

ASSIGNMENT 6.5

Atla Sreeja

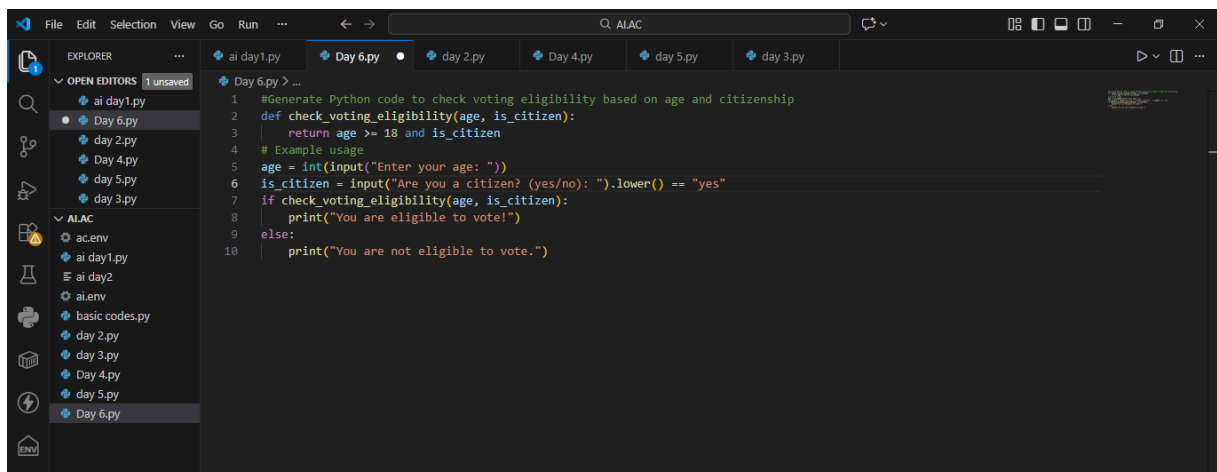
2403A51L02

B-51

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

Task 1: Conditional Eligibility Check

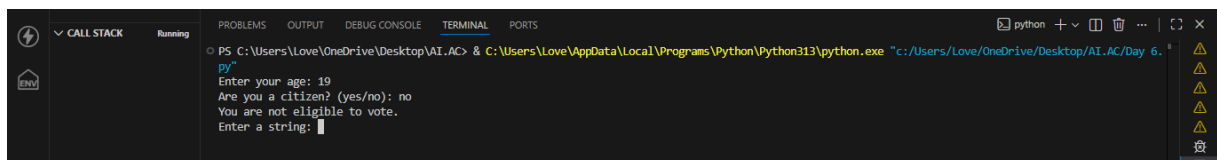
Prompt: Generate Python code to check voting eligibility based on age and citizenship.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'AIAC' with several files, including 'ai day 1.py', 'Day 6.py', 'day 2.py', 'Day 4.py', 'day 5.py', and 'day 3.py'. The code editor shows the following Python code:

```
1 #Generate Python code to check voting eligibility based on age and citizenship
2 def check_voting_eligibility(age, is_citizen):
3     return age >= 18 and is_citizen
4 # Example usage
5 age = int(input("Enter your age: "))
6 is_citizen = input("Are you a citizen? (yes/no): ").lower() == "yes"
7 if check_voting_eligibility(age, is_citizen):
8     print("You are eligible to vote!")
9 else:
10    print("You are not eligible to vote.")
```

OUTPUT:



