Python: Lists, Aliasing, and Mutability

Adapted from MIT OpenCourseWare and J.Guttag’s Introduction to Computation and Programming

Object Types

Previously we talked about variable data types: int, float, bool, string.

Now we’re going to look at a compound data type, a **list**

Lists

A list is an **ordered sequence** of elements, accessible by index, denoted by square brackets, []. Lists are usually homogeneous (i.e. contain elements of the same type) but less commonly, can contain mixed types.

a\_list= []

L = [2, 'a', 4, [1,2]]

len(L) <------------------------- 4

L[0] <---------------------------

L[2]+1 <-------------------------

L[3] <---------------------------

L[4] <---------------------------

i = 2

L[i-1] <-------------------------

List elements can be changed, so a list is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. Assigning an element at an index changes the value:

L = [2, 1, 3]

L[1] = 5 <----------------------

Manipulating Lists

You can iterate over lists. For example, compute the sum of the elements of a list:

|  |  |
| --- | --- |
| total = 0  for i in range(len(L)):  total += L[i]  print total | total = 0  for i in L:  total += i  print total |

Notice that list elements are indexed 0 to len(L)-1, and range(n) goes from 0 to n-1

Adding Elements

**Add** elements to end of list with L.append(element). This **mutates** the list.

L = [2,1,3]

L.append(5) <-------------------

To combine lists, use **concatenation** (+). This will give a new list.

L1 = [2,1,3]

L2 = [4,5,6]

L3 = L1 + L2 <------------------

Or mutate the list with L.extend(some\_list)

L1.extend([0,6]) <--------------

Removing Elements

**Delete** an element at a specific index with del(L[index]).

**Remove** the element at the end of a list with L.pop(). This **returns** the removed element

**Remove** a specific element with L.remove(element). This looks for the element, and removes the first occurrence. If the element is not in the list, it gives an error.

L = [2,1,3,6,3,7,0] # Do the following operations in order

L.remove(2) <------------------

L.remove(3) <------------------

del(L[1]) <--------------------

L.pop() <----------------------

Convert Lists to Strings and Strings to Lists

Convert a string to a list with list(s). This returns a list where each character from s is an element in L

You can use s.split() to split a string on a character parameter. It will split on spaces if called without a parameter.

Use ‘’join(L) to turn a list of characters into a string. If you enter a character in the quotes it will add that character between every element.

s = “I<3 cs” <-----------------

List(s) <----------------------

s.split(‘<’) <-----------------

L = [‘a’,’b’,’c’] <------------

‘’.join(L) <-------------------

‘\_’.join(L) <------------------

Sorting Lists

Calling sort()mutates the list and returns nothing.

Calling sorted()does not mutate the list and must assign the result to a variable

Calling reverse()mutates the list and returns nothing.

L = [9,6,0,3] <----------------

sorted(L) <--------------------

L.sort() <---------------------

L.reverse() <------------------

Lists of Lists

You can have nested lists. How would this look for representing a matrix?

Careful…

Avoid mutating lists as you iterate over them.

define remove\_dups(L1, L2):

for e in L1:

if e in L2:

L1.remove(e)

L1 = [1, 2, 3, 4]

L2 = [1, 2, 5, 6]

remove\_dups(L1, L2)

Now L1 is [2, 3, 4] not [3, 4]. Why? How could you fix this problem?