

AI ASSISTANT CODEING

Assignment 6.5

Name: S Srinidhi

HT NO:2303A51342

BATCH:20

Experiment 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.

#Code & output

The screenshot shows a code editor interface with a dark theme. At the top, there's a navigation bar with 'View', 'Go', 'Run', 'Terminal', and 'Help' options. A search bar contains the text 'Q AI Assistant'. Below the search bar, there are tabs for 'Assignment1.py', 'Assignment 1.5.py', and 'Assignment6.5.py'. The 'Assignment6.5.py' tab is active, showing the following Python code:

```
1 def check_voting_eligibility(age, is_citizen):
2     """
3         Check if a person is eligible to vote based on age and citizenship.
4
5         Args:
6             age (int): The person's age in years
7             is_citizen (bool): Whether the person is a citizen
8
9         Returns:
10            bool: True if eligible to vote, False otherwise
11        """
12        if age >= 18 and is_citizen:
13            return True
14        return False
15
16    # Example usage
17    if __name__ == "__main__":
18        # Test cases
19        print(check_voting_eligibility(25, True)) # True
20        print(check_voting_eligibility(17, True)) # False
21        print(check_voting_eligibility(20, False)) # False
22        print(check_voting_eligibility(21, True)) # True
```

At the bottom of the code editor, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is selected, showing the command line output:

```
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> & C:\Users\SRINIDHI\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/SRINIDHI/OneDrive/Desktop/AI Assistant/Assignment4.5.py"
True
False
False
True
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant>
```

On the right side of the terminal, there are two PowerShell windows listed: 'powershell' and 'powershell'.

The screenshot shows a code editor interface with several tabs open. The main tab displays a Python script named `Assignment6.5.py` containing a function `check_voting_eligibility`. The code uses conditional statements (`if`, `elif`, `else`) to determine if a person is eligible to vote based on their age and citizenship status. The output terminal shows the execution of the script with three test cases: (25, True), (16, True), and (20, False), producing the expected messages for each case.

```

def check_voting_eligibility(age, is_citizen):
    """
    Returns:
        str: Eligibility message
    """
    if age < 18:
        return "Not eligible to vote: Must be at least 18 years old."
    elif not is_citizen:
        return "Not eligible to vote: Must be a citizen."
    else:
        return "Eligible to vote!"

# Example usage
if __name__ == "__main__":
    # Test cases
    print(check_voting_eligibility(25, True))  # Eligible to vote!
    print(check_voting_eligibility(16, True))  # Not eligible to vote: Must be at least 18 years old.
    print(check_voting_eligibility(20, False))  # Not eligible to vote: Must be a citizen.

```

Explanation of Conditions

- The function `check_voting_eligibility()` accepts age and citizenship status as inputs.
- The first if condition checks whether the person is below 18 years, which is the minimum voting age.
- The elif condition checks whether the person is not a citizen.
- If both conditions are satisfied ($\text{age} \geq 18$ and `citizen` is `True`), the person is eligible to vote.
- The function returns clear, descriptive messages for each case.

Task 1: Observation (Voting Eligibility Check)

- The AI-generated code correctly uses **conditional statements** to check age and citizenship.
- Eligibility is accurately determined based on the given conditions.
- The program produces clear and correct output messages for all test cases.
- Logical flow of conditions avoids unnecessary checks.

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.”

#code&output

The screenshot shows a code editor interface with several tabs at the top: 'Welcome', 'Assignment1.py', 'Assignment 1.5.py', and 'Assignment6.5.py' (which is the active tab). The code in 'Assignment6.5.py' is as follows:

```
Assignment6.5.py > ...
1 def count_vowels_consonants(text):
2     vowels = "aeiouAEIOU"
3     vowel_count = 0
4     consonant_count = 0
5
6     for char in text:
7         if char.isalpha():
8             if char in vowels:
9                 vowel_count += 1
10            else:
11                consonant_count += 1
12
13     return vowel_count, consonant_count
14
15 # Example usage
16 text = input("Enter a string: ")
17 vowels, consonants = count_vowels_consonants(text)
18 print(f"Vowels: {vowels}")
19 print(f"Consonants: {consonants}")
```

