SET

In [ ]:
```
# Set is a python datastructure defined with curly braces {} but empty set is defin
# Set is collection of items enclosed in curly braces {}
# Set doesnt allow Duplicates
# In general we say set is unorderd collection of elements but we cant define wheth
# We cant define because behaviour of datastructure differs several times
# SO, better not to mention
# set has several methods
# Here we go
```

Defining SET

In [2]:
```
s = {}
type(s)
```

Out[2]:
```
dict
```

In [4]:
```
st = set()
st
```

Out[4]:
```
set()
```

In [5]:
```
type(st)
```

Out[5]:
```
set
```

In [11]:
```
#set of integers
#in this case it prints random numbers in sorted order
s1 = {1,20,23,45,90,12,15,84}
s1
```

Out[11]:
```
{1, 12, 15, 20, 23, 45, 84, 90}
```

In [9]:
```
# if we use print() it prints in random order
print(s1)
```

```
{1, 12, 45, 15, 20, 84, 23, 90}
{1, 12, 45, 15, 20, 84, 23, 90}
{1, 12, 45, 15, 20, 84, 23, 90}
{1, 12, 45, 15, 20, 84, 23, 90}
```

In [12]:
```
#set of float numbers
s2 = {1.2,22.4,12.2,4.5,12.2}# we can observe we defined 12.2 two times but it prin
s2
```

Out[12]:
```
{1.2, 4.5, 12.2, 22.4}
```

In [ ]:
```
#set of strings
```

In [13]:
```
s3 = {'z','a','v','e','t','g'}
s3#sorted
```

Out[13]:
```
{'a', 'e', 'g', 't', 'v', 'z'}
```

In [15]:
```
print(s3)#random
```

```
{'a', 'z', 'e', 'g', 'v', 't'}
```

In [16]:
```
#set of boolean
s4 = {'True','False',0,1}
s4
```

Out[16]:
```
{0, 1, 'False', 'True'}
```

In [17]:
```
s4 = {True,False,0,1}#it treats 0 as false and 1 as true so wont repeats
s4
```

Out[17]:
```
{False, True}
```

In [18]:
```
s5 = {True,False,0}
```

In [19]:
```
s5
```

Out[19]:
```
{False, True}
```

In [21]:
```
s5 = {1,False,0}
s5
```

Out[21]:
```
{False, 1}
```

In [22]:
```
s5 = {1,True,0}
s5
```

Out[22]:
```
{0, 1}
```

In [25]:
```
# set of complex
s6 = {4+2j,1+2j,2+3j}
s6
```

Out[25]:
```
{(1+2j), (2+3j), (4+2j)}
```

In [24]:
```
s6 = {1+2j,2+3j}
s6
```

Out[24]:
```
{(1+2j), (2+3j)}
```

In [26]:
```
s7 = {1,2.3,'sri',True,1+2j}#true will treats as 1
s7
```

Out[26]: {(1+2j), 1, 2.3, 'sri'}

In [27]:
```
s7 = {4,2.3,'sri',True,1+2j}
s7
```

Out[27]: {(1+2j), 2.3, 4, True, 'sri'}

In [ ]:
```
s8 = {1}
s8.add(2)
s8
```

add()

In [1]:
```
# add() is used to add elements in to set and it adds at any position of set
s9 = set()
s9
```

Out[1]: set()

In [2]:
```
s9.add(1)
```

In [3]:
```
s9
```

Out[3]: {1}

In [5]:
```
s9.add(2)
s9.add(3)
s9.add(4)
```

In [6]:
```
s9
```

Out[6]: {1, 2, 3, 4}

clear()

In [7]:
```
# clear() is used to remove all elements from set but set remain in memory
s9.clear()
```

In [8]:
```
s9
```

Out[8]: set()

copy()

In [18]:
```
s10 = s1.copy()
```

In [19]:
```
s10
```

Out[19]:
```
{1, 12, 15, 20, 23, 45, 84, 90}
```

pop()

In [20]:
```
# pop() is used to remove and return a random number
# unlike in list it wont use index or pop last element
s10
```

Out[20]:
```
{1, 12, 15, 20, 23, 45, 84, 90}
```

In [21]:
```
s10.pop()
```

Out[21]:
```
1
```

In [26]:
```
s10.pop()
```

Out[26]:
```
84
```

In [23]:
```
s10.pop()
```

Out[23]:
```
45
```

In [27]:
```
s10.pop()
```

Out[27]:
```
23
```

In [28]:
```
s10.pop()
```

Out[28]:
```
90
```

In [30]:
```
s10.pop()# it gives error as indexing is not allowed in set
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[30], line 1
----> 1 s10.pop()

KeyError: 'pop from an empty set'
```

remove()

