TUPLES

Tuples is exactly the same as list but the only difference is tuples is immutable. Therefore, tuples is known as read only list.

Properties:

1. Insertion order is preserved
2. Duplicates are allowed
3. Heterogeneous objects are allowed
4. Indexing and slicing concept is applicable

- **Creating a tuple**

```
t=()
print(type(t))
print(t)
```

Output:

```
<class 'tuple'>
()
```

- **Demonstrating perseverance of insertion order and allowance of duplicates:**

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

Output:

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

- **To show heterogeneous objects are allowed:**

```
t=(10,20,30,40,20,30,10.5,'ram')
print(type(t))
print(t)
```

Output:

```
<class 'tuple'>
(10, 20, 30, 40, 20, 30, 10.5, 'ram')
```

- **To create a tuple, small brackets are optional:**

```
t=10,20,30
```

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

Output:

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

Note:

t = (10) is not a tuple.

t = (10,) is a tuple

- **Alternative way of creating a tuple**:

```
l=[10,20,30]
t=tuple(l)
print(type(t))
```

Output:

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

- **Accessing tuple elements:**

Note: For the use of indexing and slicing in any data types, insertion order must be preserved.

1. Through Indexing

```
t=(10,20,30,40)
print(t[0])
print(t[-4])
```

Output:

```
10
10
```

2. Through slicing

```
t=(10,20,30,40)
print(t[:])
print(t[2:6])
print(t[2:6:2])
```

Output:

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

- **Tuple vs Immutability:**

```
t=(10,20,30,40)
t[0]=200
print(t)
```

Output:

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

Mathematical operators for tuples:

1. + operator/ concatenation

```
t1=(10,20,30,40)
t2=(100,200,300,400)
print(t1+t2)
```

Output:

```
(10, 20, 30, 40, 100, 200, 300, 400)
```

2. * operator/ repetition operator

```
t1=(10,20,30,40)
t2=(100,200,300,400)
print(t1+t2)
print(t1*2)
```

Output:

```
(10, 20, 30, 40, 100, 200, 300, 400)
(10, 20, 30, 40, 10, 20, 30, 40)
```

3. Equality operator

```
t1=(10,20,30,40)
t2=(100,200,300,400)
print(t1==t2)
```

Output:

```
False
```

4. Membership operator

```
t1=(10,20,30,40)
t2=(100,200,300,400)
print(10 in t1)
print(10 not in t1)
```

Output:

```
True
False
```

5. Relational operator

```
t1=(10,20,30,40)
t2=(100,200,300,400)
print(t1<t2)
print(t1>t2)
```

Output:

```
True
False
```

- **Functions**

1. len()

```
t1=(10,20,30,40)
print(len(t1))
```

Output:

```
4
```

2. count()

```
t1=(10,20,30,40)
print(t1.count(10))
```

Output:

```
1
```

3. index()

```
t1=(10,20,30,40)
print(t1.index(30))
```

Output:

```
2
```

4. sort

```
t1=(10,24,302,440,56,78,2,45,6)
t=sorted(t1)
print(t)
```

Output:

```
[2, 6, 10, 24, 45, 56, 78, 302, 440]
```

5. reverse

```
t1=(10,24,302,440,56,78,2,45,6)
r=reversed(t1)
t1=tuple(r)
print(t1)
```

Output:

```
(6, 45, 2, 78, 56, 440, 302, 24, 10)
```

- **Packing and unpacking of tuples:**

1. Packing of tuples:

```
a=10
b=20
c=30
d=40
t=a,b,c,d
print(t)
print(type(t))
```

Output:

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

2. Unpacking of tuples:

```
t=(10,20,30,40)
a,b,c,d=t
print(a,b,c,d)
print(type(t))
```

Output:

```
10 20 30 40
<class 'tuple'>
```

- **Tuples Comprehension**

The process of creating tuples. Comprehension of tuples is not possible in python. If we try to do it, we will always end up creating a generator.

Note:

- List is mutable while tuple is immutable
- List comprehension is possible but tuple comprehension is not possible.