The screenshot shows a terminal window with the following 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"
Enter your age: 19
Are you a citizen? (yes/no): no
You are not eligible to vote.
Enter a string: 
```

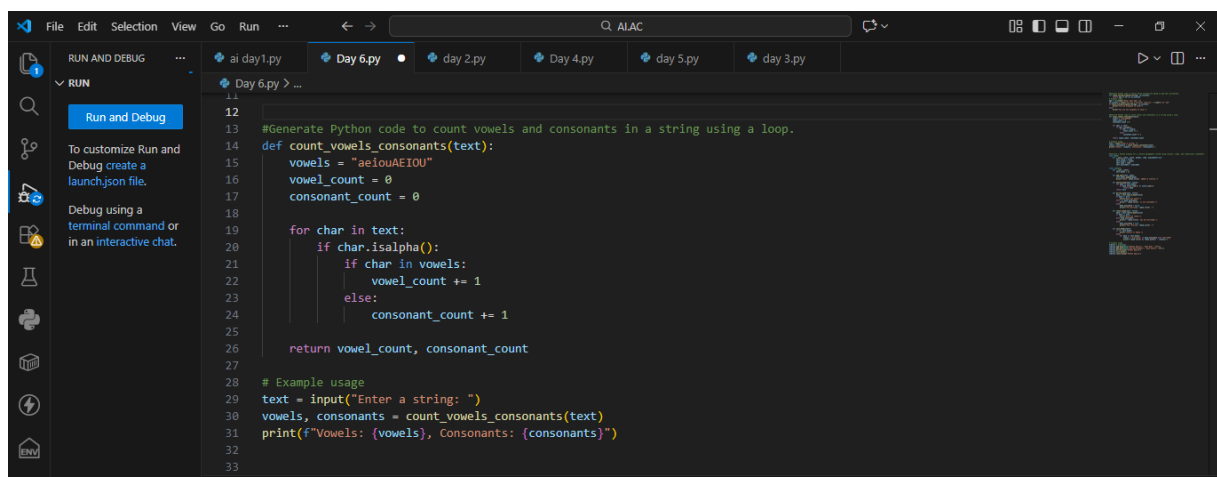
Explanation:

- `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

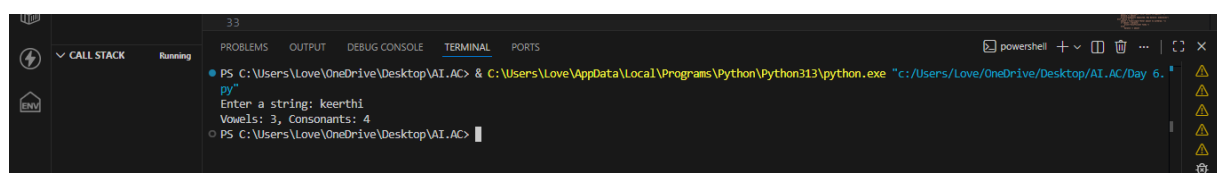
Prompt: Generate Python code to count vowels and consonants in a string using a loop.



```

11
12
13 #Generate Python code to count vowels and consonants in a string using a loop.
14 def count_vowels_consonants(text):
15     vowels = "aeiouAEIOU"
16     vowel_count = 0
17     consonant_count = 0
18
19     for char in text:
20         if char.isalpha():
21             if char in vowels:
22                 vowel_count += 1
23             else:
24                 consonant_count += 1
25
26     return vowel_count, consonant_count
27
28 # Example usage
29 text = input("Enter a string: ")
30 vowels, consonants = count_vowels_consonants(text)
31 print(f"Vowels: {vowels}, Consonants: {consonants}")
32
33
  
```

OUTPUT:



