Day 2 – Control Statements in Python

*Trainer: Your Name – Senior Data Scientist*

Today’s session focuses on detailed understanding and practical application of control flow statements, crucial for logical decision-making and repetitive task automation in Python.

# 1. Understanding Control Statements

Control statements direct the execution flow based on conditions or loops. These are essential to building dynamic and responsive programs.

**Types of Control Statements:**

- Conditional Statements (`if`, `elif`, `else`)

- Loop Control (`for`, `while`, `break`, `continue`, `pass`)

# 2. Conditional Statements (`if`, `elif`, `else`)

## The `if` Statement

The simplest conditional statement in Python. It executes code based on a single condition being True.

**Syntax Example:**

age = 20  
if age >= 18:  
 print("You are eligible to vote.")

**Output:**

You are eligible to vote.

## The `elif` Statement

Used for checking additional conditions after an `if` condition fails. It stands for 'else if'.

**Example:**

score = 85  
if score >= 90:  
 print("Excellent")  
elif score >= 80:  
 print("Very Good")  
else:  
 print("Good")

**Output:**

Very Good

## The `else` Statement

Executes code when none of the previous conditions are True.

**Example:**

temperature = 15  
if temperature > 30:  
 print("Hot day")  
elif temperature > 20:  
 print("Nice weather")  
else:  
 print("Cold day")

**Output:**

Cold day

## Real-world Scenario using Conditional Statements

Conditional statements are frequently used in user authentication systems.

username = "user123"  
password = "pass123"  
  
input\_username = input("Enter username: ")  
input\_password = input("Enter password: ")  
  
if input\_username == username and input\_password == password:  
 print("Login successful.")  
else:  
 print("Invalid username or password.")

# 3. Looping Constructs (`for`, `while`)

## The `for` Loop

Iterates over a sequence (like a list, tuple, or string).

**Example:**

for number in range(1, 6):  
 print(number)

**Output:**

1  
2  
3  
4  
5

## The `while` Loop

Continues executing until a given condition becomes False.

count = 1  
while count <= 5:  
 print(count)  
 count += 1

**Output:**

1  
2  
3  
4  
5

# 4. Loop Control (`break`, `continue`, `pass`)

## The `break` Statement

Terminates the loop immediately.

for i in range(10):  
 if i == 5:  
 break  
 print(i)

**Output:**

0  
1  
2  
3  
4

## The `continue` Statement

Skips the current iteration and continues with the next.

for i in range(5):  
 if i == 3:  
 continue  
 print(i)

**Output:**

0  
1  
2  
4

## The `pass` Statement

Acts as a placeholder. Does nothing.

for i in range(3):  
 pass # placeholder for future code

# 5. Nested Control Structures

Using control structures inside other control structures for complex logic.

for i in range(1, 4):  
 for j in range(1, 4):  
 print(i, j)