Sets

• Mathematical set: a collection of values, without duplicates or order
• Order does not matter
  \{ 1, 2, 3 \} == \{ 3, 2, 1 \}
• No duplicates
  \{ 3, 1, 4, 1, 5 \} == \{ 5, 4, 3, 1 \}
• For every data structure, ask:
  – How to create
  – How to query (look up) and perform other operations
    • (Can result in a new set, or in some other datatype)
  – How to modify
Answer:  http://docs.python.org/2/library/stdtypes.html#set
Two ways to create a set

1. Direct mathematical syntax
   
   `odd = { 1, 3, 5 }`
   `prime = { 2, 3, 5 }`

   Cannot express empty set: “{}” means something else 😞

2. Construct from a list
   
   `odd = set([1, 3, 5])`
   `prime = set([2, 3, 5])`
   `empty = set([])`

   Python always prints using this syntax
Set operations

odd = { 1, 3, 5 }
prime = { 2, 3, 5 }

• membership ∈ Python: in  4 in prime ⇒ False
• union ∪ Python: |  odd | prime ⇒ {1, 2, 3, 5 }
• intersection ∩ Python: &  odd & prime ⇒ {3, 5 }
• difference \ or - Python: −  odd − prime ⇒ {1}

• Iteration over sets:
  # iterates over items in arbitrary order
  for item in myset:
    ...

• Think in terms of set operations,
  not in terms of iteration and element operations
  – Shorter, clearer, less error-prone, faster
Modifying a set

- Add one element to a set:
  myset.add(newelt)
  myset = myset | { newelt }

- Remove one element from a set:
  myset.remove(elt)  # elt must be in myset
  myset.discard(elt)  # never errs
  myset = myset - { elt }

  What would this do?
  myset = myset - elt

- Choose and remove some element from a set:
  myset.pop()
Practice with sets
List vs. set operations (1)

Find the common elements in both list1 and list2:
```python
out1 = []
for i in list2:
    if i in list1:
        out1.append(i)
```

# We will learn about list comprehensions later
out1 = [i for i in list2 if i in list1]

Find the common elements in both set1 and set2:
```python
set1 & set2
```

Much shorter, clearer, easier to write!
List vs. set operations (2)

Find the elements in either list1 or list2 (or both):

```python
out2 = list(list1)  # make a copy
for i in list2:
    if i not in list1:
        out2.append(i)
```

```python
out2 = list1+list2
for i in out1:  # out1 = common elements in both lists
    out2.remove(i)
```

Find the elements in either set1 or set2 (or both):

```
set1 | set2
```
List vs. set operations (3)

Find the elements in either list but not in both:

```python
out3 = []
for i in list1+list2:
    if i not in list1 or i not in list2:
        out3.append(i)
```

Find the elements in either set but not in both:

```python
set1 ^ set2
```
Not every value may be placed in a set

- Set elements must be immutable values
  - int, float, bool, string, tuple
  - not: list, set, dictionary
- Goal: only set operations change the set
  - after “myset.add(x)”, x in myset ⇒ True
  - y in myset always evaluates to the same value
    Both conditions should hold until myset is changed
- Mutable elements can violate these goals

```python
list1 = ['a', 'b']
list2 = list1
list3 = ['a', 'b']
myset = { list1 }  # Hypothetical; actually illegal in Python
list1 in myset ⇒ True
list3 in myset ⇒ True
list2.append('c')
list1 in myset ⇒ ???
list3 in myset ⇒ ???
```