A.Abhiram

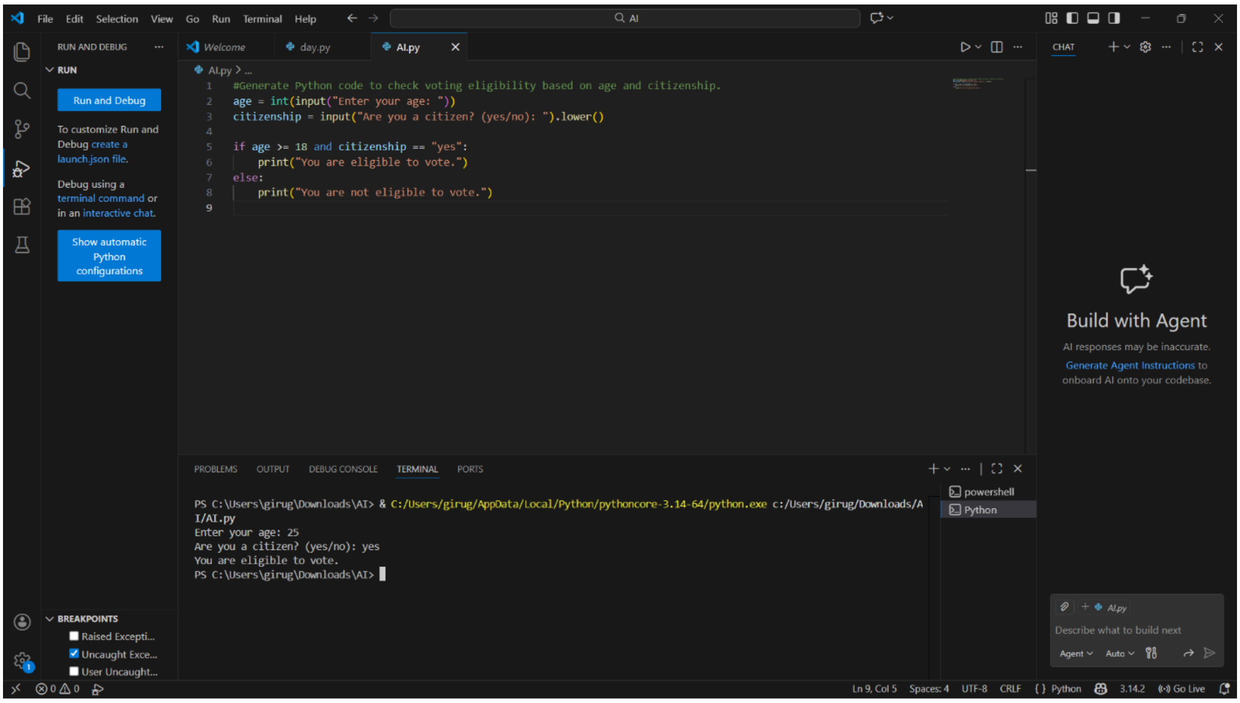
2403a51l13

Batch-51

Lab 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals

Task Description – 1: AI-Based Code Completion for Conditional Eligibility Check.

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



```python
#Generate Python code to check voting eligibility based on age and citizenship.
age = int(input("Enter your age: "))
citizenship = input("Are you a citizen? (yes/no): ").lower()

if age >= 18 and citizenship == "yes":
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

