(3.4) Tuple

Introduction

Tuple is Immutable List

but items in tuple can be mutable

Tuple is exactly same as List except that it is immutable i.e once we creates Tuple object,we cannot perform any changes in that object Hence Tuple is Read Only version of List.

If our data is fixed and never changes then we should go for Tuple.

Insertion Order is preserved

Duplicates are allowed

Heterogeneous objects are allowed

support both +ve and -ve index

+ve index means forward direction(from left to right) and -ve index means backward direction(from right to left)

Representation

We can represent Tuple elements within Parenthesis and with comma seperator.Parenethesis are optional but recommended to use.

Example:-

In [6]:
```
t = 10,20,30,40
print(t, type(t))

t = (10,20,30,40)
print(t, type(t))
```

```
(10, 20, 30, 40) <class 'tuple'>
(10, 20, 30, 40) <class 'tuple'>
```

In [7]:
```
t = 10,20,30,40,
print(t,type(t))

t = (10,20,30,40,)
print(t,type(t))
```

```
(10, 20, 30, 40) <class 'tuple'>
(10, 20, 30, 40) <class 'tuple'>
```

Note: We have to take special care about single valued tuple.compulsary the value should ends with comma,otherwise it is not treated as tuple.

In [5]:
```
t = 10
print(t, type(t))

t = 10,
print(t, type(t))
```

```
10 <class 'int'>
(10,) <class 'tuple'>
```

Which are valid tuples?

In [10]:
```
t=(); print(type(t)) #tuple
t=10,20,30,40; print(type(t)) #tuple
t=10; print(type(t)) #int
t=10,; print(type(t)) #tuple
t=(10); print(type(t)) #int
t=(10,); print(type(t)) #tuple
t=(10,20,30,40); print(type(t)) #tuple
```

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

Tuple creation

1. t=()

creation of empty tuple

2. t=(10,)

t=10,
creation of single valued tuple ,parenthesis are optional,should ends with comma

3. t=10,20,30

t=(10,20,30)
creation of multi values tuples & parenthesis are optional

4. By using tuple() function:

list=[10,20,30]
t=tuple(list)

t=tuple(range(10,20,2))

In [11]:
```
list=[10,20,30]
t=tuple(list)
print(t)

t=tuple(range(10,20,2))
print(t)
```

```
(10, 20, 30)
(10, 12, 14, 16, 18)
```

Accessing elements of tuple

By using index

In [13]:
```
t=(10,20,30,40,50,60)
print(t[0]) #10
print(t[-1]) #60
```

```
10
60
```

In [14]:
```
print(t[100])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_9004/4127211835.py in <module>
----> 1 print(t[100])

IndexError: tuple index out of range
```

By using slice operator

In [15]:
```
t=(10,20,30,40,50,60)
print(t[2:5])
print(t[2:100])
print(t[::2])
```

```
(30, 40, 50)
(30, 40, 50, 60)
(10, 30, 50)
```

Tuple vs immutability

Once we creates tuple,we cannot change its content.Hence tuple objects are immutable

In [16]:
```
t=(10,20,30,40)

t[1] = 70
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_9004/388461840.py in <module>
      1 t=(10,20,30,40)
      2
----> 3 t[1] = 70

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

Tuple with mutable object

In [42]:
```
t = ( 1, 3, 5, [ "hello", "hi"], 'good' )

print(t, type(t))
```

```
print(t[3][1])
```

```
(1, 3, 5, ['hello', 'hi'], 'good') <class 'tuple'>
hi
```

we can change in mutable part

In [43]:
```
t[3][1] = 'world'
print(t[3][1])
```

```
world
```

In [46]:
```
t = ( 1, 3, 5, [ "hello", "hi"], 'good' )

x = t
print(t, id(t))
print(x, id(x))

t = t[::-2]
print(t, id(t))
print(x, id(x))
```

```
(1, 3, 5, ['hello', 'hi'], 'good') 2325331778672
(1, 3, 5, ['hello', 'hi'], 'good') 2325331778672
('good', 5, 1) 2325299871616
(1, 3, 5, ['hello', 'hi'], 'good') 2325331778672
```

In [49]:
```
t = ( 1, 3, 5, [ "hello", "hi"], 'good' )
t[3].append('hacked it')
```

In [50]:
```
print(t, id(t))
```

```
(1, 3, 5, ['hello', 'hi', 'hacked it'], 'good') 2325299097824
```

Mathematical operators for tuple

We can apply + and * operators for tuple

Concatenation Operator(+)

In [17]:
```
t1=(10,20,30)
t2=(40,50,60)
t3=t1+t2
print(t3) # (10,20,30,40,50,60)
```

