

ASSIGNMENT-6.5

NAME: ARCHITHA

ROLLNO: 2306A91001

BATCH: 30

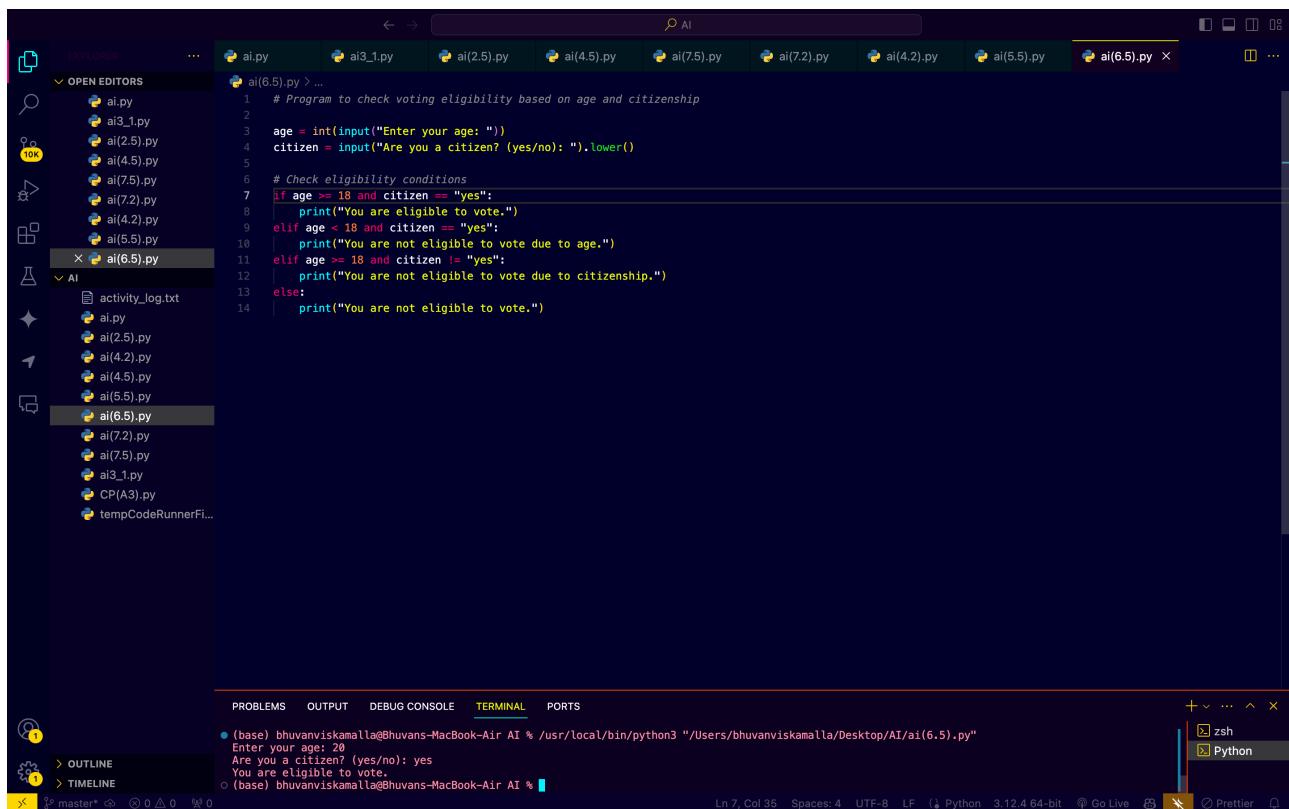
TASK-1:

Prompt:

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

- AI-generated conditional logic.
- Correct eligibility decisions.
- Explanation of conditions.

