List Comprehensions

John M. Morrison

September 29, 2015

Contents

1 List Comperehensions

List comprehensions provide a succint and convienent way to filter and transform lists. These operations produce a transformed copy of the original list. Let us begin with a simple example.

```
>>> x = [1,2,3,4,5]
>>> y = [k*k for k in x]
>>> y
[1, 4, 9, 16, 25]
>>> x
[1, 2, 3, 4, 5]
>>>
```

This operation took every item in the list `x` and squared it. You can use this technique to "butter" a function over a list. Suppose we have a function `f` and a list `x` and we want to call the function on every element on the list and return in in an new list. Then all you need do is this.

```
[f(k) for k in x]
```

You can also filter items in a list using this technique.

```
>>> names = ["smith", "jones", "sims", "boyarsky", "teague", "miller", "doyle"]
>>> hasAnE = [k for k in names if "e" in k]
>>> hasAnE
['jones', 'teague', 'miller', 'doyle']
>>>
```

This operation created a new list with the names containing the letter `e`. The general form of this construct is as follows.

```
[f(k) if k predicate(x)]
```

The item `predicate` is a boolean-valued experesion involving `x`. The items that pass the filter are those for which `predicate(x)` evaluates to `True`.