AI ASSISTED CODING

ASSIGNMENT – 6

NAME : K.GANESH

HT NO : 2303A52356

BATCH : 36

QUESTION : 06

AI-Based Code Completion: Working with suggestions for classes, loops, conditionals

LO1. Use AI-based code completion tools to generate Python code involving classes, loops, and conditionals.

LO2. Interpret and explain AI-generated code line-by-line.

LO3. Identify errors, inefficiencies, or logical flaws in AI- suggested implementations.

LO4. Optimize AI-generated code for better readability and performance.

LO5. Demonstrate ethical and responsible use of AI tools in coding tasks.

TASK -1

PROMPT : Generate Python code to check voting eligibility based on age and citizenship

Code :

```python
#Generate Python code to check voting eligibility based on age and citizenship
def check_voting_eligibility(age, is_citizen):
    if age >= 18 and is_citizen:
        return "Eligible to vote"
    elif age < 18:
        return "Not eligible to vote: Underage"
    else:
        return "Not eligible to vote: Not a citizen"
# Example usage
age = 20
is_citizen = True
result = check_voting_eligibility(age, is_citizen)
print(result)
```

Output:

```
PS D:\desktop\3-2 courses\AI ASSISTED CODING> python -u "d:\desktop\3-2 courses\AI ASSISTED CODING\Ailab-6.5.py"
Eligible to vote
PS D:\desktop\3-2 courses\AI ASSISTED CODING>
```

Explanation :

This code checks whether a person is eligible to vote based on their age and citizenship status. The function check_voting_eligibility takes two inputs: age and is_citizen. If the person is 18 years or older and is a citizen, the function returns that the person is eligible to vote. If the age is less than 18, it returns that the person is not eligible because they are underage. Otherwise, it returns that the person is not eligible because they are not a citizen. In the example given, the age is 20 and the person is a citizen, so the program prints “Eligible to vote.”

Task-2

Prompt : Generate Python code to count vowels and consonants in a string using a loop.

Code :

```python
#Generate Python code to count vowels and consonants in a string using a loop.
def count_vowels_consonants(input_string):
    vowels = "aeiouAEIOU"
    vowel_count = 0
    consonant_count = 0

    for char in input_string:
        if char.isalpha():  # Check if the character is a letter
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1

    return vowel_count, consonant_count
# Example usage
input_string = "Hello World"
vowels, consonants = count_vowels_consonants(input_string)
print(f"Vowels: {vowels}, Consonants: {consonants}")
```

Output:

```
PS D:\desktop\3-2 courses\AI ASSISTED CODING> python -u "d:\desktop\3-2 courses\AI ASSISTED CODING\Ailab-6.5.py"
Eligible to vote
PS D:\desktop\3-2 courses\AI ASSISTED CODING> python -u "d:\desktop\3-2 courses\AI ASSISTED CODING\Ailab-6.5.py"
Vowels: 3, Consonants: 7
PS D:\desktop\3-2 courses\AI ASSISTED CODING>
```

Explanation:

This code counts the number of vowels and consonants in a given string. The function count_vowels_consonants takes one input called input_string. Inside the function, a string vowels is defined containing all lowercase and uppercase vowels, and two variables are used to count vowels and consonants. The program then goes through each character in the input string using a loop. If the character is a letter (checked using isalpha()), it checks whether the character is a vowel or not. If it is a vowel, the vowel count is increased; otherwise, the consonant count is increased. Finally, the function returns both counts. In the example, the input string is "Hello World", so the program prints the number of vowels and consonants in that string.

Task -3

Prompt : Generate a Python program for a library management system using classes, loops, and conditional statements.