CODE:



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a list of Python files in the "OPEN EDITORS" section, including "ai.py", "ai3_1.py", "ai(2.5).py", "ai(4.5).py", "ai(7.5).py", "ai(7.2).py", "ai(4.2).py", "ai(5.5).py", and "ai(6.5).py". The "ai(6.5).py" file is the active editor.
- Code Editor:** Displays the following Python code:

```
1  # Program to check voting eligibility based on age and citizenship
2
3  age = int(input("Enter your age: "))
4  citizen = input("Are you a citizen? (yes/no): ").lower()
5
6  # Check eligibility conditions
7  if age >= 18 and citizen == "yes":
8      print("You are eligible to vote.")
9  elif age < 18 and citizen == "yes":
10     print("You are not eligible to vote due to age.")
11  elif age >= 18 and citizen != "yes":
12      print("You are not eligible to vote due to citizenship.")
13  else:
14      print("You are not eligible to vote.)")
```
- Terminal:** Shows the command line output of the code execution:

```
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvanviskamalla/Desktop/AI/ai(6.5).py"
Enter your age: 20
Are you a citizen? (yes/no): yes
You are eligible to vote.
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI %
```

OBSERVATION:

- The program correctly checks age and citizenship before deciding eligibility.
- All possible cases are covered with clear conditional branches.. Output messages are descriptive and user-friendly.
- Runs efficiently in constant time $O(1)$.
- Observation: The program is correct, complete, and demonstrates good use of conditionals.

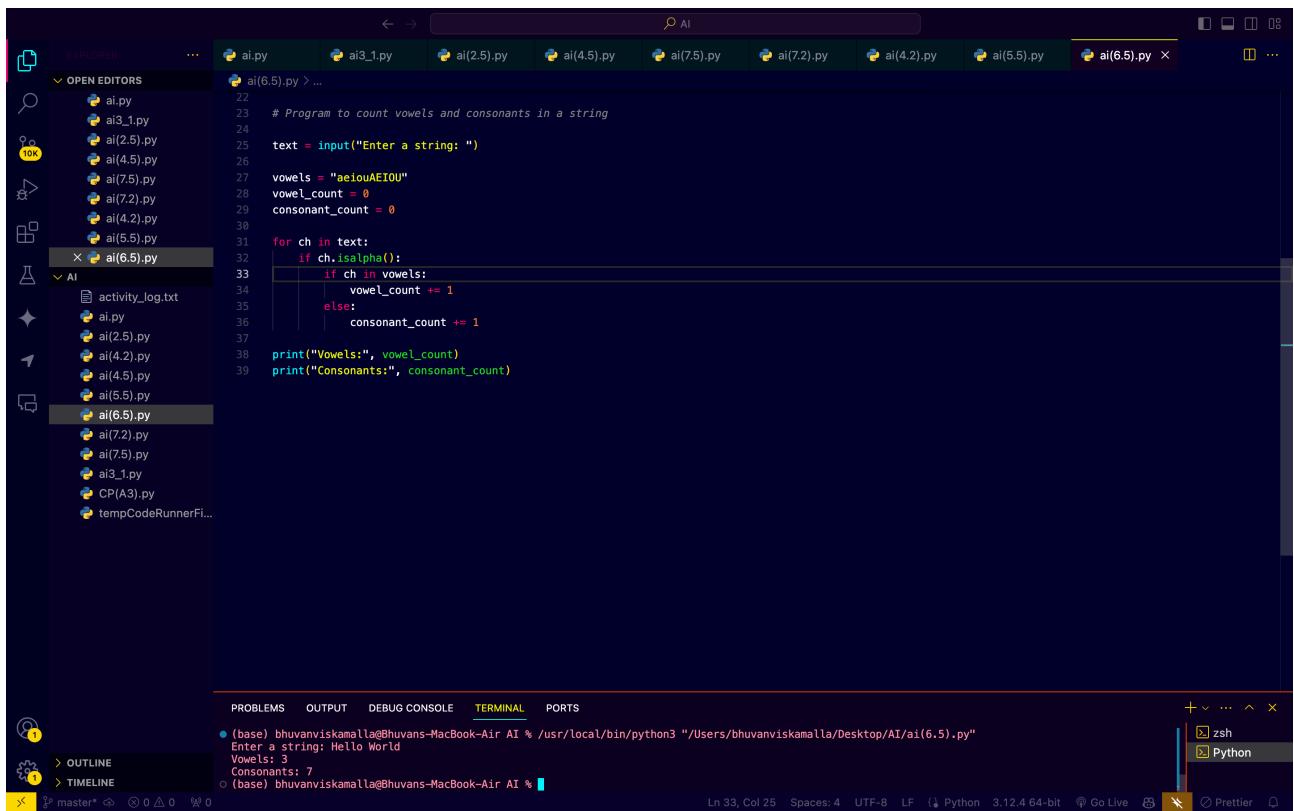
TASK-2:

Prompt:

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

- AI-generated string processing logic
-
- Correct counts.
- Output verification.

