Python range() function

The built-in function `range()` generates the **integer numbers between the given start integer to the stop integer.** Using `for` loop, we can iterate over a sequence of numbers produced by the `range()` function.

range() function syntax and arguments

```
range(start, stop[, step])
```

It takes three arguments. Out of the three 2 arguments are optional. I.e., start and step are the optional arguments.

1. A **start** argument is a starting number of the sequence. i.e., lower limit. By default, it starts with 0 if not specified.
2. A **stop** argument is an upper limit. i.e., generate numbers up to this number, The `range()` doesn't include this number in the result.
3. The **step** is a difference between each number in the result. The default value of the step is 1 if not specified.

range() function Examples

Let see all the possible scenarios now. Below are the **three variant of range()** function.

Example one – Using only one argument

```
# Print first 5 numbers using range function
for i in range(5):
    print(i, end=', ')
```

Output:

```
0, 1, 2, 3, 4,
```

Only a stop argument is passed to `range()` . So by default, it takes `start = 0` and `step = 1`.

Example Two – using two arguments (i.e., start and stop)

```
# Print integers within given start and stop number using range()
for i in range(5, 10):
    print(i, end=', ')
```

Output:

```
5, 6, 7, 8, 9,
```

Note: By default, it took step value as 1.

Example Three – using all three arguments

```
# using start, stop, and step arguments in range()
print("Printing All even numbers between 2 and 10 using range()")
for i in range(2, 10, 2):
    print(i, end=', ')
```

Output:

```
Printing All even numbers between  and 10 using range()

2, 4, 6, 8,
```

All three arguments are specified i.e., `start = 2`, `stop = 10`, `step = 2`. The step value is 2 so the difference between each number is 2.

Points to remember about range() function arguments

- `range()` only works with the integers. **All arguments must be integers**. You can not use float number or any other type in a start, stop and step argument of a **`range()`**.
- All three arguments can be positive or negative.

- The step value must not be zero. If a step is zero Python raises a **ValueError** exception.

for i in range – for loop with range()

As you know for loop executes a block of code or statement repeatedly for the fixed number of times. Using for loop we can iterate over a sequence of numbers produced by the `range()` function. Let's see how to use for loop and `range()` function to **print the odd numbers between 1 and 10**. Using this example, we can understand how `i` is getting its value when we use `range()` and for loop together.

```
for i in range(1, 10, 2):
    print("Current value of i is:", i)
```

Output:

```
Current value of i is: 1

Current value of i is: 3

Current value of i is: 5

Current value of i is: 7

Current value of i is: 9
```

Inclusive range

In this section, we will learn how to generate an inclusive range. The **`range(n)` is of exclusive nature** that is why it doesn't include the last number in the output. i.e., The given endpoint is never part of the generated result. For example, `range(0, 5) = [0, 1, 2, 3, 4]`. The result contains numbers from 0 to up to 5 but not 5 and the total count is 5. The `range(start, stop)` not include stop number in the output because the index (i) always starts with 0 in Python.

If you want to include the last number in the output i.e., If you want an inclusive range then set `stop` argument value as `stop+step`.

Inclusive range() example.

```
# Printing inclusive range
start = 1
stop  = 5
step  = 1
stop +=step #now stop is 6

for i in range(start, stop, step):
    print(i, end=', ')
```

Output:

```
1, 2, 3, 4, 5,
```

Example 2

```
# Printing inclusive range
start = 2
stop  = 10
step  = 2
stop +=step #now stop is 12

for i in range(start, stop, step):
    print(i, end=', ')
```

Output:

```
2, 4, 6, 8, 10,
```

Python range step

A step is an optional argument of a `range()`. The step is a **difference between each number** in the result sequence. If the step size is 2, then the difference between each number is 2. The default size of a step is **1** if not specified. We can perform lots of operations by effectively using step arguments such as reversing a sequence, printing negative ranges.

Decrementing with range() using a negative step

We can use negative values in all the arguments of `range()` function i.e., start, stop, and step.

```
start = -2
stop = -10
step = -2
print("Negative number range")
for number in range(start, stop, step):
    print(number, end=', ')
```

Output:

```
Negative number range

-2, -4, -6, -8,
```

Let's understand the above program, we set, `start = -2`, `stop = -10`, `step = -2`.

