List Comprehensions and Generator Expressions

Ian Ward

http://excess.org/

Ottawa Python Authors Group

February 28

For Beginners

(this talk)

For Everyone Else

Problem 1:

```
>>> x = ?
>>> [x for x in x for x in x]
?
```

Problem 2:

```
>>> y = []
>>> y.append(?)
>>> y
[]
```

Why Listcomps?

```
def get_foos(stuff):
    """
    return a list of the foos in stuff
    """
    out = []
    for item in stuff:
        out.append(item.foo)
    return out

foos = get_foos(my_stuff)
```

List Comprehension

```
def get_foos(stuff):
    """
    return a list of the foos in stuff
    """
    out = []
    for item in stuff:
        out.append(item.foo)
    return out

foos = get_foos(my_stuff)
```

```
foos = [item.foo for item in my_stuff]
```

Functional-style

```
foos = map(lambda item: item.foo, my_stuff)
```

```
foos = [item.foo for item in my_stuff]
```

Filtering

```
small = filter(lambda num: num < 5, values)
```

```
small = [num for num in values if num < 5]
```

Not Always Best

```
v2 = [num for num in values]
```

```
v2 = list(values)
```

Not Always Best

```
trunc = [int(num) for num in values]
```

```
trunc = map(int, values)
```

Deconstructing Listcomps

```
out = [A for B in C]
```

```
out = []
for B in C:
    out.append(A)
```

Deconstructing Listcomps

```
out = [A for B in C if D]
```

```
out = []
for B in C:
    if D:
        out.append(A)
```

Deconstructing Listcomps

```
out = [A for B in C if D for E in F]
```

```
out = []
for B in C:
    if D:
        for E in F:
            out.append(A)
```

Deconstructing Listcomps

```
out = [A for B in C if D for E in F if G]
```

```
out = []
for B in C:
    if D:
        for E in F:
            if G:
                out.append(A)
```

Examples

```
out = [book.title
       for author in authors if author.deceased
       for book in author.books if book.sold > 1e6]
```

Examples

```
out = [(author, book)
       for author in authors
       for book in author.books]
```

Examples

```
author_lookup = dict([(book.isbn, author)
                      for author in authors
                      for book in author.books])
```

Dict Comprehension

```
author_lookup = {book.isbn: author
                 for author in authors
                 for book in author.books}
```

Python 2.7+

Generator Expression

```
author_lookup = dict((book.isbn, author)
                     for author in authors
                     for book in author.books)
```

Python 2.4+

Generator Expression

```
num_books = sum(len(author.books)
                for author in authors)
```

Generator Expression

```
deceased = sum(1 for author in authors if author.deceased)
```

Generator Expression

```
have_x = any(book.title.startswith('X')
             for author in authors
             for book in author.books)
```

Generators

http://bit.ly/itergen1

(redirects to http://excess.org/article/2013/02/itergen1/)

End

(of serious presentation)

Stupid Listcomp Tricks

```
>>> [x for x in range(5) for y in (1,2)]
[0, 0, 1, 1, 2, 2, 3, 3, 4, 4]
```

Stupid Listcomp Tricks

```
>>> class Baz(object):
...     pass
...
>>> b = Baz()
>>> [1 for b.foo in ['bar']]
[1]
>>> b.foo
'bar'
```

Stupid Listcomp Tricks

```
>>> x = ?
>>> [x for x in x for x in x]
?
```

Stupid Listcomp Tricks

```
>>> x = [['hello', 'world'], ['nice', 'to', 'meet'], ['you']]
>>> [x for x in x for x in x]
['hello', 'world', 'nice', 'to', 'meet', 'you']
```

```
>>> [inner for outer in x for inner in outer]
```

Stupid Listcomp Tricks

```
>>> y = []
>>> y.append(?)
>>> y
[]
```

Stupid Listcomp Tricks

```
>>> y = []
>>> y.append([1 for y in [[]]])
>>> y
[]
```

(fixed in Python 3)