**Introduction to Lists**

Lists are a *datatype* you can use to store a collection of different pieces of information as a sequence under a single variable name. (Datatypes you've already learned about include strings, numbers, and booleans.)

You can assign items to a list with an expression of the form

list\_name = [item\_1, item\_2]

with the items in between brackets. A list can also be empty: empty\_list = [].

Lists are very similar to strings, but there are a few key differences.

**Access by Index**

You can access an individual item on the list by its *index*. An index is like an address that identifies the item's place in the list. The index appears directly after the list name, in between brackets, like this: list\_name[index].

*List indices begin with 0, not 1!* You access the first item in a list like this: list\_name[0]. The second item in a list is at index 1: list\_name[1]. Computer scientists love to start counting from zero.

# New Neighbors

A list index behaves like any other variable name! It can be used to access as well as assign values.

You saw how to access a list index like this:

zoo\_animals[0] # Gets the value "pangolin"

You can see how assignment works on line 5:

zoo\_animals[2] = "hyena" # Changes "sloth" to "hyena"

**Late Arrivals & List Length**

A list doesn't have to have a fixed length. You can add items to the end of a list any time you like!

letters = ['a', 'b', 'c'] letters.append('d') print len(letters) print letters

1. In the above example, we first create a list called letters.
2. Then, we add the string 'd' to the end of the letters list.
3. Next, we print out 4, the length of the letters list.
4. Finally, we print out ['a', 'b', 'c', 'd'].

# List Slicing

Sometimes, you only want to access a portion of a list. Consider the following code:

letters = ['a', 'b', 'c', 'd', 'e'] slice = letters[1:3] print slice print letters

What is this code doing?

First, we create a list called letters.

Then, we take a subsection of the list and store it in the slice list. We do this by defining the indices we want to include after the name of the list: letters[1:3]. In Python, when we specify a portion of a list in this manner, we **include** the element with the first index, 1, but we **exclude** the element with the second index, 3.

Next, we print out slice, which will print ['b','c']. Remember, in Python indices always start at 0, so the 1 element is actually b.

Finally, we print out ['a', 'b', 'c', 'd', 'e'], notice that we did not modify the original letters list.

In order to slice the last two items from the list  
myList = [0,1,2,3,4], the ending index of your slice will be one beyond the actual last index of the list. Check it out:

myList[3:5] # Returns [3, 4]

# Slicing Lists and Strings

You can slice a string exactly like a list! In fact, you can think of strings as lists of characters: each character is a sequential item in the list, starting from index 0.

my\_list[:2] # Grabs the first two items my\_list[3:] # Grabs the fourth through last items

If your list slice includes the very first or last item in a list (or a string), the index for that item doesn't have to be included.

**Maintaining Order**

Sometimes you need to search for an item in a list.

animals = ["ant", "bat", "cat"] print animals.index("bat")

1. First, we create a list called animalswith three strings.
2. Then, we print the first index that contains the string "bat", which will print 1.

We can also insert items into a list.

animals.insert(1, "dog") print animals

1. We insert "dog" at index 1, which moves everything down by 1.
2. We print out ["ant", "dog", "bat", "cat"]

# For One and All

If you want to do something with every item in the list, you can use a for loop. If you've learned about for loops in JavaScript, pay close attention! They're different in Python.

for variable in list\_name: # Do stuff!

A variable name follows the for keyword; it will be assigned the value of each list item in turn.

Then in list\_name designates list\_name as the list the loop will work on. The line ends with a colon (:) and the indented code that follows it will be executed once per item in the list.

**More with 'for'**

If your list is a jumbled mess, you may need to sort() it.

animals = ["cat", "ant", "bat"] animals.sort() for animal in animals: print animal

1. First, we create a list called animalswith three strings. The strings are not in alphabetical order.
2. Then, we sort animals into alphabetical order. Note that .sort() modifies the list rather than returning a new list.
3. Then, for each item in animals, we print that item out as "ant", "bat", "cat" on their own line each.

**Remove a Few Things**

Sometimes you need to remove something from a list.

beatles = ["john","paul","george","ringo","stuart"] beatles.remove("stuart") print beatles

This code will print:

["john","paul","george","ringo"]

1. We create a list called beatles with 5 strings.
2. Then, we remove the first item from beatles that matches the string "stuart". Note that .remove(item) does not return anything.
3. Finally, we print out that list just to see that "stuart" was actually removed.