**AI Day 06 Notes**

**Syed Mansoor ul Hassan Bukhari**

**Detailed Notes on Loops in Python**

**Introduction to Loops**

In programming, loops are used to execute a block of statements repeatedly based on a condition. Python supports several types of loops:

* **For Loop**
* **While Loop**
* **Nested Loops**

**1. For Loop**

The `for` loop in Python is used to iterate over a sequence (such as a list, tuple, dictionary, set, or string). This loop is more efficient and readable when you need to perform an action on each item in a sequence.

**Syntax**

for item in sequence:

# Execute some statements

**Example: String Iteration**

name = "Muhammad"

for i in name:

print(i, end=", ")

if i == "u":

print("This is Amazing!")

**Example: List Iteration**

colors = ["Red", "Green", "Blue", "Yellow", "Magenta", "Violet", "Indigo"]

for color in colors:

print(color)

for char in color:

print(char)

**Using `range()` with For Loop**

The `range()` function generates a sequence of numbers, which is particularly useful when you want to execute a loop a specific number of times.

**Syntax**

range(start, stop, step)

**Examples**

for k in range(5):

print(k) # Prints 0 to 4

for k in range(1, 5):

print(k) # Prints 1 to 4

for k in range(-10, 10):

print(k) # Prints from -10 to 9

for k in range(0, -10, -1):

print(k) # Prints from 0 to -9

for k in range(1, 15, 2):

print(k) # Prints odd numbers up to 15

**2. While Loop**

**The `while` loop in Python repeatedly executes a block of code as long as a given condition is `True`.**

**Syntax**

while condition:

# Execute some statements

**Example**

x = 0

while x < 5:

print(x)

x += 1

print("Done with the Loop")

**Taking Input from User**

x = int(input("Enter the Value: "))

while x <= 38:

x = int(input("Enter the Value: "))

print(x)

print("Limit exceeds")

**Example with `else`**

x = 5

while x > 0:

print(x)

x -= 1

else:

print("I am Inside Else Block")

**3. Nested Loops**

A loop inside another loop is called a nested loop. This is useful for iterating over multi-dimensional data structures.

**Example**

for i in range(1, 4):

for j in range(1, 4):

print(i, j)

**Break and Continue Statements**

**Break Statement**

The `break` statement terminates the loop and transfers control to the statement immediately following the loop.

**Example**

for i in range(12):

print("5 X", i + 1, "=", 5 \* (i + 1))

if i == 9:

break

print("Left the loop and get out of the loop")

**Continue Statement**

The `continue` statement skips the rest of the code inside the loop for the current iteration and moves to the next iteration.

**Example**

for i in range(12):

if i == 9:

print("Skip the Iteration")

continue

print("5 X", i + 1, "=", 5 \* (i + 1))

**Task: Sum of Even Numbers from 1 to 100**

total = 0

for i in range(1, 101):

if i % 2 == 0:

total += i

print("Sum of even numbers from 1 to 100 is:", total)

**Examples for Practice**

# Print Numbers from 1 to 100

for i in range(1, 101):

print(i, end=" ")

if i == 50:

break

print("Mission Passed")

print("Thank You")

**Skip Even Numbers**

for i in [2, 3, 9, 4, 5, 6, 8, 0]:

if i % 2 == 0:

continue

print(i)

**# Infinite Loop with Break Condition**

i = 0

while True:

print(i)

i += 1

if i % 100 == 21:

break

**Conclusion**

Loops are a fundamental concept in programming that allow you to execute a block of code repeatedly. Understanding how to use `for` and `while` loops, along with control statements like `break` and `continue`, can greatly enhance your ability to write efficient and readable code.