Rex Jones II

Slice, Add, Remove & Sort Python List

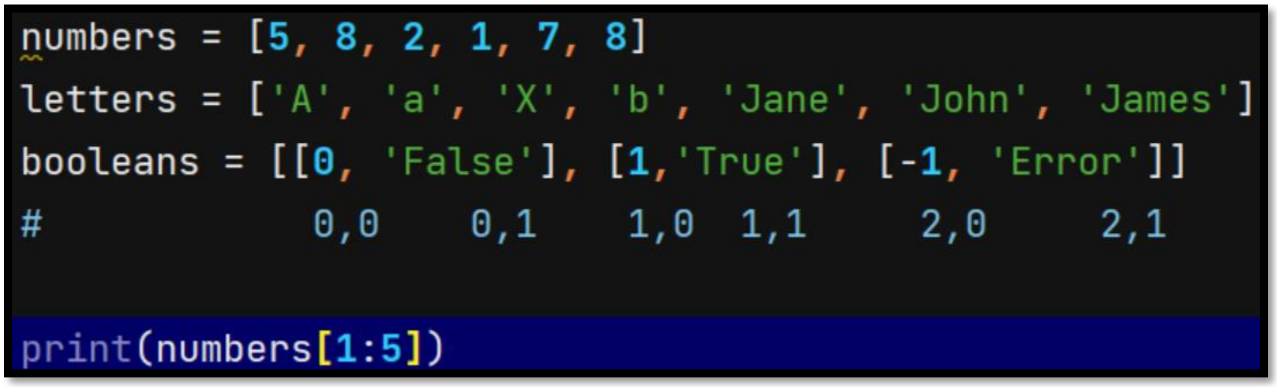
Python Video = https://youtu.be/Yh1weIzKhwk

Slice, Add, Remove, and Sort Python List

In this session, I will continue from the last session which introduced a Python list. The list value is a mutable data type. That means a value can be added, removed, or changed. Recall from the Strings video, a string value is immutable meaning it cannot be changed. The focus of this video will be slicing a list and modifying a list. If you are interested in more content, feel free to read the transcript on GitHub or download the source code. Also, like this video and subscribe to my channel. Plus follow me on Twitter, connect with me on LinkedIn and Facebook.

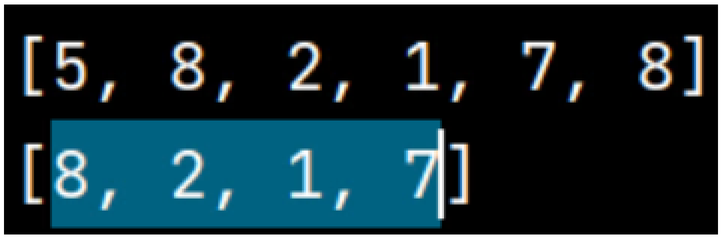
Slicing The List

In the IDE, let's start by slicing a list which is similar to using the range function. We slice by defining 2 integers that will return more than 1 value and form a new list. Therefore, if I will type [1:5].



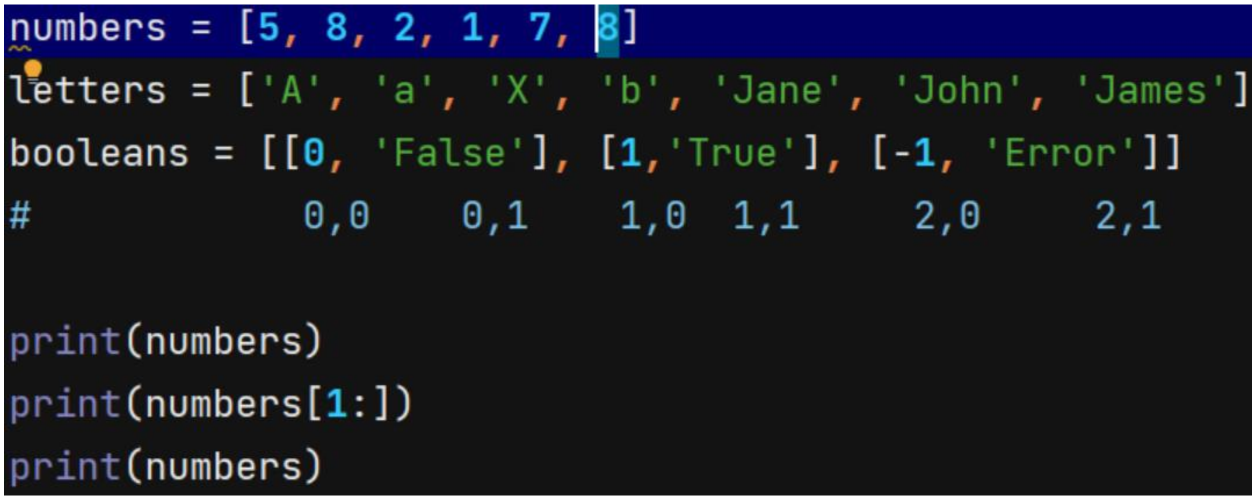
```
numbers = [5, 8, 2, 1, 7, 8]
letters = ['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
booleans = [[0, 'False'], [1,'True'], [-1, 'Error']]
#            0,0    0,1    1,0  1,1     2,0     2,1

print(numbers[1:5])
```

Index 1 will grab number 8 but index 5 will stop and not print the last item number 8 . I'm going to add a print statement before and after the slice. print(numbers) and do the same below it When I run, we only see numbers 8, 2, 1, 7.



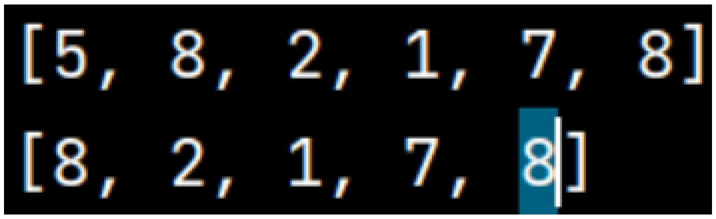
```
[5, 8, 2, 1, 7, 8]
[8, 2, 1, 7]
```

Python prints a slice of the numbers list. The 2nd integer always go up to an index position and stop but will not include that item. As a shortcut, we can remove one of these indexes or both of these indexes. Without the 2nd integer (index 5), the slice evaluates to the end of the list.