```
PS C:\Users\girug\Downloads\AI> & C:/Users/girug/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/girug/Downloads/A
I/AI.py
Enter your age: 25
Are you a citizen? (yes/no): yes
You are eligible to vote.
PS C:\Users\girug\Downloads\AI> 
```

Justification:

It checks voting eligibility using conditional statements. It verifies whether the user’s age is 18 or above and if the person is a citizen. Based on these conditions, it displays eligibility status.

age >= 18 → Checks minimum voting age.

citizen == "yes" → Ensures citizenship.

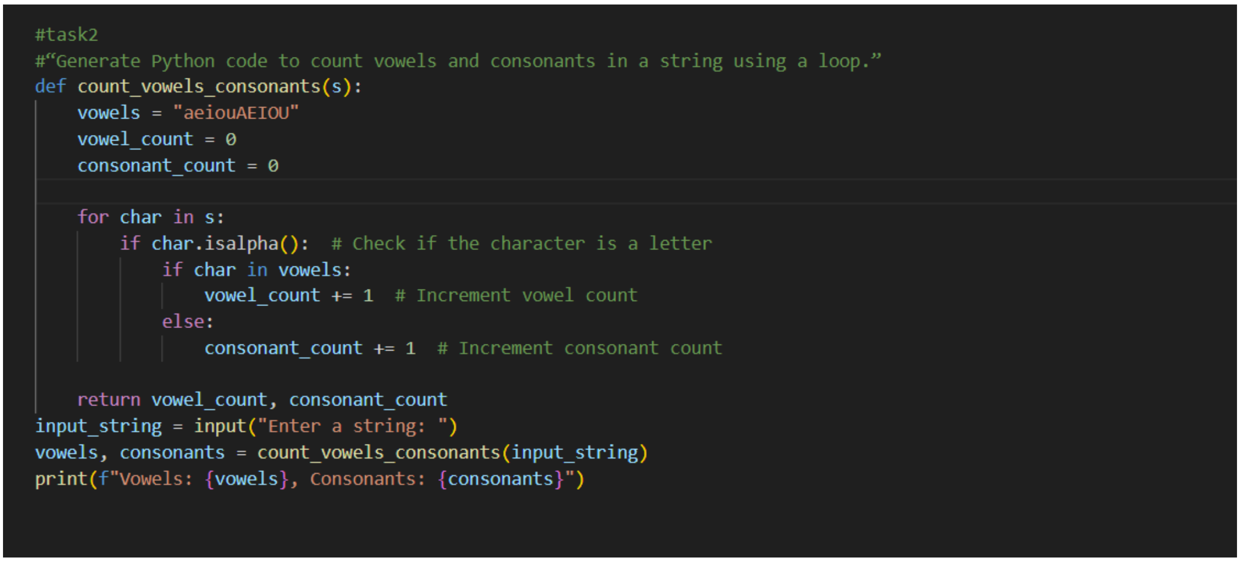
and → Both conditions must be true.

if-else → Makes the eligibility decision.

Task Description – 2: AI-Based Code Completion for Loop-Based String Processing.

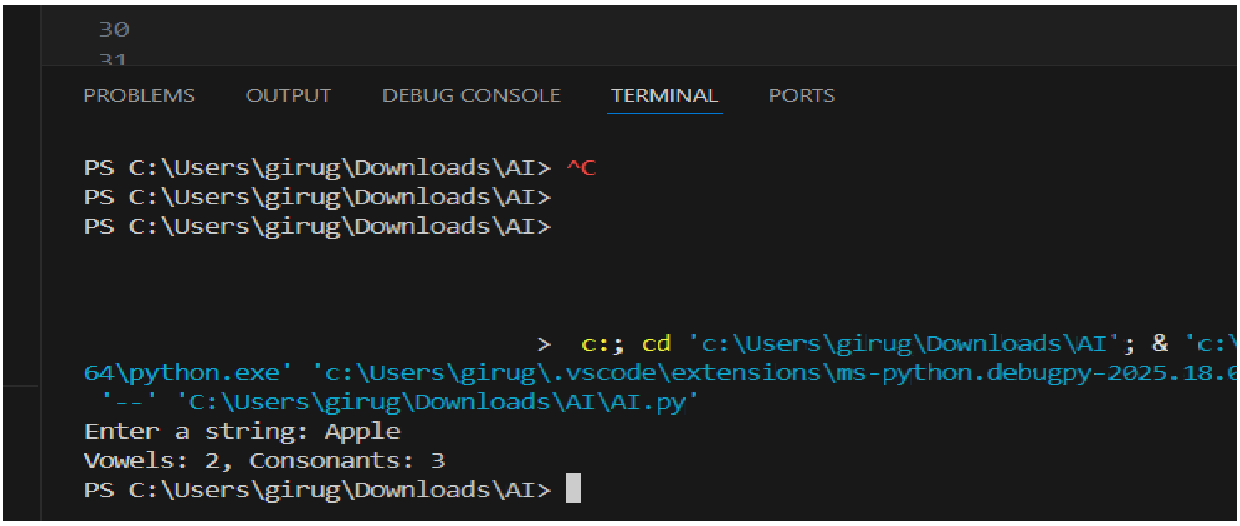
Prompt: Generate Python code to count vowels and consonants in a string using a loop.

AI-Generated Code:



```python
#task2
#“Generate Python code to count vowels and consonants in a string using a loop.”
def count_vowels_consonants(s):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

    for char in s:
        if char.isalpha():  # Check if the character is a letter
            if char in vowels:
                vowel_count += 1  # Increment vowel count
            else:
                consonant_count += 1  # Increment consonant count

    return vowel_count, consonant_count
input_string = input("Enter a string: ")
vowels, consonants = count_vowels_consonants(input_string)
print(f"Vowels: {vowels}, Consonants: {consonants}")
```

OUTPUT:



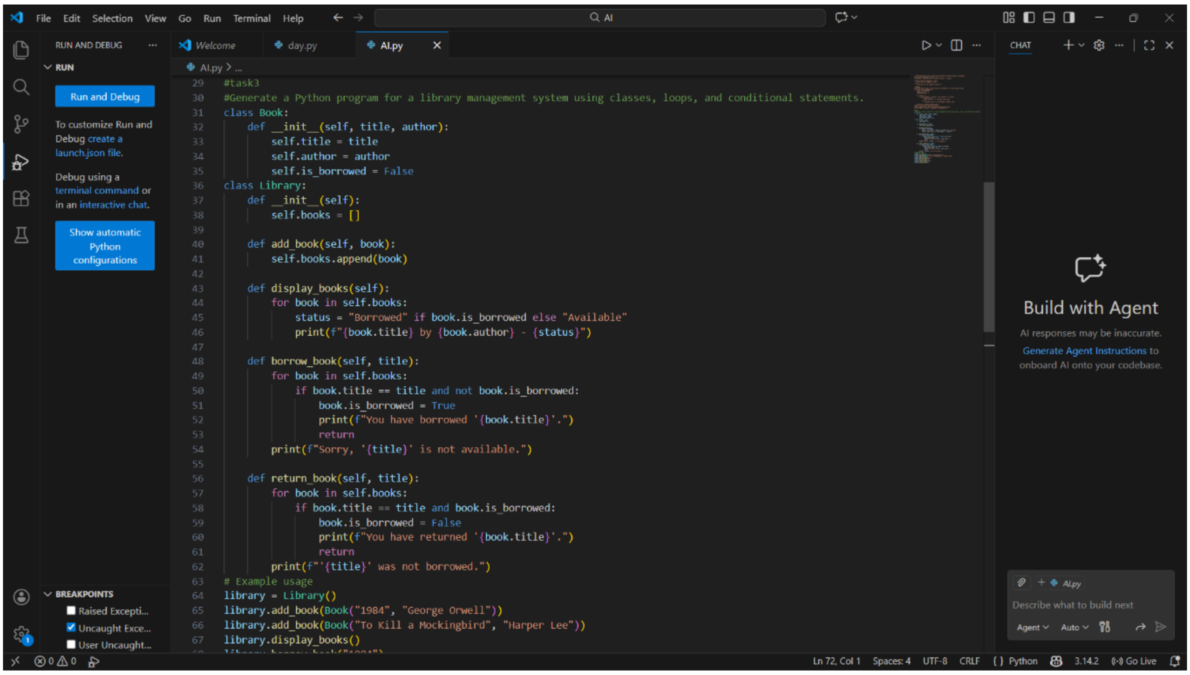
```
PS C:\Users\girug\Downloads\AI> ^C
PS C:\Users\girug\Downloads\AI>
PS C:\Users\girug\Downloads\AI>



