Week3 – Friday-AI

Prompt Engineering – Improving Prompts and Context Management

2303A51721

Batch-11

Task Description #1 (AI-Based Code Completion for Conditional

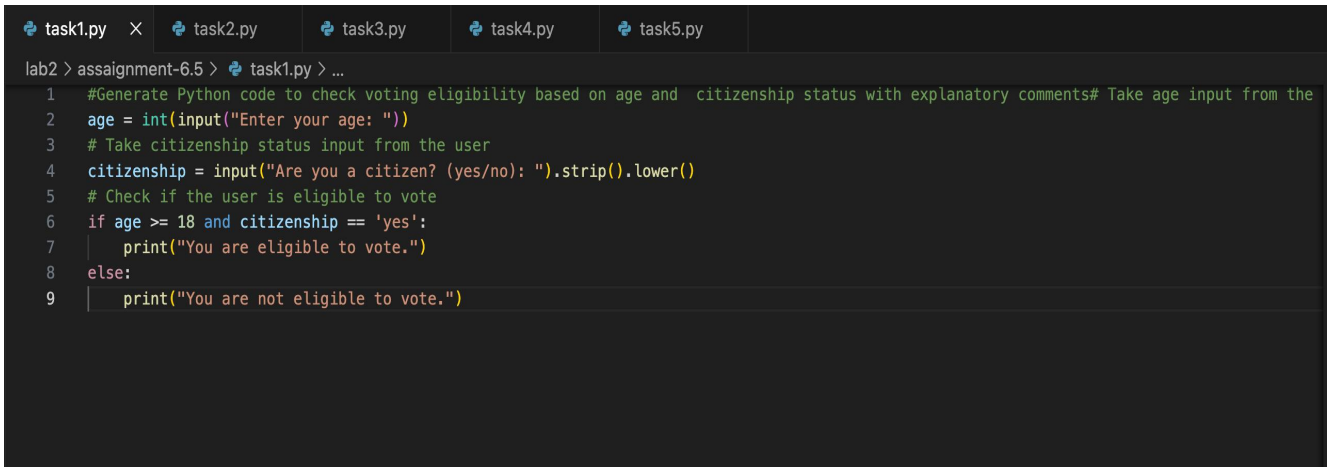
Eligibility Check)

Task: Use an AI tool to generate eligibility logic.

Prompt:

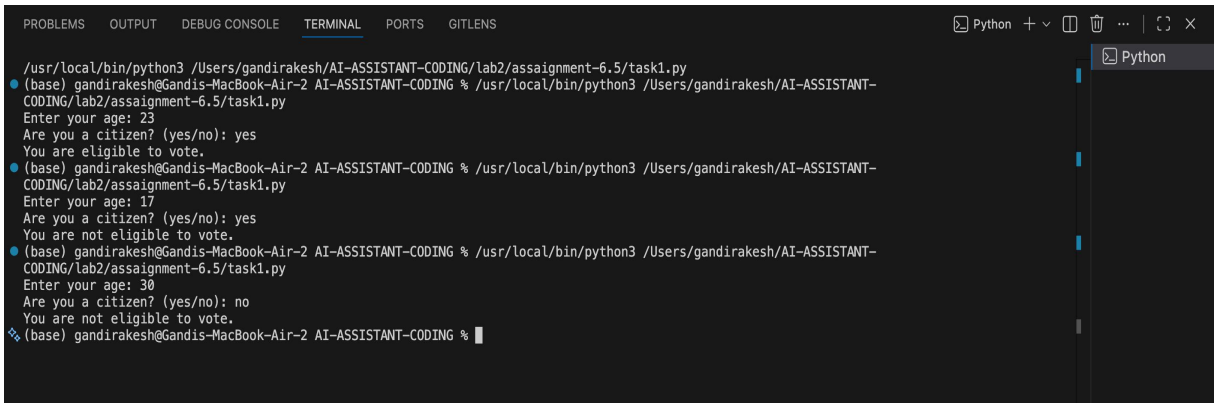
“Generate Python code to check voting eligibility based on age and

citizenship.”



Expected Output:

- AI-generated conditional logic.
- Correct eligibility decisions



Explanation of conditions.

