Python set operation

What is a set in Python?

- A **set is an unordered collection of items**. Every element is unique (no duplicates) and must be immutable (which cannot be changed)
- Sets can be used to perform mathematical set operations like **union, intersection, symmetric difference** etc.

How to create a set?

- A set is created by placing all the items (elements) inside curly braces **{}**, separated by comma or by using the built-in function **set().**

- **var={1,2,3,4,"Data1","Data2"}**
- **>>> type(var)**
- **<class 'set'>**

- **>>> v1=set()**
- **>>> type(v1)**
- **<class 'set'>**

Empty set

To make a set without any elements we use the **set()** function without any argument.

```
>>> var={}
>>> type(var)
<class 'dict'>  ← dictionary
>>> v1=set()
>>>
>>> type(v1)
<class 'set'>
>>>
>>> len(v1)
0
```

What is a set in Python?

- A set is an **unordered collection of items**.

```
>>> v1={10,20,30,40,50,"AB","/etc","xfs","ext4"}
>>> print(v1)
    {'ext4', 40, 10, 50, 20, 'AB', 'xfs', '/etc', 30}
```

- Every element is **unique** (no duplicates) and must be **immutable** (which cannot be changed).

```
>>> v2={10,20,30,10,20,"DATA1","data1","DATA1"}
>>> print(v2)
{10, 'DATA1', 'data1', 20, 30} # there is no duplicate element
```

What is a set in Python?

- We cannot access or change an element of set using indexing or slicing.
- **set** does not support it.

```
>>> v2={10,20,30}
>>>
>>> type(v2)
<class 'set'>

>>> v2[0]  ←
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object does not support indexing
>>>
>>> print(v2)
{10, 20, 30}
>>>
```

How to change a set in Python?

- Sets are mutable. But since they are unordered, indexing have no meaning.
- We cannot access or change an element of set using indexing or slicing.
- We can add single element using the **add()** method and multiple elements using the **update()** method.
- The update() method can take tuples, lists, strings or other sets as its argument.

add() vs update()

```
>>> v3={"Data1","Data2","Data3"}
>>>
>>> len(v3)
3
>>> print(v3)
{'Data1', 'Data2', 'Data3'}
>>>
>>>
>>> v3.add("Data4")
>>> v3
{'Data1', 'Data2', 'Data4', 'Data3'}
>>>
>>> v3.add("Data4")
>>> v3
{'Data1', 'Data2', 'Data4', 'Data3'}
>>>
>>> # avoiding duplicate entry
...
```

```
>>> v4={"Text1","Text2"}
>>>
>>> v4.update(["Text3\n","Text4\n","Text5\n"])
>>> v4
{'Text1', 'Text2', 'Text3\n', 'Text5\n', 'Text4\n'}
>>>
>>>
v4.update(("Text3\n","Text4\n","Text6\n"))
>>>
>>> v4
{'Text1', 'Text2', 'Text6\n', 'Text3\n', 'Text5\n',
'Text4\n'}
>>>
>>> len(v4)
6
```

How to remove elements from a set?

- A particular item can be removed from set using methods, **discard()** and **remove().**
- while using discard() if the item does not exist in the set, it remains unchanged.
- But remove() will raise an error in such condition.

remove() vs discard()

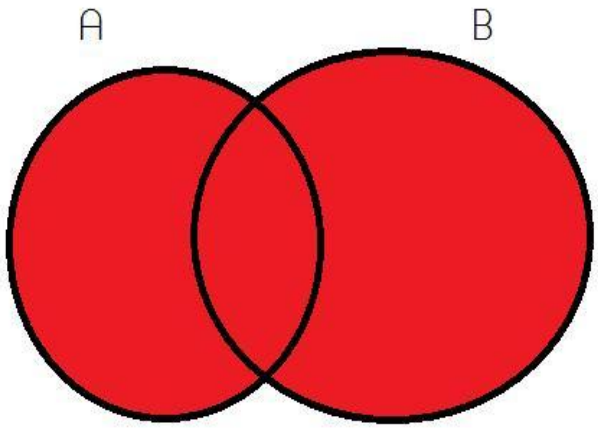
```
>>> v3={"Data1","Data2","Data3"}
>>>
>>> len(v3)
3
>>> print(v3)
{'Data1', 'Data2', 'Data3'}
>>>
>>>
>>> v3.remove("Data4")
>>> v3
{'Data1', 'Data2', 'Data3'}
>>>
>>> v3.remove("Data7")
>>> traceback (most recent call last):
 File "<stdin>", line 1, in <module>
eyError: 'data7'
```

```
>>> v4={"Text1","Text2"}
>>>
>>> v4.discard(“Text2”)
>>> v4
{'Text1‘}
>>>
>>> v4.discard(“Text5”)
>>>
>>> v4
{'Text1’}
```

