Chapter-6

PYTHON LOOPS

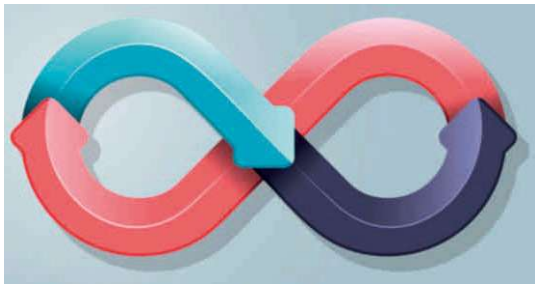
In this chapter we will look **Loops** in more detail. You may recall,

Loop (Iteration) Structure – In this structure, condition is asked over and over and over again until a certain task is complete.

Repetition – LOOPS are things that REPEAT. Like in the above picture, the iCodeT trainer will teach you the concept of computers until you understand it completely. In this "Your understanding" is a condition. Once you understand a particular topic then only the trainer will stop teaching it.

REPETITION IS BORING !

Let's do a simple exercise –



1. Count from 1 to 10
2. Again, Count from 1 to 10
3. Again, Count from 1 to 10
4. One more time. Count from 1 to 10
5. One more time. Count from 1 to 10
6. Are you tired & bored. If no. Count again from 1 to 10
7. One more time count from 1 to 10

For humans, doing the same thing over and over again is very boring. But, computer is our good friend and a machine.

Computers are programmed to do repetitive tasks.

Computer can easily and simply count from 1 to 10 for 7 times.

This process of telling computer to do same task repetitive is done using Loops in programming language.

Why Loops

- Loops allow software developers to use one command for a particular action and repeat the same for multiple times.
- Loops allow software developers to greatly reduce the programming time.
- Loops allow software developers to keep the code clean

Types of Loops – Broadly speaking, there are two types of Loops in computer programming

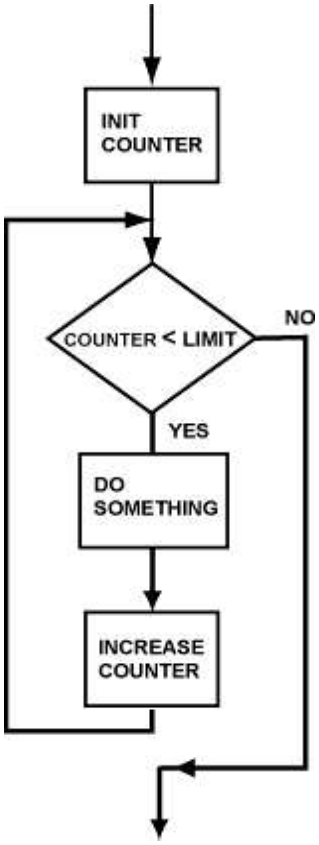
1. Counting Loops
2. Conditional Loops

Counting Loops – As the word suggests, in counting loop developer writes a code to tell the computer to **repeat** an action a ***certain number of times***.

As in the previous example of "Count 1 to 10 for 7 times". So this is counting loop for 7 counts.

Lets understand the same example using this flow chart.

1. Initial the counter INIT = 0. This means we have not started to count.
2. We check whether INIT < 7. Currently 0 so Yes
   a. Do something. This is the action of counting from 1 to 10
   b. Increase the counter. Now INIT = 1.
   c. Repeat Step 2
3. Once INIT is > 7, stop



In code.

```
for i in range(7):
    count from 1 to 10*
```

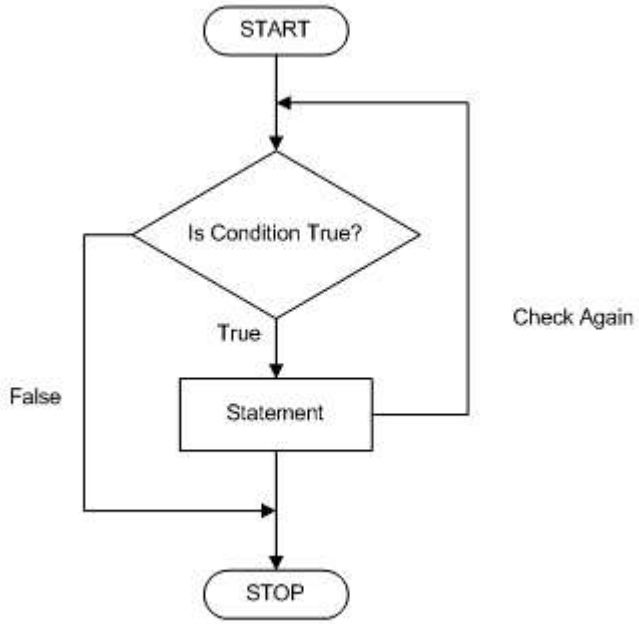
The code shown above is just for example. This is not a real code and is used to introduce "for loop"

Conditional Loops – As the word suggests, in conditional loop, developer writes a code to tell the computer to **repeat** an action until a ***condition is satisfied***.