Below the code editor is a terminal window showing the execution of the script and its output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> & C:\Users\SRINIDHI\AppData\Local\Programs\Python\Python313\py
I Assistant\Assignment6.5.py
Enter a string: abcdefghhi
Vowels: 3
Consonants: 7
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant>
```

Explanation

- The function `count_vowels_consonants()` processes the input string character by character using a `for` loop.
- `isalpha()` ensures only alphabetic characters are counted.
- Characters found in the `vowels` string are counted as vowels.
- Remaining alphabetic characters are counted as consonants.
- The function returns both counts.

Task 2: Observation (Vowel and Consonant Counting)

- The AI-generated code efficiently processes the string using a **loop**.
- Only alphabetic characters are considered for counting.
- Vowels and consonants are counted correctly.
- The output matches the expected results for different inputs.

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.”

#code&output

The screenshot shows a code editor interface with a dark theme. At the top, there are tabs for 'Welcome', 'Assignment1.py', 'Assignment 1.5.py', and 'Assignment6.5.py' (which is the active tab). Below the tabs, the code for 'Assignment6.5.py' is displayed. The code defines two classes: 'Book' and 'Library'. The 'Book' class has an __init__ method that initializes title, author, isbn, and available status. The 'Library' class has methods for adding a book to a list and removing a book by its isbn. The code uses f-strings for printing messages. At the bottom of the code editor, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is underlined), and 'PORTS'. The 'TERMINAL' tab shows command-line output related to the library management system.

```
... 48     class Book:
49         def __init__(self, title, author, isbn, available=True):
50             self.title = title
51             self.author = author
52             self.isbn = isbn
53             self.available = available
54
55
56     class Library:
57         def __init__(self):
58             self.books = []
59
60         def add_book(self, book):
61             self.books.append(book)
62             print(f"Book '{book.title}' added to library.")
63
64         def remove_book(self, isbn):
65             for book in self.books:
66                 if book.isbn == isbn:
67                     self.books.remove(book)
68                     print(f"Book removed from library.")
69                     return
70             print("Book not found.")
```

TERMINAL

```
- The Great Gatsby by F. Scott Fitzgerald (ISBN: 978-0-7432-7356-5)
You borrowed '1984'.

Available Books:
- To Kill a Mockingbird by Harper Lee (ISBN: 978-0-06-112008-4)
- The Great Gatsby by F. Scott Fitzgerald (ISBN: 978-0-7432-7356-5)
You returned '1984'.

Available Books:
- 1984 by George Orwell (ISBN: 978-0-451-52494-2)
- To Kill a Mockingbird by Harper Lee (ISBN: 978-0-06-112008-4)
- The Great Gatsby by F. Scott Fitzgerald (ISBN: 978-0-7432-7356-5)
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant>
```

Review of AI Suggestions Quality

- The AI-generated code correctly uses classes (Book, Library) to represent real-world entities.
- Loops are used to search and display books.
- Conditional statements handle borrowing, returning, and availability checks.
- Code is modular, readable, and easy to extend.
- Logic is efficient and produces correct results.

Reflection on AI-Assisted Coding Experience

- AI-assisted code completion significantly reduced development time by providing a structured and logical solution. However, reviewing and understanding the generated code was essential to ensure correctness and improve readability. This experiment demonstrates that AI is a powerful coding assistant when used responsibly and ethically.

Task 3: Observation (Library Management System)

- The AI-generated program effectively uses **classes, loops, and conditionals**.
- Book management operations such as add, borrow, return, and display work correctly.
- The code structure is modular and easy to understand.
- The program produces correct results for all test scenarios.

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.”

#code & output

```
Assignment6.5.py > ...
119     class StudentAttendance:
120         def display_all_attendance(self):
121             self.display_attendance(student_name)
122
123         def get_attendance_percentage(self, student_name):
124             if student_name in self.attendance:
125                 records = self.attendance[student_name]
126                 if len(records) == 0:
127                     return 0
128                 present_days = sum(1 for r in records if r["present"])
129                 percentage = (present_days / len(records)) * 100
130                 print(f"{student_name}'s attendance: {percentage:.2f}%")
131                 return percentage
132             else:
133                 print(f"Student '{student_name}' not found.")
134
135
136     # Example usage
137     if __name__ == "__main__":
138         attendance = StudentAttendance()
139         attendance.add_student("Alice")
140         attendance.add_student("Bob")
141         attendance.add_student("Charlie")
142
143         attendance.mark_attendance("Alice", "2024-01-15", True)
144         attendance.mark_attendance("Alice", "2024-01-16", True)
145         attendance.mark_attendance("Alice", "2024-01-17", False)
146         attendance.mark_attendance("Bob", "2024-01-15", True)
147         attendance.mark_attendance("Bob", "2024-01-16", False)
148
149         attendance.display_all_attendance()
150         attendance.get_attendance_percentage("Alice")
151         attendance.get_attendance_percentage("Bob")
```

