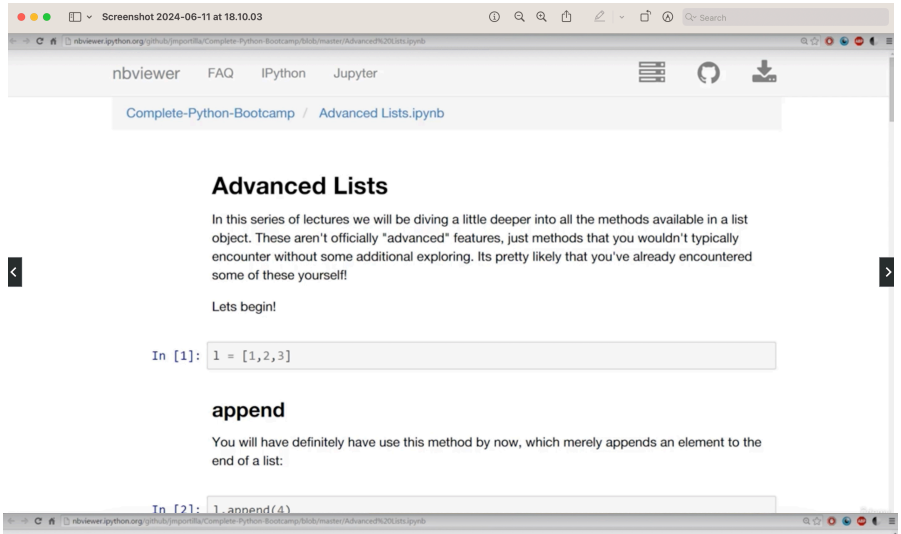
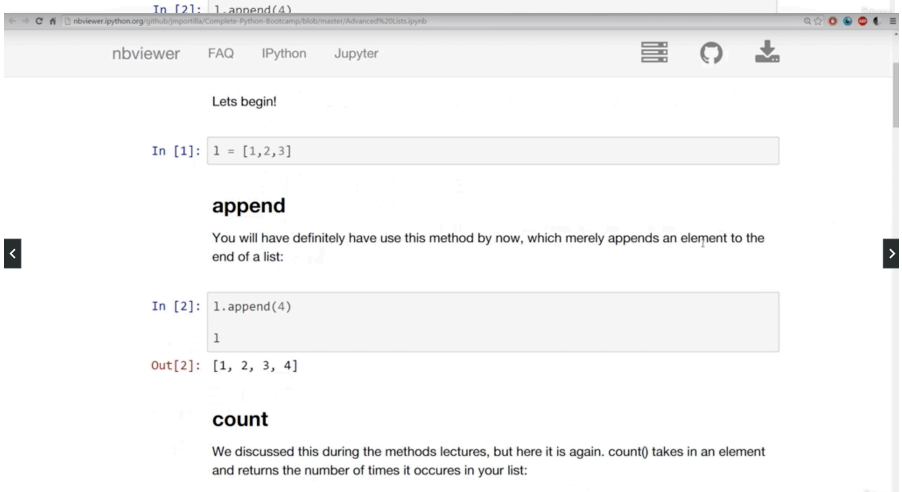
SECTION 20: ADVANCED PYTHON PROJECTS AND DATA STRUCTURES - 41 minutes, 7 parts

5/7 Advanced Lists

- -> advanced lists
- -> methods for lists
- -> methods which you wouldn't typically encounter
- -> he defines an array
- -> then uses the .append() method to add an element to the end of this
- -> the .count() method for this counts how many times an object occurs in the list