➜ A person must be at least 18 years old and must be a citizen to be eligible to vote.

Task-2

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.”

task1.py | task2.py | task3.py | task4.py | task5.py

lab2 > assaignment-6.5 > task2.py > ...

```python
#Generate Python code to count vowels and consonants in a string using a loop with explanatory comments
# Take string input from the user
user_input = input("Enter a string: ").lower()
# Initialize vowel and consonant counters
vowel_count = 0
consonant_count = 0
# Define a set of vowels
vowels = "aeiou"
# Loop through each character in the input string
for char in user_input:
    # Check if the character is an alphabet
    if char.isalpha():
        # Check if the character is a vowel
        if char in vowels:
            vowel_count += 1  # Increment vowel counter
        else:
            consonant_count += 1  # Increment consonant counter
# Print the counts of vowels and consonants
print(f"Number of vowels: {vowel_count}")
print(f"Number of consonants: {consonant_count}")
```

Expected Output:

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

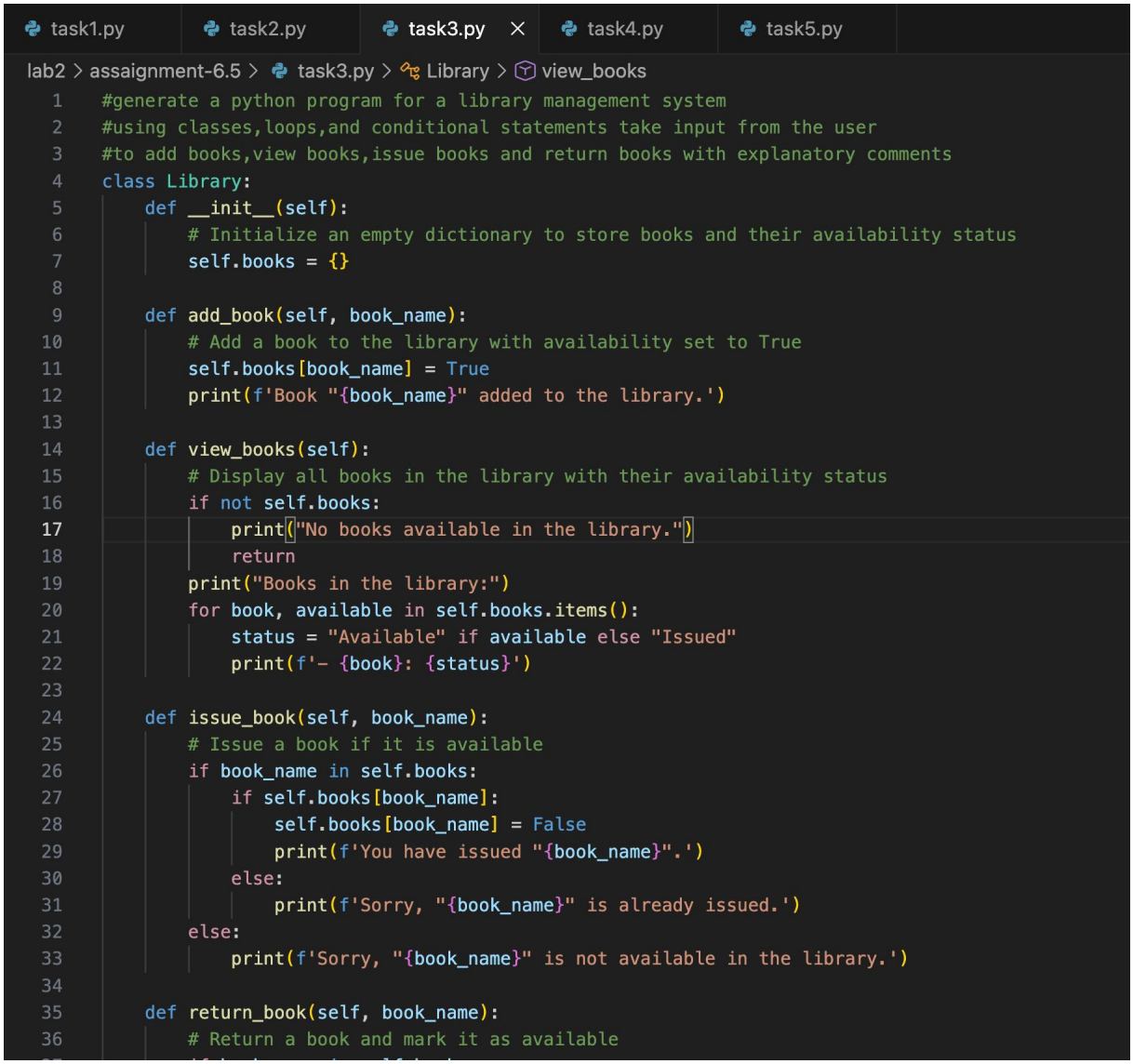
```
/usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assaignment-6.5/task2.py
(base) gandirakesh@Gandis-MacBook-Air-2 AI-ASSISTANT-CODING % /usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assaignment-6.5
/task2.py
Enter a string: rakesh
Number of vowels: 2
Number of consonants: 4
(base) gandirakesh@Gandis-MacBook-Air-2 AI-ASSISTANT-CODING %
```

