

AI ASSISTED -CODING

ASSIGNMENT-6.5

RUDROJU RUPA SRI

2303A51918

BATCH-30

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

LO1. Use AI-based code completion tools to generate Python code involving classes, loops, and conditionals.

LO2. Interpret and explain AI-generated code line-by-line.

LO3. Identify errors, inefficiencies, or logical flaws in AI-suggested implementations.

LO4. Optimize AI-generated code for better readability and performance.

LO5. Demonstrate ethical and responsible use of AI tools in coding tasks.

Task Description #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.

```
... ASS-6.5.py X Untitled-1
ASS-6.5.py > ...
1 # Take age and citizenship as input
2 # Use conditional statements to check voting eligibility
3 # Eligible if age >= 18 and citizen is True
4 # Print eligibility result with proper message
5
6 age = int(input("Enter your age: "))
7 citizen = input("Are you a citizen? (yes/no): ").lower() == "yes"
8
9 if age >= 18 and citizen:
10     print("You are eligible to vote.")
11 else:
12     print("You are not eligible to vote.")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell + ⌂ ⌂ ... | ⌂ ×
● PS C:\Users\HP\Desktop\AI> & "C:\Program Files\Python313\python.exe" c:/Users/HP/Desktop/AI/ASS-6.5.py
Enter your age: 21
Are you a citizen? (yes/no): yes
You are eligible to vote.
○ PS C:\Users\HP\Desktop\AI> ■

Ln 10, Col 39 Spaces: 4 CRLF { } Python Python 3.13 (64-bit) (i) Go Live 🔍
```

Task Description #2(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.

The screenshot shows a code editor interface with two tabs: 'ASS-6.5.py' and 'Untitled-1'. The 'ASS-6.5.py' tab contains the following Python code:

```
13
14 # Input a string from user
15 # Use a loop to iterate through characters
16 # Count vowels and consonants separately
17 # Ignore spaces and special characters
18 # Print total vowel and consonant count
19 input_string = input("Enter a string: ")
20 vowels = "aeiouAEIOU"
21 vowel_count = 0
22 consonant_count = 0
23
24 for char in input_string:
25     if char.isalpha(): # Check if character is a letter
26         if char in vowels:
27             vowel_count += 1
28         else:
29             consonant_count += 1
30
31 print(f"Total vowels: {vowel_count}")
32 print(f"Total consonants: {consonant_count}")
```

The terminal below the code editor shows the output of running the script 'ASS-6.5.py' with the input 'Rupasri'. The output is:

```
PS C:\Users\HP\Desktop\AI> & "C:\Program Files\Python313\python.exe" c:/Users/HP/Desktop/AI/ASS-6.5.py
Enter a string: Rupasri
Total vowels: 3
Total consonants: 4
PS C:\Users\HP\Desktop\AI>
```

Task Description #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.

The screenshot shows a Python script named `library_management_system.py` in a code editor. The code defines a `Book` class with an `__init__` method that sets `title` and `author`. It also defines a `Library` class with an `__init__` method that initializes an empty list of books, an `add_book` method to append books to the list, and a `display_books` method to print each book's title and author. The script then creates a `Library` instance, adds three books to it, and displays them. The terminal below shows the output of the script.

```
37
38 #Task:3
39 def library_management_system():
40     #using classes,loops, conditional statements, functions, and data structures
41     class Book:
42         def __init__(self, title, author):
43             self.title = title
44             self.author = author
45     class Library:
46         def __init__(self):
47             self.books = []
48         def add_book(self, book):
49             self.books.append(book)
50         def display_books(self):
51             for book in self.books:
52                 print(f"Title: {book.title}, Author: {book.author}")
53     # Create a library instance
54     library = Library()
55     # Add books to the library
56     library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
57     library.add_book(Book("1984", "George Orwell"))
58     library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald"))
59     # Display the books in the library
60     library.display_books()
61 if __name__ == "__main__":
62     library_management_system()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Title: To Kill a Mockingbird, Author: Harper Lee
Title: 1984, Author: George Orwell
Title: The Great Gatsby, Author: F. Scott Fitzgerald
PS C:\Users\HP\Desktop\AI> █

Ln 38, Col 8 Spaces:4 UTF-8 CRLF () Python Python 3.13 (64-bit) Go Live

Task Description #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.

