In [1]:
```
#practice programms
#lists in python
#tuple in python
#dictionary in python
```

In [ ]:
```

```

In [2]:
```
#list is a orderd collection of data items
#mutable (can be change after creation
#allows duplicate elements
# enclosed in square brackets [ ]
#create a empty list
l =[]
l
```

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

In [3]:
```
type(l)
```

Out[3]:
```
list
```

In [4]:
```
#creating list using variables
name="nx"
age="20"
persentage=80.60
person_list=[name,age,persentage]
print(person_list)
```

```
['nx', '20', 80.6]
```

In [5]:
```
pet="dog"
breed="germen shepherd"
pet_name="reily"
age=5
my_pet=[pet,breed,pet_name,age]
my_pet
```

Out[5]:
```
['dog', 'germen shepherd', 'reily', 5]
```

In [6]:
```
#len() append() pop()
```

In [7]:
```
my_list=[1,2,3,4,5]
lenght=len(my_list)
lenght
```

Out[7]:
```
5
```

In [8]:
```
my_pet
lenght=len(my_pet)
lenght
```

Out[8]:
```
4
```

In [9]:
```
my_list.append(100)
my_list
```

Out[9]:
```
[1, 2, 3, 4, 5, 100]
```

In [10]:
```
my_list.append(10)
my_list
```

Out[10]:
```
[1, 2, 3, 4, 5, 100, 10]
```

In [11]:
```
my_list.pop()
my_list #by default last element removed
```

Out[11]:
```
[1, 2, 3, 4, 5, 100]
```

In [12]:
```
my_pet.pop(-2)
my_pet  #using backwqard indexing -2 reily removed
```

Out[12]:
```
['dog', 'germen shepherd', 5]
```

In [13]:
```
#insert(), remove(), sort(), reverse()
```

In [14]:
```
new_list=[10,20,30,30,10]
```

In [15]:
```
new_list.insert(1,40)
```

```
new_list    #inseted at the position 1
            #insert expected two arguments,
```

Out[15]: [10, 40, 20, 30, 30, 10]

In [16]:
```
new_list.insert(-1,20)
new_list
```

Out[16]: [10, 40, 20, 30, 30, 20, 10]

In [17]:
```
new_list.sort()
new_list #puts on ascending order
        #sort can not be use where string and numbers come together
```

Out[17]: [10, 10, 20, 20, 30, 30, 40]

In [18]:
```
words=["karn", "arju", "bheem", "duryodhan", "bheeshma" ]
```

In [19]:
```
words.sort
words #ascending order
```

Out[19]: ['karn', 'arju', 'bheem', 'duryodhan', 'bheeshma']

In [20]:
```
new_list.reverse()
new_list #it reverse the list
```

Out[20]: [40, 30, 30, 20, 20, 10, 10]

In [21]:
```
words.reverse()
words
```

Out[21]: ['bheeshma', 'duryodhan', 'bheem', 'arju', 'karn']

In [22]:
```
#slicing
words[0:3]
#we dont use slice word insted of this use slice to print portion of the list
```

Out[22]: ['bheeshma', 'duryodhan', 'bheem']

In [23]:
```
new_list[1:4]
```

Out[23]: [30, 30, 20]

In [24]:
```
#forward indexing
#backward indexing
print(new_list)
print(new_list[-1])
print(new_list[-2])
print(new_list[-3])
print(new_list[-4])
print(new_list[-5])
print(new_list[-6])
print(new_list[-7])
```

```
[40, 30, 30, 20, 20, 10, 10]
10
10
20
20
30
30
40
```

In [25]:
```
print(new_list)
print(new_list[0])
print(new_list[1])
print(new_list[2])
print(new_list[3])
print(new_list[4])
print(new_list[5])
print(new_list[6])
```

```
[40, 30, 30, 20, 20, 10, 10]
40
30
30
20
20
10
10
```

In [26]:
```
a=[1,2,3,4,5,6,7,8,9,10]
```

```
a
```

Out[26]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [27]:
```
a[0:8] [:-2][:2] #print at a time
```

Out[27]:
```
[1, 2]
```

In [28]:
```
a.index(6)
# the index no. of element 6 is 5
```

Out[28]:
```
5
```

In [29]:
```
index1=a.index(5)
index1
```

Out[29]:
```
4
```

In [30]:
```
a.copy()
```

Out[30]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [31]:
```
a.copy()
```

Out[31]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [32]:
```
a.count(1)
#gives it element and that will tell how many times it repeated.
```

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

In [33]:
```
a.count(-1)
```

Out[33]:
```
0
```

In [34]:
```
a.remove(10)
a #10 removed
  #given element 10
```

Out[34]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [ ]:
```

```

TUPLE

In [35]:
```
'''
-->> orderd collection of items.
-->> immutable (cannot be change after creation).
-->> allows duplicate elements
-->> enclosed in parenthesis ( ) '''
```

Out[35]:
```
'\n-->> orderd collection of items.\n-->> immutable (cannot be change after creation).\n-->> allows duplicate e
lements \n-->> enclosed in parenthesis ( ) '
```

In [36]:
```
#CREATING EMPTY TUPLE
t = ()
t
```

Out[36]:
```
()
```

In [37]:
```
type(t)
```

Out[37]:
```
tuple
```

In [38]:
```
#creating tuple using
# tuple constructure
my_list=[1,2,3,4,5]
my_tuple =tuple(my_list)
my_tuple
```

Out[38]:
```
(1, 2, 3, 4, 5)
```

In [39]:
```
#create tuple using variables
car_name="dodge"
model="hellcat"
colour="dark red"
hp=786
my_car=(car_name,model,colour,hp)
my_car
```

