Lists

Revisiting Lists in Python

We discussed lists a bit in our discussion on iteration. Let's look more in-depth at what you can do with lists.

Putting elements in lists

In the preceding notes, we put elements in lists by creating a list with a single statement:

\[ \text{list} = ['a','b','c'] \]

You can also modify a list using an index:

\[ \text{list}[2] = 'x' \]

Python also provides some functions you can call to insert things in lists:

append -- appends a new element to the end of a list.

\[ \text{list} = [1,2,3] \]

\[ \text{list}.append(4) \]

\[ \text{print list} \]
\[ [1,2,3,4] \]

insert -- inserts an element somewhere in the list.

\[ \text{list}.insert(2,99) \ # should put 99 in the 2nd slot. \]

\[ \text{print list} \]
\[ [1,2,99,3,4] \]

Initializing Lists

You can initialize lists by assigning a list to the empty list [], e.g.,

\[ \text{list} = [] \]
Note that you need to initialize a list before calling 'append'—You can't append to a list that hasn't been initialized either with an empty or non-empty list (e.g., list=[1,2,3])

You also cannot access/set an element of a list unless the list already has that element. So the following code will give an error:

```python
list=[]
list[0]=4  # error
```

```python
list2=[1,2]
list2[3]=7  # error
```

**Copying and Aliasing**

The following statement does NOT make a copy of a list:

```python
list1=list2
```

It creates an alias—both variables will point at the same contents. Consider the following code:

```python
list1=[1,2,3]
list2=list1  # ***
list2[1]=55
```

Both list1 will print as [1,55,3]

You can create a copy of a list using a statement like the following:

```python
list2=list1[:]
```

If this line replaces *** in the code above, list1 will not be modified when list2 is.

The ':' syntax is an example of 'get range' operation of a list.

```python
list2=list1[0:1]  # returns the first two elements of list1.
```
Object-Oriented Function Calls

Consider the syntax used to call the list function 'append':

    nameOfList.append(item)

We call this an object-oriented function call, where the list in this case is the object.

This is different than the functions we've called so far, e.g., len(list)

In-Class Problems

1. Which of the following will cause errors? Enter them in the interpreter to check.
   a. list1[0]='abc'
   b. list2=[1,2,3]
      list2[3]=4
   c. list2=[1,2,3]
      list2.insert(3,4)
   d. list2=[1,2,3]
      list2.append(4)

2. a. Write a program that initializes an empty list, then uses a while loop to add the first n numbers to the list (use append)
   b. Write the same program, but initialize the list as a list of 0s, then use a while loop to ‘modify’ each element with the first n numbers (do not use append)