```
ASS-6.5.py > ...
66 # Student Attendance System
67 # Create a simple Attendance class
68 # Use loop to input student names and P/A status
69 # Store attendance in a dictionary
70 # Display present and absent students
71 class Attendance:
72     def __init__(self):
73         self.attendance_record = {}
74     def mark_attendance(self, name, status):
75         self.attendance_record[name] = status
76     def display_attendance(self):
77         present_students = [name for name, status in self.attendance_record.items() if status == 'P']
78         absent_students = [name for name, status in self.attendance_record.items() if status == 'A']
79         print(f"Present Students: {', '.join(present_students)}")
80         print(f"Absent Students: {', '.join(absent_students)}")
81 # Example usage
82 if __name__ == "__main__":
83     attendance = Attendance()
84     num_students = int(input("Enter the number of students: "))
85     for _ in range(num_students):
86         name = input("Enter student name: ")
87         status = input("Enter attendance status (P/A): ")
88         attendance.mark_attendance(name, status)
89     attendance.display_attendance()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\HP\Desktop\AI> & "C:\Program Files\Python313\python.exe" c:/Users/HP/Desktop/AI/ASS-6.5.py
Enter student name: rupa
Enter attendance status (P/A): A
Enter attendance status (P/A): A
Enter student name: spurthi
Enter attendance status (P/A): P
Enter student name: hruthika
Enter attendance status (P/A): P
Present Students: spurthi, hruthika
Absent Students: rupa
```

Task Description #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.

```
ASS-6.5.py X Untitled-1
ASS-6.5.py > generate_atm_menu > ATM > display_menu

#Task:5
def generate_atm_menu():
    """
    Function to request AI to generate a Python program that simulates an ATM menu.

    Returns:
    AI-generated code for an ATM menu with loops and conditionals.
    """

    class ATM:
        def __init__(self, balance=0):
            self.balance = balance

        def display_menu(self):
            print("ATM Menu:")
            print("1. Check Balance")
            print("2. Deposit")
            print("3. Withdraw")
            print("4. Exit")

        def check_balance(self):
            print(f"Your current balance is: ${self.balance}")

        def deposit(self, amount):
            self.balance += amount
            print(f"${amount} deposited successfully.")

        def withdraw(self, amount):
            if amount > self.balance:
                print("Insufficient funds.")
            else:
                self.balance -= amount
                print(f"${amount} withdrawn successfully.")

    # Example usage
    if __name__ == "__main__":
        atm = ATM(1000) # Initial balance of $1000
        while True:
            atm.display_menu()
            choice = input("Enter your choice: ")
            if choice == '1':
                atm.check_balance()
            elif choice == '2':
                amount = float(input("Enter amount to deposit: "))
                atm.deposit(amount)
            elif choice == '3':
                amount = float(input("Enter amount to withdraw: "))
                atm.withdraw(amount)
            elif choice == '4':
                print("Thank you for using the ATM. Goodbye!")
                break
            else:
                print("Invalid choice. Please try again.")

    generate_atm_menu()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
ASS-6.5.py X Untitled-1
ASS-6.5.py > generate_atm_menu > ATM > display_menu

def generate_atm_menu():
    class ATM:
        def withdraw(self, amount):
            if amount > self.balance:
                print("Insufficient funds.")
            else:
                self.balance -= amount
                print(f"${amount} withdrawn successfully.")

    # Example usage
    if __name__ == "__main__":
        atm = ATM(1000) # Initial balance of $1000
        while True:
            atm.display_menu()
            choice = input("Enter your choice: ")
            if choice == '1':
                atm.check_balance()
            elif choice == '2':
                amount = float(input("Enter amount to deposit: "))
                atm.deposit(amount)
            elif choice == '3':
                amount = float(input("Enter amount to withdraw: "))
                atm.withdraw(amount)
            elif choice == '4':
                print("Thank you for using the ATM. Goodbye!")
                break
            else:
                print("Invalid choice. Please try again.")

    generate_atm_menu()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
PS C:\Users\HP\Desktop\AI> & "C:\Program Files\Python313\python.exe" c:/Users/HP/Desktop/AI/ASS-6.5.py
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 1
Your current balance is: $1000
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 2
Enter amount to deposit: 500
$500.0 deposited successfully.
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 3
Enter amount to withdraw: 200
$200.0 withdrawn successfully.
$500.0 deposited successfully.
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 3
Enter amount to withdraw: 200
$200.0 withdrawn successfully.
4. Exit
Enter your choice: 3
Enter amount to withdraw: 200
$200.0 withdrawn successfully.
PS C:\Users\HP\Desktop\AI>
```

```
PS C:\Users\HP\Desktop\AI> & "C:\Program Files\Python313\python.exe" c:/Users/HP/Desktop/AI/ASS-6.5.py
4. Exit
Enter your choice: 3
Enter amount to withdraw: 200
$200.0 withdrawn successfully.
$500.0 deposited successfully.
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 3
Enter amount to withdraw: 200
$200.0 withdrawn successfully.
4. Exit
Enter your choice: 3
Enter amount to withdraw: 200
$200.0 withdrawn successfully.
Enter your choice: 3
Enter amount to withdraw: 200
$200.0 withdrawn successfully.
Enter your choice: 3
Enter amount to withdraw: 200
$200.0 withdrawn successfully.
Enter your choice: 4
Thank you for using the ATM. Goodbye!
PS C:\Users\HP\Desktop\AI>
```