Code :

```
#Generate a Python program for a library management system using classes, loops, and conditional statements.
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.is_borrowed = False
class Library:
    def __init__(self):
        self.books = []

    def add_book(self, book):
        self.books.append(book)
        print(f'Book "{book.title}" by {book.author} added to the library.')

    def borrow_book(self, title):
        for book in self.books:
            if book.title == title:
                if not book.is_borrowed:
                    book.is_borrowed = True
                    print(f'You have borrowed "{book.title}".')
                    return
                else:
                    print(f'Sorry, "{book.title}" is already borrowed.')
                    return
        print(f'Sorry, "{title}" is not available in the library.')

    def return_book(self, title):
        for book in self.books:
            if book.title == title:
                if book.is_borrowed:
                    book.is_borrowed = False
                    print(f'You have returned "{book.title}".')
                    return
```

```
                else:
                    print(f'"{book.title}" was not borrowed.')
                    return
        print(f'Sorry, "{title}" does not belong to this library.')

    def display_books(self):
        print("Available books in the library:")
        for book in self.books:
            status = "Borrowed" if book.is_borrowed else "Available"
            print(f'"{book.title}" by {book.author} - {status}')
# Example usage
library = Library()
library.add_book(Book("1984", "George Orwell"))
library.add_book(Book("To Kill a Mockingbird", "Harper Lee"))
library.display_books()
library.borrow_book("1984")
```

```
library.display_books()
library.return_book("1984")
library.display_books()
```

Output :

```
Book "To Kill a Mockingbird" by Harper Lee added to the library.
Available books in the library:
"1984" by George Orwell - Available
"To Kill a Mockingbird" by Harper Lee - Available
You have borrowed "1984".
Available books in the library:
"1984" by George Orwell - Borrowed
"To Kill a Mockingbird" by Harper Lee - Available
You have returned "1984".
Available books in the library:
"1984" by George Orwell - Available
"To Kill a Mockingbird" by Harper Lee - Available
PS D:\desktop\3-2 courses\AI ASSISTED CODING>
```

Explanation :

This code creates a simple library system. The Book class stores book details and whether it is borrowed. The Library class manages books by adding, borrowing, returning, and displaying them. In the example, books are added, one book is borrowed and returned, and the library shows the book status after each step.

Task -4

Prompt : Generate a Python class to mark and display student attendance using loops**.**

Code :

```python
class Student:
    def __init__(self, name):
        self.name = name
        self.attendance = []

    def mark_attendance(self, status):
        self.attendance.append(status)

    def display_attendance(self):
        print(f"Attendance record for {self.name}:")
        for day, status in enumerate(self.attendance, start=1):
            print(f"Day {day}: {'Present' if status else 'Absent'}")
# Example usage
student = Student("Alice")
student.mark_attendance(True)
student.mark_attendance(False)
student.mark_attendance(True)
student.display_attendance()
```

Output :

```
PS D:\desktop\3-2 courses\AI ASSISTED CODING> python -u "d:\desktop\3-2 courses\AI ASSISTED CODING\Ailab-6.5.py"
Attendance record for Alice:
Day 1: Present
Day 2: Absent
Day 3: Present
PS D:\desktop\3-2 courses\AI ASSISTED CODING>
```

Explanation :

This code manages a student's attendance. The Student class stores the student's name and a list of attendance records. The mark_attendance method adds True for present or False for absent to the list. The display_attendance method prints the attendance day by day, showing whether the student was present or absent. In the example, attendance is marked for three days and then displayed for the student Alice.

Task-5

Prompt : Generate a Python program using loops and conditionals to simulate an ATM menu.

Code :

```python
#Generate a Python program using loops and conditionals to simulate an ATM menu.
class ATM:
    def __init__(self, balance=0):
        self.balance = balance

    def display_menu(self):
        print("ATM Menu:")
        print("1. Check Balance")
        print("2. Deposit Money")
        print("3. Withdraw Money")
        print("4. Exit")

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

```
    def deposit_money(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"${amount:.2f} deposited successfully.")
        else:
            print("Invalid deposit amount.")

    def withdraw_money(self, amount):
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"${amount:.2f} withdrawn successfully.")
            else:
                print("Insufficient funds.")
        else:
            print("Invalid withdrawal amount.")
```

```
# Example usage
atm = ATM(1000)  # 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 try again.")
    print()  # Print a newline for better readability
print()  # Print a newline for better readability
```

Output :

```
ATM Menu:
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice (1-4): 2
Enter amount to deposit: 45000
$45000.00 deposited successfully.
```

Explanation :

This code simulates a simple ATM system. The ATM class stores the account balance and provides methods to display a menu, check the balance, deposit money, and withdraw money. The deposit method adds money only if the amount is valid, and the withdraw method allows withdrawal only if there are enough funds. In the example, an ATM is created with an initial balance of $1000. A loop repeatedly shows the menu, takes the user's choice, and performs the selected action until the user chooses to exit.