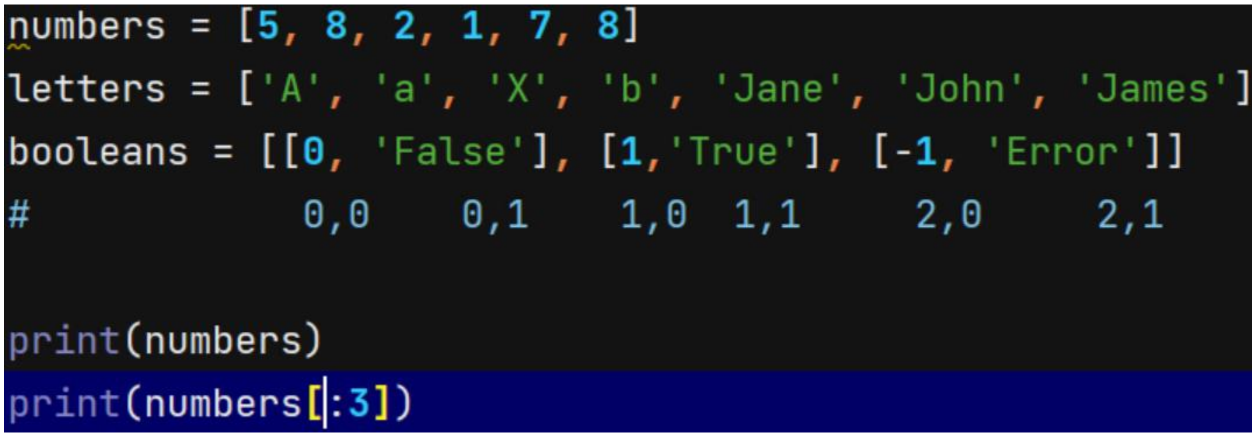
```
numbers = [5, 8, 2, 1, 7, 8]
letters = ['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
booleans = [[0, 'False'], [1,'True'], [-1, 'Error']]
#            0,0    0,1    1,0  1,1     2,0     2,1

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

Run and this time we see the last item number 8.



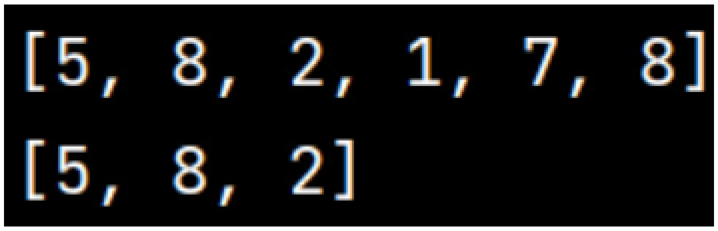
```
[5, 8, 2, 1, 7, 8]
[8, 2, 1, 7, 8]
```

All of the numbers are 8, 2, 1, 7, 8. The starting index is similar. Without a starting index then the slice starts at the beginning of the list. If I add 3 for the 2nd integer which is number 1 and remove the 1st integer then Python will automatically start our slice at index 0.



```
numbers = [5, 8, 2, 1, 7, 8]
letters = ['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
booleans = [[0, 'False'], [1,'True'], [-1, 'Error']]
#             0,0    0,1     1,0  1,1     2,0     2,1

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

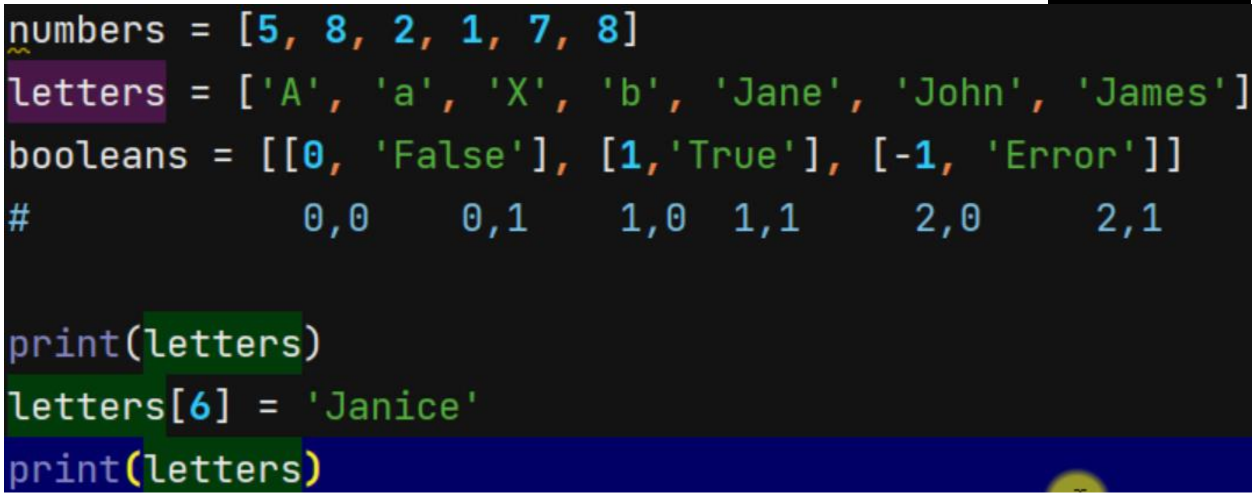
Run and this time we see 5, 8, 2.



```
[5, 8, 2, 1, 7, 8]
[5, 8, 2]
```

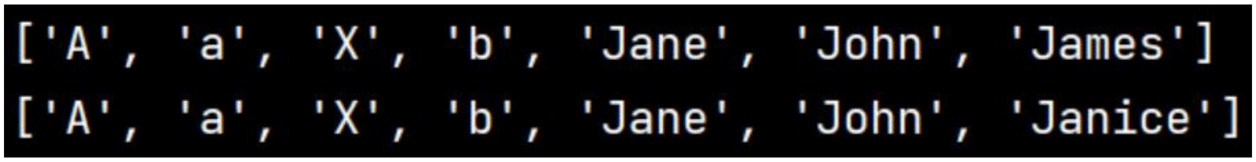
Change A List Value

Now, let's modify the list by changing, adding, removing, and sorting the items. To change a value, we write the list name letters[] then the index number [6] =. 6 is the last item value for James. Assign a new value like 'Janice': print(letters) 2 times.



```
numbers = [5, 8, 2, 1, 7, 8]
letters = ['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
booleans = [[0, 'False'], [1,'True'], [-1, 'Error']]
#            0,0    0,1    1,0  1,1     2,0     2,1

print(letters)
letters[6] = 'Janice'
print(letters)
```

