Day 3: Collections

Suggested reading: *Learning Python* (3rd Ed.)
Chapter 8: Lists and Dictionaries
Chapter 9: Tuples, Files, and Everything Else
Chapter 13: *while* and *for* Loops
Turn In Homework
Homework Review

*Will not be posted online*
Write code. At least a little. Every day. Play around!
Single-Value Objects

- So far: \texttt{int}, \texttt{float}, \texttt{str}, \texttt{bool}
- Objects of these types hold exactly one value

\[
\begin{array}{c}
\text{x} \\
\downarrow \\
\begin{array}{c}
\text{int} \\
42
\end{array}
\end{array}
\]
How can we have a collection of (related) values?
Collection Examples

- All even numbers from 40–100
- Outdoor temperatures by date/time
- List of data files to read
- Weights associated with test items
- Frequency counts of a set of tokens
- Set of all observations from an instrument
Lists
List

- Ordered
- Variable length
- Arbitrary objects and types
- Access via integer position ("index")
- Mutable

```
|   | list
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>int 42</td>
</tr>
<tr>
<td>1</td>
<td>float 3.141</td>
</tr>
<tr>
<td>2</td>
<td>str Tim</td>
</tr>
<tr>
<td>3</td>
<td>str Hello</td>
</tr>
</tbody>
</table>
```

X
Creating a List

![image]

```
names = ['Tim', 'Scot', 'Alan']
empty = []
mixed = [42, 3.14159, 'hello', True]
long = [925, 161, 164, 529, 168, 208, 896, 531, 747, 932]
some_ints = range(2, 100, 2)
```
Using a List

```
courses = ['CS 302', 'CS 367', 'CS 368']
print courses
courses[0] = 'CS 302 - Intro to Progr.'
courses[1] += ' - Data Structures'
print 'What is %s?' % (courses[2])
```
# Other List Operations

```python
x = [42, 't', 1.3]
```

<table>
<thead>
<tr>
<th>Operation</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td><code>len(x)</code></td>
<td>2</td>
</tr>
<tr>
<td>concatenate</td>
<td><code>[1, 2] + x</code></td>
<td><code>[1, 2, 42, 't', 1.3]</code></td>
</tr>
<tr>
<td>membership</td>
<td><code>42 in x</code></td>
<td><code>True</code></td>
</tr>
<tr>
<td>slice</td>
<td><code>x[0:2]</code></td>
<td><code>[42, 't']</code></td>
</tr>
<tr>
<td>append</td>
<td><code>x.append(3)</code></td>
<td><code>x: [42, 't', 1.3, 3]</code></td>
</tr>
<tr>
<td>extend</td>
<td><code>x += [3, 1]</code></td>
<td><code>x: [42, 't', 1.3, 3, 1]</code></td>
</tr>
<tr>
<td>insert</td>
<td><code>x.insert(1, 'a')</code></td>
<td><code>x: [42, 'a', 't', 1.3]</code></td>
</tr>
<tr>
<td>delete</td>
<td><code>del x[1]</code></td>
<td><code>x: [42, 1.3]</code></td>
</tr>
<tr>
<td>remove</td>
<td><code>x.pop(1)</code></td>
<td><code>'t'</code></td>
</tr>
</tbody>
</table>
```

`x: [42, 1.3]`
List Bounds

- valid index: `int` from 0 to `(length - 1)`
- lists can grow and shrink (`append`, `insert`, …)
- limited only by memory
- going out of bounds is run-time error:

```python
>>> x = ['a', 'b', 'c']
>>> x[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```
Perplexing Python
Mutable vs. Immutable

- **Mutable** types allow changes to objects in memory
  - Examples: `list`, `dict`

- **Immutable** types do not
  - Examples: `int`, `float`, `str`, `bool`

```python
>>> x = 42
>>> y = x
>>> x += 1
>>> print x
43
>>> print y  # ??
```
```python
>>> x = []
>>> y = x
>>> x += [1]
>>> print x
[1]
>>> print y  # ??
```
Back to Collections
Tuples

\((item0, item1, \ldots, item-2, item-1)\)

tuple\([n]\)