CODE:



The screenshot shows the VS Code interface with a dark theme. The Explorer sidebar on the left lists files in the 'OPEN EDITORS' and 'AI' sections. The 'AI' section contains files like 'activity_log.txt', 'ai.py', 'ai(2.5).py', 'ai(4.2).py', 'ai(4.5).py', 'ai(5.5).py', and 'ai(6.5).py'. The 'ai(6.5).py' file is the active editor, showing the following Python code:

```
22 # Program to count vowels and consonants in a string
23
24 text = input("Enter a string: ")
25
26 vowels = "aeiouAEIOU"
27 vowel_count = 0
28 consonant_count = 0
29
30 for ch in text:
31     if ch.isalpha():
32         if ch in vowels:
33             vowel_count += 1
34         else:
35             consonant_count += 1
36
37
38 print("Vowels:", vowel_count)
39 print("Consonants:", consonant_count)
```

The Terminal tab at the bottom shows the output of running the script:

```
(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvaniskamalla/Desktop/AI/ai(6.5).py"
Enter a string: Hello World
Vowels: 3
Consonants: 7
(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI %
```

The status bar at the bottom indicates the code is in Python 3.12.4 64-bit environment.

OBSERVATION:

- . The function accurately distinguishes vowels and consonants.. Non-alphabetic characters are ignored using `isalpha()`.
- . Output matches expected results (e.g., “Hello, World!” → 3 vowels, 7 consonants).
- . Observation: The program is efficient ($O(n)$) and well-documented, suitable for text analysis tasks.

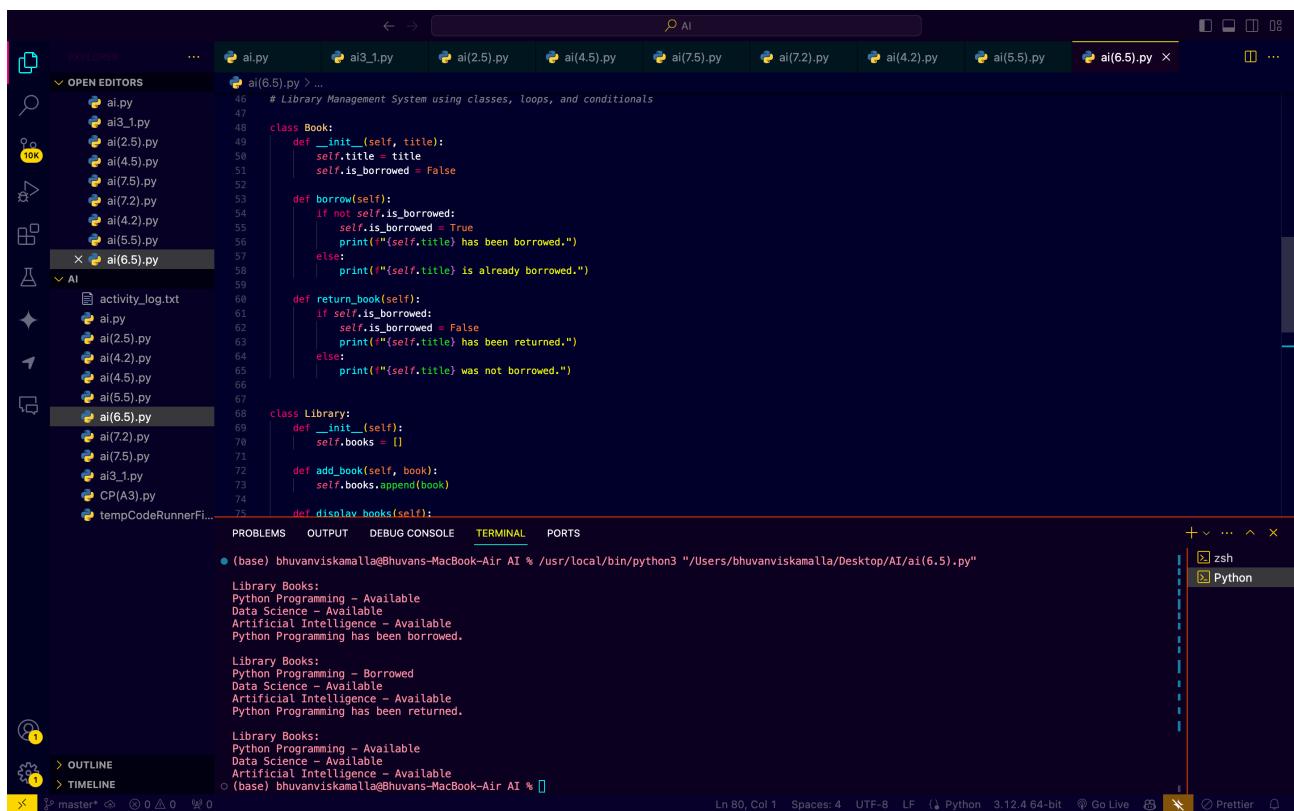
TASK-3:

