What is a list comprehension?

• Concise way to create a list from another list
• Syntax:
  • \( L_2 = \{\text{expression}(i) \text{ for } i \text{ in } L_1 \text{ if condition}(i)\} \)

```
cubes = []
for i in range(10):
    cubes.append(i**3)
```

```
cubes = \{i**3 \text{ for } i \text{ in range(10)}\}
```

Equivalent
Example using files

The existing list in a comprehension can be ad hoc:

```python
path = my_dir
files = [path + '/' + i for i in os.listdir(path)]
for f in files:
    fn = open(f, 'r')
    do something with fn
    fn.close()
```

What does this code do?

Your turn:

Use a list comprehension to create a list of Fahrenheit temperatures from a list of Centigrade temperatures.

```python
dF = [1.8 * c + 32 for c in dC]
```

What does this comprehension do:

```python
dF = [1.8 * c + 32 for c in [randint(0, 100) for i in range(10)]]
```
Filtering

We can choose which elements in the existing list are used to create the new list:

```python
roots = [math.sqrt(i) for i in nums if i > 0]
```

avoids math domain error (sqrt of negative number)

Note: `len(roots) <= len(nums)`

Your turn again:

Use a list comprehension to create a list of the odd values in a list of data.

```python
odds = [i for i in data if i % 2 == 1]
```
Multiple Lists

You can use multiple existing lists to create the new list:

\[
pairs = [(x, y) \text{ for } x \text{ in } \text{xcoords} \text{ for } y \text{ in } \text{ycoords}]\]

What are the elements of \(pairs\)?
- each is an \((x, y)\) pair in the cross-product of \(\text{xcoords}\) and \(\text{ycoords}\)
- this works even if the two input lists have different lengths

What if you want elements paired by position?

\[
pairs = [(\text{xcoords}[i], \text{ycoords}[i]) \text{ for } i \text{ in } \text{range}(\text{len}(\text{xcoords}))]\]

Can use min if the input lists have different lengths.

Using a function

Let \(\text{coords}\) be a list containing 2-element lists of GPS coordinates:

\[
[[\text{lat1}, \text{lon1}], [\text{lat2}, \text{lon2}]],[[\text{lat3}, \text{lon3}], [\text{lat4}, \text{lon4}]]...\]

Create a list of distances between the pair of cities in each sublist.

\[
def \text{distance}(\text{city1}, \text{city2}): \text{ Function definition} \\
\text{# compute and return distance} \\
\text{dists} = [\text{distance}(x, y) \text{ for } x, y \text{ in } \text{coords}]\]
One more turn for you...

Staying with the setup from the previous slide, create a list of the pairs of cities that are closer than threshold $d$:

$$\text{close} = \{[x, y] \text{ for } x, y \text{ in } \text{coords if } \text{distance}(x, y) < d\}$$