

Lab Assignment – 6.5

Hall Ticket No.: 2303A510E6

Name – Anushka Boora

Batch – 29

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.

Code:

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

```
def check_voting_eligibility(age, citizenship):
```

```
    if age >= 18 and citizenship.lower() == "yes":
```

```
        return "You are eligible to vote."
```

```
    elif age < 18:
```

```
        return "You are not eligible to vote due to age."
```

```
    else:
```

```
        return "You are not eligible to vote due to citizenship."
```

```
# Example usage
```

```
age = int(input("Enter your age: "))
```

```
citizenship = input("Are you a citizen? (yes/no): ")
```

```

eligibility_message = check_voting_eligibility(age, citizenship)
print(eligibility_message)

```

Output:

Enter your age: 21

Are you a citizen? (yes/no): yes

You are eligible to vote.

The screenshot shows the Visual Studio Code interface with the following details:

- File Structure:** A sidebar on the left shows a folder structure with a file named "aac65.py".
- Code Editor:** The main editor area contains two snippets of Python code. The first snippet checks voting eligibility based on age and citizenship. The second snippet generates Python code to count vowels and consonants in a string using a loop.
- Terminal:** The terminal at the bottom shows the execution of the code. It prompts for age (21), citizenship (yes), and a string ("varadha"). It then prints the number of vowels (3) and consonants (4).
- Output:** The output pane shows the results of the code execution.
- Breakpoints:** The breakpoints pane indicates that an exception has occurred.

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.

Code:

```
#Generate Python code to count vowels and consonants in a string using a loop.

def count_vowels_consonants(input_string):

    vowels = "aeiouAEIOU"

    vowel_count = 0

    consonant_count = 0


    for char in input_string:

        if char.isalpha(): # Check if the character is an alphabet

            if char in vowels:

                vowel_count += 1

            else:

                consonant_count += 1


    return vowel_count, consonant_count

# Example usage

input_string = input("Enter a string: ")

vowels, consonants = count_vowels_consonants(input_string)

print(f"Number of vowels: {vowels}")

print(f"Number of consonants: {consonants}")
```

Output:

Enter a string: varadha

Number of vowels: 3

Number of consonants: 4

```

File Edit Selection View ...
RUN AND DEBUG ... Welcome aac65.py
RUN To customize Run and Debug, open a folder and create a launch.json file.
Debug using a terminal command or in an interactive chat.
Show automatic Python configurations

C:\Users\boora\Downloads\3-2\AI-AC> aac65.py >
18 def count_vowels_consonants(input_string):
19     vowels = "aeiouAEIOU"
20     vowel_count = 0
21     consonant_count = 0
22
23     for char in input_string:
24         if char.isalpha(): # Check if the character is an alphabet
25             if char in vowels:
26                 vowel_count += 1
27             else:
28                 consonant_count += 1
29
30     return vowel_count, consonant_count
31
32 # Example usage
33 input_string = input("Enter a string: ")
34 vowels, consonants = count_vowels_consonants(input_string)
35 print(f"Number of vowels: {vowels}")
36 print(f"Number of consonants: {consonants}")

PS C:\Users\boora\Downloads\3-2\AI-AC> c; cd 'c:\Users\boora\Downloads\3-2\AI-AC'; & 'c:\Users\boora\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\boora\.vscode\extensions\ms-vscode.python-language-2023.18.0-win32-x86\bundle\lib\debugpy\launcher' '52813' >> 'c:\Users\boora\Downloads\3-2\AI-AC\aac65.py'
Enter your age: 21
Are you a citizen? (yes/no): yes
You are eligible to vote.
Enter a string: varadha
Number of vowels: 3
Number of consonants: 4

1.Show Books 2.Issue Book 3.Exit
Choice: 2
Book name: Python
Book issued

1.Show Books 2.Issue Book 3.Exit
Choice: 1
Java

1.Show Books 2.Issue Book 3.Exit
Choice: 3
Bye

1.Mark Attendance 2.Display Attendance 3.Exit
Choice: 1
Enter student name: devara

```

Ln 35, Col 45 Spaces:4 UTF-8 [] Python 3.11.5

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.

Code:

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

class Library:

```
def __init__(self):
```

```
self.books = ["Python", "Java"]

lib = Library()

while True:

    print("\n1.Show Books 2.Issue Book 3.Exit")

    ch = input("Choice: ")

    if ch == "1":

        for b in lib.books:

            print(b)

    elif ch == "2":

        book = input("Book name: ")

        if book in lib.books:

            lib.books.remove(book)

            print("Book issued")

        else:

            print("Not available")

    elif ch == "3":

        print("Bye")

        break

    else:

        print("Wrong choice")
```

Output:

1.Show Books 2.Issue Book 3.Exit

Choice: 2

Book name: Python

Book issued

1.Show Books 2.Issue Book 3.Exit

Choice: 1

Java

1.Show Books 2.Issue Book 3.Exit

Choice: 3

Bye

The screenshot shows the PyCharm IDE interface. The top navigation bar includes File, Edit, Selection, View, and a search bar. On the left, there's a sidebar with RUN and BREAKPOINTS sections. The main area displays a Python file named 'aad5.py' containing code for a library management system. Below the code editor is a terminal window showing the execution of the script. The bottom right corner shows a docked Python interpreter window.

```
C:\Users\boon>cd Downloads>3>AI-AC>aad5.py ->
39 #Generate a Python program for a library management system using classes, loops, and conditional statements.
40 class Library:
41     def __init__(self):
42         self.books = ["Python", "Java"]
43 lib = Library()
44 while True:
45     print("1.Show Books 2.Issue Book 3.Exit")
46     ch = input("Choice: ")
47     if ch == "1":
48         for b in lib.books:
49             print(b)
50     elif ch == "2":
51         book = input("Book name: ")
52         if book in lib.books:
53             lib.books.remove(book)
54             print("Book issued")
55         else:
56             print("Not available")
57
58 Enter a string: varada
59 Number of vowels: 3
60 Number of consonants: 4
61
62 1.Show Books 2.Issue Book 3.Exit
63 Choice: 2
64 Book name: Python
65 Book issued
66
67 1.Show Books 2.Issue Book 3.Exit
68 Choice: 1
69 Java
70
71 1.Show Books 2.Issue Book 3.Exit
72 Choice: 3
73 Bye
74
75 1.Mark Attendance 2.Display Attendance 3.Exit
76 Choice: 1
77 Enter student name: devara
78
79 1.Mark Attendance 2.Display Attendance 3.Exit
80 Choice: 2
81 Student Attendance:
82 devara: Present
```

Task Description #4 (AI-Assisted Code Completion for Class-Defining Code)

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.

Code:

```
#Generate a Python class to mark and display student attendance using loops.

class StudentAttendance:

    def __init__(self):
        self.attendance = {}

    def mark_attendance(self, student_name):
        self.attendance[student_name] = "Present"

    def display_attendance(self):
        print("Student Attendance:")
        for student, status in self.attendance.items():
            print(f"{student}: {status}")

# Example usage

attendance = StudentAttendance()

while True:
    print("\n1.Mark Attendance 2.Display Attendance 3.Exit")
    choice = input("Choice: ")

    if choice == "1":
        name = input("Enter student name: ")
        attendance.mark_attendance(name)

    elif choice == "2":
        attendance.display_attendance()

    elif choice == "3":
        print("Exiting...")
        break

    else:
```

```
print("Invalid choice. Please try again.")
```

Output:

1.Mark Attendance 2.Display Attendance 3.Exit

Choice: 1

Enter student name: devara

1.Mark Attendance 2.Display Attendance 3.Exit

Choice: 2

Student Attendance:

devara: Present

1.Mark Attendance 2.Display Attendance 3.Exit

Choice: 1

Enter student name: deva

1.Mark Attendance 2.Display Attendance 3.Exit

Choice: 2

Student Attendance:

devara: Present

deva: Present

1.Mark Attendance 2.Display Attendance 3.Exit

Choice: 1

Enter student name: white

1.Mark Attendance 2.Display Attendance 3.Exit

Choice: 2

Student Attendance:

devara: Present

deva: Present

white: Present

1.Mark Attendance 2.Display Attendance 3.Exit

Choice: 3

Exiting...

```
#Generate a Python class to mark and display student attendance using loops.
class StudentAttendance:
    def __init__(self):
        self.attendance = {}
    def mark_attendance(self, student_name):
        self.attendance[student_name] = "Present"
    def display_attendance(self):
        print("Student Attendance:")
        for student, status in self.attendance.items():
            print(f"{student}: {status}")
# Example usage
attendance = StudentAttendance()
while True:
    print("\n1.Mark Attendance 2.Display Attendance 3.Exit")
    choice = input("Choice: ")
    if choice == "1":
        student_name = input("Enter student name: ")
        attendance.mark_attendance(student_name)
    elif choice == "2":
        attendance.display_attendance()
    elif choice == "3":
        break
    else:
        print("Invalid choice")
```

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.

Code:

```
#Generate a Python program using loops and conditionals to simulate an ATM menu.

bal = 1000

while True:

    ch = input("\n1.Balance 2.Deposit 3.Withdraw 4.Exit : ")

    if ch == "1":

        print("Balance:", bal)

    elif ch == "2":

        bal += int(input("Amount: "))

    elif ch == "3":

        a = int(input("Amount: "))

        if a <= bal: bal -= a

        else: print("No money")

    elif ch == "4":

        break
```

Output:

```
1.Balance 2.Deposit 3.Withdraw 4.Exit : 1
```

```
Balance: 1000
```

```
1.Balance 2.Deposit 3.Withdraw 4.Exit : 2
```

1.Balance 2.Deposit 3.Withdraw 4.Exit : 2

Amount: 10000

1.Balance 2.Deposit 3.Withdraw 4.Exit : 3

Amount: 4000

1.Balance 2.Deposit 3.Withdraw 4.Exit : 4

PS C:\Users\boora\Downloads\3-2\AI-AC>

The screenshot shows a Python code editor interface with a dark theme. The main window displays a script named `aad5.py`. The code implements a simple ATM menu with loops and conditionals. It starts with initializing a balance of 10000, then enters a loop where it prompts the user for a choice (1 for Balance, 2 for Deposit, 3 for Withdrawal, 4 for Exit). Based on the choice, it performs the corresponding operation or prints a message. The terminal below the editor shows the execution of the script, with the user inputting choices 1, 2, 3, and 4, and the program responding with the current balance or a message like "No money".

```
#Generate a Python program using loops and conditionals to simulate an ATM menu.
bal = 10000
while True:
    ch = input("\n1.Balance 2.Deposit 3.Withdraw 4.Exit : ")
    if ch == "1":
        print("Balance:", bal)
    elif ch == "2":
        bal += int(input("Amount: "))
    elif ch == "3":
        a = int(input("Amount: "))
        if a <= bal: bal -= a
        else: print("No money")
    elif ch == "4":
        break
    else:
        print("Invalid choice")
print("Exiting...")
```