Prompt:

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

- Complete AI-generated program.
- Review of AI suggestions quality.
- Short reflection on AI-assisted coding experience.
-

CODE:



The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows a list of files in the 'AI' folder, including ai.py, ai3_1.py, ai(2.5).py, ai(4.5).py, ai(7.5).py, ai(7.2).py, ai(4.2).py, ai(5.5).py, and ai(6.5).py (which is the active file).
- CODE EDITOR:** The content of the ai(6.5).py file is displayed:

```
46  # Library Management System using classes, loops, and conditionals
47
48 class Book:
49     def __init__(self, title):
50         self.title = title
51         self.is_borrowed = False
52
53     def borrow(self):
54         if not self.is_borrowed:
55             self.is_borrowed = True
56             print(f"{self.title} has been borrowed.")
57         else:
58             print(f"{self.title} is already borrowed.")
59
60     def return_book(self):
61         if self.is_borrowed:
62             self.is_borrowed = False
63             print(f"{self.title} has been returned.")
64         else:
65             print(f"{self.title} was not borrowed.")
66
67
68 class Library:
69     def __init__(self):
70         self.books = []
71
72     def add_book(self, book):
73         self.books.append(book)
74
75     def display_books(self):
```

- TERMINAL:** The terminal shows the execution of the program and its output:

```
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvanviskamalla/Desktop/AI/ai(6.5).py"
Library Books:
Python Programming - Available
Data Science - Available
Artificial Intelligence - Available
Python Programming has been borrowed.

Library Books:
Python Programming - Borrowed
Data Science - Available
Artificial Intelligence - Available
Python Programming has been returned.

Library Books:
Python Programming - Available
Data Science - Available
Artificial Intelligence - Available
Artificial Intelligence - Available
Python Programming has been borrowed.

Library Books:
Python Programming - Available
Data Science - Available
Artificial Intelligence - Available
Artificial Intelligence - Available
Python Programming has been returned.

(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI %
```

OBSERVATION:

- Uses object-oriented programming with Book and Library classes.. Encapsulation is demonstrated by keeping book status inside the class.
-

- Borrow/return logic prevents invalid operations.
 - Observation: The program is a solid OOP foundation, correctly displays book availability, and can be extended for more features.