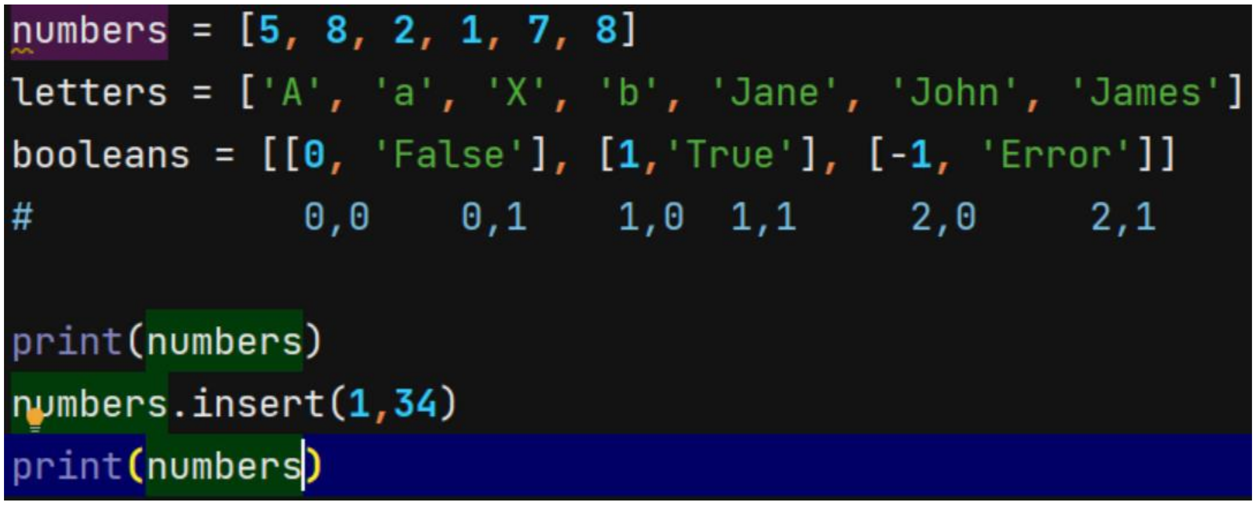
The updated list shows Janice.



```
['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
['A', 'a', 'X', 'b', 'Jane', 'John', 'Janice']
```

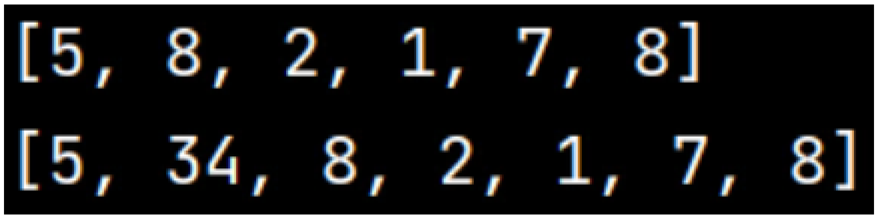
Insert A List Value

There is more than 1 way to add an item to a list. We can insert, append, and extend. To insert, we write numbers.insert(). Insert takes 2 parameters. The 1st parameter is the index where we want to add the item. How about position (1,)? The 2nd parameter is the value we want to add (1, 34). print(numbers) 2 times.



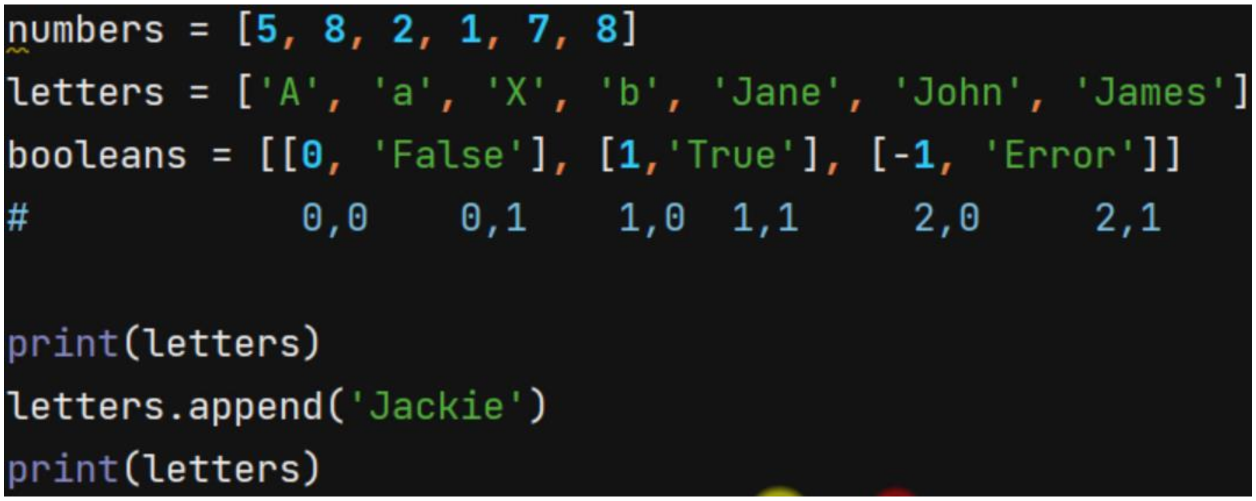
```
numbers = [5, 8, 2, 1, 7, 8]
letters = ['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
booleans = [[0, 'False'], [1,'True'], [-1, 'Error']]
#            0,0    0,1    1,0  1,1     2,0     2,1

print(numbers)
numbers.insert(1,34)
print(numbers)
```

Run and the console shows 34 was inserted at position 1. The other items get pushed to the right of 34.



