Tuple Data Structure:-

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets. Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also.

Tuple Creation

In [8]:
```
student = ("Sneha","Singh",25,5.6)
print(student)
```

```
('Sneha', 'Singh', 25, 5.6)
```

In [9]:
```
student[0:2]# Tuple Slicing
```

Out[9]:
```
('Sneha', 'Singh')
```

In [10]:
```
del student[2] #Tuple Deleting is not possible as it is immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8092\2262557108.py in <module>
----> 1 del student[2] #Tuple Deleting is not possible as it is immutable

TypeError: 'tuple' object doesn't support item deletion
```

In [11]:
```
# Tuple are immutable which means we can't CHANGE tuple items
student[2] = 3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8092\2319482190.py in <module>
      1 # Tuple are immutable which means we can't CHANGE tuple items
----> 2 student[2] = 3

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

In [12]:
```
del student # Deleting entire tuple object is possible
student # already deleted
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8092\3318104884.py in <module>
      1 del student # Deleting entire tuple object is possible
----> 2 student # already deleted

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

Count

In [13]:
```
# Number of times items occured in tuple.
count_alpha =('A','B','C','D','E','F','G','B','B','A')
count_alpha.count('B')
```

Out[13]:
```
3
```

Tuple Membership

In [14]:
```
"F" in count_alpha #Check if one exist in the list
```
Out[14]:
```
True
```

Index Position

In [15]:
```
count_alpha.index("C")
```
Out[15]:
```
2
```

Sorting

In [19]:
```
sortTuple = (12,18,98,34,56,23,975,54,1,6,3)
# Sorted new tuple doesnt change the original tuple.False is set by default no need to menti
sorted(sortTuple,reverse = False)
```
Out[19]:
```
[1, 3, 6, 12, 18, 23, 34, 54, 56, 98, 975]
```

In [21]:
```
sorted(sortTuple,reverse = True)# Sort in descending order
```
Out[21]:
```
[975, 98, 56, 54, 34, 23, 18, 12, 6, 3, 1]
```

Nested Tuple

In [6]:
```
nest_tuple = (1,4,6,3,(10.4,23.4,54.3),("Sneha","Singh","Daughter"))
nest_tuple
```
Out[6]:
```
(1, 4, 6, 3, (10.4, 23.4, 54.3), ('Sneha', 'Singh', 'Daughter'))
```

In [3]:
```
# Access the "Singh"
nest_tuple[5][1]
```
Out[3]:
```
'Singh'
```

In [4]:
```
#Access the "u" in the Daughter
nest_tuple[5][2][2]
```
Out[4]:
```
'u'
```

In [7]:
```
# update
nest_tup2 =((1,2,3,4),"A","B","C",["SNEHA","SINGH"])
nest_tup2
```
Out[7]:
```
((1, 2, 3, 4), 'A', 'B', 'C', ['SNEHA', 'SINGH'])
```

In [8]:
```
# Update "SNEHA" with "Lovely"
nest_tup2 [4][0]="Lovely"
nest_tup2
```
Out[8]:
```
((1, 2, 3, 4), 'A', 'B', 'C', ['Lovely', 'SINGH'])
```

Interpretation:

Python recognizes that "SNEHA" is sit inside nested list within tuple.So, all the rules applicable to list will apply and lists are mutable,the update is possible