Tuples Functions

- **We can use loops to iterate through a tuple:** For example, if we want to print all the elements in the tuple. We can run the following loop.

```
myTemp=(1,2,3)
for t in myTemp:
      print(t)
```

Output:

```
1
2
3
```

- **Checking whether the tuple contains a particular element:** The keyword used is `in`. We can easily get a boolean output using this keyword. It returns `True` if the tuple contains the given element, else the output is `False`.

```
>>> 10 in myTemp
False
>>> 4 in myTemp
True
```

- **Finding the length of a tuple:** We can easily find the number of elements that any tuple contains. The keyword used is `len`.

```
>>> print(len(myTemp))
6
```

- **Concatenation:** We can add elements from two different tuples and form a single tuple out of them. This process is concatenation and is similar to data types like string and list. We can use the **" + "** operator to combine two tuples.

```
a = (1,2,3,4)
b = (5,6,7,8)
d = a+b
print(d)
Out[]: (1,2,3,4,5,6,7,8)
```

We can also combine two tuples into another tuple in order to form a nested tuple. This is done as follows:

```
a = (1,2,3,4)
b = (5,6,7,8)
d = (a, b)
print(d)
--> ((1,2,3,4),(5,6,7,8))
# Here d is a nested tuple which is formed from 2 different tuples.
```

- **Repetition of a Tuple:** We can repeat the elements from a given tuple into a different tuple. The operator used is **" * "**, the multiplication operator. In other words, by using this operator, we can repeat the elements of a tuple as many times as we want.

```
>>> a = (1,2,3,4)
>>> print(a*4)
(1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4)
# The same tuple elements are repeated 4 times in the form of another tuple.
```

- **Finding minimum or maximum element in a tuple:** To find the maximum element in a tuple, the keyword used is `max` and for finding the minimum element, the keyword used is `min`. These keywords return the maximum and the minimum elements of the tuple, respectively.

```
>>> a = (1,2,3,4)
>>> print(min(a))
1
>>> print(max(a))
4
```

Note: We can find the minimum and maximum of only those tuples, which have comparable entries. We cannot compare two different data types like a string and a tuple. Such comparisons would throw an error.

For example:

```
>>> s= (1,2,"string",4)
>>> print(min(s))
TypeError: '<' not supported between instances of 'string' and 'int'
```

```
>>> e= (1,2,2.6,4)
>>> print(min(e))
1
# Even though the data types are different , however since a float can be compared to a floating-point value, hence this operation does not give an error.
```

- **Converting a list to a tuple:** We can typecast a list into a tuple. The keyword used is `tuple`. This is done as follows:

```
>>> myList= [1,2,3,4]
>>> myTuple= tuple(myList)
>>> print(myTuple)
(1,2,3,4)
```