Python Set Operations

- Sets can be used to carry out mathematical set operations like **union, intersection, difference and symmetric difference**.
- We can do this with operators or methods.
- >>> A = {1, 2, 3, 4, 5}
- >>> B = {4, 5, 6, 7, 8}

Set Union

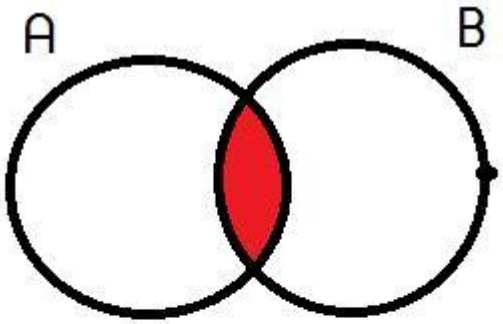


A|B
A.union(B)

B.union(A)

```
>>> A={1,2,3,4,5}
>>>
>>> B={1,2,3,6,7,8,9}
>>>
>>> A.union(B)
{1, 2, 3, 4, 5, 6, 7, 8, 9}
>>>
>>> B.union(A)
{1, 2, 3, 4, 5, 6, 7, 8, 9}
>>>
>>> A|B
{1, 2, 3, 4, 5, 6, 7, 8, 9}
>>>
>>> B|A
{1, 2, 3, 4, 5, 6, 7, 8, 9}
>>>
```

Set Intersection



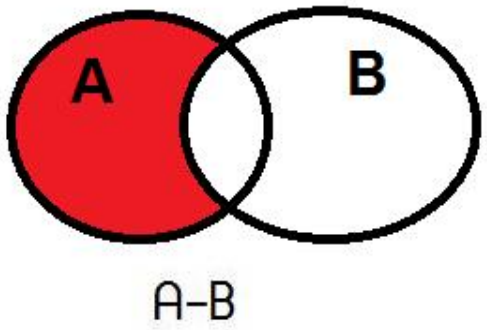
A.intersection(B)
B.intersection(A)

A & B
B &A

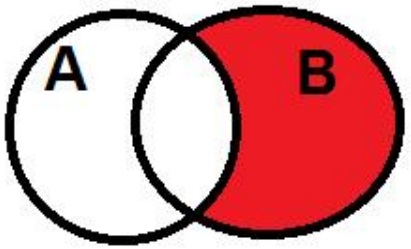
```
>>> A={1,2,3,4,5}
>>> B={1,2,3,6,7,8,9}
>>>
>>> A&B
{1, 2, 3}
>>>
>>> B&A
{1, 2, 3}
>>>
>>> A.intersection(B)
{1, 2, 3}
>>>
>>> B.intersection(A)
{1, 2, 3}
```

Set Difference

- Difference of A and B (A - B) is a set of elements that are only in A but not in B. Similarly, B - A is a set of element in B but not in A.



A–B



B–A

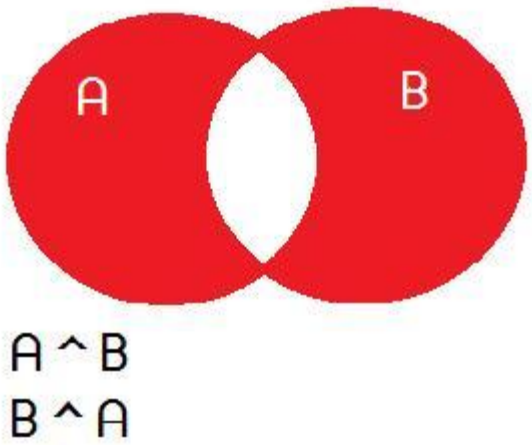
set difference

- >>> A={1,2,3,4,5}
- >>> B={1,2,3,6,7,8,9}
- >>>
- >>> **A-B**
- {4, 5}
- >>> **B-A**
- {8, 9, 6, 7}
- >>>
- >>> **A.difference(B)**
- {4, 5}
- >>>
- >>> **B.difference(A)**
- {8, 9, 6, 7}
- >>>

set symmetric difference

- Symmetric Difference of A and B is a set of elements in both A and B except those that are common in both.
- Symmetric difference is performed using **^** operator.
- Same can be accomplished using the method **symmetric_difference()**.