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.”



task1.py | task2.py | task3.py | task4.py | task5.py

lab2 > assaignment-6.5 > task3.py > Library > view_books

```
4   class Library:
35      def return_book(self, book_name):
37          if book_name in self.books:
38              self.books[book_name] = True
39              print(f'You have returned "{book_name}".')
40          else:
41              print(f'"{book_name}" does not belong to this library.')
42  def main():
43      library = Library()
44      while True:
45          # Display menu options to the user
46          print("\nLibrary Management System")
47          print("1. Add Book")
48          print("2. View Books")
49          print("3. Issue Book")
50          print("4. Return Book")
51          print("5. Exit")
52          choice = input("Enter your choice (1-5): ")
53
54          # Perform actions based on user choice
55          if choice == '1':
56              book_name = input("Enter the name of the book to add: ")
57              library.add_book(book_name)
58          elif choice == '2':
59              library.view_books()
60          elif choice == '3':
61              book_name = input("Enter the name of the book to issue: ")
62              library.issue_book(book_name)
63          elif choice == '4':
64              book_name = input("Enter the name of the book to return: ")
65              library.return_book(book_name)
66          elif choice == '5':
67              print("Exiting the Library Management System.")
68              break
69          else:
70              print("Invalid choice. Please try again.")
```

Expected Output:

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

```
/usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assaignment-6.5/task3.py
(base) gandirakesh@Gandis-MacBook-Air-2 AI-ASSISTANT-CODING % /usr/local/bin/python3 /Users/g
/task3.py

Library Management System
1. Add Book
2. View Books
3. Issue Book
4. Return Book
5. Exit
Enter your choice (1-5): 1
Enter the name of the book to add: the whispering soul
Book "the whispering soul" added to the library.

Library Management System
1. Add Book
2. View Books
3. Issue Book
4. Return Book
5. Exit
Enter your choice (1-5): 2
Books in the library:
- the whispering soul: Available

Library Management System
1. Add Book
2. View Books
3. Issue Book
4. Return Book
5. Exit
Enter your choice (1-5): 3
Enter the name of the book to issue: the whispering soul
You have issued "the whispering soul".

Library Management System
1. Add Book
2. View Books
3. Issue Book
4. Return Book
5. Exit
Enter your choice (1-5): 4
Enter the name of the book to return: the whispering soul
You have returned "the whispering soul".
```

```
Library Management System
1. Add Book
2. View Books
3. Issue Book
4. Return Book
5. Exit
Enter your choice (1-5): 5
Exiting the Library Management System.
```

Explanation of the code.

The program defines a Book class to represent individual books and a Library class to manage a collection of books.

