Over the past few modules we’ve been talking about using loops and their syntax in Python. As you have seen, a loop is an instruction that repeats multiple times as long as the condition set forth in the loop is met. You use loops to repeat a block of code. For example, if you want to send users an invitation, asking them to subscribe every time they open a new project, you would use a loop.

In this reading, you will learn the basics about **for** loops and how you can use them with strings.

## For ****Loop Recap****

As a reminder, **for** loops enable you to iterate over a sequence of values, such as numbers, names, or lines in a file. You can even iterate over a list of strings.

## ****What are strings?****

At the beginning of this course we identified strings as one of the basic data types used by Python and other programming languages. Strings represent a sequence of characters and are often used to display output to the user.

You can recognize a string because it’s surrounded by single or double quotes like the following examples:

"Hello world."

‘Ostriches can’t fly.’

"567.89"

**Pro tip**: You can’t mix single and double quotes in the same string or you’ll get a syntax error.

**Note**: In the examples above, Python does not consider “567.89” a number. It is a string because it is surrounded by quotes.

## ****Using**** for ****loops with strings****

In an earlier video, you learned that a **for** loop iterates over the items in a collection until each item in the collection has been looped over.

**Pro tip:** Every **for** loop can also be written as a while loop.

Let’s see what that looks like. In the example below, we’re using a **for** loop to print out the characters in the string “Hello.”

**>>> greeting = "Hello"**

**>>> for c in greeting:**

**>>>   print("The next character is: ", c)**

**The next character is: H**

**The next character is: e**

**The next character is: l**

**The next character is: l**

**The next character is: o**

Notice that the **for** loop will stop iterating after the “o” is printed. You can also use **for** loops with Python’s **range ()** function to generate a series of numbers, which you can see in the following example:

**>>> for i in range(0, 3):**

**>>>   print("The next value is:", i)**

**The next value is 0**

**The next value is 1**

**The next value is 2**

Practically every Python script you write will use strings, which we’ll talk about in more detail in Module 4. For now, it’s enough for you to know that they are immutable. In other words, once you create a string, it can’t be changed. But you can use **for** loops to create a new string.

You can use **for** loops with strings to perform tasks and functions such as:

* Reading text from a file
* Searching for a value or specific data in a document or spreadsheet

**Pro tip:**  If you modify or update a collection of items you are using with a **for** loop, make sure you modify your **for** loop as well.

## ****Key takeaways****

The real power of **for** loops is that they allow you to repeat sequences of any kind of data, not just a range of numbers or letters. They can help you find and use data faster. Over the next several videos and readings, you’ll get a chance to see what else you can do with loops and strings.