Python Lists and List Manipulation

Michael Galarnyk May 29, 2017

Python Lists and List Manipulation Video

Before starting, I should mention that the code in this blog post and in the video above is available on my github.

Defining a List

Lists are written within square brackets []

z =	[3,	7,	4,	2]
index	0	1	2	3

Defining a List. The second row in this table index is how you access items in the list.

```
# Define a list
z = [3, 7, 4, 2]
```

Lists store an ordered collection of items which can be of different types. The list defined above has items that are all of the same type (int), but all the items of a list do not need to be of the same type as you can see below.

```
# Define a list
heterogenousElements = [3, True, 'Michael', 2.0]
```

The list contains an int, a bool, a string, and a float.

Access Values in a List

Each item in a list has an assigned index value. It is important to note that python is a zero indexed based language. All this means is that the first item in the list is at index 0.

z =	[3,	7,	4,	2]
index	0	1	2	3

Access item at index 0 (in blue)

```
# Define a list
z = [3, 7, 4, 2]
```

```
# Access the first item of a list at index 0
print(z[0])
```

```
print(z[0])
```

```
3
```

Output of accessing the item at index 0.

Python also supports negative indexing. Negative indexing starts from the end. It can be more convienient at times to use negative indexing to get the last item in the list because you don't have to know the length of the list to access the last item.

z =	[3,	7,	4,	2]
index	0	1	2	3
negative index	-4	-3	-2	-1

Accessing the item at the last index.

```
# print last item in the list
print(z[-1])
```

```
print(z[-1])
```

```
2
```

Output of accessing the last item in the List

As a reminder, you could also access the same item using positive indexes (as seen below).

```
print(z[3])
```

```
2
```

Alternative way of accessing the last item in the list z

Slice of Lists

Slices are good for getting a subset of values in your list. For the example code below, it will return a list with the items from index 0 up to and not including index 2.

z =	[3,	7,	4,	2]
index	0	1	2	3

First index is inclusive (before the :) and last (after the :) is not

```
# Define a list
z = [3, 7, 4, 2]
```

```
print(z[0:2])
```

```
print(z[0:2])
```

```
[3, 7]
```

Slice of a list syntax.

z =	[3,	7,	4,	2]
index	0	1	2	3

```
# everything up to but not including index 3
print(z[:3])
```

```
print(z[:3])
```

```
[3, 7, 4]
```

z =	[3,	7,	4,	2]
index	0	1	2	3

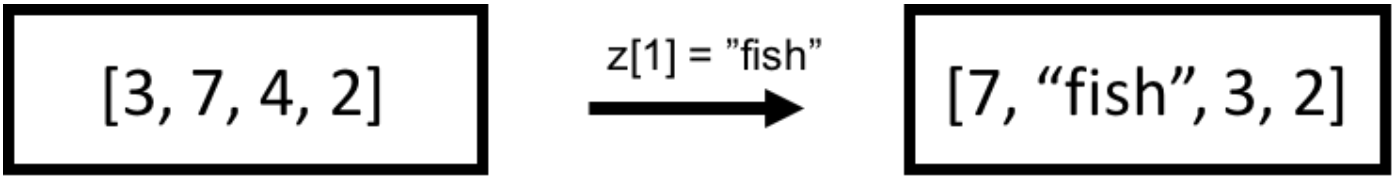
The code below returns a list with items from index 1 to the end of the list

```
# index 1 to end of list
print(z[1:])
```

```
print(z[1:])
```

```
[7, 4, 2]
```

