unit - 10

sets in python

In [1]:
```
S = {12,13,14,15}
print(S)
print(type(S))
```

```
{12, 13, 14, 15}
<class 'set'>
```

In [2]:
```
S = {12,13,14,15}
print(S)
```

```
{12, 13, 14, 15}
```

In [3]:
```
S = {12,14,16,-36,-2,3.69,4.2,"code","hello","datascience"}
print(S)
print(type(S))
```

```
{3.69, 4.2, 12, 14, 16, 'datascience', 'code', 'hello', -36, -2}
<class 'set'>
```

unordered

In [4]:
```
S = {25,24,12,-3,-9,2.36,"code","learning"}
print(s[0])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16004\1573328520.py in <module>
      1 S = {25,24,12,-3,-9,2.36,"code","learning"}
----> 2 print(s[0])

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

duplicates not allowed

In [5]:
```
m = {12,14,78,23,14,15,14}
print(m)
```

```
{23, 78, 12, 14, 15}
```

In [6]:
```
m = {12,14,78,23,14,15,14,2.36,5,2.36,"hello","boat","hello","apple"}
print(m)
```

```
{2.36, 'apple', 5, 12, 78, 15, 14, 'boat', 23, 'hello'}
```

get the length of a set

In [7]:
```
a = {12,14,15,17,32,65}
print(len(a))
```

```
6
```

In [8]:
```
a = {True, False, True,True}
print(a)
```

```
{False, True}
```

In [9]:
```
a = {12,14,16,98,31,64}
print(a)
```

```
{64, 16, 98, 12, 14, 31}
```

access items

In [10]:
```
s = {23,25,27,29,15.36,7.15,"apple","orange"}
for i in s:
    print(i)
```

```
apple
7.15
15.36
orange
23
25
27
29
```

In [11]:
```
""" check if "apple" is in the set or not ?"""
s = {23,25,27,29,15.36,7.15,"apple","orange"}
print("apple" in s)
```

```
True
```

In [12]:
```
""" check if "apple" is in the set or not ?"""
s = {23,25,27,29,15.36,7.15,"apple","orange"}
print(23 in s)
```

```
True
```

In [13]:
```
""" check if "apple" is in the set or not ?"""
s = {23,25,27,29,15.36,7.15,"apple","orange"}
print("bat" in s)
```

```
False
```

add items

In [14]:
```
a = {23,24,17,19,"red","blue","green"}
a.add("orange")
print(a)
```

```
{17, 19, 'red', 23, 24, 'orange', 'blue', 'green'}
```

In [15]:
```
a = {23,24,17,19,"red","blue","green"}
a.add(2.36)
print(a)
```

```
{17, 2.36, 19, 'red', 23, 24, 'blue', 'green'}
```

In [16]:
```
a = {23,24,17,19,"red","blue","green"}
a.add(3,5)
print(a)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16004\3766628686.py in <module>
      1 a = {23,24,17,19,"red","blue","green"}
----> 2 a.add(3,5)
      3 print(a)

TypeError: set.add() takes exactly one argument (2 given)
```

update() method

In [17]:
```
a = {12,14,15,-36,-9}
b = {"red","blue","pink"}
a.update(b)
print(a)
```

```
{'blue', 'pink', 'red', -9, -36, 12, 14, 15}
```

In [20]:
```
a = {12,14,15,-36,-9}
b = {"red","blue", 2.36,0.23}
b.update(a)
print(b)
print(a)
```

```
{0.23, 2.36, -36, 12, 14, 15, 'red', -9, 'blue'}
{-9, -36, 12, 14, 15}
```

remove item

In [21]:
```
a = {23,28,34,15,3,2.36,9,-8,-2}
a.remove(2.36)
print(a)
```

```
{34, 3, 9, 15, 23, -8, 28, -2}
```

In [22]:
```
a = {23,28,34,15,3,2.36,9,-8,-2,"north","south","east","west"}
a.remove("south")
print(a)
```

```
{34, 3, 2.36, 'west', -8, 9, 15, 'east', 23, 'north', 28, -2}
```

discard() method

In [23]:
```
a = {23,28,34,15,3,2.36,9,-8,-2,"north","south","east","west"}
a.discard("south")
print(a)
```

```
{34, 3, 2.36, 'west', 'north', 9, 15, 'east', 23, -8, 28, -2}
```

In [24]:
```
a = {23,28,34,15,3,2.36,9,-8,-2,"north","south","east","west"}
a.discard(2.36)
print(a)
```

```
{'south', 34, 3, 'west', 'north', 9, 15, 'east', 23, -8, 28, -2}
```

In [25]:
```
a = {23,28,34,15,3,2.36,9,-8,-2,"north","south","east","west"}
a.discard("south", "east")
print(a)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16004\3487421547.py in <module>
      1 a = {23,28,34,15,3,2.36,9,-8,-2,"north","south","east","west"}
----> 2 a.discard("south", "east")
      3 print(a)

TypeError: set.discard() takes exactly one argument (2 given)
```

In [26]:
```
a = {23,28,34,15,3,2.36,9,-8,-2,"north","south","east","west"}
a.remove("south", "east")
print(a)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16004\734654581.py in <module>
      1 a = {23,28,34,15,3,2.36,9,-8,-2,"north","south","east","west"}
----> 2 a.remove("south", "east")
      3 print(a)

TypeError: set.remove() takes exactly one argument (2 given)
```

pop() method

In [29]:
```
a = {12,34,56,78,90,2.36,-8,"hello","world"}
a.pop()
print(a)
```

```
{2.36, 56, -8, 12, 78, 'hello', 'world', 90}
```

In [31]:
```
a = {12,34,56,78,90,2.36,-8,"hello","world", "red", 5.69,7.8}
a.pop()
print(a)
```

```
{2.36, 56, 5.69, 7.8, -8, 12, 78, 'red', 'hello', 'world', 90}
```

clear() method

In [33]:
```
a = {23,14,15,0,-8,-2,"red","data","ball"}
a.clear()
print(a)
print(type(a))
```

```
set()
<class 'set'>
```

del() method

In [34]:
```
a = {25,14,7,2,3}
del(a)
print(a)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16004\1662306370.py in <module>
      1 a = {25,14,7,2,3}
      2 del(a)
----> 3 print(a)

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

python - join sets

In [36]:
```
a = {12,14,16,18}
b = {5,6,7,8}
c = (a+b)
print(c)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16004\433315105.py in <module>
      1 a = {12,14,16,18}
      2 b = {5,6,7,8}
----> 3 c = (a+b)
      4 print(c)

TypeError: unsupported operand type(s) for +: 'set' and 'set'
```

union method

In [37]:
```
a = {12,14,16,18}
b = {5,6,7,8}
c = a.union(b)
print(a)
print(b)
print(c)
```

```
{16, 18, 12, 14}
{8, 5, 6, 7}
{5, 6, 7, 8, 12, 14, 16, 18}
```

In [38]:
```
a = {12,14,16,18,2.36}
b = {5,6,7,8,"hello","python"}
c = a.union(b)
print(a)
print(b)
print(c)
```

```
{16, 2.36, 18, 12, 14}
{5, 6, 7, 8, 'hello', 'python'}
{2.36, 5, 6, 7, 8, 12, 14, 16, 18, 'hello', 'python'}
```

difference between union and update method

In [39]:
```
""" update method"""
a = {1,2,3}
b={4,5,6}
a.update(b)
print(a)
```

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

In [40]:
```
a = {1,2,3}
b={4,5,6}
c = a.update(b)
print(c)
```

```
None
```

In [41]:
```
a = {1,2,3}
b={4,5,6}
c = a.union(b)
print(c)
```

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

intersection_update() method

In [43]:
```
a = {23,24,25,26,"apple","cherry"}
b = {18,24,23,"cherry"}
a.intersection_update(b)
print(a)
```

```
{24, 'cherry', 23}
```

In [45]:
```
a = {23,24,25,26,"apple","cherry"}
b = {18,24,23,"cherry"}
b.intersection_update(a)
print(b)
```

```
{24, 'cherry', 23}
```

In [48]:
```
a = {23,24,25,26,"apple","cherry"}
b = {18,24,23,"cherry"}
c = b.intersection_update(a)
print(c)
```

```
None
```

intersection() method

In [49]:
```
a = {23,24,25,26,"apple","cherry"}
b = {18,24,23,"cherry"}
c = b.intersection(a)
print(c)
```

```
{24, 'cherry', 23}
```

