

AI Assisted Coding Assignment - 6.5

Name: L. Sharan Sai Varshith

Ht.no:2303A51450

Bt.no:21

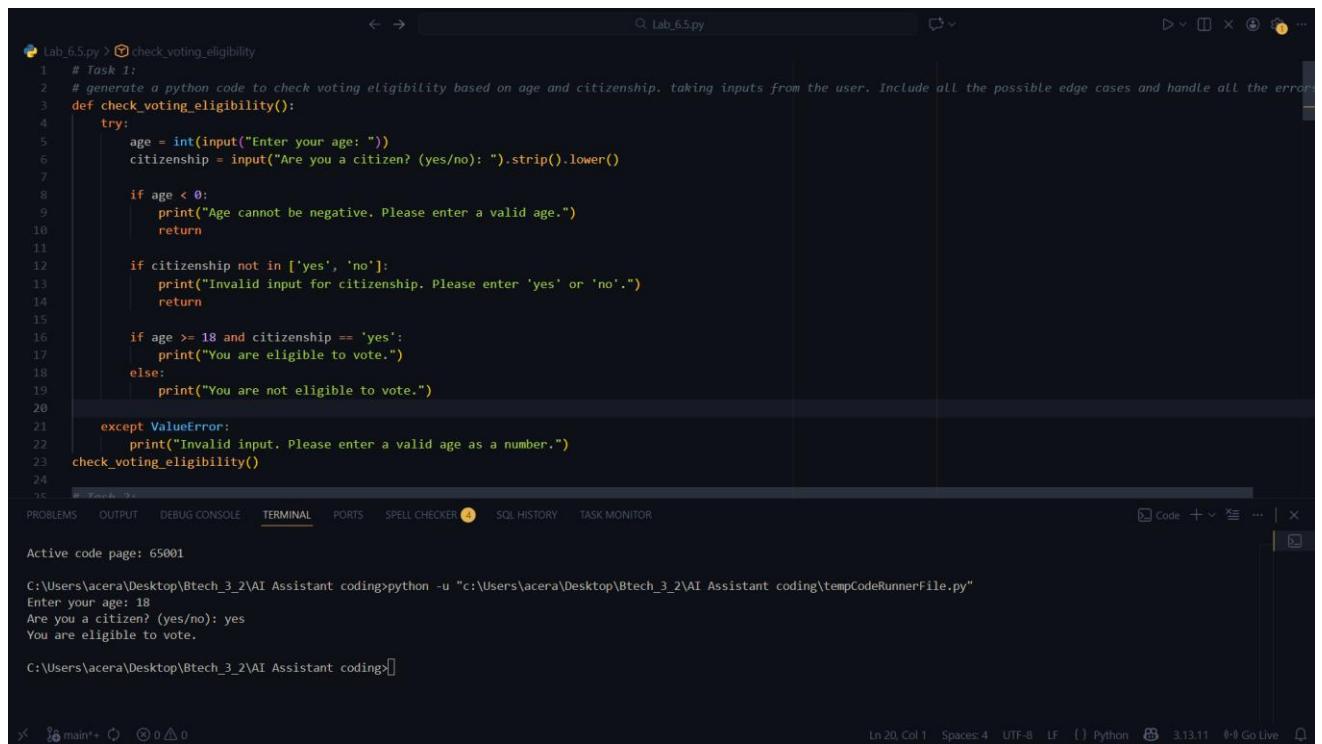
Task Description #1

(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 a code editor window with the following Python code:

```
Lab_6.5.py > check_voting_eligibility
1  # Task 1:
2  # generate a python code to check voting eligibility based on age and citizenship. taking inputs from the user. Include all the possible edge cases and handle all the errors.
3  def check_voting_eligibility():
4      try:
5          age = int(input("Enter your age: "))
6          citizenship = input("Are you a citizen? (yes/no): ").strip().lower()
7
8          if age < 0:
9              print("Age cannot be negative. Please enter a valid age.")
10             return
11
12          if citizenship not in ['yes', 'no']:
13              print("Invalid input for citizenship. Please enter 'yes' or 'no'.")
14              return
15
16          if age >= 18 and citizenship == 'yes':
17              print("You are eligible to vote.")
18          else:
19              print("You are not eligible to vote.")
20
21      except ValueError:
22          print("Invalid input. Please enter a valid age as a number.")
23  check_voting_eligibility()
```

