```
#SET

s={}
s

{}

type(s)

dict

s1=set()
type(s1)

set

s1

set()

s2={63,8,45,69,12}
s2

{8, 12, 45, 63, 69}

s3={'x','f','y','t','g'}
s3

{'f', 'g', 't', 'x', 'y'}

s4 = {1, 2.3, 'nit', 1+2j, [1,2,3], (4,5,6), True}
s4 #SET CAN CONTAIN ONLY immutable ELE SO LIST[] ARE NOT ALLOWED IN
SET

---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[20], line 1
----> 1 s4 = {1, 2.3, 'nit', 1+2j, [1,2,3], (4,5,6), True}
      2 s4

TypeError: unhashable type: 'list'

s5={6,4,3.6,(77,6.3),'Mru',True}
s5

{(77, 6.3), 3.6, 4, 6, 'Mru', True}

print(s1)
print(s2)
print(s3)
print(s5)
```

```
set()
{69, 8, 12, 45, 63}
{'f', 'x', 'y', 't', 'g'}
{True, 3.6, 4, 6, 'Mru', (77, 6.3)}

s2

{8, 12, 45, 63, 69}

s2.add(20) #it will add ele to the set where it fits in ascending
order
s2

{8, 12, 20, 45, 63, 69, 200}

s2.add(200)
s2

{8, 12, 20, 45, 63, 69, 200}

s2[:] #u cannot access the ele from the set by slicing or indexing

---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[47], line 1
----> 1 s2[:]

TypeError: 'set' object is not subscriptable

s2[2:3]

---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[49], line 1
----> 1 s2[2:3]

TypeError: 'set' object is not subscriptable

s5

{(77, 6.3), 3.6, 4, 6, 'Mru', True}

s4=s5.copy() #it will copy all ele from s5 to s4
s4

{(77, 6.3), 3.6, 4, 6, 'Mru', True}

s5.clear() #it will clear all ele from set
```

```
s5
```

```
set()
```

```
del s5 #it will delete whole s5 set
```

```
s5
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[65], line 1
----> 1 s5

NameError: name 's5' is not defined
```

```
s4
```

```
{(77, 6.3), 3.6, 4, 6, 'Mru', True}
```

```
s4.add((77,6.3))
```

```
s4
```

```
{(77, 6.3), 3.6, 4, 6, 'Mru', True}
```

```
s4.remove((77,6.3)) #IT WILL REMOVE THE GIVEN ELE
```

```
s4
```

```
{3.6, 4, 6, 'Mru', True}
```

```
s3
```

```
{'f', 'g', 't', 'x', 'y'}
```

```
s3.discard('g') #if ele is prenet then it ges discarded but if not
present then it will not show any error and set will be unchanged
```

```
s3
```

```
{'f', 't', 'x', 'y'}
```

```
s3.remove('y') #if the ele is not present then remove() will throw an
error
```

```
s3
```

```
{'f', 't'}
```

```
s6={'f', 'g', 't', 'x', 'y','s','w','u','p'}
s6
```

```
{'f', 'g', 'p', 's', 't', 'u', 'w', 'x', 'y'}
```

```
s6.pop()
```

```
'f'
```

```
s6
```

```
{'g', 'p', 's', 't', 'u', 'w', 'x', 'y'}
```

```
s6.pop()
```

```
'p'
```

```
s6
```

```
{'g', 's', 't', 'u', 'w', 'x', 'y'}
```

```
s6.pop()
```

```
'y'
```

```
for i in s2:
    print(i)
```

```
200
20
69
8
12
45
63
```

```
for i in enumerate(s2):
    print(i)
```

```
(0, 200)
(1, 20)
(2, 69)
(3, 8)
(4, 12)
(5, 45)
(6, 63)
```

```
s2
```

```
{8, 12, 20, 45, 63, 69, 200}
```

```
5 in s2
```

```
False
```

```
12 in s2
```

```
True
```

```
s2.update(s3) #concatinate and give only 1 appereance of same ele if present
s2
```

```
{12, 20, 200, 45, 63, 69, 8, 'f', 't'}
```

```
new1={1,2,3,4,5}
new2={4,5,6,7,8,}
new3={8,9,10}
```

```
new1.union(new2) # returns the union of new1 and new2 excluding same ele
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

```
new2.union(new3)
```

```
{4, 5, 6, 7, 8, 9, 10}
```

```
new1|new2 # it is a pipe operator which also means union of 2 sets
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

```
new1|new2|new3
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

```
print(new1)
print(new2)
print(new3)
```

```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

```
new1.intersection(new2) #IT WILL RETURN THE ELE WHICH ARE same ele IN BOTH SETS
```

```
{4, 5}
```

```
new1.intersection(new3)
```

```
set()
```

```
new1 & new2 # & also works as intersection
```

```
{4, 5}
```

```
new1 & new2 & new3
```

```
set()
```

```
print(new1)
print(new2)
print(new3)
```

```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

```
new1.difference(new2) #it returns the ele which are not same in both sets
```

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

```
new1 - new2 # - this sign also works as difference
```

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

```
new1-new3
```

```
{1, 2, 3, 4, 5}
```

```
print(new1)
print(new2)
print(new3)
```

```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

```
new1.symmetric_difference(new2) #it returns the ele excluding same ele in both sets
```

```
{1, 2, 3, 6, 7, 8}
```