ASSIGNMENT 6.5

Atla Sreeja

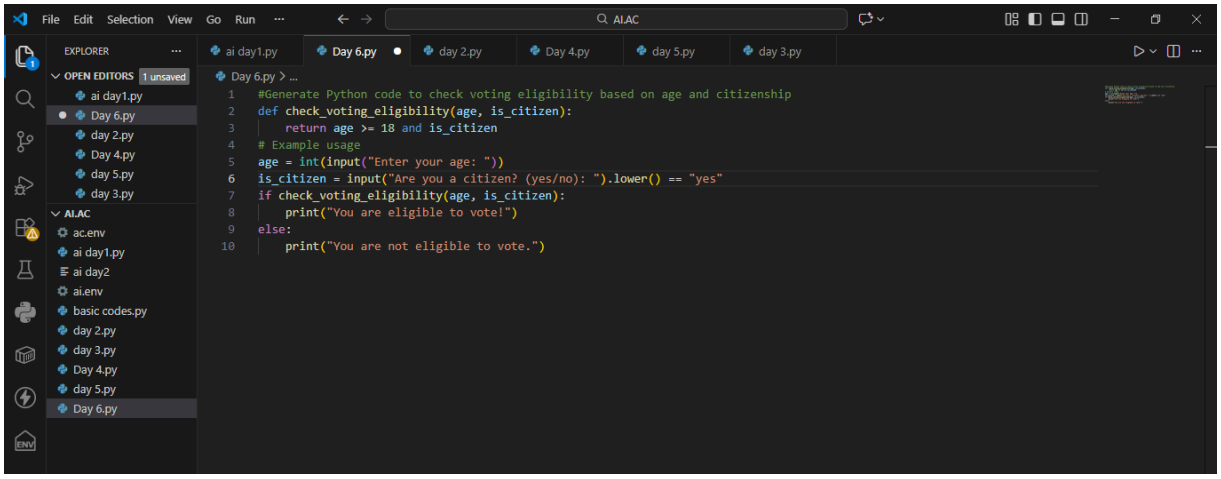
2403A51L02

B-51

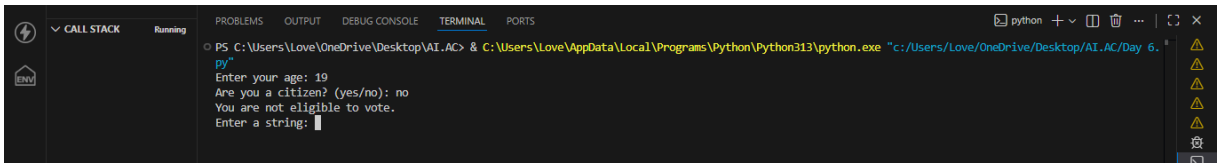
AI-Based Code Completion: Working with suggestions for classes, loops, conditionals Task Description-1: Zero-shot Prompting

Task 1: Conditional Eligibility Check

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



OUTPUT:



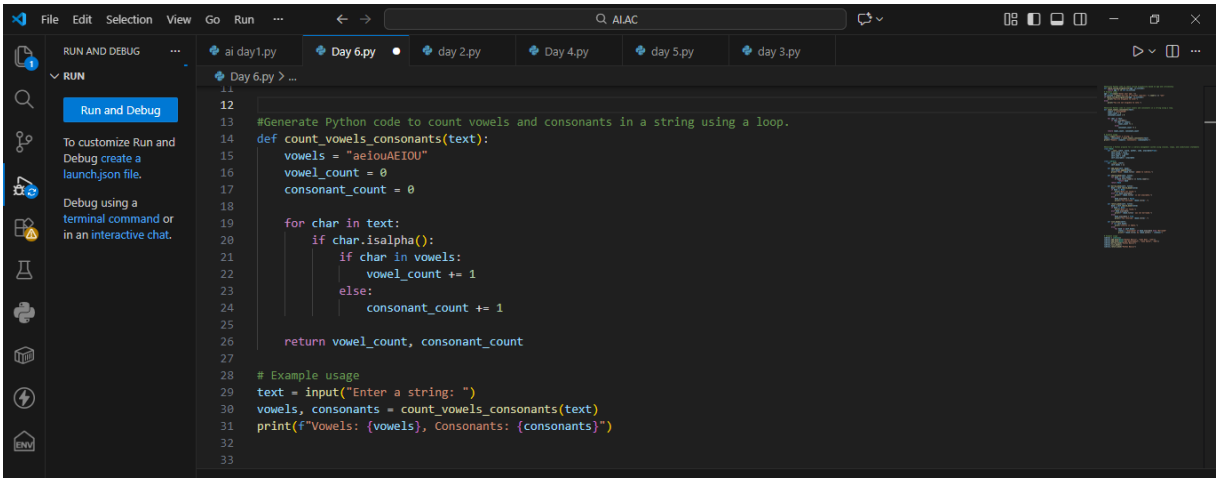
Explanation:

- age = int(input()) → Takes age as integer input
- citizen = input().lower() → Converts input to lowercase for consistency
- if age >= 18 and citizen == "yes":

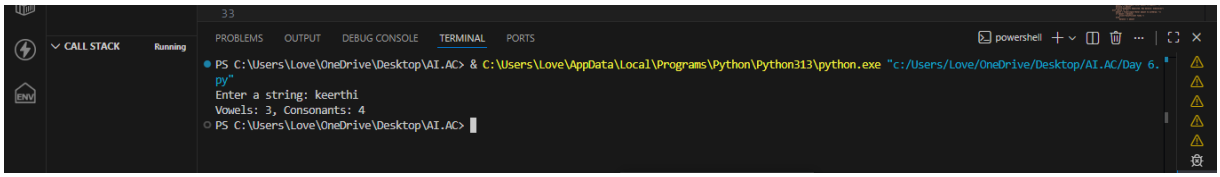
- Checks both age condition and citizenship
- Prints eligibility result accordingly

Task2: Loop-Based String Processing

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



OUTPUT:

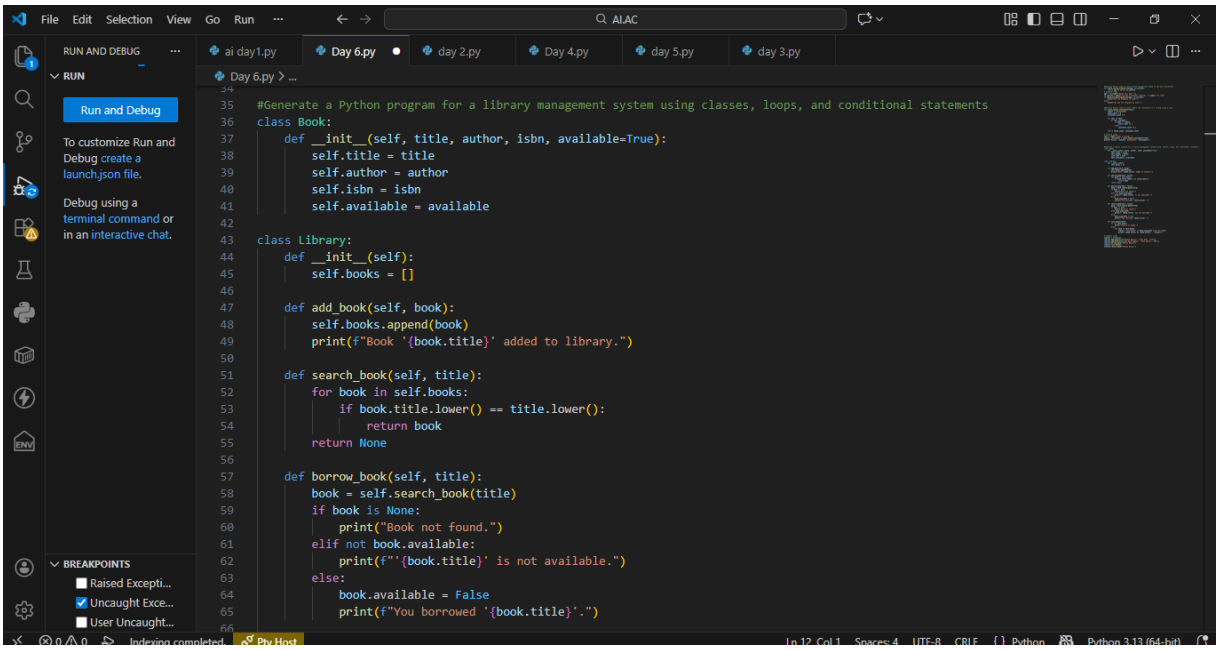


Explanation:

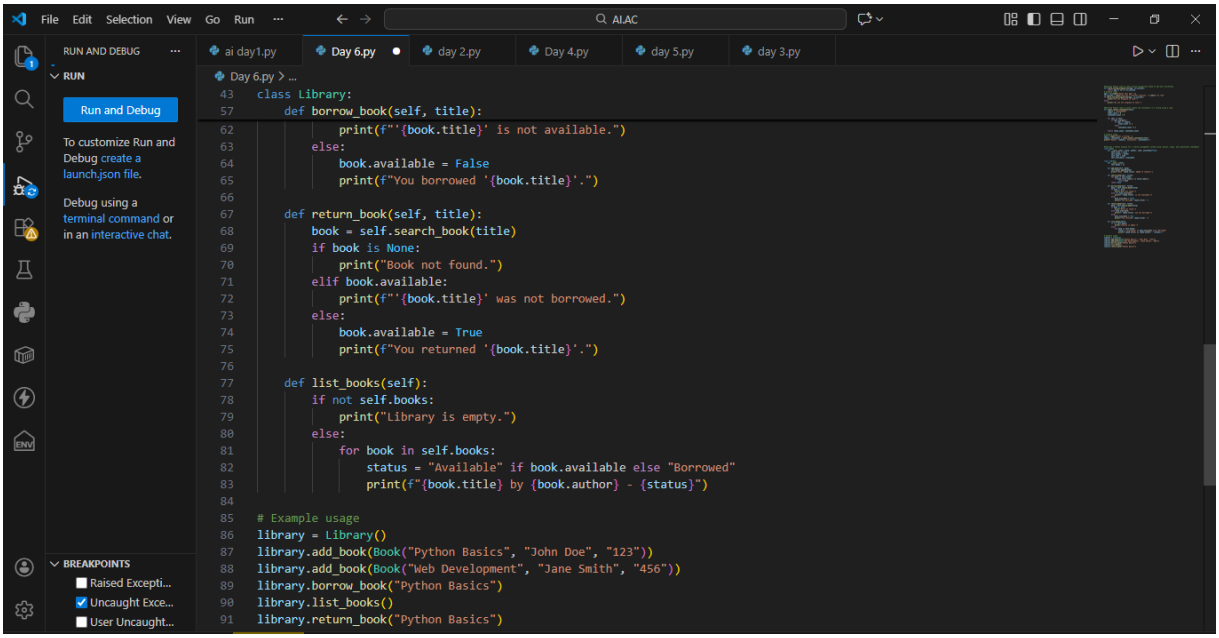
- Converts string to lowercase
- Loops through each character
- isalpha() ensures only letters are counted
- Separates vowels and consonants logically

Task 3: AI-Assisted Code Completion Reflection Task

Prompt : Generate a Python program for a library management system using classes, loops, and conditional statements.