In [36]:
```
s10
```

Out[36]:
```
set()
```

In [ ]:
```
# remove() is used to remove particular element from set
```

In [37]:
```
s11 = {1,34,2,5,6,23,76,82}
s11
```

Out[37]:
```
{1, 2, 5, 6, 23, 34, 76, 82}
```

In [38]:
```
s11.remove(6)
```

In [39]:
```
s11
```

Out[39]:
```
{1, 2, 5, 23, 34, 76, 82}
```

In [40]:
```
s11.remove(100)# as it is not member in set it gives error
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[40], line 1
----> 1 s11.remove(100)

KeyError: 100
```

discrad()

In [41]:
```
# discard() remove element if element is member,if not it doesnt give error
s11.discard(100)
```

In [42]:
```
s11.discard(76)
```

In [43]:
```
s11
```

Out[43]:
```
{1, 2, 5, 23, 34, 82}
```

In [44]:
```
100 in s11
```

Out[44]:
```
False
```

In [45]:
```
1 in s11
```

Out[45]:
```
True
```

set operation

union

In [ ]:
```
# union
# intersection
# difference
```

```
# difference_update
#
```

In [ ]:
```
# union return unique elements from all sets
# it is defined using pipe symbol (|)
```

In [47]:
```
n = {1,2,3,4,5}
n1 = {6,7,8,9,10}
n2 = {5,6,4,7,3}
```

In [48]:
```
n.union(n1)
```

Out[48]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [49]:
```
n.union(n1,n2)# as it dont allow duplicates it prints single time
#even union also prints only unique items
```

Out[49]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [55]:
```
n | n1
```

Out[55]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [56]:
```
n | n1 | n2
```

Out[56]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

intersection

In [ ]:
```
# intersection returns common elements
# it is defined with & and symbol
```

In [51]:
```
# it prints common elements in 3 sets
n.intersection(n2)
```

Out[51]: {3, 4, 5}

In [52]:
```
n1.intersection(n2)
```

Out[52]: {6, 7}

In [53]:
```
n.intersection(n1,n2)
```

Out[53]: set()

In [57]:
```
n&n1
```

Out[57]: set()

In [58]:
```
n&n2
```

Out[58]: {3, 4, 5}

In [59]:
```
n1&n2
```

Out[59]: {6, 7}

In [61]:
```
n&n1&n2
```

Out[61]: set()

difference

In [69]:
```
# it removes common from n
# it is defined using - minus symbol
n.difference(n1)
```

Out[69]: {1, 2, 3, 4, 5}

In [70]:
```
n.difference(n2)
```

Out[70]: {1, 2}

In [63]:
```
n1.difference(n2)
```

Out[63]: {8, 9, 10}

In [64]:
```
n - n1
```

Out[64]: {1, 2, 3, 4, 5}

In [65]:
```
n-n2
```

Out[65]: {1, 2}

In [66]:
```
n1-n2
```

Out[66]: {8, 9, 10}

In [67]:
```
n-n2
```

Out[67]: {1, 2}

difference_update

In [71]:
```
n.difference_update(n2)# it removes and update
```

In [75]:
```
n
```

Out[75]: {1, 2, 3, 4, 5}

In [76]:
```
n2
```

Out[76]: {3, 4, 5, 6, 7}

symmetric_difference

In [77]:
```
n.symmetric_difference(n2)# removes common
```

Out[77]: {1, 2, 6, 7}

In [78]:
```
n1.symmetric_difference(n2)
```

Out[78]: {3, 4, 5, 8, 9, 10}

superset

In [80]:
```
x = {1,2,3,4,5,6,7,8,9}
x1 = {3,4,5,6,7,8}
x2 = {15,16,14,17,13}
```

In [81]:
```
x.issuperset(x1)
```

Out[81]: True

In [82]:
```
x.issuperset(x2)
```

Out[82]: False

subset

In [83]:
```
x.issubset(x1)
```

Out[83]: False

In [87]:
```
x1.issubset(x2)
```

Out[87]: False

In [85]:
```
x1.issubset(x)
```

Out[85]: True

In [86]:
```
x2.issubset(x)
```

Out[86]: False

disjoint

In [88]:
```
x.isdisjoint(x1)
```

Out[88]: False

In [89]:
```
x.isdisjoint(x2)
```

Out[89]: True

In [90]:
```
x1.isdisjoint(x2)
```

Out[90]: True

In [91]:
```
x2.isdisjoint(x1)
```

Out[91]: True

In [92]:
```
x2.isdisjoint(x)
```

Out[92]: True

In [93]:
```
x1.isdisjoint(x)
```

Out[93]: False

set dont allow indexing and slicing

In [ ]:
```

```