```

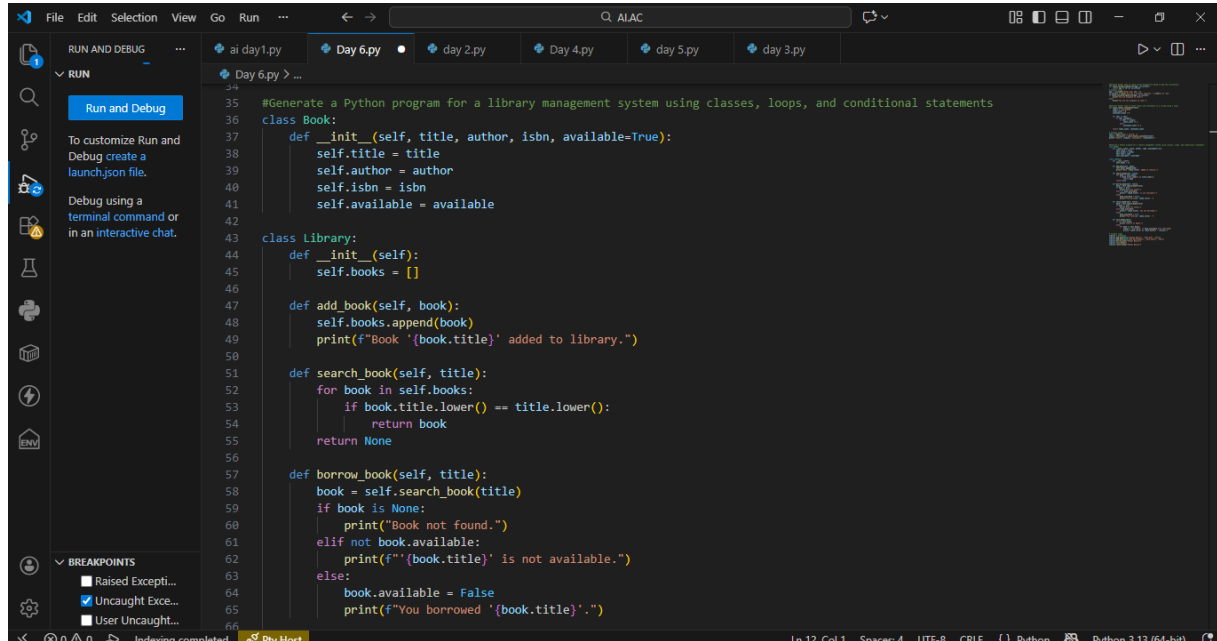
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/Day 6.py"
Enter a string: keertih
Vowels: 3, Consonants: 4
PS C:\Users\Love\OneDrive\Desktop\AI.AC>
  
```

Explanation:

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

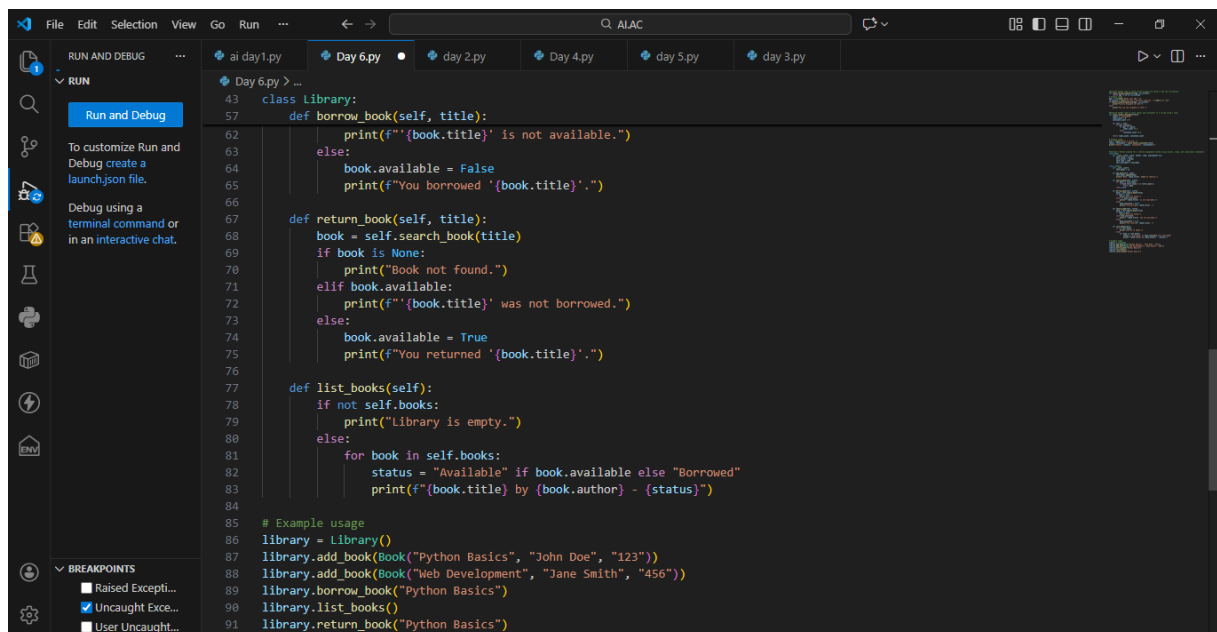
Task 3: AI-Assisted Code Completion Reflection Task

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



The screenshot shows the VS Code editor with a Python file named 'Day 6.py'. The code defines two classes: 'Book' and 'Library'. The 'Book' class has attributes 'title', 'author', 'isbn', and 'available'. The 'Library' class has a 'books' list and methods for adding, searching, and borrowing books. The code is as follows:

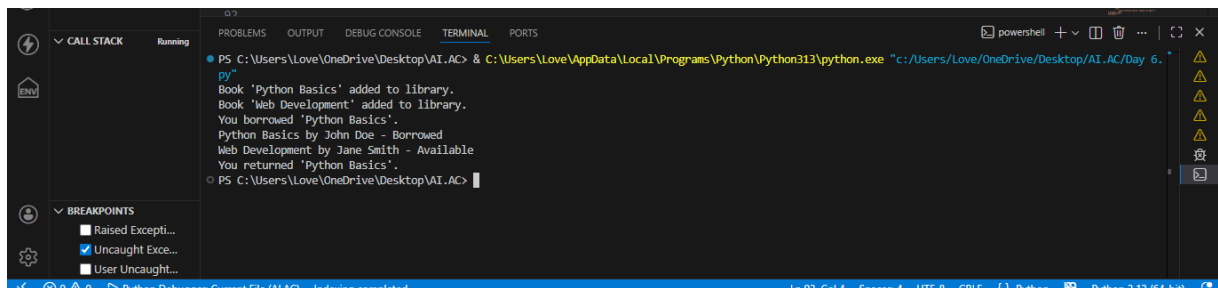
```
35 #Generate a Python program for a library management system using classes, loops, and conditional statements
36 class Book:
37     def __init__(self, title, author, isbn, available=True):
38         self.title = title
39         self.author = author
40         self.isbn = isbn
41         self.available = available
42
43 class Library:
44     def __init__(self):
45         self.books = []
46
47     def add_book(self, book):
48         self.books.append(book)
49         print(f"Book '{book.title}' added to library.")
50
51     def search_book(self, title):
52         for book in self.books:
53             if book.title.lower() == title.lower():
54                 return book
55         return None
56
57     def borrow_book(self, title):
58         book = self.search_book(title)
59         if book is None:
60             print("Book not found.")
61         elif not book.available:
62             print(f"'{book.title}' is not available.")
63         else:
64             book.available = False
65             print(f"You borrowed '{book.title}'.")
```



The screenshot shows the VS Code editor with the same Python file 'Day 6.py'. The code is now complete, including methods for returning books and listing all books. The code is as follows:

```
43 class Library:
57     def borrow_book(self, title):
62         print(f"'{book.title}' is not available.")
63     else:
64         book.available = False
65         print(f"You borrowed '{book.title}'.")
66
67     def return_book(self, title):
68         book = self.search_book(title)
69         if book is None:
70             print("Book not found.")
71         elif book.available:
72             print(f"'{book.title}' was not borrowed.")
73         else:
74             book.available = True
75             print(f"You returned '{book.title}'.")
76
77     def list_books(self):
78         if not self.books:
79             print("Library is empty.")
80         else:
81             for book in self.books:
82                 status = "Available" if book.available else "Borrowed"
83                 print(f"{book.title} by {book.author} - {status}")
84
85 # Example usage
86 library = Library()
87 library.add_book(Book("Python Basics", "John Doe", "123"))
88 library.add_book(Book("Web Development", "Jane Smith", "456"))
89 library.borrow_book("Python Basics")
90 library.list_books()
91 library.return_book("Python Basics")
```

OUTPUT:



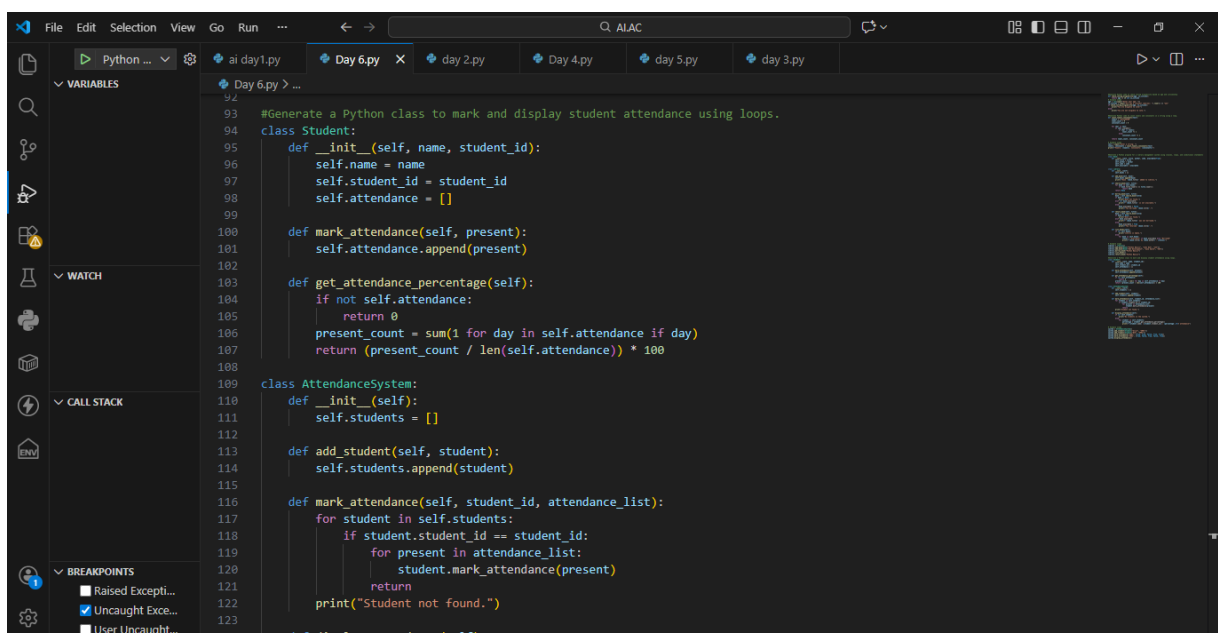
```
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/Day 6.py"
Book 'Python Basics' added to library.
Book 'Web Development' added to library.
You borrowed 'Python Basics'.
Python Basics by John Doe - Borrowed
Web Development by Jane Smith - Available
You returned 'Python Basics'.
```

Explanation:

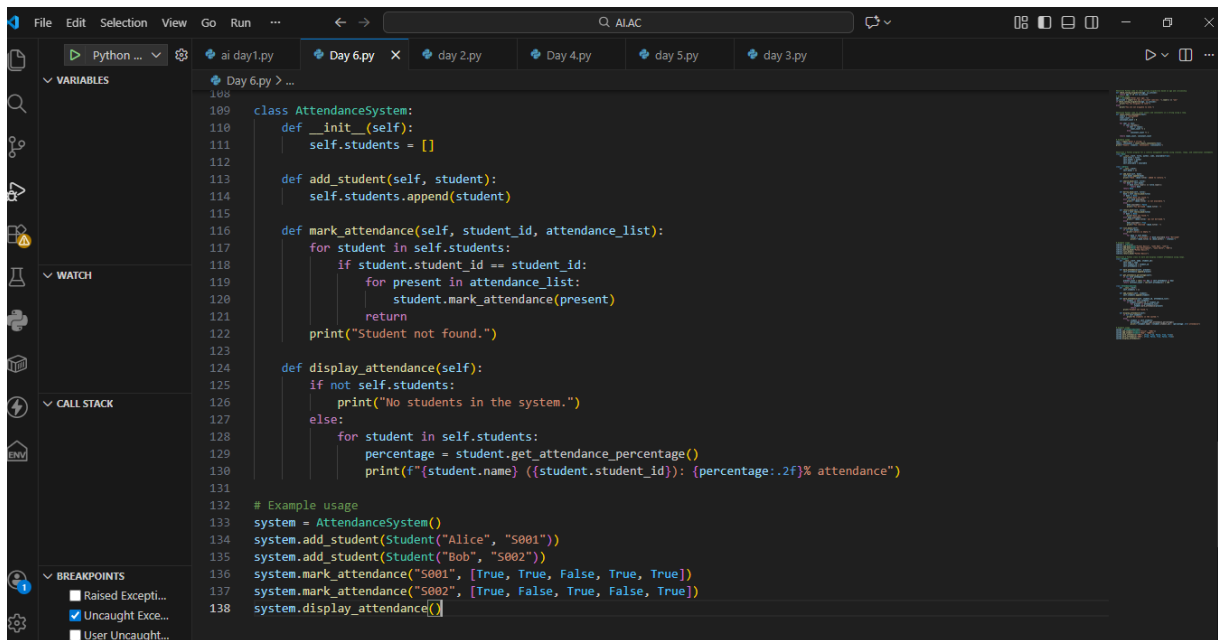
AI-assisted coding helped speed up development and provided a structured solution. However, human review was necessary to improve input validation, error handling, and real-world usability. Responsible AI use requires understanding and modifying generated code rather than copying blindly.

Task 4: Class-Based Attendance System

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



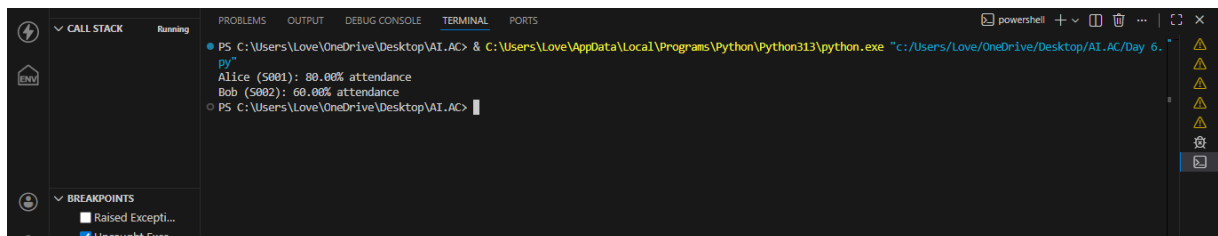
```
92
93 #Generate a Python class to mark and display student attendance using loops.
94 class Student:
95     def __init__(self, name, student_id):
96         self.name = name
97         self.student_id = student_id
98         self.attendance = []
99
100     def mark_attendance(self, present):
101         self.attendance.append(present)
102
103     def get_attendance_percentage(self):
104         if not self.attendance:
105             return 0
106         present_count = sum(1 for day in self.attendance if day)
107         return (present_count / len(self.attendance)) * 100
108
109 class AttendanceSystem:
110     def __init__(self):
111         self.students = []
112
113     def add_student(self, student):
114         self.students.append(student)
115
116     def mark_attendance(self, student_id, attendance_list):
117         for student in self.students:
118             if student.student_id == student_id:
119                 for present in attendance_list:
120                     student.mark_attendance(present)
121                 return
122         print("Student not found.")
123
124     def display_attendance(self):
```



The screenshot shows a Python IDE with a dark theme. The main editor window displays the code for an `AttendanceSystem` class. The class has three methods: `__init__` (initializes `self.students` as an empty list), `add_student` (appends a `Student` object to `self.students`), and `mark_attendance` (iterates through `self.students` and marks attendance for a given `student_id` and `attendance_list`). A `display_attendance` method is also present, which prints the attendance percentage for each student. The code includes example usage at the bottom, creating an `AttendanceSystem` object, adding two students (Alice and Bob), marking their attendance, and displaying the results.

```
108
109 class AttendanceSystem:
110     def __init__(self):
111         self.students = []
112
113     def add_student(self, student):
114         self.students.append(student)
115
116     def mark_attendance(self, student_id, attendance_list):
117         for student in self.students:
118             if student.student_id == student_id:
119                 for present in attendance_list:
120                     student.mark_attendance(present)
121                 return
122         print("Student not found.")
123
124     def display_attendance(self):
125         if not self.students:
126             print("No students in the system.")
127         else:
128             for student in self.students:
129                 percentage = student.get_attendance_percentage()
130                 print(f"{student.name} ({student.student_id}): {percentage:.2f}% attendance")
131
132 # Example usage
133 system = AttendanceSystem()
134 system.add_student(Student("Alice", "S001"))
135 system.add_student(Student("Bob", "S002"))
136 system.mark_attendance("S001", [True, True, False, True, True])
137 system.mark_attendance("S002", [True, False, True, False, True])
138 system.display_attendance()
```

OUTPUT:



The screenshot shows a PowerShell terminal window with the output of the Python program. The output displays the attendance percentage for each student: Alice (S001) with 80.00% attendance and Bob (S002) with 60.00% attendance.

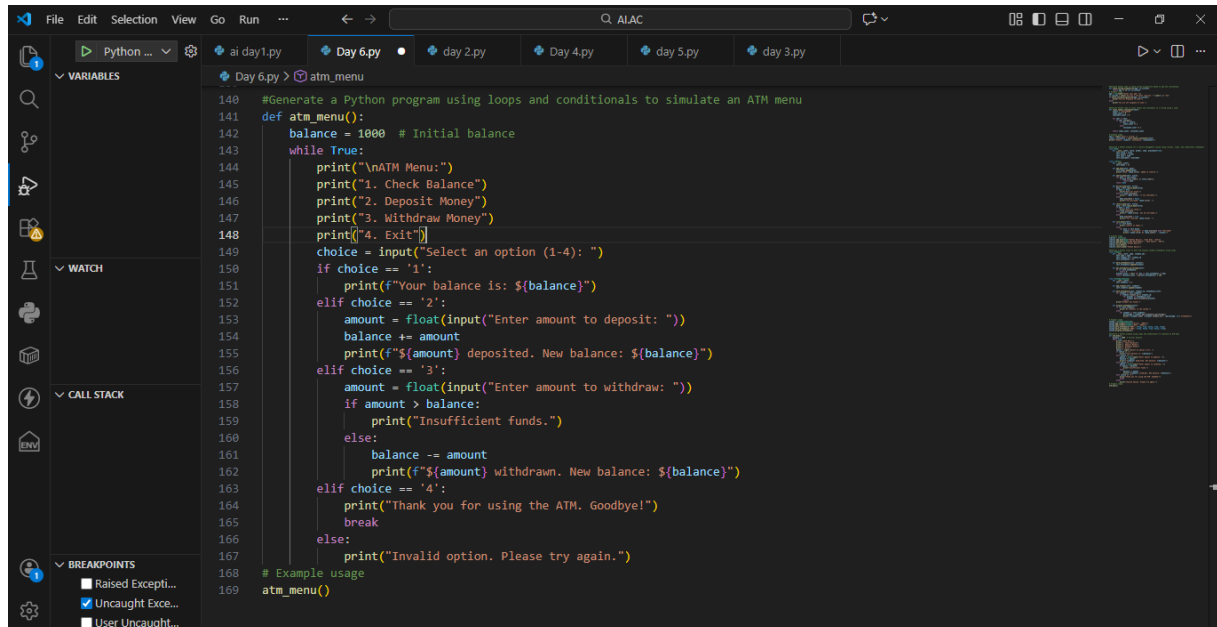
```
PS C:\Users\Love\OneDrive\Desktop\AI.AC> & C:\Users\Love\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Love/OneDrive/Desktop/AI.AC/Day 6.py"
Alice (S001): 80.00% attendance