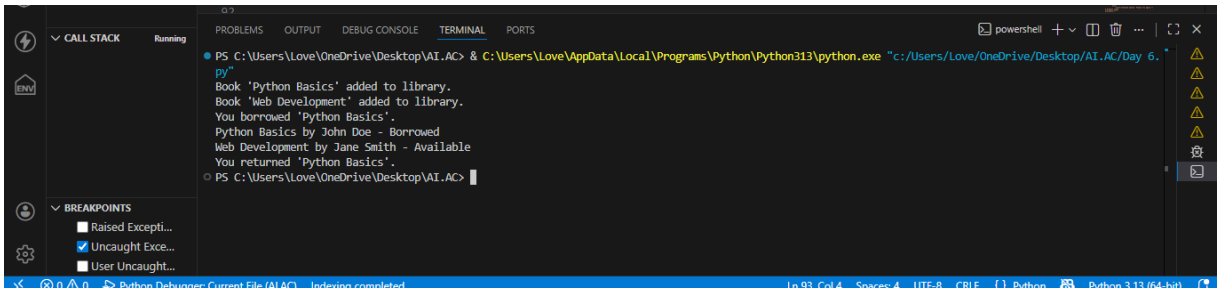


```python
#Generate a Python program for a library management system using classes, loops, and conditional statements
class Book:
    def __init__(self, title, author, isbn, available=True):
        self.title = title
        self.author = author
        self.isbn = isbn
        self.available = available

class Library:
    def __init__(self):
        self.books = []

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

    def search_book(self, title):
        for book in self.books:
            if book.title.lower() == title.lower():
                return book
        return None

    def borrow_book(self, title):
        book = self.search_book(title)
        if book is None:
            print("Book not found.")
        elif not book.available:
            print(f"'{book.title}' is not available.")
        else:
            book.available = False
            print(f"You borrowed '{book.title}'.")
```



```python
class Library:
    def borrow_book(self, title):
            print(f"'{book.title}' is not available.")
        else:
            book.available = False
            print(f"You borrowed '{book.title}'.")

    def return_book(self, title):
        book = self.search_book(title)
        if book is None:
            print("Book not found.")
        elif book.available:
            print(f"'{book.title}' was not borrowed.")
        else:
            book.available = True
            print(f"You returned '{book.title}'.")

    def list_books(self):
        if not self.books:
            print("Library is empty.")
        else:
            for book in self.books:
                status = "Available" if book.available else "Borrowed"
                print(f"{book.title} by {book.author} - {status}")

# Example usage
library = Library()
library.add_book(Book("Python Basics", "John Doe", "123"))
library.add_book(Book("Web Development", "Jane Smith", "456"))
library.borrow_book("Python Basics")
library.list_books()
library.return_book("Python Basics")
```

OUTPUT:

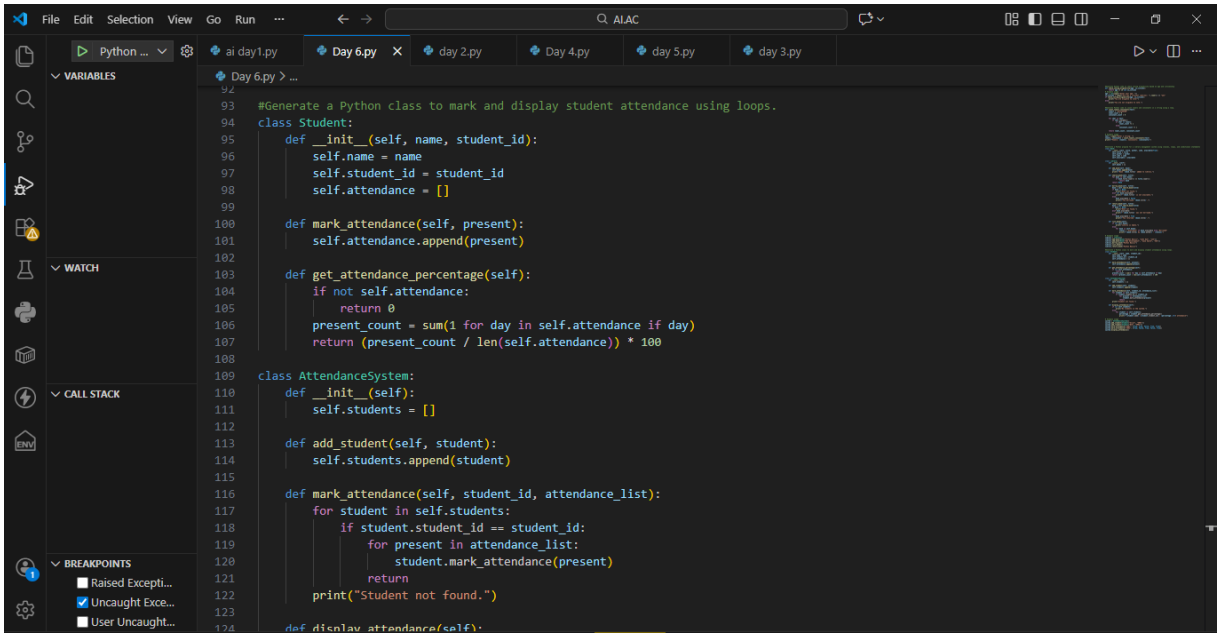


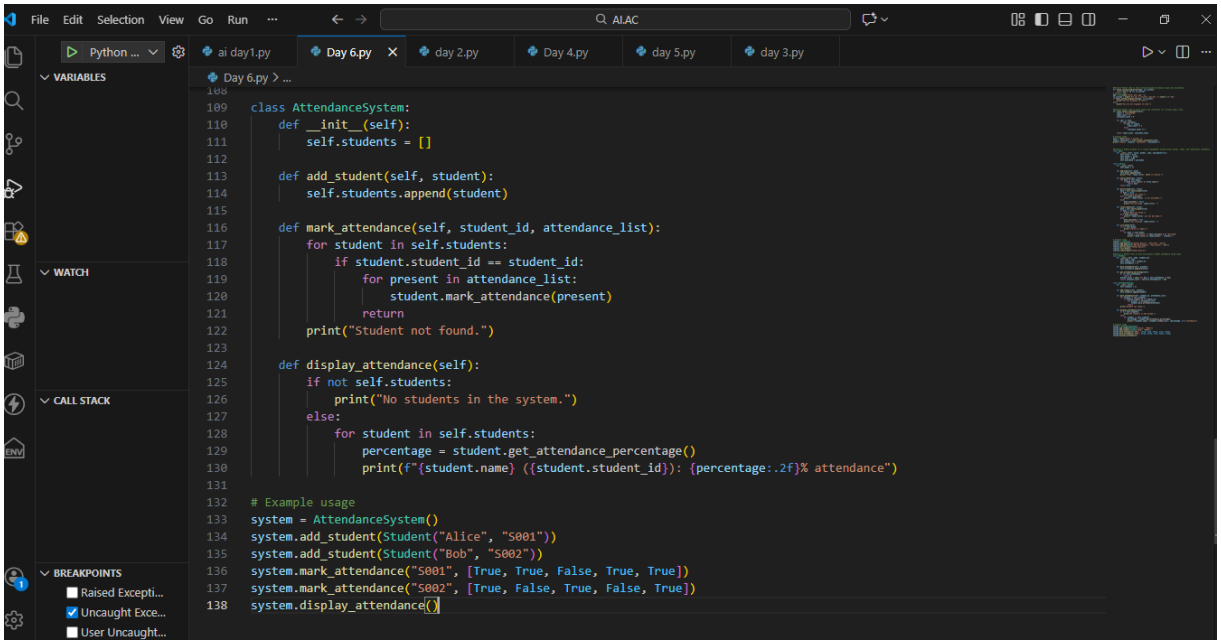
Explanation:

AI-assisted coding helped speed up development and provided a structured solution. However, human review was necessary to improve input validation, error handling, and real-world usability. Responsible AI use requires understanding and modifying generated code rather than copying blindly.

