[List Comprehensions in Python](http://www.pythonforbeginners.com/lists/list-comprehensions-in-python/)***1***

21 Mar 2013   | [Lists](http://www.pythonforbeginners.com/category/basics/)

Tags: [Lists](http://www.pythonforbeginners.com/tag/lists/)

List Comprehensions

List comprehensions provide a concise way to create lists.

It consists of brackets containing an expression followed by a for clause, then

zero or more for or if clauses. The expressions can be anything, meaning you can

put in all kinds of objects in lists.

The result will be a new list resulting from evaluating the expression in the

context of the for and if clauses which follow it.

The list comprehension always returns a result list.

If you used to do it like this:

|  |  |
| --- | --- |
| 1  2  3  4 | new\_list = []  for i in old\_list:      if filter(i):          new\_list.append(expressions(i)) |

You can obtain the same thing using list comprehension:

|  |  |
| --- | --- |
| 1 | new\_list = [expression(i) for i in old\_list if filter(i)] |

Syntax

The list comprehension starts with a '**[' and ']'**, to help you remember that the

result is going to be a list.

The basic syntax is

|  |  |
| --- | --- |
| 1 | [ expression for item in list if conditional ] |

This is equivalent to:

|  |  |
| --- | --- |
| 1  2  3 | for item in list:      if conditional:          expression |

Let's break this down and see what it does

|  |  |
| --- | --- |
| 1 | new\_list = [expression(i) for i in old\_list if filter(i)] |

**new\_list**

The new list (result).

**expression(i)**

Expression is based on the variable used for each element in the old list.

**for i in old\_list**

The word for followed by the variable name to use, followed by the word in the

old list.

**if filter(i)**

Apply a filter with an If-statement.

This [blog](http://blog.cdleary.com/2010/04/learning-python-by-example-list-comprehensions/) shows an example of how to visually break down the list comprehension:

**new\_range = [i \* i for i in range(5) if i % 2 == 0]**

Which corresponds to:

**\*result\* = [\*transform\* \*iteration\* \*filter\* ]**

The \* operator is used to repeat. The filter part answers the question if the

item should be transformed.

Examples

Now when we know the syntax of list comprehensions, let's show some examples and

how you can use it.

Create a simple list

Let's start easy by creating a simple list.

|  |  |
| --- | --- |
| 1  2  3  4  5 | x = [i for i in range(10)]  print x    # This will give the output:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] |

That's how you can create a simple list.

Create a list using loops and list comprehension

For the next example, assume we want to create a list of squares.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | # You can either use loops:  squares = []    for x in range(10):      squares.append(x\*\*2)    print squares  [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]    # Or you can use list comprehensions to get the same result:  squares = [x\*\*2 for x in range(10)]    print squares  [0, 1, 4, 9, 16, 25, 36, 49, 64, 81] |

Just remember the syntax: [ expression for item in list if conditional ]

Multiplying parts of a list.

Multiply every part of a list by three and assign it to a new list.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | list1 = [3,4,5]    multiplied = [item\*3 for item in list1]    print multiplied  [9,12,15] |

Note how the item\*3 multiplies each piece by 3.

Show the first letter of each word

We will take the first letter of each word and make a list out of it.

|  |  |
| --- | --- |
| 1  2  3  4  5 | listOfWords = ["this","is","a","list","of","words"]    items = [ word[0] for word in listOfWords ]    print items |

The output should be: ['t', 'i', 'a', 'l', 'o', 'w']

Lower/Upper case converter

Let's show how easy you can convert lower case / upper case letters.

|  |  |
| --- | --- |
| 1  2  3  4  5 | >>> [x.lower() for x in ["A","B","C"]]  ['a', 'b', 'c']    >>> [x.upper() for x in ["a","b","c"]]  ['A', 'B', 'C'] |

Print numbers only from a given string

This example show how to extract all the numbers from a string.

|  |  |
| --- | --- |
| 1  2  3  4  5 | string = "Hello 12345 World"  numbers = [x for x in string if x.isdigit()]  print numbers    >> ['1', '2', '3', '4', '5'] |

Change x.isdigit() to x.isalpha() if you don't want any numbers.

Parsing a file using list comprehension

In this example, we can see how to get specific lines out from a text file.

Create a text file and put in some text in it.

this is line1

this is line2

this is line3

this is line4

this is line5

Save the file as test.txt

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # Then create the filter by using list comprehension:    fh = open("test.txt", "r")    result = [i for i in fh if "line3" in i]    print result |

Output: ['this is line3\n']

Using list comprehension in functions

Now, let's see how we can use list comprehension in functions.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # Create a function and name it double:  def double(x):    return x\*2    # If you now just print that function with a value in it, it should look like this:  >>> print double(10)  20 |

We can easily use list comprehension on that function.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | >>> [double(x) for x in range(10)]    print double  [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]    # You can put in conditions:    >>> [double(x) for x in range(10) if x%2==0]  [0, 4, 8, 12, 16]    # You can add more arguments:    >>> [x+y for x in [10,30,50] for y in [20,40,60]]  [30, 50, 70, 50, 70, 90, 70, 90, 110] |

See how you can put in conditions and add more arguments.

Sources

<http://docs.python.org/2/tutorial/datastructures.html>

<http://www.dalkescientific.com/writings/NBN/list_comps.html>

<http://effbot.org/zone/python-list.htm>

<http://blog.cdleary.com/2010/04/learning-python-by-example-list-comprehensions/>