9.7. Mutating Methods

You've seen some methods already, like the `count` and `index` methods. Methods are either mutating or non-mutating. Mutating methods are ones that change the object after the method has been used. Non-mutating methods do not change the object after the method has been used.

The `count` and `index` methods are both non-mutating. Count returns the number of occurances of the argument given but does not change the original string or list. Similarly, index returns the leftmost occurance of the argument but does not change the original string or list. Below we'll talk about list methods in general. Keep an eye out for methods that are mutating!

9.7.1. List Methods

The dot operator can also be used to access built-in methods of list objects. `append` is a list method which adds the argument passed to it to the end of the list. Continuing with this example, we show several other list methods. Many of them are easy to understand.

Save & Run 5/13/2021, 2:17:27 PM - 2 of 2 Show in CodeLens

```
mylist = []
mylist.append(5)
mylist.append(27)
mylist.append(3)
mylist.append(12)
print(mylist)

mylist.insert(1, 12)
print(mylist)
print(mylist.count(12))

print(mylist.index(3))
print(mylist.count(5))

mylist.reverse()
```

```
[5, 27, 3, 12]
[5, 12, 27, 3, 12]
2
3
1
[12, 3, 27, 12, 5]
[3, 5, 12, 12, 27]
[3, 12, 12, 27]
27
[3, 12, 12]
```

Activity: 1 -- ActiveCode (ac8_6_1)

There are two ways to use the `pop` method. The first, with no parameter, will remove and return the last item of the list. If you provide a parameter for the position, `pop` will remove and return the item at that position. Either way the list is changed.

The following table provides a summary of the list methods shown above. The column labeled `result` gives an explanation as to what the return value is as it relates to the new value of the list. The word **mutator** means that the list is changed by the method but nothing is returned (actually `None` is returned). A **hybrid** method is one that not only changes the list but also returns a value as its result. Finally, if the result is simply a return, then the list is unchanged by the method.

Be sure to experiment with these methods to gain a better understanding of what they do.

Method	Parameters	Result	Description
append	item	mutator	Adds a new item to the end of a list
insert	position, item	mutator	Inserts a new item at the position given
pop	none	hybrid	Removes and returns the last item
pop	position	hybrid	Removes and returns the item at position
sort	none	mutator	Modifies a list to be sorted
reverse	none	mutator	Modifies a list to be in reverse order
index	item	return idx	Returns the position of first occurrence of item
count	item	return ct	Returns the number of occurrences of item
remove	item	mutator	Removes the first occurrence of item

Details for these and others can be found in the Python Documentation.

It is important to remember that methods like `append`, `sort`, and `reverse` all return `None`. They change the list; they don't produce a new list. So, while we did reassignment to increment a number, as in `x = x + 1`, doing the analogous thing with these operations will lose the entire list contents (see line 8 below).

Save & Run 5/13/2021, 2:17:32 PM - 2 of 2 Show in CodeLens

```
1 mylist = []
2 mylist.append(5)
3 mylist.append(27)
4 mylist.append(3)
5 mylist.append(12)
```

```
print(mylist)

mylist = mylist.sort()    #probably an error
print(mylist)

```

```
[5, 27, 3, 12]
None
```

Activity: 2 -- ActiveCode (ac8_6_2)

Check your understanding

seqmut-6-1: What is printed by the following statements?

```
alist = [4,2,8,6,5]
alist.append(True)
alist.append(False)
print(alist)
```

○ A. [4,2,8,6,5,False,True]

● B. [4,2,8,6,5,True,False]

○ C. [True,False,4,2,8,6,5]

Check me Compare me

✓ Yes, each item is added to the end of the list.

Activity: 3 -- Multiple Choice (question8_6_1)

You have attempted 4 of 3 activities on this page

✓ Completed. Well Done!

9.6. Cloning Lists">

ning Lists">

9.8. Append versus Concatenate">

| Back to top