Task 4: Class-Based Attendance System

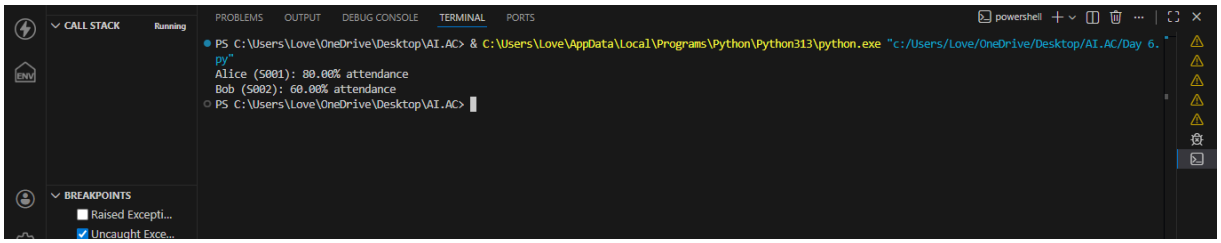
Prompt: Generate a Python class to mark and display student attendance using loops.





```python
class AttendanceSystem:
    def __init__(self):
        self.students = []

    def add_student(self, student):
        self.students.append(student)

    def mark_attendance(self, student_id, attendance_list):
        for student in self.students:
            if student.student_id == student_id:
                for present in attendance_list:
                    student.mark_attendance(present)
                return
        print("Student not found.")

    def display_attendance(self):
        if not self.students:
            print("No students in the system.")
        else:
            for student in self.students:
                percentage = student.get_attendance_percentage()
                print(f"{student.name} ({student.student_id}): {percentage:.2f}% attendance")

# Example usage
system = AttendanceSystem()
system.add_student(Student("Alice", "S001"))
system.add_student(Student("Bob", "S002"))
system.mark_attendance("S001", [True, True, False, True, True])
system.mark_attendance("S002", [True, False, True, False, True])
system.display_attendance()
```

OUTPUT:



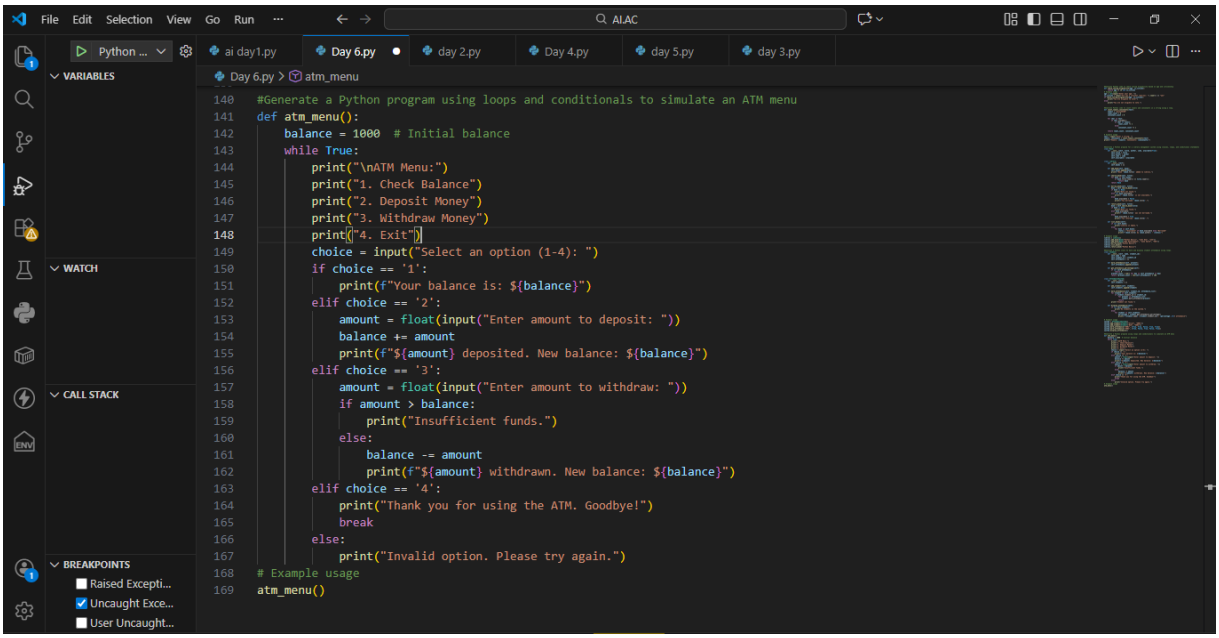
```
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/Day 6.py"
Alice (S001): 80.00% attendance
Bob (S002): 60.00% attendance
PS C:\Users\Love\OneDrive\Desktop\AI.AC>
```

