Chapter 5 notes.

Before you can update a variable, you have to *initialize* it, usually with a simple

assignment:

>>> x = 0

>>> x = x + 1

Updating a variable by adding 1 is called an *increment*; subtracting 1 is called a

*decrement*.

More formally, here is the flow of execution for a while statement:

1. Evaluate the condition, yielding True or False.

2. If the condition is false, exit the while statement and continue execution at

the next statement.

3. If the condition is true, execute the body and then go back to step 1.

This type of flow is called a *loop* because the third step loops back around to the

top. We call each time we execute the body of the loop an *iteration*. For the above

loop, we would say, “It had five iterations”, which means that the body of the loop

was executed five times.

An endless source of amusement for programmers is the observation that the directions

on shampoo, “Lather, rinse, repeat,” are an infinite loop because there is

no *iteration variable* telling you how many times to execute the loop.

In the case of countdown, we can prove that the loop terminates because we know

that the value of n is finite, and we can see that the value of n gets smaller each

time through the loop, so eventually we have to get to 0. Other times a loop is

obviously infinite because it has no iteration variable at all.

Sometimes you don’t know it’s time to end a loop until you get half way through

the body. In that case you can write an infinite loop on purpose and then use the

break statement to jump out of the loop.

This loop is obviously an *infinite loop* because the logical expression on the while

statement is simply the logical constant True:

The loop condition is True, which is always true, so the loop runs repeatedly until

it hits the break statement.

Each time through, it prompts the user with an angle bracket. If the user types

done, the break statement exits the loop. Otherwise the program echoes whatever

the user types and goes back to the top of the loop.

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.

**accumulator** A variable used in a loop to add up or accumulate a result.

**counter** A variable used in a loop to count the number of times something happened.

We initialize a counter to zero and then increment the counter each

time we want to “count” something.

**decrement** An update that decreases the value of a variable.

**initialize** An assignment that gives an initial value to a variable that will be

updated.

**increment** An update that increases the value of a variable (often by one).

**infinite loop** A loop in which the terminating condition is never satisfied or for

which there is no terminating condition.

**iteration** Repeated execution of a set of statements using either a function that

calls itself or a loop.