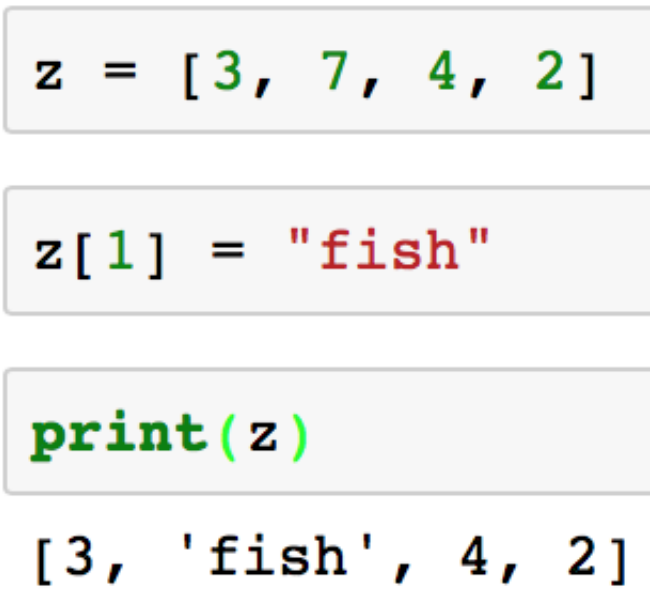
Update Item in a List



Lists in Python are mutable. All that means is that after defining a list, it is possible to update the individual items in a list.

```
# Defining a list
z = [3, 7, 4, 2]
```

```
# Update the item at index 1 with the string "fish"
z[1] = "fish"
print(z)
```



Code to modify an item in a list

List Methods

Python lists have different methods that help you modify a list. This section of the tutorial just goes over various python list methods.

Index Method

z =	[4,	1,	5,	4,	10,	4]
index	0	1	2	3	4	5

```
# Define a list
```

```
z = [4, 1, 5, 4, 10, 4]
```

z =	[4,	1,	5,	4,	10,	4]
index	0	1	2	3	4	5

The index method returns the first index at which a value occurs. In the code below, it will return 0.

```
print(z.index(4))
```

```
print(z.index(4))
```

```
0
```

z =	[4,	1,	5,	4,	10,	4]
index	0	1	2	3	4	5

You can also specify where you start your search.

```
print(z.index(4, 3))
```

```
print(z.index(4, 3))
```

```
3
```

Count Method

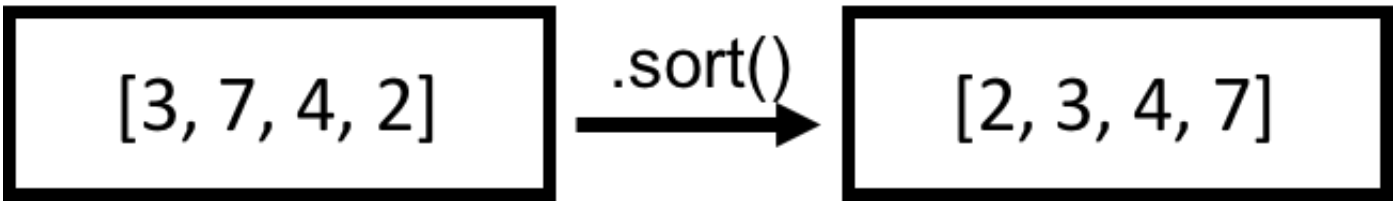
The count method works just like how it sounds. It counts the number of times a value occurs in a list

```
random_list = [4, 1, 5, 4, 10, 4]
random_list.count(5)
```

```
print(random_list.count(5))
```

```
1
```

Sort Method



Sort a Python List - the actual code would be: z.sort()

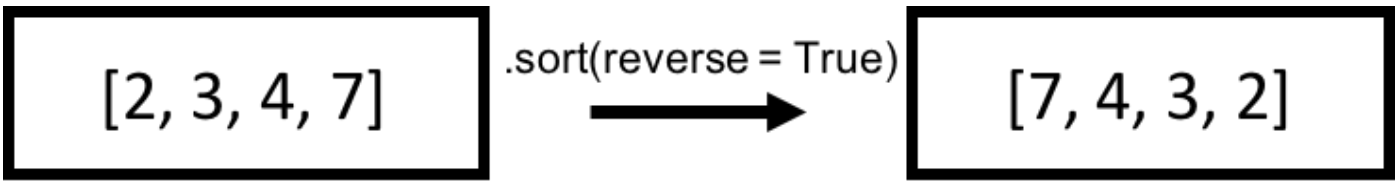
The sort method sorts and alters the original list in place.

```
z = [3, 7, 4, 2]
z.sort()
print(z)
```

```
z.sort()
print(z)
```

```
[2, 3, 4, 7]
```