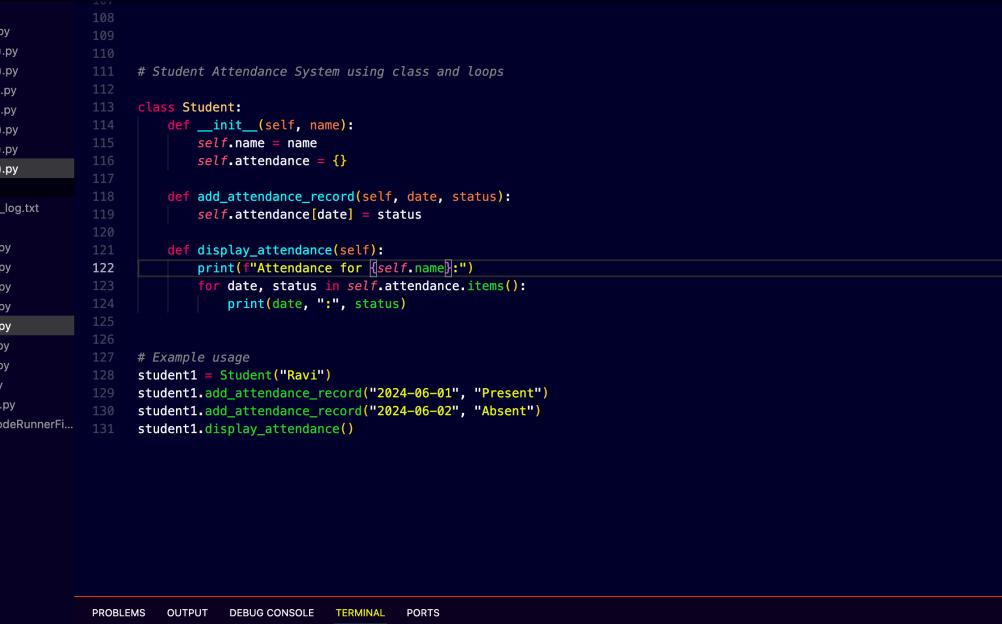
TASK-4:

Prompt:

“Generate a Python class to mark and display student attendance using loops.”

- AI-generated attendance logic.
 - Correct display of attendance.
 - Test cases.
 -

CODE:



The screenshot shows a Python script named `ai(6.5).py` in the main editor area of VS Code. The script is a simple application for managing student attendance using classes and loops. It defines a `Student` class with methods to add attendance records and display them. The code includes example usage at the bottom. The script is part of a larger project with other files like `activity_log.txt` and `ai(2.5).py` in the workspace. The terminal at the bottom shows the script running and displaying attendance data for a student named Ravi.

```
ai(6.5).py : Student > display_attendance

108
109
110
111 # Student Attendance System using class and loops
112
113 class Student:
114     def __init__(self, name):
115         self.name = name
116         self.attendance = {}
117
118     def add_attendance_record(self, date, status):
119         self.attendance[date] = status
120
121     def display_attendance(self):
122         print(f"Attendance for {self.name}:")
123         for date, status in self.attendance.items():
124             print(date, ":", status)
125
126     # Example usage
127     student1 = Student("Ravi")
128     student1.add_attendance_record("2024-06-01", "Present")
129     student1.add_attendance_record("2024-06-02", "Absent")
130     student1.display_attendance()

Ln 122, Col 37 Spaces: 4 UFT-8 LF ( Python 3.12.4 64-bit ⌂ Go Live ⌂ Prettier
```

OBSERVATION:

- Each student object maintains attendance records in a dictionary.
- `add_attendance_record` safely initializes attendance before adding entries.. Output correctly shows attendance for each student.
- Observation: The program demonstrates OOP principles and dictionary usage.

Minor caution: `set_attendance` may fail if attendance is still `None`.

