

# ASSIGNMENT - 6.5

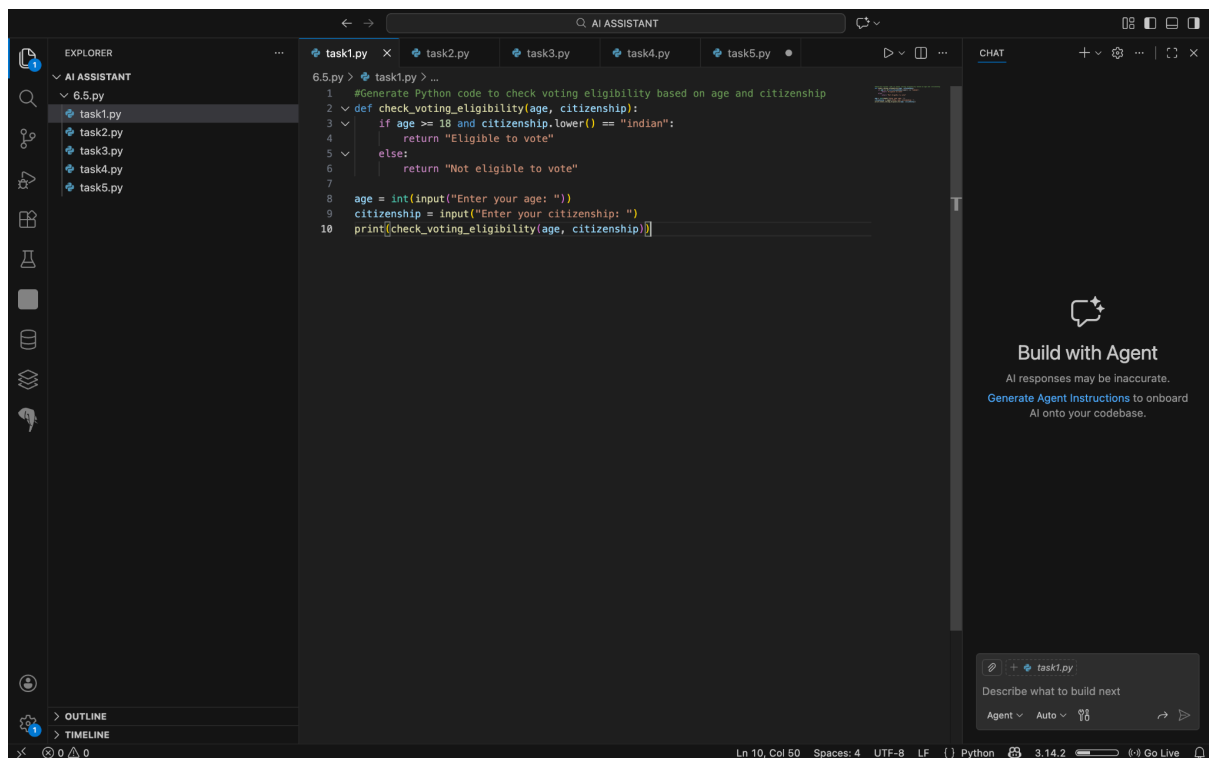
2303A51980

Batch - 30

Task-1 :

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

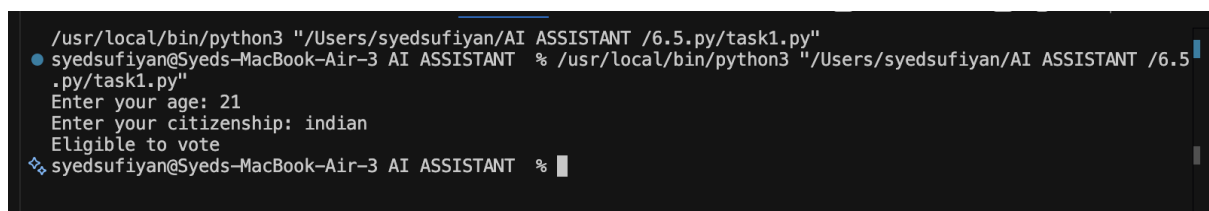
Code :



