**Chapter 4: Decision Flow Control Statements**

**Lecture Notes**

**Introduction:**

Decision flow control statements allow us to control the execution flow of our code based on certain conditions. In this chapter, we shall cover if-else statements, nested loops, various loop statements (while, do-while, for), and additional control statements like continue, break, and pass. These concepts are essential for mastering Python programming and are particularly useful in data science applications.

**1. If-Else Statement:**

The if-else statement is used to execute different blocks of code based on a condition. It allows us to create branching paths in our program. The general syntax of an if-else statement is as follows:

if condition:

# code to execute if condition is True

else:

# code to execute if condition is False

Example:

x = 10

if x > 5:

print("x is greater than 5")

else:

print("x is less than or equal to 5")

**2. Nested if statement:**

In Python, a nested if statement is an if statement inside another. It provides a way to check multiple conditions sequentially, allowing for more complex decision structures. Using nested ifs, you can execute a block of code if the outer condition is true and then check for further conditions within that block.

if condition1:

if condition2:

# Code to execute if both condition1 and condition2 are true

**Example:**

num = 6

if num > 0:

if num % 2 == 0:

print("Positive and even")

**3. Loop Statements:**

1. **For Loop:**

The for loop is used to iterate over a sequence (such as a list, tuple, or string) or other iterable objects. It allows us to perform a set of actions a specific number of times. Here's the general syntax of a for loop:

for item in iterable:

# code to execute for each item

Example:

fruits = ["apple", "banana", "orange"]

for fruit in fruits:

print(fruit)

1. **While Loop:**

The while loop allows us to repeatedly execute a block of code as long as a given condition is True. The loop continues until the condition becomes False. Here's the syntax of a while loop:

while condition:

# code to execute while condition is True

Example:

count = 0

while count < 5:

print(count)

count += 1

1. **Do-While Loop:**

Python doesn't have a built-in do-while loop like some other programming languages. However, we can achieve the same functionality using a while loop with a conditional check at the end. Here's an example:

count = 0

while True:

print(count)

count += 1

if count >= 5:

break

**4. Additional Control Statements:**

1. **Continue Statement:**

The continue statement is used within loops to skip the remaining code and move to the next iteration. It allows us to bypass certain conditions or iterations. Here's an example:

for i in range(5):

if i == 2:

continue

print(i)

1. **Break Statement:**

The break statement is used to exit the loop prematurely. It terminates the loop and moves the control to the next statement outside the loop. Here's an example:

for i in range(5):

if i == 2:

break

print(i)

1. **Pass Statement:**

The pass statement is used as a placeholder when we need a statement syntactically but don't want any code execution. It is commonly used in empty function or class definitions. Here's an example:

def my\_function():

pass