5.8: COUNTING AND SUMMING LOOPS

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

For example, to count the number of items in a list, we would write the following `for` loop:

CODE 5.8.1 (PYTHON):

```
%%python3

count = 0
for itervar in [3, 41, 12, 9, 74, 15]:
    count = count + 1
print('Count: ', count)
```

run restart

We set the variable `count` to zero before the loop starts, then we write a `for` loop to run through the list of numbers. Our *iteration* variable is named `itervar` and while we do not use `itervar` in the loop, it does control the loop and cause the loop body to be executed once for each of the values in the list.

In the body of the loop, we add 1 to the current value of `count` for each of the values in the list. While the loop is executing, the value of `count` is the number of values we have seen "so far".

Once the loop completes, the value of `count` is the total number of items. The total number "falls in our lap" at the end of the loop. We construct the loop so that we have what we want when the loop finishes.

Another similar loop that computes the total of a set of numbers is as follows:

CODE 5.8.1 (PYTHON):

```
%%python3

total = 0
for itervar in [3, 41, 12, 9, 74, 15]:
    total = total + itervar
print('Total: ', total)
```

run restart

In this loop we *do* use the *iteration variable*. Instead of simply adding one to the `count` as in the previous loop, we add the actual number (3, 41, 12, etc.) to the running total during each loop iteration. If you think about the variable `total`, it contains the "running total of the values so far". So before the loop starts `total` is zero because we have not yet seen any values, during the loop `total` is the running total, and at the end of the loop `total` is the overall total of all the values in the list.

As the loop executes, `total` accumulates the sum of the elements; a variable used this way is sometimes called an *accumulator*.

Neither the counting loop nor the summing loop are particularly useful in practice because there are built-in functions `len()` and `sum()` that compute the number of items in a list and the total of the items in the list respectively.