set algebra[1]

1 Set algebra with Python scripts

> ***Set theory is a branch of mathematical logic that studies sets, which informally are collections of objects. Although any type of object can be collected into a set, set theory is applied most often to objects that are relevant to mathematics. The language of set theory can be used in the definitions of nearly all mathematical objects.***

Set theory is commonly employed as a foundational system for modern mathematics, particularly in the form of **Zermelo–Fraenkel set theory** with the axiom of choice.

Python offers a native data structure called set, which can be used as a proxy for a mathematical set for almost all purposes.[2]

```
In [8]: import IPython.display as disp
```

1.1 Various ways to create a 'set' object in Python

```
In [4]: # Directly with curly braces
        Set1 = {1,2}
        print (Set1)

{1, 2}
```

```
In [5]: type(Set1)

Out[5]: set
```

```
In [6]: # By calling the 'set' function i.e. typecasting
        Set2 = set({2,3})
        print(Set2)

{2, 3}
```

```
In [7]: my_list=[1,2,3,4]
        my_set_from_list = set(my_list)
        print(my_set_from_list)

{1, 2, 3, 4}
```

***** Empty (Null) set is a special set *****

$$\forall x, x \notin \varnothing$$

[1] This sample output from Jupyter Notebook is formatted using Tufte-Handout Class with additions. This style choice is arbitrary, since any LaTeX style may be used or developed for handsome LaTeX/PDF output.

[2] Boxing each code snippet and its result makes reading the code easier.

(Sample adapted from StatsUsingPython: Set_Algebra_with_Python.ipynb by Tirthajyoti Sarkar, PhD.

Click on link above to see the original Jupyter Notebook.

⇏ ***Do not try to create the empty set by declaring an empty {}. That denotes an empty dictionary object:***

```
In [8]: my_set = {}
        print(type(my_set))

<class 'dict'>
```

⇒ ***Instead, use the set() function to create the empty (null) set from any empty data type e.g. dictionary or list***

```
In [9]: my_set = set({})
        print(type(my_set))
        my_set_2 = set([])
        print(type(my_set_2))

<class 'set'>
<class 'set'>
```

2 Membership and size testing

2.1 Membership testing by 'in' and 'not in' keywords

```
In [10]: my_set = set([1,3,5])
         print("Here is my set:",my_set)
         print("1 is in the set:",1 in my_set)
         print("2 is in the set:",2 in my_set)
         print("4 is NOT in the set:",4 not in my_set)

Here is my set: {1, 3, 5}
1 is in the set: True
2 is in the set: False
4 is NOT in the set: True
```

2.2 Size checking by 'len' or 'not'

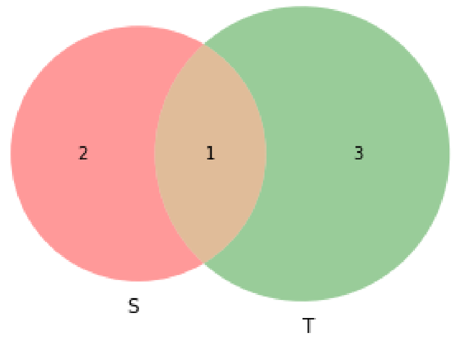
```
In [11]: S = {1,2}
         not S

Out[11]: False
```

```
In [12]: T = set()
         not T

Out[12]: True
```

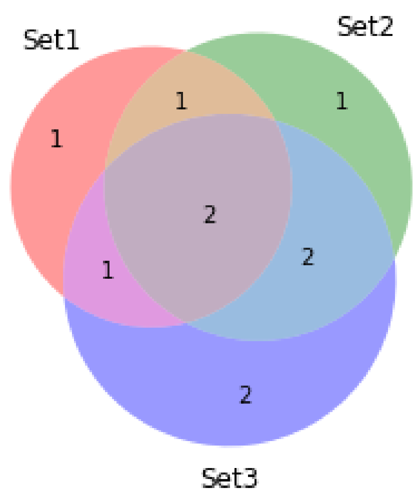
```
In [13]: print("Size of S:", len(S))
         print("Size of T:", len(T))

Size of S: 2
Size of T: 0
```

3 *Venn diagrams*

```
In [14]: import matplotlib.pyplot as plt
         import matplotlib_venn as venn
         S = {1, 2, 3}
         T = {0, 2, -1, 5}
         venn.venn2([S, T], set_labels=('S','T'))
         plt.show()
```




```
In [15]:
venn.venn3(subsets
=(1, 1,
1, 2,
1, 2,
2), set_labels =
 ('Set1', 'Set2', 'Set3'))
         plt.show()
```