In [51]:
```
m = {2.36,2.3,4.5,5.6,6.9}
n = {2.36,5.6,1.2}
r = m.intersection(n)
print(r)
```

```
{2.36, 5.6}
```

difference() method

In [52]:
```
""" return a set that contains the items that only exist in set x not in set y
x = {12,13,14,15,16,"delhi", "bombay","chennai"}
y = {10,11,12,14,"delhi"}
c = x.difference(y)
print(c)
```

```
{13, 'bombay', 15, 16, 'chennai'}
```

In [53]:
```
x = {12,13,14,15,16,"delhi", "bombay","chennai"}
y = {10,11,12,14,"delhi"}
c = y.difference(x)
print(c)
```

```
{10, 11}
```

difference_update() method

In [54]:
```
""" remove the items that exist in both sets"""
a = {17,18,19,20,"hello","coding","class"}
b = {17,"coding"}
a.difference_update(b)
print(a)
```

```
{18, 19, 20, 'hello', 'class'}
```

In [55]:
```
a = {17,18,19,20,"hello","coding","class"}
b = {17,"coding"}
b.difference_update(a)
print(a)
```

```
{17, 18, 19, 20, 'hello', 'coding', 'class'}
```

symmetric_difference_update() method

The symmetric_difference_update() method will keep only the elements that are NOT present in both sets.

In [56]:
```
a = {23,24,25,26,"apple","cherry"}
b = {18,19,23,93,"cherry"}
a.symmetric_difference_update(b)
print(a)
```

```
{'apple', 18, 19, 24, 25, 26, 93}
```

In [57]:
```
a = {23,24,25,26,"apple","cherry"}
b = {18,19,23,93,"cherry"}
b.symmetric_difference_update(a)
print(b)
```

```
{'apple', 18, 19, 24, 25, 26, 93}
```

symmetric_difference() method

""" The symmetric_difference() method will return a new set, that contains only the elements that are NOT present in both sets. """

In [58]:
```
a = {23,24,25,26,"apple","cherry"}
b = {18,19,23,93,"cherry"}
c = a.symmetric_difference(b)
print(c)
```

```
{'apple', 18, 19, 24, 25, 26, 93}
```

In [59]:
```
a = {23,24,25,26,"apple","cherry"}
b = {18,19,23,93,"cherry"}
c = b.symmetric_difference(a)
print(c)
```

```
{'apple', 18, 19, 24, 25, 26, 93}
```

isdisjoint method

return True if no item in set x is present in set y

In [1]:
```
x = {12,14,1,6,18,"hello","class"}
y = {10,23,25,27}
z = x.isdisjoint(y)
print(z)
```

```
True
```

In [2]:
```
x = {12,14,1,6,18,"hello","class"}
y = {10,23,25,27}
z = y.isdisjoint(x)
print(z)
```

```
True
```

In [3]:
```
x = {12,14,1,6,18,"hello","class"}
y = {10,23,25,12}
z = y.isdisjoint(x)
print(z)
```

```
False
```

issubset() method

Returns whether another set contains this set or not

In [4]:
```
x = {1,2,3,4,5,6}
y = {10,12,13,14,1,2,3,4,5,6}
c = x.issubset(y)
print(c)
```

```
True
```

In [5]:
```
x = {1,2,3,4,5,6}
y = {10,12,13,14,1,2,3,4,5,6}
c = y.issubset(x)
print(c)
```

```
False
```

In [6]:
```
x = {1,2,3,4,5,6}
y = {1,2,3,4,5,6}
c = y.issubset(x)
print(c)
```

```
True
```

In [7]:
```
x = {1,2,3,4,5,6}
y = {1,2,3,4,5,6}
c = x.issubset(y)
print(c)
```

```
True
```

issuperset() method

Return True if all items set y are present in set x:

In [11]:
```
x = {"apple", "cherry","north","south","python",1,2,3}
y = {"apple",1,2,3,"north"}
z = x.issuperset(y)
print(z)
```

```
True
```

In [12]:
```
x = {"apple", "cherry","north","south","python",1,2,3}
y = {"apple",1,2,3,"north"}
z = y.issuperset(x)
print(z)
```

```
False
```

In [13]:
```
p = {1,2,3}
q ={1,2,3}
c = p.issuperset(q)
print(c)
```

```
True
```

In [14]:
```
p = {1,2,3}
q ={1,2,3}
c = q.issuperset(p)
print(c)
```

```
True
```

In [17]:
```
s = {3}
print(s)
print(type(s))
```

```
{3}
<class 'set'>
```

In [18]:
```
a = {}
print(a)
print(type(a))
```

```
{}
<class 'dict'>
```

In [19]:
```
a = set()
print(a)
print(type(a))
```

```
set()
<class 'set'>
```

set() function

In [20]:
```
L = [12,14,16,18]
S = set(L)
print(L)
print(type(L))
print(S)
print(type(S))
```

```
[12, 14, 16, 18]
<class 'list'>
{16, 18, 12, 14}
<class 'set'>
```

In [21]:
```
s= {12,18,16,45}
l = list(s)
print(s)
print(type(s))
print(l)
print(type(l))
```

```
{16, 18, 12, 45}
<class 'set'>
[16, 18, 12, 45]
<class 'list'>
```

In [22]:
```
t = (10,12,15,17)
s = set(t)
print(t)
print(type(t))
print(s)
print(type(s))
```

```
(10, 12, 15, 17)
<class 'tuple'>
{17, 10, 12, 15}
<class 'set'>
```

In [23]:
```
s = {12,14,17,19}
t = tuple(s)
print(s)
print(type(s))
print(t)
print(type(t))
```

```
{17, 19, 12, 14}
<class 'set'>
(17, 19, 12, 14)
<class 'tuple'>
```

min in set

In [24]:
```
p = {12,18,24,27}
print(min(p))
```

```
12
```

In [25]:
```
p = {"boy","cat","fan","python"}
print(min(p))
```

```
boy
```

max in set

In [26]:
```
p = {12,18,24,27}
print(max(p))
```

```
27
```

In [27]:
```
p = {"boy","cat","fan","python"}
print(max(p))
```

```
python
```

copy method

why you need copy method ?

In [28]:
```
a = {12,7,8,60,53,21}
b = a
a.add(10)
print(a)
print(b)
```

```
{21, 53, 7, 8, 10, 12, 60}
{21, 53, 7, 8, 10, 12, 60}
```

In [29]:
```
a = {12,7,8,60,53,21}
b = a.copy()
a.add(10)
print(a)
print(b)
```

```
{21, 53, 7, 8, 10, 12, 60}
{21, 53, 7, 8, 12, 60}
```

In [31]:
```
""" intersection of 2 lists """
a = [3,4,5,7,8,9]
b = [1,2,3,4,5]
a1 = set(a)
b1= set(b)
print(a1)
print(b1)
c = a1.intersection(b1)
print(c)
c = list(c)
print(c)
```

```
{3, 4, 5, 7, 8, 9}
{1, 2, 3, 4, 5}
{3, 4, 5}
[3, 4, 5]
```

In [32]:
```
""" taking input in set from user"""
a = set()
n = int(input("enter the number of items you want in set"))
for i in range(n):
    b = input("enter elements :")
    a.add(b)
print(a)
```

```
enter the number of items you want in set5
enter elements :40
enter elements :45
enter elements :50
enter elements :55
enter elements :60
{'55', '45', '40', '60', '50'}
```

In [33]:
```
""" taking input in set from user"""
a = set()
n = int(input("enter the number of items you want in set"))
for i in range(n):
    b = input("enter elements :")
    a.add(b)
print(a)
```

```
enter the number of items you want in set10
enter elements :12
enter elements :13
enter elements :14
enter elements :15
enter elements :16
enter elements :17
enter elements :18
enter elements :19
enter elements :20
enter elements :21
{'20', '18', '13', '15', '17', '16', '19', '14', '12', '21'}
```

In [39]:
```
"""python program to find the common items in 3 list using sets  """

ar1 = [1,5,10,20,40,80]
ar2 = [6,7,20,80,100]
ar3 = [3,4,15,20,30,70,80,120]
ar1 = set(ar1)
ar2 = set(ar2)
ar3 = set(ar3)
print(ar1)
print(ar2)
print(ar3)
n = ar1.intersection(ar2)
print(n)
m = n.intersection(ar3)
print(m)
print(list(m))
```

```
{1, 5, 40, 10, 80, 20}
{100, 6, 7, 80, 20}
{3, 4, 70, 15, 80, 20, 120, 30}
{80, 20}
{80, 20}
[80, 20]
```

In [ ]: