

AI Assisted Coding

Assignment 6.5

Name: V.Harivamsh

Hall ticket no: 2303A51266

Batch no: 19

Task 1:

Prompt:

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

Code & Output:

The screenshot shows a code editor interface with a terminal window below it. The code editor has a file named 'assignment_6.5.py' open, containing the following Python script:

```
# Task 1: AI-Based Code Completion for Conditional Eligibility Check
# 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.")
```

The terminal window below shows the execution of the script. It prompts for age and citizenship, and then prints the result based on the input. The terminal output is as follows:

```
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hariv/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_6.5.py"
● Enter your age: 18
Are you a citizen? (yes/no): no
You are not eligible to vote.
● PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hariv/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_6.5.py"
Enter your age: 18
Are you a citizen? (yes/no): yes
You are eligible to vote.
○ PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> []
```

Explanation:

The AI-generated code applies conditional statements to determine voting eligibility. It checks if the person's age is 18 or above and confirms their citizenship status. Only when both conditions are satisfied is the user considered eligible. This shows an accurate use of conditional logic produced through AI-based code generation.

Task 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 window with a dark theme. At the top, there's a tab labeled 'assignment_6.5.py X'. Below the tabs, there's a toolbar with icons for 'PROBLEMS', 'OUTPUT', 'TERMINAL' (which is selected), and 'PORTS'. The main area contains the following Python code:

```
1 # Task 2: AI-Based Code Completion for Loop-Based String Processing
2 # Generate Python code to count vowels and consonants in a string using a loop
3
4 input_string = input("Enter a string: ")
5 vowel_count = 0
6 consonant_count = 0
7 vowels = "aeiouAEIOU"
8
9 for char in input_string:
10     if char.isalpha():
11         if char in vowels:
12             vowel_count += 1
13         else:
14             consonant_count += 1
15
16 print(f"Vowels: {vowel_count}, Consonants: {consonant_count}")
17
```

Below the code editor, there's a terminal window titled 'TERMINAL'. It shows the command 'python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_6.5.py"' and its output: 'Enter a string: Hello World' followed by 'Vowels: 3, Consonants: 7'. The terminal window has a dropdown menu icon and a close button.

Explanation:

The AI-generated code handles the input string by looping through each character one by one. It checks whether each character is a vowel or a consonant and counts only the alphabetic characters, ignoring any non-letter symbols. The final output confirms that the logic functions correctly as intended.

Task 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:

```
assignment_6.5.py X
assignment_6.5.py > Library
1 # Task 3: AI-Assisted Code Completion Reflection Task
2 # Library Management System using Classes, Loops, and Conditional Statements
3
4 class Book:
5     def __init__(self, title, author):
6         self.title = title
7         self.author = author
8         self.is_borrowed = False
9
10    def borrow(self):
11        if not self.is_borrowed:
12            self.is_borrowed = True
13            return True
14        return False
15
16    def return_book(self):
17        if self.is_borrowed:
18            self.is_borrowed = False
19            return True
20        return False
21
```



```
assignment_6.5.py X
assignment_6.5.py > Library
21
22
23 class Library:
24     def __init__(self):
25         self.books = []
26
27     def add_book(self, book):
28         self.books.append(book)
29
30     def display_books(self):
31         if not self.books:
32             print("No books available.")
33             return
34
35         for idx, book in enumerate(self.books):
36             status = "Borrowed" | (variable) book: Any | e "Available"
37             print(f"{idx + 1}. {book.title} by {book.author} - {status}")
38
39     def borrow_book(self, index):
40         if 0 <= index < len(self.books):
41             if self.books[index].borrow():
42                 print(f"You have borrowed '{self.books[index].title}'")
43             else:
44                 print(f"'{self.books[index].title}' is already borrowed.")
45         else:
46             print("Invalid book index.")
47
48     def return_book(self, index):
49         if 0 <= index < len(self.books):
50             if self.books[index].return_book():
51                 print(f"You have returned '{self.books[index].title}'")
52             else:
53                 print(f"'{self.books[index].title}' was not borrowed.")
54         else:
55             print("Invalid book index.")
```

The image shows a code editor interface with two tabs open, both titled "assignment_6.5.py".

Code Editor Content:

```
assignment_6.5.py X
assignment_6.5.py > Library
1/ 58 def main():
59     library = Library()
60
61     # Adding sample books
62     library.add_book(Book("1984", "George Orwell"))
63     library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
64     library.add_book(Book("The Great Gatsby", "F. Scott Fitzgerald"))
65
66     while True:
67         print("\nLibrary Menu:")
68         print("1. Display Books")
69         print("2. Borrow Book")
70         print("3. Return Book")
71         print("4. Exit")
72
73         choice = input("Enter your choice: ")
74
75         if choice == "1":
76             library.display_books()
77
78         elif choice == "2":
79             index = int(input("Enter the book number to borrow: ")) - 1
80             library.borrow_book(index)
81
82         elif choice == "3":
83             index = int(input("Enter the book number to return: ")) - 1
84             library.return_book(index)
85
86         elif choice == "4":
87             print("Exiting the library system.")
88             break
89
90         else:
91             print("Invalid choice. Please try again.")
92
93
94     if __name__ == "__main__":
95         main()
```