```
... PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> & C:\Users\SRINIDHI\AppData\Local\Programs\Python\Python313\python.exe "I Assistant\Assignment6.5.py"
Student 'Alice' added.
Student 'Bob' added.
Student 'Charlie' added.
'Alice' marked present on 2024-01-15.
'Alice' marked present on 2024-01-16.
'Alice' marked absent on 2024-01-17.
'Bob' marked present on 2024-01-15.
'Bob' marked absent on 2024-01-16.

All Student Attendance:

Attendance for Alice:
2024-01-15: Present
2024-01-16: Present
2024-01-17: Absent

Attendance for Bob:
2024-01-15: Present
2024-01-16: Absent

Attendance for Charlie:
Alice's attendance: 66.67%
Bob's attendance: 50.00%
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant>
```

Explanation

- The Student Attendance class stores attendance using a dictionary.
- Each student's attendance is maintained as a list of daily records.
- Loops are used to iterate over attendance records and students.
- Conditional statements validate student existence and attendance status.
- Attendance percentage is calculated using logical conditions and loops.

Task 4: Observation (Class-Based Attendance System)

- The attendance system accurately stores and displays student attendance.
- Loops are used to iterate through attendance records efficiently.
- Conditional checks prevent invalid operations.
- Attendance percentage is calculated correctly.

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.”

#code&output

```
82     account_balance = 1000
83
84     while True:
85         print("\nATM Menu:")
86         print("1. Deposit")
87         print("2. Withdraw")
88         print("3. Check Balance")
89         print("4. Exit")
90
91         choice = input("Enter your choice (1-4): ")
92
93         if choice == '1':
94             try:
95                 deposit_amount = float(input("Enter deposit amount: "))
96                 if deposit_amount > 0:
97                     account_balance += deposit_amount
98                     print(f"Deposit successful. New balance: ${account_balance:.2f}")
99                 else:
00                     print("Deposit amount must be positive.")
01             except ValueError:
02                 print("Invalid input. Please enter a number.")
03
04         elif choice == '2':
05             try:
06                 withdraw_amount = float(input("Enter withdrawal amount: "))
07                 if withdraw_amount > 0:
08                     if account_balance >= withdraw_amount:
09                         account_balance -= withdraw_amount
10                         print(f"Withdrawal successful. New balance: ${account_balance:.2f}")
11                     else:
12                         print("Insufficient balance.")
13                 else:
14                     print("Withdrawal amount must be positive.")
15             except ValueError:
16                 print("Invalid input. Please enter a number.")
17
18         else:
19             print("Invalid choice. Please enter 1, 2, 3, or 4.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant> & C:\Users\SRINIDHI\AppData\Local\Programs\Python\Python3.9\python "C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant\Assignment6.5.py"

ATM Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 1
Enter deposit amount: 3000
Deposit successful. New balance: $4000.00

ATM Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 2
Enter withdrawal amount: 2000
Withdrawal successful. New balance: $2000.00

ATM Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 3
Current balance: $2000.00

ATM Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice (1-4): 4
Thank you for using the ATM. Goodbye!
PS C:\Users\SRINIDHI\OneDrive\Desktop\AI Assistant>
```

Explanation

- A while loop continuously displays the ATM menu until the user exits.
- Conditional statements (if, elif, else) handle menu options.
- Input validation is performed using try-except blocks.
- Balance updates are done securely with proper checks.

Task 5: Observation (ATM Menu Navigation)

- The AI-generated ATM menu works correctly using **loops and conditionals**.
- All menu options are handled properly with input validation.
- The program prevents invalid and unsafe operations.
- Output is clear and user-friendly.