Python While loop

The Python while loop allows a part of the code to be executed until the given condition returns false. It is also known as a pre-tested loop.

It can be viewed as a repeating if statement. When we don't know the number of iterations then the while loop is most effective to use.

The syntax is given below.

1. **while** expression:
2. statements

Here, the statements can be a single statement or a group of statements. The expression should be any valid Python expression resulting in true or false. The true is any non-zero value and false is 0.0M

694

OOPs Concepts in Java

1. i=1
2. #The **while** loop will iterate until condition becomes **false**.
3. While(i<=10):
4. print(i)
5. i=i+1

**Output:**

1

2

3

4

5

6

7

8

9

10

Example -2: Program to print table of given numbers.

1. i=1
2. number=0
3. b=9
4. number = **int**(input("Enter the number:"))
5. **while** i<=10:
6. print("%d X %d = %d \n"%(number,i,number\*i))
7. i = i+1

**Output:**

Enter the number:10

10 X 1 = 10

10 X 2 = 20

10 X 3 = 30

10 X 4 = 40

10 X 5 = 50

10 X 6 = 60

10 X 7 = 70

10 X 8 = 80

10 X 9 = 90

10 X 10 = 100

Infinite while loop

If the condition is given in the while loop never becomes false, then the while loop will never terminate, and it turns into the **infinite while loop.**

Any **non-zero** value in the while loop indicates an **always-true** condition, whereas zero indicates the always-false condition. This type of approach is useful if we want our program to run continuously in the loop without any disturbance.

Example 1

1. **while** (1):
2. print("Hi! we are inside the infinite while loop")

**Output:**

Hi! we are inside the infinite while loop

Hi! we are inside the infinite while loop

Example 2

1. var = 1
2. **while**(var != 2):
3. i = **int**(input("Enter the number:"))
4. print("Entered value is %d"%(i))

**Output:**

Enter the number:10

Entered value is 10

Enter the number:10

Entered value is 10

Enter the number:10

Entered value is 10

Infinite time

Using else with while loop

Python allows us to use the else statement with the while loop also. The else block is executed when the condition given in the while statement becomes false. Like for loop, if the while loop is broken using break statement, then the else block will not be executed, and the statement present after else block will be executed. The else statement is optional to use with the while loop. Consider the following example.

Example 1

1. i=1
2. **while**(i<=5):
3. print(i)
4. i=i+1
5. **else**:
6. print("The while loop exhausted")

Example 2

1. i=1
2. **while**(i<=5):
3. print(i)
4. i=i+1
5. **if**(i==3):
6. **break**
7. **else**:
8. print("The while loop exhausted")

**Output:**

1

2

In the above code, when the break statement encountered, then while loop stopped its execution and skipped the else statement.

Example-3 Program to print Fibonacci numbers to given limit

1. terms = **int**(input("Enter the terms "))
2. # first two intial terms
3. a = 0
4. b = 1
5. count = 0
7. # check **if** the number of terms is Zero or negative
8. **if** (terms <= 0):
9. print("Please enter a valid integer")
10. elif (terms == 1):
11. print("Fibonacci sequence upto",limit,":")
12. print(a)
13. **else**:
14. print("Fibonacci sequence:")
15. **while** (count < terms) :
16. print(a, end = ' ')
17. c = a + b
18. # updateing values
19. a = b
20. b = c
22. count += 1

**Output:**

Enter the terms 10

Fibonacci sequence:

0 1 1 2 3 5 8 13 21 34

Python for loop

The for **loop in Python** is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like list, tuple, or dictionary.

The syntax of for loop in python is given below.

1. **for** iterating\_var **in** sequence:
2. statement(s)

For loop Using Sequence

**Example-1: Iterating string using for loop**

1. str = "Python"
2. **for** i **in** str:
3. **print**(i)

**Output:**

P

y

t

h

o

n

**Example- 2: Program to print the table of the given number .**

1. list = [1,2,3,4,5,6,7,8,9,10]
2. n = 5
3. **for** i **in** list:
4. c = n\*i
5. **print**(c)

**Output:**

5

10

15

20

25

30

35

40

45

50s

**Example-4: Program to print the sum of the given list.**

1. list = [10,30,23,43,65,12]
2. sum = 0
3. **for** i **in** list:
4. sum = sum+i
5. **print**("The sum is:",sum)

**Output:**

The sum is: 183

For loop Using range() function

**The range() function**

The **range()** function is used to generate the sequence of the numbers. If we pass the range(10), it will generate the numbers from 0 to 9. The syntax of the range() function is given below.

**Syntax:**

1. range(start,stop,step size)

* The start represents the beginning of the iteration.
* The stop represents that the loop will iterate till stop-1. The **range(1,5)** will generate numbers 1 to 4 iterations. It is optional.
* The step size is used to skip the specific numbers from the iteration. It is optional to use. By default, the step size is 1. It is optional.

Consider the following examples:

**Example-1: Program to print numbers in sequence.**

1. **for** i **in** range(10):
2. **print**(i,end = ' ')

**Output:**

0 1 2 3 4 5 6 7 8 9

**Example - 2: Program to print table of given number.**

1. n = int(input("Enter the number "))
2. **for** i **in** range(1,11):
3. c = n\*i
4. **print**(n,"\*",i,"=",c)

**Output:**

Enter the number 10

10 \* 1 = 10

10 \* 2 = 20

10 \* 3 = 30

10 \* 4 = 40

10 \* 5 = 50

10 \* 6 = 60

10 \* 7 = 70

10 \* 8 = 80

10 \* 9 = 90

10 \* 10 = 100

**Example-3: Program to print even number using step size in range().**

1. n = int(input("Enter the number "))
2. **for** i **in** range(2,n,2):
3. **print**(i)

**Output:**

Enter the number 20

2

4

6

8

10

12

14

16

18

We can also use the **range()** function with sequence of numbers. The **len()** function is combined with range() function which iterate through a sequence using indexing. Consider the following example.

1. list = ['Peter','Joseph','Ricky','Devansh']
2. **for** i **in** range(len(list)):
3. **print**("Hello",list[i])

**Output:**

Hello Peter

Hello Joseph

Hello Ricky

Hello Devansh

Nested for loop in python

Python allows us to nest any number of for loops inside a **for** loop. The inner loop is executed n number of times for every iteration of the outer loop. The syntax is given below.

**Syntax**

1. **for** iterating\_var1 **in** sequence:  #outer loop
2. **for** iterating\_var2 **in** sequence:  #inner loop
3. #block of statements
4. #Other statements

Example- 1: Nested for loop

1. # User input for number of rows
2. rows = int(input("Enter the rows:"))
3. # Outer loop will print number of rows
4. **for** i **in** range(0,rows+1):
5. # Inner loop will print number of Astrisk
6. **for** j **in** range(i):
7. **print**("\*",end = '')
8. **print**()

**Output:**

Enter the rows:5

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

Example-2: Program to number pyramid.

1. rows = int(input("Enter the rows"))
2. **for** i **in** range(0,rows+1):
3. **for** j **in** range(i):
4. **print**(i,end = '')
5. **print**()

Output:->>

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5