# Data Science & AI
# &
# NIC - Param

## Python-For Data Science

## Set & Tuple

## One Shot

# Recap of Previous Lecture

**Topic** List Part-02

# Topics to be Covered

Topic Sets

Topic Tuples

# Topic : Sets and Tuples

tuple

1) Same as list but immutable
2) Ordered
3) Indexing
4) slicing
5) Heterogen.
6) Duplicates allowed
7) (1, 2, 3)

t = (1, 2, 3)
0 1 2

t[0] 1
t[1] 2


bar. is optional

print(type(a))

modify

sorted → string
sorted → tuple

```
l = [1,2,3,40,-5]
l.sort()
```

by default ⇒ increasing
(Non-decreasing)

min( ), max( ) → list, tuple

# Sets

1) Duplicates not allowed
2) Unordered → but we can sort
3) No indexing concept
4) No slicing concept
5) Heterogen. ✓
6) Mutable

```
a = (1,2,3)
b = a
```

```
a = 11
b = a
```

```
a = [1,2,3]
b = a
```

```
a = 1,2,3
```

{1,2,3}
{2,1,3} — same set
{3,2,1}

ordered ✗

## Twist

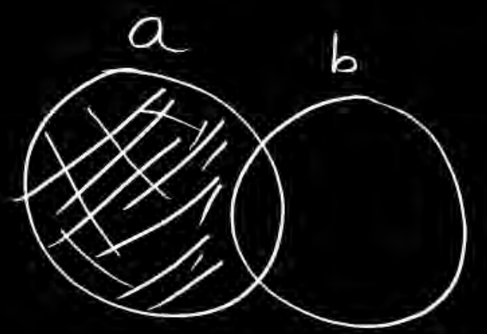
{1,2,3}

s = {}

Not a set
⇓
it is a dictionary

# Empty set  s = set()

a - b = {set of ele in set a
which are not in b}



Problem solving

1. dictionary

Problem solving

In [1]:
```
a=1,2 #parenthesis is optional
#as we are assigning multiple values to a variable by default it becomes a tuple
print(type(a))
```

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

In [2]:
```
b=()
print(type(b))
```

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

In [3]:
```
a=(10 + 20)*30
print(a)
b=(10 + 20) *(30)
print(b)
```

```
900
900
```

In [4]:
```
#parenthesis expresssion
#parenthesis value ke sath   ===> (30) allowed
#   (30)  ===>value
t=(10)
print(type(t))
```

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

In [5]:
```
#with single element u cant create a tuple like above
t1=(1,) #now it becomes a tuple
print(type(t1))
```

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

In [11]:
```
#t=() represent empty tuple
t=()
s=(1,)
a=(1,2,3,4)
b=tuple([1,2,3,4])
c=tuple("pankaj")
print(t,s,a,b,c)
```

```
() (1,) (1, 2, 3, 4) (1, 2, 3, 4) ('p', 'a', 'n', 'k', 'a', 'j')
```

In [12]:
```
#just like list and string ==> + works as concatenation and * works as repeatition
#operator
a=(1,2,3)
b=a*3
print(b)
```

```
(1, 2, 3, 1, 2, 3, 1, 2, 3)
```

In [13]:
```
a=(1,2,3)
b=(4,5,6)
c=a+b     #concatenation
print(c)
```

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

In [14]:
```
#indexing and slicing same as list and string
a=(1,2,3,"pankaj")
a[0]
```

Out[14]:
```
1
```

In [15]:
```
a[3]
```

Out[15]:
```
'pankaj'
```

In [16]:
```
a[100]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[16], line 1
----> 1 a[100]

IndexError: tuple index out of range
```

In [17]:
```
a=(10,20,30,"pankaj",12.34)
a[2:] #from index 2 till end of the tuple
```

Out[17]:
```
(30, 'pankaj', 12.34)
```

In [18]:
```
#immutable
a=(1,2,3,4)
a[0]="pankaj"  #Error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[18], line 3
      1 #immutable
      2 a=(1,2,3,4)
----> 3 a[0]="pankaj"

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

In [19]:
```
a=(1,2,3) + [1,2,3]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[19], line 1
----> 1 a=(1,2,3) + [1,2,3]

TypeError: can only concatenate tuple (not "list") to tuple
```

In [20]:
```
a=[1,2,3] + (1,2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[20], line 1
----> 1 a=[1,2,3] + (1,2)

TypeError: can only concatenate list (not "tuple") to list
```

In [21]:
```
#important function on tuple
#add,remove,insert ==>no such methods on tuple
#len ===>list,string,tuple,....
a=(1,2,3,4)
print(len(a)) #no. of elements in a
```

```
4
```

In [22]:
```
b=((1,2),"pankaj") # 2 elements are there
#1st element is a tuple
#2nd element is a string
print(len(b))
```

```
2
```

In [23]:
```
b[0]
```

Out[23]:
```
(1, 2)
```

In [24]:
```
#count() ==>to count the number of occurences of an element in tuple
a=(1,2,3,1,2,1,1,1,1)
print(a.count(1))
```

```
6
```

In [25]:
```
print(a.count(12))
```

```
0
```

In [26]:
```
a=(1,2,3,4)
b=list(a)
print(b)
```

```
[1, 2, 3, 4]
```

In [28]:
```
a=(12,3,45,0)
b=sorted(a)#a list is returned
print(b)#with sorted data a list is created
print(a)# a is same as it is immutable
```

```
[0, 3, 12, 45]
(12, 3, 45, 0)
```

In [29]:
```
a=sorted("pankaj")
print(a)
```

```
['a', 'a', 'j', 'k', 'n', 'p']
```

In [30]:
```
a=(1,2,3,"pankaj")#we can not apply sorted bcz elements are not of same type
```

In [31]:
```
sorted(a)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[31], line 1
----> 1 sorted(a)

TypeError: '<' not supported between instances of 'str' and 'int'
```

In [32]:
```
a=(12,3,45,-900,1)
b=sorted(a)
print(b)
```

```
[-900, 1, 3, 12, 45]
```

In [33]:
```
a=(12,3,45,-900,1)
b=sorted(a,reverse=True)
```

```
print(b)
```

```
[45, 12, 3, 1, -900]
```

In [34]:
```
max(a)
```

Out[34]:
```
45
```

In [35]:
```
min(a)
```

Out[35]:
```
-900
```

In [36]:
```
#packing and unpacking
a=10
b="pankaj"
c=20
d=12.34
t=a,b,c,d #a,b,c,d are packed into a tuple
print(t)
```

```
(10, 'pankaj', 20, 12.34)
```

In [37]:
```
#unpacking
e,f,g,h=t
print(e,f,g,h)
```

```
10 pankaj 20 12.34
```

In [38]:
```
a,b=t #error  RHS is a pack of 4 elements
#at the time of unpacking ===>problem
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[38], line 1
----> 1 a,b=t

ValueError: too many values to unpack (expected 2)
```

In [39]:
```
#comprehension concepts does not works for tuple
#Quries ??
```

In [40]:
```
#empty set
s=set()
print(type(s))
```

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

In [41]:
```
a={1,2}
print(type(a))
```

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

In [42]:
```
s=set("pankaj")#5 elements 'a' will come only once
print(s)
```

```
{'k', 'n', 'a', 'p', 'j'}
```

In [43]:
```
s=set([1,2,1,2,1,3,4])
```

In [44]:
```
s
```

Out[44]:
```
{1, 2, 3, 4}
```

In [45]:
```
a=set(range(1,10))
print(a)
```

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

In [46]:
```
#functions
#set is dynamic==>add,remove an element
a=set()
a.add(1)
```

In [47]:
```
a
```

Out[47]:
```
{1}
```

In [48]:
```
a.add(2)
a
```

Out[48]:
```
{1, 2}
```

In [49]:
```
a.add(1)#no effect of adding a duplicate ,even no error
```

In [50]:
```
a
```

Out[50]:
```
{1, 2}
```

In [51]:
```
a.add([1,2,34])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[51], line 1
----> 1 a.add([1,2,34])

