Lists

Lists are ordered sequences that can hold a variety of object types

They use [] brackets and commas to separate objects in the list [1,2,3,4,5]

Lists support indexing and slicing. Lists can be nested and also have a variety of useful methods that can be called off of them.

In [9]:
```
my_list = [1,2,3]
```

In [16]:
```
my_list = ['STRING',100,23.2]
```

In [17]:
```
print(my_list)
```

```
['STRING', 100, 23.2]
```

In [20]:
```
mylist = ['one', 'two', 'three']
```

In [21]:
```
mylist[0]
```

Out[21]:
```
'one'
```

In [22]:
```
mylist[1:]
```

Out[22]:
```
['two', 'three']
```

In [24]:
```
another_list = ['four', 'five']
```

In [25]:
```
mylist + another_list
```

Out[25]:
```
['one', 'two', 'three', 'four', 'five']
```

In [26]:
```
mylist
```

Out[26]:
```
['one', 'two', 'three']
```

In [27]:
```
another_list
```

Out[27]:
```
['four', 'five']
```

We have added the list, however we did not set the list together, we can do so by running the following:

In [28]:
```
new_list = mylist + another_list
```

In [29]:
```
new_list
```

Out[29]:
```
['one', 'two', 'three', 'four', 'five']
```

Whats different from a list and string is we can actually modify or mutate the list

In [30]:
```
new_list
```

Out[30]:
```
['one', 'two', 'three', 'four', 'five']
```

So now, lets change one of the elements in the list

In [31]:
```
new_list[0] = 'ONE ALL CAPS'
```

In [32]:
```
new_list
```

Out[32]:
```
['ONE ALL CAPS', 'two', 'three', 'four', 'five']
```

We can also add an element to the very end of a list like so:

In [38]:
```
new_list.append('six')
```

In [39]:
```
new_list
```

Out[39]:
```
['ONE ALL CAPS', 'two', 'three', 'four', 'five', 'six']
```

In [41]:
```
new_list.append('seven')
```

In [42]:
```
new_list
```

Out[42]:
```
['ONE ALL CAPS', 'two', 'three', 'four', 'five', 'six', 'seven']
```

We can also remove elements from a list using the pop method like so:

In [44]:
```
new_list.pop()
```

Out[44]:
```
'seven'
```

In [45]:
```
new_list
```

Out[45]:
```
['ONE ALL CAPS', 'two', 'three', 'four', 'five', 'six']
```

We can save the popped item like so:

In [46]:
```
popped_item = new_list.pop()
```

In [47]:
```
popped_item
```

Out[47]:
```
'six'
```

We can also pop (or remove) an element from a specific index like so:

In [48]:
```
new_list.pop(0)
```

Out[48]:
```
'ONE ALL CAPS'
```

In [49]:
```
new_list
```

Out[49]:
```
['two', 'three', 'four', 'five']
```

We can also do reverse indexing for pop removal

In [50]:
```
new_list.pop(-1)
```

Out[50]:
```
'five'
```

In [51]:
```
new_list
```

Out[51]:
```
['two', 'three', 'four']
```

Sorts and Reverse

In [52]:
```
new_list = ['a' , 'e', 'x', 'b', 'c' ]
num_list = [4,1,8,3]
```

Let us sort these list

In [53]:
```
new_list.sort()
```

In [54]:
```
new_list
```

Out[54]:
```
['a', 'b', 'c', 'e', 'x']
```

We now see the new_list is now in alphabetical order

We can assign the new sorted listed to a new object like so:

In [55]:
```
new_list.sort()
my_sorted_list = my_list
```

In [57]:
```
my_sorted_list = new_list
```

In [58]:
```
new_list
```

Out[58]:
```
['a', 'b', 'c', 'e', 'x']
```

In [59]:
```
num_list.sort()
```

In [60]:
```
num_list
```

Out[60]:
```
[1, 3, 4, 8]
```

Now let us discuss the reverse method

In [61]:
```
num_list.reverse()
```

In [62]:
```
num_list
```

Out[62]:
```
[8, 4, 3, 1]
```

How do I index a nested list? For example if I wanted to grab 2 from [1,1,[1,2]]?

You would just add another set of brackets for indexing the nested list, for example: my_list[2][1] We will discover later on more nested objects.

If lst=[0,1,2] what is the result of lst.pop()

A. 0 B. 1 C. 2

Answer: C. 2

Lists can have multiple object types.

A. True B. False

Answer: A. True

If lst=['a','b','c'] What is the result of lst[1:]?

A. 'b' B. ['b', 'c'] C. 'c' D. ['a','b']

Answer: B. ['b', 'c']

In [ ]:
```

```