List Comprehensions

- Problem: given a list of prices, generate a new list that has a 20% discount to each.
- Formally: input: list of old prices; output: list of new prices
- Can solve it recursively.
- Or can use List comprehensions.
- Syntax for list comprehension:

```
[x*0.8 for x in old_price]
>>> price = [10, 20, 30, 100]
>>> [x*0.8 for x in price]
[8.0, 16.0, 24.0, 80.0]
```

List Comprehensions

What is going on here?

```
>>> [ 2*x for x in [0,1,2,3,4,5] ]
[0, 2, 4, 6, 8, 10]

>>> [ y**2 for y in range(6) ]
[0, 1, 4, 9, 16, 25]

>>> [ c == 'a' for c in 'go away!' ]
[False, False, False, True, False,
True, False, False]

>>> [x for x in 'go away!' if x == 'a']
['a', 'a']
```

List Comprehensions

Any operation you want to apply to each element of the list

name that takes on the value of each element *in turn*

any name is OK!

the list (or string)

```
>>> [ 2*x for x in [0,1,2,3,4,5] ]
[0, 2, 4, 6, 8, 10]
>>> [ y**2 for y in range(6) ]
[0, 1, 4, 9, 16, 25]
>>> [ c == 'a' for c in 'go away!' ]
[False, False, False, True, False, True,
False, False]
>>> [x for x in 'go away!' if x == 'a']
['a', 'a']
```

Raw recursion vs. list comprehensions

`my_len(t)`

```
def my_len(t):
    if t == []:
        return 0
    else:
        return 1 + my_len(t[1:])
```

```
def my_len(t):
      list_comp = [1 for x in t]
      return sum(list_comp)
```

```
''' or simply '''
def my_len(t):
      return sum([1 for x in t])
```

Raw recursion vs. list comprehensions

`count_vows(s)` # of vowels

```
def count_vows(s):
    if len(s) == 0:
        return 0
    else:
        if s[0] not in 'aeiou':
            return count_vows(s[1:])
        else:
            return 1 + count_vows(s[1:])
```

```
def count_vows(s):
      return sum([1 for x in s if x in 'aeiou'])
```

List comprehension with filtering

```
def only_evens(t):
   return [x for x in t if is_even(x)]
```

$\underbrace{\texttt{x for x in t}}_{\text{list comprehension}}\ \underbrace{\texttt{if is\_even(x)}}_{\text{with filter}}$

```
>>>only_evens([13, 2, 5, 6, 9])
[2, 6]
```

More examples of comprehensions

Generate all powers of 2 from 0 to 10

```
my_list = [2** i for i in range (10) ] # [1 ,2 ,4 ,8 ,16 ,...2^9]
```

Given a list, get a list of square roots of its elements

```
from math import sqrt
my_list = [sqrt (x) for x in otherlist ] # produced a squared list
```

Interesting. Generate a list of odd numbers from 0 to 10

```
list = [x for x in range (10) if x % 2 == 1]  # [1, 3, 5, 7, 9]
```