tuple

Tuples are immutable sequences, typically used to store collections of heterogeneous data (such as the 2-tuples produced by the enumerate() built-in). Tuples are also used for cases where an immutable sequence of homogeneous data is needed (such as allowing storage in a set or dict instance).

In application development tuples are used,

1. To represent immutable sequence
2. To represent data in set and dictionary
3. Tuple is also used by enumerate for generating data

Tuples may be constructed in a number of ways:

1. Using a pair of parentheses to denote the empty tuple: ()
2. Using a trailing comma for a singleton tuple: a, or (a,)
3. Separating items with commas: a, b, c or (a, b, c)
4. Using the tuple() built-in: tuple() or tuple(iterable)

Tuple does not provide mutable operations.

1. Append
2. Insert
3. Remove
4. Extend
5. Sort
6. Pop
7. Clear
8. Del keyword
9. Replacing

Example:

```
>>> t1=()
>>> print(t1,type(t1))
() <class 'tuple'>
l1=[]
>>> print(l1,type(l1))
[] <class 'list'>
>>> l1.append(10)
>>> print(l1)
[10]
>>> t1.append(10)
```

```
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    t1.append(10)
AttributeError: 'tuple' object has no attribute 'append'
>>> l1.insert(0,20)
>>> print(l1)
[20, 10]
>>> t1.insert(0,20)
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    t1.insert(0,20)
AttributeError: 'tuple' object has no attribute 'insert'
```

Example:

```
>>> t2=()
>>> print(t2)
()
>>> t3=(10)
>>> print(t3)
10
>>> print(type(t3))
<class 'int'>
>>> t4=(10,)
>>> print(t4,type(t4))
(10,) <class 'tuple'>
>>> t5=100,
>>> print(t5,type(t5))
(100,) <class 'tuple'>
>>> t6=(10,20,30,40,50)
>>> print(t6,type(t6))
(10, 20, 30, 40, 50) <class 'tuple'>
>>> t7=10,20,30,40,50
>>> print(t7,type(t7))
(10, 20, 30, 40, 50) <class 'tuple'>
>>> t8=tuple()
>>> print(t8,type(t8))
() <class 'tuple'>
>>> t9=tuple([10,20,30])
>>> t10=tuple(range(10,60,10))
>>> t11=tuple((10,20,30))
```

```
>>> t12=tuple("ABC")
>>> print(t9,t10,t11,t12)
(10, 20, 30) (10, 20, 30, 40, 50) (10, 20, 30) ('A', 'B', 'C')
>>> t9
(10, 20, 30)
>>> t9[0]=99
Traceback (most recent call last):
  File "<pyshell#31>", line 1, in <module>
    t9[0]=99
TypeError: 'tuple' object does not support item assignment
```

How to read content of sequence?

1. Index
2. Slicing
3. For
4. Iterator
5. Enumerate

Example

```
# Reading content of tuple
t1=(10,20,30,40,50)
# using index
for i in range(0,len(t1)): # start=0,stop=5,step=1
    print(t1[i])

for i in range(-1,-(len(t1)+1),-1): #start=-1,stop=-6,step=-1
    print(t1[i])

#using for loop

for x in t1:
    print(x)

# using slicing
t2=t1[:3]
t3=t1[3:]
t4=t1[1:-1]
print(t2,t3,t4)

# using iterator
```

```
a=iter(t1)
x=next(a)
y=next(a)
print(x,y)

# using enumerate
b=enumerate(t1)
x=next(b)
print(x)
y=next(b)
print(y)
```

Output:

```
10
20
30
40
50
50
40
30
20
10
10
20
30
40
50
(10, 20, 30) (40, 50) (20, 30, 40)
10 20
(0, 10)
(1, 20)
```

Example:

```
# Write a program to find size of tuple

import sys

t1=(10,20,30)
print(sys.getsizeof(t1))
```

```
print(len(t1))
```

Output:
64
3

Python – Maximum and Minimum K elements in Tuple
Sometimes, while dealing with tuples, we can have problem in which we need to extract only extreme K elements, i.e maximum and minimum K elements in Tuple. This problem can have applications across domains such as web development and Data Science.

Input : test_tup = (3, 7, 1, 18, 9), k = 2
Output : (3, 1, 9, 18)
Input : test_tup = (3, 7, 1), k=1
Output : (1, 7)

```
test_tup=(3,7,1,18,9)
k=2
sort_tup=tuple(sorted(test_tup))
res_tup=sort_tup[:k]+sort_tup[-k:]
print(test_tup)
print(res_tup)
```

Python program to create a list of tuples from given list having number and its cube in each tuple

Given a list of numbers of list, write a Python program to create a list of tuples having first element as the number and second element as the cube of the number.

```
Input: list1 = [1, 2, 3]
Output: [(1, 1), (2, 8), (3, 27)]


Input: list2 = [9, 5, 6]
Output: [(9, 729), (5, 125), (6, 216)]
```

```
list1=[1,2,3]
list2=[(n,n**3) for n in list1]
print(list1)
print(list2)
```

Python – Adding Tuple to List and vice – versa

Sometimes, while working with Python containers, we can have a problem in which we need to perform addition of one container with another. This kind of problem can have occurrence in many data domains across Computer Science and Programming.

```
list1=[10,20,30]
tuple1=(40,50,60)
result1=list1+list(tuple1)
result2=tuple1+tuple(list1)
print(result1)
print(result2)

list1=[1,2,3]
tuple1=(4,5,6)
list1+=tuple1
print(list1)
l1=[1,2,3]
t1=(4,5,6)
t1+=tuple(l1)
print(t1)
```

Output:

```
[10, 20, 30, 40, 50, 60]
(40, 50, 60, 10, 20, 30)
[1, 2, 3, 4, 5, 6]
(4, 5, 6, 1, 2, 3)
```

Python – Join Tuples if similar initial element

Sometimes, while working with Python tuples, we can have a problem in which we need to perform concatenation of records from the similarity of initial element. This problem can have applications in data domains such as Data Science.

```
Input  : test_list = [(5, 6), (5, 7), (5, 8), (6, 10), (7, 13)]
```

```
Output : [(5, 6, 7, 8), (6, 10), (7, 13)]
Input  : test_list = [(5, 6), (6, 7), (6, 8), (6, 10), (7, 13)]
Output : [(5, 6), (6, 7, 8, 10), (7, 13)]
```