[Lists](http://www.pythonforbeginners.com/python-lists-cheat-sheet/) ***2***

This is a new serie of articles here at Python for beginners, that are supposed to

be a starting point for completely beginners of Python.

See it as a cheat sheet, reference, manual or whatever you want.

The purpose is to very short write down the basics of Python.

This page will describe how to use lists in Python.

What is a List?

The simplest data structure in Python and is used to store a list of values.

Lists are collections of items (strings, integers, or even other lists).

Each item in the list has an assigned index value.

Lists are enclosed in [ ]

Each item in a list is separated by a comma

Unlike strings, lists are mutable, which means they can be changed.

List Creation

Lists are created using a comma separated list of values surrounded by

square brackets.

Lists hold a sequence of values (just like strings can hold a sequence

of characters).

Lists are very easy to create, these are some of the ways to make lists

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | emptyList = [ ]    list1 = ['one, two, three, four, five']    numlist = [1, 3, 5, 7, 9]    mixlist = ['yellow', 'red', 'blue', 'green', 'black']    An empty list is created using just square brackets:  list = [] |

List Length

With the length function we can get the length of a list

|  |  |
| --- | --- |
| 1  2  3 | list = ["1", "hello", 2, "world"]  len(list)  >>4 |

List Append

List append will add the item at the end.

If you want to add at the beginning, you can use the insert function (see below)

list.insert(0, "Files")

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | list = ["Movies", "Music", "Pictures"]    list.append(x) will add an element to the end of the list  list.append("Files")    print list  ['Movies', 'Music', 'Pictures', 'Files'] |

List Insert

The syntax is: list.insert(x, y) #will add element y on the place before x

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | list = ["Movies", "Music", "Pictures"]    list.insert(2,"Documents")    print list  ['Movies', 'Music', 'Documents', 'Pictures', 'Files']    You can insert a value anywhere in the list    list = ["Movies", "Music", "Pictures"]  list.insert(3, "Apps") |

List Remove

To remove an element's first occurrence in a list, simply use list.remove

The syntax is: list.remove(x)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | List = ['Movies', 'Music', 'Files', 'Documents', 'Pictures']    list.remove("Files")    print list  ['Movies', 'Music', 'Documents', 'Pictures']    a = [1, 2, 3, 4]  a.remove(2)  print a  [1, 3, 4] |

List Extend

The syntax is: list.extend(x) #will join the list with list x

|  |  |
| --- | --- |
| 1  2  3  4  5 | list2 = ["Music2", "Movies2"]  list1.extend(list2)    print list1  ['Movies', 'Music', 'Documents', 'Pictures', 'Music2', 'Movies2'] |

List Delete

Use del to remove item based on index position.

|  |  |
| --- | --- |
| 1  2  3  4  5 | list = ["Matthew", "Mark", "Luke", "John"]  del list[1]    print list  >>>Matthew, Luke, John |

List Keywords

The keyword "in" can be used to test if an item is in a list.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | list = ["red", "orange", "green", "blue"]  if "red" in list:      do\_something()    #Keyword "not" can be combined with "in".    list = ["red", "orange", "green", "blue"]  if "purple" not in list:      do\_something() |

List Reverse

The reverse method reverses the order of the entire list.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | L1 = ["One", "two", "three", "four", "five"]    #To print the list as it is, simply do:  print L1    #To print a reverse list, do:  for i in L1[::-1]:      print i    #OR    L = [0, 10, 20, 40]  L.reverse()    print L  [40, 20, 10, 0] |

List Sorting

The easiest way to sort a List is with the sorted(list) function.

That takes a list and returns anew list with those elements in sorted order.

The original list is not changed.

The sorted() function can be customized though optional arguments.

The sorted() optional argument reverse=True, e.g. sorted(list, reverse=True),

makes it sort backwards.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | #create a list with some numbers in it  numbers = [5, 1, 4, 3, 2, 6, 7, 9]    #prints the numbers sorted  print sorted(numbers)    #the original list of numbers are not changed  print numbers  my\_string = ['aa', 'BB', 'zz', 'CC', 'dd', "EE"]    #if no argument is used, it will use the default (case sensitive)  print sorted(my\_string)    #using the reverse argument, will print the list reversed  print sorted(strs, reverse=True)   ## ['zz', 'aa', 'CC', 'BB']    This will not return a value, it will modify the list  list.sort() |

List Split

Split each element in a list.

|  |  |
| --- | --- |
| 1  2  3  4  5 | mylist = ['one', 'two', 'three', 'four', 'five']  newlist = mylist.split(',')    print newlist  ['one', ' two', ' three', ' four', 'five'] |

List Indexing

Each item in the list has an assigned index value starting from 0.

Accessing elements in a list is called indexing.

|  |  |
| --- | --- |
| 1  2  3  4 | list    = ["first", "second", "third"]  list[0] == "first"  list[1] == "second"  list[2] == "third" |

List Slicing

Accessing parts of segments is called slicing.

Lists can be accessed just like strings by using the [ ] operators.

The key point to remember is that the :end value represents the first value that

is not in the selected slice.

So, the difference between end and start is the number of elements selected

(if step is 1, the default).

Let's create a list with some values in it

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | colors = ['yellow', 'red', 'blue', 'green', 'black']    print colors[0]  >>> yellow    print colors [1:]  >>> red, blue, green, black |

Let’s look at this example taken from [this](http://stackoverflow.com/questions/509211/the-python-slice-notation) stackoverflow post.

|  |  |
| --- | --- |
| 1  2  3  4 | a[start:end]            # items start through end-1  a[start:]               # items start through the rest of the array  a[:end]                 # items from the beginning through end-1  a[:]                    # a copy of the whole array |

There is also the step value, which can be used with any of the above

|  |  |
| --- | --- |
| 1 | a[start:end:step]       # start through not past end, by step |

The other feature is that start or end may be a negative number, which means it counts  
from the end of the array instead of the beginning.

|  |  |
| --- | --- |
| 1  2  3 | a[-1]               # last item in the array  a[-2:]              # last two items in the array  a[:-2]              # everything except the last two items |

List Loops

When using loops in programming, you sometimes need to store the results of the

loops.

One way to do that in Python is by using lists.

This short section will show how you can loop through a Python list and process

the list items.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #It can look something like this:  matching = []  for term in mylist:      do something    #For example, you can add an if statement in the loop, and add the item to the (empty) list<br>if it's matching.  matching = []    #creates an empty list using empty square brackets []  for term in mylist:      if test(term):          matching.append(term)    #If you already have items in a list, you can easily loop through them like this:  items = [ 1, 2, 3, 4, 5 ]  for i in items:      print i |

List Methods

Calls to list methods have the list they operate on appear before the method name.

Any other values the method needs to do its job is provided in the normal way as

an extra argument inside the round brackets.

s = ['h','e','l','l','o'] #create a list

s.append('d') #append to end of list

len(s) #number of items in list

s.sort() #sorting the list

s.reverse() #reversing the list

s.extend(['w','o']) #grow list

s.insert(1,2) #insert into list

s.remove('d') #remove first item in list with value e

s.pop() #remove last item in the list

s.pop(1) #remove indexed value from list

s.count('o') #search list and return number of instances found

s = range(0,10) #create a list over range

s = range(0,10,2) #same as above, with start index and increment

List Examples

Let's end this article, with showing some list examples:

First, create a list containing only numbers.

|  |  |
| --- | --- |
| 1  2  3  4  5 | list = [1,2,3,5,8,2,5.2]    #creates a list containing the values 1,2,3,5,8,2,5.2  i = 0  while i < len(list):     #The while loop will print each element in the list      print list[i]        #Each element is reached by the index (the letter in the square bracket)      i = i + 1            #Increase the variable i with 1 for every time the while loop runs. |

The next example will count the average value of the elements in the list.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | list = [1,2,3,5,8,2,5.2]  total = 0  i = 0  while i < len(list):      total = total + list[i]      i = i + 1    average = total / len(list)  print average |

Please don't forget to check out the other Cheat Sheets at the top of the page.