

## **ASSIGNMENT -6.5**

**NAME: HABEEBA KHANAM**

**HALLTICKET NO: 2303A51474**

**BATCH:29**

**EXPERIMENT 6: AI-BASED CODE COMPLETION:  
WORKING WITH SUGGESTIONS FOR CLASSES,  
LOOPS, CONDITIONALS**

**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.”**

**CODE:**

The screenshot shows the Visual Studio Code interface. The top part displays a Python script named `AI - 6.5.py` with the following code:

```

1 def check_voting_eligibility(age, citizenship):
2     if age < 0:
3         return "Invalid age"
4     if age >= 18 and citizenship.strip().lower() == "indian":
5         return "Eligible to vote"
6     return "Not eligible to vote"
7
8
9
10 # Taking user input
11 age = int(input("Enter your age: "))
12 citizenship = input("Enter your citizenship: ")
13
14 # Checking eligibility
15 result = check_voting_eligibility(age, citizenship)
16 print(result)
17
18
19

```

The bottom part shows the terminal window with the output of running the script:

```

PS C:\Users\Sameera Khan\Desktop> & 'c:\Users\Sameera Khan\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Sameera Khan\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '65401' '--' 'C:\Users\Sameera Khan\Desktop\AI - 6.5.py'
Enter your age: 18
Enter your citizenship: INDIAN
Eligible to vote
PS C:\Users\Sameera Khan\Desktop> ^C
PS C:\Users\Sameera Khan\Desktop>
PS C:\Users\Sameera Khan\Desktop> cd 'c:\Users\Sameera Khan\OneDrive\Desktop'; & 'c:\Users\Sameera Khan\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Sameera Khan\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61559' '--' 'C:\Users\Sameera Khan\OneDrive\Desktop\AI - 6.5.py'
Enter your age: 25
Enter your citizenship: AMERICAN
Not eligible to vote
PS C:\Users\Sameera Khan\Desktop>

```

## OBSERVATION:

- The program ensures that **both conditions must be satisfied** ( $\text{age} \geq 18$  and valid citizenship) before declaring a user eligible to vote.
- Input handling is simple and user-friendly, allowing easy modification for additional eligibility rules.
- The logic flow is clear, readable, and follows standard Python programming practices.
- AI-assisted code completion significantly **reduces development time** while maintaining logical correctness.

## TASK DESCRIPTION #2

# (AI-Based Code Completion for Loop-Based String Processing)

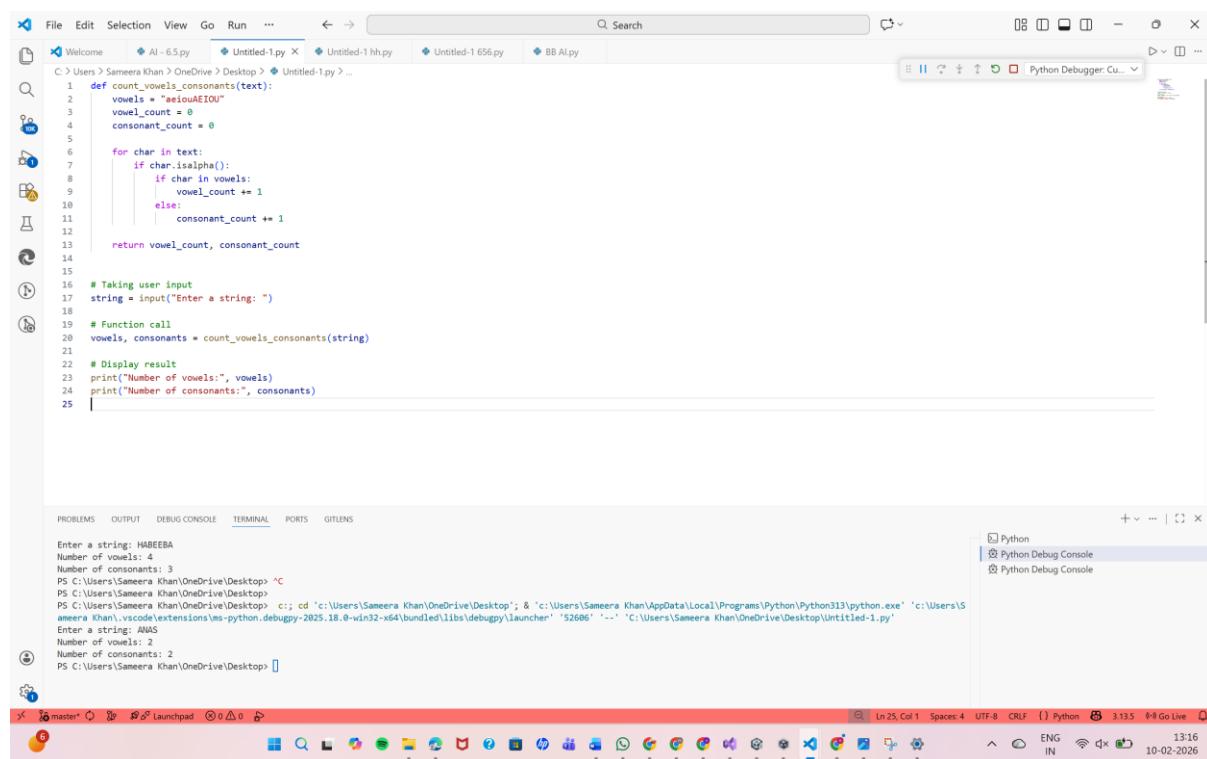
## TASK:

Use an AI tool to process strings using loops.

## PROMPT:

**“Generate Python code to count vowels and consonants in a string using a loop.”**

## CODE:



The screenshot shows a code editor interface with a Python file open. The code defines a function `count_vowels_consonants` that takes a string as input and returns the counts of vowels and consonants. It uses a loop to iterate through each character, checking if it is alphabetic. If it is, it checks if it is a vowel; if so, the vowel count is incremented; if not, the consonant count is incremented. The code then prompts the user for input, calls the function, and prints the results. The code editor has tabs for multiple files, a search bar, and various icons for file operations. Below the code editor is a terminal window showing the execution of the script and its output. The terminal shows the user entering the string "HABEEBA", which is then processed by the script to output the number of vowels (4) and consonants (3). The terminal also shows the command used to run the script from the command line.

```
File Edit Selection View Go Run ... ← → Search
C:\Users\Sameera Khan>OneDrive\Desktop> Untitled-1 hh.py Untitled-1 656.py BB AI.py
Python Debugger: Cu...
File Welcome AI - 6.5.py Untitled-1.py ...
Search
C:\Users\Sameera Khan>OneDrive\Desktop> Untitled-1 hh.py Untitled-1 656.py BB AI.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
16 # Taking user input
17 string = input("Enter a string: ")
18
19 # Function call
20 vowels, consonants = count_vowels_consonants(string)
21
22 # Display result
23 print("Number of vowels:", vowels)
24 print("Number of consonants:", consonants)
25

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
+ v ... | x
Enter a string: HABEEBA
Number of vowels: 4
Number of consonants: 3
PS C:\Users\Sameera Khan>OneDrive\Desktop> <C
PS C:\Users\Sameera Khan>OneDrive\Desktop>
PS C:\Users\Sameera Khan>OneDrive\Desktop> c:; cd 'c:\Users\Sameera Khan\OneDrive\Desktop'; & 'c:\Users\Sameera Khan\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Sameera Khan\vscodeextensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52606' '--' 'C:\Users\Sameera Khan\OneDrive\Desktop\Untitled-1.py'
Enter a string: ANA
Number of vowels: 2
Number of consonants: 2
PS C:\Users\Sameera Khan>OneDrive\Desktop> [

master ⚡ Launchpad ⚡ Col 0 ▲ 0 🔍
In 25, Col 1 Spaces: 4 UTR-8 CRLF Python 3.13.5 Go Live ENG IN 13:16 10-02-2026
```

## OBSERVATION:

- The AI-generated code correctly uses **loops and conditionals** for string processing.
- Non-alphabetic characters are ignored, improving accuracy.
- The logic is efficient, readable, and easy to extend.

## **TASK DESCRIPTION #3**

**(AI-Assisted Code Completion Reflection Task)**

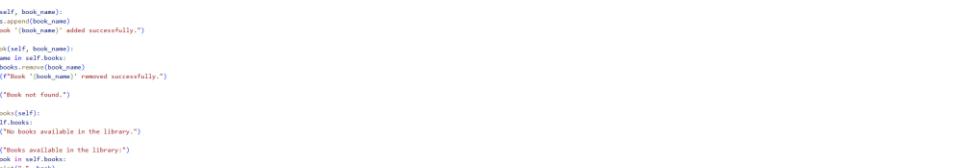
### **TASK:**

**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.”**

### **CODE:**



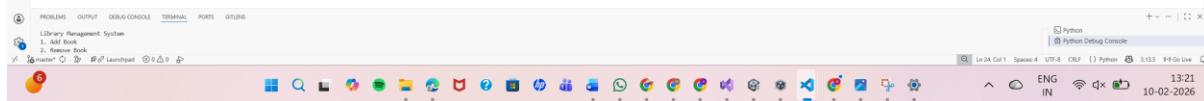
The screenshot shows a Python code editor with the following details:

- Title Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** Search
- File List:** Untitled - 65.py, Untitled - 1.py, Untitled - 1 Mypy X, Untitled - 1 Mypy.py, Untitled - 1 Mypy > ...
- Code Area:** The code is a Python script for a Library Management System. It defines a class `Library` with methods for adding, removing, and displaying books. It also includes a main program loop for user interaction.

```
File Edit Selection View Go Run ... ← → Search

C:\Users\Sameer Chauhan\Desktop> Untitled - 1 Mypy X Untitled - 1 Mypy.py Untitled - 1 Mypy > ...

1 class Library:
2     def __init__(self):
3         self.books = []
4
5     def add_book(self, book_name):
6         if book_name not in self.books:
7             self.books.append(book_name)
8             print(f"Book '{book_name}' added successfully.")
9
10    def remove_book(self, book_name):
11        if book_name in self.books:
12            self.books.remove(book_name)
13            print(f"Book '{book_name}' removed successfully.")
14        else:
15            print("Book not found.")
16
17    def display_books(self):
18        if not self.books:
19            print("No books available in the library!")
20        else:
21            print("Books available in the library:")
22            for book in self.books:
23                print("-", book)
24
25 # Main program
26 library = Library()
27
28 while True:
29     print("Library Management System")
30     print("1. Add Book")
31     print("2. Remove Book")
32     print("3. Display Books")
33     print("4. Exit")
34
35     choice = input("Enter your choice: ")
36
37     if choice == "1":
38         book = input("Enter book name to add: ")
39         library.add_book(book)
40
41     elif choice == "2":
42         book = input("Enter book name to remove: ")
43         library.remove_book(book)
44
45     elif choice == "3":
46         library.display_books()
47
48     elif choice == "4":
49         print("Exiting Library Management System.")
50         break
51
52     else:
53         print("Invalid choice. Please try again.")
```



The screenshot shows a Jupyter Notebook interface with several tabs open. The current tab displays a Python script for a library management system. The code includes imports, environment setup, and a loop for user interaction. The output pane shows the execution of the script, displaying menu options (1. Add Book, 2. Remove Book, 3. Display Books, 4. Exit) and their corresponding results (adding 'LOVE IS NOT MEANT FOR ME', removing it, and displaying the list). The bottom status bar indicates the notebook is in master mode, has 24 columns and 4 spaces, and is using Python 3.13.5.

```
PS C:\Users\Sameera Khan\OneDrive\Desktop> c;; cd 'c:\Users\Sameera Khan\OneDrive\Desktop'; & "c:\Users\Sameera Khan\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\Sameera Khan\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher" "53390" ... "C:\Users\Sameera Khan\OneDrive\Desktop\Untitled-1 hh.py"
1 class Library:
    pass

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit

Enter your choice: 1
Enter book name to add: LOVE IS NOT MEANT FOR ME
Book 'LOVE IS NOT MEANT FOR ME' added successfully.

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit

Enter your choice: 2
Enter book name to remove: LOVE IS NOT MEANT FOR ME
Book not found.

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit

Enter your choice: 3
Books available in the library:
- LOVE IS NOT MEANT FOR ME

Library Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit

Enter your choice: 4
Exiting Library Management System.
PS C:\Users\Sameera Khan\OneDrive\Desktop>
```

## OBSERAVTION:

- The AI-generated program correctly integrates **classes, loops, and conditional logic**.
  - The menu-driven structure improves usability and clarity.