TypeError: unhashable type: 'list'
```

In [52]:
```
a.add("pankaj")
```

In [53]:
```
a
```

Out[53]:
```
{1, 2, 'pankaj'}
```

In [54]:
```
a.add((1,2,3))
```

In [55]:
```
a
```

Out[55]:
```
{(1, 2, 3), 1, 2, 'pankaj'}
```

In [56]:
```
a.add([1,2,3])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[56], line 1
----> 1 a.add([1,2,3])

TypeError: unhashable type: 'list'
```

In [57]:
```
#update()
a={1,2,3,"pankaj"}
a
```

Out[57]:
```
{1, 2, 3, 'pankaj'}
```

In [58]:
```
a.update([10,20,30])
```

In [59]:
```
a
```

Out[59]:
```
{1, 10, 2, 20, 3, 30, 'pankaj'}
```

In [60]:
```
a.update(12.34)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[60], line 1
----> 1 a.update(12.34)

TypeError: 'float' object is not iterable
```

In [61]:
```
a={1,2,3,4}
b=a.copy()
```

In [62]:
```
b
```

Out[62]:
```
{1, 2, 3, 4}
```

In [63]:
```
a
```

Out[63]:
```
{1, 2, 3, 4}
```

In [64]:
```
#set==>unordered
#pop() ===>remove and return any element(not fix) ie. randomly
a.pop()
a
```

Out[64]:
```
{2, 3, 4}
```

In [65]:
```
a.remove(2)
```

In [66]:
```
a
```

Out[66]:
```
{3, 4}
```

In [67]:
```
#remove==>not return any thing
s=set()
s.pop()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[67], line 3
      1 #remove==>not return any thing
      2 s=set()
----> 3 s.pop()

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

In [68]:
```
s={1,2,3}
s.remove(12)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[68], line 2
      1 s={1,2,3}
----> 2 s.remove(12)

KeyError: 12
```

In [69]:
```
s={1,2,34,50}
s.discard(100)
```

In [70]:
```
s.clear()
```

In [71]:
```
s
```

Out[71]:
```
set()
```

In [73]:
```
#union,intersection
a={1,2,3}
b={4,5,6}
print(a.union(b)) #no set is updated but the result is a union b
print(a)
```

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

In [75]:
```
s=a.union(b)
t=b.union(a)
print(s,t)
```

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

In [76]:
```
a|b
```

Out[76]:
```
{1, 2, 3, 4, 5, 6}
```

In [77]:
```
#intersection
a.intersection(b)
```

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

In [78]:
```
a={1,2,3}
b={1,34,56}
```

In [79]:
```
a.intersection(b)
```

Out[79]:
```
{1}
```

In [80]:
```
b.intersection(a)
```

Out[80]:
```
{1}
```

In [81]:
```
a&b
```

Out[81]:
```
{1}
```

In [82]:
```
#difference
a={1,2,3}
b={1,23,45}
a.difference(b)
```

Out[82]:
```
{2, 3}
```

In [83]:
```
a-b
```

Out[83]:
```
{2, 3}
```

In [85]:
```
a^b
```

Out[85]:
```
{2, 3, 23, 45}
```

In [86]:
```
a.symmetric_difference(b)
```

Out[86]:
```
{2, 3, 23, 45}
```

In [87]:
```
#no indexing
#no slicing
#comprehension
s=(2*i for i in range(1,10))
print(s)
```

```
<generator object <genexpr> at 0x0000028705801700>
```

In [88]:
```
s
```

Out[88]:
```
<generator object <genexpr> at 0x0000028705801700>
```

In [ ]:

# THANK - YOU