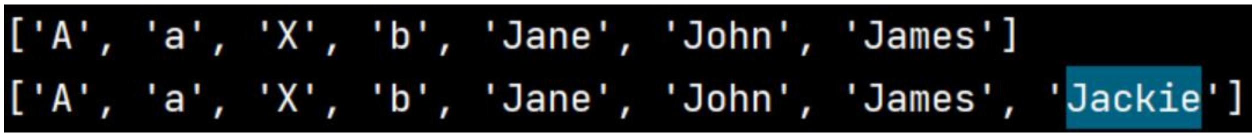
```
[5, 8, 2, 1, 7, 8]
[5, 34, 8, 2, 1, 7, 8]
```

If we wanted to add 1 item to the end of our list then we can use the append() method. Write letters.append(‘Jackie’) then print() letters before and after the append() method.



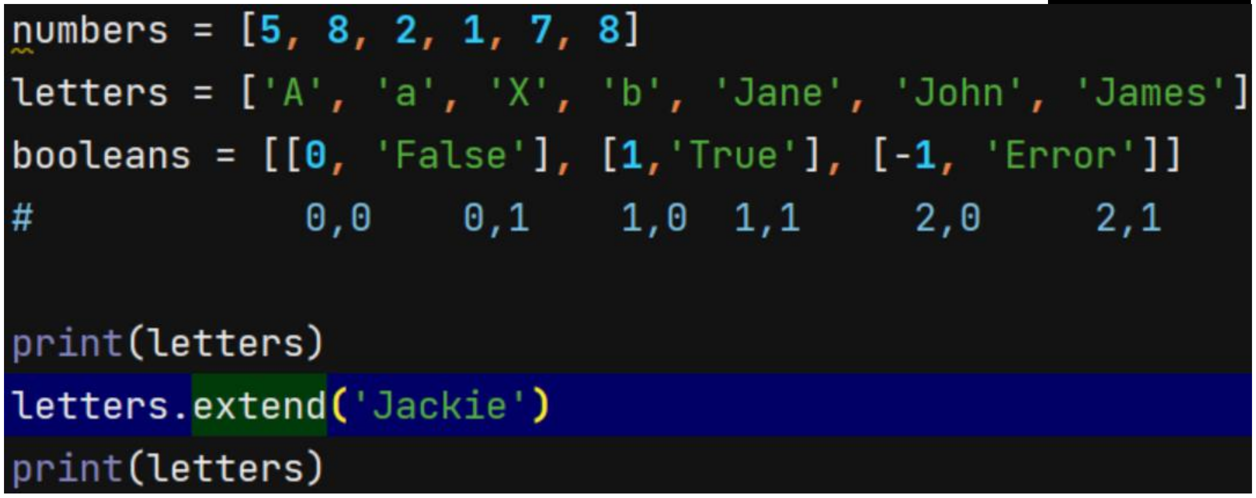
```
numbers = [5, 8, 2, 1, 7, 8]
letters = ['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
booleans = [[0, 'False'], [1,'True'], [-1, 'Error']]
#             0,0    0,1    1,0  1,1     2,0     2,1

print(letters)
letters.append('Jackie')
print(letters)
```

Run and the console shows Jackie.



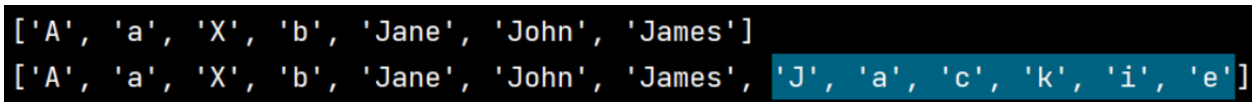
```
['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
['A', 'a', 'X', 'b', 'Jane', 'John', 'James', 'Jackie']
```

The append() and extend() methods are similar and can be confusing. Both methods add to the end of a list. However, it’s best to use append when adding 1 item and extend when adding more than 1 item. Let me show you the difference. We see how append displays Jackie at the end as 1 item. Watch how extend operates.



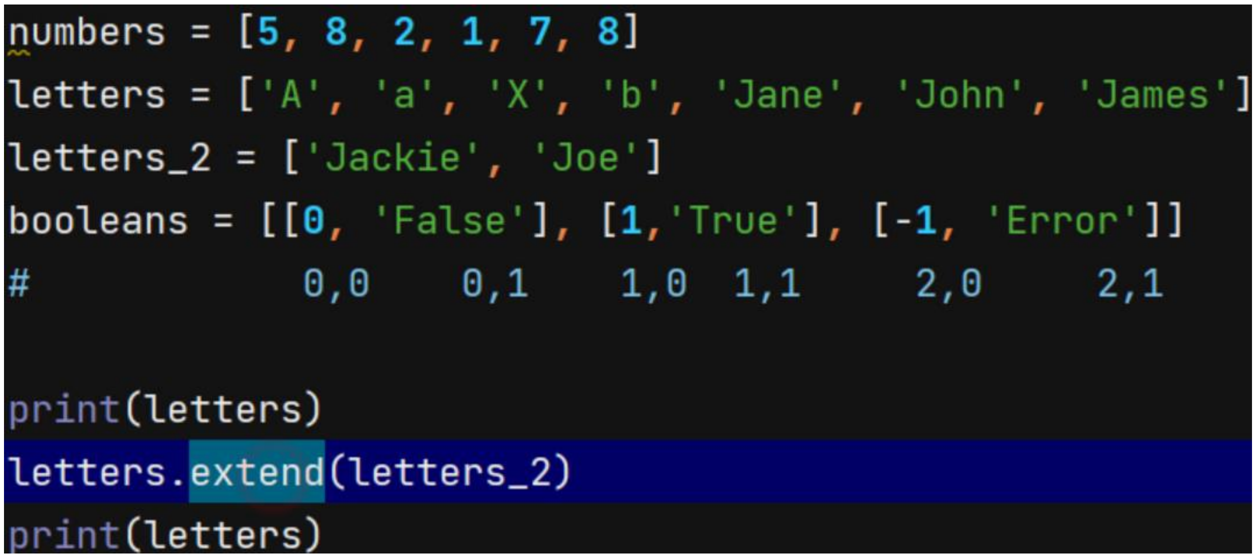
```
numbers = [5, 8, 2, 1, 7, 8]
letters = ['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
booleans = [[0, 'False'], [1,'True'], [-1, 'Error']]
#             0,0    0,1    1,0  1,1     2,0     2,1

print(letters)
letters.extend('Jackie')
print(letters)
```

I'm going to run and we see how the console shows each letter as 1 item.

