Sets

Sets, as in Mathematical Sets

- In mathematics, a set is a collection of objects, potentially of many different types.
- In a set, no two elements are identical. That is, a set consists of elements each of which is unique compared to the other elements.
- There is no order to the elements of a set
- A set with no elements is the empty set

Creating a Set

```
mySet = set("abcd")
```

- The "set" keyword creates a set.
- The single argument that follows must be *iterable*, that is, something that can be walked through one item at a time with a `for`.
- The result is a set data structure:

```
print(mySet)
{'a', 'c', 'b', 'd'}
```

Diverse Elements

- A set can consist of a mixture of different types of elements:

```
mySet = {‘a’,1,3.14159,True}
```

- As long as the single argument can be iterated through, you can make a set of it.

No Duplicates

- Duplicates are automatically removed.

```
mySet = set(“aabbccdd”)
print(mySet)
{'a', 'c', 'b', 'd`}
```

Common Operators

Most data structures respond to these:

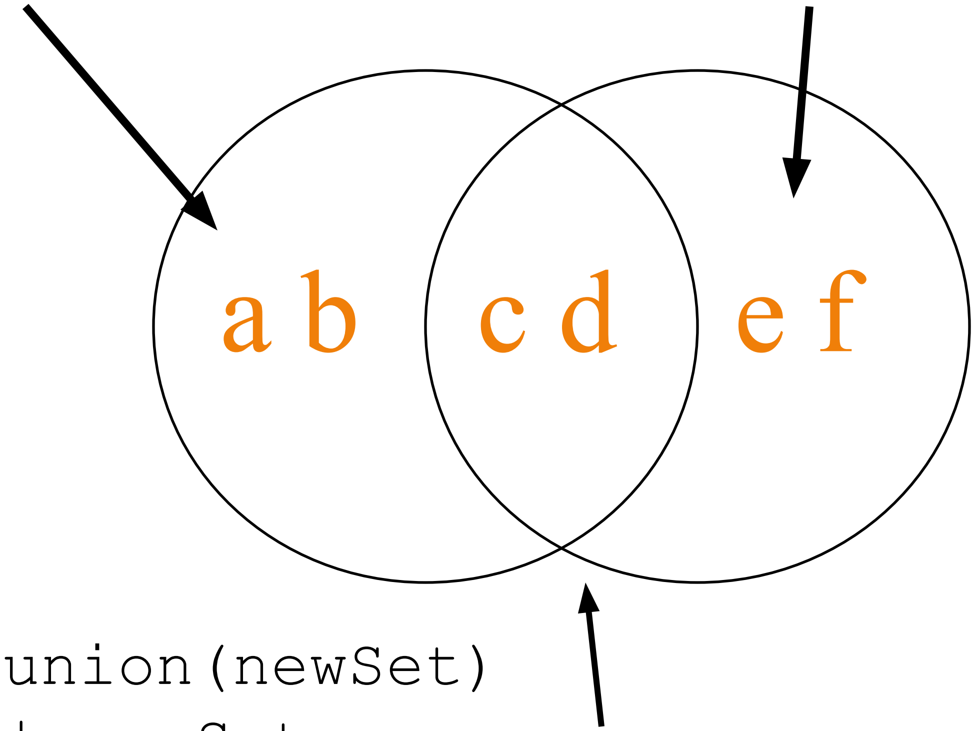
- `len(mySet)`
  - the number of elements in a set
- `element in mySet`
  - boolean indicating whether element is in the set
- `for element in mySet:`
  - iterate through the elements in `mySet`

Set Operators

- The set data structure provides some special operators that correspond to the operators you learned in middle school.
- These are various combinations of set contents.

Set Ops, Union

mySet=set(“abcd”); newSet=set(“cdef”)



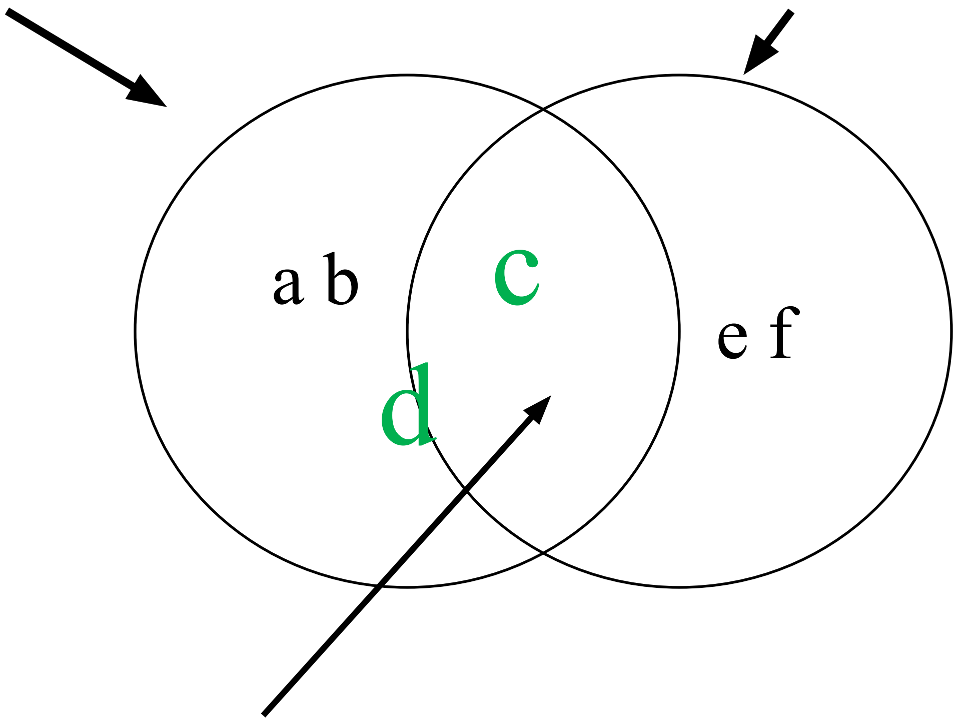
```
mySet.union(newSet)
mySet | newSet
```

returns

```
{‘a’,’b’,‘c’,’d’,’e’,’f’}
```

Set Ops, Intersection

mySet=set(“abcd”); newSet=set(“cdef”)

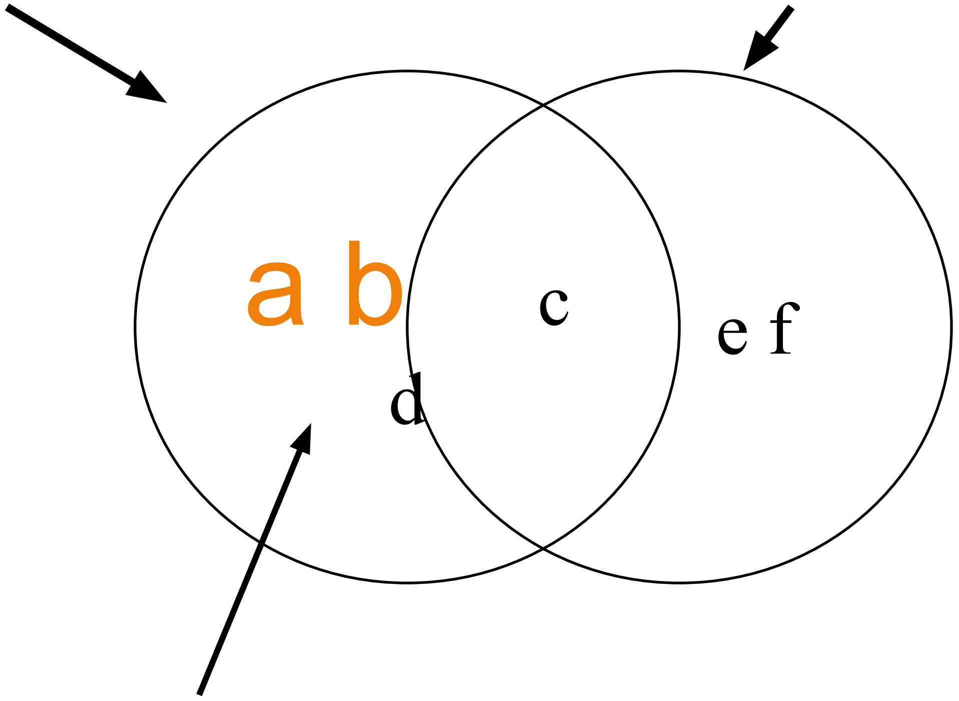


```
mySet.intersection(newSet)
mySet & newSet
```

returns {‘c’,’d’}

Set Ops, Difference

mySet=set(“abcd”); newSet=set(“cdef”)