As in the previous example of "Count 1 to 10 for 7 times". In this we are constantly checking whether the counting is done for 1 time or 2 time or 7 time. Once the counting is done for 7 times then only we stop.

Lets understand the same example using this flow chart.

1. Check if we have not counted yet for 7 times.
    a. If not. Count from 1 to 10.
2. If the condition at step 1 is false. That means we have counted for 7 times. We will break and stop.



In code.

```
While counting < 8
    count from 1 to 10*
    increase counting
```

- The code shown above is just for example.
- This is not a real code and is used to introduce "While loop"

Important Note – As you saw, the same example of counting from 1 to 10 for 7 times is achieved using **Counting loop (for)** as well as **Conditional loop (While).**

Both **for** and **while** are popular and have different advantages. We will learn them soon.

One more example

1. Reading a 100 page book
    - **Counting Loops**
        - You will read 100 pages of the book from page 1.
        - You will read 10 pages of the book from page 1.
    - **Conditional Loops**
        - You will start reading the book from page 1 and stop the book when its completed.
        - You will start reading the book from page 1 and stop when you will feel tired.
        - You will start reading the book from page 1 and stop when its midnight.
        - You will start reading the book from page 1 and stop when lights are turned off.
        - You will start reading the book from page 1 and stop when 3 chapters are completed.

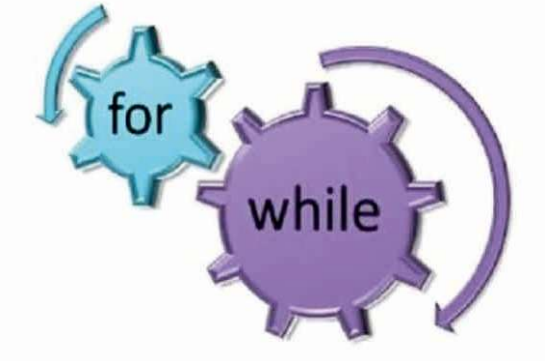
FOR vs WHILE Loop

We saw in our last example that **all** ***"for"*** loops can be **written** as ***"while"*** loops and vice-versa. The difference between **"for"** and **"while"** is that in **for loop** the **number of repetition is already known** to obtain a certain result.

Whereas **in while loop, the number of repetition** is **decided** by the **outcome of condition**. The loop breaks when a certain condition is false.

In other words, you should use

- ***for*** loops when you know how many times the loop should run i.e, **counting** loop.
- ***while*** loops when you require to break the loop on certain condition i.e, **condition** loop.



Fun Fact – <u>Use whichever loop seems more appropriate for a given problem</u>

Comparison		
Parameters	**For Loop**	**While Loop**
Type	Counting Loop	Conditional Loop
Use	For loop is used when we already know the number of iterations	While loop is used when we do not know the number of iterations
Format	Initialization Counter Condition Statement Increment Repeat	Condition Statement Repeat

Syntax	for (initialization; condition; iteration)
	{
	Statements;
	//body of for loop
	}
		while (condition)
		{
		statements;
		//body of while loop
		}

Example	sum = 0 for i in range(10): sum = sum + i print(sum)	sum = 0 i = 0 while (sum<=44): sum = sum + i i = i+1 print(sum)
Result	C:\Users\Work\Desktop>python sum.py 0 1 3 6 10 15 21 28 36 45 C:\Users\Work\Desktop>	C:\Users\Work\Desktop>python sum.py 0 1 3 6 10 15 21 28 36 45 C:\Users\Work\Desktop>

Examples

- ✓ Computer should ask your name and print it for 5 times
  Python Code -

```
name = input("Enter your name: ")
for i in range(5):
   print(name)
```

Enter your name: Rahul
Rahul
Rahul
Rahul
Rahul
Rahul

- ✓ Add a number 2 by 3 and replace it with the result. Do this till the final result is not 14
  Python Code-

```
num = 2
to_add = 3
while(num != 14):
   num = num + to_add
   print(num)
```

5
8
11
14

More Examples

Example 1: Printing Numbers using a While Loop

```
num = 1
while num <= 5:
    print(num)
    num += 1
```

Output

```
1
2
3
4
5
```

Explanation:

- This example demonstrates a **while** loop that **prints** numbers from **1** to **5**.
- The loop continues until the variable `num` becomes greater than **5**.
- Inside the loop, we **print** the value of `num` and **increment** it by **1** using the **+=** operator.

