1) List

2) Tuple

3) Dictionary

4) Sets ---> Frozen Set

- List

1) You do not require to define the size of the list.

2) It is mutable (you can update)

3) The indexing of list starts from 0

4) you can store different datatypes inside your list

In [ ]:
```
1
```

In [1]:
```
1 mylist = [100,200,300,400,500]
```

In [2]:
```
1 print(mylist)
```

```
[100, 200, 300, 400, 500]
```

In [3]:
```
1 print(type(mylist))
```

```
<class 'list'>
```

In [ ]:
```
1
```

In [4]:
```
1 hobbies = ["cricket","swimming","coding","sleeping"]
```

In [5]:
```
print(hobbies)
```

```
['cricket', 'swimming', 'coding', 'sleeping']
```

In [ ]:
```

```

In [6]:
```
mixed = ["demo",100,200,55.33,22.22,True,False,"demo"]
```

In [7]:
```
print(mixed)
```

```
['demo', 100, 200, 55.33, 22.22, True, False, 'demo']
```

In [8]:
```
mylist
```

Out[8]:
```
[100, 200, 300, 400, 500]
```

In [9]:
```
mylist.append("coding")
```

In [10]:
```
mylist
```

Out[10]:
```
[100, 200, 300, 400, 500, 'coding']
```

1) [100, 200, 300, 400, 500, 'coding',1000] 2) Error 3) None 4) None of above

In [12]:
```
print(mylist.append(1000))
```

```
None
```

In [13]:
```
mylist
```

Out[13]:
```
[100, 200, 300, 400, 500, 'coding', 1000]
```

In [ ]:
```

```

In [14]:
```
1 mylist
```
Out[14]: [100, 200, 300, 400, 500, 'coding', 1000]

In [15]:
```
1 mylist.insert(1,"hello")
```

In [16]:
```
1 mylist
```
Out[16]: [100, 'hello', 200, 300, 400, 500, 'coding', 1000]

In [17]:
```
1 mylist
```
Out[17]: [100, 'hello', 200, 300, 400, 500, 'coding', 1000]

In [18]:
```
1 mylist.clear()
```

In [19]:
```
1 mylist
```
Out[19]: []

In [20]:
```
1 mylist = [100, 'hello', 200, 300, 400, 500, 'coding', 1000]
```

In [21]:
```
1 mylist
```
Out[21]: [100, 'hello', 200, 300, 400, 500, 'coding', 1000]

In [22]:
```
1 mylist.pop()
```
Out[22]: 1000

In [23]:
```
1 mylist
```
Out[23]: [100, 'hello', 200, 300, 400, 500, 'coding']

In [24]:
```
1 mylist.pop(1)
```
Out[24]: 'hello'

In [25]:
```
1 mylist
```

Out[25]:
```
[100, 200, 300, 400, 500, 'coding']
```

In [26]:
```
1 mylist.remove('coding')
```

In [27]:
```
1 mylist
```

Out[27]:
```
[100, 200, 300, 400, 500]
```

In [29]:
```
1 mylist.remove('apple')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [29], in <module>
----> 1 mylist.remove('apple')

ValueError: list.remove(x): x not in list
```