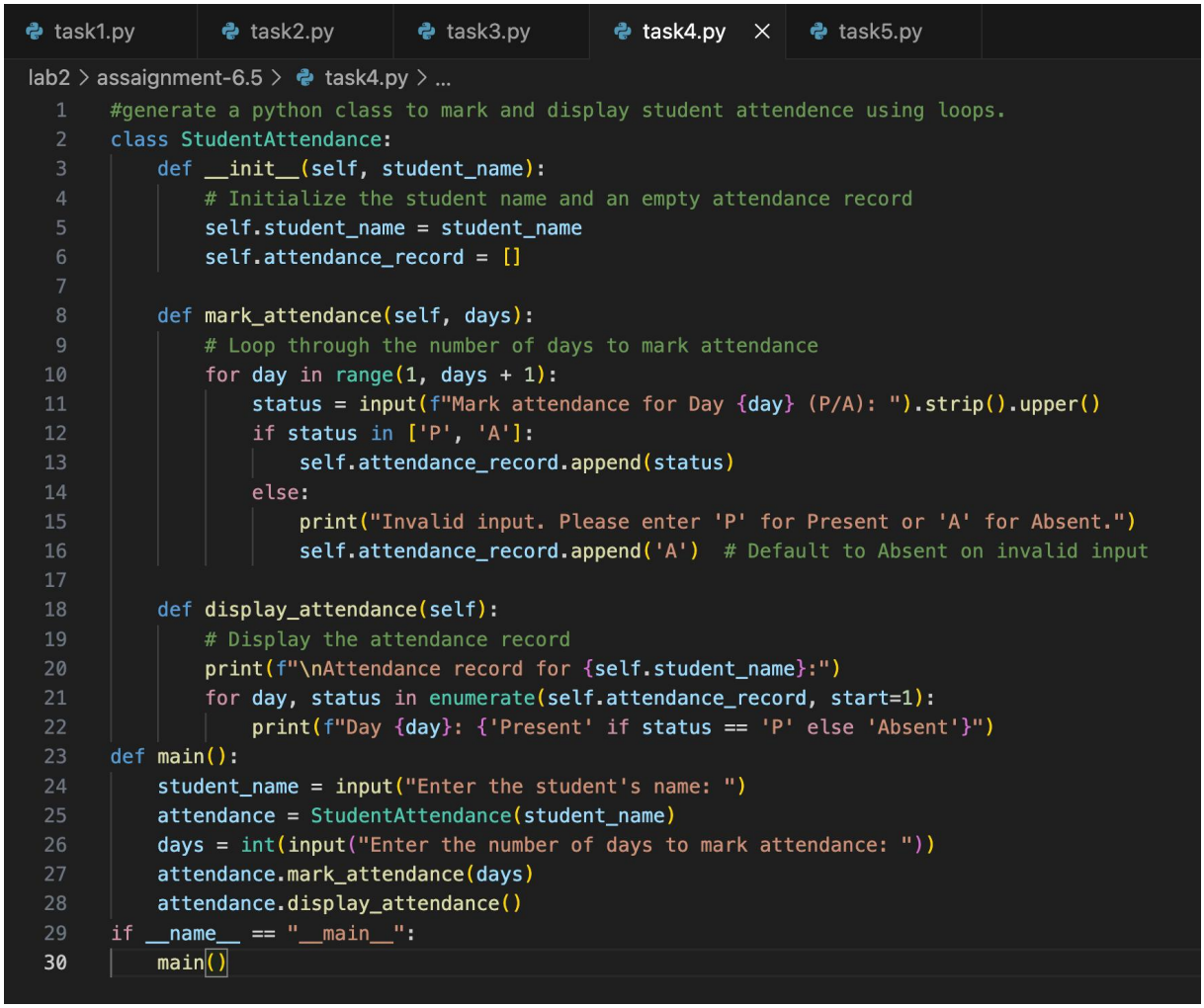
It provides a menu-driven interface for users to add, display, borrow, and return books using loops and conditional statements.

The program continues to run until the user chooses to exit.

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.”



Explanation of the code.

The Student class allows marking attendance for a specified number of days.