Set Symmetric Difference



- >>> A={1,2,3,4,5}
- >>>
- >>> B={1,2,3,6,7,8,9}
- >>>
- **>>> A^B**
- {4, 5, 6, 7, 8, 9}
- >>>
- **>>> B^A**
- {4, 5, 6, 7, 8, 9}
- >>>
- **>>> A.symmetric_difference(B)**
- {4, 5, 6, 7, 8, 9}
- >>>
- **>>> B.symmetric_difference(A)**
- {4, 5, 6, 7, 8, 9}
- >>>

File1 : process1

```
student@krosumlabs ~]$ cat -n process1.log
     1  UID        PID  PPID  C STIME TTY          TIME CMD
     2  student   2317  2310  0 23:13 pts/0    00:00:00 bash
     3  student   2541  2317  0 23:15 pts/0    00:00:00 ps -f
     4  UID        PID  PPID  C STIME TTY          TIME CMD
     5  student   2317  2310  0 23:13 pts/0    00:00:00 bash
     6  student   2551  2317  0 23:15 pts/0    00:00:00 ps -f
     7  UID        PID  PPID  C STIME TTY          TIME CMD
     8  student   2317  2310  0 23:13 pts/0    00:00:00 bash
     9  student   2558  2317  0 23:15 pts/0    00:00:00 ps -f
    10  student   2559  2317  0 23:15 pts/0    00:00:00 [tee]
    11  Filesystem     Type  Size  Used Avail Use% Mounted on
    12  /dev/sda3      xfs    19G  474M   19G   3% /home
student@krosumlabs ~]$
```

File1 : process2

```
[student@krosumlabs ~]$ cat -n process2.log
     1  UID        PID  PPID  C STIME TTY          TIME CMD
     2  student   2317  2310  0 23:13 pts/0    00:00:00 bash
     3  student   2541  2317  0 23:15 pts/0    00:00:00 ps -f
     4  UID        PID  PPID  C STIME TTY          TIME CMD
     5  student   2317  2310  0 23:13 pts/0    00:00:00 bash
     6  23:16:37 up 17 min,  2 users,  load average: 0.05, 0.09, 0.19
     7  student   2551  2317  0 23:15 pts/0    00:00:00 ps -f
     8  UID        PID  PPID  C STIME TTY          TIME CMD
     9  student   2317  2310  0 23:13 pts/0    00:00:00 bash
    10  student   2558  2317  0 23:15 pts/0    00:00:00 ps -f
    11  student   2559  2317  0 23:15 pts/0    00:00:00 [tee]
    12  Filesystem     Type  Size  Used Avail Use% Mounted on
    13  /dev/sda2      xfs    47G  3.8G   43G   9% /
[student@krosumlabs ~]$
```

Combine two files filter unique (common) lines

```
>>> f1=open("process1.log")
>>> f2=open("process2.log")
>>>
>>> L1=f1.readlines() # convert to list
>>> L2=f2.readlines() # convert to list
>>>
>>> # combine process1.log and process2.log file together
...
>>> s1=set(L1) # convert to set
>>> s2=set(L2) # convert to set
>>>
>>> s3=s1|s2  # set union operation
>>> type(s3)
<type 'set'>
>>> #set s3 is holding both s1 and s2 unique data
...
>>> L3=list(s3) # convert to list
>>>
>>> with open("process3.log","w") as f3:
...     for v in L3:
...             f3.write(v) # writing to file(process3.log)
...
>>>
```

File 3:process3.log

```
[student@krosumlabs ~]$ cat -n process3.log
     1  student   2317  2310  0 23:13 pts/0    00:00:00 bash
     2  student   2541  2317  0 23:15 pts/0    00:00:00 ps -f
     3  23:16:37 up 17 min,  2 users,  load average: 0.05, 0.09, 0.19
     4  student   2551  2317  0 23:15 pts/0    00:00:00 ps -f
     5  /dev/sda2      xfs    47G  3.8G   43G   9% /
     6  Filesystem     Type  Size  Used Avail Use% Mounted on
     7  UID        PID  PPID  C STIME TTY          TIME CMD
     8  student   2558  2317  0 23:15 pts/0    00:00:00 ps -f
     9  /dev/sda3      xfs    19G  474M   19G   3% /home
    10  student   2559  2317  0 23:15 pts/0    00:00:00 [tee]
[student@krosumlabs ~]$
```