In [31]:
```
1 mylist.remove(100,200)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [31], in <module>
----> 1 mylist.remove(100,200)

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

In [33]:
```
1 mylist
```

Out[33]:
```
[100, 200, 300, 400, 500]
```

In [32]:
```
1 mylist.count(100)
```

Out[32]:
```
1
```

In [34]:
```
1 mylist.index(400)
```

Out[34]: 3

In [35]:
```
1 mylist[4]
```

Out[35]: 500

In [ ]:
```
1
```

In [40]:
```
1 number = [33,12,300,50,2,44]
```

In [37]:
```
1 number
```

Out[37]: [33, 12, 300, 50, 2, 44]

In [38]:
```
1 number.sort()
```

In [39]:
```
1 print(number)
```

[2, 12, 33, 44, 50, 300]

In [41]:
```
1 number
```

Out[41]: [33, 12, 300, 50, 2, 44]

In [42]:
```
1 number.sort(reverse=True)
```

In [43]:
```
1 number
```

Out[43]: [300, 50, 44, 33, 12, 2]

In [ ]:
```
1
```

In [44]:
```
1 city = ["mumbai","pune","chennai","ahmedabad"]
```

In [45]:
```
1 city
```

Out[45]: ['mumbai', 'pune', 'chennai', 'ahmedabad']

In [46]:
```
1 city.sort()
```

In [47]:
```
1 city
```

Out[47]: ['ahmedabad', 'chennai', 'mumbai', 'pune']

In [48]:
```
1 city = ['pune','punjab','panji','peru']
```

In [49]:
```
1 city.sort()
```

In [50]:
```
1 city
```

Out[50]: ['panji', 'peru', 'pune', 'punjab']

In [ ]:
```
1
```

In [51]:
```
1 number = [33,12,300,50,2,44]
```

In [52]:
```
1 number.reverse()
```

In [53]:
```
1 number
```

Out[53]: [44, 2, 50, 300, 12, 33]

In [ ]:
```
1
```

In [1]:
```
1 mylist = [100,200,300,400,500]
```

In [2]:
```
1 mylist
```

Out[2]: [100, 200, 300, 400, 500]

In [3]:
```
1 copied_list = mylist.copy()
```

In [4]:
```
1 copied_list
```

Out[4]: [100, 200, 300, 400, 500]

In [5]:
```
1 copied_list.append('hello')
```

In [6]:
```
1 copied_list
```

Out[6]: [100, 200, 300, 400, 500, 'hello']

In [7]:
```
1 mylist
```

Out[7]: [100, 200, 300, 400, 500]

In [8]:
```
1 mylist.pop()
```

Out[8]: 500

In [9]:
```
1 mylist
```

Out[9]: [100, 200, 300, 400]

In [10]:
```
1 copied_list
```

Out[10]: [100, 200, 300, 400, 500, 'hello']

In [ ]:
```
1
```

In [11]:
```
1 mylist
```

Out[11]: [100, 200, 300, 400]

In [12]:
```
1 mycopy = mylist
```

In [13]:
```
1 mycopy
```

Out[13]: [100, 200, 300, 400]

In [14]:
```
1 mylist
```

Out[14]: [100, 200, 300, 400]

In [15]:
```
1 mycopy.append('test')
```

In [16]:
```
1 mycopy
```

Out[16]: [100, 200, 300, 400, 'test']

In [17]:
```
1 mylist
```

Out[17]: [100, 200, 300, 400, 'test']

In [18]:
```
1 mylist.remove(100)
```

In [19]:
```
1 mylist
```

Out[19]: [200, 300, 400, 'test']

In [20]:
```
1 mycopy
```

Out[20]: [200, 300, 400, 'test']

In [ ]:
```
1
```

In [21]:
```
1 mylist.extend(["mumbai","pune","chennai","delhi","nagpur"])
```

In [22]:
```
1 mylist
```

Out[22]: [200, 300, 400, 'test', 'mumbai', 'pune', 'chennai', 'delhi', 'nagpur']

In [ ]:
```
1
```

In [23]:
```
1 mylist.append(['python','java','c'])
```

In [24]:
```
1 mylist
```

Out[24]:
```
[200,
 300,
 400,
 'test',
 'mumbai',
 'pune',
 'chennai',
 'delhi',
 'nagpur',
 ['python', 'java', 'c']]
```

In [25]:
```
1 mylist[9]
```

Out[25]:
```
['python', 'java', 'c']
```

In [26]:
```
1 mylist[9][1]
```

Out[26]:
```
'java'
```

In [27]:
```
1 nested_list = [
2     [
3         ["hello","welcome",33]
4     ],
5     [
6         [222,333,444]
7     ]
8 
9 ]
```

In [28]:
```
1 nested_list
```

Out[28]:
```
[[['hello', 'welcome', 33]], [[222, 333, 444]]]
```

In [29]:
```
1 nested_list[0]
```

Out[29]:
```
[['hello', 'welcome', 33]]
```

In [30]:
```
1 nested_list[0][0]
```

Out[30]:
```
['hello', 'welcome', 33]
```

In [31]:
```
1 nested_list[0][0][1]
```

Out[31]:
```
'welcome'
```

In [34]:
```
1 nested_list[1][0][2]
```

Out[34]:
```
444
```

In [35]:
```
1 l = [
2     [
3         [
4             [
5                 ["i","have","the"],["knowledge","of","programming"]
6             ]
7         ]
8     ]
9 ]
```

In [36]:
```
1 l
```

Out[36]:
```
[[[[['i', 'have', 'the'], ['knowledge', 'of', 'programming']]]]]
```

In [39]:
```
1 l
```

Out[39]:
```
[[[[['i', 'have', 'the'], ['knowledge', 'of', 'programming']]]]]
```

In [40]:
```
1 l[0]
```

Out[40]:
```
[[[['i', 'have', 'the'], ['knowledge', 'of', 'programming']]]]
```

In [41]:
```
l[0][0]
```

Out[41]:
```
[['i', 'have', 'the'], ['knowledge', 'of', 'programming']]
```

In [42]:
```
l[0][0][0]
```

Out[42]:
```
['i', 'have', 'the'], ['knowledge', 'of', 'programming']]
```

In [43]:
```
l[0][0][0][0]
```

Out[43]:
```
['i', 'have', 'the']
```

In [44]:
```
l[0][0][0][0][1]
```

Out[44]:
```
'have'
```

In [45]:
```
l[0]
```

Out[45]:
```
[[['i', 'have', 'the'], ['knowledge', 'of', 'programming']]]
```

In [46]:
```
l[0][0]
```

Out[46]:
```
[['i', 'have', 'the'], ['knowledge', 'of', 'programming']]
```

In [47]:
```
l[0][0][0]
```

Out[47]:
```
['i', 'have', 'the'], ['knowledge', 'of', 'programming']]
```

In [49]:
```
l[0][0][0][1][2]
```

Out[49]:
```
'programming'
```

In [ ]:
```

```

In [50]:
```
new_list = [[[[['i', 'have', 'the', 'power', 'of', 'coding'],
    ['knowledge', 'in', 'programming', 'python']]]]]
```

In [51]:
```
1 new_list
```

Out[51]:
```
[[[['i', 'have', 'the', 'power', 'of', 'coding'],
   ['knowledge', 'in', 'programming', 'python']]]]
```

In [57]:
```
1 new_list[0][0][0][0][0]
```

Out[57]:
```
'i'
```

In [58]:
```
1 new_list[0][0][0][0][3]
```

Out[58]:
```
'power'
```

In [60]:
```
1 new_list[0][0][0][1][0]
```

Out[60]:
```
'knowledge'
```

In [61]:
```
1 new_list[0][0][0][1][1]
```

Out[61]:
```
'in'
```

In [ ]:
```
1
```

In [ ]:
```
1
```

Tuples

- Tuple is immutable -- (cannot be updated)
- Tuples can store duplicate values.
- Tuple can store any data type
- the indexing in tuple starts from 0

In [62]:
```
1 mytup = (100,200,300,400,500)
```

In [63]:
```
print(mytup)
```

```
(100, 200, 300, 400, 500)
```

In [67]:
```
print(type(mytup[0]))
```

```
<class 'int'>
```

In [65]:
```
mytup1 = ('mumbai','pune',100,200,33.33,True,'mumbai')
```

In [66]:
```
mytup1
```

Out[66]:
```
('mumbai', 'pune', 100, 200, 33.33, True, 'mumbai')
```

In [ ]:
```

```

In [68]:
```
mytup1.index('pune')
```

Out[68]:
```
1
```

In [69]:
```
mytup1.count('mumbai')
```

Out[69]:
```
2
```

In [ ]:
```

```

In [70]:
```
nested_tup = (

    (100,200,300,400),
    ("mumbai","pune","chennai")

)
```

In [71]:
```
nested_tup
```

Out[71]:
```
((100, 200, 300, 400), ('mumbai', 'pune', 'chennai'))
```

In [ ]:
```
1
```

In [72]:
```
1 mixed = [
2
3     (1000,2000,3000,4000),
4     ("mumbai","new york","tokyo"),
5     ["hello","hi","bye"]
6 ]
7
```

In [73]:
```
1 mixed
```

Out[73]:
```
[(1000, 2000, 3000, 4000),
 ('mumbai', 'new york', 'tokyo'),
 ['hello', 'hi', 'bye']]
```

In [76]:
```
1 mixed[0]
```

Out[76]:
```
(1000, 2000, 3000, 4000)
```

In [78]:
```
1 mixed[2].append('tata')
```

In [79]:
```
1 mixed
```

Out[79]:
```
[(1000, 2000, 3000, 4000),
 ('mumbai', 'new york', 'tokyo'),
 ['hello', 'hi', 'bye', 'tata']]
```

In [81]:
```
1 mixed.append('python')
```

In [82]:
```
1 mixed
```

Out[82]:
```
[(1000, 2000, 3000, 4000),
 ('mumbai', 'new york', 'tokyo'),
 ['hello', 'hi', 'bye', 'tata'],
 'python']
```

In [ ]:
```
1
```

In [83]:
```
1 mixed2= (
2
3     ['python','c','c++'],
4     ['mumbai','pune','chennai'],
5     [1000,200,3000],
6     ("hi","bye")
7 )
```

In [87]:
```
1 mixed2[3]
```

Out[87]:
```
('hi', 'bye')
```

In [85]:
```
1 mixed2.append(5000)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [85], in <module>
----> 1 mixed2.append(5000)

AttributeError: 'tuple' object has no attribute 'append'
```

In [ ]:
```
1
```

In [ ]:
```
1
```

Dictionary

- Dictionary are mutable data structure in python
- The data is stored in key value pair
- The keys can be (integer, boolean,float,string)
- The keys should not be repeated

In [ ]:
```
1
```

In [88]:
```
1 mydictionary = {
2     'name':'punit',
3     'age':30,
4     'designation':'SDE',
5     'location':'Mumbai'
6 }
```

In [89]:
```
1 mydictionary
```

Out[89]:
```
{'name': 'punit', 'age': 30, 'designation': 'SDE', 'location': 'Mumbai'}
```

In [90]:
```
1 mydictionary['location']
```

Out[90]:
```
'Mumbai'
```

In [91]:
```
1 mydictionary['Location']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [91], in <module>
----> 1 mydictionary['Location']

KeyError: 'Location'
```

In [92]:
```
1 mydictionary['age']
```

Out[92]:
```
30
```

In [ ]:
```
1
```

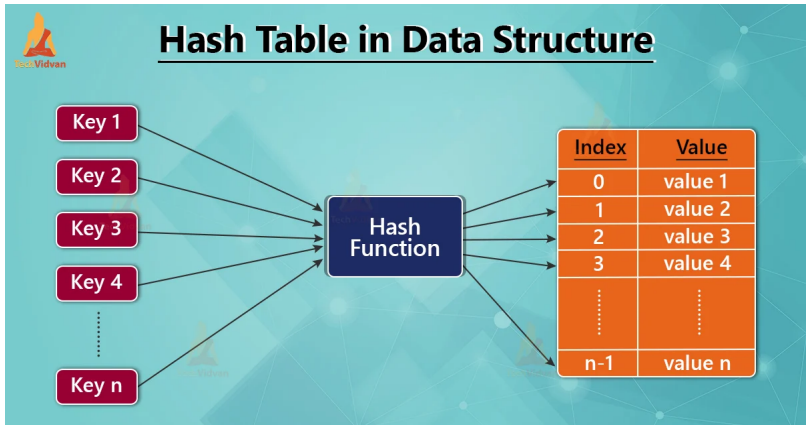
In [93]:
```
mixed = {
    'city':'Mumbai',
    10 : 'This is an integer',
    33.33 : 'This is a float',
    True:'This is a boolean'

}
```

In [94]:
```
mixed
```

Out[94]:
```
{'city': 'Mumbai',
 10: 'This is an integer',
 33.33: 'This is a float',
 True: 'This is a boolean'}
```

In [98]:
```
mixed[True]
```

Out[98]:
```
'This is a boolean'
```



In [99]:
```
mixed1 = {
    1 : 'This is an integer',
    1.0 : 'This is a float',
    True: 'This is a boolean'
}
```

In [107]:
```
1 mixed1[True]
```

Out[107]: 'This is a boolean'

In [102]:
```
1 hash(1)
```

Out[102]: 1

In [103]:
```
1 hash(1.0)
```

Out[103]: 1

In [104]:
```
1 hash(True)
```

Out[104]: 1

In [ ]:
```
1
```

In [ ]:
```
1
```

In [108]:
```
1 mydictionary
```

Out[108]: {'name': 'punit', 'age': 30, 'designation': 'SDE', 'location': 'Mumbai'}

In [16]:
```
1 mydidctionary = {'name': 'punit', 'age': 30, 'designation': 'SDE', 'location': 'Mumbai'}
```

In [2]:
```
1 print(mydidctionary)
```

```
{'name': 'punit', 'age': 30, 'designation': 'SDE', 'location': 'Mumbai'}
```

In [3]:
```
1 mydidctionary.clear()
```

In [4]:
```
1 mydidctionary
```

Out[4]: {}

In [6]:
```
1 copied = mydidctionary.copy()
```

In [7]:
```
1 copied
```

Out[7]:
```
{'name': 'punit', 'age': 30, 'designation': 'SDE', 'location': 'Mumbai'}
```

In [8]:
```
1 mydidctionary
```

Out[8]:
```
{'name': 'punit', 'age': 30, 'designation': 'SDE', 'location': 'Mumbai'}
```

In [9]:
```
1 mydidctionary.clear()
```

In [10]:
```
1 mydidctionary
```

Out[10]:
```
{}
```

In [11]:
```
1 copied
```

Out[11]:
```
{'name': 'punit', 'age': 30, 'designation': 'SDE', 'location': 'Mumbai'}
```

In [12]:
```
1 new_copy = copied
```

In [13]:
```
1 copied.clear()
```

In [14]:
```
1 new_copy
```

Out[14]:
```
{}
```

In [15]:
```
1 copied
2
```

Out[15]:
```
{}
```

In [17]:
```
1 mydidctionary.get('name')
```

Out[17]:
```
'punit'
```

In [18]:
```
1 mydidctionary['name']
```

Out[18]:
```
'punit'
```

In [19]:
```
1 mydidctionary.get('loc')
```

In [20]:
```
1 mydidctionary['loc']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [20], in <module>
----> 1 mydidctionary['loc']

KeyError: 'loc'
```

In [ ]:
```
1
```

In [21]:
```
1 mydidctionary
```

Out[21]:
```
{'name': 'punit', 'age': 30, 'designation': 'SDE', 'location': 'Mumbai'}
```

In [22]:
```
1 mydidctionary.keys()
```

Out[22]:
```
dict_keys(['name', 'age', 'designation', 'location'])
```

In [23]:
```
1 mydidctionary.values()
```

Out[23]:
```
dict_values(['punit', 30, 'SDE', 'Mumbai'])
```

In [24]:
```
1 mydidctionary.items()
```

Out[24]:
```
dict_items([('name', 'punit'), ('age', 30), ('designation', 'SDE'), ('location', 'Mumbai')])
```

In [ ]:
```
1
```

In [26]:
```
1 mydidctionary.pop('age')
```

Out[26]:
```
30
```

In [27]:
```
1 mydidctionary
```

Out[27]:
```
{'name': 'punit', 'designation': 'SDE', 'location': 'Mumbai'}
```

In [28]:
```
1 mydidctionary.pop('loc')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [28], in <module>
----> 1 mydidctionary.pop('loc')

KeyError: 'loc'
```

In [29]:
```
1 mydidctionary.popitem()
```

Out[29]:
```
('location', 'Mumbai')
```

In [36]:
```
1 mydidctionary
2
```

Out[36]:
```
{'name': 'demo', 'designation': 'SDE', 'loc': 'Khar'}
```

In [ ]:
```
1
```

In [31]:
```
1 mydidctionary.update({'name':'demo'})
```

In [32]:
```
1 mydidctionary
```

Out[32]:
```
{'name': 'demo', 'designation': 'SDE'}
```

In [34]:
```
1 mydidctionary.update({'loc':'Khar'})
```

In [35]:
```
1 mydidctionary
```

Out[35]:
```
{'name': 'demo', 'designation': 'SDE', 'loc': 'Khar'}
```

In [37]:
```
1 mydidctionary['qualification'] = 'MCA'
```

In [38]:
```
1 mydidctionary
```

Out[38]:
```
{'name': 'demo', 'designation': 'SDE', 'loc': 'Khar', 'qualification': 'MCA'}
```

In [39]:
```
1 mydidctionary = {'age':}
```

