SET

set and dict defines with flower brackets{}

in python when you mention curly braces{} it willl by default returns as dict

if you create set () & then you worked on { } it become set now

if you define set {} all are in similar data types then it will return answer in order

duplicates are not allowed in set

indexing and slicing are not allowed in set

if we perform pop operation in set it will remove random numbers ¶

pop operation with argument doesnot work

set operations

add

copy

pop

clear

remove

discard

union

intersection

difference

symmetric difference

```
In [1]: s={}
        type(s)

Out[1]: dict
```

```
In [2]: s1=set()
        s1

Out[2]: set()
```

```
In [3]: s1={90,4,50,32,3,1} # if the elements in similar datatype it will give output in order
        s1

Out[3]: {1, 3, 4, 32, 50, 90}
```

```
In [4]: s2={'z','m','r','a'}
```

```
In [5]: type(s1)

Out[5]: set
```

```
In [6]: type(s2)

Out[6]: set
```

```
In [7]:  len(s1)

Out[7]:  6

In [8]:  s3={1,3.2,'nit',1+2j,True} # if you define differnent datatypes in set it will generates output in random not in order
         s3

Out[8]:  {(1+2j), 1, 3.2, 'nit'}

In [9]:  s1.add(1)

In [10]: s1    # duplicates are not allowed in set

Out[10]: {1, 3, 4, 32, 50, 90}

In [11]: s1

Out[11]: {1, 3, 4, 32, 50, 90}

In [12]: s1.add(5) # if you add random number into the set which is similar to the elements present in the set it will returns the out

In [13]: s1

Out[13]: {1, 3, 4, 5, 32, 50, 90}

In [14]: print(s1)

         {32, 1, 3, 4, 5, 50, 90}

In [15]: s3.clear()

In [16]: s3

Out[16]: set()

In [17]: s2

Out[17]: {'a', 'm', 'r', 'z'}

In [18]: s4=s1.copy()

In [19]: s4

Out[19]: {1, 3, 4, 5, 32, 50, 90}

In [20]: s1

Out[20]: {1, 3, 4, 5, 32, 50, 90}

In [22]: s1[0]

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-22-bfed54b371ac> in <module>
         ----> 1 s1[0]

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

index is not allowed in set

```
In [23]: s1[1:5]

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-23-605e5aea24fd> in <module>
         ----> 1 s1[1:5]

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

slice is not allowed in set

```
In [24]:  s1.pop()

Out[24]:  32

In [25]:  s1

Out[25]:  {1, 3, 4, 5, 50, 90}

In [26]:  s1.pop()

Out[26]:  1
```

random num bers are deleted from the set when we perform pop operation

```
In [27]:  s1.pop(0)

          ---------------------------------------------------------------------------
          TypeError                                 Traceback (most recent call last)
          <ipython-input-27-b92d0bacc3b8> in <module>
          ----> 1 s1.pop(0)

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

pop function with argrment doesnot works

```
In [28]:  s1.remove(4)

In [29]:  s1

Out[29]:  {3, 5, 50, 90}

In [30]:  s1

Out[30]:  {3, 5, 50, 90}

In [31]:  s1.remove()

          ---------------------------------------------------------------------------
          TypeError                                 Traceback (most recent call last)
          <ipython-input-31-2cda532a0446> in <module>
          ----> 1 s1.remove()

          TypeError: remove() takes exactly one argument (0 given)

In [32]:  s1.remove(1000)

          ---------------------------------------------------------------------------
          KeyError                                  Traceback (most recent call last)
          <ipython-input-32-87b930b55c5a> in <module>
          ----> 1 s1.remove(1000)

          KeyError: 1000
```

remove() removes the element if the element is member

```
In [33]:  s1.discard(1000)
```

discard () discard the element if the element is not a member of that set or member of that set

```
In [34]:  s1

Out[34]:  {3, 5, 50, 90}
```

```
In [35]: a={1,2,3,4,5}
         b={4,5,6,7,8}
         c={8,9,10}

In [36]: a.union(b)

Out[36]: {1, 2, 3, 4, 5, 6, 7, 8}

In [37]: a.union(b,c)

Out[37]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [38]: a|b

Out[38]: {1, 2, 3, 4, 5, 6, 7, 8}

In [39]: a|b|c

Out[39]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [40]: print(a)
         print(b)
         print(c)

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

In [41]: a.intersection(b)

Out[41]: {4, 5}

In [42]: a.intersection(c)

Out[42]: set()

In [43]: c

Out[43]: {8, 9, 10}

In [44]: a&b

Out[44]: {4, 5}

In [45]: print(a)
         print(b)
         print(c)

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

In [46]: a.difference(b)

Out[46]: {1, 2, 3}

In [47]: b-c

Out[47]: {4, 5, 6, 7}

In [48]: b.difference_update(c)

In [49]: b

Out[49]: {4, 5, 6, 7}

In [50]: a.symmetric_difference(b) # it will removes the common elements and returns remaining elements

Out[50]: {1, 2, 3, 6, 7}

In [ ]:
```

```
In [ ]:
```