The terminal below the code editor shows the execution of the script:

```
C:\Users\acer\Desktop\Btech_3_2\AI Assistant coding>python -u "c:\Users\acer\Desktop\Btech_3_2\AI Assistant coding\tempCodeRunnerFile.py"
Enter your age: 18
Are you a citizen? (yes/no): yes
You are eligible to vote.

C:\Users\acer\Desktop\Btech_3_2\AI Assistant coding>
```

Task Description #2

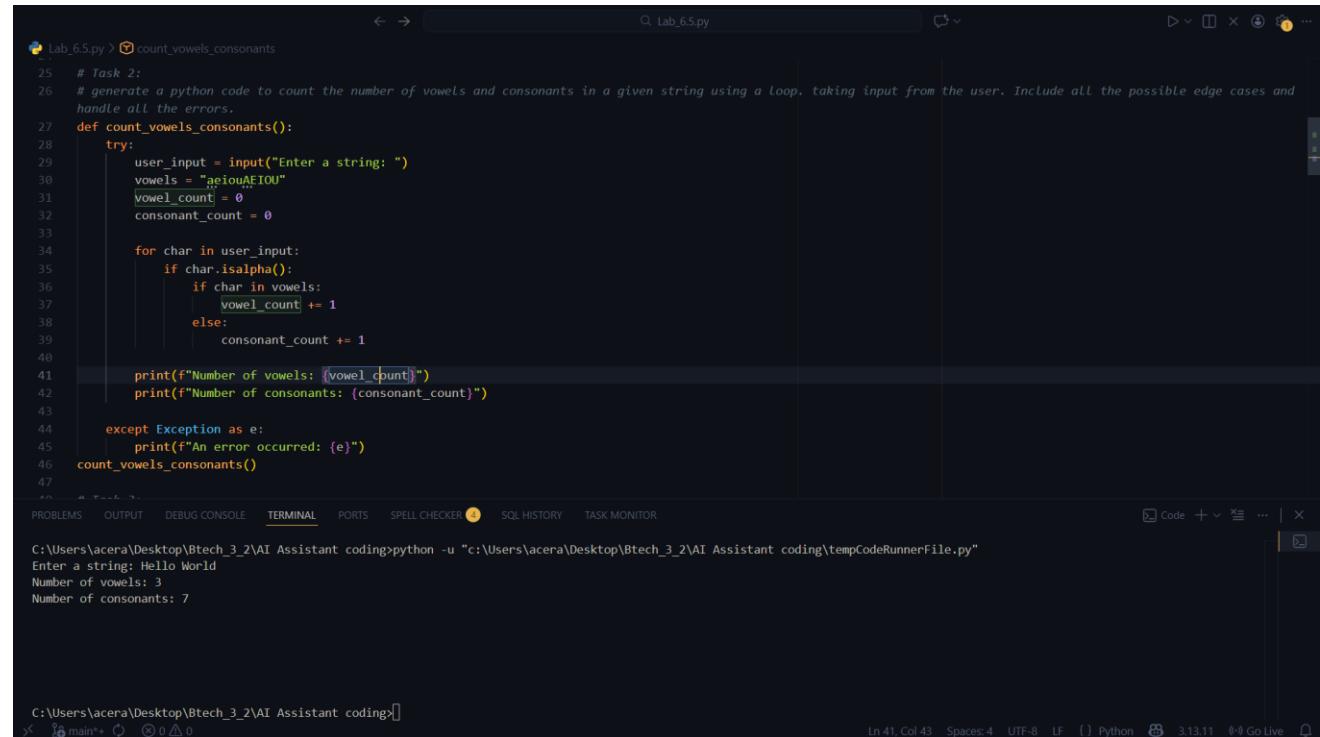
(AI-Based Code Completion for Loop-Based String Processing)

Task: Use an AI tool to process strings using loops.