                               >  c:; cd 'c:\Users\girug\Downloads\AI'; & 'c:\
64\python.exe' 'c:\Users\girug\.vscode\extensions\ms-python.debugpy-2025.18.
 '--' 'C:\Users\girug\Downloads\AI\AI.py'
Enter a string: Apple
Vowels: 2, Consonants: 3
PS C:\Users\girug\Downloads\AI>
```

Justification:

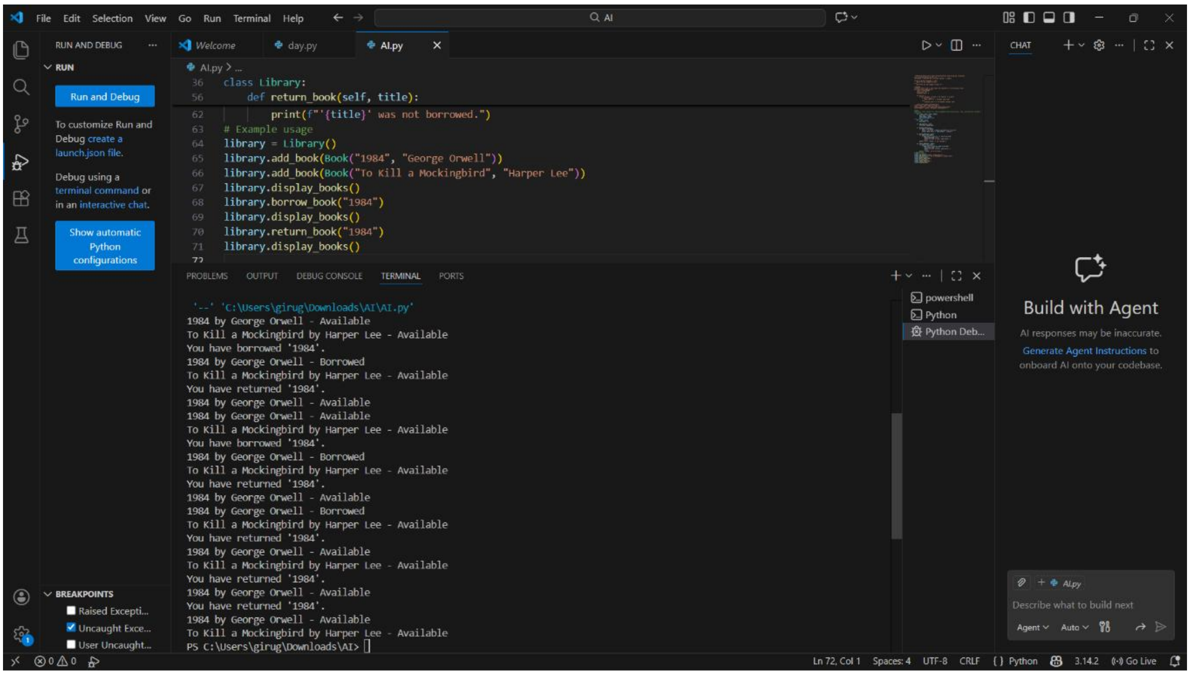
This task uses AI-generated Python code to count vowels and consonants in a given string using a loop. The program reads a string input from the user and converts it to lowercase to avoid case sensitivity. A for loop iterates through each character in the string.If it is a vowel, the vowel count is incremented; otherwise, the consonant count is increased. This task demonstrates the use of loops and nested conditionals for string processing and character analysis.

Task Description – 3: AI-Assisted Code Completion Reflection Task

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



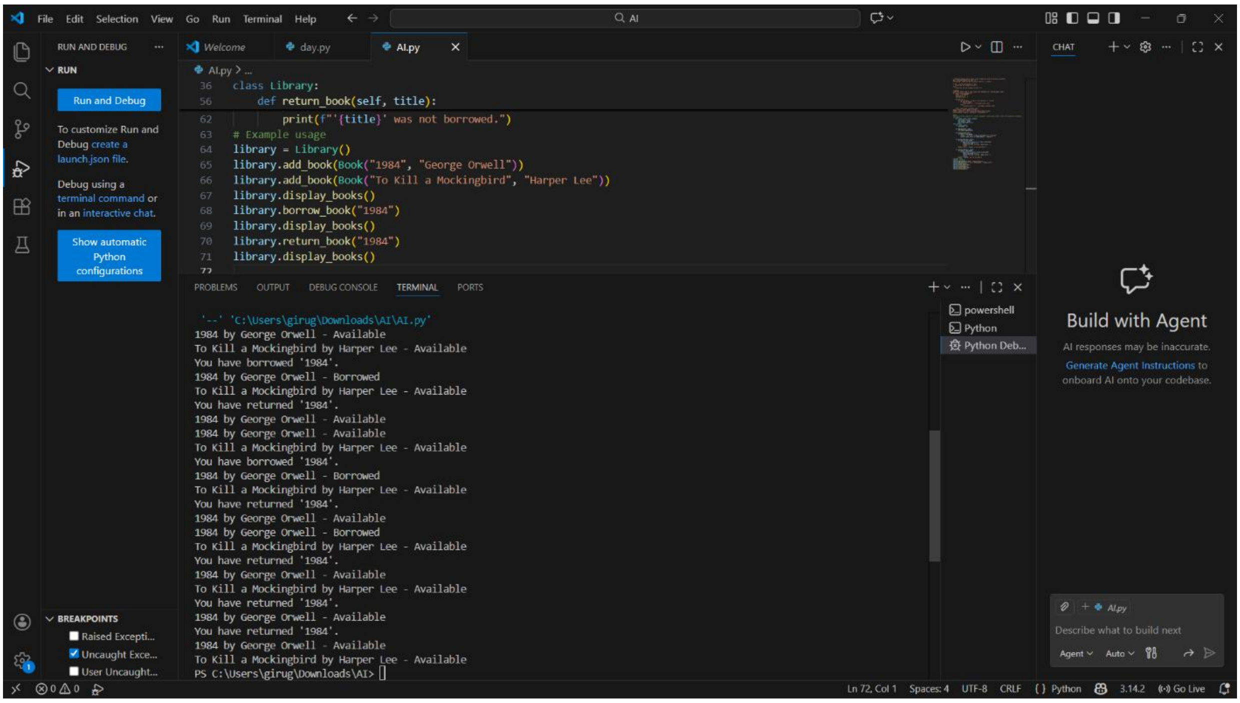
```python
#task3
#Generate a Python program for a library management system using classes, loops, and conditional statements.
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.is_borrowed = False
class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book):
        self.books.append(book)

    def display_books(self):
        for book in self.books:
            status = "Borrowed" if book.is_borrowed else "Available"
            print(f"{book.title} by {book.author} - {status}")

    def borrow_book(self, title):
        for book in self.books:
            if book.title == title and not book.is_borrowed:
                book.is_borrowed = True
                print(f"You have borrowed '{book.title}'.")
                return
        print(f"Sorry, '{title}' is not available.")

