Python Loops

The following loops are available in Python to fulfil the looping needs. Python offers 3 choices for running the loops. The basic functionality of all the techniques is the same, although the syntax and the amount of time required for checking the condition differ.

We can run a single statement or set of statements repeatedly using a loop command.

The following sorts of loops are available in the Python programming language.

Sr.No.	Name of the loop	Loop Type & Description
1	**While loop**	Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
2	**For loop**	This type of loop executes a code block multiple times and abbreviates the code that manages the loop variable.
3	**Nested loops**	We can iterate a loop inside another loop.

The for Loop

Python's for loop is designed to repeatedly execute a code block while iterating through a list, tuple, dictionary, or other iterable objects of Python. The process of traversing a sequence is known as iteration.

Syntax of the for Loop

```
for value in sequence:
    { code block }
```

Code

Python program to show how the for loop works

```
numbers = [4, 2, 6, 7, 3, 5, 8, 10, 6, 1, 9, 2]
```

```
square = 0
squares = []
for value in numbers:
    square = value ** 2
    squares.append(square)
print("The list of squares is", squares)
```

Code

Python program to show how if-else statements work

```
string = "Python Loop"
for s in a string:
        if s == "o":
        print("If block")
    else:
        print(s)
```

Code

Python program to show how to use else statement with for loop

```
tuple_ = (3, 4, 6, 8, 9, 2, 3, 8, 9, 7)
for value in tuple_:
    if value % 2 != 0:
        print(value)
else:
    print("These are the odd numbers present in the tuple")
```

The range() Function

With the help of the range() function, we may produce a series of numbers. range(10) will produce values between 0 and 9. (10 numbers).

Code

Python program to show the working of range() function

```
print(range(15))
print(list(range(15)))
print(list(range(4, 9)))
print(list(range(5, 25, 4)))
```

Code

Python program to iterate over a sequence with the help of indexing

```
tuple_ = ("Python", "Loops", "Sequence", "Condition", "Range")
for iterator in range(len(tuple_)):
    print(tuple_[iterator].upper())
```

While Loop

While loops are used in Python to iterate until a specified condition is met. However, the statement in the program that follows the while loop is executed once the condition changes to false.

Syntax of the while loop is:

```
while <condition>:
    { code block }
```

Program code 1:

Now we give code examples of while loops in Python for printing numbers from 1 to 10. The code is given below -

```
i=1
while i<=10:
    print(i, end=' ')
    i+=1
```

Program Code 2:

Now we give code examples of while loops in Python for Printing those numbers divisible by either 5 or 7 within 1 to 50 using a while loop. The code is given below -

```
i=1
while i<51:
   if i%5 == 0 or i%7==0 :
      print(i, end=' ')
   i+=1
```

Code

Python program to show how to use a while loop

```
counter = 0
while counter < 10:
   counter = counter + 3
   print("Python Loops")
```

Code

```
#Python program to show how to use else statement with the while loop
counter = 0
while (counter < 10):
   counter = counter + 3
   print("Python Loops") else:
   print("Code block inside the else statement")
```

Single statement while Block

Code

Python program to show how to write a single statement while loop

```
counter = 0
while (count < 3): print("Python Loops")
```

Continue Statement

Code

Python program to show how the continue statement works

```
for string in "Python Loops":
    if string == "o" or string == "p" or string == "t":
        continue
    print('Current Letter:', string)
```

Break Statement

It stops the execution of the loop when the break statement is reached.

Code

Python program to show how the break statement works

```
for string in "Python Loops":
    if string == 'L':
        break
    print('Current Letter: ', string)
```

Program Code:

Now we give code examples of while loops in Python for a number is Prime number or not. The code is given below -

```
num = [34, 12, 54, 23, 75, 34, 11]
def prime_number(number):
    condition = 0
    iteration = 2
    while iteration <= number / 2:
        if number % iteration == 0:
            condition = 1
            break
        iteration = iteration + 1
    if condition == 0:
        print(f"{number} is a PRIME number")
    else:
        print(f"{number} is not a PRIME number")
for i in num:
    prime_number(i)
```

Multiplication Table using While Loop

Program Code:

In this example, we will use the while loop for printing the multiplication table of a given number. The code is given below -

```
num = 21
counter = 1
```

```
print("The Multiplication Table of: ", num)
while counter <= 10:
    ans = num * counter
    print (num, 'x', counter, '=', ans)
    counter += 1
```

Program Code 2:

Now we give code examples of while loops in Python for determine odd and even number from every number of a list. The code is given below -

```
list_ = [3, 4, 8, 10, 34, 45, 67,80]
index = 0
while index < len(list_):
    element = list_[index]
    if element % 2 == 0:
        print('It is an even number')
    else:
        print('It is an odd number')
    index += 1
```

Program Code:

Now we give code examples of while loops in Python for multiple condition. The code is given below -

```
num1 = 17
num2 = -12
while num1 > 5 and num2 < -5 :
```

```
    num1 -= 2
    num2 += 3
    print( (num1, num2) )
```

Let's look at another example of multiple conditions with an OR operator.

Code

```
num1 = 17
num2 = -12
while num1 > 5 or num2 < -5 :
    num1 -= 2
    num2 += 3
    print( (num1, num2) )
```

Python break statement

The break is a keyword in python which is used to bring the program control out of the loop. The break statement breaks the loops one by one.

The break is commonly used in the cases where we need to break the loop for a given condition. The syntax of the break statement in Python is given below.

Syntax:

```
#loop statements
break;
```

Example 1 :

break statement with for loop

Code

```
my_list = [1, 2, 3, 4]
count = 1
for item in my_list:
```

```
    if item == 4:
        print("Item matched")
        count += 1
        break
print("Found at location", count)
```

Example 2 :

Breaking out of a loop early

Code

```
my_str = "python"
for char in my_str:
    if char == 'o':
        break
    print(char)
```

Example 3:

break statement with while loop

Code

```
i = 0;
while 1:
    print(i," ",end=""),
    i=i+1;
    if i == 10:
        break;
print("came out of while loop");
```

Example 4 :

break statement with nested loops

Code

```
n = 2
while True:
    i = 1
    while i <= 10:
        print("%d X %d = %d\n" % (n, i, n * i))
        i += 1
    choice = int(input("Do you want to continue printing the table? Press 0 for no: "))
    if choice == 0:
        print("Exiting the program...")
        break
    n += 1
print("Program finished successfully.")
```

Python continue Statement

Python continue keyword is used to skip the remaining statements of the current loop and go to the next iteration. In Python, loops repeat processes on their own in an efficient way. However, there might be occasions when we wish to leave the current loop entirely, skip iteration, or dismiss the condition controlling the loop.

Code

Python code to show example of continue statement looping from 10 to 20

```
for iterator in range(10, 21):
    if iterator == 15:
        continue
        print( iterator )
```

Python Continue Statements in while Loop

Code

```
  string = "Computer"
iterator = 0
while iterator < len(string):
    if string[iterator] == 'p':
        continue
        print(string[ iterator ])
    iterator += 1
```