- Immutable lists
- List operators work, methods do not

\[
t = (1, 2)
t += (3, 4) \quad \# \text{wait, what?!?}
\]

\[
\text{if } 3 \text{ in } t:\n\text{print } t[1:3]
\]
Dictionaries
Dictionary

- Unordered
- Variable length
- Arbitrary objects and types
- Access via arbitrary, unique immutable object ("key")
- Mutable

```
<table>
<thead>
<tr>
<th></th>
<th>'en'</th>
<th>'fr'</th>
<th>'de'</th>
<th>'es'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'en'</td>
<td>'fr'</td>
<td>'de'</td>
<td>'es'</td>
<td></td>
</tr>
<tr>
<td>'English'</td>
<td>'French'</td>
<td>'German'</td>
<td>'Spanish'</td>
<td></td>
</tr>
</tbody>
</table>
```

-\( x \rightarrow \text{dict} \)

```
<table>
<thead>
<tr>
<th></th>
<th>'English'</th>
<th>'French'</th>
<th>'German'</th>
<th>'Spanish'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'str'</td>
<td>'str'</td>
<td>'str'</td>
<td>'str'</td>
<td></td>
</tr>
<tr>
<td>'English'</td>
<td>'French'</td>
<td>'German'</td>
<td>'Spanish'</td>
<td></td>
</tr>
</tbody>
</table>
```
Creating a Dictionary

```
languages = {'en': 'English',
             'fr': 'French',
             'de': 'German',
             'es': 'Spanish'}

# print languages
# {'fr': 'French', 'en': 'English', ...}

readings = {'TI3a': 43.23, 'TF3a': 47.09,
             'TI3b': 38.22, 'TF3b': 42.96,
             'TI4a': 42.98, 'TF4a': 47.00,
             ...
```
Using a Dictionary

dictionary[key]

```python
languages = {'en': 'English',...}
print languages['de']
languages['es'] = 'Espanol'
languages['de'] = 'Deutsche'
languages['de'] += ' (German)'
print '"es' => '%s'" % (languages['es'])

len(languages)  # 4
languages['ja'] = 'Japanese'  # new entry
len(languages)  # 5
```
## Other Dictionary Operations

```python
c = {'a': 5, 'the': 7, 'an': 2}
```

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td><code>len(c)</code></td>
<td>3</td>
</tr>
<tr>
<td>membership</td>
<td><code>'an' in c</code></td>
<td>True</td>
</tr>
<tr>
<td>safe lookup</td>
<td><code>c.get('an')</code></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><code>c.get('no')</code></td>
<td>None</td>
</tr>
<tr>
<td>keys</td>
<td><code>c.keys()</code></td>
<td>['the', 'an', 'a']</td>
</tr>
<tr>
<td>items</td>
<td><code>c.items()</code></td>
<td>[('the', 7), ('an', 2), ...]</td>
</tr>
<tr>
<td>delete</td>
<td><code>del c['the']</code></td>
<td>c: {'an': 2, 'a': 5}</td>
</tr>
<tr>
<td>remove</td>
<td><code>c.pop('the')</code></td>
<td>7 c: {'an': 2, 'a': 5}</td>
</tr>
</tbody>
</table>
Sets

• models mathematical concept of a set
• unordered, variable-length, mutable collection
• somewhat between a list and a dictionary

```python
a = set([1, 2, 3])
b = set([2, 3, 4])

1 in a  # True
1 in b  # False

a & b   # set([2, 3])
a | b    # set([1, 2, 3, 4])
a - b   # set([1])
```
Collections and Loops
Sequences and Loops

```python
for item in seq:    # list, tuple, set, str
    print item

sum = 0
for n in xrange(1, 11):
    sum += n
print sum

s = 'Hello, world!'
print 'Char   ASCII'
print '----   -----'
for char in s:
    print '%4s   %5s' % (char, ord(char))
```
Dictionaries and Loops

```python
for key in dict.keys():
    print '%s => %s' % (key, dict[key])

for pair in dict.items():
    print '%s => %s' % pair

for key, value in dict.items():
    print '%s => %s' % (key, value)

valid_tests = []
for key, value in readings.items():
    if value > 0:
        valid_tests.append(key)
```
Phew!
Other Scripting Languages

• All have arrays and associative arrays

• Check for different or additional:
  – **Terminology** (list, array; hash, map, dictionary, …)
  – **Syntax** ([ ] vs. {}, \texttt{len(array)} vs. \texttt{array.length})
  – **Operations** (sort, unique elements, flatten, shuffle)
  – **Collections** (e.g., set)
Homework

• Implement a simple data analysis tool
  – Collect data observations
  – Display the items and their count, sum, mean, …

• BE SURE TO LABEL YOUR PRINTOUT!!!

#!/usr/bin/env python

"""Homework for CS 368-4 (2012 Fall)
Assigned on Day 03, 2012-10-29
Written by <Your Name>
"""