7th March set

```
In [62]: s = {}
         s
```

```
Out[62]: {}
```

```
In [63]: type(s)
```

```
Out[63]: dict
```

```
In [64]: s1=set() # defining a set
         type(s1)
```

```
Out[64]: set
```

```
In [65]: s2 = {10,40,20,70,15,100} # sorted (if it contains similar data types)
         s2
```

```
Out[65]: {10, 15, 20, 40, 70, 100}
```

```
In [66]: s3 = {'d','a','z','c','h'}
         s3
```

```
Out[66]: {'a', 'c', 'd', 'h', 'z'}
```

```
In [67]: s4 = {5,'nit',(1+2j),2.5,[1,3,4,5],True,False} # lists can't go in sets directly
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[67], line 1
----> 1 s4 = {5,'nit',(1+2j),2.5,[1,3,4,5],True,False}

TypeError: unhashable type: 'list'
```

```
In [68]: s5 = {2,3.5,'nit',(1+2j),True,False}
         s5
```

```
Out[68]: {(1+2j), 2, 3.5, False, True, 'nit'}
```

```
In [69]: print(s1)
         print(s2)
         print(s3)
         print(s5)
```

```
set()
{100, 20, 70, 40, 10, 15}
{'c', 'z', 'h', 'd', 'a'}
{False, True, 'nit', 2, 3.5, (1+2j)}
```

```
In [70]: s2.add(15)
```

```
In [71]: s2
```

```
Out[71]: {10, 15, 20, 40, 70, 100}
```

```
In [72]: s2.add(200)
```

```
In [73]: s2
```

```
Out[73]: {10, 15, 20, 40, 70, 100, 200}
```

```
In [74]: s2[:] # indexing and slicing is not allowed in set
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[74], line 1
----> 1 s2[:]

TypeError: 'set' object is not subscriptable
```

```
In [75]: s4 = s5.copy() # copy the set
```

```
In [76]: s4
```

```
Out[76]: {(1+2j), 2, 3.5, False, True, 'nit'}
```

```
In [77]: s5
```

```
Out[77]: {(1+2j), 2, 3.5, False, True, 'nit'}
```

```
In [78]: s4.add(10) # add the elements
```

```
In [79]: s4
```

```
Out[79]: {(1+2j), 10, 2, 3.5, False, True, 'nit'}
```

```
In [82]: s4.add (10) # duplication is not allowed in set
         s4
```

```
Out[82]: {(1+2j), 10, 2, 3.5, False, True, 'nit'}
```

```
In [83]: s5.clear() #clear the set (it will become empty set)
```

```
In [84]: s5
```

```
Out[84]: set()
```

```
In [85]: del s5 # delete the set
```

```
In [86]: s4
```

```
Out[86]:  {(1+2j), 10, 2, 3.5, False, True, 'nit'}
```

```
In [87]:  s4.remove(3.5) # removes the element
          s4
```

```
Out[87]:  {(1+2j), 10, 2, False, True, 'nit'}
```

```
In [88]:  s3
```

```
Out[88]:  {'a', 'c', 'd', 'h', 'z'}
```

```
In [89]:  s3.discard ('a') # it will remove the element if it is a member,if not a member no
```

```
In [90]:  s3
```

```
Out[90]:  {'c', 'd', 'h', 'z'}
```

```
In [91]:  s3.discard('a') # never gives a error
```

```
In [92]:  s3.add ('a')
          s3
```

```
Out[92]:  {'a', 'c', 'd', 'h', 'z'}
```

```
In [93]:  s3.pop() #randomly elements are deleted
```

```
Out[93]:  'c'
```

```
In [94]:  s3
```

```
Out[94]:  {'a', 'd', 'h', 'z'}
```

```
In [95]:  s3.pop(1) # pop takes no arguments
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[95], line 1
----> 1 s3.pop(1)

TypeError: set.pop() takes no arguments (1 given)
```

```
In [96]:  s3.pop('a')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[96], line 1
----> 1 s3.pop('a')

TypeError: set.pop() takes no arguments (1 given)
```

```
In [97]:  s2
```

```
Out[97]:  {10, 15, 20, 40, 70, 100, 200}
```

In [98]:
```
for i in s2:
    print(i)
```

```
100
20
70
40
10
200
15
```

In [101…
```
for i in enumerate(s2):
    print(i)
```

```
(0, 100)
(1, 20)
(2, 70)
(3, 40)
(4, 10)
(5, 200)
(6, 15)
```

In [102…
```
for i in enumerate (s3):
    print(i)
```

```
(0, 'z')
(1, 'h')
(2, 'd')
(3, 'a')
```

In [103…
```
0 in s3
```

Out[103…
```
False
```

In [104…
```
20 in s2
```

Out[104…
```
True
```

In [107…
```
s2.update (s3) #like extend() in list (s3 will be added to s2)
```

In [108…
```
s2
```

Out[108…
```
{10, 100, 15, 20, 200, 40, 70, 'a', 'd', 'h', 'z'}
```

In [ ]:
```

```

Set operations

In [111…
```
# SET OPERATION -->
#UNION - | .union()
#INTERSECTION  & .intersection()
#DISJOINT
```

```
#DIFFERENCE - .difference()
#SYMMETRIC DIFFERENCE
```

Union ---> union() or |

In [112…

```
s6 = {1,2,3,4,5}
s7 = {4,5,6,7,8}
s8 = {8,9,10}
```

In [116…

```
s6.union(s7) # used for combining sets
```

Out[116…

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

In [117…

```
s7 | s8 |s6
```

Out[117…

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

Intersection ---> intersection() or &

In [119…

```
print(s6)
print(s7)
print(s8)
```

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

In [128…

```
s6.intersection(s7) # used to find the common sets
```

Out[128…

```
{4, 5}
```

In [129…

```
s6 & s7 &s8
```

Out[129…

```
set()
```

In [130…

```
s7 & s8
```

Out[130…

```
{8}
```

Difference ---> difference() or -

In [132…

```
print(s6)
print(s7)
print(s8)
```

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

```
In [134…  s6.difference(s7) # used to find what is only in the first set and not in the secon

Out[134…  {1, 2, 3}

In [127…  s7.difference(s6)

Out[127…  {6, 7, 8}

In [135…  s8 - s7

Out[135…  {9, 10}
```

Symmetric difference --> symmetric_difference()

```
In [136…  print(s6)
          print(s7)
          print(s8)

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

In [138…  s6.symmetric_difference(s7) #  used to find the elements that are in either of the

Out[138…  {1, 2, 3, 6, 7, 8}

In [139…  s8.symmetric_difference(s7)

Out[139…  {4, 5, 6, 7, 9, 10}

In [ ]:

In [ ]:
```