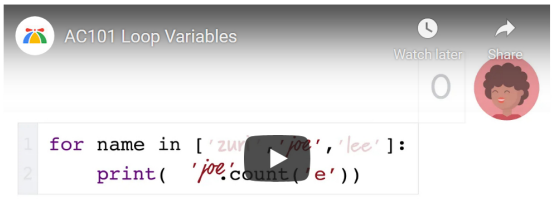
7.2. The **for** Loop



Watch on YouTube

Activity: 1 -- Video: (forloopvid)

Back when we drew the images with turtle it could be quite tedious. If we wanted to draw a square then we had to move then turn, move then turn, etc. etc. four times. If we were drawing a hexagon, or an octagon, or a polygon with 42 sides, it would have been a nightmare to duplicate all that code.

A basic building block of all programs is to be able to repeat some code over and over again. We refer to this repetitive idea as **iteration**. In this section, we will explore some mechanisms for basic iteration.

In Python, the **for** statement allows us to write programs that implement iteration. As a simple example, let's say we have some friends, and we'd like to send them each an email inviting them to our party. We don't quite know how to send email yet, so for the moment we'll just print a message for each friend.

Save & Run 4/30/2021, 8:34:15 PM - 2 of 2 Show in CodeLens Audio Tour

```
for name in ["Joe", "Amy", "Brad", "Angelina", "Zuki", "Thandi", "Paris"]:
    print("Hi", name, "Please come to my party on Saturday!")

```

```
Hi Joe Please come to my party on Saturday!
Hi Amy Please come to my party on Saturday!
Hi Brad Please come to my party on Saturday!
Hi Angelina Please come to my party on Saturday!
Hi Zuki Please come to my party on Saturday!
Hi Thandi Please come to my party on Saturday!
Hi Paris Please come to my party on Saturday!
```

Activity: 2 -- ActiveCode (ac6_2_1)

Take a look at the output produced when you press the `run` button. There is one line printed for each friend. Here's how it works:

- **name** in this `for` statement is called the **loop variable** or, alternatively, the **iterator variable**.
- The list of names in the square brackets is the sequence over which we will iterate.
- Line 2 is the **loop body**. The loop body is always indented. The indentation determines exactly what statements are "in the loop". The loop body is performed one time for each name in the list.
- On each *iteration* or *pass* of the loop, first a check is done to see if there are still more items to be processed. If there are none left (this is called the **terminating condition** of the loop), the loop has finished. Program execution continues at the next statement after the loop body.
- If there are items still to be processed, the loop variable is updated to refer to the next item in the list. This means, in this case, that the loop body is executed here 7 times, and each time `name` will refer to a different friend.
- At the end of each execution of the body of the loop, Python returns to the `for` statement, to see if there are more items to be handled.

The overall syntax is `for <loop_var_name> in <sequence>:`

- Between the words for and in, there must be a variable name for the loop variable. You can't put a whole expression there.
- A colon is required at the end of the line
- After the word in and before the colon is an expression that must evaluate to a sequence (e.g, a string or a list or a tuple). It could be a literal, or a variable name, or a more complex expression.

You have attempted 3 of 2 activities on this page

7.1. Introduction: Iteration">
oduction: Iteration">
7.3. Flow of Execution of the for Loop">

✔ Completed. Well Done!

| Back to top