Lecture_4_Code_Lists_Tuples_Comprehensions

April 19, 2017

```
In [22]: fruit = ["apples","pears","bananas","cherimoyas"]

         print (fruit[::-1])
```

```
['cherimoyas', 'bananas', 'pears', 'apples']
```

0.1 Basic List Operations

0.1.1 Lists have a length

```
In [23]: print (len(fruit))
```

```
4
```

0.1.2 Lists can be concatenated

```
In [24]: #a friendly reminder that + works the same for strings
         fruit1 = ["apples","pears"]
         fruit2 = ["bananas","cherimoyas"]

         print (fruit1 + fruit2)
         print (fruit2 + fruit1)
         fruit = fruit1 + fruit2
         print (fruit)
```

```
['apples', 'pears', 'bananas', 'cherimoyas']
['bananas', 'cherimoyas', 'apples', 'pears']
['apples', 'pears', 'bananas', 'cherimoyas']
```

0.1.3 We can check if an element is inside a list

```
In [33]: fruit = ["apples","pears",["bananas","cherimoyas"]]
         if "b" in fruit[2][0]:
             print ("Yup looks like we got bananas!")
         else:
             print ("There doesn't seem to be any pears in the fruit list")
```

```
Yup looks like we got bananas!
```

In [ ]:

In [ ]:

0.1.4 We can pass lists to functions

```
In [29]: fruits = ["apples","pears",["bananas","cherimoyas"]]
         my_fruit = "bananas"

         def checker(a_list,a_query):
             if a_query in a_list:
                 print ("Query found!")
             else:
                 print ("Query not found")

         checker(fruits,my_fruit)
```

```
Query not found
```

0.1.5 Lists elements can be multiplied

```
In [1]: fruits = ["apples","pears","bananas","cherimoyas"]
        fruits = fruits * 2
        print (fruits)

        #remember strings?
        name = "Spam"
        print (name*3)
```

```
['apples', 'pears', 'bananas', 'cherimoyas', 'apples', 'pears', 'bananas', 'cherimo
SpamSpamSpam
```

0.1.6 Iterating through lists

```
In [1]: #accessing list elements
        fruits = ["apples","pears","bananas","cherimoyas"]
        for i in fruits:
            print (i)
        #accessing list elements by index
        for i in range(0,len(fruits)):
            print (fruits[i])
        #accessing list elements by index with a while loop
        x = 0
```

```
        while x < 4:
            print (fruits[x])
            x += 1

        #of course instead of print we can do something fancier here
```

```
apples
pears
bananas
cherimoyas
apples
pears
bananas
cherimoyas
apples
pears
bananas
cherimoyas
```

0.1.7 List functions

```
In [2]: #len, max, min, list, cmp
        fruits = ["apples","pears","bananas","cherimoyas","peart"]
        fruits_1 = ["apples","pears"]
        #examine this closely for lists of strings
        print (max(fruits))

        #range object will give me a sequence of numbers
        #I can typecast that to make a list
        my_nums = list(range(1,6))
        print (type(range(1,6)))
        print (my_nums)
        print (min(my_nums))

        #let's see what happens if we typecast a string
        #the sequence gets converted to a list
        a = list("Spam")
        print (a)
        #we can make a string back out of a list
        b = "".join(a)
        print (b)
```

```
peart
<class 'range'>
[1, 2, 3, 4, 5]
1
['S', 'p', 'a', 'm']
```

```
Spam
```

0.2 List Methods

0.2.1 append and extend

```
In [37]: #append will take only one object
         my_list = []
         print (my_list)
         my_list.append("Spam")
         print (my_list)

         #let's try extend, extend works with a sequence of many objects
         #here is a tuple example
         #contents of the tuple extend the list
         my_list.extend(("Cheese","Bread","Wine"))
         print (my_list)

         #it's better and slightly less expensive though to do
         my_list += ["Butter"]
         print (my_list)
```

```
[]
['Spam']
['Spam', 'Cheese', 'Bread', 'Wine']
['Spam', 'Cheese', 'Bread', 'Wine', 'Butter']
```

0.2.2 index and count

```
In [38]: num_list = [1,2,3,3,3,4,5]
         print (num_list.count(3))
         print (num_list.index(3))
```

```
3
2
```

