TUPLE & its FUNCTION

1).count()

2).index()

In [2]:
```
t=()#Empty Tuple, Tuple start from open and close bracket
t
```

Out[2]:
```
()
```

In [3]:
```
print(type(t))
```

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

In [4]:
```
t1 = (10,20,30,40,40)
t1
```

Out[4]:
```
(10, 20, 30, 40, 40)
```

In [5]:
```
len(t1)
```

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

In [6]:
```
t1[0]
```

Out[6]:
```
10
```

In [7]:
```
t1[0:4]
```

Out[7]:
```
(10, 20, 30, 40)
```

In [8]:
```
t1[:4]#it wil take all the element because 4 indx are not present
      #but more than the indexs given
```

Out[8]:
```
(10, 20, 30, 40)
```

In [9]:
```
t1
```

Out[9]:
```
(10, 20, 30, 40, 40)
```

In [41]:
```
t1[0:0:0]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[41], line 1
----> 1 t1[0:0:0]

ValueError: slice step cannot be zero
```

In [43]:
```
t1[1:1:1]#start & stop index is same, means the tuple is empty
```

Out[43]: ()

In [45]:
```
t1[1:2:1]
```

Out[45]: (20,)

In [47]:
```
t1
```

Out[47]: (10, 20, 30, 40, 40)

In [49]:
```
t1[1:3:-1]#slicing direction is incorrect
```

Out[49]: ()

In [51]:
```
t1
```

Out[51]: (10, 20, 30, 40, 40)

In [53]:
```
t1[3:1:-1]#for backward slicing, you shoud start from a higher indx.
```

Out[53]: (40, 30)

In [55]:
```
t1
```

Out[55]: (10, 20, 30, 40, 40)

In [57]:
```
t1[::1]
```

Out[57]: (10, 20, 30, 40, 40)

In [59]:
```
t1[:1:1]
```

Out[59]: (10,)

In [61]:
```
t1[1:-1:1]
```

Out[61]: (20, 30, 40)

In [63]:
```
t1
```

Out[63]: (10, 20, 30, 40, 40)

In [65]:
```
t1[::-1]
```

Out[65]: (40, 40, 30, 20, 10)

In [67]:
```
t1[::]
```

Out[67]: (10, 20, 30, 40, 40)

In [69]:
```
t1[:]
```

Out[69]: (10, 20, 30, 40, 40)

In [71]:
```
t1[4:3:2]#cause the start ind is greater than stop indx
```

Out[71]: ()

In [73]:
```
t1[4:3:-1]#step is negative(it will move backward)
```

Out[73]: (40,)

In [75]:
```
t2=(10,20,30,40,50)
t2
```

Out[75]: (10, 20, 30, 40, 50)

In [77]:
```
t2[4:3:-1]
```

Out[77]: (50,)

In [79]:
```
t1.count(20)
```

Out[79]: 1

In [81]:
```
t1.index(30)
```

Out[81]: 2

In [83]:
```
t3=(20,2.5,"sk",True,1+2j)
t3
```

Out[83]: (20, 2.5, 'sk', True, (1+2j))

In [85]:
```
t1[1]=34#Tuple is immutable or unchangeable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[85], line 1
----> 1 t1[1]=34

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

In [87]:
```
t2=t3+t1#nested tuple
t2
```

Out[87]: (20, 2.5, 'sk', True, (1+2j), 10, 20, 30, 40, 40)

In [89]:
```
t1
```

Out[89]: (10, 20, 30, 40, 40)

In [91]:
```
t4 = t1 * 3 #tuple is immutable but it allow the element to repeat
```

In [93]:
```
t4
```

Out[93]: (10, 20, 30, 40, 40, 10, 20, 30, 40, 40, 10, 20, 30, 40, 40)

In [95]:
```
t4[:0]
```

Out[95]: ()

In [97]:
```
t4[4:3:1]
```

Out[97]: ()

In [99]:
```
t4[3:4:1]#cause it is a tuple comma will come with one value.
```

Out[99]: (40,)

In [101…
```
 t4[4:3:-1]
```

Out[101… (40,)

In [103…
```
t4
```

Out[103… (10, 20, 30, 40, 40, 10, 20, 30, 40, 40, 10, 20, 30, 40, 40)

In [105…
```
t4[::1]
```

Out[105… (10, 20, 30, 40, 40, 10, 20, 30, 40, 40, 10, 20, 30, 40, 40)

In [107…
```
t4[:-1:]
```

Out[107… (10, 20, 30, 40, 40, 10, 20, 30, 40, 40, 10, 20, 30, 40)

In [109…
```
t4
```

Out[109… (10, 20, 30, 40, 40, 10, 20, 30, 40, 40, 10, 20, 30, 40, 40)

In [117…
```
t3
```

Out[117… (20, 2.5, 'sk', True, (1+2j))

In [127…
```
for i in t3:
    print(i)
```

```
20
2.5
sk
True
(1+2j)
```

In [125…
```
for i in enumerate(t3):
```

```
    print(i)
```

```
(0, 20)
(1, 2.5)
(2, 'sk')
(3, True)
(4, (1+2j))
```

In [ ]:

In [ ]: