Sets

Sets are unordered collections
Sets insertion order is not same or not preserved.
Sets are non index based collections, where we cannot use indexing and slicing.
Sets do not allows duplicate values/duplicate elements.
In application development sets are used to group individual values/elements where duplicates are not allowed and perform mathematical operations (union, intersection, difference,..) ,remove duplicates from sequences (list, tuple, string,…) and membership testing.

Python support 2 set types

1. set (mutable)
2. frozenset (immutable)

set allows only immutable objects (int, float, complex, bool, NoneType, tuple, string, range, bytes)

set
set is mutable collection. After creating set changes can be done.
It allows only 2 mutable operations.

1. adding
2. removing

How to create set?
Set can be created in different ways

1. using curly braces with elements separated by comma
   set-name={ele,ele,ele,ele,…}

```
A={10,20,30,40,50}
>>> print(A)
{50, 20, 40, 10, 30}
>>> B=[10,20,30,40,50]
>>> print(B)
[10, 20, 30, 40, 50]
```

```
>>> print(type(A),type(B))
<class 'set'> <class 'list'>
>>> C={10,10,10,10,10,10}
>>> print(C)
{10}
>>> D={10,20,10,20,10,20,30,30,40,40,50,50}
>>> print(D)
{50, 20, 40, 10, 30}
>>> E={10,10.5,1+2j,"Python"}
>>> print(E)
{'Python', 10.5, 10, (1+2j)}
```

Note: set organize data using a data structure called "HASHING".

2. Set can be using set() function
    1. Set() : This creates empty set
    2. Set(iterable) : This create set using existing collections or iterables (converting other iterables into set)

Note: empty curly braces represents empty dictionary

```
>>> F={}
>>> print(F,type(F))
{} <class 'dict'>

>>> S1=set()
>>> print(S1,type(S1))
set() <class 'set'>
>>. S2=set(range(10,60,10))
>>> print(S2)
{40, 10, 50, 20, 30}
>>> S3=set("PYTHON")
>>> print(S3)
{'Y', 'P', 'N', 'T', 'H', 'O'}
>>> A=[10,10,20,30,40,50,50,40]
>>> S4=set(A)
>>> print(A,S4,sep="\n")
[10, 10, 20, 30, 40, 50, 50, 40]
```

```
{40, 10, 50, 20, 30}
>>> S5=set({10,20,30,40,50})
>>> print(S5,type(S5))
{50, 20, 40, 10, 30} <class 'set'>
>>> S6=set((10,20,30,40,50))
>>> print(S6,type(S6))
{40, 10, 50, 20, 30} <class 'set'>
```

How to read content of set?

1. Using for loop
2. Using iterator
3. Using enumerate

Example:

```
# Reading content of set
A={10,20,30,40,50}

print("Reading using for loop")
for x in A:
   print(x,end=' ')

print()
print("Reading using iterator")
b=iter(A)
e1=next(b)
e2=next(b)
e3=next(b)
print(e1,e2,e3)

e=enumerate(A)
t1=next(e)
t2=next(e)
t3=next(e)
print(t1,t2,t3)
```

Output

```
Reading using for loop
```

50 20 40 10 30
Reading using iterator
50 20 40
(0, 50) (1, 20) (2, 40)
(99, 50) (100, 20)

Mutable Operations of Set

1. add()
2. remove()
3. discard()
4. clear()
5. pop()

add()

This method is used for adding element/value within set

Syntax: set-name.add(element)

Example:

```
>>> A=set()
>>> A.add(10)
>>> A.add(20)
>>> A.add(30)
>>> A.add(40)
>>> A.add(50)
>>> print(A)
{40, 10, 50, 20, 30}
```

Example:

```
# Write a program to add n players names into set

P=set()
n=int(input("How many players "))

# adding
for i in range(n):
    name=input("PlayerName :")
```

```
    P.add(name)
# reading and printing
for name in P:
    print(name)

print(P)
```

Output

```
How many players 2
PlayerName :virat
PlayerName :rohit
virat
rohit
{'virat', 'rohit'}
```

https://www.hackerrank.com/challenges/py-set-add/problem?isFullScreen=false

```
n=int(input())
A=set()
for i in range(n):
    country=input()
    A.add(country)

print(len(A))
```

https://www.hackerrank.com/challenges/py-introduction-to-sets/problem?isFullScreen=false

```
n=int(input())
A=list(map(int,input().split()))
B=set(A)
avg=sum(B)/len(B)
print(round(avg,3))
```

Removing or deleting elements from set

Set provides various methods for removing or deleting elements

1. remove()
2. discard()
3. pop()
4. clear()

remove() method

This method remove an element from set if exists else raises KeyError.

Syntax: set-name.remove(element)

```
>>> A={10,20,30,40,50}
>>> print(A)
{50, 20, 40, 10, 30}
>>> A.remove(10)
>>> print(A)
{50, 20, 40, 30}
>>> A.remove(50)
>>> print(A)
{20, 40, 30}
>>> A.remove(60)
Traceback (most recent call last):
  File "<pyshell#39>", line 1, in <module>
    A.remove(60)
KeyError: 60
```

Example:

```
A={10,20,30,40,50,60,70,80,90,100}
print(f'Before Removing {A}')
value=int(input("Value/element to remove "))
if value in A:
    A.remove(value)
    print(f'After Removing {A}')
else:
    print(f'{value} not exists in set')
```

Output

```
Before Removing {100, 70, 40, 10, 80, 50, 20, 90, 60, 30}
Value/element to remove 40
After Removing {100, 70, 10, 80, 50, 20, 90, 60, 30}

Before Removing {100, 70, 40, 10, 80, 50, 20, 90, 60, 30}
Value/element to remove 200
200 not exists in set
```

discard()

This method of set remove an element/value from set if exists
If element not exists, this method does not raises KeyError

Syntax: set-name.discard(value)

```
>>> A=set(range(10,110,10))
>>> print(A)
{100, 70, 40, 10, 80, 50, 20, 90, 60, 30}
>>> A.discard(100)
>>> print(A)
{70, 40, 10, 80, 50, 20, 90, 60, 30}
>>> A.discard(50)
>>> print(A)
{70, 40, 10, 80, 20, 90, 60, 30}
>>> A.discard(100)
>>> A.remove(100)
Traceback (most recent call last):
  File "<pyshell#47>", line 1, in <module>
    A.remove(100)
KeyError: 100
```

pop()

This method performs 2 operations

1. read
2. remove

Syntax: variable-name=set-name.pop()

```
>>> A={10,20,30,40,50,60,70}
>>> print(A)
{50, 20, 70, 40, 10, 60, 30}
>>> value1=A.pop()
>>> print(value1)
50
>>> print(A)
{20, 70, 40, 10, 60, 30}
>>> value2=A.pop()
>>> print(value2)
20
>>> print(A)
{70, 40, 10, 60, 30}
```

clear()
This method is used to remove all the elements from set (OR) empty set

Syntax: set-name.clear()

```
>>> A=set(range(10,110,10))
>>> print(A)
{100, 70, 40, 10, 80, 50, 20, 90, 60, 30}
>>> A.clear()
>>> print(A)
set()
```

https://www.hackerrank.com/challenges/py-set-discard-remove-pop/problem?isFullScreen=false

```
n = int(input())
s = set(map(int, input().split()))
N=int(input())
for i in range(N):
    cmd=input().split()
    if cmd[0]=="pop":
        s.pop()
```

```
    elif cmd[0]=="remove":
        s.remove(int(cmd[1]))
    elif cmd[0]=="discard":
        s.discard(int(cmd[1]))

print(sum(s))
```