Terminal Output:

```
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hariv/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_6.5.py"

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 2
Enter the book number to borrow: 3
You have borrowed 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Borrowed

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 3
Enter the book number to return: 3
You have returned 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
```

```
assignment_6.5.py >
assignment_6.5.py > Library
PROBLEMS OUTPUT TERMINAL PORTS
> < TERMINAL
PS C:\Users\hari\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hari/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/hari/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_6.5.py"
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Borrowed

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 3
Enter the book number to return: 3
You have returned 'The Great Gatsby'.

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 1
1. 1984 by George Orwell - Available
2. To Kill a Mockingbird by Harper Lee - Available
3. The Great Gatsby by F. Scott Fitzgerald - Available

Library Menu:
1. Display Books
2. Borrow Book
3. Return Book
4. Exit
Enter your choice: 4
Exiting the library system.
PS C:\Users\hari\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

Explanation:

The AI-generated program models a library using a class and incorporates loops and conditional statements to create a menu-driven interface. The loop enables the program to repeatedly accept user input, while the conditional logic manages the flow of operations. Overall, the program clearly demonstrates the effective use of object-oriented programming concepts with AI assistance.

Reflection on AI-Assisted Coding:

The AI tool was able to generate a complete and working program in a short time. Although the logic is correct, there is still room for improvement by adding input validation and more advanced features. This task highlights how AI can accelerate development, while human review and refinement are still important for achieving better quality.

Task 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:

```
assignment_6.5.py X
assignment_6.5.py > ...
1 # Task 4: AI-Assisted Code Completion for Class-Based Attendance System
2 # Generate a Python class to mark and display student attendance using loops
3 class AttendanceSystem:
4     def __init__(self):
5         self.attendance = {}
6     def mark_attendance(self, student_name):
7         self.attendance[student_name] = "Present"
8     def display_attendance(self):
9         print("\nAttendance Record:")
10        if not self.attendance:
11            print("No attendance marked yet.")
12            return
13        for student, status in self.attendance.items():
14            print(f"{student}: {status}")
15 def main():
16     attendance_system = AttendanceSystem()
17     while True:
18         name = input("Enter student name to mark attendance (or 'exit' to finish): ")
19         if name.lower() == "exit":
20             break
21         attendance_system.mark_attendance(name)
22         attendance_system.display_attendance()
23 if __name__ == "__main__":
24     main()

PROBLEMS OUTPUT TERMINAL PORTS ... | X
> ▾ TERMINAL
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hariv/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_6.5.py"
● Enter student name to mark attendance (or 'exit' to finish): koushik
Enter student name to mark attendance (or 'exit' to finish): harivamsh
Enter student name to mark attendance (or 'exit' to finish): exit

Attendance Record:
koushik: Present
harivamsh: Present
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> []
```

Explanation:

The AI-generated attendance system uses a class to manage and store attendance information. It includes a loop to collect details for multiple students and another loop to display the recorded attendance. The program functions correctly and clearly demonstrates the use of class-based code generated with AI assistance.

Task 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:

```
assignment_6.5.py X
assignment_6.5.py > ...
1 # Task 5: AI-Based Code Completion for Conditional Menu Navigation
2 # Generate a Python program using loops and conditionals to simulate an ATM menu
3 balance = 1000.0
4 while True:
5     print("\nATM Menu:")
6     print("1. Check Balance")
7     print("2. Deposit Money")
8     print("3. Withdraw Money")
9     print("4. Exit")
10    choice = input("Enter your choice: ")
11    if choice == "1":
12        print(f"Your current balance is: ${balance:.2f}")
13    elif choice == "2":
14        amount = float(input("Enter amount to deposit: "))
15        if amount > 0:
16            balance += amount
17            print(f"${amount:.2f} deposited successfully.")
18            print(f"Your current balance is: ${balance:.2f}")
19        else:
20            print("Invalid amount. Please enter a positive value.")
21    elif choice == "3":
22        amount = float(input("Enter amount to withdraw: "))
23        if amount > 0 and amount <= balance:
24            balance -= amount
25            print(f"${amount:.2f} withdrawn successfully.")
26            print(f"Your current balance is: ${balance:.2f}")
27        else:
28            print("Invalid amount or insufficient balance.")
29    elif choice == "4":
30        print("Exiting the ATM. Thank you!")
31        break
32    else:
33        print("Invalid choice. Please try again.")
```

```
assignment_6.5.py X
assignment_6.5.py > ...
1 # Task 5: AI-Based Code Completion for Conditional Menu Navigation
2 # Generate a Python program using loops and conditionals to simulate an ATM menu
3 balance = 1000.0
4
5 PROBLEMS OUTPUT TERMINAL PORTS
6 > < TERMINAL
7 PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hariv/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_6.5.py"
8
9
10 ATM Menu:
11 1. Check Balance
12 2. Deposit Money
13 3. Withdraw Money
14 4. Exit
15 Enter your choice: 1
16 Your current balance is: $1000.00
17
18 ATM Menu:
19 1. Check Balance
20 2. Deposit Money
21 3. Withdraw Money
22 4. Exit
23 Enter your choice: 2
24 Enter amount to deposit: 3
25 $3.00 deposited successfully.
26 Your current balance is: $1003.00
27
28 ATM Menu:
29 1. Check Balance
30 2. Deposit Money
31 3. Withdraw Money
32 4. Exit
33 Enter your choice: 4
34 Exiting the ATM. Thank you!
35
36 PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

Explanation:

The AI-generated ATM program uses a loop to repeatedly show the menu and applies conditional statements to process the user's choices. The logic updates the account balance correctly and blocks withdrawals when they are not valid. Overall, this task shows how AI-based code completion can be effectively used to build menu-driven programs.

Final Conclusion:

This experiment demonstrates that AI-based code completion tools are capable of generating practical Python programs that make use of classes, loops, and conditional statements. Although AI significantly accelerates the development process, it is still the responsibility of developers to review the logic, address edge cases, and ensure that the generated code is used ethically and responsibly.