```
(10, 20, 30, 40, 50, 60)
```

Multiplication operator or repetition operator(*)

In [18]:
```
t1=(10,20,30)
t2=t1*3
print(t2) #(10,20,30,10,20,30,10,20,30)
```

```
(10, 20, 30, 10, 20, 30, 10, 20, 30)
```

Important functions of tuple

In [20]:
```
print(dir(tuple))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'count', 'index']
```

A. len()

In [21]:
```
t=(10,20,30,40)
print(len(t)) #4
```

```
4
```

B. count()

In [22]:
```
t=(10,20,10,10,20)
print(t.count(10)) #3
```

```
3
```

C. index()

In [23]:
```
t=(10,20,10,10,20)
print(t.index(10)) #0
```

```
0
```

In [24]:
```
print(t.index(30))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_9004/2668145195.py in <module>
----> 1 print(t.index(30))

ValueError: tuple.index(x): x not in tuple
```

D. sorted()

In [26]:
```
t=(40,10,30,20)
t1=sorted(t)
print(t1)
print(t)
```

```
[10, 20, 30, 40]
(40, 10, 30, 20)
```

reverse sorting

In [27]:
```
t1=sorted(t,reverse=True)
print(t1)
```

```
[40, 30, 20, 10]
```

E. min() and max() functions:

In [28]:
```
t=(40,10,30,20)
```

```
print(min(t)) #10
print(max(t)) #40
```

```
10
40
```

F. cmp():

```
It compares the elements of both tuples.
If both tuples are equal then returns 0
If the first tuple is less than second tuple then it returns -1
If the first tuple is greater than second tuple then it returns +1

t1=(10,20,30)
t2=(40,50,60)
t3=(10,20,30)
print(cmp(t1,t2)) # -1
print(cmp(t1,t3)) # 0
print(cmp(t2,t3)) # +1
```

Note: cmp() function is available only in Python2 but not in Python 3

Tuple Packing and Unpacking

```
We can create a tuple by packing a group of variables.
```

Here a,b,c,d are packed into a tuple t. This is nothing but tuple packing

In [31]:
```
a = 10
b = 20
c = 30
d = 40

t=a,b,c,d
print(t,type(t))
```

```
(10, 20, 30, 40) <class 'tuple'>
```

Tuple unpacking is the reverse process of tuple packing

We can unpack a tuple and assign its values to different variables

In [32]:
```
t = (10,20,30,40)
a,b,c,d = t

print(a,b,c,d)
```

```
10 20 30 40
```

Note: At the time of tuple unpacking the number of variables and number of values should be same. ,otherwise we will get ValueError.

In [34]:
```
t=(10,20,30,40)
a,b,c=t
print(a,b,c)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_9004/3641926236.py in <module>
      1 t=(10,20,30,40)
```

```
----> 2 a,b,c=t
      3 print(a,b,c)

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

Tuple Comprehension

Tuple Comprehension is not supported by Python.

t= (x**2 for x in range(1,6))

Here we are not getting tuple object and we are getting generator object.

In [37]:
```
t= ( x**2 for x in range(1,6))
print(type(t))

for x in t:
    print(x)
```

```
<class 'generator'>
1
4
9
16
25
```

Differences between List and Tuple

List and Tuple are exactly same except small difference: List objects are mutable where as Tuple objects are immutable.

In both cases insertion order is preserved, duplicate objects are allowed, heterogenous objects are allowed, index and slicing are supported.

In [39]:
```
from IPython.display import Image
Image(filename="tuple_and_list.jpg")
```

Out[39]:

List	Tuple
1) List is a Group of Comma separeated Values within Square Brackets and Square Brackets are mandatory. Eg: i = [10, 20, 30, 40]	1) Tuple is a Group of Comma separeated Values within Parenthesis and Parenthesis are optional. Eg: t = (10, 20, 30, 40) t = 10, 20, 30, 40
2) List Objects are Mutable i.e. once we creates List Object we can perform any changes in that Object. Eg: i[1] = 70	2) Tuple Objeccts are Immutable i.e. once we creates Tuple Object we cannot change its content. t[1] = 70 ➔ ValueError: tuple object does not support item assignment.
3) If the Content is not fixed and keep on changing then we should go for List.	3) If the content is fixed and never changes then we should go for Tuple.
4) List Objects can not used as Keys for Dictionries because Keys should be Hashable and Immutable.	4) Tuple Objects can be used as Keys for Dictionries because Keys should be Hashable and Immutable.

In [ ]:
```
```