Touples in Python

The word "tuple" is derived from mathematics, where it refers to a finite ordered list of elements. In programming, a tuple is a data structure that is similar to a list, but with some key differences. Just like in mathematics, a tuple in programming is an ordered collection of elements, but unlike a list, a tuple is immutable, meaning that its elements cannot be changed once the tuple is created.
The idea of tuples as a data structure was first introduced in the programming language LISP, and it has since been adopted by many other programming languages, including Python. In Python, a tuple is defined using parentheses, with its elements separated by commas. For example:

In [1]:
```
t = (10, 20, 30)
#We can access the elements of a tuple in the same way you would with a list, using square brackets and an
#index:
t[0] #return 10
```

Out[1]:
```
10
```

In [2]:
```
#let check if we can alter touple value #impossible
t[0] = 45;
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[2], line 2
      1 #let check if we can alter touple value #impossible
----> 2 t[0] = 45

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

As shown above the error is that tuple object does not support item assignment because it is immuatble, once it is created we can't alter its values. let see some other examples,

In [3]:
```
t=(1)
type(t)
```

Out[3]:
```
int
```

why this return int, this is because when we use a single element of any type in brackets as (1) or ("hello") it is not considered as touple in python instead it will be consider as int or string as per value, to make a single value topule we have to add a trailing comma at end of that element as,

In [4]:
```
t = (1,)
print(type(t))
```

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

now it return us touple object. this is how we can create a single element touple

In [5]:
```
#duplicate values in touple
t = (1,1,2,2,3,3,44,5)
print(t)
```

```
(1, 1, 2, 2, 3, 3, 44, 5)
```

yes touple do support doublicate values as the elements of touple are stored in same order as they are insreted and because elements are indexed they can have multiple same values as shown above

Adding, Removing, Replacing element in Touple

Touples are Immutable which means any operation which is going to change its element, size are not allowed, like adding,removing,replacing items. In case you have a touple and you want to perform such operation on touple you have to first convert that touple into list, perform your operation on list and convert the list back to touple as,

In [6]:
```
t = ("Monday", "Tuesday","Wednesday","Thursday","Friday","Saturday")
#in above touple we don't have Sunday so what if we want to add sunday into it, now if your touple
#contains only limited values simply create new touple and add that value in as shown below in commed line,

#t = ("Monday", "Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday")

#but what if your touple contains complex data and have huge number of values, like there is a chance your created
#topule from list element which has complex data and huge no of elements, in taht case simply convert you
#toup into list as,

myWeekDayList = list(t)

#now perform operation on list as,

myWeekDayList.append("Sunday")

#print list if you want to check
# print(myWeekDayList)

#convert your list back to touple as,

t = tuple(myWeekDayList)

#lets print touple
```

```
print(t)
#lets print last element of tuple
print(t[-1])
```

```
('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday')
Sunday
```

Assigning multiple values to multiple variables using touple (Unpacking Touple)

In [7]:
```
t = (1,2,3)
x,y,z = t
print(x)
print(y)
print(z)
```

```
1
2
3
```

```
This is one of the fantastic thing I have ever seen, In many programming language they don't support multiple
variables on left side of assignment operator but python does, something make me interested in python.
as shown above we can easily assign touple values to variables.
```

In [8]:
```
#wait wait wait, you have to take care than the number of element in tuple must be euqal to the number
#of variables passed on left side of assignment operator, if the values in topule are less than variables
#we will get error as,

#not enough values to unpack (expected 3, got 2)

#if the values are more than the variables used in left side of assingment operator we will get error as,

#too many values to unpack

#so be careful while unpacking, assigning value from tuple to variables.
t = (1,2,4,5)
x,y,z = t
print(x)
print(y)
print(z)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[8], line 13
      1 #wait wait wait, you have to take care than the number of element in tuple must be euqal to the number
      2 #of variables passed on left side of assignment operator, if the values in topule are less than variables
      3 #we will get error as,
   (...)
     10
     11 #so be careful while unpacking, assigning value from tuple to variables.
     12 t = (1,2,4,5)
---> 13 x,y,z = t
     14 print(x)
     15 print(y)

ValueError: too many values to unpack (expected 3)
```

Combining Two Tuple in Single Tuple

In [9]:
```
#we can combine two seperate touple in one tuple object as we have done in list using + operator as,
tuple1 = (1,2,3,4)
tuple2 = (99,191)
tuple3 = tuple1 + tuple2
print(tuple3)
```

```
(1, 2, 3, 4, 99, 191)
```

Multiplying Tuple With Const Value

In [13]:
```
#we can also multiply a tuple with any const value, which will mutiply the num of occurence of each element as
tuple1 = (1,2,3,4)
tuple3 = tuple1 * 5
print(tuple3)
```

```
(1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4)
```

Sorting a tuple

In [25]:
```
#since a tuple is unchangable it can't be sorted and it does not have any sort method, but there is a
#function (global function) which takes an iteratable and return a sorted list (keep in mind list) so we
#can sort a touple by using that method and recreaste a tuple object from that returning list object as,

tuple3 = (1,9,8,44,5,0)
print(tuple3)
tuple3 = tuple((sorted(tuple3))) #here we first sort the tuple values using sorted() which returned list, and using
#tuple() constructor function we simply convert that list into tuple again
print(tuple3)
```

```
(1, 9, 8, 44, 5, 0)
(0, 1, 5, 8, 9, 44)
```

Counting the number of occurence of an element in tuple

we can use a tuple.count(element) method to check how many times a specific element occur in our touple

In [35]:
```
tuple1 = (1,2,3,4,)
print(tuple1.count(1))
tuple1 = tuple1 * 50
print(tuple1.count(1))
```

```
1
50
```

Finding the index of element in tuple:

```
To find the index of specific element in tuple we can use index() method of tuple as,
```

In [36]:
```
tuple1 = (1,2,3,4,)
print(tuple1.index(4))
```

```
3
```

In [ ]: