* While Loop
* For Loop
* Nested Loops

Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.

Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

Loops may be nested in other loops.

A loop becomes an *infinite* loop if a condition never becomes FALSE. You must use caution when using *while* loops because of the possibility that this condition never resolves to a FALSE value. This results in a loop that never ends. Such a loop is called an *infinite* loop. (Python while loop, 2017).

This *for* loop sets up *i* as its iterating variable, and the sequence exists in the range 0 to 5. Within the loop, we print out one integer per loop iteration. Note that the program begins at index 0 and prints out five numbers. Thus, this is why the range of output numbers is only from 0-4.

This *for* loop sets up *i* as its iterating variable, and the sequence exists in the range 0 to 5. Within the loop, we print out one integer per loop iteration. Note that the program begins at index 0 and prints out five numbers. Thus, this is why the range of output numbers is only from 0-4.

Within the *for* loop, we print out one integer per loop iteration.

One of Python’s built-in immutable sequence types is *range()*. In loops, *range()* is used to control how many times a loop will be repeated. When working with *range()*, you can pass between 1 and 3 integer arguments to it.

* start: states the integer value at which the sequence begins; if this is not included then start begins at 0.
* stop: is always required and is the integer that is counted up to but not included
* step: sets how much to increase (or decrease in the case of negative numbers) the next iteration; if this is omitted then stepdefaults to 1.

Lists and other data sequence types can also be incorporated as iteration parameters in for loops. Consider a *for* loop that iterates through a list of fruits.

"The Julia set of *f*, denoted by *J(f)* is the set of numbers such that the tiniest change will radically change the value under iteration of the function.

Python *for* loops may also have an *else* block. If the *else* statement is used with a *for* loop, then that *else* statement is executed when the loop has exhausted iterating through the list.