    def return_book(self, title):
        for book in self.books:
            if book.title == title and book.is_borrowed:
                book.is_borrowed = False
                print(f"You have returned '{book.title}'.")
                return
        print(f"'{title}' was not borrowed.")
# Example usage
library = Library()
library.add_book(Book("1984", "George Orwell"))
library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
library.display_books()
```



```python
class Library:
    def return_book(self, title):
        print(f"'{title}' was not borrowed.")
# Example usage
library = Library()
library.add_book(Book("1984", "George Orwell"))
library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
library.display_books()
library.borrow_book("1984")
library.display_books()
library.return_book("1984")
library.display_books()
```

```
 '--' 'C:\Users\girug\Downloads\AI\AI.py'
1984 by George Orwell - Available
To Kill a Mockingbird by Harper Lee - Available
You have borrowed '1984'.
1984 by George Orwell - Borrowed
To Kill a Mockingbird by Harper Lee - Available
You have returned '1984'.
1984 by George Orwell - Available
1984 by George Orwell - Available
To Kill a Mockingbird by Harper Lee - Available
You have borrowed '1984'.
1984 by George Orwell - Borrowed
To Kill a Mockingbird by Harper Lee - Available
You have returned '1984'.
1984 by George Orwell - Available
1984 by George Orwell - Borrowed
To Kill a Mockingbird by Harper Lee - Available
You have returned '1984'.
1984 by George Orwell - Available
To Kill a Mockingbird by Harper Lee - Available
You have returned '1984'.
1984 by George Orwell - Available
You have returned '1984'.
1984 by George Orwell - Available
To Kill a Mockingbird by Harper Lee - Available
PS C:\Users\girug\Downloads\AI> 
```

OUTPUT:


File Edit Selection View Go Run Terminal Help
AI
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Debug using a terminal command or in an interactive chat.
Show automatic Python configurations
BREAKPOINTS
☐ Raised Excepti...
☑ Uncaught Exce...
☐ User Uncaught...
Welcome
day.py
AI.py
AI.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell
Python
Python Deb...
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
AI.py
Describe what to build next
Agent
Auto
Ln 72, Col 1
Spaces: 4
UTF-8
CRLF
{ } Python
3.14.2
Go Live

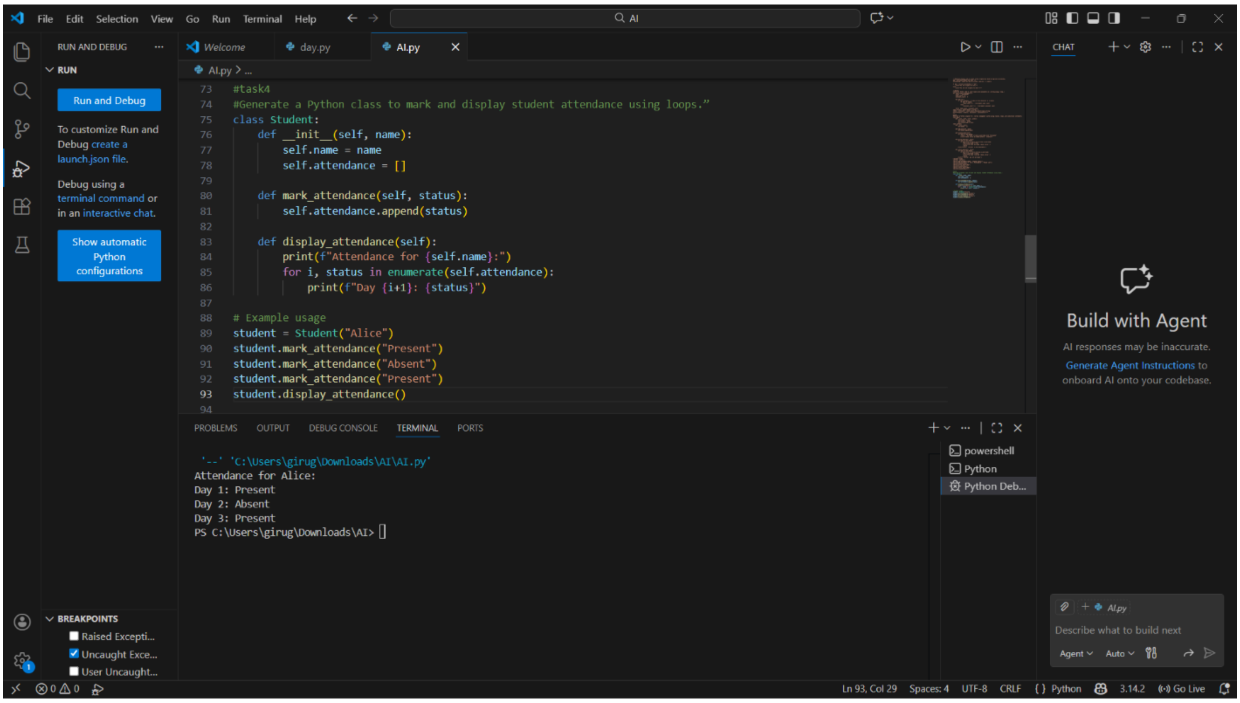
```
class Library:
    def return_book(self, title):
            print(f"'{title}' was not borrowed.")
# Example usage
library = Library()
library.add_book(Book("1984", "George Orwell"))
library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
library.display_books()
library.borrow_book("1984")
library.display_books()
library.return_book("1984")
library.display_books()
```

