**A Closer Look at the Range() Function**

The **in** keyword, when used with the **range()** function, generates a sequence of integer numbers, which can be used with a **for** loop to control the start point, the end point, and the incremental values of the loop.

**Syntax:**

for n in range(x, y, z):

    print(n)

The **range()** function uses a set of indices that point to integer values, which start at the number 0. The numeric values 0, 1, 2, 3, 4 correlate to ordinal index positions 1st, 2nd, 3rd, 4th, 5th. So, when a range call to the 5th index position is made using **range(5)** the index is pointing to the numeric value of 4.

| **Index Number** | **1st index** | **2nd index** | **3rd index** | **4th index** | **5th index** |
| --- | --- | --- | --- | --- | --- |
| Value | 0 | 1 | 2 | 3 | 4 |

The **range()** function can take up to three parameters:  **range(start, stop, step)**

**Start** The first item in the **range()** function parameters is the starting position of the range. The default is the first index position, which points to the numeric value 0. This value is included in the range.

**Stop** The second item in the **range()** function parameters is the ending position of the range. There is no default index position, so this index number must be given to the **range()** parameters. For example, the line **for n in range(4)** will loop 4 times with the **n** variable starting at 0 and looping 4 index positions: 0, 1, 2, 3. As you can see, **range(4)** (meaning index position 4) ends at the numeric value 3. In Python, this structure may be phrased as “the end-of-range value is *excluded* from the range.” In order to include the value 4 in  **range(4)**, the syntax can be written as **range(4+1)** or **range(5)**. Both of these ranges will produce the numeric values 0, 1, 2, 3, 4.

**Step** The third item in the **range()** function parameters is the incremental step value. The default increment is +1. The default value can be overridden with any valid increment. However, note that the loop will still end at the end-of-range index position, regardless of the incremental value. For example, if you have a loop with the range: **for n in range(1, 5, 6)**, the range will only produce the numeric value 1. This is because the incremental value of 6 exceeded the ending point of the range.

**Practice Exercise**

You can use the code block below to test the values of **n** with various **range()** parameters. A few suggestions to test are:

**range(stop)**

* range(3)
* range(3+1)

**range(start, stop)**

* range(2, 6)
* range(5,10+1)

**range(start, stop, step)**

* range(4, 15+1, 2)
* range(2\*2, 25, 3+2)
* range(10, 0, -2)

for n in range(1, 5, 6):

    print(n)

**Examples of the range() function in code:**

**Example 1**

# This loop iterates on the value of the "n" variable in a range

# of 0 to 10 (the value of the end-of-range index 11 is excluded).

# The incremental value for the loop is 2. The print() function will

# output the resulting value of "n" as the loop counts from 0 to 10

# (end-of-range index 11) in incremental steps of 2. This is one

# method that can be used in Python to print a list of even numbers.

for n in range(0,11,2):

    print(n)

# The loop should print 0, 2, 4, 6, 8, 10

**Example 2**

# This loop iterates on the value of the "number" variable in a range

# of 2 to 7+1 (the value of the end-of-range index 7 is excluded, so

# +1 has been added to the parameter to include the numeric value 7 in

# the range). The incremental value for the loop is the default of +1.

# The print() function will output the resulting value of "number"

# multiplied by 3.

for number in range(2,7+1):

    print(number\*3)

# The loop should print 6, 9, 12, 15, 18, 21

**Example 3**

# This loop iterates on the value of the "x" variable in a range

# of 2 to -1 (the end-of-range index -2 is excluded). The third

# parameter is also a negative number, making it a decremental value

# of -1. The print() function will output the resulting value of

# "x" as it starts at 2 and counts down to -1 (index -2).

for x in range(2, -2, -1):

    print(x)

# The loop should print 2, 1, 0, -1

**Key takeaways**

The roles of the **range(start, stop, step)** function parameters are:

* **Start** - Beginning of range
  + value included in range
  + default = 0
* **Stop** - End of range
  + value excluded from range (to include, use stop+1)
  + no default
  + must provide the ending index number
* **Step** - Incremental value
  + default = 1

**Resources for more information**

* [Python range() function](https://www.geeksforgeeks.org/python-range-function/) - This site provides some helpful visualizations for the range index positions. It also offers multiple **for** x **in** **range()** examples and practice exercises.

For additional Python practice, the following links will take you to several popular online interpreters and codepads:

* [Welcome to Python](https://www.python.org/shell/)
* [Online Python Interpreter](https://www.onlinegdb.com/online_python_interpreter)
* [Create a new Repl](https://repl.it/languages/python3)
* [Online Python-3 Compiler (Interpreter)](https://www.tutorialspoint.com/execute_python3_online.php)
* [Compile Python 3 Online](https://rextester.com/l/python3_online_compiler)
* [Your Python Trinket](https://trinket.io/python3)