```
  Input In [39]
    mydidctionary = {'age':}
                           ^
SyntaxError: invalid syntax
```

In [40]:
```
1 mydidctionary.setdefault('pincode')
```

In [41]:
```
1 mydidctionary
```

Out[41]:
```
{'name': 'demo',
 'designation': 'SDE',
 'loc': 'Khar',
 'qualification': 'MCA',
 'pincode': None}
```

In [42]:
```
1 mydidctionary.update({'pincode':400001})
```

In [43]:
```
1 mydidctionary
```

Out[43]:
```
{'name': 'demo',
 'designation': 'SDE',
 'loc': 'Khar',
 'qualification': 'MCA',
 'pincode': 400001}
```

In [ ]:
```
1
```

nested dictionary

In [44]:

```
fb_data = {
    40001 : {
        "name":"punit",
        "age":30,
        "likes":1000,
        "friends":500,
        "comments":100

    },


    40002:{
        "name":"aditya",
        "age":21,
        "likes":2000,
        "friends":278,
        "comments":200
    },


    40003:{
        "name":"tejas",
        "age":24,
        "likes":2500,
        "friends":1000,
        "comments":10
    },

    40004:{
        "name":"javed",
        "age":21,
        "likes":1021,
        "friends":800,
        "comments":400

    }

}
```

In [45]:
```
1 fb_data
```

Out[45]:
```
{40001: {'name': 'punit',
  'age': 30,
  'likes': 1000,
  'friends': 500,
  'comments': 100},
 40002: {'name': 'aditya',
  'age': 21,
  'likes': 2000,
  'friends': 278,
  'comments': 200},
 40003: {'name': 'tejas',
  'age': 24,
  'likes': 2500,
  'friends': 1000,
  'comments': 10},
 40004: {'name': 'javed',
  'age': 21,
  'likes': 1021,
  'friends': 800,
  'comments': 400}}
```

In [47]:
```
1 fb_data[40001]['friends']
```

Out[47]:
```
500
```

In [50]:
```
1 fb_data[40002]['comments']
```

Out[50]:
```
200
```

In [52]:
```
1 fb_data[40003]['age']
```

Out[52]:
```
24
```

In [54]:
```
1 fb_data[40004]['likes']
```

Out[54]:
```
1021
```

In [56]:
```
1 fb_data[40001].update({'hobbies':['cooking','travelling','music','sleeping']})
```

In [57]:
```
1 fb_data
```

Out[57]:
```
{40001: {'name': 'punit',
  'age': 30,
  'likes': 1000,
  'friends': 500,
  'comments': 100,
  'hobbies': ['cooking', 'travelling', 'music', 'sleeping']},
 40002: {'name': 'aditya',
  'age': 21,
  'likes': 2000,
  'friends': 278,
  'comments': 200},
 40003: {'name': 'tejas',
  'age': 24,
  'likes': 2500,
  'friends': 1000,
  'comments': 10},
 40004: {'name': 'javed',
  'age': 21,
  'likes': 1021,
  'friends': 800,
  'comments': 400}}
```

In [60]:
```
1 fb_data[40001]['hobbies'][2]
```

Out[60]:
```
'music'
```

In [ ]:
```
1
```

In [ ]:
```
1
```

Sets

- sets are un-ordered collection in python

- It does not support indexing
- It does not allow us to add duplicate values
- You can use different data type to define the sets

In [78]:
```
myset = {10,20,30,0,5,12,33,4,8,32}
```

In [62]:
```
print(myset)
```

```
{0, 33, 32, 4, 5, 8, 10, 12, 20, 30}
```

In [63]:
```
mixed = {'hello','hi',True,False,44.33,22}
```

In [64]:
```
mixed
```

Out[64]:
```
{22, 44.33, False, True, 'hello', 'hi'}
```

In [65]:
```
myset2 = {10,20,30,10,20}
```

In [66]:
```
myset2
```

Out[66]:
```
{10, 20, 30}
```

In [67]:
```
myset
```

Out[67]:
```
{0, 4, 5, 8, 10, 12, 20, 30, 32, 33}
```

In [70]:
```
myset.add(100)
```

In [71]:
```
myset
```

Out[71]:
```
{0, 4, 5, 8, 10, 12, 20, 30, 32, 33, 100}
```

In [72]:
```
myset.clear()

```

In [73]:
```
1 myset
```

Out[73]:
```
set()
```

In [74]:
```
1 myset3 = {}
```

In [75]:
```
1 print(type(myset3))
```

