

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.”

```
task1.py × task2.py task3.py task4.py task5.py
lab2 > assignment-6.5 > task1.py > ...
1 #Generate Python code to check voting eligibility based on age and citizenship status with explanatory comments# Take age input from the
2 age = int(input("Enter your age: "))
3 # Take citizenship status input from the user
4 citizenship = input("Are you a citizen? (yes/no): ").strip().lower()
5 # Check if the user is eligible to vote
6 if age >= 18 and citizenship == 'yes':
7     print("You are eligible to vote.")
8 else:
9     print("You are not eligible to vote.")
```

Expected Output:

- AI-generated conditional logic.
- Correct eligibility decisions



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
/usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assignment-6.5/task1.py
(base) gandirakesh@Gandis-MacBook-Air-2 AI-ASSISTANT-CODING % /usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assignment-6.5/task1.py
Enter your age: 23
Are you a citizen? (yes/no): yes
You are eligible to vote.
(base) gandirakesh@Gandis-MacBook-Air-2 AI-ASSISTANT-CODING % /usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assignment-6.5/task1.py
Enter your age: 17
Are you a citizen? (yes/no): yes
You are not eligible to vote.
(base) gandirakesh@Gandis-MacBook-Air-2 AI-ASSISTANT-CODING % /usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assignment-6.5/task1.py
Enter your age: 30
Are you a citizen? (yes/no): no
You are not eligible to vote.
(base) gandirakesh@Gandis-MacBook-Air-2 AI-ASSISTANT-CODING %
```

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 X task3.py task4.py task5.py
lab2 > assignment-6.5 > task2.py > ...
1  #Generate Python code to count vowels and consonants in a string using a loop with explanatory comments
2  # Take string input from the user
3  user_input = input("Enter a string: ").lower()
4  # Initialize vowel and consonant counters
5  vowel_count = 0
6  consonant_count = 0
7  # Define a set of vowels
8  vowels = "aeiou"
9  # Loop through each character in the input string
10 for char in user_input:
11     # Check if the character is an alphabet
12     if char.isalpha():
13         # Check if the character is a vowel
14         if char in vowels:
15             vowel_count += 1 # Increment vowel counter
16         else:
17             consonant_count += 1 # Increment consonant counter
18 # Print the counts of vowels and consonants
19 print(f"Number of vowels: {vowel_count}")
20 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/assignment-6.5/task2.py
(base) gandirakesh@Gandis-MacBook-Air-2 AI-ASSISTANT-CODING % /usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assignment-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 X task4.py task5.py
lab2 > assignment-6.5 > task3.py > Library > view_books
1  #generate a python program for a library management system
2  #using classes,loops,and conditional statements take input from the user
3  #to add books,view books,issue books and return books with explanatory comments
4  class Library:
5      def __init__(self):
6          # Initialize an empty dictionary to store books and their availability status
7          self.books = {}
8
9      def add_book(self, book_name):
10         # Add a book to the library with availability set to True
11         self.books[book_name] = True
12         print(f'Book "{book_name}" added to the library.')
13
14     def view_books(self):
15         # Display all books in the library with their availability status
16         if not self.books:
17             print("No books available in the library.")
18             return
19         print("Books in the library:")
20         for book, available in self.books.items():
21             status = "Available" if available else "Issued"
22             print(f'- {book}: {status}')
23
24     def issue_book(self, book_name):
25         # Issue a book if it is available
26         if book_name in self.books:
27             if self.books[book_name]:
28                 self.books[book_name] = False
29                 print(f'You have issued "{book_name}".')
30             else:
31                 print(f'Sorry, "{book_name}" is already issued.')
32         else:
33             print(f'Sorry, "{book_name}" is not available in the library.')
34
35     def return_book(self, book_name):
36         # Return a book and mark it as available
```

```
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/assignment-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."