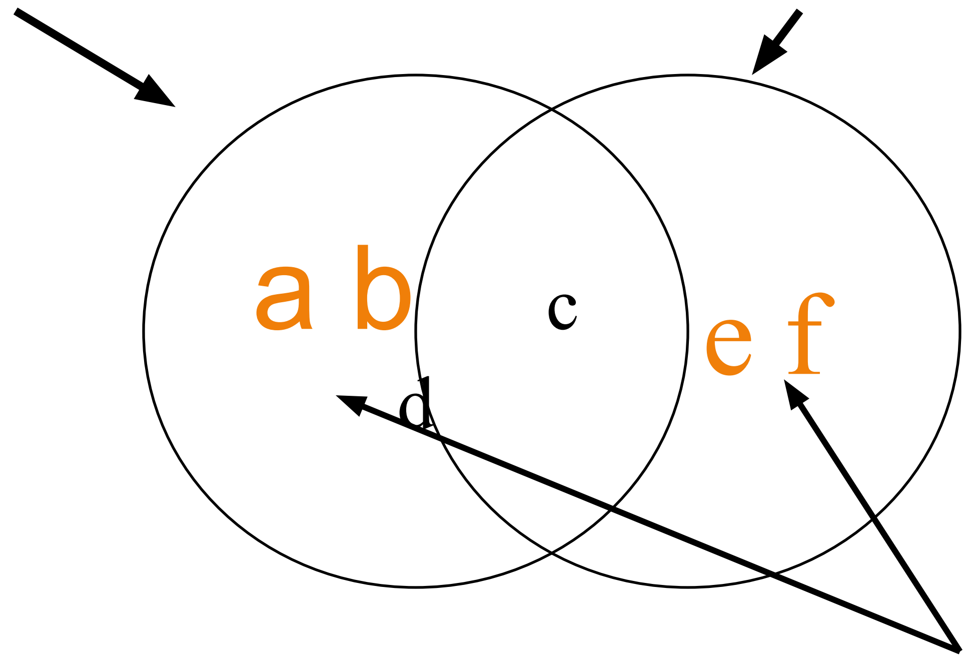


```
mySet.difference(newSet)
mySet - newSet
```

returns `{‘a’,’b’}`

Set Ops, symmetricDifference

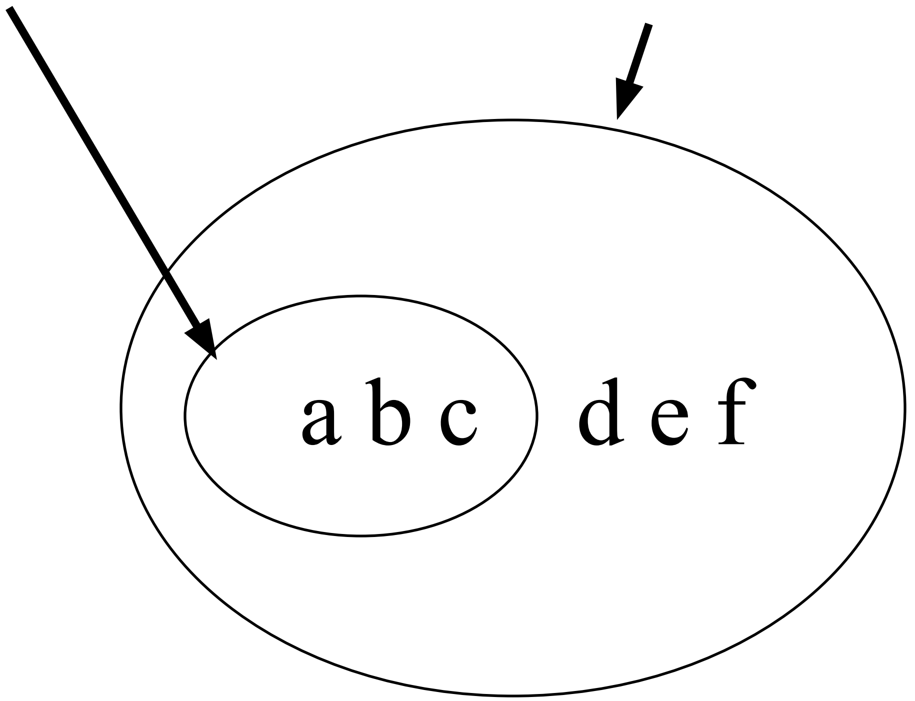
mySet=set(“abcd”); newSet=set(“cdef”)



```
mySet.symmetric_difference(newSet)
mySet ^ newSet
```
returns {‘a’,’b’,’e’,’f’}

Set Ops, super and sub set

mySet=set(“abc”); newSet=set(“abcdef”)