Prompt:

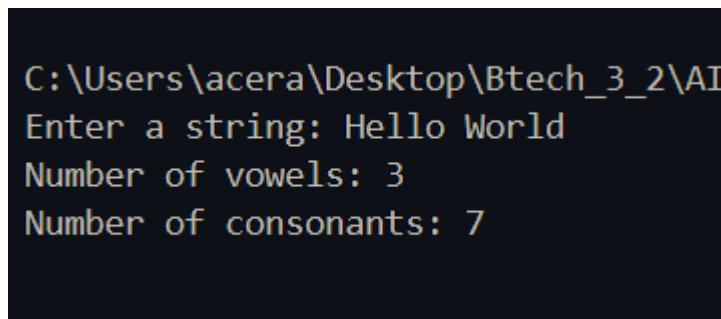
“Generate Python code to count vowels and consonants in a string using a loop.”

Code:



```
Lab_6.5.py > count_vowels_consonants
25  # Task 2:
26  # generate a python code to count the number of vowels and consonants in a given string using a Loop. taking input from the user. Include all the possible edge cases and handle all the errors.
27 def count_vowels_consonants():
28     try:
29         user_input = input("Enter a string: ")
30         vowels = "aeiouAEIOU"
31         vowel_count = 0
32         consonant_count = 0
33
34         for char in user_input:
35             if char.isalpha():
36                 if char in vowels:
37                     vowel_count += 1
38                 else:
39                     consonant_count += 1
40
41         print(f"Number of vowels: {vowel_count}")
42         print(f"Number of consonants: {consonant_count}")
43
44     except Exception as e:
45         print(f"An error occurred: {e}")
46     count_vowels_consonants()
47
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>python -u "c:\Users\acera\Desktop\Btech_3_2\AI Assistant coding\tempCodeRunnerFile.py"
Enter a string: Hello World
Number of vowels: 3
Number of consonants: 7
```

Output:



```
C:\Users\acera\Desktop\Btech_3_2\AI Assistant coding>
C:\Users\acera\Desktop\Btech_3_2\AI
Enter a string: Hello World
Number of vowels: 3
Number of consonants: 7
```

Task Description #3

(AI-Assisted Code Completion Reflection Task)

Task: Use an AI tool to generate a complete program using classes, loops, and conditionals.

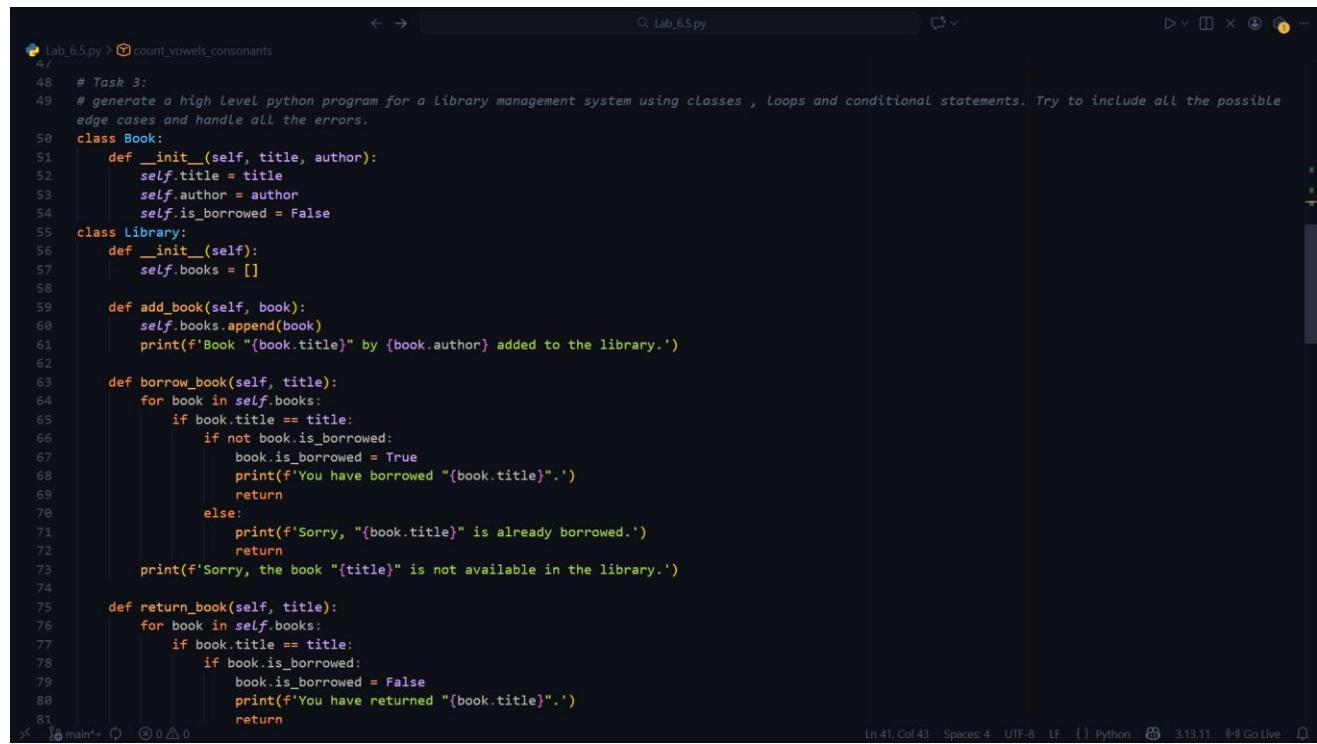
Prompt:

“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:



```
Lab_6.5.py > count_vowels_consonants
4/
48 # Task 3:
49 # generate a high level python program for a Library management system using classes , Loops and conditional statements. Try to include all the possible
edge cases and handle all the errors.
50 class Book:
51     def __init__(self, title, author):
52         self.title = title
53         self.author = author
54         self.is_borrowed = False
55 class Library:
56     def __init__(self):
57         self.books = []
58
59     def add_book(self, book):
60         self.books.append(book)
61         print(f'Book "{book.title}" by {book.author} added to the library.')
62
63     def borrow_book(self, title):
64         for book in self.books:
65             if book.title == title:
66                 if not book.is_borrowed:
67                     book.is_borrowed = True
68                     print(f'You have borrowed "{book.title}"')
69                     return
70                 else:
71                     print(f'Sorry, "{book.title}" is already borrowed.')
72                     return
73         print(f'Sorry, the book "{title}" is not available in the library.')
74
75     def return_book(self, title):
76         for book in self.books:
77             if book.title == title:
78                 if book.is_borrowed:
79                     book.is_borrowed = False
80                     print(f'You have returned "{book.title}"')
81
In 41, Col 43 Spaces 4 UTM-a LF {} Python 3.13.11 ⓘ Go Live
```

```

55  class Library:
56      print(f'Sorry, the book "{title}" does not belong to this library.')
57
58  def main():
59      library = Library()
60      while True:
61          print("\nLibrary Management System")
62          print("1. Add Book")
63          print("2. Borrow Book")
64          print("3. Return Book")
65          print("4. Exit")
66          choice = input("Enter your choice (1-4): ")
67
68          if choice == '1':
69              title = input("Enter book title: ")
70              author = input("Enter book author: ")
71              book = Book(title, author)
72              library.add_book(book)
73
74          elif choice == '2':
75              title = input("Enter the title of the book to borrow: ")
76              library.borrow_book(title)
77
78          elif choice == '3':
79              title = input("Enter the title of the book to return: ")
80              library.return_book(title)
81
82          elif choice == '4':
83              print("Exiting the Library Management System.")
84              break
85
86          else:
87              print("Invalid choice. Please enter a number between 1 and 4.")
88
89  if __name__ == "__main__":
90      main()
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

```

C:\Users\acer\Desktop\Btech_3_2\AI Assistant coding>

Output:

```

C:\Users\acer\Desktop\Btech_3_2\AI Assistant coding>python -u "c:\Users\acer\Desktop\Btech_3_2\AI Assistant coding\tempCodeRunnerFile.py"
Library Management System
1. Add Book
2. Borrow Book
3. Return Book
4. Exit
Enter your choice (1-4): 1
Enter book title: Tales
Enter book author: Herculus
Book "Tales" by Herculus added to the library.

Library Management System
1. Add Book
2. Borrow Book
3. Return Book
4. Exit
Enter your choice (1-4): 4
Exiting the Library Management System.

```

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.”

Expected Output:

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

Code:

```
Lab_6.5.py > Attendance
116 # Task 4:
117 # generate a python class to mark and display attendance of the students using loops and conditional statements. Include all the possible edge cases and handle all the errors.
118 class Attendance:
119     def __init__(self):
120         self.attendance_record = {}
121
122     def mark_attendance(self, student_name, status):
123         if status.lower() not in ['present', 'absent']:
124             print("Invalid status. Please enter 'present' or 'absent'.")
125             return
126         self.attendance_record[student_name] = status.lower()
127         print(f"Attendance marked for {student_name} as {status}.")
128
129     def display_attendance(self):
130         if not self.attendance_record:
131             print("No attendance records available.")
132             return
133         print("\nAttendance Record:")
134         for student, status in self.attendance_record.items():
135             print(f"{student}: {status.capitalize()}")
136
137     def main():
138         attendance = Attendance()
139         while True:
140             print("\nAttendance Management System")
141             print("1. Mark Attendance")
142             print("2. Display Attendance")
143             print("3. Exit")
144             choice = input("Enter your choice (1-3): ")
145
146             if choice == '1':
147                 student_name = input("Enter student name: ")
148                 status = input("Enter attendance status (present/absent): ")
149                 attendance.mark_attendance(student_name, status)
150             elif choice == '2':
151                 attendance.display_attendance()
152             elif choice == '3':
153                 print("Exiting the Attendance Management System.")
154                 break
155             else:
156                 print("Invalid choice. Please enter a number between 1 and 3.")
157
158 if __name__ == "__main__":
159     main()
```

```
Lab_6.5.py > Attendance
118 class Attendance:
119     def display_attendance(self):
120         for student, status in self.attendance_record.items():
121             print(f"{student}: {status.capitalize()}")
122
123     def main():
124         attendance = Attendance()
125         while True:
126             print("\nAttendance Management System")
127             print("1. Mark Attendance")
128             print("2. Display Attendance")
129             print("3. Exit")
130             choice = input("Enter your choice (1-3): ")
131
132             if choice == '1':
133                 student_name = input("Enter student name: ")
134                 status = input("Enter attendance status (present/absent): ")
135                 attendance.mark_attendance(student_name, status)
136             elif choice == '2':
137                 attendance.display_attendance()
138             elif choice == '3':
139                 print("Exiting the Attendance Management System.")
140                 break
141             else:
142                 print("Invalid choice. Please enter a number between 1 and 3.")
143
144 if __name__ == "__main__":
145     main()
```

Output:

```
Attendance Management System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice (1-3): 1
Enter student name: sharan
Enter attendance status (present/absent): present
Attendance marked for sharan as present.
```

```
Attendance Management System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice (1-3): 2
```

```
Attendance Record:
sharan: Present
```

```
Attendance Management System
1. Mark Attendance
2. Display Attendance
3. Exit
Enter your choice (1-3): 3
Exiting the Attendance Management System.
```

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:

The screenshot shows a code editor window with the file 'Lab_6.5.py' open. The code defines an 'atm_menu()' function that simulates an ATM menu. It starts with an initial balance of 1000 and a loop that continues until option 4 is selected. The menu options are: 1. Check Balance, 2. Deposit Money, 3. Withdraw Money, and 4. Exit. Option 1 prints the current balance. Option 2 prompts for a deposit amount, checks if it's positive, adds it to the balance, and prints the new balance. Option 3 prompts for a withdrawal amount, checks if it's positive and less than or equal to the balance, subtracts it from the balance, and prints the new balance. Option 4 exits the loop. Error handling is included for invalid input and ValueErrors.

```
Lab_6.5.py > atm_menu
161 # Task 5:
162 # Generate a python program using Loops and conditional statements to simulate an ATM menu.
163 # Expected output :
164 # AI-generated menu logic for ATM menu:
165 # correct option handling.
166 # output verification.
167 def atm_menu():
168     balance = 1000 # Initial balance
169     while True:
170         print("\nATM Menu")
171         print("1. Check Balance")
172         print("2. Deposit Money")
173         print("3. Withdraw Money")
174         print("4. Exit")
175         choice = input("Enter your choice (1-4): ")
176
177         if choice == '1':
178             print(f"Your current balance is: ${balance}")
179         elif choice == '2':
180             try:
181                 amount = float(input("Enter the amount to deposit: "))
182                 if amount <= 0:
183                     print("Please enter a positive amount.")
184                 else:
185                     balance += amount
186                     print(f"You have deposited ${amount}. New balance: ${balance}")
187             except ValueError:
188                 print("Invalid input. Please enter a valid number.")
189         elif choice == '3':
190             try:
191                 amount = float(input("Enter the amount to withdraw: "))
192                 if amount <= 0:
193                     print("Please enter a positive amount.")
194                 elif amount > balance:
195                     print("Insufficient funds. Please enter a smaller amount.")
196                 else:
197                     balance -= amount
198                     print(f"You have withdrawn ${amount}. New balance: ${balance}")
199             except ValueError:
200                 print("Invalid input. Please enter a valid number.")
201         elif choice == '4':
202             print("Thank you for using the ATM. Goodbye!")
203             break
204         else:
205             print("Invalid choice. Please enter a number between 1 and 4.")
206     atm_menu()

