



# Sets

## Introduction



Copyright © Software Carpentry 2010

This work is licensed under the Creative Commons Attribution License

See <http://software-carpentry.org/license.html> for more information.

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 uses *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 uses *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 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"

- 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

---

```
primes = set([2, 3,  
5])
```

Sets were added to Python after most of the language was already defined

- But at least they're there...

### Python 2.6

```
primes = set([2, 3,  
5])
```

### Python 2.7

```
primes = {2, 3, 5}
```

Sets were added to Python after most of the language was already defined

- But at least they're there...

### Python 2.6

```
primes = set([2, 3, 5])  
  
empty = set()
```

### Python 2.7

```
primes = {2, 3, 5}  
  
empty = set()
```

Sets were added to Python after most of the language was already defined

- But at least they're there...

### Python 2.6

```
primes = set([2, 3, 5])  
  
empty = set()
```

### Python 2.7

```
primes = {2, 3, 5}  
  
empty = set()
```

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...

Python 2.6	Python 3.1
<pre>primes = set([2, 3, 5])  empty = set()</pre>	<pre>primes = {2, 3, 5}  empty = set()</pre>

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

**# 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

**# 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

**# 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

```
# 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

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

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

```
>>> lows.difference(odds)
```

```
set([0, 2, 4])
```

```
# difference
```

```
>>> lows.difference(odds)
```

```
set([0, 2, 4])
```

```
# intersection
```

```
>>> lows.intersection(odds)
```

```
set([1, 3])
```

```
# difference
```

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

```
# intersection
```

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

```
# subset
```

```
>>> lows.issubset(ten)  
True
```

# superset

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

*False*

```
# superset
```

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

```
False
```

```
# remove an element
```

```
>>> lows.remove(0)
```

```
>>> lows
```

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



```
# superset
```

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

```
False
```

```
# remove an element
```

```
>>> lows.remove(0)
```

```
>>> lows
```

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

```
# symmetric difference (also called "exclusive or")
```

```
>>> lows.symmetric_difference(odds)
```

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

# union

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

```
# union
```

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

```
# size
```

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

```
# union
```

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

```
# size
```

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

```
# membership
```

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

Methods	Operators
<code>lows.difference(odds)</code>	<code>lows - odds</code>
<code>lows.intersection(odds)</code>	<code>lows &amp; odds</code>
<code>lows.issubset(ten)</code>	<code>lows &lt;= ten</code>
	<code>lows &lt; ten</code>
<code>lows.issuperset(ten)</code>	<code>lows &gt;= odds</code>
	<code>lows &gt; odds</code>
<code>lows.symmetric_difference(odds)</code>	<code>lows ^ odds</code>
<code>lows.union(odds)</code>	<code>lows   odds</code>

# STOP











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."""

**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."""

**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."""

**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."""

**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."""

**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()**



```
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 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
```

```
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
```

removals.txt	observations.txt	result.txt
	loon duck loon ostrich loon	loon duck loon ostrich loon
ostrich	loon duck loon ostrich loon	loon duck loon loon
duck loon ostrich	loon duck loon ostrich loon	



created by

Greg Wilson

July 2010



Copyright © Software Carpentry 2010

This work is licensed under the Creative Commons Attribution License

See <http://software-carpentry.org/license.html> for more information.