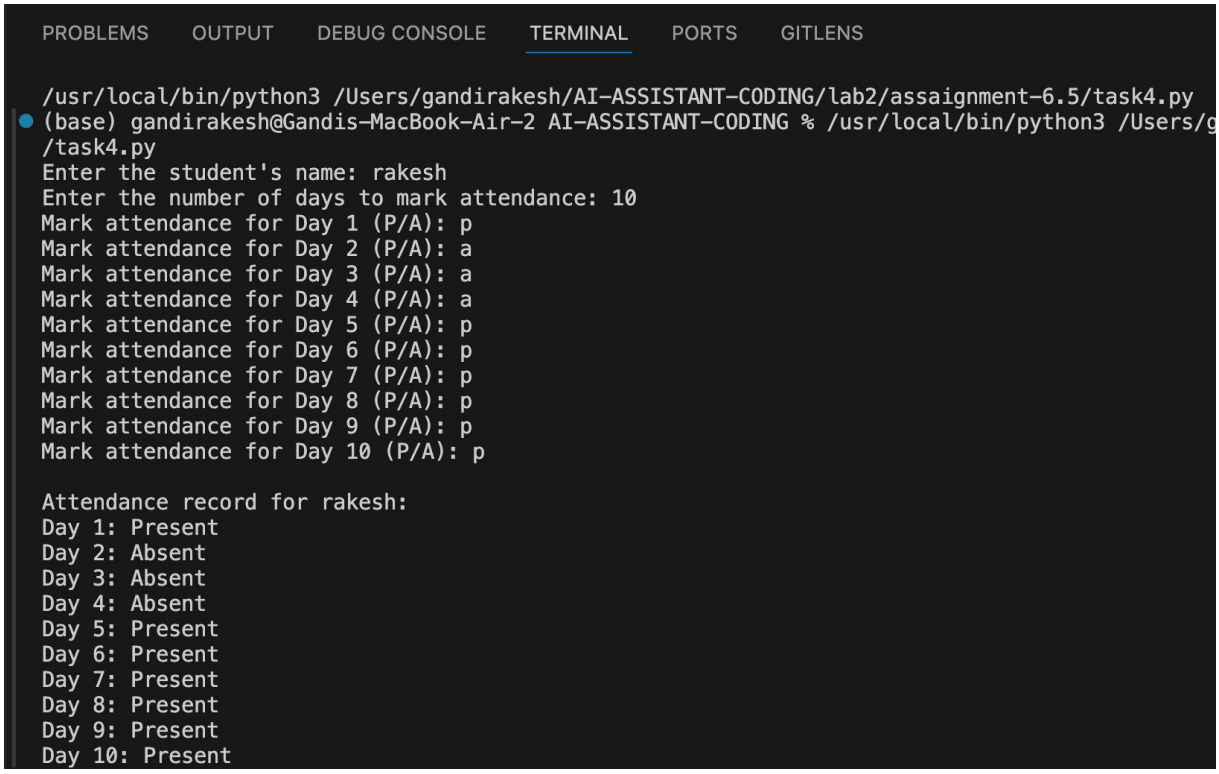
It uses a loop to take user input for each day's attendance status and stores it in a list.

Finally, it displays the attendance record for the student.

Expected Output:

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

• Test cases



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.”

task1.py | task2.py | task3.py | task4.py | task5.py

lab2 > assaignment-6.5 > task5.py > ...

```python
#generate a pytho program using loops and conditional statements to simulate an atm menu.
class ATM:
    def __init__(self, initial_balance=0):
        # Initialize the ATM with a starting balance
        self.balance = initial_balance

    def display_menu(self):
        # Display the ATM menu options
        print("\nATM Menu:")
        print("1. Check Balance")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Exit")

    def check_balance(self):
        # Display the current balance
        print(f"Your current balance is: ${self.balance:.2f}")

    def deposit_money(self, amount):
        # Deposit money into the account
        if amount > 0:
            self.balance += amount
            print(f"${amount:.2f} deposited successfully.")
        else:
            print("Deposit amount must be positive.")

    def withdraw_money(self, amount):
        # Withdraw money from the account if sufficient funds are available
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"${amount:.2f} withdrawn successfully.")
            else:
                print("Insufficient funds for this withdrawal.")
        else:
            print("Withdrawal amount must be positive.")
```

