List comprehensions
(and other shortcuts)

BBM 101 - Introduction to Programming I
Hacettepe University
Fall 2016

Fuat Akal, Aykut Erdem, Erkut Erdem
Three Ways to Define a List

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

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

• 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
Two ways to convert Centigrade to Fahrenheit

ctemps = [17.1, 22.3, 18.4, 19.1]

With a loop:

```python
temps = []
for c in ctemps:
    f = celsius_to_farenheit(c)
    temps.append(f)
```

With a list comprehension:

```python
temps = [celsius_to_farenheit(c) for c in ctemps]
```

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

\[
[(x,y) \text{ for } x \text{ in } \text{seq1} \text{ for } y \text{ in } \text{seq2} \text{ if } \text{sim}(x,y) > \text{threshold}]\]

- **expression**
- **for clause (required)** assigns value to the variable \( x \)
- **zero or more additional for clauses**
- **zero or more if clauses**
Semantics of a comprehension

\[
[(x,y) \text{ for } x \text{ in } \text{seq1} \text{ for } y \text{ in } \text{seq2} \text{ if } \text{sim}(x,y) > \text{threshold}]
\]

result = []
for x in seq1:
    for y in seq2:
        if sim(x,y) > threshold:
            result.append( (x,y) )

... use result ...
Types of comprehensions

List

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

Set

\{ i \times 2 \text{ for } i \text{ in range(3)} \}

Dictionary

\{ key: value \text{ for } item \text{ in sequence } \ldots \}
\{ i: i \times 2 \text{ for } i \text{ in range(3)} \}
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: $[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]$

$[2^{**i} \text{ for } i \text{ in } \text{range}(11)]$
Even elements of a list

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

$\text{nums} = [3, 1, 4, 1, 5, 9, 2, 6, 5]$

$\Rightarrow [4, 2, 6]$

$[\text{num for num in nums if num % 2 == 0}]$
**Goal**: A list of all possible dice rolls.

**With a loop**: 
rolls = []
for r1 in range(1,7):
    for r2 in range(1,7):
        rolls.append((r1,r2))

**With a list comprehension**: 
rolls = [ (r1,r2) for r1 in range(1,7) for r2 in range(1,7)]
All above-average 2-die rolls

**Goal:** Result list should be a list of 2-tuples:

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

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

OR

\[\{(r1, r2) \text{ for } r1 \text{ in range(1, 7)} \]
for r2 in range(8-r1, 7)\]
All above-average 2-die rolls

**Goal:** Result list should be a list of 2-tuples:

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

\[
[(r_1, r_2) \text{ for } r_1 \text{ in } [1, 2, 3, 4, 5, 6]
\text{ for } r_2 \text{ in } [1, 2, 3, 4, 5, 6]
\text{ if } r_1 + r_2 > 7]
\]

**Remove Duplicates: Use Set Comprehensions**

\[
\{ r_1 + r_2 \text{ for } r_1 \text{ in range}(1, 7)
\text{ for } r_2 \text{ in range}(1, 7)
\text{ if } r_1 + r_2 > 7}\]

\[\Rightarrow \text{set}([ (6, 4), (5, 4), (2, 6), (4, 6), (6, 6), (4, 5), (4, 4), (5, 5), (6, 3), (5, 6), (6, 2), (3, 6),
(5, 3), (6, 5), (3, 5)])}\]
Making a Matrix

**Goal:** A matrix were each element is the sum of it's row and column.

**With a loop:**

```python
matrix = []
for i in range(5):
    row = []
    for j in range(5):
        row.append(i+j)
    matrix.append(row)
```

**With a list comprehension:**

```python
matrix = [[i+j for j in range(5)] for i in range(5)]
```
More examples
function $4x^2 - 4$

With a loop:

```python
num_list = []
for i in range(-10,11):
    num_list.append(4*i**2 - 4)
```

With a list comprehension:

```python
num_list = [4*i**2 - 4 for i in range(-10,11)]
```
Normalize a list

```python
num_list = [6, 4, 2, 8, 9, 10, 3, 2, 1, 3]
total = float(sum(num_list))

With a loop:
for i in range(len(num_list)):
    num_list[i] = num_list[i]/float(total)

With a list comprehension:
num_list = [i/total for i in num_list]
```
Matrix of zeros