Explanation:

- Dictionary stores student name and attendance
- Loop iterates through records
- Simple and efficient design

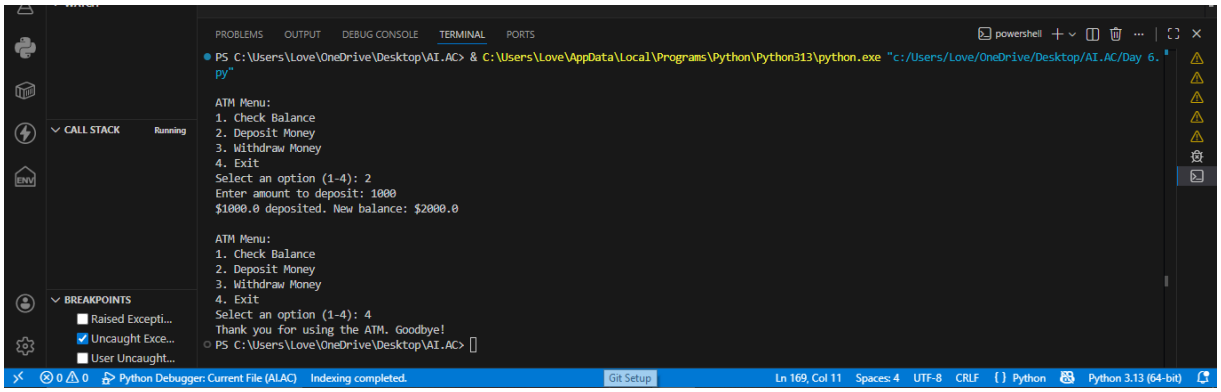
Task 5: Conditional Menu Navigation (ATM Menu)

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



```python
#Generate a Python program using loops and conditionals to simulate an ATM menu
def atm_menu():
    balance = 1000  # Initial balance
    while True:
        print("\nATM Menu:")
        print("1. Check Balance")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Exit")
        choice = input("Select an option (1-4): ")
        if choice == '1':
            print(f"Your balance is: ${balance}")
        elif choice == '2':
            amount = float(input("Enter amount to deposit: "))
            balance += amount
            print(f"${amount} deposited. New balance: ${balance}")
        elif choice == '3':
            amount = float(input("Enter amount to withdraw: "))
            if amount > balance:
                print("Insufficient funds.")
            else:
                balance -= amount
                print(f"${amount} withdrawn. New balance: ${balance}")
        elif choice == '4':
            print("Thank you for using the ATM. Goodbye!")
            break
        else:
            print("Invalid option. Please try again.")
# Example usage
atm_menu()
```

OUTPUT:



```
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/Day 6.py"

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Select an option (1-4): 2
Enter amount to deposit: 1000
$1000.0 deposited. New balance: $2000.0

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Select an option (1-4): 4
Thank you for using the ATM. Goodbye!
PS C:\Users\Love\OneDrive\Desktop\AI.AC>
```

Explanation:

- Correct balance update
- Prevents overdraft
- Loop exits safely