Assignment 1

Create a simple class, **MaxSizeList**, that acts a little bit like a list, with a pre-configured limit on its size.

Here some test calling code that imports the completed **MaxSizeList** and then uses it to create two new **MaxSizeList** objects.

[calling code]

from assignments import MaxSizeList # assumes "class MaxSizeList"

# is in a script called "assignments.py"

a = MaxSizeList(3)

b = MaxSizeList(1)

a.push("hey")

a.push("hi")

a.push("let's")

a.push("go")

b.push("hey")

b.push("hi")

b.push("let's")

b.push("go")

print(a.get\_list())

print(b.get\_list())

# ['hi', "let's", 'go']

# ['go']

Note: in the above example, we're premising that the class **MaxSizeList** is in a module called **assignments.py**. You can make your solution simpler by simply putting the class code in the same script as the calling code above:

class MaxSizeList(object):

# your code for MaxSizeList

a = MaxSizeList(3)

b = MaxSizeList(1)

# etc., same calling code as above

The calling code demonstrates the **MaxSizeList** class by:

1. creating two **MaxSizeList** instances, and passing a different integer to each constructor

2. calling the **push()** method to add four strings into each instance

3. calling a **get\_list()** method which is intended to return a list of the elements that the instance contains

Notice the correspondence between the number of elements returned by **get\_list()** and the integer value that we originally passed in the constructor: in the first instance, we passed the integer **3** to the constructor to produce the instance labeled **a**, and **get\_list()** called on **a** returns three elements; in the second instance, we passed the integer **1** to the constructor to produce the instance labeled **b**, and **get\_list()** called on **b** returns one element. In other words, the integer argument passed to the constructor will determine the maximum size of the list being held by the **MaxSizeList** object.

So, what attributes will each instance have to have to support this behavior?

Well, certainly it needs to have an attribute that is a list. Now so far, we've seen attributes that only hold integers or strings. But an attribute can hold any object, in the same way that a variable label can be associated with any object. So, it could certainly hold a list:

class MyClass(object):

def set\_attrs(self):

self.attr1 = 5

self.attr2 = [1, 2, 3, 4, 5]

xx = MyClass()

xx.set\_attrs()

print xx.attr2 # [1, 2, 3, 4, 5]

What other attributes would the object have to hold? Well, it would need to know what its maximum size is. Because as you can see from the calling code demo, if we try to add an element that would make the size of the list greater than its maximum size, it drops an element from that list.

In case there are any gaps in your basic knowledge that would prevent you from completing this assignment, here are a few usage examples:

# adding an element to a list

mylist = []

mylist.append('hello')

mylist.append('world')

print mylist # ['hello', 'world']

# removing the first element from a list

mylist.pop(0)

print mylist # ['world']

Beyond these hints, you will need the lessons we learned about the constructor for initializing an instance, and setter and getter methods for setting attributes in the instance and getting attributes from the instance (but in this case **push()** will just add an element to the list held in the object, not just set it, and **get\_list()** will show us all the elements in the list).

Good luck!