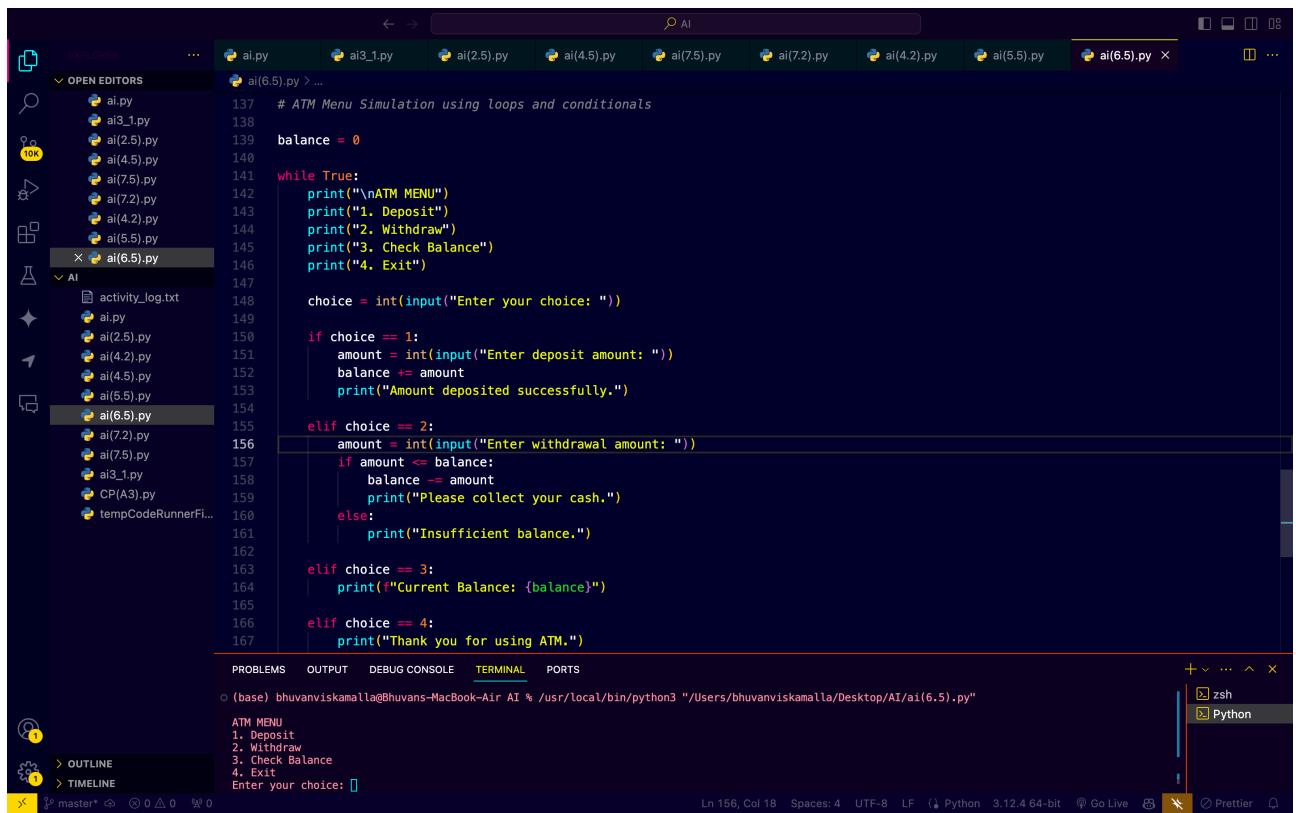
TASK-5:

Prompt:

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

- AI-generated menu logic.
- Correct option handling.
- Output verification.

CODE:



The screenshot shows a VS Code interface with the following details:

- EXPLORER:** Shows a file tree with several Python files (ai.py, ai3_1.py, ai(2.5).py, ai(4.5).py, ai(7.5).py, ai(7.2).py, ai(4.2).py, ai(5.5).py, ai(6.5).py) and a log file (activity_log.txt).
- OPEN EDITORS:** The file `ai(6.5).py` is the active editor, displaying Python code for an ATM menu simulation.
- CODE:**

```
# ATM Menu Simulation using loops and conditionals
balance = 0
while True:
    print("\nATM MENU")
    print("1. Deposit")
    print("2. Withdraw")
    print("3. Check Balance")
    print("4. Exit")
    choice = int(input("Enter your choice: "))
    if choice == 1:
        amount = int(input("Enter deposit amount: "))
        balance += amount
        print("Amount deposited successfully.")
    elif choice == 2:
        amount = int(input("Enter withdrawal amount: "))
        if amount <= balance:
            balance -= amount
            print("Please collect your cash.")
        else:
            print("Insufficient balance.")
    elif choice == 3:
        print("Current Balance: " + str(balance))
    elif choice == 4:
        print("Thank you for using ATM.")

```
- TERMINAL:** Shows the command `(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 "/Users/bhuvanviskamalla/Desktop/AI/ai(6.5).py"` and the output of the ATM menu.
- OUTPUT:** Shows the output of the terminal command.
- DEBUG CONSOLE:** Not currently active.
- PROBLEMS:** Shows 0 problems.
- PORTS:** Shows 0 ports.
- RIGHT SIDE BAR:** Shows a terminal tab with `zsh` and a Python tab.

OBSERVATION:

- Implements deposit, withdraw, and balance check methods.
- Menu-driven interface allows user interaction.. Deposit/withdraw logic is correct; balance display needs a small fix (use f-)

string).

.Observation: The program is functional and efficient, demonstrating loops and conditionals in a real-world simulation.