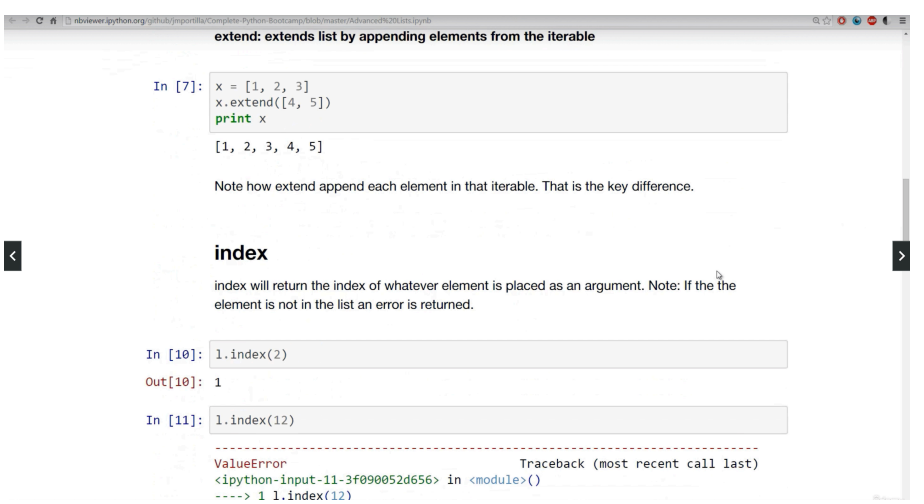
- **-> the .extend() method**
  - -> he makes an array
  - -> then appends another list to it
  - -> he's appended an array to an array
  - -> this adds an element to the array, and that entire element is a new array
  - -> compared to the .extend() method, which extends the original array to another



- **-> the index() method returns the index of the argument which is passed into it**
  - -> an error is returned if this is not the case

- **-> insert()**
  - -> this method takes two arguments
  - -> we have an array of numbers
  - -> the first element in this argument is the index in the array (list)
  - -> the second argument is the content of the element which we want at that point in the list
  - -> he then returns the list to check this has worked

- **-> to find the index of an object**
  - -> .pop() <- this method is to pop off the last element of the list
  - -> an index can be passed into this method
  - -> this will remove the element at that index of the list
  - -> once the element is popped off, this can't be undone

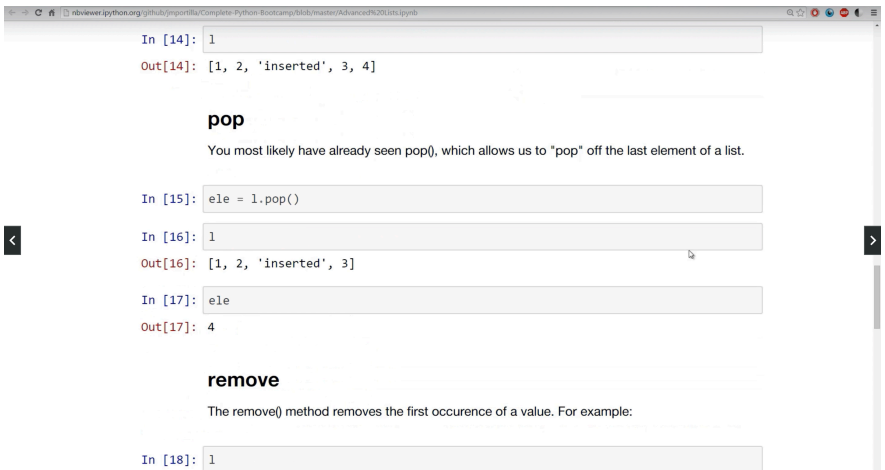


- **-> the .remove() method**

- -> this removes the first occurrence of a value
- -> the argument of this is the element in the list which we want to remove
- -> it then removes the first time in the list where we see this element

- **-> the .reverse() method**
  - -> this method reverses the list and the indices
  - -> this occurs in-place, which affects the list permanently
  - -> this is like the .pop() method (the list is changed, rather than a new one returned)

- **-> overview of methods**
  - -> .append <- add an element to the end of the list
  - -> .count <- count the number of object which take place in the list
  - -> .extend <- append each element in the iterable
    - -> adding a list to the end of another list
  - -> .index <- to return the index of the element passed into the argument
    - -> you have to pass in a list of there will be an error returned
  - -> .insert <- this takes the index and the object
  - -> .pop <- remove the last element of the list
    - -> an index can also be passed into this
  - -> .remove <- remove the first ocurrance of a value
  - -> .reverse <- reverses the list
  - -> .sort <- to sort the list in place



