

5.6: DEFINITE LOOPS USING FOR



Contributed by [Chuck Severance](#)
Clinical Associate Professor (School of Information) at [University of Michigan](#)

Sometimes we want to loop through a *set* of things such as a list of words, the lines in a file, or a list of numbers. When we have a list of things to loop through, we can construct a *definite* loop using a `for` statement. We call the `while` statement an *indefinite* loop because it simply loops until some condition becomes `False`, whereas the `for` loop is looping through a known set of items so it runs through as many iterations as there are items in the set.

The syntax of a `for` loop is similar to the `while` loop in that there is a `for` statement and a loop body:

CODE 5.6.1 (PYTHON):

```
%%python3

friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends:
    print('Happy New Year:', friend)
print('Done!')
```

run restart

In Python terms, the variable `friends` is a list¹ of three strings and the `for` loop goes through the list and executes the body once for each of the three strings in the list resulting in this output:

```
Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally
Done!
```

Translating this `for` loop to English is not as direct as the `while`, but if you think of `friends` as a *set*, it goes like this: "Run the statements in the body of the `for` loop once for each friend *in* the set named `friends`."

Looking at the `for` loop, `for` and `in` are reserved Python keywords, and `friend` and `friends` are variables.

```
for friend in friends:
    print('Happy New Year:', friend)
```

In particular, `friend` is the *iteration variable* for the `for` loop. The variable `friend` changes for each iteration of the loop and controls when the `for` loop completes. The *iteration variable* steps successively through the three strings stored in the `friends` variable.