```
 '--' 'C:\Users\girug\Downloads\AI\AI.py'
1984 by George Orwell - Available
To Kill a Mockingbird by Harper Lee - Available
You have borrowed '1984'.
1984 by George Orwell - Borrowed
To Kill a Mockingbird by Harper Lee - Available
You have returned '1984'.
1984 by George Orwell - Available
1984 by George Orwell - Available
To Kill a Mockingbird by Harper Lee - Available
You have borrowed '1984'.
1984 by George Orwell - Borrowed
To Kill a Mockingbird by Harper Lee - Available
You have returned '1984'.
1984 by George Orwell - Available
1984 by George Orwell - Borrowed
To Kill a Mockingbird by Harper Lee - Available
You have returned '1984'.
1984 by George Orwell - Available
To Kill a Mockingbird by Harper Lee - Available
You have returned '1984'.
1984 by George Orwell - Available
You have returned '1984'.
1984 by George Orwell - Available
To Kill a Mockingbird by Harper Lee - Available
PS C:\Users\girug\Downloads\AI> 
```

Justification: 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 Description – 4: AI-Assisted Code Completion for Class-Based Attendance System

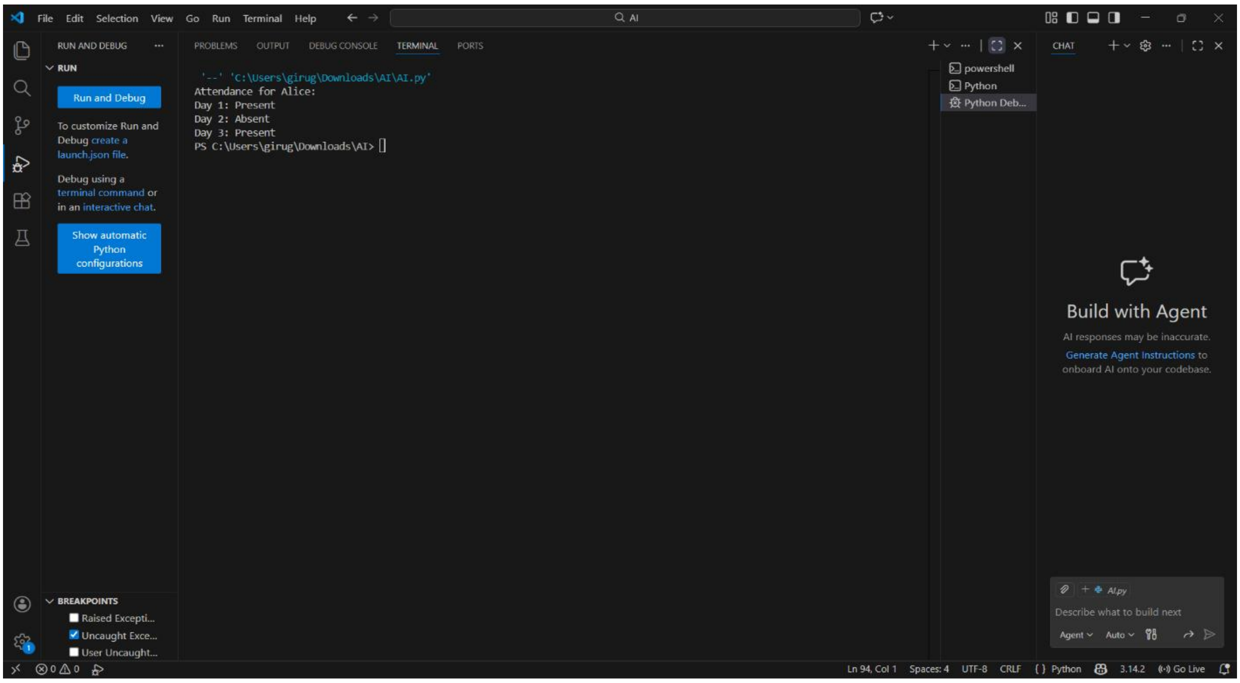
Prompt: Generate a Python class to mark and display student attendance using loops.



