Every Boilermaker Engineer Codes: 101
Entry-Level Programming in Python

Lecture 09b

Dr. John H. Cole
<jhcole@purdue.edu>

PURDUE
UNIVERSITY®

COLLEGE OF ENGINEERING

Spring 2021

Part II

Sets

Table of Contents

Sets

Set a store of unordered, unique, hashable elements

- dynamically sized
- are mutable

Useful for:

- membership testing
- removing duplicates
- supports intersection, union, difference, symmetric difference

Syntax

`s = {value1, value2, ...}` set literal notation

- enclosed in braces `{}`
- values separated by commas ‘,’

or

`s = set(iterable)` create a new set from the items of `iterable`

Terminal

```
>>> a = {1,2,3}
>>> a
{1, 2, 3}
>>> type(a)
<class 'set'>
>>> b = set([1,2,3])
>>> b
{1, 2, 3}
```

Terminal

```
>>> type({})
<class 'dict'>
>>> c = set()
>>> c
set()
>>> type(c)
<class 'set'>
>>>
```

Unordered

- elements in sets are unordered
- creating a set from a dictionary results in an unordered set of the dictionary's keys

Terminal

```
>>> a = dict(one=1, two=2, five=5)
>>> a
{'one': 1, 'two': 2, 'five': 5}
>>> b = set(a)
>>> b
{'five', 'two', 'one'}
>>>
```

Unique

- elements in sets are unique
- creating a set from a list results in a set of the unique elements of the list

Terminal

```
>>> a = set([1,1,2,1,2,2])
>>> a
{1, 2}
>>> b = set('Mississippi')
>>> b
{'p', 's', 'M', 'i'}
```

Hashable

- elements in sets must be hashable

Terminal

```
>>> a = {(), []}
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
>>> b = set([(), {}])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'dict'
```

Indexing

Sets cannot be indexed

`s[index]` raises a `TypeError`

Terminal

```
>>> a = set(1,2,3)
>>> a
{1, 2, 3}
>>> a[1]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object does not support indexing
```

s.pop()

`s.pop()` remove and return an arbitrary element

- raises a `KeyError` if the set is empty

Terminal

```
>>> a = {1,2,3}
>>> a
{1, 2, 3}
>>> a.pop()
1
>>> a
{2, 3}
>>> a.pop()
2
>>>
```

Terminal

```
>>> a
{3}
>>> a.pop()
3
>>> a
set()
>>> a.pop()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'pop from an empty set'
```

s.add(element)

`s.add(element)` add element `element` to set `s`

Terminal

```
>>> a = {1,2,3}
>>> a
{1, 2, 3}
>>> a.add(5)
>>> a
{1, 2, 3, 5}
>>> a.add(5)
>>> a
{1, 2, 3, 5}
```

s.update(iterable[, ...])

`s.update(iterable)` add each element in `iterable` to `s`

Terminal

```
>>> a = {1,2}
>>> a.update([5])
>>> a
{1, 2, 5}
>>> a.update('abc', [3,4])
>>> a
{1, 2, 'c', 3, 5, 4, 'b', 'a'}
>>> b = set('Mississippi')
>>> a.update(b)
>>> a
{1, 2, 'c', 3, 5, 4, 'i', 'p', 'b', 's', 'a', 'M'}
```

s.remove(element)

`s.remove(element)` remove element `element` from set `s`

- raises a `KeyError` if `element` is not in `s`

Terminal

```
>>> a = {1,2,5}
>>> a
{1, 2, 5}
>>> a.remove(5)
>>> a
{1, 2}
>>> a.remove(5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 5
```

s.discard(element)

`s.discard(element)` remove element `element` from set `s` if present

- does not raises an error if `element` is missing

Terminal

```
>>> a = {1,2,5}
>>> a
{1, 2, 5}
>>> a.discard(5)
>>> a
{1, 2}
>>> a.discard(5)
>>>
```

s.clear()

`s.clear()` remove all the elements from s

Terminal

```
>>> a = {1,2,5}
>>> a
{1, 2, 5}
>>> a.clear()
>>> a
set()
>>>
```

Union

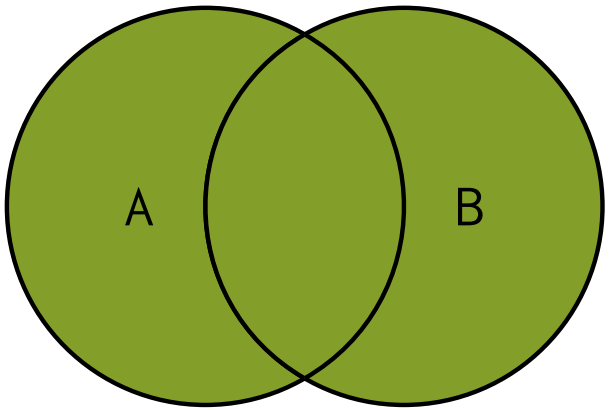
`a.union(b)`

or

`a | b` returns the *union* of a and b

Terminal

```
>>> a = {1, 2, 5, 7}
>>> b = {5, 7, 9, 0}
>>> a.union(b)
{0, 1, 2, 5, 7, 9}
>>> a | b
{0, 1, 2, 5, 7, 9}
>>> a
{1, 2, 5, 7}
>>> b
{0, 9, 5, 7}
```



Intersection

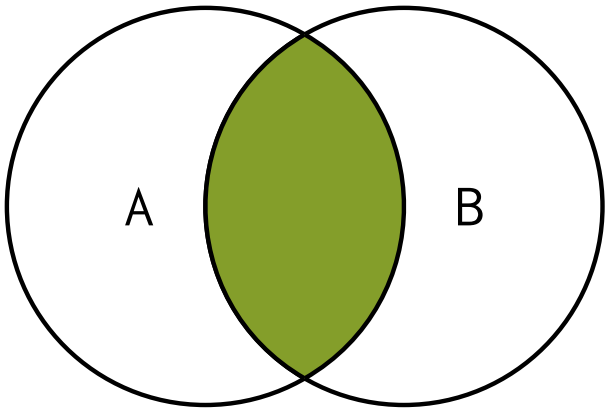
`a.intersection(b)`

or

`a & b` returns the *intersection* of a and b

Terminal

```
>>> a = {1, 2, 5, 7}
>>> b = {5, 7, 9, 0}
>>> a.intersection(b)
{5, 7}
>>> a & b
{5, 7}
>>> a
{1, 2, 5, 7}
>>> b
{0, 9, 5, 7}
```



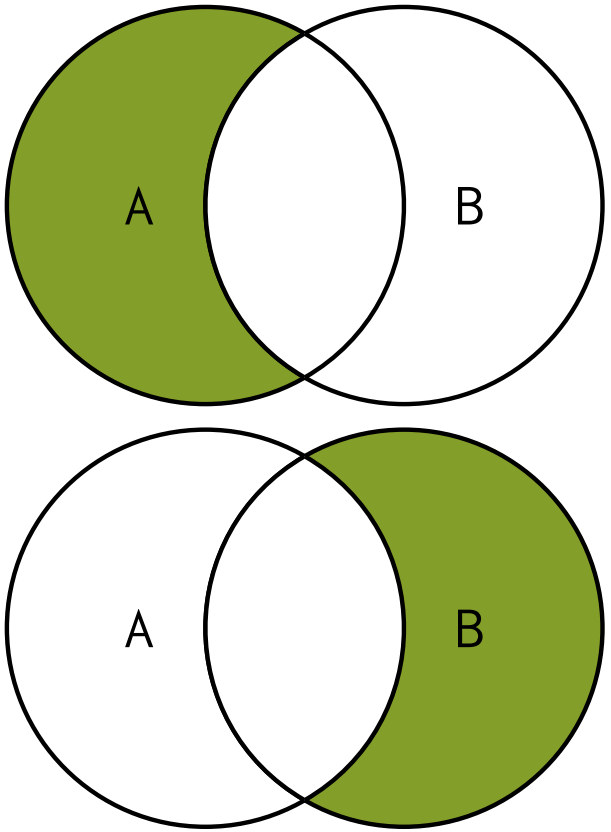
Difference

`a.difference(b)`

or

`a - b` returns the elements in a that are not in b

`b - a` returns the elements in b that are not in a



Terminal

```
>>> a = {1, 2, 5, 7}
>>> b = {5, 7, 9, 0}
>>> a - b
{1, 2}
>>> b - a
{0, 9}
```

Symmetric Difference

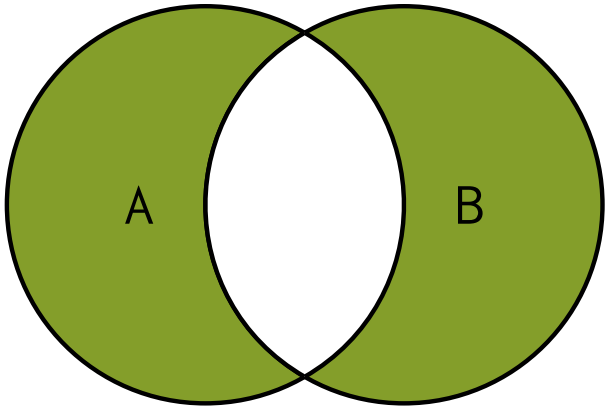
`a.symmetric_difference(b)`

or

`a ^ b` returns the elements that are in either `a` or `b` but not in both `a` and `b`

Terminal

```
>>> a = {1, 2, 5, 7}
>>> b = {5, 7, 9, 0}
>>> a ^ b
{0, 1, 2, 9}
>>> b ^ a
{0, 1, 2, 9}
```



Sorting

- sets can be arguments to the `sorted()` function

Terminal

```
>>> a = set('Mississippi')
>>> a
{'i', 'p', 's', 'M'}
>>> sorted(a)
['M', 'i', 'p', 's']
>>> a
{'i', 'p', 's', 'M'}
>>> sorted({'a', 1})
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '<' not supported between instances of
'str' and 'int'
```

Reversing

- sets cannot be `reversed()` because they do not have an order

Terminal

```
>>> a = set('Mississippi')
>>> a
{'i', 'p', 's', 'M'}
>>> reversed(a)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object is not reversible
>>> sorted(a, reverse=True)
['s', 'p', 'i', 'M']
>>> a
{'i', 'p', 's', 'M'}
>>>
```

list(s) AND tuple(s)

`list(s)` returns all the elements in s as a list

`tuple(s)` returns all the elements in s as a tuple

- the set s is not modified

Terminal

```
>>> a = {1, 2, 5}
>>> list(a)
[1, 2, 5]
>>> tuple(a)
(1, 2, 5)
>>> dups = [5, 1, 5, 2, 5, 5, 1]
>>> list(set(dups))
[1, 2, 5]
```

iter(s)

`iter(s)` returns an iterator over the elements in s

- iterates over each element in no particular order

Terminal

```
>>> i = iter({5, 1, 5, 2, 5, 5, 1})
>>> next(i)
1
>>> next(i)
2
>>> next(i)
5
>>> next(i)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

s.copy()

`s.copy()` returns a shallow copy of s

Terminal

```
>>> a = {1, 2, 5}
>>> b = a
>>> a.add(3)
>>> a
{1, 2, 3, 5}
>>> b
{1, 2, 3, 5}
```

Terminal

```
>>> c = {1, 2, 5}
>>> d = c.copy()
>>> c.add(3)
>>> c
{1, 2, 3, 5}
>>> d
{1, 2, 5}
```

len(s)

`len(s)` returns the number of elements in s

Terminal

```
>>> a = {1,2,5}
>>> a
{1, 2, 5}
>>> len(a)
3
>>> len(set('Mississippi'))
4
>>>
```

Inclusion

`element in s` returns True if s contains the element `element`

`element not in s` returns False if s contains the element `element`

Terminal

```
>>> a = set([1,2,5])
>>> a
{1, 2, 5}
>>> 5 in a
True
>>> 5 not in a
False
>>> 3 in a
False
>>> 3 not in a
True
```

Comparison – Equality

`a == b` returns `True` if `a` and `b` contain exactly the same elements

Terminal

```
>>> a = {1, 2, 3}
>>> b = {1, 2, 3}
>>> a == b
True
>>> b == a
True
```

Terminal

```
>>> c = {3, 3, 1, 3, 2}
>>> a == c
True
>>> d = {1, 2, 5}
>>> a == d, b == d, c == d
(False, False, False)
```

Comparison – Subsets

`a <= b` returns `True` if `a` is a **subset** of `b`

`a < b` returns `True` if `a` is a **proper subset** of `b`

Terminal

```
>>> a = {1, 2, 3}
>>> b = {1, 2, 3}
>>> b <= a
True
>>> b < a
False
```

Terminal

```
>>> c = {1, 2}
>>> c <= a
True
>>> c < a
True
>>>
```

Terminal

```
>>> d = {1, 2, 5}
>>> d <= a
False
>>> d < a
False
>>>
```

Comparison – Supersets

`a >= b` returns `True` if `a` is a **superset** of `b`

`a > b` returns `True` if `a` is a **proper superset** of `b`

Terminal

```
>>> a = {1, 2, 3}
>>> b = {1, 2, 3}
>>> a >= b
True
>>> a > b
False
```

Terminal

```
>>> c = {1, 2}
>>> a >= c
True
>>> a > c
True
>>>
```

Terminal

```
>>> d = {1, 2, 5}
>>> a >= d
False
>>> a > d
False
>>>
```

Loops

- iterate over the elements in a set using a for loop

Terminal

```
>>> a = set(1,2,5)
>>> for e in a:
...     print(e)
...
1
2
5
```


Thanks for watching!