With a loop:

```python
matrix = []
for i in range(10):
    matrix.append([0]*10)
```

With a list comprehension:

```python
matrix = [[0]*10 for i in range(10)]
```
Multiplication table

With a loop:

```python
table = []
for r in range(1,10):
    row = []
    for c in range(1,10):
        row.append(r*c)
    table.append(row)
```

With a list comprehension:

```python
table = [[r*c for c in range(1,10)] for r in range(1,10)]
```
Mapping of powers of ten

With a loop:

```python
powers = {}
for i in range(-6,7,3):
    powers[i] = 10**i
```

With a list comprehension:

```python
powers = {i:10**i for i in range(-6,7,3)}
```
Dictionary mapping integers to multiples under 100

With a loop:

```python
for n in range(1,11):
    multiples_list = []
    for i in range(1,101):
        if i%n == 0:
            multiples_list.append(i)
    multiples[n] = multiples_list
```

With a dictionary comprehension:

```python
multiples = {n:[i for i in range(1,101) if i%n == 0] for n in range(1,11) }
```
A word of caution

List comprehensions are great, but they can get confusing. Error on the side of readability.

```python
nums = [n for n in range(100) if 
    sum([int(j) for j in str(n)]) % 7 == 0]
```

```python
nums = []
for n in range(100):
    digit_sum = sum([int(j) for j in str(n)])
    if digit_sum % 7 == 0:
        nums.append(n)
```
A word of caution

List comprehensions are great, but they can get confusing. Error on the side of readability.

```python
nums = [n for n in range(100) if sum([int(j) for j in str(n)]) % 7 == 0]

def sum_digits(n):
    digit_list = [int(i) for i in str(n)]
    return sum(digit_list)

nums = [n for n in range(100) if sum_digits(n) % 7 == 0]
```
More shortcuts!
Enumerate a list

```python
the_list = [10**i for i in range(10)]
for i in range(len(the_list)):
    print(str(i) + ': ' + str(the_list[i]))
```

Or:

```python
for index, value in enumerate(the_list):
    print(str(index) + ': ' + str(value))
```

Like `dict.items()`
Enumerate a list

**Goal**: add each element’s index itself

```python
the_list = range(10)
new_list = []
for i,v in enumerate(the_list):
    new_list.append(i+v)
```

**With a list comprehension**:

```python
the_list = range(10)
new_list = [ i+v for i,v in enumerate(the_list) ]
```
Ternary Assignment

A common pattern in python

```python
if x > threshold:
    flag = True
else:
    flag = False
Or
flag = False
if x > threshold:
    flag = True
```
Ternary Assignment

A common pattern in python

```python
if x > threshold:
    flag = True
else:
    flag = False

flag = True if x > threshold else False
```

Ternary Expression
Three elements
Ternary Assignment

\[ \text{flag} = \text{True if } x > \text{threshold else False} \]

- Only works for single expressions as results.
- Only works for if and else (no elif)
Ternary Assignment

Goal: A list of 'odd' or 'even' if that index is odd or even.

```python
the_list = []
for i in range(16):
    if i%2 == 0:
        the_list.append('even')
    else:
        the_list.append('odd')
```

or

```python
the_list = []
for i in range(16):
    the_list.append('even' if i%2 == 0 else 'odd')
```
Ternary Assignment

Goal: A list of 'odd' or 'even' if that index is odd or even.

```python
the_list = []
for i in range(16):
    if i%2 == 0:
        the_list.append('even')
    else:
        the_list.append('odd')

or

the_list = ['even' if i%2 == 0 else 'odd' for i in range(16)]
```
Get more practice

List Comprehensions:

\[
[(x,y) \text{ for } x \text{ in } \text{seq1} \text{ for } y \text{ in } \text{seq2} \text{ if } \text{sim}(x,y) > \text{threshold}]
\]

Enumerate:

\[
\text{for index, value in enumerate(seq):}
\]
\[
\ldots
\]

Ternary If Statement:

\[
\text{flag = True if } x > \text{threshold} \text{ else False}
\]