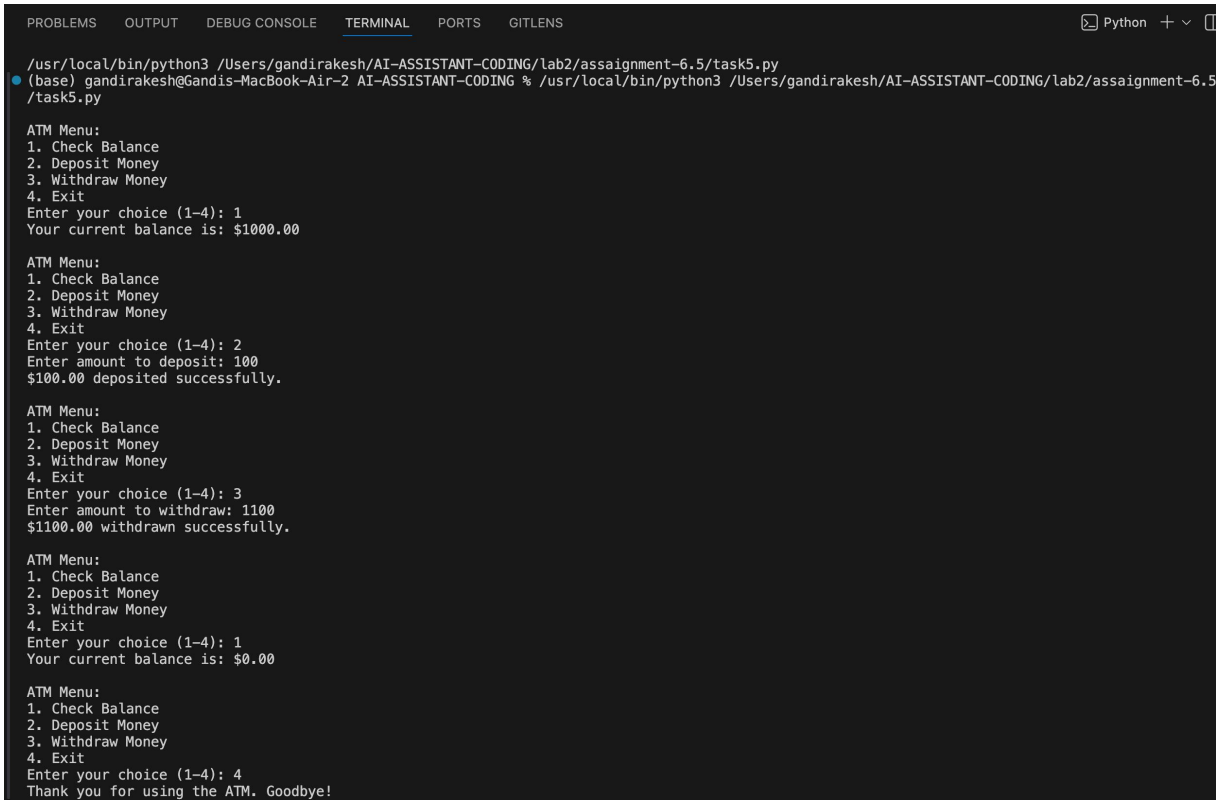
```python
def main():
    atm = ATM(initial_balance=1000)  # Starting with an initial balance of $1000
    while True:
        atm.display_menu()
        choice = input("Enter your choice (1-4): ")

        if choice == '1':
            atm.check_balance()
        elif choice == '2':
            amount = float(input("Enter amount to deposit: "))
            atm.deposit_money(amount)
        elif choice == '3':
            amount = float(input("Enter amount to withdraw: "))
            atm.withdraw_money(amount)
        elif choice == '4':
            print("Thank you for using the ATM. Goodbye!")
            break
        else:
            print("Invalid choice. Please select a valid option.")
if __name__ == "__main__":
    main()
```

Explanation of the code.

The program simulates an ATM menu using a loop to continuously display options until the user chooses to exit.

It uses conditional statements to handle different user choices such as checking balance, depositing money, and

withdrawing money, ensuring valid inputs for each operation.


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GITLENS
Python
/usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assaignment-6.5/task5.py
(base) gandirakesh@Gandis-MacBook-Air-2 AI-ASSISTANT-CODING % /usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assaignment-6.5
/task5.py
ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 1
Your current balance is: $1000.00
ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 2
Enter amount to deposit: 100
$100.00 deposited successfully.
ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 3
Enter amount to withdraw: 1100
$1100.00 withdrawn successfully.
ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 1
Your current balance is: $0.00
ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 4
Thank you for using the ATM. Goodbye!