

AI Assisted Coding

Assignment-6.5

Name: D.Akash

Ht.no:2303A51422

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:

```
lab 06.py 1 X
C: > Users > akash > OneDrive > Desktop > betech_3_2 > AI_Assisted_Coding > lab 06.py > ...
1  '''# Lab 06: Eligibility Logic Generator
2  # Generate Python code to check voting eligibility based on age and citizenship.
3  # Expected Output:
4  #   • AI-generated conditional logic.
5  #   • Correct eligibility decisions.
6  #   • Explanation of conditions.'''
7
8  def generate_eligibility_code():
9      code = ''
10     def check_voting_eligibility(age, is_citizen):
11         if age >= 18 and is_citizen:
12             return "Eligible to vote"
13         else:
14             return "Not eligible to vote"
15     ...
16     return code
17 def explain_conditions():
18     explanation = ''
19     The function `check_voting_eligibility` checks two conditions:
20     1. Age: The individual must be 18 years or older.
21     2. Citizenship: The individual must be a citizen.
22     Both conditions must be true for the individual to be eligible to vote.
23     ...
24     return explanation
```

```

C:> Users > akash > OneDrive > Desktop > betech_3_2 > AI_Assisted_Coding > lab 06.py > ...
25     # Generate the eligibility code
26     eligibility_code = generate_eligibility_code()
27     print("Generated Eligibility Code:")
28     print(eligibility_code)
29
30     # Explain the conditions
31     conditions_explanation = explain_conditions()
32     print("Explanation of Conditions:")
33     print(conditions_explanation)
34
35     # Define the generated code as a function
36     exec(eligibility_code)
37
38     # Test the generated function
39     test_cases = [
40         (20, True),
41         (17, True),
42         (19, False),
43         (16, False)
44     ]
45     for age, is_citizen in test_cases:
46         result = check_voting_eligibility(age, is_citizen)
47         print(f"Age: {age}, Citizen: {is_citizen} => {result}")
48

```

Output:

```

PS C:\Users\akash\AppData\Local\Programs\Microsoft VS Code & C:\Users\akash\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/akash/OneDrive/Desktop/betech_3_2/AI_Assisted_Coding/" Generated Eligibility code:
def check_voting_eligibility(age, is_citizen):
    if age >= 18 and is_citizen:
        return "Eligible to vote"
    else:
        return "Not eligible to vote"

Explanation of Conditions:

The function `check_voting_eligibility` checks two conditions:
1. Age: The individual must be 18 years or older.
2. Citizenship: The individual must be a citizen.
Both conditions must be true for the individual to be eligible to vote.

Age: 20, Citizen: True => Eligible to vote
Age: 17, Citizen: True => Not eligible to vote
Age: 19, Citizen: False => Not eligible to vote
Age: 16, Citizen: False => Not eligible to vote

```



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 06.py > ...
1   ...
2   Task: process strings using loops.
3   Prompt:
4   "Generate Python code to count vowels and consonants in a string
5   using a loop."
6   Expected Output:
7   • AI-generated string processing logic.
8   • Correct counts.
9   • Output verification.''
10
11  def count_vowels_consonants(input_string):
12      vowels = "aeiouAEIOU"
13      vowel_count = 0
14      consonant_count = 0
15
16      for char in input_string:
17          if char.isalpha(): # Check if the character is a letter
18              if char in vowels:
19                  vowel_count += 1
20              else:
21                  consonant_count += 1
22
23      return vowel_count, consonant_count
24
25 # Example usage
26 input_string = "Hello, World!"
27 vowels, consonants = count_vowels_consonants(input_string)
28 print(f"Input String: {input_string}")
29 print(f"Vowels: {vowels}, Consonants: {consonants}")
30
31 # Output verification
32 assert vowels == 3, "Vowel count is incorrect"
33 assert consonants == 7, "Consonant count is incorrect"
34
35 # Output verification
36 print("Output verification passed.")
37
38
39
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding> & C:\Users\akash\AppData\Local\Programs\Python\Python313\python.exe
Input String: Hello, World!
Vowels: 3, Consonants: 7
Output verification passed.
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding>

Output verification passed.
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding>

Output verification passed.
Output verification passed.
Output verification passed.
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_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.”

Expected Output:

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

Code:

```
.lab 06.py > ...
1   ...
2   Task: to generate a complete program using classes,
3   loops, and conditionals.
4   Prompt:
5   "Generate a Python program for a library management system
6   using classes, loops, and conditional statements."
7   Expected Output:
8   • Complete AI-generated program.
9   • Review of AI suggestions quality.
10  • Short reflection on AI-assisted coding experience.''
11
12 class Book:
13     def __init__(self, title, author):
14         self.title = title
15         self.author = author
16         self.is_available = True
17
18     def __str__(self):
19         status = "Available" if self.is_available else "Checked Out"
20         return f"{self.title} by {self.author} - {status}"
21 class Library:
22     def __init__(self):
23         self.books = []
24
25     def add_book(self, book):
26         self.books.append(book)
27         print(f'Book "{book.title}" added to the library.')
28
29     def display_books(self):
30         if not self.books:
31             print("No books in the library.")
32             return
33         for idx, book in enumerate(self.books, start=1):
34             print(f'{idx}. {book}')
```

```
35     def check_out_book(self, book_index):
36         if 0 <= book_index < len(self.books):
37             book = self.books[book_index]
38             if book.is_available:
39                 book.is_available = False
40                 print(f'You have checked out "{book.title}".')
41             else:
42                 print(f'Sorry, "{book.title}" is already checked out.')
43         else:
44             print("Invalid book index.")
45
46
47     def return_book(self, book_index):
48         if 0 <= book_index < len(self.books):
49             book = self.books[book_index]
50             if not book.is_available:
51                 book.is_available = True
52                 print(f'You have returned "{book.title}".')
53             else:
54                 print(f'"{book.title}" was not checked out.')
55         else:
56             print("Invalid book index.")
57
58     def main():
59         library = Library()
60         while True:
61             print("\nLibrary Management System")
62             print("1. Add Book")
63             print("2. Display Books")
64             print("3. Check Out Book")
65             print("4. Return Book")
66             print("5. Exit")
67             choice = input("Enter your choice: ")
```

```
if choice == '1':
    title = input("Enter book title: ")
    author = input("Enter book author: ")
    book = Book(title, author)
    library.add_book(book)
elif choice == '2':
    library.display_books()
elif choice == '3':
    library.display_books()
    index = int(input("Enter the book index to check out: ")) - 1
    library.check_out_book(index)
elif choice == '4':
    library.display_books()
    index = int(input("Enter the book index to return: ")) - 1
    library.return_book(index)
elif choice == '5':
    print("Exiting the system. Goodbye!")
    break
else:
    print("Invalid choice. Please try again.")
if __name__ == "__main__":
    main()
# Review of AI suggestions quality:
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR

PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding> & C:\Users\akash\AppData\Local\Programs\Python\Python313\g\lab 06.py

Library Management System
1. Add Book
2. Display Books
3. Check Out Book
4. Return Book
5. Exit
Enter your choice: 1
Enter book title: llb
Enter book author: prabhas

```

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:

```

temp.py > ...
1
2 Task:To generate an attendance management class.
3 Prompt: "Generate a Python class to mark and display student
4 attendance using loops."
5 Expected Output:
6 • AI-generated attendance logic.
7 • Correct display of attendance.
8 • Test cases"
9
10 class AttendanceManager:
11     def __init__(self):
12         self.attendance = {}
13
14     def mark_attendance(self, student_name, present=True):
15         """Marks attendance for a student."""
16         self.attendance[student_name] = 'Present' if present else 'Absent'
17
18     def display_attendance(self):
19         """Displays the attendance of all students."""
20         print("Attendance Record:")
21         for student, status in self.attendance.items():
22             print(f"{student}: {status}")
23 # Test cases
24 if __name__ == "__main__":
25     manager = AttendanceManager()
26     # Mark attendance for students
27     manager.mark_attendance("Alice", True)
28     manager.mark_attendance("Bob", False)
29     manager.mark_attendance("Charlie", True)
30     # Display attendance
31     manager.display_attendance()
32

```

Output:

```

PS C:\Users\akash\OneDrive\Desktop\betech_3_2\Devpos and fullStack> & C:\Users\akash\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\akash\OneDrive\Desktop\betech_3_2\Devpos and fullStack\temp.py"
Attendance Record:
Alice: Present
Bob: Absent
Charlie: Present
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\Devpos and fullStack>

```

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:

```
temp.py > ...
1   """
2   Task: To complete a navigation menu.
3   Prompt: "Generate a Python program using loops and conditionals
4   to simulate an ATM menu."
5   Expected Output:
6   • AI-generated menu logic.
7   • Correct option handling.
8   • Output verification"""
9
10  def atm_menu():
11      balance = 1000 # Initial balance
12      while True:
13          print("\nATM Menu:")
14          print("1. Check Balance")
15          print("2. Deposit Money")
16          print("3. Withdraw Money")
17          print("4. Exit")
18
19          choice = input("Please select an option (1-4): ")
20
21          if choice == '1':
22              print(f"Your current balance is: ${balance}")
23
24          elif choice == '2':
25              amount = float(input("Enter amount to deposit: $"))
26              if amount > 0:
27                  balance += amount
28                  print(f"${amount} deposited successfully.")
29              else:
30                  print("Invalid amount. Please enter a positive number.")
31
32          elif choice == '3':
33              amount = float(input("Enter amount to withdraw: $"))
34              if 0 < amount <= balance:
35                  balance -= amount
36                  print(f"${amount} withdrawn successfully.")
37              else:
38                  print("Invalid amount. Please check your balance and try again.")
39
40          elif choice == '4':
41              print("Thank you for using the ATM. Goodbye!")
42              break
43
44          else:
45              print("Invalid option. Please select a valid option (1-4).")
46  if __name__ == "__main__":
47      atm_menu()
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR

PS C:\Users\akash\OneDrive\Desktop\betech_3_2\Devpos_and_fullstack> & C:\users\akash\AppData\Local\Programs\Python\Python313\python.exe tech_3_2/Devpos_and_fullstack/temp.py

ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Please select an option (1-4): 2
Enter amount to deposit: $100000
$100000.0 deposited successfully.
```