- The code is readable, modular, and easy to extend with features like book search or file storage.

## **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.”**

### **CODE:**

The screenshot shows a code editor interface with several tabs open. The active tab contains Python code for an attendance system. The code defines a class `AttendanceSystem` with methods to mark attendance for multiple students and display the results. The terminal window below shows the execution of the program, where the user enters student names and their attendance status (Present or Absent), which are then stored in a dictionary and printed out.

```

1  class AttendanceSystem:
2      def __init__(self):
3          self.attendance = {}
4
5      def mark_attendance(self):
6          n = int(input("Enter number of students: "))
7          for i in range(n):
8              name = input("Enter student name: ")
9              status = input("Enter attendance (P/A): ").upper()
10             if status == 'P':
11                 self.attendance[name] = "Present"
12             else:
13                 self.attendance[name] = "Absent"
14
15     def display_attendance(self):
16         print("\nAttendance Report")
17         print("-----")
18         for name, status in self.attendance.items():
19             print(name, ":", status)
20
21
22 # Main Program
23 system = AttendanceSystem()
24 system.mark_attendance()
25 system.display_attendance()
~
```

PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/BB AI.py"

Enter number of students: 5  
Enter student name: HABEEBA  
Enter attendance (P/A): P  
Enter student name: KHAN  
Enter attendance (P/A): A  
Enter student name: ANAS  
Enter attendance (P/A): P  
Enter student name: ANHA  
Enter attendance (P/A): R  
Enter student name: SANDA  
Enter attendance (P/A): P

Attendance Report

-----

HABEEBA : Present  
KHAN : Absent  
ANAS : Present  
ANHA : Present  
SANDA : Present

## OBSERVATION:

- The AI-generated solution correctly uses a **class-based structure** to manage student attendance data.
- **Loops** are effectively applied to record attendance for multiple students and to display stored records.
- **Conditional statements** accurately distinguish between present and absent students.
- The program logic is simple, readable, and modular, making it easy to extend for additional features such as attendance percentage or file storage.

## TASK DESCRIPTION #5

### (AI-Based Code Completion for Conditional Menu Navigation)

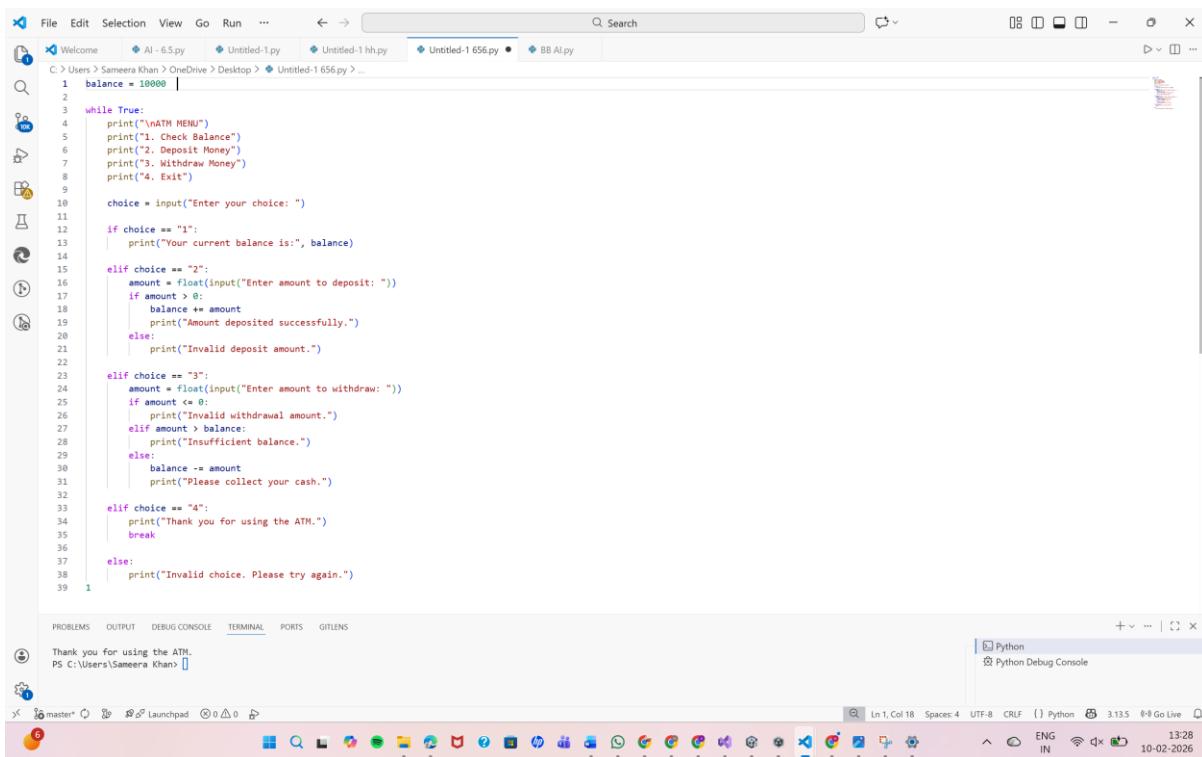
#### TASK:

Use an AI tool to complete a navigation menu.

PROMPT:

“Generate a Python program using loops and conditionals to simulate an ATM menu.”

CODE:



The screenshot shows a code editor window with a Python script titled 'AI - 6.5.py'. The script implements a simple ATM menu system with four options: Check Balance, Deposit Money, Withdraw Money, and Exit. It uses a while loop to repeatedly prompt the user for a choice until they select '4'. The code includes error handling for invalid inputs and attempts to withdraw more than the current balance.

```
1 balance = 10000
2
3 while True:
4     print("\nATM MENU")
5     print("1. Check Balance")
6     print("2. Deposit Money")
7     print("3. Withdraw Money")
8     print("4. Exit")
9
10    choice = input("Enter your choice: ")
11
12    if choice == "1":
13        print("Your current balance is:", balance)
14
15    elif choice == "2":
16        amount = float(input("Enter amount to deposit: "))
17        if amount > 0:
18            balance += amount
19            print("Amount deposited successfully.")
20        else:
21            print("Invalid deposit amount.")
22
23    elif choice == "3":
24        amount = float(input("Enter amount to withdraw: "))
25        if amount <= 0:
26            print("Invalid withdrawal amount.")
27        elif amount > balance:
28            print("Insufficient balance.")
29        else:
30            balance -= amount
31            print("Please collect your cash.")
32
33    elif choice == "4":
34        print("Thank you for using the ATM.")
35        break
36
37    else:
38        print("Invalid choice. Please try again.")
39
```

The terminal below the code editor shows the output of running the script, which ends with 'Thank you for using the ATM.' The status bar at the bottom right indicates the date and time as 10-02-2026 and 13:28.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows files like "Welcome", "AI - 6.5.py", "Untitled-1.py", "Untitled-1 hh.py", "Untitled-1 656.py", and "BB AI.py".
- Search Bar:** A search bar at the top right.
- Terminal:** The active tab, showing the output of a Python script. The script is an ATM program.

```
C:\> Users> Sameera Khan> OneDrive> Desktop > Untitled-1 656.py >_
 1 balance = 100000
 2
 3 while True:
Thank you for using the ATM.
PS C:\Users\Sameera Khan> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/Untitled-1 656.py"
ATH MENU
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 1
Your current balance is: 100000

ATH MENU
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 2
Enter amount to deposit: 2570
Amount deposited successfully.

ATH MENU
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 3
Enter amount to withdraw: 2379
Please collect your cash.

ATH MENU
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 4
Thank you for using the ATM.
PS C:\Users\Sameera Khan>
```
- Output:** Tab showing the output of the terminal session.
- Debug Console:** Tab showing the Python Debug Console.
- Bottom Status Bar:** Shows file status (master), line count (Ln 1, Col 18), spaces (Spaces: 4), encoding (UTF-8), CR/LF, Python 3.13.5, and a Go Live button.
- Taskbar:** Shows various pinned icons including File Explorer, GitHub, and browser tabs.

## OBSERVATION:

- The AI-generated program correctly implements a **menu-driven system** using loops and conditional statements.
  - The **loop structure** allows continuous menu navigation until the user chooses to exit.
  - **Conditional logic (if–elif–else)** ensures accurate execution of ATM operations based on user selection.