

# AI Assisted Coding

Week3 – Wednesday

**Name:** Lalu prasad Aroori

**Roll No.:** 2303A51948

**Batch No.:** 07

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

Week 3 – Wednesday

---

### Task Description #1: Classes (Student Class)

#### 1) Prompt to be given (Humanized – VS Code / AI Tool)

Complete a Python Student class with attributes name, roll number, and branch. Include a constructor and a method display\_details() that prints student information.

---

#### 2) Code

```
class Student:
```

```
    def __init__(self, name, roll_number, branch):
```

```
        self.name = name
```

```
        self.roll_number = roll_number
```

```
        self.branch = branch
```

```
    def display_details(self):
```

```
        print("Name:", self.name)
```

```
        print("Roll Number:", self.roll_number)
```

```
        print("Branch:", self.branch)
```

```
# Object creation

student1 = Student("Anita Rao", 101, "CSE")

student1.display_details()
```

---

### 3) Output

Name: Anita Rao

Roll Number: 101

Branch: CSE

---

### 4) Remarks

- AI correctly generated a class with constructor and method.
  - Code is readable and logically structured.
  - Uses proper object-oriented principles.
  - Suitable for beginner-level student management systems.
- 
- 

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

### A) Using for Loop

#### 1) Prompt to be given (Humanized)

Write a Python function that prints the first 10 multiples of a given number using a for loop.

---

#### 2) Code

```
def print_multiples_for(num):
    for i in range(1, 11):
        print(num * i)
```

```
print_multiples_for(5)
```

---

### **3) Output**

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

---

## **B) Using while Loop**

### **1) Prompt to be given (Humanized)**

Generate the same program to print the first 10 multiples of a number,  
but use a while loop instead of a for loop.

---

### **2) Code**

```
def print_multiples_while(num):
    i = 1
    while i <= 10:
        print(num * i)
        i += 1
```

```
print_multiples_while(5)
```

---

#### 4) Remarks

- Both loops produce correct output.
  - for loop is shorter and more readable.
  - while loop provides more control but requires manual increment.
  - AI-generated logic is correct in both cases.
- 
- 

### Task Description #3: Conditional Statements (Age Classification)

#### A) Nested if-elif-else Approach

##### 1) Prompt to be given (Humanized)

Generate a Python function using if-elif-else statements to classify age as child, teenager, adult, or senior.

---

##### 2) Code

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

---

#### B) Alternative (Dictionary-Based Logic)

##### 1) Prompt to be given (Humanized)

Generate an alternative age classification approach using simplified logic or dictionary-based conditions.

---

## 2) Code

```
def classify_age_alt(age):  
    if age < 0:  
        return "Invalid Age"  
  
    categories = {  
        "Child": age < 13,  
        "Teenager": 13 <= age < 20,  
        "Adult": 20 <= age < 60,  
        "Senior": age >= 60  
    }  
  
    for category, condition in categories.items():  
        if condition:  
            return category
```

---

## 3) Output

Input: 10 → Child

Input: 17 → Teenager

Input: 35 → Adult

Input: 65 → Senior

---

## 4) Remarks

- Conditions are clear and logically correct.

- if-elif-else is simpler for beginners.
  - Dictionary approach improves scalability.
  - AI-generated alternatives are efficient.
- 
- 

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

### **A) Using for Loop**

#### **1) Prompt to be given (Humanized)**

Write a Python function to calculate the sum of the first n natural numbers using a for loop.

---

#### **2) Code**

```
def sum_to_n_for(n):  
    total = 0  
    for i in range(1, n + 1):  
        total += i  
    return total
```

---

### **B) Using while Loop**

#### **1) Prompt to be given (Humanized)**

Generate an alternative implementation using a while loop or a mathematical formula.

---

#### **2) Code**

```
def sum_to_n_while(n):  
    total = 0  
    i = 1  
    while i <= n:
```

```
total += i  
  
i += 1  
  
return total
```

---

### 3) Output

Input: 10

For Loop Output: 55

While Loop Output: 55

---

### 4) Remarks

- Both implementations are correct.
  - Mathematical formula would be most efficient.
  - Loop-based solutions are easier to understand.
  - AI suggestions are logically sound.
- 
- 

## Task Description #5: Classes (Bank Account Class)

### 1) Prompt to be given (Humanized – VS Code)

Generate a Python Bank Account class with methods deposit(), withdraw(), and check\_balance(). Include comments explaining the working of each method.

---

### 2) Code

```
class BankAccount:  
  
    def __init__(self, account_holder, balance=0):  
  
        # Initialize account holder name and balance  
  
        self.account_holder = account_holder  
  
        self.balance = balance
```

```
def deposit(self, amount):  
    # Add money to the account  
    if amount > 0:  
        self.balance += amount  
        print("Deposited:", amount)  
    else:  
        print("Invalid deposit amount")
```

```
def withdraw(self, amount):  
    # Withdraw money if sufficient balance is available  
    if amount <= self.balance:  
        self.balance -= amount  
        print("Withdrawn:", amount)  
    else:  
        print("Insufficient balance")
```

```
def check_balance(self):  
    # Display current balance  
    print("Current Balance:", self.balance)
```

```
# Demonstration  
account = BankAccount("Rahul Kumar", 1000)  
account.deposit(500)  
account.withdraw(300)  
account.check_balance()
```

---

### 3) Output

Deposited: 500

Withdrawn: 300

Current Balance: 1200

---

### 4) Remarks

- AI-generated class structure is clear and functional.
  - Methods follow correct banking logic.
  - Comments improve transparency and understanding.
  - Suitable for basic banking applications.
- 
- 

### Overall Lab Conclusion

- AI-assisted code completion effectively generates classes, loops, and conditionals.
  - Generated code is readable, correct, and efficient.
  - Human review is necessary to verify edge cases and logic.
  - AI tools significantly speed up development and learning.
-