Sets and Dictionaries

Introduction
The world is *not* made of lists and arrays
The world is *not* made of lists and arrays

Mathematicians *uses* *sets* far more often
The world is *not* made of lists and arrays
Mathematicians uses *sets* far more often

*An unordered collection of distinct items*
The world is *not* made of lists and arrays
Mathematicians use *sets* far more often
An *unordered collection* of *distinct* items
Collection: contains zero or more items
The world is *not* made of lists and arrays.
Mathematicians use *sets* far more often.
An *unordered collection* of *distinct* items.
Collection: contains zero or more items.
Distinct: no item appears more than once.
The world is *not* made of lists and arrays
Mathematicians uses *sets* far more often
An *unordered collection* of *distinct* items
Collection: contains zero or more items
Distinct: no item appears more than once
Unordered: no such thing as "first" or "last"
The world is *not* made of lists and arrays.
Mathematicians use *sets* far more often.
An *unordered collection* of *distinct* items.
Collection: contains zero or more items.
Distinct: no item appears more than once.
Unordered: no such thing as "first" or "last".
- This is the part people tend to trip over most.
Sets were added to Python after most of the language was already defined.
Sets were added to Python after most of the language was already defined

- But at least they're there...
Sets were added to Python after most of the language was already defined
- But at least they're there...

**Python 2.6**

```python
primes = set([2, 3, 5])
```
Sets were added to Python after most of the language was already defined
- But at least they're there...

<table>
<thead>
<tr>
<th>Python 2.6</th>
<th>Python 2.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>primes = set([2, 3, 5])</td>
<td>primes = {2, 3, 5}</td>
</tr>
</tbody>
</table>
Sets were added to Python after most of the language was already defined
- But at least they're there...

<table>
<thead>
<tr>
<th>Python 2.6</th>
<th>Python 2.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>primes = set([2, 3, 5])</td>
<td>primes = {2, 3, 5}</td>
</tr>
<tr>
<td>empty = set()</td>
<td>empty = set()</td>
</tr>
</tbody>
</table>
Sets were added to Python after most of the language was already defined.
- But at least they're there...

<table>
<thead>
<tr>
<th>Python 2.6</th>
<th>Python 2.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>primes = set([2, 3, 5])</td>
<td>primes = {2, 3, 5}</td>
</tr>
<tr>
<td>empty = set()</td>
<td>empty = set()</td>
</tr>
</tbody>
</table>

Because `{}` was already used for something else.
Sets were added to Python after most of the language was already defined
- But at least they're there...

<table>
<thead>
<tr>
<th>Python 2.6</th>
<th>Python 3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>primes = set([2, 3, 5])</td>
<td>primes = {2, 3, 5}</td>
</tr>
<tr>
<td>empty = set()</td>
<td>empty = set()</td>
</tr>
</tbody>
</table>

Because {} was already used for something else
We'll use Python 2.7 notation in this lecture
Naturally used to find unique items in a collection
Naturally used to find unique items in a collection

# What letters are used?
letters = set()
for char in 'ichthyosaur':
    letters.add(char)
print letters

set(['a', 'c', 'i', 'h', 'o', 's', 'r', 'u', 't', 'y'])
Naturally used to find unique items in a collection

```python
# What letters are used?
letters = set()
for char in 'ichthyosaur':
    letters.add(char)
print letters

set(['a', 'c', 'i', 'h', 'o', 's', 'r', 'u', 't', 'y'])
```

Not ordered alphabetically or by order of addition
Naturally used to find unique items in a collection

```python
# What letters are used?
letters = set()
for char in 'ichthyosaur':
    letters.add(char)
print letters

set(['a', 'c', 'i', 'h', 'o', 's', 'r', 'u', 't', 'y'])
```

Not ordered alphabetically or by order of addition

Because set elements are *not ordered*
A much shorter way to accomplish the same goal
A much shorter way to accomplish the same goal

```python
# What letters are used?
print set('ichthyosaur')

set(['a', 'c', 'i', 'h', 'o', 's', 'r', 'u', 't', 'y'])
```
A much shorter way to accomplish the same goal

```python
# What letters are used?
print set('ichthyosaur')
```

```python
set(['a', 'c', 'i', 'h', 'o', 's', 'r', 'u', 't', 'y'])
```

If you can loop over it, you can build a set from it
A much shorter way to accomplish the same goal

```python
# What letters are used?
print set('ichthyosaur')
```

```python
set(['a', 'c', 'i', 'h', 'o', 's', 'r', 'u', 't', 'y'])
```

If you can loop over it, you can build a set from it

Can *not* build a set from several separate items
A much shorter way to accomplish the same goal