The screenshot shows a code editor with a dark theme. The Explorer panel on the left shows a project named 'AI ASSISTANT' with a file '6.5.py' containing sub-files 'task1.py' through 'task5.py'. The main editor area shows the content of 'task1.py', which is a Python script. The script defines a function 'check\_voting\_eligibility' that takes 'age' and 'citizenship' as arguments. It checks if the age is 18 or older and if the citizenship is 'indian'. If both conditions are met, it returns 'Eligible to vote'; otherwise, it returns 'Not eligible to vote'. Below the function definition, there is a main block of code that prompts the user for their age and citizenship, and then prints the result of the 'check\_voting\_eligibility' function. The status bar at the bottom indicates the current line is 10, column 50, with 4 spaces, UTF-8 encoding, and LF line endings. The language is set to Python, and the version is 3.14.2. There is also a 'Go Live' button.

```
6.5.py > task1.py > ...
1 #Generate Python code to check voting eligibility based on age and citizenship
2 def check_voting_eligibility(age, citizenship):
3     if age >= 18 and citizenship.lower() == "indian":
4         return "Eligible to vote"
5     else:
6         return "Not eligible to vote"
7
8 age = int(input("Enter your age: "))
9 citizenship = input("Enter your citizenship: ")
10 print(check_voting_eligibility(age, citizenship))
```

Output :



The screenshot shows a terminal window with a dark background. The prompt is '/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task1.py"'. The user enters '21' for age and 'indian' for citizenship. The script outputs 'Eligible to vote'. The prompt is then '/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task1.py"'. The user enters '21' for age and 'indian' for citizenship. The script outputs 'Eligible to vote'. The prompt is then '/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task1.py"'. The user enters '21' for age and 'indian' for citizenship. The script outputs 'Eligible to vote'.

```
/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task1.py"
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT % /usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task1.py"
Enter your age: 21
Enter your citizenship: indian
Eligible to vote
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT %
```

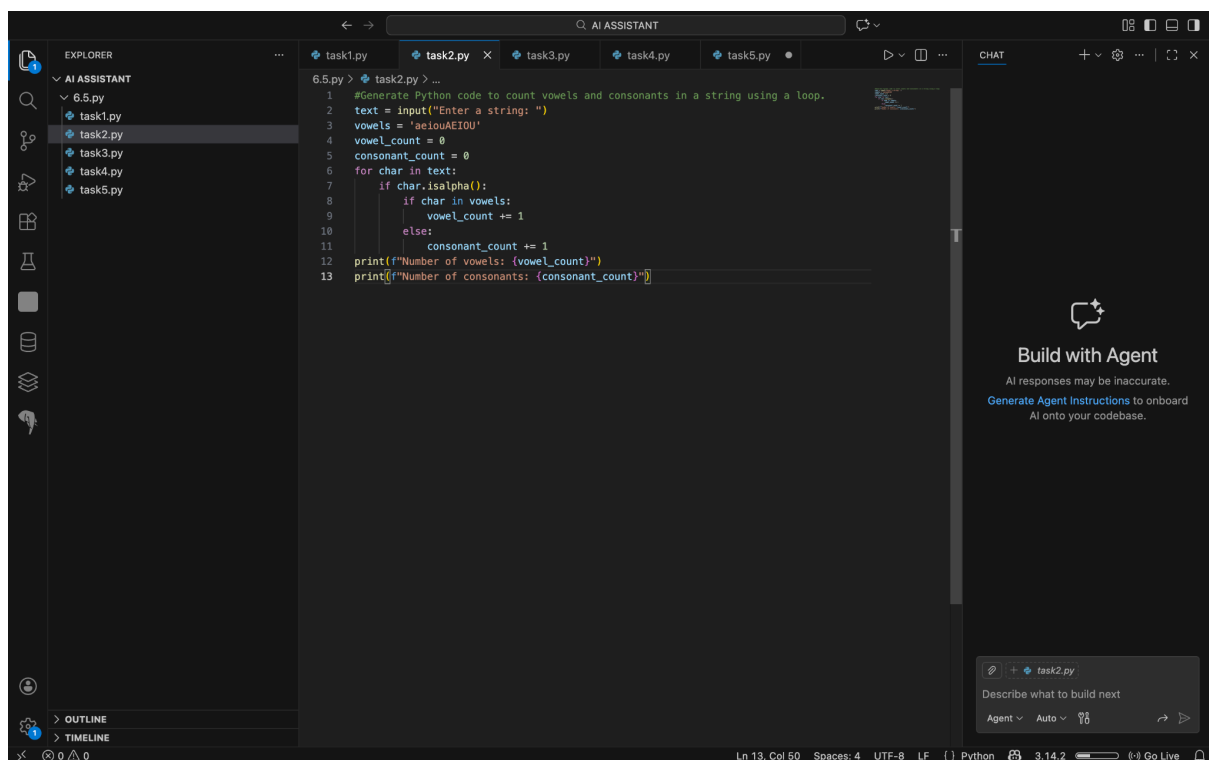
Observation:

The program accurately determines voting eligibility by checking the user's age and citizenship. It correctly identifies eligible voters when the age is 18 or above and the citizenship is Indian, while all other cases are marked as not eligible. The use of a function and case-insensitive input handling improves clarity, reliability, and reusability of the code.

