TUPLE

A tuple is a sequence of comma separated values (items) enclosed in round brackets. The items in a tuple need not have the same type. For example, consider this tuple with multiple values defined using commas:

T1 = ('Aman', 12, 4500, 'Jatin', 34, 8793)

T2 = (12, 'Monica', 89.8, 'Shilpa', (45, 67), 78.5)

T3 = ('E001', ['D01', T05'], 23, 12)

Tuple	**List**
Tuple is immutable i.e., its contents cannot be updated or changed.	List is mutable i.e., its contents can be updated or changed.
The elements in tuple are within parentheses.	The elements in List are within square brackets.
Example, T1 = ('Manish', 12, 345)	Example, L1 = ['Manish', 12, 345]

An empty tuple is written as two parentheses with no content, just like a list with two square brackets.

To write a tuple with single value, you have to include a comma.

T1 = ()

T2 = (23,)

Note: Like string indices, tuple indices also start at 0, and can be sliced and concatenated too. Basic Tuple Operations

Expression	**Results**	**Description**
(1, 5, 6) + (7, 8, 10)	(1,5,6,7,8,10)	**Concatenation** – joins the tuple elements into a single tuple
('Welcome') * 2	('Welcome', 'Welcome')	**Repetition** – repeats the tuple members
4 in (3, 4, 5) 2 in (3, 4, 5) 2 not in (3, 4, 5)	True False True	**Membership** – returns True if the element is a member of tuple otherwise False.

```
>>> T1 = (1, 5, 6) # Defines tuple T1
>>> T2 = (7, 8, 10) # Defines tuple T2
>>> T1 + T2 # Concatenation
(1, 5, 6, 7, 8, 10)
>>> ('Welcome') * 2 # Repetition
'WelcomeWelcome'
>>> 5 in T1 # Membership
True
>>> 5 in T2 # Membership
False
>>> 5 not in T2 # Membership
True
```

Tuple Slicing

Slicing operation	**Explanation**
T1[1]	It returns the second item in the tuple
T1[1:4]	It returns a subset of the original members. It starts accessing elements from the first index and stops before the fourth index.
T1[1:-2]	It starts accessing elements from the first index and stops at the third index, i.e., the third element from the right.
T1[: 3]	Omitting the first index means that it starts accessing elements from the beginning, i.e., 0th index and stops at index 2.
T1[2:]	Omitting the second index means that it starts accessing elements from the second index, i.e., 2 and stops at the end.
T1[2] = ‘garima’	You cannot change the contents of a tuple as it is an immutable object.
del T1	It removes an entire tuple. You cannot remove individual elements of a tuple

```
>>> T1 = ('monica', 12, 'seema', (13, 17)) # Defines tuple T1
>>> T1[1:4] # It returns the elements from index 1 to 3.
(12, 'seema', (13, 17))
>>> T1[1:-2] # It returns the elements from index 1 to -3.
(12,)
>>> T1[ :3] # It returns the elements from index 0 to 2.
('monica', 12, 'seema')
>>> T1[2: ] # It returns the elements from index 2 to end.
('seema', (13, 17))
>>> T1[2] = 'garima' # It will give an error because tuple is immutable.
Traceback (most recent call last):
  File "<pyshell#234>", line 1, in <module>
    T1[2] = 'garima' # It will give an error because tuple is immutable.
TypeError: 'tuple' object does not support item assignment
>>> del T1 # It deletes the tuple T1
```

Built-in Tuple Functions

Function	Description
len(t1)	It returns the number of elements in the tuple.
max(t1)	It returns the item from the tuple with maximum value.
min(t1)	It returns the item from the tuple with minimum value.
sum(t1)	It returns the sum of all the elements in the tuple.
list(t1)	It converts the tuple to list.

```
>>> T1 = (23, 12, 17, 15) # Defines tuple T1
>>> len(T1) # It returns number of elements in a tuple T1.
4
>>> max(T1) # It returns the maximum value in tuple T1.
23
>>> min(T1) # It returns the minimum value in tuple T1.
12
>>> sum(T1) # It returns the sum of values in tuple T1.
67
```

function and Explanation	Example

T = tuple()
It creates an empty tuple. It is similar to T = ()
	`>>> T = tuple()`
	`>>> T`
	`()`

T**.count(**obj**)**
Returns how many times obj occurs in the tuple.
	`>>> T = ( 23, 12, 44, 12, 45, 12)`
	`>>> T.count(12)`
	`3`

T.**index(**obj**)**
Returns the index of first occurrence of obj
	`>>> T = ( 23, 12, 44, 12, 45, 12)`
	`>>> T.index(45)`
	`4`

T1 = sorted(T)
It takes the tuple as parameter and creates a new list containing the elements in sorted order
 	
Note: sort() and copy() are not available in tuple because it is immutable object.
	`>>> T = (56,34,23,11,90)`
	`>>> T1 = sorted(T)`
	`>>> T1`
	`[11, 23, 34, 56, 90]`
	`>>> T1 = sorted(T, reverse = True)`
	`>>> T1`
	`[90, 56, 34, 23, 11]`