```python
#task4
#Generate a Python class to mark and display student attendance using loops.”
class Student:
    def __init__(self, name):
        self.name = name
        self.attendance = []

    def mark_attendance(self, status):
        self.attendance.append(status)

    def display_attendance(self):
        print(f"Attendance for {self.name}:")
        for i, status in enumerate(self.attendance):
            print(f"Day {i+1}: {status}")

# Example usage
student = Student("Alice")
student.mark_attendance("Present")
student.mark_attendance("Absent")
student.mark_attendance("Present")
student.display_attendance()
```

```
 '--' 'C:\Users\girug\Downloads\AI\AI.py'
Attendance for Alice:
Day 1: Present
Day 2: Absent
Day 3: Present
PS C:\Users\girug\Downloads\AI> 
```

OUTPUT:



```
 '--' 'C:\Users\girug\Downloads\AI\AI.py'
Attendance for Alice:
Day 1: Present
Day 2: Absent
Day 3: Present
PS C:\Users\girug\Downloads\AI> 
```

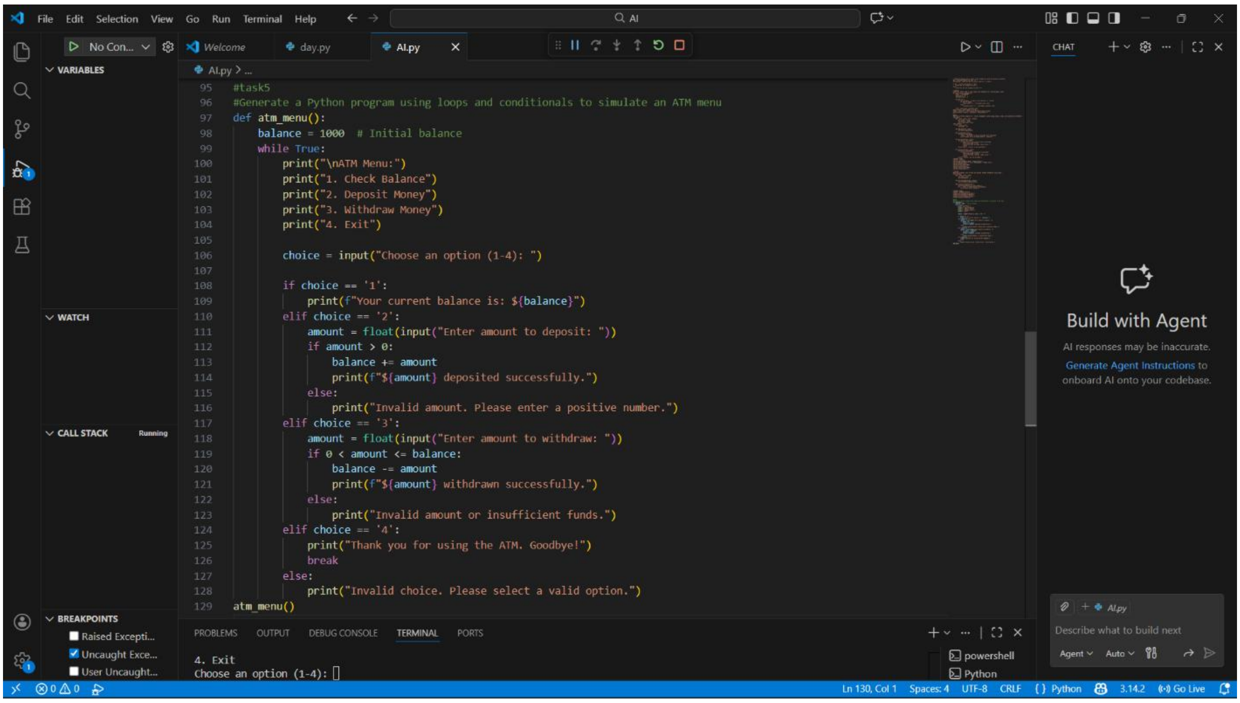
Observations:

- Dictionary stores student name and attendance

- Loop iterates through records
- Simple and efficient design

Task Description – 5: AI-Based Code Completion for Conditional Menu Navigation

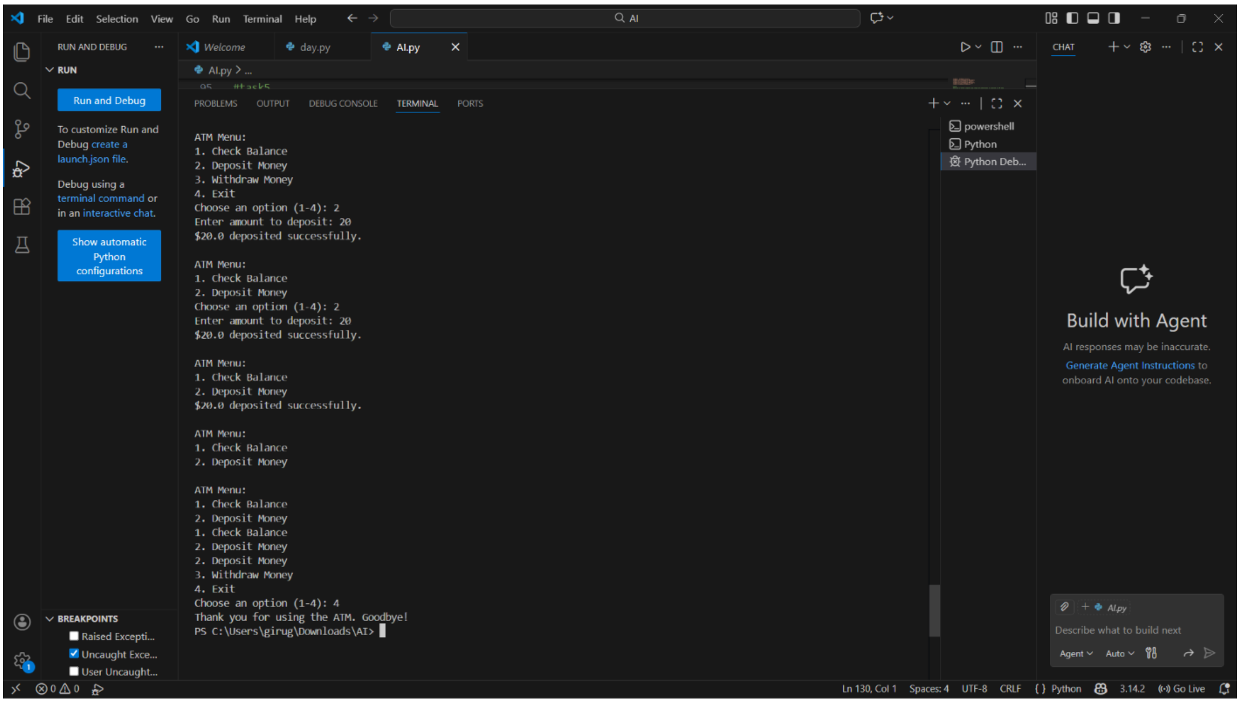
Prompt: Generate a Python program using loops and conditionals to simulate an ATM menu.



