Sets

A set is an unordered collection data type that is iterable,mutable, and has no duplicate elements.Python's set class represents the mathematical notion of a set. This is based on a data structure known as a hash table

In [1]:
```
##  Defining an empty set
set_var= set()
print(set_var)
print(type(set_var))
```

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

In [2]:
```
set_var={1,2,3,4,3}      # dublicate variables
```

In [3]:
```
set_var
```

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

In [4]:
```
set_var={"Avengers","Ironmen","Hitman"}
print(set_var)
type(set_var)
```

```
{'Avengers', 'Hitman', 'Ironmen'}
```

Out[4]:
```
set
```

In [5]:
```
## set does not support indexing function
```

In [6]:
```
## Inbuilt funtion in sets
set_var.add("bulk")
```

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

```
{'Avengers', 'Hitman', 'bulk', 'Ironmen'}
```

In [8]:
```
set_var.     ## all fucntion of set can be look by tab button after point(.)
```

```
  Input In [8]
    set_var.     ## all fucntion of set can be look by tab button after point(.)
                 ^
SyntaxError: invalid syntax
```

In [18]:
```
set1= {"avengers","ironmen","hitmen"}
set2= {"avengers","ironmen","hitmen","hulk2"}
```

In [19]:
```
set2.intersection(set1)
```

Out[19]:
```
{'avengers', 'hitmen', 'ironmen'}
```

In [20]:
```
set2
```

Out[20]:
```
{'avengers', 'hitmen', 'hulk2', 'ironmen'}
```

In [22]:
```
set2.intersection_update(set1)
```

In [23]:
```
set2
```

Out[23]:
```
{'avengers', 'hitmen', 'ironmen'}
```

In [10]:
```
## Differences
set2.difference(set1)
```

Out[10]:
```
{'hulk2'}
```

In [13]:
```
## Difference update
set2.difference_update(set1)
```

In [14]:
```
print(set2)
```

```
{'hulk2'}
```

Dictionaries

A dictionary is a collection which is unordered,changeable and indexed.In python dictionaries are written with curly brackets,and they have keys and values.

In [24]:
```
dic={}
```

In [25]:
```
type(dic)
```

Out[25]:
```
dict
```

In [26]:
```
dic={1,2,3,4.5}
```

In [27]:
```
type(dic)
```

Out[27]:
```
set
```

In [28]:
```
## let create a dictionary
my_dict={"car1":"Audi","car2":"BMW","car3":"Mercidies benz"}
```

In [29]:
```
type(my_dict)
```

Out[29]:
```
dict
```

In [30]:
```
## Access the item values based on key's
my_dict['car1']
```

Out[30]:
```
'Audi'
```

In [32]:
```
# we can even loop thoughtout the dictionaries keys
for x in my_dict:
    print(x)
```

```
car1
car2
car3
```

In [34]:
```
# we can also check throught the dictionaries values
for x in my_dict.values():
    print(x)
```

```
Audi
BMW
Mercidies benz
```

In [35]:
```
# we can also check both keys and values
for x in my_dict.items():
    print(x)
```

```
('car1', 'Audi')
('car2', 'BMW')
('car3', 'Mercidies benz')
```

In [36]:
```
## Adding items in Dictionaries
my_dict['car4']='Audi 2.0'
```

In [37]:
```
my_dict
```

Out[37]:
```
{'car1': 'Audi', 'car2': 'BMW', 'car3': 'Mercidies benz', 'car4': 'Audi 2.0'}
```

In [38]:
```
my_dict['car1']='Maruti'  ## replacment
```

In [39]:
```
my_dict
```

Out[39]:
```
{'car1': 'Maruti', 'car2': 'BMW', 'car3': 'Mercidies benz', 'car4': 'Audi 2.0'}
```

In [ ]:
```

```

Nested Dictionary

In [40]:
```
car1_model={'Mercedes':1960}
car2_model={'Audi':1970}
car3_model={'Ambassador':1980}
car_type={'car1':car1_model,'car2':car2_model,'car3':car3_model}
```

In [41]:
```
print(car_type)
```

```
{'car1': {'Mercedes': 1960}, 'car2': {'Audi': 1970}, 'car3': {'Ambassador': 1980}}
```

In [42]:
```
## Accessing the items in the dictionary

print(car_type['car1'])
```

```
{'Mercedes': 1960}
```

In [43]:
```
print(car_type['car1']['Mercedes'])
```

```
1960
```

Tuples

tuples is not mutable

In [44]:
```
## create an empty Tuples

my_tuples=tuple()
```

In [45]:
```
type(my_tuples)
```

Out[45]:
```
tuple
```

In [46]:
```
my_tuple=()
```

In [47]:
```
type(my_tuple)
```

Out[47]:
```
tuple
```

In [48]:
```
my_tuple=('krish','ankur','john')
```

In [49]:
```
my_tuple[0]
```

Out[49]:
```
'krish'
```

In [50]:
```
print(type(my_tuple))
print(my_tuple)
```

```
<class 'tuple'>
('krish', 'ankur', 'john')
```

In [51]:
```
type(my_tuple)
```

Out[51]:
```
tuple
```

In [52]:
```
## Inbuilt function
my_tuple.count('krish')
```

Out[52]:
```
1
```

In [54]:
```
my_tuple.index('ankur')
```

Out[54]:
```
1
```

In [ ]:
```
## tuple does not assignment or replacement of singal element its change whole ele
```