In [ ]:
```
#Agenda of the Day:

                    1. Tuples in Python.

A Tuple is a Data Structure that is used to store multiple data at one Time.

#Keypoints:

        1. We can store multiple data of different data types
             like strings, integers, object,floats,Boolean values.
        2. Compared to list , Tuple is immutable (not changable)

        3. Data in tuple is acccesed by thier index (start from zero)

        4. Its increases the performance as iterating in a tuple is
            faster than list.
        5. All values in tuple are separated by comma and
           enclosed by parenthesis ().
```

In [ ]:
```
#How to create a tuple?
#syntax:
tuples = (item1,item2,item3.....)
```

In [2]:
```
#how to create tuple?
tup = ()                                    #empty tuple
print(type(tup))
tup1 = tuple()                          #empty tuple
print(type(tup1))
```

```
<class 'tuple'>
<class 'tuple'>
```

In [9]:
```
#How create tuples with values?
tup = (1,2,3,"Tuesday",50.3,True)
tup1 = ("Kits","IT","AI","DATA Science","Machine Learning")
print(tup)
tup[0]
tup[-1]
```

```
(1, 2, 3, 'Tuesday', 50.3, True)
```

Out[9]:
```
True
```

In [20]:
```
#Example:
tup = ("Python",[1,2,3],(5,6,9),50,100)
print(tup[0][1])                                #nesting indexing
print(tup[0][2])
print(tup[1][2])
print(tup[2][1])
print(tup[4])                                    #single indexing
```

```
y
t
3
6
100
```

In [29]:
```
#Slicing a Tuple:
tup = (1,2,3,4,5,6,"python")
print(tup[1:5])
print(tup[-4:-1])
print(tup[-3:])
print(tup[:1])
```

```
(2, 3, 4, 5)
(4, 5, 6)
(5, 6, 'python')
(1,)
```

In [33]:
```
#Change the tuple values? (tuple is immutable)
tup = (50,60,100,"Hello","Coding")
print("orginal tuple=",tup)
tuple[0]=500
#Note:we can not use append(),extend(),insert() functions on tuple data
# as well as we can not use remove or pop() functions
```

```
orginal tuple= (50, 60, 100, 'Hello', 'Coding')

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-33-be111195c14e> in <module>
      2 tup = (50,60,100,"Hello","Coding")
      3 print("orginal tuple=",tup)
----> 4 tuple[0]=500

TypeError: 'type' object does not support item assignment
```

In [37]:
```
#coverting list to tuple
li= [10,20,30,40,50,60]
print(type(li))

tup = tuple(li)           #type conversion
print(type(tup))

li1 = list(tup)
print(type(li1))

str1 = str(tup)
print(type(str1))
```

```
<class 'list'>
<class 'tuple'>
<class 'list'>
<class 'str'>
```

In [44]:
```
#Sort tuple in python:
#sorted(arg1)
tup = (100,90,75,1,9,23,45,67,91,15,28)
print(tup)
print(type(tup))
tup1= sorted(tup)
print(type(tup1))
print(tup1)
```

```
(100, 90, 75, 1, 9, 23, 45, 67, 91, 15, 28)
<class 'tuple'>
<class 'list'>
[1, 9, 15, 23, 28, 45, 67, 75, 90, 91, 100]
```

In [43]:
```
def sortTuple(tup):
    sort = sorted(tup)
    return sort
tup = (100,90,75,1,9,23,45,67,91,15,28)
print("before sorted tuple:",tup)
sortedtuple = sortTuple(tup)
print("after sorted Tuple:",sortedtuple)
```

```
before sorted tuple: (100, 90, 75, 1, 9, 23, 45, 67, 91, 15, 28)
after sorted Tuple: [1, 9, 15, 23, 28, 45, 67, 75, 90, 91, 100]
```

In [45]:
```
#explore the functions in tuple:
print(dir(tuple),end=" ")
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewa
rgs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__l
e__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__red
uce_ex__', '__repr__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__s
ubclasshook__', 'count', 'index']
```

In [55]:
```
#count and index in tuple():
tup = (1,2,3,4,5,1,2,3,6,9,10,5,10,3,10)
print(tup.count(1))             #for getting frequecy of the values
print(tup.count(10))
print(tup.count(3))
print(tup.index(2))         #for getting index of the values
print(tup.index(4))
```

```
2
3
3
1
3
```