Out[39]:
```
('dodge', 'hellcat', 'dark red', 786)
```

In [40]:
```
movie_name="leo"
actor ="vijay"
movie_number=67
tagline= "bloody sweet"
movie=(movie_name,actor,movie_number,tagline)
movie
```

Out[40]:
```
('leo', 'vijay', 67, 'bloody sweet')
```

In [41]:
```
#ressigning tuple
my_tuple
```

Out[41]:
```
(1, 2, 3, 4, 5)
```

In [42]:
```
my_tuple=(15,16,17,18,19)
my_tuple #this is possible , we cant change in tuple but we can simply resign it.
```

Out[42]:
```
(15, 16, 17, 18, 19)
```

In [43]:
```
#  len(),|| count() ,|| index()
```

In [44]:
```
words= ("salman","sharukh","amir","sanju","jackie")
words
```

Out[44]:
```
('salman', 'sharukh', 'amir', 'sanju', 'jackie')
```

In [45]:
```
len(words)
```

Out[45]:
```
5
```

In [46]:
```
my_tuple.count(15) #count 15 one time
```

Out[46]:
```
1
```

In [47]:
```
my_tuple.count("jackie")
```

Out[47]:
```
0
```

In [48]:
```
words.index("amir")
```

Out[48]:
```
2
```

In [49]:
```
words.index("sanju")
```

Out[49]:
```
3
```

In [50]:
```
dir({})
```

Out[50]:
```
['__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__ror__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'clear',
 'copy',
 'fromkeys',
 'get',
 'items',
 'keys',
 'pop',
 'popitem',
 'setdefault',
 'update',
 'values']
```

In [ ]:

SET

In [51]:
```
#CREATING SET
SET1 ={}
SET1
```

Out[51]:
```
{}
```

In [52]:
```
type(SET1)
```

Out[52]:
```
dict
```

In [53]:
```
'''
-->> unorderd collection of items
-->> mutable
-->> does't allow duplicate element
-->> enclosed in curly braces { }'''
```

Out[53]:
```
" \n-->> unorderd collection of items \n-->> mutable\n-->> does't allow duplicate element \n-->> enclosed in cu
rly braces { }"
```

In [54]:
```
#set creation using set constructor
my_set = set([1,2,3,4,5])
my_set
```

Out[54]:
```
{1, 2, 3, 4, 5}
```

In [55]:
```
new_set={100,200,300,400,500}
new_set
```

Out[55]:
```
{100, 200, 300, 400, 500}
```

In [56]:
```
#len() ||   add() ||   remove() || pop()
```

In [57]:
```
len(new_set)
```

Out[57]:
```
5
```

In [58]:
```
new_set.add(600)
new_set
```

Out[58]:
```
{100, 200, 300, 400, 500, 600}
```

In [59]:
```
new_set.remove(500)
new_set
```

Out[59]:
```
{100, 200, 300, 400, 600}
```

In [60]:
```
new_set.pop()
new_set #remove randomly
```

Out[60]:
```
{100, 200, 300, 600}
```

In [61]:
```
new_set
```

Out[61]:
```
{100, 200, 300, 600}
```

In [62]:
```
new_set.discard(600)
new_set
```

Out[62]:
```
{100, 200, 300}
```

In [ ]:
```

```

In [63]:
```
# union |  ||   intersection &  ||   difference -    ||  symmetric_difference ^
```

In [64]:
```
a={1,2,3,4,5,6,7,8,9}
b={1,2,3,4,5}
c={10,11}
```

In [65]:
```
a.union(b)
```

Out[65]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [66]:
```
a|b
```

Out[66]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [67]:
```
a.intersection(b)
```

Out[67]:
```
{1, 2, 3, 4, 5}
```

In [68]:
```
a&b
```

Out[68]:
```
{1, 2, 3, 4, 5}
```

In [69]:
```
a&c
```

Out[69]:
```
set()
```

In [70]:
```
a.difference(b)
```

Out[70]:
```
{6, 7, 8, 9}
```

In [71]:
```
a-b
```

Out[71]:
```
{6, 7, 8, 9}
```

In [72]:
```
b-c
```

Out[72]:
```
{1, 2, 3, 4, 5}
```

In [73]:
```
a-c
```

Out[73]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [74]:
```
a.symmetric_difference(b)
```

Out[74]:
```
{6, 7, 8, 9}
```

In [75]:
```
a^b
```

Out[75]:
```
{6, 7, 8, 9}
```

In [76]:
```
a^c
```

Out[76]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
```

In [77]:
```
b^c
```

Out[77]:
```
{1, 2, 3, 4, 5, 10, 11}
```

In [78]:
```
a.issuperset(b)
```

Out[78]:
```
True
```

In [79]:
```
a.isdisjoint(c)
```

Out[79]:
```
True
```

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

Out[80]:
```
True
```

In [81]:
```
b.issuperset(a)
```

Out[81]:
```
False
```

In [82]:
```
b.isdisjoint(c)
```

Out[82]:
```
True
```

In [ ]:
```

```

dictionary

In [83]:
```
'''
-->> unorderd collection of key value pairs
-->> mutable
-->> keys must be unique and immutable
(like number , string or tuple )
-->> enclosed in curly braces { } , with key value pairs
saprated by colons.
```

```
  Cell In[83], line 1
    '''
    ^
SyntaxError: incomplete input
```

In [ ]:
```
a={"name":"jon",age:30,"city":"new york"}
a
```

In [ ]:
```
type(a)
```

In [ ]:
```
a.items()
```

In [ ]:
```
a.keys()
```

In [ ]:
```
a.values()
```

In [ ]:
```
a.copy()
```

In [ ]:
```
a.pop(5)
a
```

In [ ]:
```
a.get("jon")
a
```

In [ ]:
```
''' here is short note code for tuple , lists , srt snd dict '''
```

In [ ]:
```

```