Bob (S002): 60.00% attendance
PS C:\Users\Love\OneDrive\Desktop\AI.AC>
```

Explanation:

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

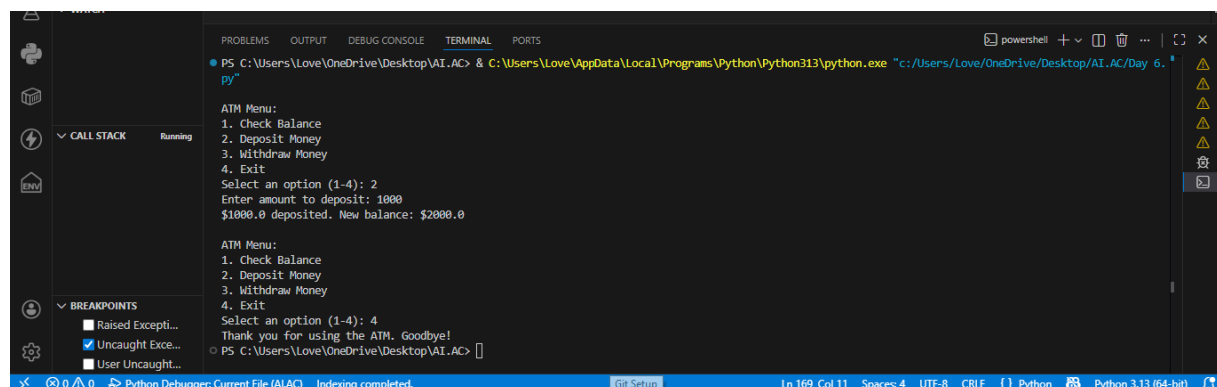
Task 5: Conditional Menu Navigation (ATM Menu)

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



```
140 #Generate a Python program using loops and conditionals to simulate an ATM menu
141 def atm_menu():
142     balance = 1000 # Initial balance
143     while True:
144         print("\nATM Menu:")
145         print("1. Check Balance")
146         print("2. Deposit Money")
147         print("3. Withdraw Money")
148         print("4. Exit")
149         choice = input("Select an option (1-4): ")
150         if choice == '1':
151             print(f"Your balance is: ${balance}")
152         elif choice == '2':
153             amount = float(input("Enter amount to deposit: "))
154             balance += amount
155             print(f"${amount} deposited. New balance: ${balance}")
156         elif choice == '3':
157             amount = float(input("Enter amount to withdraw: "))
158             if amount > balance:
159                 print("Insufficient funds.")
160             else:
161                 balance -= amount
162                 print(f"${amount} withdrawn. New balance: ${balance}")
163         elif choice == '4':
164             print("Thank you for using the ATM. Goodbye!")
165             break
166         else:
167             print("Invalid option. Please try again.")
168 # Example usage
169 atm_menu()
```

OUTPUT:



```
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!
```

Explanation:

- Correct balance update
- Prevents overdraft
- Loop exits safely