```

Ln 175, Col 52 Spaces: 4 UTF-8 LF [] Python 3.13.11 ⓘ Go Live

The screenshot shows the same code editor window with the code now fully implemented. The 'atm_menu()' function now includes a final 'atm_menu()' call at the end of the loop to allow for multiple interactions. The code handles all four menu options correctly, including deposit, withdrawal, and exit logic, and includes appropriate error messages for invalid inputs and ValueErrors.

```
Lab_6.5.py > atm_menu
167 def atm_menu():
168     balance += amount
169     print(f"You have deposited ${amount}. New balance: ${balance}")
170 except ValueError:
171     print("Invalid input. Please enter a valid number.")
172 elif choice == '3':
173     try:
174         amount = float(input("Enter the amount to withdraw: "))
175         if amount <= 0:
176             print("Please enter a positive amount.")
177         elif amount > balance:
178             print("Insufficient funds. Please enter a smaller amount.")
179         else:
180             balance -= amount
181             print(f"You have withdrawn ${amount}. New balance: ${balance}")
182     except ValueError:
183         print("Invalid input. Please enter a valid number.")
184 elif choice == '4':
185     print("Thank you for using the ATM. Goodbye!")
186     break
187 else:
188     print("Invalid choice. Please enter a number between 1 and 4.")
189 atm_menu()

```

Ln 175, Col 52 Spaces: 4 UTF-8 LF [] Python 3.13.11 ⓘ Go Live

Output:

```
Lab_6.5.py > atm_menu
167 def atm_menu():
168     balance = 0
169     print("You have deposited ${amount}. New balance: ${balance}")
170
171     print("ATM Menu")
172     print("1. Check Balance")
173     print("2. Deposit Money")
174     print("3. Withdraw Money")
175     print("4. Exit")
176
177     choice = input("Enter your choice (1-4): ")
178     amount = float(input("Enter the amount to deposit: "))
179
180     if choice == "1":
181         print("Your current balance is: ${balance}")
182     elif choice == "2":
183         balance += amount
184         print("You have deposited ${amount}. New balance: ${balance}")
185     elif choice == "3":
186         withdrawal = float(input("Enter the amount to withdraw: "))
187         if withdrawal <= balance:
188             balance -= withdrawal
189             print("You have withdrawn ${withdrawal}. New balance: ${balance}")
190         else:
191             print("Insufficient funds")
192     elif choice == "4":
193         print("Thank you for using the ATM. Goodbye!")
194     else:
195         print("Invalid choice")
196
197     print("ATM Menu")
198     print("1. Check Balance")
199     print("2. Deposit Money")
200     print("3. Withdraw Money")
201     print("4. Exit")
202
203     choice = input("Enter your choice (1-4): ")
204     amount = float(input("Enter the amount to withdraw: "))
205
206     if choice == "1":
207         print("Your current balance is: ${balance}")
208     elif choice == "2":
209         balance += amount
210         print("You have deposited ${amount}. New balance: ${balance}")
211     elif choice == "3":
212         withdrawal = float(input("Enter the amount to withdraw: "))
213         if withdrawal <= balance:
214             balance -= withdrawal
215             print("You have withdrawn ${withdrawal}. New balance: ${balance}")
216         else:
217             print("Insufficient funds")
218     elif choice == "4":
219         print("Thank you for using the ATM. Goodbye!")
220     else:
221         print("Invalid choice")
```

ATM Menu
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 2
Enter the amount to deposit: 100000
You have deposited \$100000.0. New balance: \$101000.0

ATM Menu
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 3
Enter the amount to withdraw: 25000
You have withdrawn \$25000.0. New balance: \$76000.0

ATM Menu
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 1
Your current balance is: \$76000.0

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!