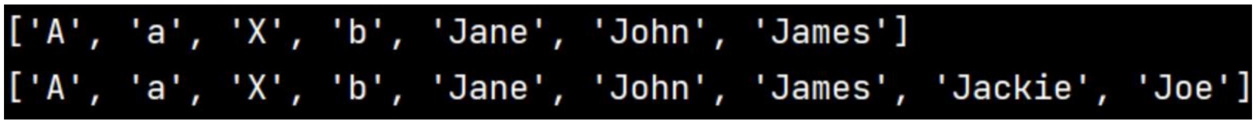


```
['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
['A', 'a', 'X', 'b', 'Jane', 'John', 'James', 'J', 'a', 'c', 'k', 'i', 'e']
```

Watch what happens when adding more than 1 item. Let's say we have another list called letters_2 = [] and pass in some values like ['Jackie', 'Joe']. To extend, we write the list name (letters_2) then run.

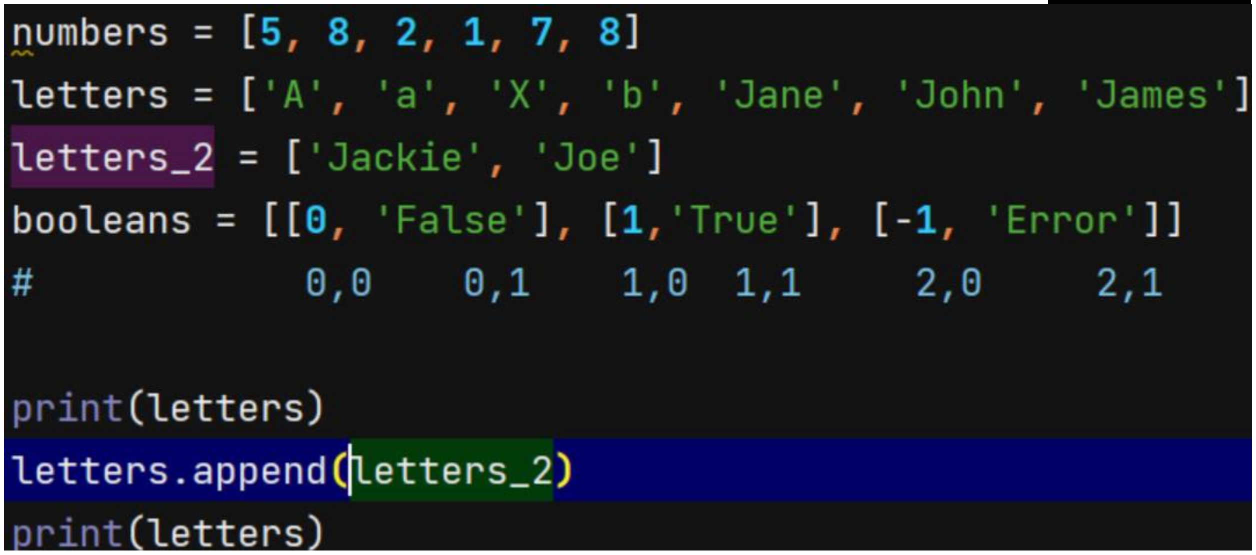


```
numbers = [5, 8, 2, 1, 7, 8]
letters = ['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
letters_2 = ['Jackie', 'Joe']
booleans = [[0, 'False'], [1,'True'], [-1, 'Error']]
#             0,0    0,1    1,0  1,1     2,0     2,1

print(letters)
letters.extend(letters_2)
print(letters)
```

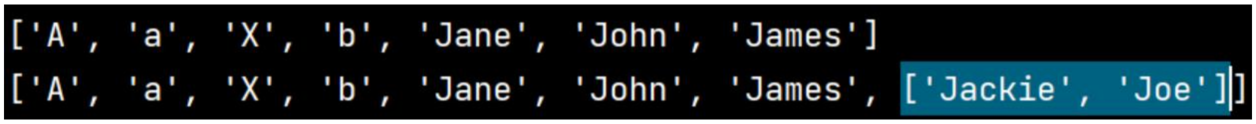


```
['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
['A', 'a', 'X', 'b', 'Jane', 'John', 'James', 'Jackie', 'Joe']
```

Jackie and Joe are added to the end of the list. That's good. However, when it comes to append, Jackie and Joe. When I run, Jackie and Joe are still added to the end but look how it's added to the list.

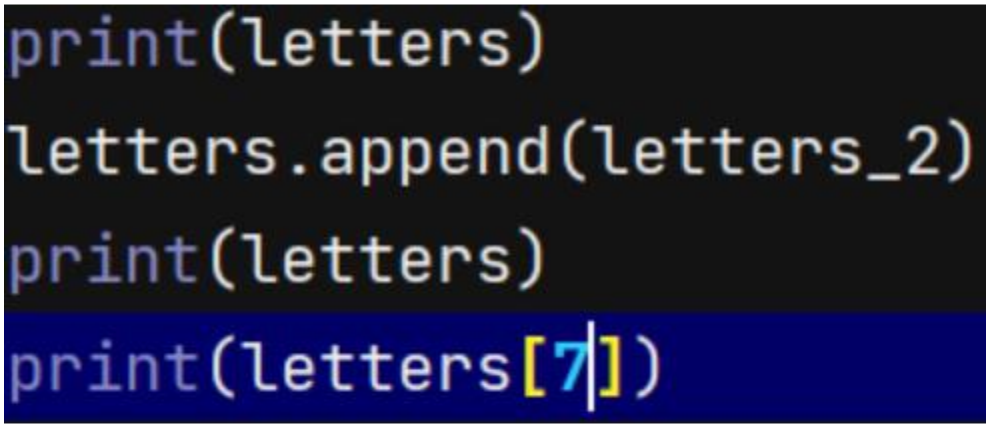


```
numbers = [5, 8, 2, 1, 7, 8]
letters = ['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
letters_2 = ['Jackie', 'Joe']
booleans = [[0, 'False'], [1,'True'], [-1, 'Error']]
#             0,0    0,1    1,0  1,1     2,0     2,1

print(letters)
letters.append(letters_2)
print(letters)
```

