Learning Tuple : here we can't add,delete,change, Tuple used to store multiple items in a asingle variable (Duplicate allowed), it is a collection which is oredred and unchangeable. tuple represented with () bracket. we cannot predict the order sequence

In [9]:
```
tup=(1,7,8)
tup
```

Out[9]:
```
(1, 7, 8)
```

In [10]:
```
type(tup)
```

Out[10]:
```
tuple
```

In [14]:
```
tup*3
```

Out[14]:
```
(1, 7, 8, 1, 7, 8, 1, 7, 8)
```

In [16]:
```
tup[0]=2     # can not to updated
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_11872/3268092060.py in <module>
----> 1 tup[0]=2     # can not to updated

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

In [ ]:
```
del tup  # can not be deleted
```

In [ ]:
```
tup
```

Learning Slicing :slicing refers to accessing a specific portion or a subset of the list for some operation while the orginal list remains unaffected.

In [17]:
```
tup[1:]    # here,will get all values after index no.1
```

Out[17]:
```
(7, 8)
```

In [18]:
```
tup[0:]
```

Out[18]:
```
(1, 7, 8)
```

Learnin = Set - it's a collection which is unordered & unindexed . NO duplicate memebrs,it represented by { } curly bracket, it will give UNIQUE ELEMENTS.

In [19]:
```
set1={ 'Ram','Sam','Venus'}
print(set1)
```

```
{'Venus', 'Sam', 'Ram'}
```

In [20]:
```
type(set1)
```

Out[20]:
```
set
```

In [21]:
```
tup*2
```

Out[21]:
```
(1, 7, 8, 1, 7, 8)
```

In [ ]:
```
# as above tup elements got repeated ,will convert in set and see what happen?
```

In [22]:
```
set(tup)  # repeated elements got filtered
```

Out[22]:
```
{1, 7, 8}
```

In [23]:
```
abc=set(tup)
```

In [24]:
```
abc
```

Out[24]:
```
{1, 7, 8}
```

In [25]:
```
tuple(abc)   # we can not predict order sequence
```

Out[25]:
```
(8, 1, 7)
```

In [26]:
```
list(abc)
```

Out[26]:
```
[8, 1, 7]
```

In [27]:
```
set(abc)  # Will get values in order
```

Out[27]:
```
{1, 7, 8}
```

learning comparision operator

In [28]:
```
1<5
```

Out[28]:
```
True
```

In [29]:
```
1>5
```

Out[29]:
```
False
```

In [30]:
```
3==4
```

Out[30]:
```
False
```

In [31]:
```
3==3
```

Out[31]:
```
True
```

In [32]:
```
1>=2
```

Out[32]:
```
False
```

In [33]:
```
'sam'=='sam'
```

Out[33]:
```
True
```

Learning logical operators

In [34]:
```
(1<8) and (3>5)       # while using 'and operator' it will highlight false statment only.
```

Out[34]:
```
False
```

In [35]:
```
(1<8) or (3>5)        # while using 'or operator' it will highlight true statment only.
```

Out[35]:
```
True
```

In [36]:
```
(1<8) and (3>5) or (4>5)     # under 'and'operator both condition should be true otherwise will get false
```

Out[36]:
```
False
```

In [37]:
```
(1<8) or (3>5) and (4>5)    # under'or' operator any of condition is true will get ans as true
```

Out[37]:
```
True
```

learning = if....else condition & elif

In [38]:
```
if 2>4:
    print('2 is greater than 4')
    print('end condition')
print('The End')
```

```
The End
```

In [39]:
```
if 2>4:
    print('2 is greater that 4')
else:
    print('2 is less than 4')
print('The End')
```

```
2 is less than 4
The End
```

In [40]:
```
if 10>4:
    print('10 is greater that 4')
else:
    print('10 is less than 4')
print('The End')
```

```
10 is greater that 4
The End
```

In [41]:
```
if 10<4:
    print('10 is greater that 4')
else:
    print('10 is less than 4')
print('The End')
```

```
10 is less than 4
The End
```

multiple condition

In [42]:
```
if 10<4:
    print('10 is greater than 4')
elif 7<6:
    print('7 is greater than 6')
else:
    print('10 is less than 4')
print('The End')
```

```
10 is less than 4
The End
```

Learning= FOR loop

In [43]:
```
list1=[2,0,4,7,8,9]
```

In [44]:
```
list1
```

Out[44]:
```
[2, 0, 4, 7, 8, 9]
```

In [45]:
```
for item in list1:
    print(item)
print('end')
```

```
2
0
4
7
8
9
end
```

In [46]:
```
for i in list1:               # square  root of each value
    print(i*i)
```

```
4
0
16
49
64
81
```

Learning 'while loop'- With the while loop we can execute a set of statements as long as a condition is true.

In [47]:
```
i=3                # we have define 3 value to i variable
while i<=8:
    print(i)
    i=i+2    # so here we know i=3, now it will get it added like 3+2=5,5+2=7 ....till max.value as '8'
print("end")
```

```
3
5
7
end
```

In [ ]:
```
i=3
while i<=8:
    print(i)
    i=i+1      # so here we know i=3, now it will get it added like 3+1=4,   4+1=5 ....till max.value as '8'
print("end")
```

Learning 'Range'-The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

In [48]:
```
for a in range(5):   # range start from '0' and end at'4 excluding '5'
    print(a)
```

```
0
```

```
1
2
3
4
```

In [ ]:
```
range(7)
```

In [49]:
```
for x in range (0,7):
    print(x)
```

```
0
1
2
3
4
5
6
```

In [ ]:
```
list1=[4,6,9,8]
print(list1)
```

In [50]:
```
for x in range (0,4):
    print(list1[x])
```

```
2
0
4
7
```

List Comprehension - List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

In [51]:
```
[i*i for i in list1]
```

Out[51]:
```
[4, 0, 16, 49, 64, 81]
```

Learning 'function'-A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result. it will define as 'def'

In [52]:
```
def square(var):        # we have make function for square root and named as 'square'
    return var*var
```

In [53]:
```
square(3)         # we have to write 'square' and value will get ans.
```

Out[53]:
```
9
```

In [54]:
```
square(9)
```

Out[54]:
```
81
```

In [55]:
```
def cube(var):        # we have make function for cube and named as 'cube'
    return var*var*var
```

In [56]:
```
cube(2)
```

Out[56]:
```
8
```

In [57]:
```
cube(9)
```

Out[57]:
```
729
```

In [58]:
```
def sum(a,b):      # we have make function for summation and named as 'sum'
    return a+b
```

In [59]:
```
sum (12,67)
```

Out[59]:
```
79
```

In [60]:
```
sum(9,7)
```

Out[60]:
```
16
```

In [61]:
```
def sum(x=2,y=8):
    return (x-y,x+y)
```

In [62]:
```
sub,add=sum()
```

In [63]:
```
sub
```

Out[63]:
```
-6
```

In [64]:
```
add
```

Out[64]:
```
10
```

Learning The map() function : it executes a specified function for each item in an iterable. The item is sent to the function as a parameter.

In [65]:
```
list1
```

Out[65]:
```
[2, 0, 4, 7, 8, 9]
```

In [66]:
```
map(square,list1)     #got NO value because we have not assigned map with data type i.e.list
```

Out[66]:
```
<map at 0x13ec3c67760>
```

In [67]:
```
list(map(square,list1))   # got the result after applying data type
```

Out[67]:
```
[4, 0, 16, 49, 64, 81]
```

In [69]:
```
def stringadd (a,b,):
    return (a+b)
```

In [70]:
```
x=map(stringadd,('red','blue','green'),('black','orange','yellow'))
```

In [71]:
```
print(list(x))
```

```
['redblack', 'blueorange', 'greenyellow']
```

Learning Filter :Filter() is a built-in function in Python. The filter function can be applied to an iterable such as a list or a dictionary and create a new iterator. This new iterator can filter out certain specific elements based on the condition

that you provide very efficiently.

In [78]:
```
def even(n):      # created function to get even nos. ( %2==0)
    if n%2==0:
        return n
```

In [79]:
```
seq1=[5,8,3,9,4]
```

In [81]:
```
list(filter(even,seq1))
```

Out[81]:
```
[8, 4]
```

In [82]:
```
list(map(even,seq1)) # check by using 'map' in map function odd nons. definec as None
```

Out[82]:
```
[None, 8, None, None, 4]
```

In [85]:
```
def odd(n):      # created function to get odd nos. ( %2!=0)
    if n%2!=0:
        return n
```

In [86]:
```
list(filter(odd,seq1))
```

Out[86]:
```
[5, 3, 9]
```

also we can write true,false with if,else function

In [93]:
```
def even(n):      # created function to get even nos. ( %2==0)
    if n%2==0:
        return True
    else:
        return False
```

In [94]:
```
list(filter(even,seq1))
```

Out[94]:
```
[8, 4]
```

In [95]:
```
list(map(even,seq1))
```

Out[95]:
```
[False, True, False, False, True]
```