image

Sets

- Contains only unique elements
- Are unordered in nature

```
# Creating a set

s = {}
type(s)
```

```
dict
```

```
s = {"Rahul"}
type(s)
```

```
set
```

```
s
```

```
{'Rahul'}
```

```
# Creating an empty set
```

```
s = set()
```

```
type(s)
```

```
set
```

```
s
```

```
set()
```

```
len(s)
```

```
0
```

```
s = set("Rahul Janghu")
```

```
s
```

```
{' ', 'J', 'R', 'a', 'g', 'h', 'l', 'n', 'u'}
```

```
# using set
# Adding value to a set

# add

s.add("Hello")

s

{' ', 'Hello', 'J', 'R', 'a', 'g', 'h', 'l', 'n', 'u'}

hash("2134")

4151583095651663224
```

```
# Sets are mutable type of data

id(s)

140617784634272

s.add("rahul")

s

{' ', 'Hello', 'J', 'R', 'a', 'g', 'h', 'l', 'n', 'rahul', 'u'}

id(s)

140617784634272
```

```
# update

s1 = {"Yash ji", 25}

s.update(s1)

s

{' ', 25, 'Hello', 'J', 'R', 'Yash ji', 'a', 'g', 'h', 'l', 'n',
'rahul', 'u'}

s.update("Rohit")

s

{' ',
 25,
 'Hello',
 'J',
 'R',
 'Yash ji',
 'a',
 'g',
 'h',
 'i',
 'l',
 'n',
 'o',
 'rahul',
 't',
 'u'}
```

Deleting an element

- pop: removes random element. We are not sure what it is
- remove(element): Removes particular element

```
# pop
```

```
print(s)
```

```
{'u', 'R', 'Yash ji', 'h', ' ', 'g', 'rahul', 'o', 'Hello', 'i', 'a',
'n', 25, 't', 'l', 'J'}
```

```
deleted = s.pop()
```

```
deleted
```

```
'u'
```

```
s.pop()
```

```
'g'
```

```
s
```

```
{25, 'Hello', 'J', 'a', 'i', 'l', 'n', 'o', 'rahul', 't'}
```

```
# remove
```

```
# s.remove("Football")
```

```
s.remove("rahul")
```

```
s
```

```
{25, 'Hello', 'J', 'a', 'i', 'l', 'n', 'o', 't'}
```

```
# Membership operator
# in
```

```
s
```

```
{25, 'Hello', 'J', 'a', 'i', 'l', 'n', 'o', 't'}
```

```
"Rahul" in s
```

```
False
```

```
25 in s
```

```
True
```

General operations

- min
- max
- len
- sum

```
unique = {12, 3, 4, 5, 67}
```

```
min(unique)
```

```
3
```

```
max(unique)
```

```
67
```

```
len(unique)
```

```
5
```

```
sum(unique)
```

```
91
```

Iterating on a set

```
s
```

```
{25, 'Hello', 'J', 'a', 'i', 'l', 'n', 'o', 't'}
```

```
for i in s:
    print(i, end=" ")
```

```
o Hello i a n 25 t l J
```

```
# s[0]

# Sets doesnt support indexing?
```

```
# Vocabulary: All unique words

sen = "Be the change that you want to see in the world"

# set(sen)

len(set(sen.split()))
```

```
10
```

Intersection

- Suppose you want to find out which students are enrolled in both the Calculus and Linear Algebra Course. Then you can use the intersection method.

```
# Common in both sets

# Same as high school maths

linear = {"Rahul", "Manish", "Aniket", "tony stark"}

algebra = {"Rahul", "Anjali", "Captain America"}

linear.intersection(algebra)
```

```
{'Rahul'}
```

Union

- Suppose you want to find out which students are enrolled in either the Calculus or the Linear Algebra Course or in both. Then you can use the union method.

```
# All elements in both sets

linear.union(algebra)
```

```
{'Aniket', 'Anjali', 'Captain America', 'Manish', 'Rahul', 'tony
stark'}
```

Difference

- Suppose you want to find out the set of students who have enrolled in the Calculus course but not in Linear Algebra course or vice-versa, then we can use the difference method.

```
linear

{'Aniket', 'Manish', 'Rahul', 'tony stark'}

algebra

{'Anjali', 'Captain America', 'Rahul'}

linear.difference(algebra)

{'Aniket', 'Manish', 'tony stark'}

algebra.difference(linear)

{'Anjali', 'Captain America'}
```

Linear Search

```
li = [2, 4, 8, 1, 3, 9]
target = 1

def linear_search(search_space, target):
    n = len(search_space)

    for i in range(n):
        if search_space[i] == target:
            return i
    return "Not found"

linear_search(li, target)

3
```

Binary Search

```
li = [1, 2, 10, 11, 19, 29, 30]
target = 45
```

```python
# s: 0, 4, 6, 7
# e: 6
# mid: 3, 5, 6

def binary_search(search_space, target):

    s = 0
    e = len(search_space) - 1

    # We will run loop while s <= end
    while s <= e:
        # find mid

        mid = (s + e)//2
        # compare
        if target == search_space[mid]:
            return mid
        elif target < search_space[mid]:
            # discard right
            e = mid - 1
        else:
            # discard left
            s = mid + 1
    return "Not found"
```

```python
binary_search(li, target)
```

```
'Not found'
```

```python
# HW : Write code for reverse sorted list
```