Tuples in Python

In Python, a tuple is a comma-separated sequence of values. Very similar to a list. However, there is an important difference between the two.

The main difference between tuples and lists is that lists are *mutable* and tuples are not. A mutable object is one that can be changed. An *immutable* object is one that contains a fixed value that cannot be changed. If it needs to be changed, a new object must be created.

Therefore, you would generally store different data inside a tuple than you would a list. Lists typically contain items that are like each other, whereas tuples usually contain items that are diverse in type or character. However, there is no hard and fast rule to this. But if you want to be able to update the individual items, use a list. Otherwise, use a tuple.

Basic Example:

```
data = (43, "Ram", "Python", 59, 99)
print(type(data))
print(data)
data2 = 65, 43, "Akshay", "Language"
print(type(data2))
print(data2)
```

Output

```
<class 'tuple'>
(43, 'Ram', 'Python', 59, 99)
<class 'tuple'>
(65, 43, 'Akshay', 'Language')
```

Python Empty Tuple Example

There can be an empty Tuple also which contains no object. Let us see an example of empty tuple: -

```
tuple1 = ()
print(tuple1)
# Python Single Object Tuple Example
# For a single valued tuple, there must be a comma at the end of the value.
tuple1 = (10,)
print(tuple1)
```

Output

```
()
(10,)
```

Python Tuple of Tuples Example

Tuples can also be nested; it means we can pass tuple as an element to create a new tuple. See, the following example in which we have created a tuple that contains tuples the object: -

```
tupl1 = 'a', 'mahesh', 10.56
tupl2 = tupl1, (10, 20, 30)
print(tupl1)
print(tupl2)
```

Accessing Tuple

Accessing of tuple is easy; we can access tuple in the same way as List. See, the following example: -

```
data1 = (1, 2, 3, 4)
data2 = ('x', 'y', 'z')
print(data1[0])
print(data1[0:2])
print(data2[-3:-1])
print(data1[0:])
print(data2[:2])
```

Output

```
1
(1, 2)
('x', 'y')
(1, 2, 3, 4)
('x', 'y')
```

As we know Tuples are immutable that means we can't change/update value at position

Example

```
data = (54, "Python", 21, 43, 54, "87", "Language")
data[3] = 100
print(data)
```

Output

```
Traceback (most recent call last):
  File "C:\Users\aniru\Desktop\Python Programming\tuples.py", line 2, in <module>
    data[3] = 100
TypeError: 'tuple' object does not support item assignment
```

Python Tuple Operations

Python allows us to perform various operations on the tuple. Following are the common tuple operations: -

Adding Tuples Example

Tuple can be added by using the concatenation operator (+) to join two tuples.

```
data1 = (54, "Python", 21, 43, 54, "87", "Language")
data2 = (65, 11, "Computer", "Cycle", 76, 88)
data3 = data1 + data2
print(data1)
print(data2)
print(data3)
```

Output

```
(54, 'Python', 21, 43, 54, '87', 'Language')
(65, 11, 'Computer', 'Cycle', 76, 88)
(54, 'Python', 21, 43, 54, '87', 'Language', 65, 11, 'Computer', 'Cycle', 76, 88)
```

Replicating Tuple Example

Replicating means repeating. It can be performed by using '*' operator by a specific number of times.

```
data1 = (54, "Python", 21, 43, 54, "87", "Language")
data2 = (65, 11, "Computer", "Cycle", 76, 88)
data3 = data1 * 2
data4 = data2 * 3
print(data1)
print(data2)
print(data3)
print(data4)
```

Output

```
(54, 'Python', 21, 43, 54, '87', 'Language')
(65, 11, 'Computer', 'Cycle', 76, 88)
(54, 'Python', 21, 43, 54, '87', 'Language', 54, 'Python', 21, 43, 54, '87', 'Language')
(65, 11, 'Computer', 'Cycle', 76, 88, 65, 11, 'Computer', 'Cycle', 76, 88, 65, 11, 'Computer', 'Cycle', 76, 88)
```

Python Tuple Slicing Example

A subpart of a tuple can be retrieved based on index. This subpart is known as tuple slice: -

```
data1 = (54, "Python", 21, 43, 54, "87", "Language")
print(data1[0:2])
print(data1[4])
print(data1[:-1])
print(data1[-5:])
print(data1)
```

Output

```
(54, 'Python')
54
(54, 'Python', 21, 43, 54, '87')
(21, 43, 54, '87', 'Language')
(54, 'Python', 21, 43, 54, '87', 'Language')
```

Python Tuple Deleting Example

Deleting individual element from a tuple is not supported. However, the whole of the tuple can be deleted using the **del** statement.

```
data = (54, "Python", 21, 43, 54, "87", "Language")
print(data)
del data[2]
print(data)
```

Output

```
(54, 'Python', 21, 43, 54, '87', 'Language')
Traceback (most recent call last):
  File "C:\Users\aniru\Desktop\Python Programming\tuples.py", line 3, in <module>
    del data[2]
TypeError: 'tuple' object doesn't support item deletion
```

Built-In Tuple Methods

FUNCTION	DESCRIPTION
min(tuple)	It returns the minimum value from the tuple.
max(tuple)	It returns the maximum value from the tuple.
len(tuple)	It returns the length of the tuple.
tuple(sequence)	It converts sequence to tuple.

Python Tuple min(tuple) method

This method is used to get min value from the sequence of tuple.

```
data = (10, 20, 40.6, -33, 540)
x = min(data)
print(f"Minimum value is {x}")
```

Output

```
Minimum value is -33
```

Python Tuple max(tuple) method

This method is used to get maximum value from the sequence of tuple.

```
data = (10, 20, 40.6, -33, 540)
x = max(data)
print(f"Maximum value is {x}")
```

Output

```
Maximum value is 540
```

Python Tuple len(tuple) method

This method is used to get the length of the tuple.

```
data = (10, 20, 40.6, -33, 540)
x = len(data)
print(f"Length of tuple is {x}")
```

Output

```
Length of tuple is 5
```

Python tuple(sequence) Method Example

It is used to convert sequence into tuple.

```
data1 = [54, 76, 98, "Python", 12]
print(type(data1), data1)
data2 = tuple(data1)
print(type(data2), data2)
```

Output

```
<class 'list'> [54, 76, 98, 'Python', 12]
<class 'tuple'> (54, 76, 98, 'Python', 12)
```

Why should we use Tuple? (Advantages of Tuple)

- Processing of Tuples are faster than Lists.
- It makes the data safe as Tuples are immutable and hence cannot be changed.

Practice Examples (Tuples)

1. Calculate the sum of all the numbers in a tuple.
2. Create a tuple with some numbers. Ask a random number from user and add that number to the end of the tuple.
3. Write a Python program to get the 4th element and 4th element from last of a tuple.
4. Write a Python program to find the repeated items of a tuple.
5. Write a Python program to remove a number entered by user from a tuple.
6. Write a Python program calculate the product, multiplying all the numbers of a given tuple.
   Original Tuple:
   (4, 3, 2, 2, -1, 18)
   Product - multiplying all the numbers of the said tuple: -864

For solutions, check the end of the book.