In [67]:
```
#Basic Operations (concatenation,min,max,sum,sorted)
tup1 = (3,6,"KITS")
tup2 = (9,12,"APSSDC")
print(tup1+tup2)
tup = (10,100,1000,30,3,60,90,-1,0)
print(min(tup))
print(max(tup))
print(len(tup))
print(sorted(tup))
print(sum(tup))
```

```
(3, 6, 'KITS', 9, 12, 'APSSDC')
-1
1000
9
[-1, 0, 3, 10, 30, 60, 90, 100, 1000]
1292
```

In [71]:
```
#deleting a tuple:
tup = (1,2,3,4,5,6,7,8,9)
li = [1,2,3,4,5,6,7,8]
del li
del tup
```

In [72]:
```
print(tup)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-72-805acd190a95> in <module>
----> 1 print(tup)
      2 print(li)

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

In [73]:
```
print(li)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-73-a6b754927a10> in <module>
----> 1 print(li)

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

In [ ]:
```
#Sets in Python:
   A set is an Unordered Collection of items.(Its doesnot follows the index)
    #Key Points:
       1. Every set element is unique.
       2. Its Doest allow the duplication.
       3. sets are itself is mutable, we can add or remove items from it.
       4. We can also perform mathematical operations
              union, intersection...etc
       5. sets are represented by {} or curly braces.
```

In [75]:
```
#How to create sets?
s1 = {}
print(type(s1))
s1 = set()
print(type(s1))                 #creating empty set
```

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

In [83]:
```
#How to create set with values?
s1 = {1,2,3,4,5,2,3,4,5,61,2,3,4,5,7,9,5,6,9,2,1,3}
#print(s1)               #sets doest not allow duplicate values.
s2 = {10,20,30,"sets","tuples","dicts",50.5}
print(s2)
s2[1]
#Note: we can not perform indexing and slicing on set items
 #because its doest follow the order.
```

```
{'dicts', 10, 'tuples', 50.5, 20, 'sets', 30}

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-83-5d361c302c4b> in <module>
      4 s2 = {10,20,30,"sets","tuples","dicts",50.5}
      5 print(s2)
----> 6 s2[1]

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

In [97]:
```
#Modifiying a set? (add(),update())
s = {10,20,30,40,50,60,70,80}
print(len(s))
s.add(100)   #to add a single element into set
s.add(500.6)
s.add((10,100,500))
s.add("hello")
print(type(s))
print(s)
```

```
8
<class 'set'>
{(10, 100, 500), 100, 70, 40, 10, 80, 50, 20, 500.6, 'hello', 60, 30}
```

In [106]:
```
#Update():#To add multiple elements as separately.
s = {10,20,30,40,50,60,70,80}
s.update(["python","TCs","Google","AI",5000,60.5])
s.update([90,80,70,60,50,10,20])
s.update("python")
print(s)
print(len(s))
```

```
{5000, 10, 20, 30, 'TCs', 'p', 40, 'Google', 50, 'o', 60.5, 60, 't', 70, 80,
'n', 90, 'y', 'AI', 'python', 'h'}
21
```

In [124]:
```
#removing elements from the sets:
s= {5000, 10, 20, 30, 70, 80, 'n', 90, 'y', 'AI', 'python', 'h'}
print("initial set=",s)
print(len(s))
s.remove(90)              #to delete specific element
s.remove("python")
s.remove(5000)
s.remove("AI")
s.discard(20)
print("after removing=",s)
print(len(s))
```

```
initial set= {70, 5000, 10, 80, 'h', 'python', 20, 'n', 'AI', 90, 'y', 30}
12
after removing= {70, 10, 80, 'h', 'n', 'y', 30}
7
```

In [147]:
```
#pop() operation: (It deletes the items from set randomly)
s= {5000, 10, 20, 30, 70, 80, 'n', 90, 'y', 'AI', 'python', 'h'}
#s.pop()
#s.pop()
s.clear()            #clear the all contents at a time
print(s)
del s                           #deletes the set entirely
print(s)
```

```
set()

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-147-1bd9e7f3e8f6> in <module>
      6 print(s)
      7 del s
----> 8 print(s)

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

In [148]:
```
#Set Operations:
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 [151]:
```
#union() and intersection():
setA = {1,2,3,4,5}
setB = {4,5,6,7,8}
print(setA|setB)          #union means set of all elements of both sets
print(setA.union(setB))
print(setB|setA)
print(setB.union(setA))
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [156]:
```
#Intersection():
setA = {1,2,3,4,5}
setB = {4,5,6,7,8}
print(setA&setB)
print(setA.intersection(setB))
print(setB&setA)
print(setB.intersection(setA))
```

```
{4, 5}
{4, 5}
{4, 5}
{4, 5}
```

In [ ]:
```

```