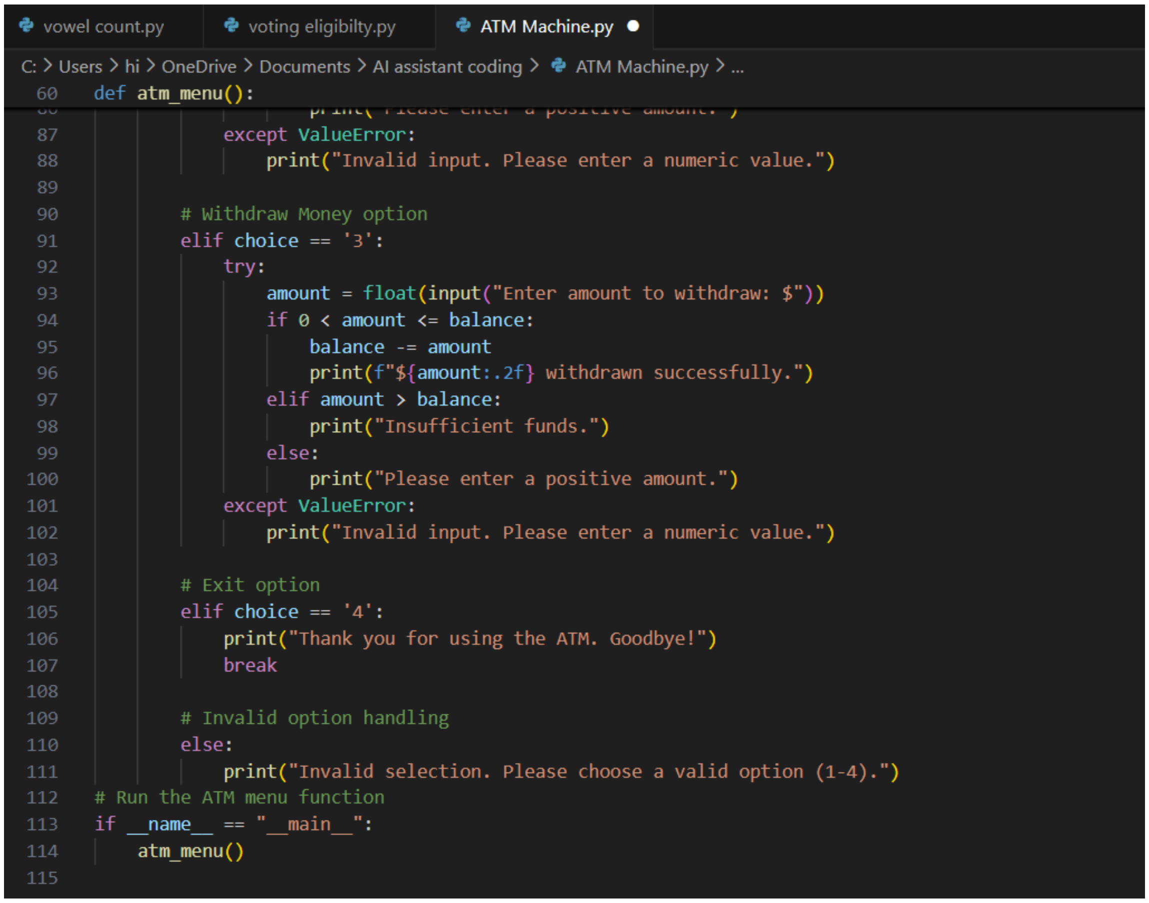
```python
                else:
                    print("Please enter a positive amount.")
            except ValueError:
                print("Invalid input. Please enter a numeric value.")

        # Withdraw Money option
        elif choice == '3':
            try:
                amount = float(input("Enter amount to withdraw: $"))
                if 0 < amount <= balance:
                    balance -= amount
                    print(f"${amount:.2f} withdrawn successfully.")
                elif amount > balance:
                    print("Insufficient funds.")
                else:
                    print("Please enter a positive amount.")
            except ValueError:
                print("Invalid input. Please enter a numeric value.")

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

        # Invalid option handling
        else:
            print("Invalid selection. Please choose a valid option (1-4).")
# Run the ATM menu function
if __name__ == "__main__":
    atm_menu()
def atm_menu():
```



vowel count.py | voting eligibilty.py | ATM Machine.py

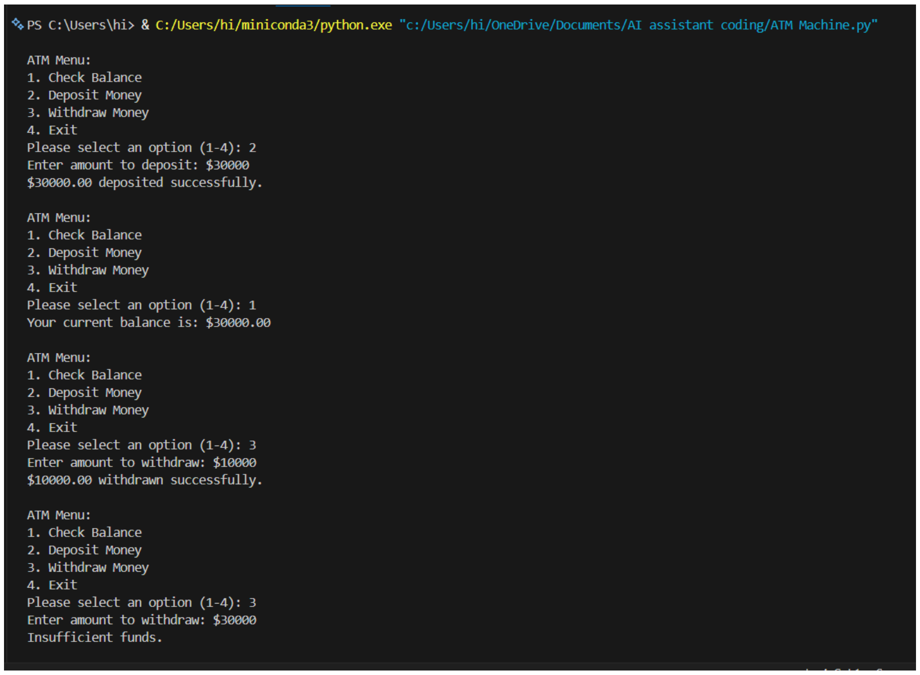
C: > Users > hi > OneDrive > Documents > AI assistant coding > ATM Machine.py > ...

```python
# Run the ATM menu function
if __name__ == "__main__":
    atm_menu()
def atm_menu():
    balance = 0.0  # Initialize the account balance

    while True:
        # Display the ATM menu options
        print("\nATM Menu:")
        print("1. Check Balance")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Exit")

        # Get user input for menu choice
        choice = input("Please select an option (1-4): ")

        # Check Balance option
        if choice == '1':
            print(f"Your current balance is: ${balance:.2f}")

        # Deposit Money option
        elif choice == '2':
            try:
                amount = float(input("Enter amount to deposit: $"))
                if amount > 0:
                    balance += amount
                    print(f"${amount:.2f} deposited successfully.")
                else:
                    print("Please enter a positive amount.")
            except ValueError:
                print("Invalid input. Please enter a numeric value.")
```


vowel count.py
voting eligibilty.py
ATM Machine.py
C: > Users > hi > OneDrive > Documents > AI assistant coding > ATM Machine.py > ...

```python
def atm_menu():
            except ValueError:
                print("Invalid input. Please enter a numeric value.")

        # Withdraw Money option
        elif choice == '3':
            try:
                amount = float(input("Enter amount to withdraw: $"))
                if 0 < amount <= balance:
                    balance -= amount
                    print(f"${amount:.2f} withdrawn successfully.")
                elif amount > balance:
                    print("Insufficient funds.")
                else:
                    print("Please enter a positive amount.")
            except ValueError:
                print("Invalid input. Please enter a numeric value.")

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

        # Invalid option handling
        else:
            print("Invalid selection. Please choose a valid option (1-4).")
# Run the ATM menu function
if __name__ == "__main__":
    atm_menu()
```

OUTPUT :



```
PS C:\Users\hi> & C:/Users/hi/miniconda3/python.exe "c:/Users/hi/OneDrive/Documents/AI assistant coding/ATM Machine.py"

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Please select an option (1-4): 2
Enter amount to deposit: $30000
$30000.00 deposited successfully.

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Please select an option (1-4): 1
Your current balance is: $30000.00

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Please select an option (1-4): 3
Enter amount to withdraw: $10000
$10000.00 withdrawn successfully.

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Please select an option (1-4): 3
Enter amount to withdraw: $30000
Insufficient funds.
```

EXPLANATION :

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.