

AI Assisted coding

Assignment-6.5

S.Rajesh

2303a52301

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.

The screenshot shows a code editor interface with a Python file named 'assign-6.5.py' open. The code defines a function 'check_voting_eligibility' that takes age and is_citizen as parameters and returns 'Eligible to vote' if age is 18 or older and is_citizen is True, otherwise it returns 'Not eligible to vote'. It includes example usage with different age and citizenship values. Below the code editor is a terminal window showing the execution of the script and its output. A sidebar on the right is titled 'Build with Agent' and contains a note about AI responses being inaccurate and instructions for onboard AI.

```
1  """Generate Python code to check voting eligibility based on age and citizenship."""
2  def check_voting_eligibility(age, is_citizen):
3      if age >= 18 and is_citizen:
4          return "Eligible to vote"
5      else:
6          return "Not eligible to vote"
7  # Example usage
8  age = 20
9  is_citizen = True
10 print(check_voting_eligibility(age, is_citizen)) # Output: Eligible to vote
11 age = 16
12 is_citizen = True
13 print(check_voting_eligibility(age, is_citizen)) # Output: Not eligible to vote
14 age = 25
15 is_citizen = False
16 print(check_voting_eligibility(age, is_citizen)) # Output: Not eligible to vote
17
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

nithyadugyala@Nithyas-MacBook-Air-2 ~ % /usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/assign-6.5.py"
Eligible to vote
Not eligible to vote
Not eligible to vote
nithyadugyala@Nithyas-MacBook-Air-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.”

Expected Output:

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

```
19 #Generate Python code to count vowels and consonants in a string
20 def count_vowels_and_consonants(input_string):
21     vowels = "aeiouAEIOU"
22     vowel_count = 0
23     consonant_count = 0
24
25     for char in input_string:
26         if char.isalpha():
27             if char in vowels:
28                 vowel_count += 1
29             else:
30                 consonant_count += 1
31
32     return vowel_count, consonant_count
33 # Example usage
34 input_string = "Hello World"
35 vowels, consonants = count_vowels_and_consonants(input_string)
36 print(f"Vowels: {vowels}, Consonants: {consonants}")
37 # Output: Vowels: 3, Consonants: 7
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

/usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/assign-6.5.py"
● nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/assign-6.5.py"
Vowels: 3, Consonants: 7
↳ nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding %

Describe what to bui
. . . < > < >

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.

```
40 #Generate a Python program for a library management system using classes, loops, and conditionals
41 class Book:
42     def __init__(self, title, author):
43         self.title = title
44         self.author = author
45         self.is_available = True
46
47     def borrow(self):
48         if self.is_available:
49             self.is_available = False
50             return f"You have borrowed '{self.title}' by {self.author}."
51         else:
52             return f"Sorry, '{self.title}' is currently not available."
53
54     def return_book(self):
55         self.is_available = True
56         return f"You have returned '{self.title}'. Thank you!"
57 class Library:
58     def __init__(self):
59         self.books = []
60
61     def add_book(self, book):
62         self.books.append(book)
63         return f"Added '{book.title}' by {book.author} to the library."
64
65     def display_books(self):
66         for book in self.books:
67             status = "Available" if book.is_available else "Not Available"
68             print(f"'{book.title}' by {book.author} - {status}")
```

```
57 class Library:
58     def __init__(self):
59         self.books = []
60
61     def add_book(self, book):
62         self.books.append(book)
63         return f"Added '{book.title}' by {book.author} to the library."
64
65     def display_books(self):
66         for book in self.books:
67             status = "Available" if book.is_available else "Not Available"
68             print(f"'{book.title}' by {book.author} - {status}")
69
70 # Example usage
71 library = Library()
72 book1 = Book("1984", "George Orwell")
73 book2 = Book("To Kill a Mockingbird", "Harper Lee")
74 print(library.add_book(book1))
75 print(library.add_book(book2))
76 print(book1.borrow())
77 library.display_books()
78 print(book1.return_book())
79 library.display_books()
```