```
task1.py task2.py task3.py task4.py X task5.py
lab2 > assignment-6.5 > task4.py > ...
1  #generate a python class to mark and display student attendance using loops.
2  class StudentAttendance:
3      def __init__(self, student_name):
4          # Initialize the student name and an empty attendance record
5          self.student_name = student_name
6          self.attendance_record = []
7
8      def mark_attendance(self, days):
9          # Loop through the number of days to mark attendance
10         for day in range(1, days + 1):
11             status = input(f"Mark attendance for Day {day} (P/A): ").strip().upper()
12             if status in ['P', 'A']:
13                 self.attendance_record.append(status)
14             else:
15                 print("Invalid input. Please enter 'P' for Present or 'A' for Absent.")
16                 self.attendance_record.append('A') # Default to Absent on invalid input
17
18         def display_attendance(self):
19             # Display the attendance record
20             print(f"\nAttendance record for {self.student_name}:")
21             for day, status in enumerate(self.attendance_record, start=1):
22                 print(f"Day {day}: {'Present' if status == 'P' else 'Absent'}")
23     def main():
24         student_name = input("Enter the student's name: ")
25         attendance = StudentAttendance(student_name)
26         days = int(input("Enter the number of days to mark attendance: "))
27         attendance.mark_attendance(days)
28         attendance.display_attendance()
29     if __name__ == "__main__":
30         main()
```

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

/usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assignment-6.5/task4.py
(base) gandirakesh@Gandis-MacBook-Air-2 AI-ASSISTANT-CODING % /usr/local/bin/python3 /Users/g
/task4.py
Enter the student's name: rakesh
Enter the number of days to mark attendance: 10
Mark attendance for Day 1 (P/A): p
Mark attendance for Day 2 (P/A): a
Mark attendance for Day 3 (P/A): a
Mark attendance for Day 4 (P/A): a
Mark attendance for Day 5 (P/A): p
Mark attendance for Day 6 (P/A): p
Mark attendance for Day 7 (P/A): p
Mark attendance for Day 8 (P/A): p
Mark attendance for Day 9 (P/A): p
Mark attendance for Day 10 (P/A): p

Attendance record for rakesh:
Day 1: Present
Day 2: Absent
Day 3: Absent
Day 4: Absent
Day 5: Present
Day 6: Present
Day 7: Present
Day 8: Present
Day 9: Present
Day 10: Present
```

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

X

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

```
1  #generate a pytho program using loops and conditional statements to simulate an atm menu.
2  class ATM:
3      def __init__(self, initial_balance=0):
4          # Initialize the ATM with a starting balance
5          self.balance = initial_balance
6
7      def display_menu(self):
8          # Display the ATM menu options
9          print("\nATM Menu:")
10         print("1. Check Balance")
11         print("2. Deposit Money")
12         print("3. Withdraw Money")
13         print("4. Exit")
14
15     def check_balance(self):
16         # Display the current balance
17         print(f"Your current balance is: ${self.balance:.2f}")
18
19     def deposit_money(self, amount):
20         # Deposit money into the account
21         if amount > 0:
22             self.balance += amount
23             print(f"${amount:.2f} deposited successfully.")
24         else:
25             print("Deposit amount must be positive.")
26
27     def withdraw_money(self, amount):
28         # Withdraw money from the account if sufficient funds are available
29         if amount > 0:
30             if amount <= self.balance:
31                 self.balance -= amount
32                 print(f"${amount:.2f} withdrawn successfully.")
33             else:
34                 print("Insufficient funds for this withdrawal.")
35         else:
36             print("Withdrawal amount must be positive.")
```



```

37 def main():
38     atm = ATM(initial_balance=1000) # Starting with an initial balance of $1000
39     while True:
40         atm.display_menu()
41         choice = input("Enter your choice (1-4): ")
42
43         if choice == '1':
44             atm.check_balance()
45         elif choice == '2':
46             amount = float(input("Enter amount to deposit: "))
47             atm.deposit_money(amount)
48         elif choice == '3':
49             amount = float(input("Enter amount to withdraw: "))
50             atm.withdraw_money(amount)
51         elif choice == '4':
52             print("Thank you for using the ATM. Goodbye!")
53             break
54         else:
55             print("Invalid choice. Please select a valid option.")
56 if __name__ == "__main__":
57     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 + v

```
/usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assignment-6.5/task5.py  
● (base) gandirakesh@Gandis-MacBook-Air-2 AI-ASSISTANT-CODING % /usr/local/bin/python3 /Users/gandirakesh/AI-ASSISTANT-CODING/lab2/assignment-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!