# What letters are used?
print set('ichthyosaur')

```
set(['a', 'c', 'i', 'h', 'o', 's', 'r', 'u', 't', 'y'])
```

If you can loop over it, you can build a set from it

Can *not* build a set from several separate items

```
set('a', 'e', 'i', 'o', 'u')
```

_TypeError: set expected at most 1 arguments, got 5_
>>>
>>> ten = set(range(10))  # {0...9}
>>> lows = {0, 1, 2, 3, 4}
>>> odds = {1, 3, 5, 7, 9}
>>> ten = set(range(10))  # {0...9}
>>> lows = {0, 1, 2, 3, 4}
>>> odds = {1, 3, 5, 7, 9}

# add an element

>>> lows.add(9)
>>> lows
set([0, 1, 2, 3, 4, 9])
>>> ten = set(range(10))     # {0...9}
>>> lows = {0, 1, 2, 3, 4}
>>> odds = {1, 3, 5, 7, 9}

# add an element
>>> lows.add(9)
>>> lows
set([0, 1, 2, 3, 4, 9])

# remove all elements
>>> lows.clear()
>>> lows
set()
# difference

```python
>>> lows.difference(odds)
set([0, 2, 4])
```
# difference

```python
>>> lows.difference(odds)
set([0, 2, 4])
```

# intersection

```python
# intersection

>>> lows.intersection(odds)
set([1, 3])
```
# difference

```python
>>> lows.difference(odds)
set([0, 2, 4])
```

# intersection

```python
>>> lows.intersection(odds)
set([1, 3])
```

# subset

```python
>>> lows.issubset(ten)
True
```
# superset

```python
>>> lows.issuperset(odds)
False
```
# superset

```python
>>> lows.issuperset(odds)
False
```

# remove an element

```python
# remove an element

>>> lows.remove(0)

>>> lows

set([1, 2, 3, 4])
```
# superset

```python
>>> lows.issuperset(odds)
False
```

# remove an element

```python
>>> lows.remove(0)
>>> lows
set([1, 2, 3, 4])
```

# symmetric difference (also called "exclusive or")

```python
>>> lows.symmetric_difference(odds)
set([2, 4, 5, 7, 9])
```
# union

```python
>>> lows.union(odds)
set([1, 2, 3, 4, 5, 7, 9])
```
# union

```python
gerun
lows.union(odds)
```

```
set([1, 2, 3, 4, 5, 7, 9])
```

# size

```python
gerun
len(odds)
```

```
7
```
# union

```python
>>> lows.union(odds)
set([1, 2, 3, 4, 5, 7, 9])
```

# size

```python
>>> len(odds)
7
```

# membership

```python
>>> 6 in odds
False
```

Sets and Dictionaries

Introduction
<table>
<thead>
<tr>
<th>Methods</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lows.difference(odds)</code></td>
<td><code>lows - odds</code></td>
</tr>
<tr>
<td><code>lows.intersection(odds)</code></td>
<td><code>lows &amp; odds</code></td>
</tr>
<tr>
<td><code>lows.issubset(ten)</code></td>
<td><code>lows &lt;= ten</code></td>
</tr>
<tr>
<td></td>
<td><code>lows &lt; ten</code></td>
</tr>
<tr>
<td><code>lows.issuperset(ten)</code></td>
<td><code>lows &gt;= odds</code></td>
</tr>
<tr>
<td></td>
<td><code>lows &gt; odds</code></td>
</tr>
<tr>
<td><code>lows.symmetric_difference(odds)</code></td>
<td><code>lows ^ odds</code></td>
</tr>
<tr>
<td><code>lows.union(odds)</code></td>
<td>`lows</td>
</tr>
</tbody>
</table>
Cannot negate a set
Cannot *negate* a set

Common in mathematics...
Cannot *negate* a set

Common in mathematics...

...but what's the negation of \{1, 2\} in a program?
Cannot *negate* a set

Common in mathematics...

...but what's the negation of \{1, 2\} in a program?

We'll solve this problem when we get to object-oriented programming
Problem: cleaning up field observations
Problem: cleaning up field observations

One file has the names of birds our supervisor thinks are uninteresting.
Problem: cleaning up field observations
One file has the names of birds our supervisor thinks are uninteresting.
Another contains the names of all birds observed during a three-week period in a mosquito-infested hellhole in northern Ontario.
Problem: cleaning up field observations
One file has the names of birds our supervisor thinks are uninteresting.
Another contains the names of all birds observed during a three-week period in a mosquito-infested hellhole in northern Ontario.
Copy the observation file, removing uninteresting birds along the way.
'''Copy file, removing items along the way.'''