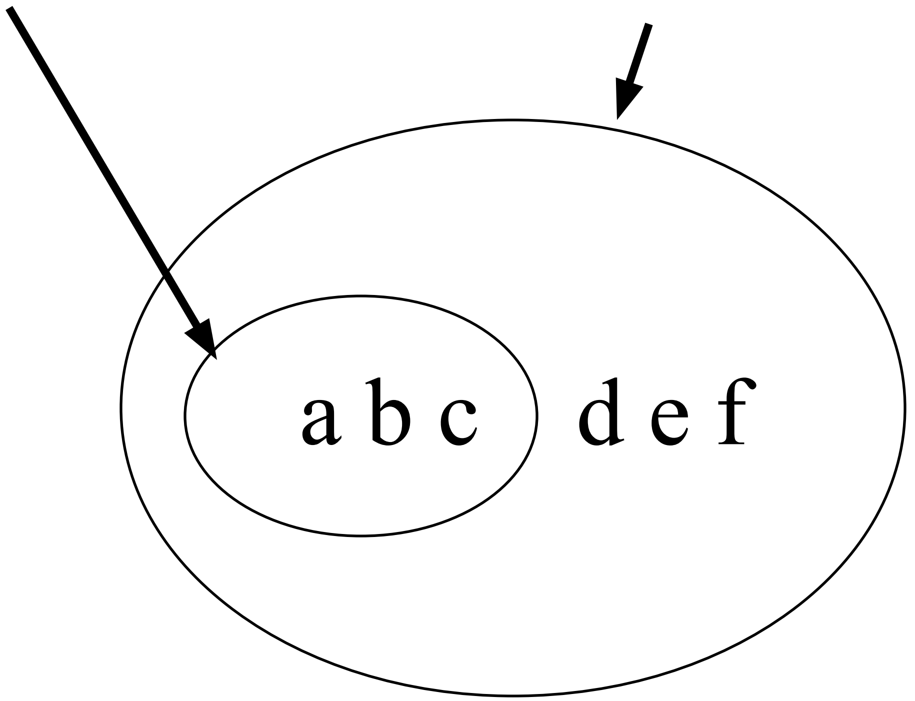


```
mySet.issubset(newSet)
mySet <= newSet
```

returns `True`

Set Ops, super and sub set

mySet=set(“abc”); newSet=set(“abcdef”)



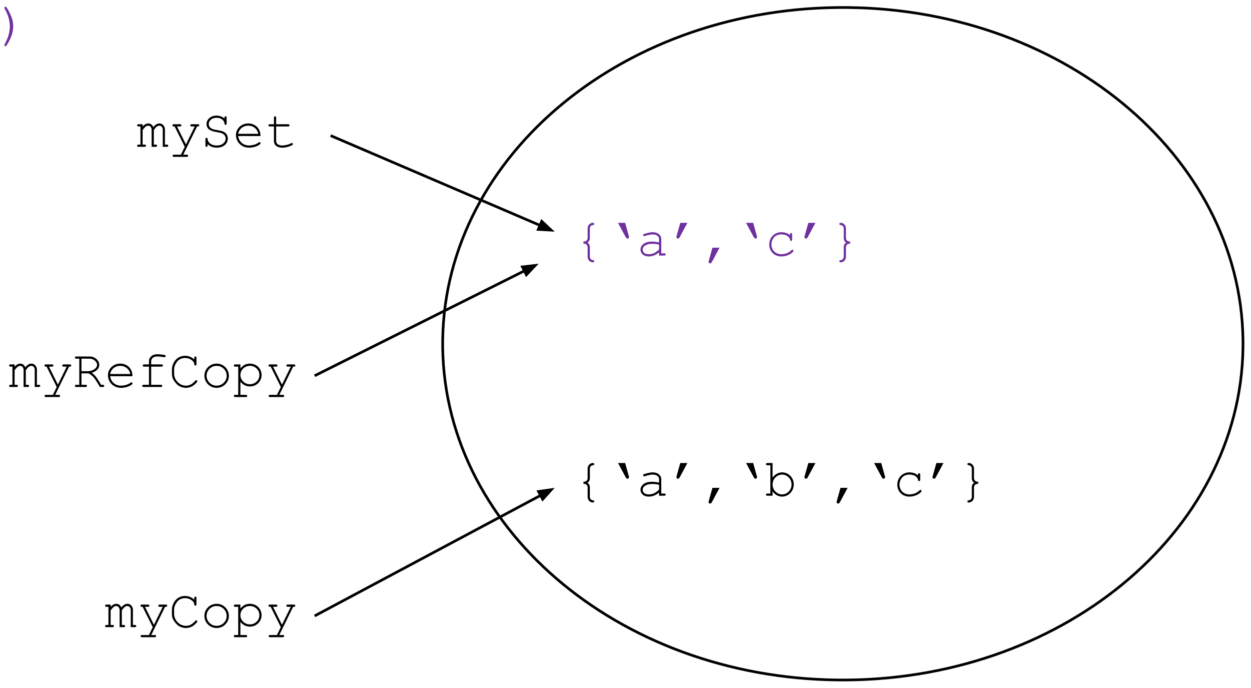
```
newSet.issuperset(mySet)
newSet>= mySet
```
returns `True`

Other Set Ops

- `mySet.add(“g”)`
  - Adds to the set, no effect if item is in set already.
- `mSet.clear()`
  - Empties the set.
- `mySet.remove(“g”)` versus `mySet.discard(“g”)`
  - `remove` throws an error if “`g`” isn’t there. `discard` doesn’t care. Both remove “`g`” from the set.
- `mySet.copy()`
  - Returns a shallow copy of `mySet.`

Copy vs. Assignment

```
mySet=set("abc")
myCopy=mySet.copy()
myRefCopy=mySet
mySet.remove('b')
```



A few examples…