In [14]: l
Out[14]: [1, 2, 'inserted', 3, 4]

pop

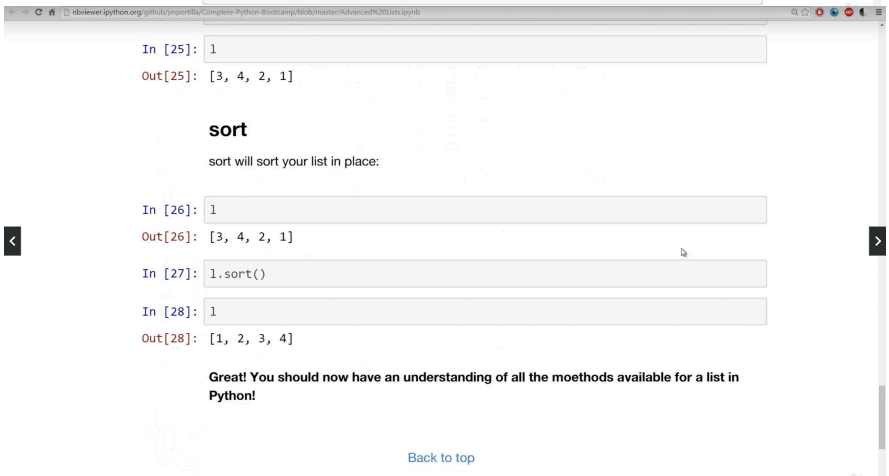
You most likely have already seen pop(), which allows us to "pop" off the last element of a list.

In [15]: ele = l.pop()
In [16]: l
Out[16]: [1, 2, 'inserted', 3]
In [17]: ele
Out[17]: 4

remove

The remove() method removes the first occurence of a value. For example:

In [18]: l



In [25]: l
Out[25]: [3, 4, 2, 1]

sort

sort will sort your list in place:

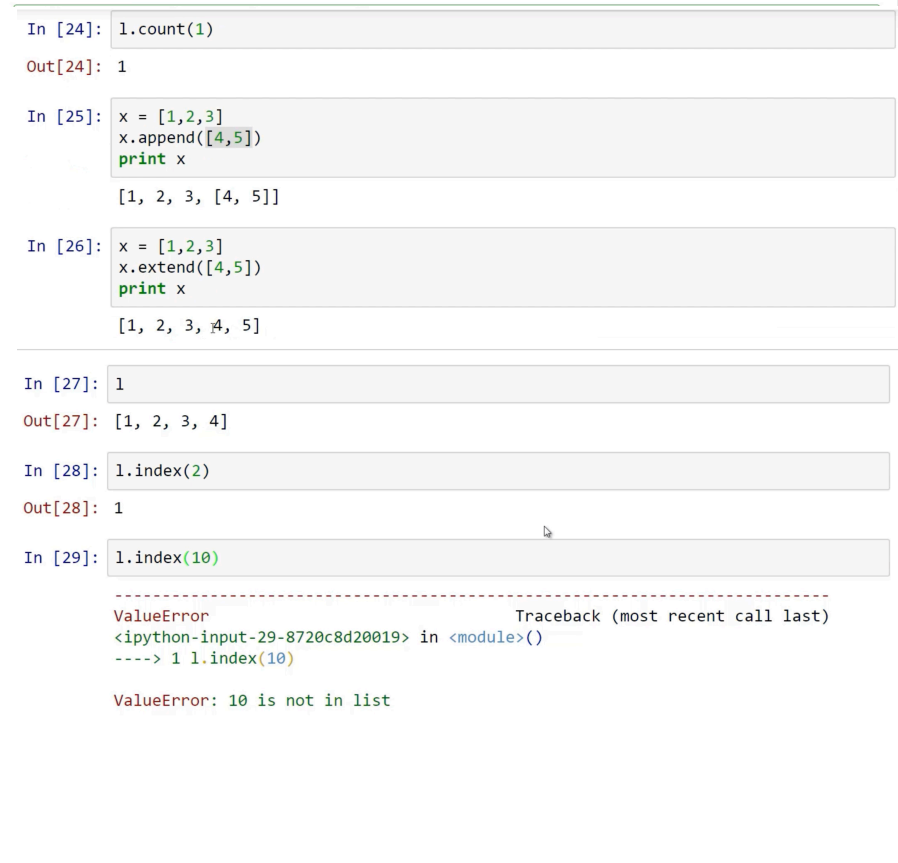
In [26]: l
Out[26]: [3, 4, 2, 1]
In [27]: l.sort()
In [28]: l
Out[28]: [1, 2, 3, 4]

Great! You should now have an understanding of all the moethods available for a list in Python!

Back to top

Advanced Lists

In [20]: l = [1,2,3]
In [21]: l.append(4)
In [22]: l
Out[22]: [1, 2, 3, 4]
In [23]: l.count(10)
Out[23]: 0



In [24]: l.count(1)
Out[24]: 1

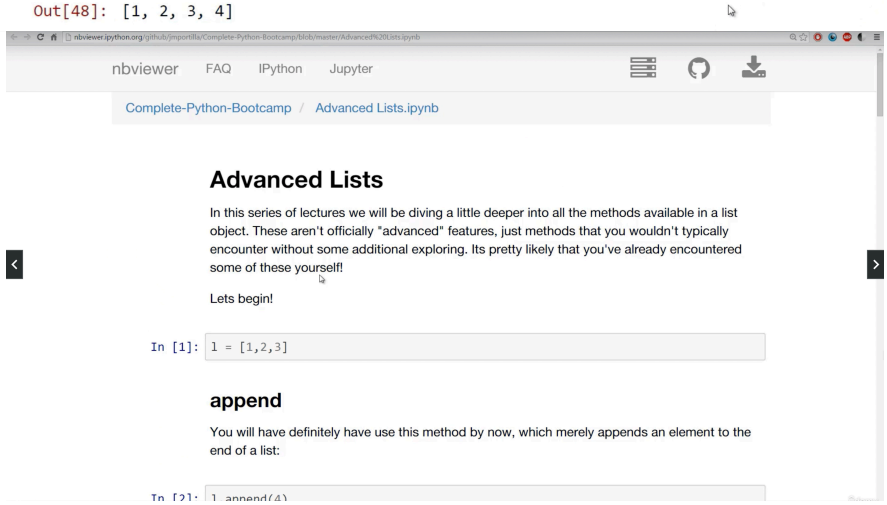
In [25]:
```
x = [1,2,3]
x.append([4,5])
print x
```
[1, 2, 3, [4, 5]]

In [26]:
```
x = [1,2,3]
x.extend([4,5])
print x
```
[1, 2, 3, 4, 5]

In [27]: l
Out[27]: [1, 2, 3, 4]
In [28]: l.index(2)
Out[28]: 1
In [29]: l.index(10)

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-29-8720c8d20019> in <module>()
----> 1 l.index(10)

ValueError: 10 is not in list
```

```
In [30]: l
Out[30]: [1, 2, 3, 4]

In [31]: l.insert(2,'inserted')

In [32]: l
Out[32]: [1, 2, 'inserted', 3, 4]

In [33]: l
Out[33]: [1, 2, 'inserted', 3, 4]

In [ ]: ele = l.pop()

In [35]: ele
Out[35]: 4

In [36]: l
Out[36]: [1, 2, 'inserted', 3]

In [37]: l.pop(0)
Out[37]: 1

In [38]: l
Out[38]: [2, 'inserted', 3]

In [39]: l.remove('inserted')

In [40]: l
Out[40]: [2, 3]

In [41]: l = [1,2,3,4,3]

In [42]: l.remove(3)

In [43]: l
Out[43]: [1, 2, 4, 3]

In [43]: l
Out[43]: [1, 2, 4, 3]

In [44]: l
Out[44]: [1, 2, 4, 3]

In [45]: l.reverse()

In [46]: l
Out[46]: [3, 4, 2, 1]

In [46]: l
Out[46]: [3, 4, 2, 1]

In [47]: l.sort()

In [48]: l
Out[48]: [1, 2, 3, 4]
```



nbviewer FAQ IPython Jupyter

Complete-Python-Bootcamp / Advanced Lists.ipynb

Advanced Lists

In this series of lectures we will be diving a little deeper into all the methods available in a list object. These aren't officially "advanced" features, just methods that you wouldn't typically encounter without some additional exploring. Its pretty likely that you've already encountered some of these yourself!

Lets begin!

```
In [1]: l = [1,2,3]
```

append

You will have definitely have use this method by now, which merely appends an element to the end of a list:

```
In [2]: l.append(4)
```

Lets begin!

In [1]:
```
l = [1,2,3]
```

append

You will have definitely have use this method by now, which merely appends an element to the end of a list:

In [2]:
```
l.append(4)

l
```

Out[2]: [1, 2, 3, 4]

count

We discussed this during the methods lectures, but here it is again. count() takes in an element and returns the number of times it occures in your list:

nbviewer.ipython.org/github/jmportilla/Complete-Python-Bootcamp/blob/master/Advanced%20Lists.ipynb

extend

Many times people find the difference between extend and append to be unclear. So note:

append: Appends object at end

In [6]:
```
x = [1, 2, 3]
x.append([4, 5])
print x
```

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

extend: extends list by appending elements from the iterable

In [7]:
```
x = [1, 2, 3]
x.extend([4, 5])
print x
```

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

Note how extend append each element in that iterable. That is the key difference.

index

nbviewer.ipython.org/github/jmportilla/Complete-Python-Bootcamp/blob/master/Advanced%20Lists.ipynb

Note how extend append each element in that iterable. That is the key difference.

index

index will return the index of whatever element is placed as an argument. Note: If the the element is not in the list an error is returned.

In [10]:
```
l.index(2)
```

Out[10]: 1

In [11]:
```
l.index(12)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-11-3f090052d656> in <module>()
----> 1 l.index(12)

ValueError: 12 is not in list
```

insert

insert takes in two arguments: insert(index,object) This method places the object at the index supplied. For example:

nbviewer.ipython.org/github/jmportilla/Complete-Python-Bootcamp/blob/master/Advanced%20Lists.ipynb

insert

insert takes in two arguments: insert(index,object) This method places the object at the index supplied. For example:

In [12]:
```
l
```

Out[12]: [1, 2, 3, 4]

In [13]:
```
# Place a letter at the index 2
l.insert(2,'inserted')
```

In [14]:
```
l
```

Out[14]: [1, 2, 'inserted', 3, 4]

pop

You most likely have already seen pop(), which allows us to "pop" off the last element of a list.

In [15]:
```
ele = l.pop()
```

In [16]:
```
l
```

pop

You most likely have already seen pop(), which allows us to "pop" off the last element of a list.

```
In [15]: ele = l.pop()

In [16]: l
Out[16]: [1, 2, 'inserted', 3]

In [17]: ele
Out[17]: 4
```

remove

The remove() method removes the first occurence of a value. For example:

```
In [18]: l
Out[18]: [1, 2, 'inserted', 3]

In [19]: l.remove('inserted')

In [20]: l
```

nbviewer.ipython.org/github/jmportilla/Complete-Python-Bootcamp/blob/master/Advanced%20Lists.ipynb

```
In [17]: ele
Out[17]: 4
```

remove

The remove() method removes the first occurence of a value. For example:

```
In [18]: l
Out[18]: [1, 2, 'inserted', 3]

In [19]: l.remove('inserted')

In [20]: l
Out[20]: [1, 2, 3]

In [21]: l = [1,2,3,4,3]

In [22]: l.remove(3)

In [23]: l
Out[23]: [1, 2, 4, 3]
```

nbviewer.ipython.org/github/jmportilla/Complete-Python-Bootcamp/blob/master/Advanced%20Lists.ipynb

reverse

As you might have guessed, reverse() reverses a list. Note this occurs in place! Meaning it effects your list permanently.

```
In [24]: l.reverse()

In [25]: l
Out[25]: [3, 4, 2, 1]
```

sort

sort will sort your list in place:

```
In [26]: l
Out[26]: [3, 4, 2, 1]

In [27]: l.sort()

In [28]: l
Out[28]: [1, 2, 3, 4]
```