Lecture 11: Iterating over Sequences

Announcements

Prelim 1
The exam is in one week. It covers material in lecture up through today and in lab up through this week’s lab.
Processing sequences: The map Function

General form: \( \text{map}(\langle \text{function} \rangle, \langle \text{sequence} \rangle) \)

if \( x \) is a sequence of \( n \) items and 
f is a function with one parameter:

\[
\text{map}(f, x) \rightarrow [f(x[0]), f(x[1]), \ldots, f(x[n-1])] 
\]

calls the function once for each list item

if \( x \) is a sequence of \( n \) items and 
m is a method with no parameters:

\[
\text{map}(m, x) \rightarrow [x[0].m(), x[1].m(), \ldots, x[n-1].m()] 
\]

which could be a list or a string
Mapping a function over a list

x = [1.2, 3.9, -4.8, 0.1]
y = map(round, x)

>>> y
[1.0, 4.0, -5.0, 0.0]
Mapping a function over a list

```python
x = ['a', 'list', 'of', 'things']
y = map(len, x)
```

>>> y
[1, 4, 2, 6]
Mapping a method over a list

x = ['Lee', 'Python', 'CS1110', 'A+']

y = map(str.lower, x)

>>> y
['lee', 'python', 'cs1110', 'a+']

the method lower in the class str: a method with no parameters*

*In this case you could also use string.lower instead, which is the function lower in the module string.
Mapping a function over a string

```python
def is_vowel(c):
    return c.lower() in 'aeiou'
y = map(is_vowel, 'Anacondas')
```

```
>>> map(int, y)
[1, 0, 1, 0, 1, 0, 0, 1, 0]
```
Which causes an error?

A: map(round, [1,2,3])
B: map(str, [1,2,3])
C: map(int, ['1','2','3'])
D: map(sum, [[1,2],[3,4]])
E: All are OK
Processing lists: The **for** Statement

General form:

```plaintext
for ⟨variable⟩ in ⟨sequence⟩:
  ⟨statements⟩
```

```python
for a in x:
  print 3 * a
```

- `executes the body once for each list item`
- `print 3 * x[0]`
- `print 3 * x[1]`
- `print 3 * x[2]`
- ...`
- `print 3 * x[n–1]`

- `when the body is executed, the value of a is the current list item`
Processing sequences: The for Statement

General form:
for \langle variable \rangle in \langle sequence \rangle:
\langle statements \rangle

The loop body is executed once for each list item.

(assume \(x\) refers to a list of \(n\) items)

for \(a\) in \(x\):
    print \(3 \times a\)

\(a = x[0]\)
print \(3 \times a\)
\(a = x[1]\)
print \(3 \times a\)
\(a = x[2]\)
print \(3 \times a\)
...
\(a = x[n-1]\)
print \(3 \times a\)

Before each iteration, the value of the next list item is assigned to the loop variable.
Iterating over a list with for

```python
x = [1.2, 3.9, -4.8, 0.1]
for a in x:
    print round(a)
```

```plaintext
1.0
4.0
-5.0
0.0
```
Iterating over a list with for

```python
x = ['a', 'list', 'of', 'things']
for s in x:
    print s + ': ' + str(len(s))
```

```
s = 'a'
print s + ': ' + str(len(s))
s = 'list'
print s + ': ' + str(len(s))
s = 'of'
print s + ': ' + str(len(s))
s = 'things'
print s + ': ' + str(len(s))
```

```
a: 1
list: 4
of: 2
things: 6
```
Iterating over a string with for

```python
s = 'Python'
for c in s:
    print 'Gimme a ' + c + '!
```

```python
c = 'P'
print 'Gimme a ' + c + '!
c = ' + c + '!
c = ' + c + '!
c = ' + c + '!
c = ' + c + '!
c = ' + c + '!
c = ' + c + '!
c = ' + c + '!
c = ' + c + '!
```

Gimme a P!
Gimme a y!
Gimme a t!
Gimme a h!
...
Extended example: spell checker

- Goal: a simple spell checker for plain text files
- Plan:
  - put all the words in a dictionary into a list
  - check each word in the file to see if it’s in the list
  - if it’s not in the list, complain that it’s misspelled.
- We’ll play more word games in Lab 6…