4 *Set relations*

- **Subset**
- **Superset**
- **Disjoint**
- **Universal set**
- **Null set**

```
In [16]: Univ = set([x for x in range(11)])
         Super = set([x for x in range(11) if x%2==0])
         disj = set([x for x in range(11) if x%2==1])
         Sub = set([4,6])
         Null = set([x for x in range(11) if x>10])
```

```
In [17]: print("Universal set (all the positive integers up to 10):",Univ)
         print("All the even positive integers up to 10:",Super)
         print("All the odd positive integers up to 10:",disj)
         print("Set of 2 elements, 4 and 6:",Sub)
         print("A null set:", Null)

Universal set (all the positive integers up to 10): {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
All the even positive integers up to 10: {0, 2, 4, 6, 8, 10}
All the odd positive integers up to 10: {1, 3, 5, 7, 9}
Set of 2 elements, 4 and 6: {4, 6}
A null set: set()
```

```
In [18]: Super.issuperset(Sub)

Out[18]: True
```

5 *Algebra of inclusion*

If A, B and C are sets then the following hold:

Reflexivity

$$A \subseteq A$$

Antisymmetry

$$A \subseteq B \text{ and } B \subseteq A \text{ if and only if } A = B$$

Transitivity

$$\text{If } A \subseteq B \text{ and } B \subseteq C, \text{ then } A \subseteq C$$

6 *Set algebra/Operations*

- **Equality**
- **Intersection**
- **Union**
- **Complement**
- **Difference**
- **Cartesian product**

6.1 Intersection between sets

In mathematics, the intersection A ∩ B of two sets A and B is the set that contains all elements of A that also belong to B (or equivalently, all elements of B that also belong to A), but no other elements. Formally,

$$A \cap B = \{x : x \in A \text{ and } x \in B\}.$$

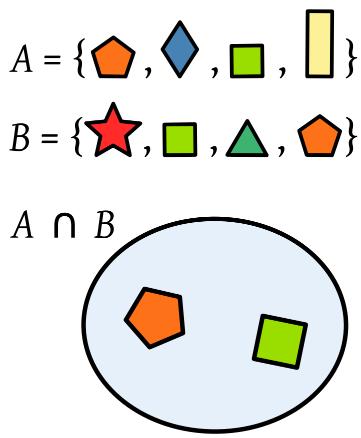


Figure 1: Set intersection

```
In [27]: # Define a set using list comprehension
         S1 = set([x for x in range(1,11) if x%3==0])
         print("S1:", S1)

S1: {9, 3, 6}
```

```
In [28]: S2 = set([x for x in range(1,5)])
         print("S2:", S2)

S2: {1, 2, 3, 4}
```

```
In [29]: # Both intersection method or & can be used
         S_intersection = S1.intersection(S2)
         print("Intersection of S1 and S2:", S_intersection)

         S_intersection = S1 & S2
         print("Intersection of S1 and S2:", S_intersection)

Intersection of S1 and S2: {3}
Intersection of S1 and S2: {3}
```

*** One can chain the methods to get intersection with more than 2 sets ***

```
In [30]: S3 = set([x for x in range(4,10)])
         print("S3:", S3)

S3: {4, 5, 6, 7, 8, 9}
```

```
In [31]: S1_S2_S3 = S1.intersection(S2).intersection(S3)
         print("Intersection of S1, S2, and S3:", S1_S2_S3)

Intersection of S1, S2, and S3: set()
```

*** Now change the S3 to contain 3***

```
In [32]: S3 = set([x for x in range(3,10)])
         print("S3:", S3)
         S1_S2_S3 = S1.intersection(S2).intersection(S3)
         print("Intersection of S1, S2, and S3:", S1_S2_S3)

S3: {3, 4, 5, 6, 7, 8, 9}
Intersection of S1, S2, and S3: {3}
```

6.2 The symbol '&' can be used for intersection

```
In [1]: A = {1, 2, 3}
        B = {5,3,1}
        print("Intersection of {} and {} is: {} with size {}".format(A,B,A&B,len(A&B)))

Intersection of {1, 2, 3} and {1, 3, 5} is: {1, 3} with size 2
```

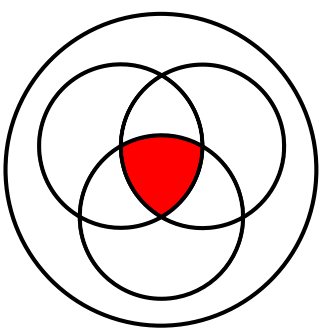


Figure 2: 3 sets intersection

Commutative law:

$$A \cap B = B \cap A$$

Associative law:

$$(A \cap B) \cap C = A \cap (B \cap C)$$