In [ ]:
```
#Agenda of the day:
                    1. Sets in Python
                    2. Dictionaries in Python
                    3. Regular Expressions in Python
```

In [ ]:
```
#Sets:(union & Intersection)
```

In [1]:
```
print(dir(set),end=" ")
```

```
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__
iand__', '__init__', '__init_subclass__', '__ior__', '__isub__', '__iter__', '_
_ixor__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__rand
__', '__reduce__', '__reduce_ex__', '__repr__', '__ror__', '__rsub__', '__rxor_
_', '__setattr__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__x
or__', 'add', 'clear', 'copy', 'difference', 'difference_update', 'discard', 'i
ntersection', 'intersection_update', 'isdisjoint', 'issubset', 'issuperset', 'p
op', 'remove', 'symmetric_difference', 'symmetric_difference_update', 'union',
'update']
```

In [12]:
```
#Copy() and clear() pop():
s1 = {1,2,3,4,5}
s2 = {}
s2= s1.copy()
s2.add((10,30,50,90,"Wednesday")) #to add single element to set
s2.update([10,50,30,90,"Wednesday"])  #to add individual elements
print("before clear:",s2)
s2.clear()                    #To clear all contents of the set
print("after clear:",s2)
```

```
before clear: {1, 2, 3, 4, 5, 'Wednesday', 90, 1050, 30, (10, 30, 50, 90, 'Wedn
esday')}
after clear: set()
```

In [31]:
```
#pop(): (Its deletes the one elements at a time randomly)
s2 = {1, 2, 3, 4, 5, 'Wednesday', 90, 1050, 30, (10, 30, 50, 90, 'Wednesday')}
s2.pop()
s2.pop()
s2.pop()
s2.pop()
s2.pop()
s2.pop()
s2.pop()
s2.pop()
s2.pop()
s2.pop()
s2.pop()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-31-c15df6bcfbff> in <module>
     11 s2.pop()
     12 s2.pop()
---> 13 s2.pop()

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

In [ ]:
```
#Set Difference:
                A-B---result remaining elements of A
                B-A---resutt remaining elements of B
```

In [36]:
```
#Ex:
SetA = {1,2,3,4,5}
SetB = {4,5,6,7,8}
#Difference can be done by "-" operator
print(SetA-SetB)
print(SetB-SetA)
#using method:
print(SetA.difference(SetB))
print(SetB.difference(SetA))
```

```
{1, 2, 3}
{8, 6, 7}
{1, 2, 3}
{8, 6, 7}
```

In [41]:
```
#Set Symmetric Difference:
#To get elements in A and B sets but not intersection values
SetA = {1,2,3,4,5}
SetB = {4,5,6,7,8}
print(SetA ^ SetB)
print(SetB ^ SetA)
#using method:
print(SetA.symmetric_difference(SetB))
print(SetB.symmetric_difference(SetA))
```

```
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
```

In [ ]:
```
#set Difference_update:
#Syntax:
A.difference_update(B)
#result be
 1. A will be updated with difference(A-B)
 2. B will not change
 3. Result will be none object.
```

In [48]:
```
#Ex:
A = {1,2,3,4,5}
B = {2,3,4}
res=A.difference_update(B)
print(res)
print(A)
print(B)
```

```
None
{1, 5}
{2, 3, 4}
```

In [ ]:
```
#symmetric_difference_update():
#Syntax:
A.symmetric_difference_update(B)
#result be
 1. Set A will be updated with symmetric difference of Set A and Set B
 2. Set B will not change
 3. Result will be none object.
```

In [54]:
```
#Ex:
A = {1,2,3,4,5}
B = {2,3,4,6,8}
res=A.symmetric_difference_update(B)
print(res)
print(A)
print(B)
```

```
None
{1, 5, 6, 8}
{2, 3, 4, 6, 8}
```

In [ ]:
```
#Boolean Operations(issubset(),issuperset(),isdisjoint()):

#Superset: if SetA contains all elements of SetB--then A is Superset of B
#Subset:  if all elements of a are in B.
#disjoint : set a and set b havent common values.
```

In [59]:
```
#issuperset():
A = {1,2,3,4,5}
B = {1,2,3}
C = {1,2,3}
print(A.issuperset(B))
print(B.issuperset(A))
print(C.issuperset(B))
print(B.issuperset(C))
```

```
True
False
True
True
```

In [64]:
```
#issubset():
#Set A is said to be subset of Set B if all elements of A are in B.
#syntax: A.issubset(B)
A = {1,2,3}
B = {1,2,3,4,5}
C=  {1,2,4,5}
print(A.issubset(B))
print(B.issubset(A))
print(A.issubset(C))
print(C.issubset(B))
```

```
True
False
False
True
```

In [ ]:
```
#isdisjoint():
If two sets are said as disjoint if they have no common elements.
#syntax:
A.isdisjoint(B)
```

In [69]:
```
#Ex:
A = {50,60,90,100}
B =  {150,200,300,500}
C = {60,600,1000}
print(A.isdisjoint(B))
print(A.isdisjoint(C))
print(B.isdisjoint(C))
```

```
True
False
True
```

In [ ]:
```
#Dictionaries:
--A Dictionary is a Collection of key/value pairs.
#syntax:    dict = {key:value}
            dict = {1:"APSSDC"}
#Keypoints:
1. Keys are unique, there is no multiple keys for single value
2. values are mutable (changeble)
3. Its respresented by {} or dict().
4. for representing keys we use intergers or strings
5. for represeenting values we use any datatype of values.
```

In [71]:
```
#How to create dictionary?
dic1 = {}                    #empty dictionary-way1
print(type(dic1))
dic2 = dict()                  ##empty dictionary-way2
print(type(dic2))
```

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

In [84]:
```
#How to assign items to Dictionary?
dic = {1:"Python",2:"Data Science",3:"KITS",4:"AI",5:"Python"}
dic1 = {"Hello":"Surya","Today":"Wednesday","Date":23,4:50.6}
print(dic)
print(dic1)
```

```
{1: 'Python', 2: 'Data Science', 3: 'KITS', 4: 'AI', 5: 'Python'}
{'Hello': 'Surya', 'Today': 'Wednesday', 'Date': 23, 4: 50.6}
```

In [92]:
```
#Changing and adding items to dictionary:
dic = {1:"Python",2:"Data Science",3:"KITS",4:"AI",5:"Python"}
dic[5]="JAVA"   #changing value python to JAVA
dic[2]="Machine Learning"
dic[6]="C Lan"   #adding new item(key,value pair)
dic[7]="PHP"
dic["Apssdc"]=123
dic[6]=56
print(dic)
```

```
{1: 'Python', 2: 'Machine Learning', 3: 'KITS', 4: 'AI', 5: 'JAVA', 6: 56, 7:
'PHP', 'Apssdc': 123}
```

In [106]:
```
#Deleting items from Dictinaries?():
#(pop(),popitem(),clear())
dic = {1:"Python",2:"Data Science",3:"KITS",4:"AI",5:"Python"}
dic1 = {"Hello":"Surya","Today":"Wednesday","Date":23,4:50.6}
#dic.pop(3)
#dic.pop(2)
#dic.pop(4)
#dic.pop(5)
#dic.pop(1)
dic1.pop("Hello")
print(dic1)
```

```
{'Today': 'Wednesday', 'Date': 23, 4: 50.6}
```

In [120]:
```
#popitem()
dic = {1:"Python",2:"Data Science",3:"KITS",4:"AI",5:"Python"}
dic1 = {"Hello":"Surya","Today":"Wednesday","Date":23,4:50.6}
#dic.popitem()
#dic1.popitem()
dic.clear()
print(dic)
```

```
{}
```

In [132]:
```
#How to accessing items from the dictionary:
dic = {1:"Python",2:"Data Science",3:"KITS",4:"AI",5:"Python"}
dic1 = {"Hello":"Surya","Today":"Wednesday","Date":23,4:50.6}
print(dic[2])
print(dic[5])
print(dic1["Date"])
print(dic1["Hello"])
#2-way of accessing values by passing keys
print(dic.get(3))
print(dic1.get("Today"))
```

```
Data Science
Python
23
Surya
KITS
Wednesday
```

In [133]:
```
#Exploring the  dictionary methods:
print(dir(dict),end=" ")
```

```
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt_
_', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len_
_', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__reversed__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subcl
asshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popite
m', 'setdefault', 'update', 'values']
```

In [140]:
```
#methods():(items(),keys(),values())
dic = {1:"Python",2:"Data Science",3:"KITS",4:"AI",5:"Python"}
dic1 = {"Hello":"Surya","Today":"Wednesday","Date":23,4:50.6}
print(dic.keys())
print(dic1.keys())
print(dic.values())
print(dic1.values())
print(dic.items())
print(dic1.items())
```

```
dict_keys([1, 2, 3, 4, 5])
dict_keys(['Hello', 'Today', 'Date', 4])
dict_values(['Python', 'Data Science', 'KITS', 'AI', 'Python'])
dict_values(['Surya', 'Wednesday', 23, 50.6])
dict_items([(1, 'Python'), (2, 'Data Science'), (3, 'KITS'), (4, 'AI'), (5, 'Py
thon')])
dict_items([('Hello', 'Surya'), ('Today', 'Wednesday'), ('Date', 23), (4, 50.
6)])
```

In [149]:
```
#copy() and clear() and del
dic = {1:"Python",2:"Data Science",3:"KITS",4:"AI",5:"Python"}
dic2={}
dic2 = dic.copy()
print(dic2)
#dic==dic2
dic2.clear()
print(dic2)
del dic
print(dic)
```

```
{1: 'Python', 2: 'Data Science', 3: 'KITS', 4: 'AI', 5: 'Python'}
{}

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-149-e9f37a9e393f> in <module>
      8 print(dic2)
      9 del dic
---> 10 print(dic)

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

In [154]:
```
#fromkeys()
#(Its creates a new dictionary from the given sequence of elements
#with a value)
#syntax: dict.fromkeys(sequence[,value])
keys = {1,2,3,4,5}
#value = "Coding is fun"
#value = 1
value = 100.5
newdict=dict.fromkeys(keys,value)
print(newdict)
```

```
{1: 100.5, 2: 100.5, 3: 100.5, 4: 100.5, 5: 100.5}
```

In [171]:
```
#Update:
dic = {1:"Python",2:"Data Science",3:"KITS",4:"AI",5:"Python"}
dic2 = {1:"Machine Learning",2:"C",3:"PHP",4:500,5:90.5}
#dic.update(dic2)
dic2.update(dic)           #updating one dictionary with another dictionary
#print(dic)
dic2= {6:"New Value"}     #updating the dictionary
dic[1]="MySQL"            #changing the values
dic[8]="Java Script"      #assiging new item(key,value)
print(dic)
print(dic2)
```

```
{1: 'MySQL', 2: 'Data Science', 3: 'KITS', 4: 'AI', 5: 'Python', 8: 'Java Scrip
t'}
{6: 'New Value'}
```

In [175]:
```
#Set default():
#This method returns the value of a key (if that key is in that
#                                  dictionary)

#Syntax: dict.setdefault(key[,default_value])
profile = {"name":"surya","profession":"Python Trainer"}
profession = profile.setdefault("profession")
print(profile)
print(profession)
```

```
{'name': 'surya', 'profession': 'Python Trainer'}
Python Trainer
```

In [ ]:

In [ ]: