List comprehensions

UW CSE 140
Winter 2014
Ways to express a list

1. Explicitly write the whole thing:
   squares = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

2. Write a loop to create it:
   squares = []
   for i in range(11):
       squares.append(i*i)

3. Write a list comprehension:
   squares = [i*i for i in range(11)]

   A list comprehension is a concise description of a list
   A list comprehension is shorthand for a loop
Let $I$ be the integers

- $\{ x : x \in I \text{ and } x = x^2 \}$ is the set $\{ 0, 1 \}$
- $\{ x : x \in I \text{ and } x > 0 \}$ is the set of all positive integers
- $\{ x^2 : x \in I \text{ and } 0 \leq x < 10 \text{ and prime}(x) \}$

**Python notation:**

- $\{ x^2 \text{ for } x \text{ in range}(10) \text{ if prime}(x) \}$
Two ways to convert Centigrade to Fahrenheit

ctemps = [17.1, 22.3, 18.4, 19.1]

With a loop:

ftemps = []
for c in ctemps:
    f = celsius_to_farenheit(c)
    ftemps.append(f)

With a list comprehension:

ftemps = [celsius_to_farenheit(c) for c in ctemps]

The comprehension is usually shorter, more readable, and more efficient
Syntax of a comprehension

```
[(x,y) for x in org1 for y in org2 if sim(x,y) > threshold]
```
Semantics of a comprehension

\[
[(x,y) \text{ for } x \text{ in org1 for } y \text{ in org2 if } \text{sim}(x,y) > \text{threshold}]
\]

result = []
for x in org1:
    for y in org2:
        if \text{sim}(x,y) > \text{threshold}:
            result.append( (x,y) )

... use result ...
Types of comprehension

List

\[ \text{[ i*2 for i in range(3) ]} \]

Set

\{ i*2 for i in range(3) \}

Dictionary

d = \{ \text{key: value for item in sequence ...} \}

\{ i: i*2 for i in range(3) \}
Preparing names for alphabetization

Goal: convert “firstname lastname” to “lastname, firstname”


With a loop:
result = []
for name in names:
    split_name = name.split(" ")
    last_name_first = split_name[1] + ", " + split_name[0]
    result.append(last_name_first)

With a list comprehension:
split_names = [name.split(" ") for name in names]
last_names_first = [sn[1] + ", " + sn[0] for sn in split_names]
# Bonus: last_names = [split_name[1] for split_name in split_names]

Another idea: write a function, then use the function in a comprehension
Cubes of the first 10 natural numbers

Goal:
Produce: [0, 1, 8, 27, 64, 125, 216, 343, 512, 729]

With a loop:
cubes = []
for x in range(10):
    cubes.append(x**3)

With a list comprehension:
cubes = [x**3 for x in range(10)]
Powers of 2, $2^0$ through $2^{10}$

Goal: [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]

[2**i for i in range(11)]
Even elements of a list

Goal: Given an input list \texttt{nums}, produce a list of the even numbers in \texttt{nums}

\texttt{nums} = [3, 1, 4, 1, 5, 9, 2, 6, 5]
\Rightarrow [4, 2, 6]

\texttt{[num for num in nums if num \% 2 == 0]}
Gene sequence similarity

Goal: Find all similar pairs of genome sequences (one sequence from org1, one from org2)

```
org1 = ["ACGTTTCA", "AGGCCTTA", "AAAACCTG"]
org2 = ["AGCTTTGA", "GCCGGAAT", "GCTACTGA"]
```

“Similar” means: similarity(seq1, seq2) > threshold

```
def similarity(sequence1, sequence2):
    """Return a number representing the similarity score between the two arguments"""
    ...
```

```
[(s1,s2) for s1 in org1 for s2 in org2 if similarity(s1,s2) > threshold]
```
All above-average 2-die rolls

Result list should be a list of 2-tuples:

\[
[(2, 6), (3, 5), (3, 6), (4, 4), (4, 5), (4, 6), (5, 3), (5, 4),
(5, 5), (5, 6), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6)]
\]

\[
[(r1, r2) \text{ for } r1 \text{ in } [1,2,3,4,5,6]
\text{ for } r2 \text{ in } [1,2,3,4,5,6]
\text{ if } r1 + r2 > 7]
\]

OR

\[
[(r1, r2) \text{ for } r1 \text{ in range}(1, 7)
\text{ for } r2 \text{ in range}(8-r1, 7)]
\]
Get more practice

• Use comprehensions where appropriate
• Convert loops to comprehensions
• Convert comprehensions to loops