Example 2: Calculating the Sum of Numbers using a For Loop

```
numbers = [1, 2, 3, 4, 5]
sum = 0
for num in numbers:
    sum += num
print("Sum:", sum)
```

Output

```
Sum: 15
```

Explanation:

- In this example, we use a **for** loop to **iterate** over a list of numbers.
- For each iteration, the variable `num` takes the value of the current number, and we **add** it to the `sum` variable.
- After the loop, we **print** the final sum of all the numbers in the list.

Example 3: Printing Characters in a String using a For Loop

```python
message = "Hello"
for char in message:
    print(char)
```

Output

```
H
e
l
l
o
```

Explanation:

- In this example, we have a **for** loop that iterates over each character in the string variable `message`.
- For each iteration, the variable `char` **takes** the value of the current character, and we **print** it.
- The loop **continues until** all characters in the string have been printed.

Example 4: Calculating the Sum of Numbers using a While Loop

```python
numbers = [1, 2, 3, 4, 5]
sum = 0
index = 0

while index < len(numbers):
    sum += numbers[index]
    index += 1

print("Sum:", sum)
```

Output

```makefile
Sum: 15
```

Explanation:

- In this example, we have a **list** of numbers `[1, 2, 3, 4, 5]`.
- We initialize the variables `sum` and `index` to `0`. `sum` will hold the **running sum** of the numbers, and `index` will keep **track** of the **current index** in the list.
- The **while** loop continues as long as the `index` is **less** than the length of the `numbers` list.
- Inside the loop, we **add** the number at the current index to the running sum using the `+=` operator.
- We increment the `index` by `1` to **move** to the **next number** in the list.
- After the loop, we **print** the **final sum** of all the numbers in the list.

Example 5: Finding the Factorial of a Number using a while loop

```python
number = int(input("Enter a number: "))
factorial = 1
i = 1

while i <= number:
    factorial *= i
    i += 1

print("The factorial of", number, "is", factorial)
```

Output

```
Enter a number: 5
The factorial of 5 is 120
```

Explanation:

- In this example, we **take input** from the **user** to determine the number for which we want to calculate the factorial.
- We initialize the `factorial` variable to 1 and the `i` variable to 1.
- The **while** loop continues as long as `i` is **less** than or **equal** to the `number`.
- Inside the loop, we **multiply** the current value of `factorial` with `i` and update `factorial` accordingly.
- We increment `i` by 1 in each iteration.
- After the loop ends, we **print** the calculated factorial.

This example showcases how to use a while loop to calculate the factorial of a number. The loop iterates until a specific condition is met, allowing us to perform repetitive calculations.

Example 6: Printing a Pattern using a for loop

```python
rows = int(input("Enter the number of rows: "))

for i in range(1, rows + 1):
    for j in range(i):
        print("*", end=" ")
    print()
```

Output

```markdown
Enter the number of rows: 5
*
* *
* * *
* * * *
* * * * *
```

Explanation:

- In this example, we **take input** from the user to determine the number of rows in the pattern.
- The **outer for** loop **iterates** over the range from `1` to the `rows + 1`.
- Inside the outer loop, we have **another for** loop that iterates `i` times, where `i` represents the current row number.
- In each iteration of the **inner** loop, we **print** an **asterisk** followed by a space, using the `end=" "` parameter to ensure the asterisks are printed on the same line.
- After printing each row, we move to the next line using the `print()` statement without any arguments.

This example demonstrates how to use a for loop to print a pattern of asterisks in a triangular shape. **The nested loop** structure allows us to control the number of asterisks printed in each row based on the current row number.

Practice Program

1. Create a program to find the average of a list of numbers using a while loop. List contains 3 numbers
    a. my_list = [25, 1225, 15]

2. Ask the user for a sentence and count the number of words in that sentence using a for loop.

3. Ask the user for a number and determine if it is a perfect square or not using a while loop.

4. Print the sum of all odd numbers from 1 to 50 using a for loop.

5. Create a program to find the smallest number in a list using a for loop. List contains 10 numbers
    a. my_list = [25, 1225, 15, 7, 8, 51, 99, 132, 879, 62]