

## ASSIGNMENT-6.1

HT No:2303A51585

Batch No:28

### Task Description1: AI-Based Code Completion for Loops

**Task:** Use an AI code completion tool to generate a loop-based program.

**Prompt:**

Generate Python code to print all even numbers between 1 and N using a loop.

**AI Generated Code:**

```
#Generate Python code to print all even numbers between 1 and N using a Loop.  
def even_numbers(N):  
    for num in range(1, N + 1):  
        if num % 2 == 0:  
            print(num)
```

```
#Generate Python code to print all even numbers between 1 and N using a Loop.  
def even_numbers(N):  
    for num in range(1, N + 1):  
        if num % 2 == 0:  
            print(num)  
# Example usage  
N = 20  
even_numbers(N)
```

**Output:**

```
● 2  
4  
6  
8  
10  
12  
14  
16  
18  
20
```

### Explanation:

- 1.AI suggested using a for loop to iterate from 1 to N
- 2.AI suggested checking even numbers using the modulo operator %
- 3.If the number is divisible by 2, it is printed
- 4.The logic correctly prints all even numbers in the given range

### Task Description2: AI-Based Code Completion for Loop with Conditionals

**Task:** Use an AI code completion tool to combine loops and conditionals.

#### Prompt:

Generate Python code to count how many numbers in a list are even and odd.

#### AI Generated Code:

```
#Generate Python code to count how many numbers in a List are even and odd.
def even_odd_count(numbers):    Expected "("
    even_count = 0
    odd_count = 0
    for num in numbers:
        if num % 2 == 0:
            even_count += 1
        else:
            odd_count += 1
    return even_count, odd_count
```

```
#Generate Python code to count how many numbers in a List are even and odd.
def even_odd_count(numbers):
    even_count = 0
    odd_count = 0
    for num in numbers:
        if num % 2 == 0:
            even_count += 1
        else:
            odd_count += 1
    return even_count, odd_count
# Example usage
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_count, odd_count = even_odd_count(numbers)
print(f"Even numbers: {even_count}, Odd numbers: {odd_count}")
```

**Output:**

```
Assisant_Coding/.venv/Scripts/python.exe  C:/Users/DELL
Even numbers: 5, Odd numbers: 5
(C:\Users\DELL\OneDrive\Desktop\LEARNING COURSES)
```

**Explanation:**

- 1.AI suggested using a for loop to traverse the list
- 2.AI suggested an if-else condition to distinguish even and odd numbers
- 3.Counters are incremented based on condition result
- 4.The AI-generated logic correctly counts even and odd values

**Task Description3: AI-Based Code Completion for Class Attributes Validation**

**Task:** Use an AI tool to complete a Python class that validates user input.

**Prompt:**

Generate a Python class User that validates age and email using conditional statements.

**AI Generated Code:**

```
#Generate a Python class User that validates age and email using conditional statements.
class User:
    def __init__(self, name, age, email):
        self.name = name
        self.age = age
        self.email = email
        self.validate_age()
        self.validate_email()

    def validate_age(self):
        if self.age < 0 or self.age > 120:
            raise ValueError("Age must be between 0 and 120.")

    def validate_email(self):
        if "@" not in self.email or "." not in self.email.split("@")[-1]:
            raise ValueError("Invalid email address.")
```

```

#Generate a Python class User that validates age and email using conditional statements.
class User:
    def __init__(self, name, age, email):
        self.name = name
        self.age = age
        self.email = email
        self.validate_age()
        self.validate_email()

    def validate_age(self):
        if self.age < 0:
            raise ValueError("Age cannot be negative.")
        elif self.age < 18:
            print(f"{self.name} is a minor.")
        else:
            print(f"{self.name} is an adult.")

    def validate_email(self):
        if "@" not in self.email or "." not in self.email.split("@")[-1]:
            raise ValueError("Invalid email address.")
        else:
            print(f"{self.name}'s email is valid.")

# Example usage
user1 = User("Alice", 17, "alice@gmail.com")
user2 = User("Bob", 25, "bob@gmail.com")

```

### Output:

```

(.venv) PS C:\Users\DELL\OneDrive\Desktop\LEARNING COURSES\AI_Assistant_Coding> & "C:/Assistant_Coding/.venv/Scripts/python.exe" "c:/Users/DELL/OneDrive/Desktop/LEARNING CO
Alice is a minor.
Alice's email is valid.
Bob is an adult.
Bob's email is valid.