import sys

if __name__ == '__main__':
    to_remove = read_set(sys.argv[1])
    reader = open(sys.argv[2], 'r')
    writer = open(sys.argv[3], 'w')
    for line in reader:
        line = line.strip()
        if line not in to_remove:
            writer.write(line)
    reader.close()
    writer.close()
'''Copy file, removing items along the way.'''

```python
import sys

if __name__ == '__main__':
    to_remove = read_set(sys.argv[1])
    reader = open(sys.argv[2], 'r')
    writer = open(sys.argv[3], 'w')
    for line in reader:
        line = line.strip()
        if line not in to_remove:
            writer.write(line)
    reader.close()
    writer.close()
```
'Copy file, removing items along the way.'

```python
import sys

if __name__ == '__main__':
    to_remove = read_set(sys.argv[1])
    reader = open(sys.argv[2], 'r')
    writer = open(sys.argv[3], 'w')
    for line in reader:
        line = line.strip()
        if line not in to_remove:
            writer.write(line)
    reader.close()
    writer.close()
```
'''Copy file, removing items along the way.'''

```python
import sys

if __name__ == '__main__':
    to_remove = read_set(sys.argv[1])
    reader = open(sys.argv[2], 'r')
    writer = open(sys.argv[3], 'w')

    for line in reader:
        line = line.strip()
        if line not in to_remove:
            writer.write(line)

    reader.close()
    writer.close()
```

Sets and Dictionaries

Introduction
'''Copy file, removing items along the way.'''

```python
import sys

if __name__ == '__main__':
    to_remove = read_set(sys.argv[1])
    reader = open(sys.argv[2], 'r')
    writer = open(sys.argv[3], 'w')
    for line in reader:
        line = line.strip()
        if line not in to_remove:
            writer.write(line)
    reader.close()
    writer.close()
```
'''Copy file, removing items along the way.'''

```python
import sys

if __name__ == '__main__':
    to_remove = read_set(sys.argv[1])
    reader = open(sys.argv[2], 'r')
    writer = open(sys.argv[3], 'w')
    for line in reader:
        line = line.strip()
        if line not in to_remove:
            writer.write(line)
    reader.close()
    writer.close()
```

Sets and Dictionaries

Introduction
def read_set(filename):
    '''Read set elements from a file.'''

    result = set()
    reader = open(filename, 'r')
    for line in result:
        line = line.strip()
        set.add(line)
    reader.close()

    return result
def read_set(filename):
    '''Read set elements from a file.'''

    result = set()
    reader = open(filename, 'r')
    for line in result:
        line = line.strip()
        set.add(line)
    reader.close()

    return result
def read_set(filename):
    '''Read set elements from a file.'''

    result = set()
    reader = open(filename, 'r')
    for line in result:
        line = line.strip()
        set.add(line)
    reader.close()
    return result
def read_set(filename):
    '''Read set elements from a file.'''

    result = set()
    reader = open(filename, 'r')
    for line in reader:
        line = line.strip()
        set.add(line)
    reader.close()
    return result
def read_set(filename):
    '''Read set elements from a file.'''

    result = set()
    reader = open(filename, 'r')
    for line in result:
        line = line.strip()
        set.add(line)

    reader.close()
    return result
to_remove = read_set(sys.argv[1])

reader = open(sys.argv[2], 'r')
writer = open(sys.argv[3], 'w')

for line in reader:
    line = line.strip()
    if line not in to_remove:
        writer.write(line)

reader.close()
writer.close()

result = set()

reader = open(filename, 'r')

for line in result:
    line = line.strip()
    set.add(line)

reader.close()

return result
<table>
<thead>
<tr>
<th>removals.txt</th>
<th>observations.txt</th>
<th>result.txt</th>
</tr>
</thead>
<tbody>
<tr>
<td>loon</td>
<td>loon</td>
<td>loon</td>
</tr>
<tr>
<td>duck</td>
<td>duck</td>
<td>duck</td>
</tr>
<tr>
<td>loon</td>
<td>loon</td>
<td>loon</td>
</tr>
<tr>
<td>ostrich</td>
<td>ostrich</td>
<td>ostrich</td>
</tr>
<tr>
<td>loon</td>
<td>loon</td>
<td>loon</td>
</tr>
<tr>
<td>ostrich</td>
<td>loon</td>
<td>loon</td>
</tr>
<tr>
<td>loon</td>
<td>loon</td>
<td>loon</td>
</tr>
<tr>
<td>ostrich</td>
<td>loon</td>
<td>loon</td>
</tr>
<tr>
<td>duck</td>
<td>loon</td>
<td>loon</td>
</tr>
<tr>
<td>loon</td>
<td>ostrich</td>
<td>ostrich</td>
</tr>
<tr>
<td>ostrich</td>
<td>loon</td>
<td>loon</td>
</tr>
<tr>
<td>loon</td>
<td>loon</td>
<td>loon</td>
</tr>
</tbody>
</table>
created by

Greg Wilson

July 2010