```
<class 'dict'>
```

In [76]:
```
1 myset3 = set()
```

In [77]:
```
1 myset3
```

Out[77]:
```
set()
```

In [79]:
```
1 copied = myset.copy()
```

In [80]:
```
1 copied
```

Out[80]:
```
{0, 4, 5, 8, 10, 12, 20, 30, 32, 33}
```

In [ ]:
```
1
```

In [81]:
```
1 myset.pop()
```

Out[81]:
```
0
```

In [82]:
```
1 myset.remove(33)
```

In [83]:
```
1 myset
```

Out[83]:
```
{4, 5, 8, 10, 12, 20, 30, 32}
```

In [84]:
```
1 myset1 = {10,20,30,40}
2 myset2 = {40,50,60,70}
```

In [85]:
```
1 myset1.union(myset2)
```

Out[85]: {10, 20, 30, 40, 50, 60, 70}

In [86]:
```
1 myset1.intersection(myset2)
```

Out[86]: {40}

In [ ]:
```
1
```

In [87]:
```
1 myset1 = {'apple','samsung','nokia'}
2 myset2 = {'redmi','oppo','samsung'}
```

In [88]:
```
1 myset1.difference(myset2)
```

Out[88]: {'apple', 'nokia'}

In [89]:
```
1 myset2.difference(myset1)
```

Out[89]: {'oppo', 'redmi'}

In [ ]:
```
1
```

In [90]:
```
1 x1 ={'mumbai','pune','chennai'}
2 x2 = {'delhi','nagpur','nasik'}
```

In [91]:
```
1 x1.update(x2)
```

In [92]:
```
1 x1
```

Out[92]: {'chennai', 'delhi', 'mumbai', 'nagpur', 'nasik', 'pune'}

In [93]:
```
1 x2
```

Out[93]: {'delhi', 'nagpur', 'nasik'}

In [ ]:
```
1
```

In [109]:
```
1 x = {'a1','b1','c1'}
2
3 y = {'f','g','h','i','b1','c1','a1'}
```

In [100]:
```
1 x.issubset(y)
```

Out[100]:
```
True
```

In [101]:
```
1 x.issuperset(y)
```

Out[101]:
```
False
```

In [102]:
```
1 y.issubset(x)
```

Out[102]:
```
False
```

In [103]:
```
1 x
```

Out[103]:
```
{'a1', 'b1', 'c1'}
```

In [104]:
```
1 x.remove('demo')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [104], in <module>
----> 1 x.remove('demo')

KeyError: 'demo'
```

In [105]:
```
1 x.discard('a1')
```

In [106]:
```
1 x
```

Out[106]:
```
{'b1', 'c1'}
```

In [107]:
```
1 x.discard('demo')
```

In [112]:
```
1 x
```

Out[112]:
```
{'a1', 'b1', 'c1'}
```

In [113]:
```
1 y
```

Out[113]:
```
{'a1', 'b1', 'c1', 'f', 'g', 'h', 'i'}
```

In [110]:
```
1 x.symmetric_difference(y)
```

Out[110]:
```
{'f', 'g', 'h', 'i'}
```

In [ ]:
```
1
```

In [114]:
```
1 x = {'apple','banana','cherry'}
2 y = {'google','microsoft','apple'}
```

In [115]:
```
1 x.symmetric_difference(y)
```

Out[115]:
```
{'banana', 'cherry', 'google', 'microsoft'}
```

In [ ]:
```
1
```

In [116]:
```
1 myset = {
2     {10,20,30}
3 }
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [116], in <module>
----> 1 myset = {
      2     {10,20,30}
      3 }

TypeError: unhashable type: 'set'
```

In [ ]:
```
1
```

In [117]:
```
1 x
```

Out[117]:
```
{'apple', 'banana', 'cherry'}
```

In [118]:
```
1 froze_set = frozenset(x)
```

In [119]:
```
1 froze_set
```

Out[119]:
```
frozenset({'apple', 'banana', 'cherry'})
```

In [ ]:
```
1 froze_set.
```