Rex Jones II

Learn For Loops Step-By-Step

Python Video = https://youtu.be/kO6jlxrlmDs

For Loops

In this session, we are going to focus on Python For Loops. A For Loop allows us to cycle through a sequence. The sequence can be numbers, letters, strings. It can be a lot of things. The good part about a for loop is how we can automate a repetitive function.

In our IDE, we create a for loop by writing for. At this point, we need to define a temporary variable. It's temporary because the variable represents a different value each time our program iterate this for loop. letter is the variable in. Now, we need to mention the sequence or collection. Let's use a string with "Python Tutorials": This code line states for each letter in Python Tutorials; we are going to take some kind of action. The action we are going to take is print(letter) for every iteration. So, the console will print every letter.

```
for letter in "Python Tutorials":
    print(letter)
```

Run and we see Python Tutorials with each letter on a separate line.

```
P
y
t
h
o
n

T
u
```

That's how for loops work. The variable holds 1 letter at a time. The 1st iteration was 'P' and 2nd iteration was 'y'.

Now, let's use a list. The list can be some names = like ["Jane", "Joe", "John"]. Change letter to name so the code says for each name in names: print(name).

```
names = ["Jane", "Joe", "John"]
for name in names:
    print(name)
```

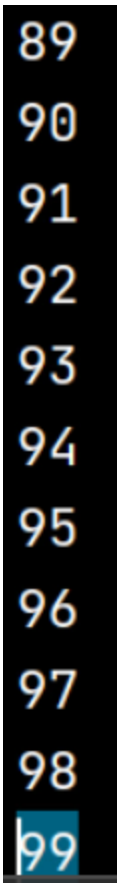
This will print each name. We see Jane, Joe, and John in the console.

```
Jane
Joe
John
```

For numbers, our sequence can be [1, 2, 3]. Run and we see 1, 2, 3. However, the best way to iterate a group of numbers is to use the range() function. Change for name to for number and print(number). The range() function is more efficient because we may have a scenario that have way more than 3 numbers like 100. When I hover range, the description shows “Return an object that produces a sequence of integers from start (inclusive) to stop (exclusive)”. That means, the range() function will print up to 100 so it will not include 100.

```
names = [1, 2, 3]
for number in range(100):
    print(number)
```

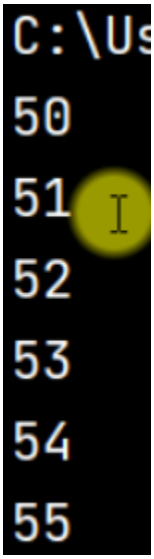
Run and we see 99 as the stopping and 0 as the starting point so we 0 all the way to 99.



Recall how the description for range showed it start (inclusive). We can add a number like 50, to start our sequence.

```
names = [1, 2, 3]
for number in range(50, 100):
  print(number)
```

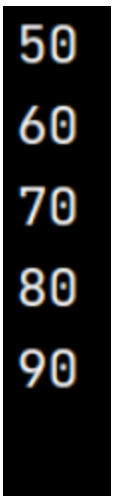
Run and now we see 50 all the way to 99.



There's one more feature to the range() function. It can also accept the number of steps. So Step tells the compiler what number to increment or decrement. If write ,10 after 100 then the for loop will increment by 10.

```
names = [1, 2, 3]
for number in range(50, 100, 10):
  print(number)
```

Run and we see 50, 60, 70, 80, 90.



Do you know what the console will show if we count backwards by 10? First, we have to change 50 to 100 so our counter start at 100 then stop at 50. Add a negative - so it executes backwards by 10.

```
names = [1, 2, 3]
for number in range(100, 50, -10):
  print(number)
```

We see 100, 90, 80, 70, and 60.

```
100
90
80
70
60
```

It does not include 50 because 50 is exclusive. That's it for Python for loops.

Contact Info

- ✔ Email Rex.Jones@Test4Success.org
- ✔ YouTube https://www.youtube.com/c/RexJonesII/videos
- ✔ Facebook https://facebook.com/JonesRexII
- ✔ Twitter https://twitter.com/RexJonesII
- ✔ GitHub https://github.com/RexJonesII/Free-Videos
- ✔ LinkedIn https://www.linkedin.com/in/rexjones34/