2] Tuple data-type:

Tuple is a collection of similar or different type data enclosed within round bracket ( ) and elements are separated by 'comma'

Syntax:

```
tuple_name = (val1, val2, val3, ..., valn)
```

Example:

```
tuple = ('1', '7', '13', '19', '22', '25')
tuple2 = ('hii', 37.4, True, 'None', None)
tuple3 = ('On', 7-4j, 0b101, [1, 2, 0x1A])
```

Tuple are immutable data-type i.e. we can modify the tuple element but we cannot change the size of tuple (can't insert or remove elements)

There are various operations that can be performed on tuple.

Since, tuple are immutable we cannot change even the element of tuple, some operations cannot be performed on tuple

1) Changing or updating tuple can't be done methods like append(), insert(), extend() can't be used.

2) Deleting element of tuple del(), clear(), pop(), remove() are not applicable to tuple.

1) Creating or Declaring a Tuple:

We can create or declare Tuple using following syntax

```
tuple_name = (val1, val2, val3, ..., valN)
```

where tuple name is a valid variable and values can be any-data type variables, input, expression.

Example:

```
# tuple of similar data-type
tuple1 = (2, 24, 28, 216, 232, 264)
# tuple of mixed data-type
tuple2 = ('hii', None, True, 2.74, 0b101)
tuple3 = ([1,2,3], (2,2,2), {0,1,23}, 0xAZ)
```

2) Accessing element of Tuple:

As tuple is a collection of elements. we can access each and every element. There are three ways to access the elements,

a) Using Positive Index:

Syntax:

```
tuplename[index]
```

by puting the index we can access that element.

Example:

```
tuple = (2.4, 3.14, 9.8, 4.1)
print(tuple[0]) ; print(tuple[3])
print(tuple[2]) ; print(tuple[1])
```

b) Using negative Index:

Syntax:

tuple name [negative index]

Example:

```
tuple_name=('sam', 'adam', 'Jhon', 'Luke')
print(tuple_name[-4])
print(tuple_name[-1])
print(tuple_name[-3])
```

c) Using slicing [:] operator:

Syntax:

tuple-name [start : ending index]

by using this method we can access more than one element from starting to ending index.

Example:

```
tuple=(1, 2, 3, 4, 5.5, 6.6, 7, 'H', 'K', 'N')
print(tuple[0:5]) ; print(tuple[-10:-5])
print(tuple[5:9] ; print(tuple[-5:-1])
```

3] Tuple Operations:

There are two operations that can be performed on tuple.

a) Concatenation '+': used to add two or more tuple to form sing tuple

Example:

```
tuple1=(2, 4, 6, 8, 10) ; tuple2=(0, 1, 3, 5, 7, 9)
tuple3 = tuple1 + tuple2
print(tuple3)
```

b) Repetition '*': used to repete two or more times a particular tuple.

Example:

```
tuple1 = (1, 2, 3, 4, 5)
print(tuple1 * 3)
tuple2 = ('a', 'e', 'i', 'o', 'u')
print((tupl + tuple) * 2)
```

4] Different Methods or Built in function on Tuple

There various Methods or Buit in function that can be Performed on ___. Tuple.

1) Count(): count method return the count of particular element of Tuple.

tuple_name.count(element)

Example:

```
tuple = (1, 2, 3, 4, 'hi', '101', 2, 3, 2.2, 3, 1, 3)
tuple.count(2) ; tup.count(3)
print(tuple.count('XYZ'))
print(tuple.count('101'))
```

2) Index(): Index function return the index of a particular elem.

tuple_name.index(element)

Example:

```
tuple = ('a', 'e', 'i', 'o', 'u')
print(tuple.index('a'))
print(tuple.index('e'))
print(tuple.index('o'))
print(tuple.index('None'))
```

3) len(): Length function return the length i.e. total no. elem. present in tuple.

len(tuple-name)

Example:

```
tuple1 = (2, 4, 6, 8, 10)
tuple2 = (1, 3, 5, 7, 9)
tuple3 = tuple1 + tuple2
print(len(tuple1)); print(len(tuple2)
print(len(tuple3))
```

4) max(): maximum function return the maximum or largest element of the tuple.

max(tuple-name)

Example:

```
tuple1 = (1, 1.11, 1.111, 1.1111)
tuple2 = (-1, -1.11, -1.111, -1.1111)
print(max(tuple1)); print(max(tuple
print(max(tuple1 + tuple2))
```

5) min(): minimum function return the minimum or least element of tuple.

min(tuple-name)

Example:

```
tuple1 = (1, 1.11, 1, 1.11, 1.111)
tuple2 = (-1, -1.1, -1.11, -1.111)
print(min(tuple1))
print(min(tuple2))
print(min(tuple1 * 2))
```

6) sum(): sum function return the sum of all element of tuple.
elem. of tuple should be of same data-type

sum(tuple)

Example:

```
tuple_integers = (2, 4, 6, 8, 10)
print(sum(tuple_integers))
string = ('L', 'U', 'C', 'K', 'Y')
print(sum(string))
```

7) all(): return true value when all element are true or tuple is empty else return False value.

Empty - T

all(tuple_name)

Example:

```
tuple = (True, bool(1), 57, 24, ('i', 'know'))
tuple2 = (0, 0.0, 0.00, 0.1)
print(all(()))  ; print(all(tuple))
print(all(tuple2))
```

8) any(): this function - return the True value when atleast one value of tuple is True else it return false value.

Empty - F

any(tuple_name)

Example:

```
tuple = (True, 5, 8, 6-4j, False)
tup2 = (2, 4, 8, 10, 14, 12)
print(any(tuple))
print(any(tup2))
```

9) tuple(): tuple is a builtin function used to convert other data-type or sequence of values into tuple form.

tuple(data or sequence)

Example:

```
sequence = 'a', 'e', 2, 7-4j, 0b11
print(tuple(sequence))
list = [2, 4, 6, 8, 10]
print(tuple(list))
```

- Tuple Comprehension:

As tuple Immutable, it is not possible to perform Tuple comprehension. But indirectly using tuple function it can be done.

An elegent way of Performing Tuple comprehension is by using asterix and comma

Syntax: *(item for item in range( )),

Example:

```
sequence = *(i for i in ('a','e','i','o','u')),
print(sequence)
group = tuple((i for i in range(1,11) if i%2==0))
print(group)
collection = ['even' if n%2==0 else 'odd' for
                n in range(11)]
print(tuple(collection))
```