The code above sorts a list from low to high. The code below shows that you can also sort a list from high to low.



Sort a python list from high to low

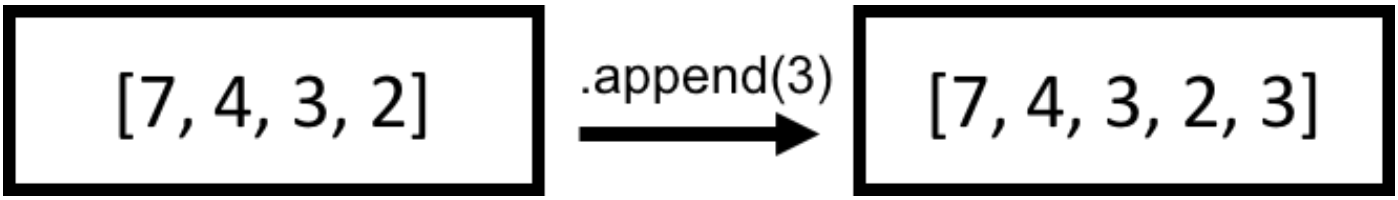
```
# Sorting and Altering original list
# high to low
z.sort(reverse = True)
print(z)
```

```
z.sort(reverse = True)
print(z)
```

```
[7, 4, 3, 2]
```

As an aside, I should mention that you can also sort a list of strings from a-z and z-a as you can see here.

Append Method



Add the value 3 to the end of the list.

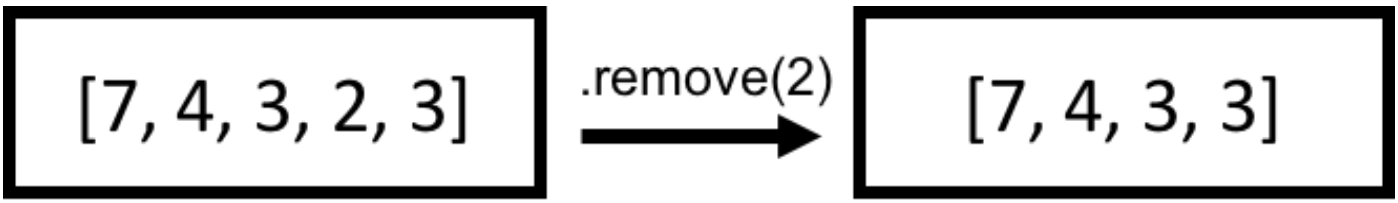
The append method adds an element to the end of a list. This happens inplace.

```
z = [7, 4, 3, 2]
z.append(3)
print(z)
```

```
z.append(3)
print(z)
```

```
[7, 4, 3, 2, 3]
```

Remove Method



The remove method removes the first occurrence of a value in a list.

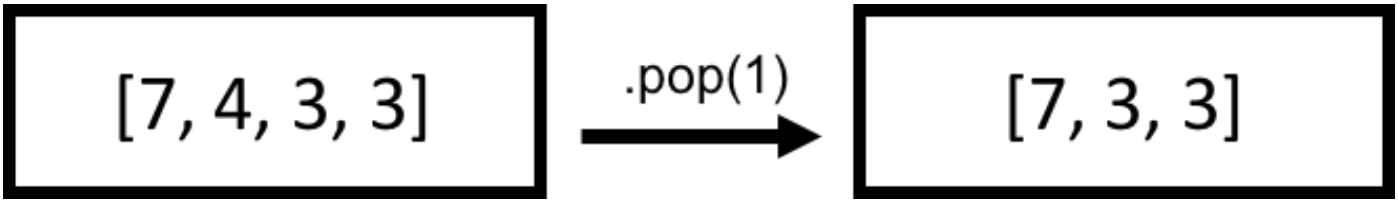
```
z = [7, 4, 3, 2, 3]
z.remove(2)
print(z)
```

```
z.remove(2)
print(z)
```

```
[7, 4, 3, 3]
```