The terminal output shows:

```
/usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/assign-6.5.py"
nithyadugyala@Nithyas-MacBook-Air-2: ~ ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/assign-6.5.py"
Added '1984' by George Orwell to the library.
Added 'To Kill a Mockingbird' by Harper Lee to the library.
'1984' by George Orwell - Available
'To Kill a Mockingbird' by Harper Lee - Available
You have borrowed '1984' by George Orwell.
'1984' by George Orwell - Not Available
'To Kill a Mockingbird' by Harper Lee - Available
You have returned '1984'. Thank you!
'1984' by George Orwell - Available
'To Kill a Mockingbird' by Harper Lee - Available
nithyadugyala@Nithyas-MacBook-Air-2: ~ ai assistedcoding %
```

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.

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

- Title Bar:** assign-6.5.py — ai assistedcoding
- Code Editor:** The file "assign-6.5.py" contains Python code for a "Student" class. The code includes methods for initializing a student with a name and marking attendance for each day. It also includes a usage example where a student named "Alice" is marked as present or absent for four days.
- Terminal:** The terminal shows the output of running the script, which prints the attendance status for "Alice" across four days.
- Right Panel:**
 - Build with Agent:** A section with a speech bubble icon and the text "AI responses may be inaccurate." and "Generate Agent Instructions to onboard AI onto your codebase."
 - Code Completion Suggestion:** A floating panel titled "Describe what to build" with a text input field and several icons.
- Bottom Status Bar:** Shows the file path, line number (Ln 79), column number (Col 28), spaces count (Spaces: 4), encoding (UTF-8 LF), language (Python), and version (3.14.2).

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

The screenshot shows a code editor window with the following Python code:

```
107 #Generate a Python program using loops and conditionals to simulate an ATM menu."
108 def atm_menu():
109     balance = 1000 # Initial balance
110     while True:
111         print("\nATM Menu:")
112         print("1. Check Balance")
113         print("2. Deposit Money")
114         print("3. Withdraw Money")
115         print("4. Exit")
116         choice = input("Please select an option (1-4): ")
117
118         if choice == '1':
119             print(f"Your current balance is: ${balance}")
120         elif choice == '2':
121             amount = float(input("Enter amount to deposit: $"))
122             if amount > 0:
123                 balance += amount
124                 print(f"${amount} deposited successfully.")
125             else:
126                 print("Invalid amount. Please enter a positive value.")
127         elif choice == '3':
128             amount = float(input("Enter amount to withdraw: $"))
129             if 0 < amount <= balance:
130                 balance -= amount
131                 print(f"${amount} withdrawn successfully.")
132             else:
133                 print("Invalid amount or insufficient funds.")
134         elif choice == '4':
135             print("Thank you for using the ATM. Goodbye!")
136             break
137         else:
138             print("Invalid selection. Please choose a valid option.")
```

The code defines a function `atm_menu()` that prints an ATM menu and handles four options: Check Balance, Deposit Money, Withdraw Money, and Exit. It uses a loop to keep the menu running until the user selects option 4.

The screenshot shows the same Python code as above, but with some additions:

```
108 def atm_menu():
109     # Example usage
110     # atm_menu()
111     # Output will vary based on user input
112
113     # Example usage
114     # nithyadugyala@Nithyas-MacBook-Air-2: ~ % /usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/assign-6.5.py"
115
116     # ATM Menu:
117     # 1. Check Balance
118     # 2. Deposit Money
119     # 3. Withdraw Money
120     # 4. Exit
121
122     choice = input("Please select an option (1-4): ")
123
124     # Your current balance is: $1000
125
126     # ATM Menu:
127     # 1. Check Balance
128     # 2. Deposit Money
129     # 3. Withdraw Money
130     # 4. Exit
131
132     # Please select an option (1-4): 1
133
134     # Your current balance is: $1000
```

The code now includes a section at the top for example usage, followed by a comment block for the ATM menu. The code then continues with the original logic for handling user input and printing the current balance.