ASSIGNMENT-6.3

Name: Bandana Giri

HT. No: 2303A51590

Batch: 07

Lab 6: AI-Based Code Completion – Classes, Loops, and Conditionals

Task Description #1: Classes (Student Class)

Scenario

You are developing a simple student information management module.

```python
#Task1
#generate a student class with name rollno branch
class Student:
    def __init__(self, name, roll_no, branch):
        self.name = name
        self.roll_no = roll_no
        self.branch = branch

    def display_details(self):
        print("Student Name:", self.name)
        print("Roll Number:", self.roll_no)
        print("Branch:", self.branch)


# Object creation
student1 = Student("Vyshnavi", 1895, "Cse")
student1.display_details()
```

```
Student Name: Vyshnavi
Roll Number: 1895
Branch: Cse
```

Task Description #2: Loops (Multiples of a Number)

Scenario

You are writing a utility function to display multiples of a given number.

Task

- Prompt the AI tool to generate a function that prints the first 10 multiples of a given number using a loop.
- Analyze the generated loop logic.
- Ask the AI to generate the same functionality using another controlled looping structure (e.g., while instead of for).

```python
#Task2
#generate the same functionality using another controlled looping structure
def print_multiples(num):
    for i in range(1, 11):
        print(num * i)
print_multiples(5)
def print_multiples_while(num):
    i = 1
    while i <= 10:
        print(num * i)
        i += 1

print_multiples_while(5)
```

```
5
10
15
20
25
30
35
40
45
50
5
10
15
20
25
30
35
40
45
50
```

Task Description #3: Conditional Statements (Age Classification)

Scenario

You are building a basic classification system based on age.

Task

- Ask the AI tool to generate nested if-elif-else conditional statements to classify age groups (e.g., child, teenager, adult, senior).
- Analyze the generated conditions and logic.
- Ask the AI to generate the same classification using alternative conditional structures (e.g., simplified conditions or dictionary-based logic).

```python
#Task3
#to generate nested if-elif-else conditional statements to classify age groups (e.g., child, teenager, adult, senior).

def classify_age(age):
    if age < 13:
        return "Child"
    elif age < 20:
        return "Teenager"
    elif age < 60:
        return "Adult"
    else:
        return "Senior"

print(classify_age(25))
```

```
Adult
```

Task Description #4: For and While Loops (Sum of First n Numbers)

Scenario

You need to calculate the sum of the first n natural numbers.

Task

• Use AI assistance to generate a sum_to_n() function using a for loop.

• Analyze the generated code.

• Ask the AI to suggest an alternative implementation using a while loop or a mathematical formula.

```python
#Task4
#For and While Loops (Sum of First n Numbers)
def sum_to_n(n):
    total = 0
    for i in range(1, n + 1):
        total += i
    return total

print(sum_to_n(10))
def sum_to_n_while(n):
    total = 0
    i = 1
    while i <= n:
        total += i
        i += 1
    return total

print(sum_to_n_while(10))
```

```
55
55
```

Task Description #5: Classes (Bank Account Class)

Scenario

You are designing a basic banking application.
Task
• Use AI tools to generate a Bank Account class with methods such as deposit(), withdraw(), and check_balance().
• Analyze the AI-generated class structure and logic.
• Add meaningful comments and explain the working of the code.

```python
#Task5
#generate a Bank Account class with methods such as deposit(), withdraw(), and check_balance().

class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print("Deposited:", amount)

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print("Withdrawn:", amount)
        else:
            print("Insufficient balance")

    def check_balance(self):
        print("Current Balance:", self.balance)


# Object creation
account = BankAccount("Vyshnavi", 5000)
account.deposit(2000)
account.withdraw(1500)
account.check_balance()
```

```
Deposited: 2000
Withdrawn: 1500
Current Balance: 5500
```