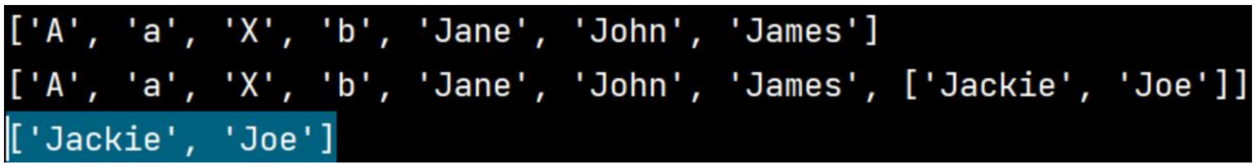


```
['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
['A', 'a', 'X', 'b', 'Jane', 'John', 'James', ['Jackie', 'Joe']]
```

It's added to the end of the list by looking like a list within a list. Just like the boolean list of values. We prefer not to have a list within a list but have each individual name. Let me show you why. Add one more print() statement at line 10 and return letters and pass in [7] then run.



```
print(letters)
letters.append(letters_2)
print(letters)
print(letters[7])
```

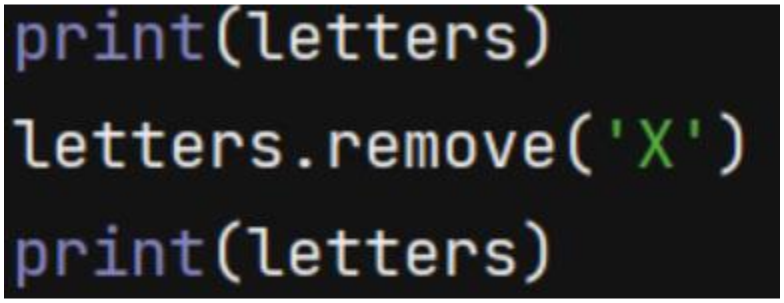


```
['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
['A', 'a', 'X', 'b', 'Jane', 'John', 'James', ['Jackie', 'Joe']]
['Jackie', 'Joe']
```

The console shows complete list and not just Jackie. That's why it's best to use append for adding 1 item and extend for more than 1 item.

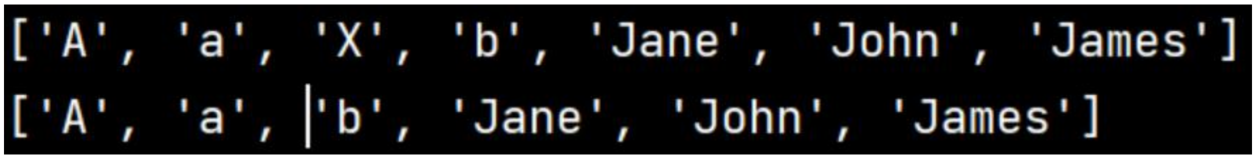
Remove a List Item

Let me show you about remove. letters.remove() then pass in the value like ('X').



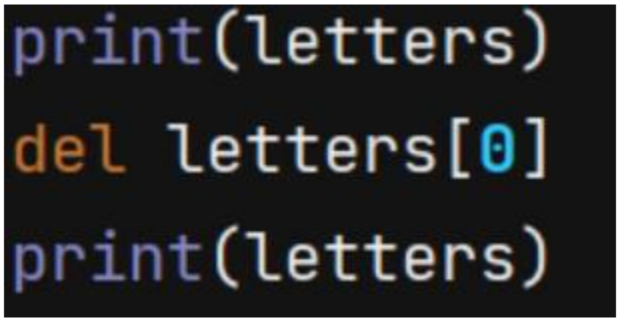
```
print(letters)
letters.remove('X')
print(letters)
```

Run and the console does not show the letter ‘X’. It was removed from the list.

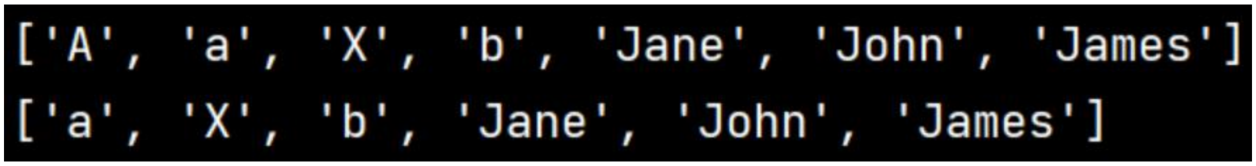


```
['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
['A', 'a', 'b', 'Jane', 'John', 'James']
```

If you know the position of an item then you can use the del statement then write the list which is name letters[] and pass [0] then run.

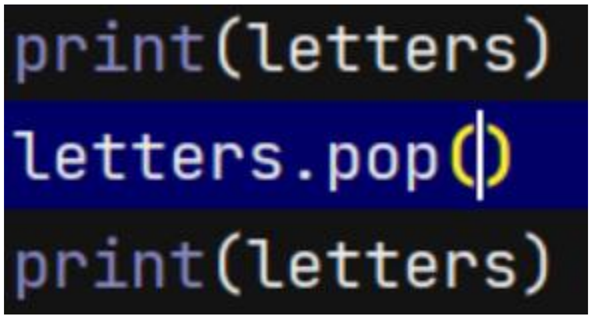


```
print(letters)
del letters[0]
print(letters)
```

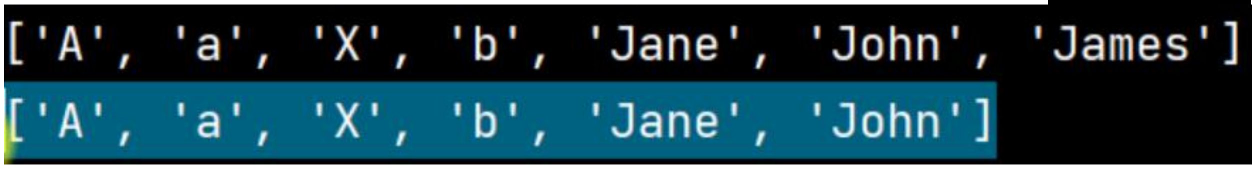


```
['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
['a', 'X', 'b', 'Jane', 'John', 'James']
```

We see the capital letter ‘A’ was removed from the list. Another method is the pop() method. By default, it removes the last item from a list. We write letters.pop() then Run.