Task-2 :

Prompt : Generate Python code to count vowels and consonants in a string using a loop.

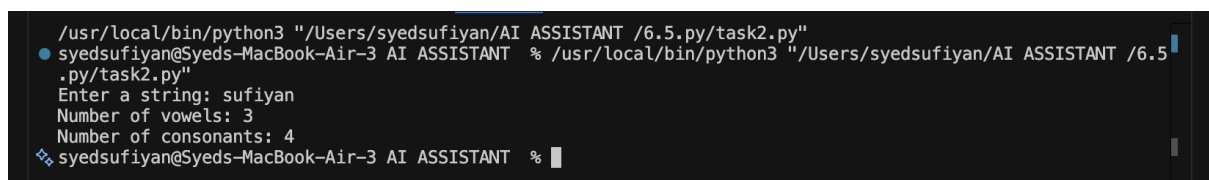
Code :



The screenshot shows a code editor with a dark theme. The Explorer panel on the left shows a project named 'AI ASSISTANT' with a file '6.5.py' containing several Python files: 'task1.py', 'task2.py', 'task3.py', 'task4.py', and 'task5.py'. The 'task2.py' file is selected and its code is displayed in the main editor. The code is a Python script that takes a string input and counts the number of vowels and consonants using a loop. The output is printed to the console. The status bar at the bottom indicates the current line and column (Ln 13, Col 50), the number of spaces (4), the encoding (UTF-8), the line ending (LF), the language (Python), and the version (3.14.2). The right sidebar shows a 'CHAT' panel with a 'Build with Agent' section and a 'task2.py' file selected.

```
1 #Generate Python code to count vowels and consonants in a string using a loop.
2 text = input("Enter a string: ")
3 vowels = 'aeiouAEIOU'
4 vowel_count = 0
5 consonant_count = 0
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 print(f"Number of vowels: {vowel_count}")
13 print(f"Number of consonants: {consonant_count}")
```

Output :



The screenshot shows a terminal window with a dark theme. The command prompt is '/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task2.py"'. The output shows the user entering the string 'sufiyan' and the program outputting 'Number of vowels: 3' and 'Number of consonants: 4'. The prompt is then 'syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT %'.

```
/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task2.py"
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT % /usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task2.py"
Enter a string: sufiyan
Number of vowels: 3
Number of consonants: 4
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT %
```

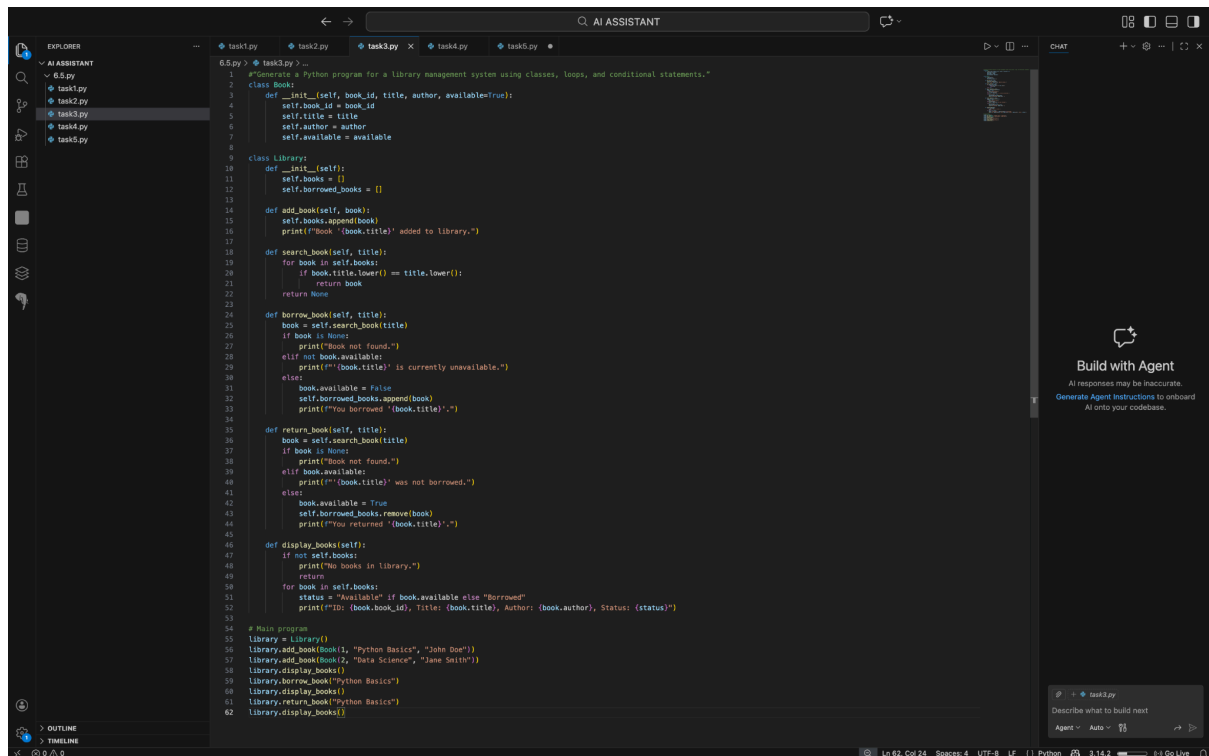
Observation:

The program correctly processes the input string using a loop to identify and count vowels and consonants. It checks each character to ensure it is an alphabet, thereby ignoring spaces and special symbols. Vowels are accurately detected using a predefined vowel set, while remaining alphabets are counted as consonants. The output displays the correct number of vowels and consonants, demonstrating effective use of loops and conditional statements.

### Task-3 :

Prompt : “Generate a Python program for a library management system using classes, loops, and conditional statements.

Code :



```
6.5.py > task3.py > ...
1 #Generate a Python program for a library management system using classes, loops, and conditional statements."
2 class Book:
3     def __init__(self, book_id, title, author, available=True):
4         self.book_id = book_id
5         self.title = title
6         self.author = author
7         self.available = available
8
9 class Library:
10     def __init__(self):
11         self.books = []
12         self.borrowed_books = []
13
14     def add_book(self, book):
15         self.books.append(book)
16         print(f"Book '{book.title}' added to library.")
17
18     def search_book(self, title):
19         for book in self.books:
20             if book.title.lower() == title.lower():
21                 return book
22         return None
23
24     def borrow_book(self, title):
25         book = self.search_book(title)
26         if book is None:
27             print("Book not found.")
28         elif not book.available:
29             print(f"'{book.title}' is currently unavailable.")
30         else:
31             book.available = False
32             self.borrowed_books.append(book)
33             print(f"'You borrowed '{book.title}'")
34
35     def return_book(self, title):
36         book = self.search_book(title)
37         if book is None:
38             print("Book not found.")
39         elif book.available:
40             print(f"'{book.title}' was not borrowed.")
41         else:
42             book.available = True
43             self.borrowed_books.remove(book)
44             print(f"'You returned '{book.title}'")
45
46     def display_books(self):
47         if not self.books:
48             print("No books in library.")
49         return self.books
50
51     for book in self.books:
52         status = "Available" if book.available else "Borrowed"
53         print(f"'ID: {book.book_id}, Title: {book.title}, Author: {book.author}, Status: {status}")
54
55 # Main program
56 library = Library()
57 library.add_book(Book(1, "Python Basics", "John Doe"))
58 library.add_book(Book(2, "Data Science", "Jane Smith"))
59 library.display_books()
60 library.borrow_book("Python Basics")
61 library.display_books()
62 library.return_book("Python Basics")
63 library.display_books()
```

Output :

```
self.available = available

/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task3.py"
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT % /usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5
.py/task3.py"
Book 'Python Basics' added to library.
Book 'Data Science' added to library.
ID: 1, Title: Python Basics, Author: John Doe, Status: Available
ID: 2, Title: Data Science, Author: Jane Smith, Status: Available
You borrowed 'Python Basics'.
ID: 1, Title: Python Basics, Author: John Doe, Status: Borrowed
ID: 2, Title: Data Science, Author: Jane Smith, Status: Available
You returned 'Python Basics'.
ID: 1, Title: Python Basics, Author: John Doe, Status: Available
ID: 2, Title: Data Science, Author: Jane Smith, Status: Available
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT %
```

Observation:

The program successfully implements a library management system using object-oriented programming concepts. It uses a class to manage book data and employs loops and conditional statements to provide a menu-driven interface. Users can add and view books, and the program handles invalid inputs effectively. The structure of the code improves readability, reusability, and overall efficiency.

Task-4 :

Prompt : Generate a Python class to mark and display student attendance using loops.

Code :

```
6.5.py > task4.py > ...
1 class StudentAttendance:
2     def __init__(self):
3         self.attendance = {}
4
5     def add_student(self, student_id, name):
6         """Add a student to the attendance system"""
7         if student_id not in self.attendance:
8             self.attendance[student_id] = {'name': name, 'days': []}
9
10    def mark_attendance(self, student_id, day, status):
11        """Mark attendance for a student (Present/Absent)"""
12        if student_id in self.attendance:
13            self.attendance[student_id]['days'].append((day, status))
14
15    def display_attendance(self):
16        """Display attendance for all students"""
17        for student_id, data in self.attendance.items():
18            print(f"\nStudent ID: {student_id}, Name: {data['name']}")
19            for i, record in enumerate(data['days'], 1):
20                for day, status in record.items():
21                    print(f"    Day {i} ({day}): {status}")
22
23    def get_attendance_summary(self, student_id):
24        """Get attendance summary for a specific student"""
25        if student_id in self.attendance:
26            student = self.attendance[student_id]
27            present = sum(1 for day in student['days'] for status in day.values() if status == 'Present')
28            total = len(student['days'])
29            percentage = (present / total * 100) if total > 0 else 0
30            print(f"\nStudent {student['name']}: {present}/{total} days present ({percentage}%)")
31
32
33 # Example usage
34 if __name__ == "__main__":
35     system = StudentAttendance()
36     system.add_student(1, "Alice")
37     system.add_student(2, "Bob")
38
39     system.mark_attendance(1, "Monday", "Present")
40     system.mark_attendance(1, "Tuesday", "Absent")
41     system.mark_attendance(2, "Monday", "Present")
42     system.mark_attendance(2, "Tuesday", "Present")
43
44     system.display_attendance()
```

Output :

```
/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task4.py"
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT % /usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5
.py/task4.py"

Student ID: 1, Name: Alice
  Day 1 (Monday): Present
  Day 2 (Tuesday): Absent

Student ID: 2, Name: Bob
  Day 1 (Monday): Present
  Day 2 (Tuesday): Present

Alice: 1/2 days present (50.0%)
Bob: 2/2 days present (100.0%)
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT %
```

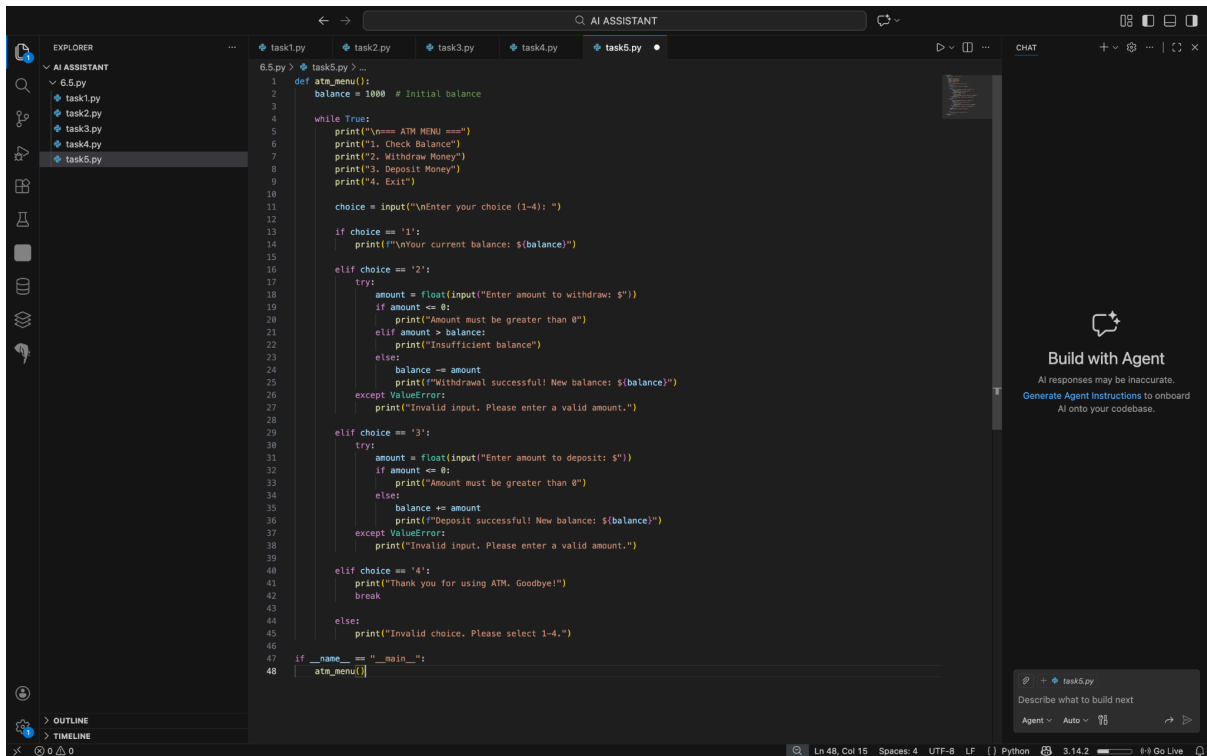
Observation:

The program efficiently manages student attendance using a class-based approach. It uses loops to collect and display attendance information for multiple students. The attendance details are stored in a dictionary, allowing easy updates and retrieval. The code demonstrates effective use of object-oriented programming, loops, and conditional-free structured logic for clarity and readability.

Task-5 :

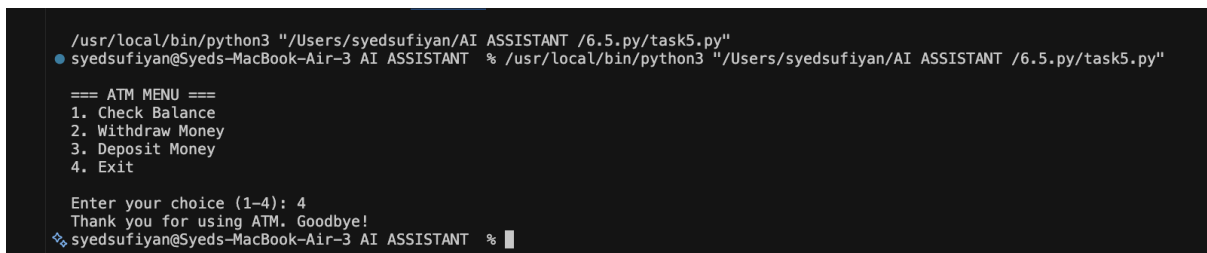
Prompt : Generate a Python program using loops and conditionals to simulate an ATM menu.

Code :



```
6.5.py > task5.py
1 def atm_menu():
2     balance = 1000 # Initial balance
3
4     while True:
5         print("\n== ATM MENU ==")
6         print("1. Check Balance")
7         print("2. Withdraw Money")
8         print("3. Deposit Money")
9         print("4. Exit")
10
11        choice = input("\nEnter your choice (1-4): ")
12
13        if choice == '1':
14            print("\nYour current balance: ${balance}")
15
16        elif choice == '2':
17            try:
18                amount = float(input("Enter amount to withdraw: $"))
19                if amount <= 0:
20                    print("Amount must be greater than 0")
21                elif amount > balance:
22                    print("Insufficient balance")
23                else:
24                    balance -= amount
25                    print("Withdrawal successful! New balance: ${balance}")
26            except ValueError:
27                print("Invalid input. Please enter a valid amount.")
28
29        elif choice == '3':
30            try:
31                amount = float(input("Enter amount to deposit: $"))
32                if amount <= 0:
33                    print("Amount must be greater than 0")
34                else:
35                    balance += amount
36                    print("Deposit successful! New balance: ${balance}")
37            except ValueError:
38                print("Invalid input. Please enter a valid amount.")
39
40        elif choice == '4':
41            print("Thank you for using ATM. Goodbye!")
42            break
43
44        else:
45            print("Invalid choice. Please select 1-4.")
46
47    if __name__ == "__main__":
48        atm_menu()
```

Output :



```
/usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task5.py"
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT % /usr/local/bin/python3 "/Users/syedsufiyan/AI ASSISTANT /6.5.py/task5.py"

=== ATM MENU ===
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit

Enter your choice (1-4): 4
Thank you for using ATM. Goodbye!
syedsufiyan@Syeds-MacBook-Air-3 AI ASSISTANT %
```

Observation:

The program successfully simulates an ATM system using a loop-based menu and conditional statements. It allows users to check balance, deposit money, and withdraw funds with proper validation. The loop ensures continuous operation until the exit option is selected. The program handles invalid choices effectively and demonstrates correct use of loops and conditionals for menu-driven applications.