```python
#task5
#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("Choose an option (1-4): ")

        if choice == '1':
            print(f"Your current balance is: ${balance}")
        elif choice == '2':
            amount = float(input("Enter amount to deposit: "))
            if amount > 0:
                balance += amount
                print(f"${amount} deposited successfully.")
            else:
                print("Invalid amount. Please enter a positive number.")
        elif choice == '3':
            amount = float(input("Enter amount to withdraw: "))
            if 0 < amount <= balance:
                balance -= amount
                print(f"${amount} withdrawn successfully.")
            else:
                print("Invalid amount or insufficient funds.")
        elif choice == '4':
            print("Thank you for using the ATM. Goodbye!")
            break
        else:
            print("Invalid choice. Please select a valid option.")
atm_menu()
```

```
4. Exit
Choose an option (1-4): 
```

OUTPUT:



To customize Run and Debug create a launch.json file.

Debug using a terminal command or in an interactive chat.

```
ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Choose an option (1-4): 2
Enter amount to deposit: 20
$20.0 deposited successfully.

ATM Menu:
1. Check Balance
2. Deposit Money
Choose an option (1-4): 2
Enter amount to deposit: 20
$20.0 deposited successfully.

ATM Menu:
1. Check Balance
2. Deposit Money
$20.0 deposited successfully.

ATM Menu:
1. Check Balance
2. Deposit Money

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

Build with Agent

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

Describe what to build next

Justification:

- Correct balance update
- Prevents overdraft
- Loop exits safely