

Looping Control Statements

Looping control statements are used to repeat one or more than statement number of times or until given condition.

Python support 2 types of looping control statements

1. for loop
2. while loop

for loop

“for” is a keyword in python which represents for loop.

The for statement is used to iterate over the elements of a sequence (such as a string, tuple or list) or other iterable object:

Syntax:

for variable-name in iterable:

 statement-1

 statement-2

for loop each time read value generated by iterable and assign to variable. After reading value it execute block of statements (statement-1,statement-2)

(OR)

Statement-1,statement-2 are repeated until read all the values from iterable/collection.

range datatype

range is an immutable sequence data type.

The range type represents an immutable **sequence of numbers** and is commonly used for looping a specific number of times in for loops.

The range type is used to generate sequence of integer values, which are used to repeat for loop number of times (OR) it is used to represent data for other collection data types.

range type is used to generate sequence of integers in increment order or decrement order.

Syntax1: range(stop)

Syntax2: range(start,stop,step)

range data type required 3 inputs

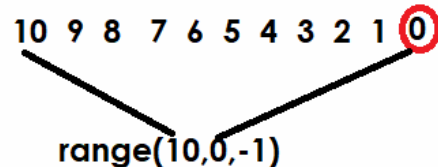
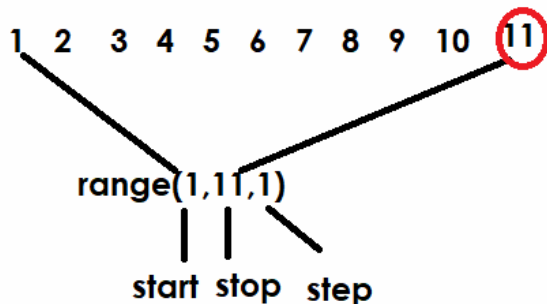
1. start (included)
2. stop (excluded)
3. step

start : start is an integer value, which represents starting value of range, which is included. This value can be +ve or -ve

stop: stop is an integer value, which represents ending value of range, which is not included/excluded. This value can be +ve or -ve

step: step is an integer value, which represents difference between values within range. It represents increment value or decrement value. This value should not be 0. This value can be +ve or -ve

Syntax2: range(start,stop,step=1)



In range always start,stop values are given based on step value.

If step +ve, start<stop

If step -ve, start>stop

Note: step should not be zero, if step is zero it raises ValueError

-1 -2 -3 -4 -5 **-6**
range(-1,-6,-1)

-5 -4 -3 -2 -1 **0**
range(-5,0,1)

1 3 5 7 9 11 **12**
range(1,12,2)

2 4 6 8 10 12 **13**
range(2,13,2)

-5 -4 -3 -2 -1 0 1 2 3 4 5 **6**
range(-5,6,1)

5 4 3 2 1 0 -1 -2 -3 -4 -5 **-6**
range(5,-6,-1)

Note: if step value is not given, the default step value is +1

Example:

```
for x in range(1,6): # 1 2 3 4 5  
    print("Hello")
```

```
for x in range(1,6): # 1 2 3 4 5  
    print(x,end=' ')
```

Output

Hello
Hello
Hello

Hello
Hello
1 2 3 4 5

Example:

```
for n in range(1,11,1):  
    print(n,end=' ')
```

```
print()  
for n in range(10,0,-1):  
    print(n,end=' ')
```

```
print()  
for n in range(-1,-11,-1):  
    print(n,end=' ')
```

```
print()  
for n in range(-10,0,1):  
    print(n,end=' ')
```

```
print()  
for n in range(1,11,2):  
    print(n,end=' ')
```

```
print()  
for n in range(2,11,2):  
    print(n,end=' ')
```

```
print()  
for n in range(11,0,-2):  
    print(n,end=' ')
```

```
print()  
for n in range(10,0,-2):  
    print(n,end=' ')
```

```
print()  
for n in range(-5,6,1):  
    print(n,end=' ')  
print()
```

```
for n in range(5,-6,-1):  
    print(n,end=' ')
```

```
print()
```

Output

```
1 2 3 4 5 6 7 8 9 10  
10 9 8 7 6 5 4 3 2 1  
-1 -2 -3 -4 -5 -6 -7 -8 -9 -10  
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1  
1 3 5 7 9  
2 4 6 8 10  
11 9 7 5 3 1  
10 8 6 4 2  
-5 -4 -3 -2 -1 0 1 2 3 4 5  
5 4 3 2 1 0 -1 -2 -3 -4 -5
```

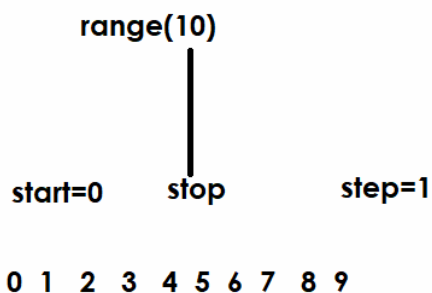
Syntax-1: range(stop)

This syntax takes default start and step values

Default start value 0

Default step value 1

This allows to generate sequence of +ve integer values



Example:

```
for n in range(10):  
    print(n,end=' ')
```

Output

```
0 1 2 3 4 5 6 7 8 9
```