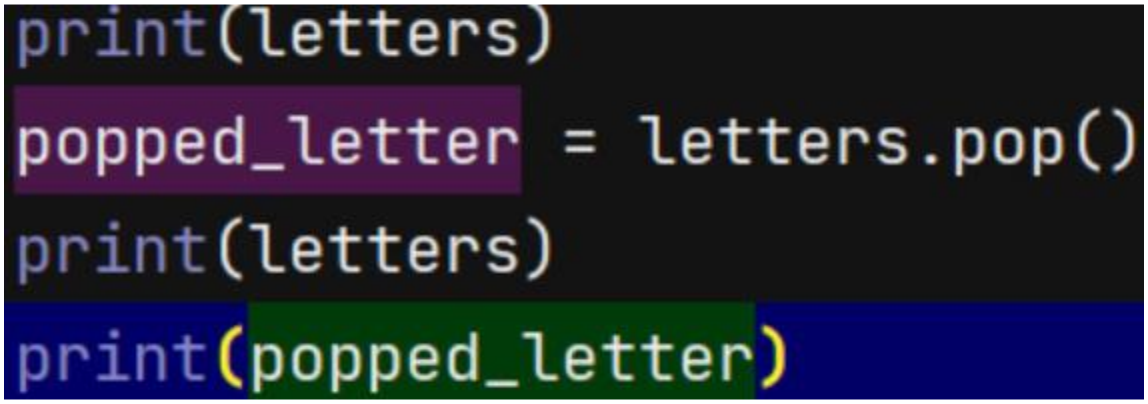


```
print(letters)
letters.pop()
print(letters)
```

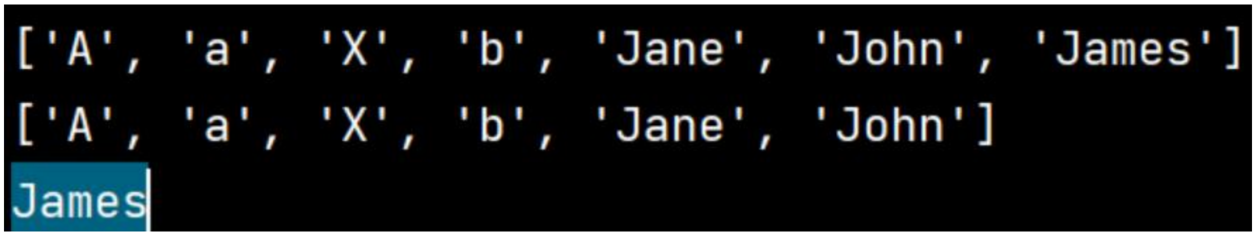


```
['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
['A', 'a', 'X', 'b', 'Jane', 'John']
```

We see James was removed from this new formed list. Pop is unique because it allows us to remove an item but still work with the item. Assign letters.pop() to a value like popped_letter =. I'm going to add popped_letter by adding another print() statement then run.



```
print(letters)
popped_letter = letters.pop()
print(letters)
print(popped_letter)
```

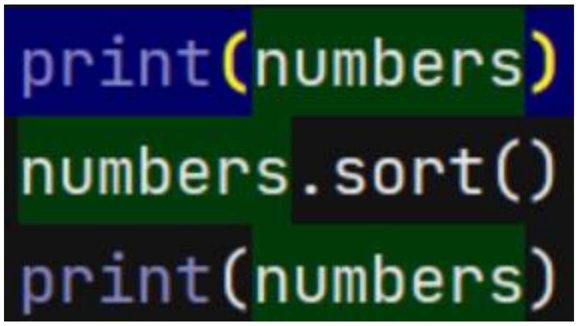


```
['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
['A', 'a', 'X', 'b', 'Jane', 'John']
James
```

We see James was removed from the list but also displayed at the end. So we can still play with the list.

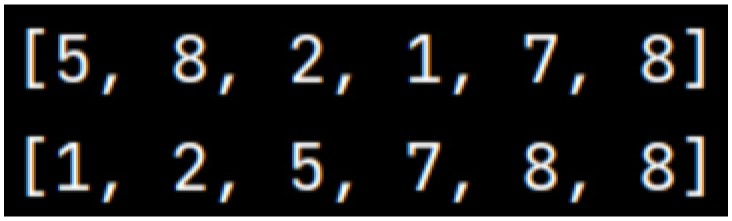
Sort A List Item

For sorting an item, we write numbers.sort(), print(numbers) before and after the sort() method.



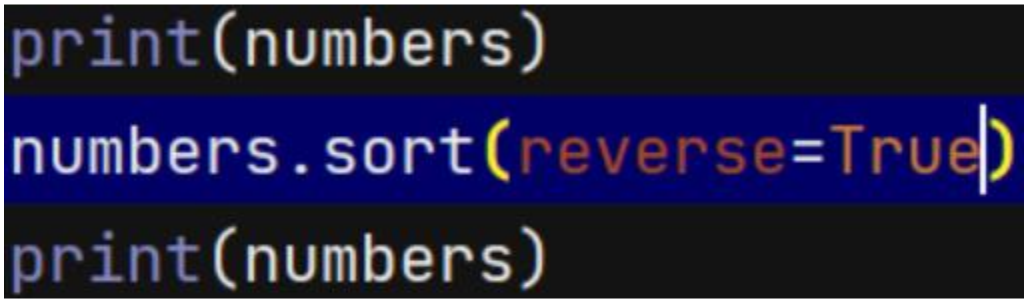
```
print(numbers)
numbers.sort()
print(numbers)
```

We run and the items are in ascending order 1, 2, 5, 7, 8, 8.



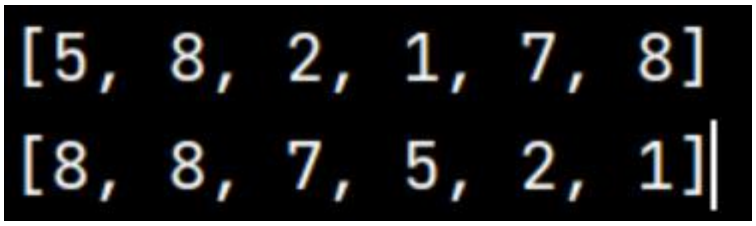
```
[5, 8, 2, 1, 7, 8]
[1, 2, 5, 7, 8, 8]
```

If you want the items in descending order then pass in (reverse=True).