```

### Explanation:

1. AI suggested validating age using a conditional check ( $\text{age} \geq 18$ )
2. AI suggested simple email validation using string checks
3. Invalid inputs are handled using conditionals
4. AI-generated logic works but can be improved with advanced validation

### Task Description4: AI-Based Code Completion for Classes

**Task:** Use an AI code completion tool to generate a Python class for managing student details.

### Prompt:

Generate a Python class Student with attributes (name, roll number, marks) and methods to calculate total and average marks.

### AI Generated Code:

```
#Generate a Python class Student with attributes (name, roll number, marks) and methods to calculate total and average marks.
def class Student:    Expected function name after "def"
    def __init__(self, name, roll_number, marks):    Expected indented block
        self.name = name
        self.roll_number = roll_number
        self.marks = marks

    def total_marks(self):
        return sum(self.marks)

    def average_marks(self):
        return self.total_marks() / len(self.marks) if self.marks else 0
```

```
#Generate a Python class Student with attributes (name, roll number, marks) and methods to calculate total and average marks.
class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks

    def total_marks(self):
        return sum(self.marks)

    def average_marks(self):
        return self.total_marks() / len(self.marks) if self.marks else 0
# Example usage
student = Student("Alice", 1, [85, 90, 78, 92])
print(f"Total Marks: {student.total_marks()}")
print(f"Average Marks: {student.average_marks()}")
```

### Output:

```
(.venv) PS C:\Users\DELL\OneDrive\Desktop\LEARNING COURSES\AI_Assistant_Coding> & "C:/Users/DELL/OneDrive/Scripts/python.exe" "c:/Users/DELL/OneDrive/Desktop/LEARNING COURSES/AI_Assistant_Coding/assignment-5.1
Total Marks: 345
Average Marks: 86.25
```

### Verification & Improvement

1. AI suggested class structure and methods
2. Manual improvement: reused total\_marks() for better readability
3. Code is complete and logically correct

### Explanation :

1. AI suggested storing student data using class attributes
2. AI suggested calculating total using sum()

3.Average is calculated using total divided by number of subjects

### Task Description5: AI-Assisted Code Completion Review

**Task:** Use an AI tool to generate a complete Python program using classes, loops, and conditionals together.

#### Prompt:

Generate a Python program for a simple bank account system using class, loops, and conditional statements.

#### AI Generated Code:

```
#Generate a Python program for a simple bank account system using class, loops, and conditional statements.
class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited: {amount}. New Balance: {self.balance}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"Withdrew: {amount}. New Balance: {self.balance}")
            else:
                print("Insufficient funds.")
        else:
            print("Withdrawal amount must be positive.")

    def display_balance(self):
        print(f"Account Holder: {self.account_holder}, Balance: {self.balance}")
```

```
#Generate a Python program for a simple bank account system using class, loops, and conditional statements.
class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited: {amount}. New Balance: {self.balance}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"Withdrew: {amount}. New Balance: {self.balance}")
            else:
                print("Insufficient funds.")
```

```

def withdraw(self, amount):
    else:
        print("Withdrawal amount must be positive.")

def display_balance(self):
    print(f"Account Holder: {self.account_holder}, Balance: {self.balance}")

# Example usage
account = BankAccount("John Doe", 1000)
while True:
    print("\nOptions: 1. Display Balance 2. Deposit 3. Withdraw 4. Exit")
    choice = input("Enter your choice (1-4): ")
    if choice == '1':
        account.display_balance()
    elif choice == '2':
        amount = float(input("Enter amount to deposit: "))
        account.deposit(amount)
    elif choice == '3':
        amount = float(input("Enter amount to withdraw: "))
        account.withdraw(amount)
    elif choice == '4':
        print("Exiting the program.")
        break
    else:
        print("Invalid choice. Please try again.")

```

### Output:

```

Options: 1. Display Balance 2. Deposit 3. Withdraw 4. Exit
Enter your choice (1-4): 2
Enter amount to deposit: 2000
Deposited: 2000.0. New Balance: 3000.0

Options: 1. Display Balance 2. Deposit 3. Withdraw 4. Exit
Enter your choice (1-4): 3
Enter amount to withdraw: 1000
Withdrew: 1000.0. New Balance: 2000.0

Options: 1. Display Balance 2. Deposit 3. Withdraw 4. Exit
Enter your choice (1-4): 1
Account Holder: John Doe, Balance: 2000.0

Options: 1. Display Balance 2. Deposit 3. Withdraw 4. Exit
Enter your choice (1-4): 4
Exiting the program.

```

### Explanation:

1. AI suggested combining class, loop, and conditionals
2. Menu-driven loop allows repeated operations
3. Conditional statements handle user choices
4. AI-generated program is functional but basic