0.2.3 inserting elements at an index location

```
In [39]: #here 2 is the index and 3 is what we are injecting at that index
         num_list = [1,2,4,5]
         num_list.insert(2,3)
         print (num_list)

         #inserting more elements
         #we can use this slicing trick
         other_list = [4.5,4.9]
```

```
        num_list = num_list[:4] + other_list + num_list[4:]
        print (num_list)
```

```
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 4.5, 4.9, 5]
```

0.2.4 Popping and removing

```
In [40]: num_list = [1,2,4,5,6,7,["a","b"]]
         # we will now get the last element which is ["a","b"]
         popped = num_list.pop()
         print (popped)
         print (num_list)

         #we can remove a value from a list
         #removes only the first occurence
         num_list = [1,2,4,4,4,4,5]
         num_list.remove(4)
         print (num_list)
```

```
['a', 'b']
[1, 2, 4, 5, 6, 7]
[1, 2, 4, 4, 4, 5]
```

0.2.5 Reversing and Sorting

```
In [41]: num_list = [5,4,2,3,1]
         #reversing the values in place
         num_list.reverse()
         print (num_list)

         #sorting a list
         #find out what algorithm python users for sort
         num_list.sort()
         print (num_list)
```

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

```
In [ ]: #remember this means :len(fruit)-2 = 0:4-2 = 0:2
        print (fruit[:-2])
```

0.2.6 Lists are mutable

remember when we tried this with strings it didn't work

Strings are not mutable

```
In [42]: new_fruit = fruit
         print (new_fruit)
         fruit[0] = "peaches"
         print (new_fruit)
```

```
['apples', 'pears', ['bananas', 'cherimoyas']]
['peaches', 'pears', ['bananas', 'cherimoyas']]
```

How can we copy a list?

```
In [ ]: print (fruit)
```

```
In [50]: c = "Hello World"
         d = c
         e = "Hello World"
         print (id(c),id(d),id(e))

         print (c is d is e)

         a = [1,2,3,4,5]
         b = a
         print (id(a))
         print (id(b))

         a[0] = 0
         print (a)
         print (b)
```

```
4562828464 4562828464 4562827184
False
4562826440
4562826440
[0, 2, 3, 4, 5]
[0, 2, 3, 4, 5]
```

```
In [2]: from copy import copy
        fruit = ["apples","pears","bananas","cherimoyas"]


        #easiest most common way
        fruit_copy = fruit[::]
        print (fruit)
        #another way
        fruit_copy = list(fruit)
        fruit[0] = "blueberries"
        print (fruit)
        print (fruit_copy)
```

```
['apples', 'pears', 'bananas', 'cherimoyas']
['blueberries', 'pears', 'bananas', 'cherimoyas']
['apples', 'pears', 'bananas', 'cherimoyas']
```

0.3 Tuples

```
In [3]: tup1 = ("Erlich","Bachman","347-92-1234")
        print (type(tup1[0:2]))
        #creating a new tuple from slicing
        a = tup1[0:2]
        print (a)

        #length of tuples
        print (len(tup1))

        tup2 = (1,2,3,4,5)
        #minimum mand maximum values of tuples
        print (max(tup2))
        print (min(tup2))

        #accessing elements of a tuple
        print (tup1[2])

        #create a new tuple by concatenation
        tup3 = tup1 + tup2

        print (tup3)

        #if you have one element you must include a comma
        tup4 = (4)
        print (type(tup4))
        #if you want a tuple of one element
        tup4 = (4,)
        print (type(tup4))

        #we can check if something is in a tuple with the in operator
        if 3 in tup2:
            print ("Got 3")


        #tup5 repetition *
        tup5 = ("Hello") * 5
        print (tup5)
```

```
<class 'tuple'>
('Erlich', 'Bachman')
3
```

```
5
1
347-92-1234
('Erlich', 'Bachman', '347-92-1234', 1, 2, 3, 4, 5)
<class 'int'>
<class 'tuple'>
Got 3
HelloHelloHelloHelloHello
```

0.4 List Comprehensions

```
In [52]: my_nums = list(range(0,11))
         y = 10

         a = [x + y for x in my_nums if x%2==1]
         print (a)
```

```
[11, 13, 15, 17, 19]
```

```
In [ ]:
```