Lesson 10 – Lists

1. **Discussion**

A **list** is a sequence of several variables, grouped together under a single name. Instead of writing a program with many variables x0, x1, x2, etc., you can define a single variable mylist and access its members by mylist[0], mylist[1], mylist [2], etc.

One way to create a list is by enclosing several values, separated with commas, between square brackets:

myList = ["one", 1999, 4.5, True]

This creates a list called myList of length 4. Each element of the list gets a number called its **index**. The first element has index 0 and we count up from 0.

You can assign a value to a list item as follows:

myList[1] = 90

but only if myList[1] already exists. We will discuss how to add elements to an empty list a bit later on.

A common error is to ask for an index that doesn't exist. In the above example, because myList has length 4 and the first index is 0, the maximum possible index is 3. Asking for an index of 4 or larger will give an error. Note that in Python, negative indices have a different meaning than most other languages.

Lists are very similar to strings - both of them can be passed to the len() function to get their lengths, and both of them use X[i] to extract individual items. Lists and strings are called "[sequence types](http://docs.python.org/py3k/library/stdtypes.html?highlight=sequence#sequence-types-str-bytes-bytearray-list-tuple-range)" in Python. The one major difference between the two is that strings are immutable, while lists can be modified.

Also, last lesson we introduced the idea of a “tuple”. Tuples are essentially immutable lists. The lesson on def showed how tuples can be used – if you think about it, it also shows why tuples should be immutable.

1. **Common Operations**

The length of a list is given by len(listName). We have discussed this already above as well as with strings.

Top of Form

Bottom of Form

You need to create an account and log in to ask a question.

You can merge two lists together as follows:

A = [1, 2,3]

B = [4, 5, 6, 7, 8]

C = A + B

If you print C out, you will get 1, 2, 3, 4, 5, 6, 7, 8



Similarly, you to extend a list as follows:

A = [0]

A = A\*10

If you print out A now, you will get 10 0’s. This is useful for creating a list of a desired length. Note that you need to start with an existing list.

You can add items to an empty list in two ways as shown below:

A = []

for i in range(0, 10):

A.append(i)

A = []

for i in range(0, 10):

A = A + [i]



You need to create an account and log in to ask a question.

You can use negative indices to obtain items from the end of a list by using A[-k] which returns the kth item from the end of the list. Note that this shortcut notation works for strings as well.

The function max(L) returns the largest number in the list. Likewise, the function sum(L) returns the sum of the elements in list L.

1. **Looping through lists**

Top of Form

Bottom of Form

It is very common to loop through a list to access every value. Python allows a shortcut to perform this type of an operation, usually called a "for all" loop ("for each" loopin Java)*.* An example is given below, assuming a list of numbers.

for x in L:

sum += x

We will discuss how this works in class.

Top of Form

Bottom of Form

Note: “For all" loops work for strings too: try [for char in "hello"](http://cscircles.cemc.uwaterloo.ca/console/?consolecode=for%20char%20in%20%22hello%22%3A%20print%28char%29" \t "_blank).

1. **Exercises**

* Sorting - Given a list, write a program that determines if it is sorted or not.
* Natural Numbers - Write a function naturalNumbers which takes a positive integer n as input, and returns a list [1, 2, ...] consisting of the first n natural numbers
* Peaks - A peak in a list of integers is where one number is greater than the previous number as well as the successive number. Write a program that finds all peaks and prints them out along with the index that it occurs at.
* Sieve of Eratosthenes - To find all the prime numbers less than or equal to a given integer n by Eratosthenes' method:

1. Create a list of size n;
2. Initially, let p equal 2, the first prime number (you can ignore 0 and 1)
3. Mark all multiples of p greater than p. I would suggest a boolean but there are other ways
4. Find the first number remaining on the list greater than p (this number is the next prime); let p equal this number.
5. Repeat steps 3 and 4 until p is greater than n.
6. All the remaining numbers on the list are prime.

* Treasure Hunt - Given a list of integers and a starting location, use the value at the starting location to move to the next location and continue until one of two things happen. Either you repeat a location, in which case you lose, or you go past the end of the array, in which case you have found the treasure.

For example, if we have the list [2, 4, 1, 7, 3, 2, 8] and a starting position of 0 would give the sequence of 0, 2, 1, 4, 3, 7 which leads to treasure!

However, if we have the list [2, 4, 1, 1, 3, 0, 8] and a starting position of 3, we get the sequence of 3, 1, 4, 3 which means you lose!

* Juniper Green - The game is played by two players who alternate moves. The first move must be an even number. Each subsequent move must either be an integer multiple or an integer divisor of the previous move and you may NOT take a number twice. The last player who is able to make a move is the winner. Use the numbers from 1 to 116 and you may omit the primes over 50.
* Big number manipulation – While python can handle arbitrarily large numbers, most languages cannot. A convenient way to do so is to place all the digits of the number in a list and then add and subtract the digits in a list in a similar manner to conventional addition and subtraction. I would like to discuss this process to make sure that you understand what to do so ask before you get started.