- In the 1st iteration of for loop, the result is `-2`
- In the 2nd iteration of for loop, the result is `-2, -4` because `-2+(-2) = -`4 and so on.
- And Last iteration output is `-2, -4, -6,-8`

Decrementing with the range from Negative to Positive number

Here in this example, we will learn how to use a step argument to display a range of numbers from negative to positive. Range of negative numbers.

```
# printing range from negative to positive
for num in range(-2, 5, 1):
    print(num, end=", ")
```

The output of the above program

```
-2, -1, 0, 1, 2, 3, 4,
```

Python range from Positive to Negative number

Here in this example, we can learn how to use step argument effectively to display numbers from positive to negative.

```
print (" printing range from Positive to Negative")
for num in range(2,-5,-1):
    print(num, end=", ")
```

Output:

```
printing range from Positive to Negative

2, 1, 0, -1, -2, -3, -4,
```

Convert range() to List

If you execute `print( type( range(10) ) )` you will get `<class 'range'>` as output. Python `range()` function doesn't return a list type. It returns a range object, i.e., sequence object of type range, So as a result, we get an immutable sequence object of integers.

We can convert the output of a `range()` to the Python list. **Use `list` class to convert range output to list**. Let's understand this with the following example.

```
print("Converting python range() to list")
even_list = list( range(2, 10, 2))
print("printing list", even_list)
```

Output:

```
Converting python range() to list

printing list [2, 4, 6, 8]
```

We can also use `range()` function to access Python list items using its index number.

```
print("Use of range() to access Python list using index number")
sample_list = [10, 20, 30, 40, 50]
for i in range(len(sample_list)):
  print("List item at index ", i, "is ", sample_list[i])
```

Output:

```
Use of range() to access Python list using index number

List item at index  0 is  10

List item at index  1 is  20

List item at index  2 is  30

List item at index  3 is  40

List item at index  4 is  50
```

Note: Using a `len(list)`, we can get a count of list items, We used this count in `range()` to iterate for loop fixed number of times.

Reverse range

If you want to print the sequence of numbers within range by descending order or reverse order in Python then its possible, there are two ways to do this.

The first is to use a negative or down step value. i.e., set the `step` argument of a `range()` to `-1`. For example, if you want to display a number sequence like [5, 4, 3, 2, 1, 0] i.e., we want reverse iteration or backward iteration of for loop with `range()` function.

Let's see how to **loop backward using indices** in Python to display a range of numbers from 5 to 0.

```
print ("Displaying a range of numbers by reverse order")
for i in range(5, -1, -1):
    print (i, end=', ')
```

Output:

```
Displaying a range of numbers by reverse order

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

Use the `reversed` function to reverse range in Python

Alternatively, using The `reversed()` function, we can reverse any sequence. If we use the `reversed()` function with `range()`, that will return a **`range_iterator`** that accesses the given range of numbers in the reverse order. The below example will let you know how to make a reverse for loop in Python.

```
print("Printing reverse range using reversed()")
for i in reversed(range(0, 5)):
    print(i)
```

Output:

```
Printing reverse range using reversed()

4

3

2

1

0
```

Check the **output type** if we use `range()` with `reversed()`

```
print("Checking the type")
print(type(range(0, 5)))
print(type(reversed(range(0,5))))
```

Output:

```
Checking the type

<class 'range'>

<class 'range_iterator'>
```

Also, If you need the list out of it, you need to convert the output of the `reversed()` function to list. So you can get the reverse list of ranges.

Print a list in reverse order with `range()`.

```
print("Printing list in reverse order with range")
reverseed_list = list(reversed(range(0, 5)))
print(reverseed_list)

print("Second example to reverse list with range")
reverse_list2 = list(range(5, -1, -1))
print(reverse_list2)

print("Third Example to reverse list with range")
reverse_list3 = list(range(2, 20, 2)[::-1])
print(reverse_list3)
```

Output:

```
Printing list in reverse order with range

[4, 3, 2, 1, 0]

Second example to reverse list with range

[5, 4, 3, 2, 1, 0]

Third Example to reverse list with range

[18, 16, 14, 12, 10, 8, 6, 4, 2]
```