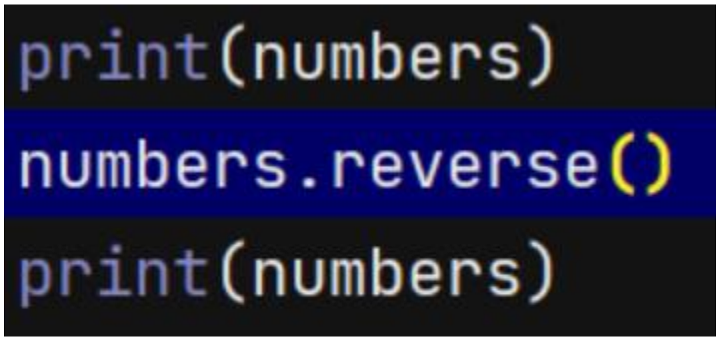
```
print(numbers)
numbers.sort(reverse=True)
print(numbers)
```

So, now when I run. It starts from the back by showing 8, 8, 7, 5, 2, 1.



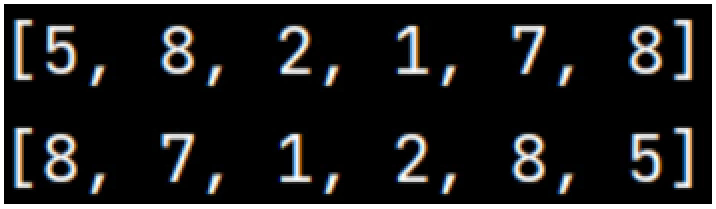
```
[5, 8, 2, 1, 7, 8]
[8, 8, 7, 5, 2, 1]
```

We can reverse the list to show the last item 1st and the 1st item last by writing numbers.reverse().



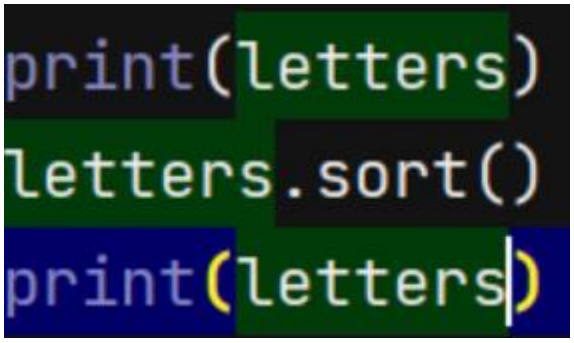
```
print(numbers)
numbers.reverse()
print(numbers)
```

This time when I run. We see it shows 8 which was the last item and 5 which was the 1st item last.

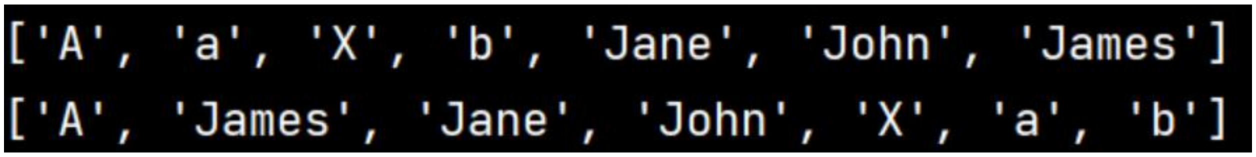


```
[5, 8, 2, 1, 7, 8]
[8, 7, 1, 2, 8, 5]
```

When it comes to sorting alphabets, this is interesting because if I write letters.sort(). Watch what happens when we print.

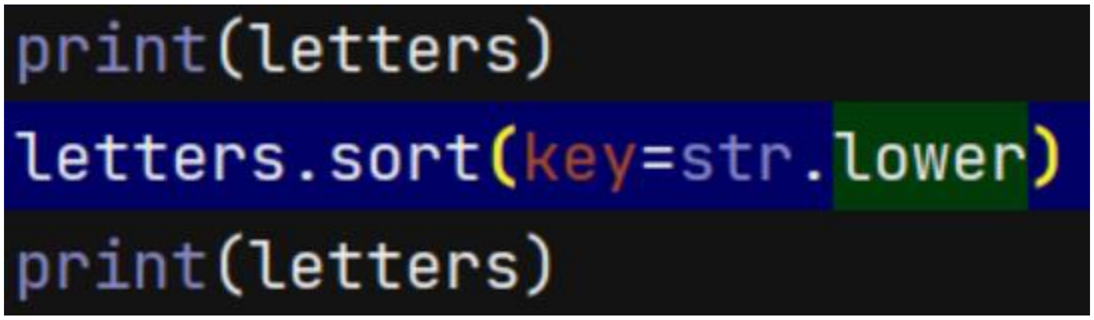


```
print(letters)
letters.sort()
print(letters)
```

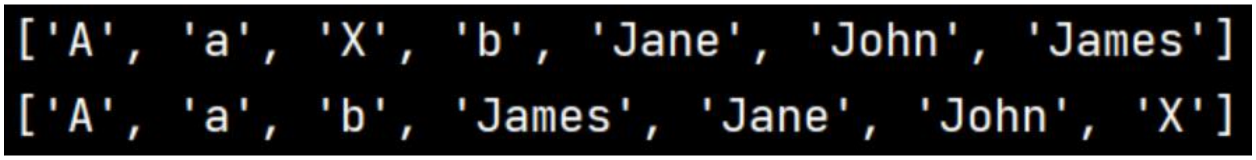


```
['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
['A', 'James', 'Jane', 'John', 'X', 'a', 'b']
```

Notice, the lowercase letters a and b are at the end of the list. They are not sorted in alphabetical order. The sort method uses an ASCIIbetical order and not alphabetical order. ASCIIbetical order means all uppercase letters come before the lowercase letters. To sort in alphabetical order, we can pass in (key=str.lower) then run.



```
print(letters)
letters.sort(key=str.lower)
print(letters)
```



```
['A', 'a', 'X', 'b', 'Jane', 'John', 'James']
['A', 'a', 'b', 'James', 'Jane', 'John', 'X']
```

Now, we see the list in alphabetical order.

Rex
Jones II

Contact Info

- ✓ Email Rex.Jones@Test4Success.org
- ✓ YouTube https://www.youtube.com/c/RexJonesII/videos
- ✓ Facebook https://facebook.com/JonesRexII
- ✓ Twitter https://twitter.com/RexJonesII
- ✓ GitHub https://github.com/RexJonesII/Free-Videos
- ✓ LinkedIn https://www.linkedin.com/in/rexjones34/