Code removes the first occurrence of the value 2 from the list z

Pop Method



z.pop(1) removes the value at index 1 and returns the value 4.

The pop method removes an item at the index you provide. This method will also return the item you removed from the list. If you don't provide an index, it will by default remove the item at the last index.

```
z = [7, 4, 3, 3]
print(z.pop(1))
print(z)
```

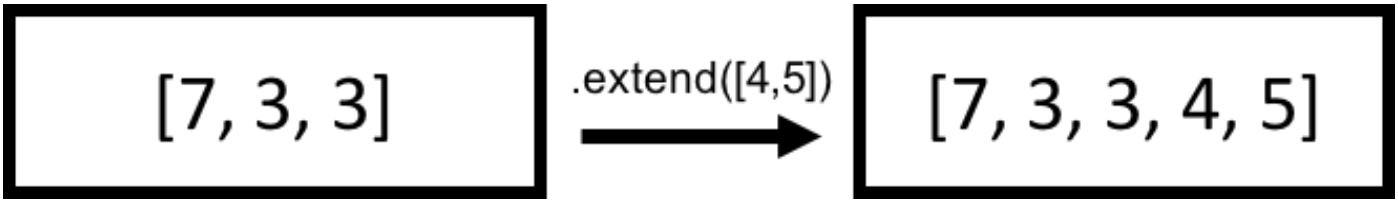
```
print(z.pop(1))
```

```
4
```

```
print(z)
```

```
[7, 3, 3]
```

Extend Method



The method extends a list by appending items. The benefit of this is you can add lists together.

```
z = [7, 3, 3]
z.extend([4,5])
print(z)
```

```
z.extend([4,5])
print(z)
```

```
[7, 3, 3, 4, 5]
```

Add the list [4, 5] to the end of the list z.

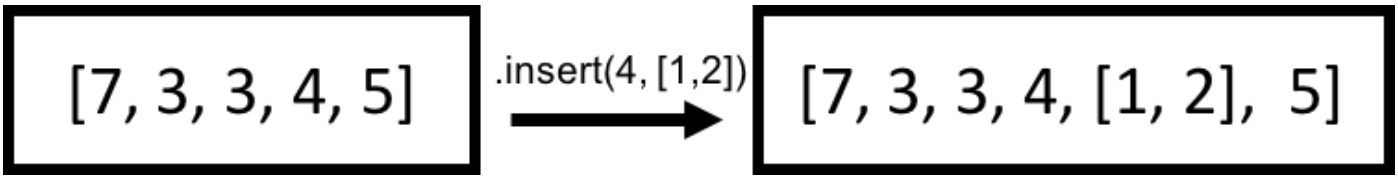
Alternatively, the same thing could be accomplished by using the + operator.

```
print([1,2] + [3,4])
```

```
print([1,2] + [3,4])
```

```
[1, 2, 3, 4]
```

Insert Method



insert the list [1,2] at index 4

The insert method inserts an item before the index you provide

```
z = [7, 3, 3, 4, 5]
z.insert(4, [1, 2])
print(z)
```

```
z.insert(4, [1, 2])
print(z)
```

```
[7, 3, 3, 4, [1, 2], 5]
```

Closing Remarks

Please let me know if you have any questions either here or in the comments section of the youtube video! The code in the post is also available on my github. Next post reviews for loops.