ITERATIVE STATEMENT
LOOPING

ITERATIVE STATEMENT (LOOPING)

Python programming language provides the following types of loops to handle looping requirements. Python provides three ways for executing the loops. While all the ways provide similar basic functionality, they differ in their syntax and condition checking time.

1. While Loop:- In python, while loop is used to execute a block of statements repeatedly until a given a condition is satisfied. And when the condition becomes false, the line immediately after the loop in program is executed.

Syntax of while loop

```
While(Condition)
Statement.
```

EXAMPLES

The program below print the value of number if number is less than 6.

Example 1

```
number = 1
while (number < 6):
    print(number)
    number += 1
```

what will happen if i change number to hold the value of 0

Example 2.

```
#Python program to illustrate
# while loop
count = 0
while (count < 3):
    count = count + 1
    print("Welcome To the Python programming Class")
```

What are the outputs of the programs?

EXAMPLES

Example 3:- Write a python program that display ten (10) consecutive numbers from 1 to 10.

```
x=0
while(x<10):
  x+=1
  print(x)
```

How many times the program will iterates

Example 4:- Write a python program that sum up all consecutive numbers between the range of 1 to 10.

```
x=0
sum=0
while(x<=10):
  sum+=x
  x+=1
  summation = "The sum of 10 consecutive numbers is {}"
print(summation.format(sum))
```

To add even numbers between the range of 0 to 10 we then increment x by 2 (x+=2)

Output

The sum of 10 consecutive numbers are 55.

EXAMPLES

Example 5 : - Write a python program that print all even numbers between the range of 0 to 100.

```
even=0
while(even<100):
  even+=2
  print(even)
```

Example 6:- Modified the above program for odd numbers.

```
odd=1
while(odd<99):
  odd+=2
  print(odd)
```

While loop with else statement

- **Using else statement with while loops:** As discussed above, while loop executes the block until a condition is satisfied. When the condition becomes false, the statement immediately after the loop is executed.
  The else clause is only executed when your while condition becomes false. If you break out of the loop, or if an exception is raised, it won't be executed.

```
while condition:
    # execute these statements
else:
    # execute these statements
```

Example 7

Python program to illustrate

```
# combining else with while
count = 0
while (count < 3):
    count = count + 1
    print("Hello World")
else:
    print(“I am out of the loop")
```

Output

```
Hello Word
Hello Word
Hello Word
I am out of the loop
```

Single Statement with While loop

Just like the if block, if the while block consists of a single statement we can declare the entire loop in a single line as shown below:

Example 8:

```
#Python program to illustrate
# Single statement while block
count = 0
while (count == 0): print(“Welcome To python")
```

Output

```
Welcome To python.
```

Break Statement

With the break statement we can stop the loop even if the while condition is true:

Example 9:

#Exist the loop when i is 3

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

Output

```
1
2
3
```

Continue Statement

Continue statement is statement that stop the current iteration, and continue with the next:

Example 10:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

Output

```
1
2
4
5
6
```

FOR LOOP

- A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
- This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.
- With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Example 11: The program will print each fruit in a fruit list.

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

What is the output of the program?

Looping Through a String

Strings are iterable objects, they contain a sequence of characters:

Example 12: The program Loop through the letters in the word "banana":

```
for x in "banana":
  print(x)
```

```
Output
b
a
n
a
n
a
```

Break Statement

Example 13: With the break statement we can stop the loop before its looped through all the items:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Output

```
apple
Banana
```

Example 14:

Exit the loop when x is "banana", but this time the break comes before the print:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

Output

```
apple
```

Continue Statement

With the `continue` statement we can stop the current iteration of the loop, and continue with the next:

Example 15:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

Output

```
apple
cherry
```

Range() Functions

- To loop through a set of code a specified number of times, we can use the range() function,
- The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Example 16

```
for x in range(6):
  print(x)
```

Output

```
0
1
2
3
4
5
```

Parameter Range

The `range()` function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: `range(2, 6)`, which means values from 2 to 6 (but not including 6):

Example 17

```
for x in range(2, 6):
  print(x)
```

```
Output
2
3
4
5
```

Parameter Range

The `range()` function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter:

`range(2, 30, 3)`:

Example 18 A python program that increment the starting value which (2) by 3 and ends at 30.

```
for x in range(2, 30, 3):
  print(x)
```

Output

2

5

8

11

14

17

20

23

26

29

Else in For loop

The else keyword in a for loop specifies a block of code to be executed when the loop finished executing.

Example 19: Print all numbers from 0 to 5, and print a message when the loop ends.

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Output

0

1

2

3

4

5

Finally finished!

Break in For loop

Break the loop when x is 3, and see what happens with the else block:

Example 20:

```
for x in range(6):
  if x == 3: break
  print(x)
else:
  print("Finally finished!")
```

```
0
1
2
```

- A nested loop is a loop inside a loop.
- The "inner loop" will be executed one time for each iteration of the "outer loop":

Example 21: The program prints each course for every students.

```
course= ["csc221", "csc341"]
students = ["Hannatu", "sajidah", "Patince"]
for x in course:
  for y in students:
    print(course, students)
```

Output

```
['csc221', 'csc341'] ['Hannatu', 'sajidah', 'Patince']
['csc221', 'csc341'] ['Hannatu', 'sajidah', 'Patince']
['csc221', 'csc341'] ['Hannatu', 'sajidah', 'Patince']
['csc221', 'csc341'] ['Hannatu', 'sajidah', 'Patince']
['csc221', 'csc341'] ['Hannatu', 'sajidah', 'Patince']
['csc221', 'csc341'] ['Hannatu', 'sajidah', 'Patince']
```