Tuples

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The main difference between the tuples and the lists is that the tuples cannot be changed unlike lists. Tuples use parentheses, whereas lists use square brackets.

1. They are immutable like strings.
2. If we try to change a value it returns an error.
3. Unlike list, Tuple is written in between set of ‘()’ –parenthesis
4. Just like lists tuples can also be used to store multiple data typed values in one single variable but you cannot modify them.

Ex: 1

# Declaring an empty tuple

t = ()

Ex: 2

# Initializing a tuple with single value

t = (5)

t = (5,)

print(t)

Ex: 3

# Tuple with multiple values

t1 = (11, 22, 33, 44, 55)

t2 = (11, 22, 33, 44, 55)

print(t1)

print(t2)

Ex: 4

# Tuple with multiple datatyped values

t = (11, 22.33, True, 'a', 'abcd')

print(t)

Ex: 5

# Tuple supports forward and reverse indexing

t = (11, 22.33, True, 'a', 'abcd')

print(t[0], t[1], t[2], t[3], t[4])

print(t[-1], t[-2], t[-3], t[-4], t[-5])

# Index fun to get index of an element

print(t.index(22.33))

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")

Ex: 6

# No two tuples can be extended directly, since they are immutable

# But we can attach two tupels into a new one

tup1 = (12, 34.56)

tup2 = ('abc', 'xyz')

# Following action is not valid for tuples

# tup1[0] = 100;

# So let's create a new tuple as follows

tup3 = tup1 + tup2

print(tup3)

Ex: 7

# Deleting a tuple

tup = ('physics', 'chemistry', 'maths', 'english')

print(tup)

del tup

print("After deleting tup : ")

#print(tup)

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")

Ex: 8

# Length of a tuple

t = (11,2,33,4,55,6)

print(len(t))

Ex: 9

# Multiplication with a tuple

t = (11,2,33,4,55,6)

print(t \* 5)

Ex:10

# if condition with a tuple

t = (11,2,33,4,55,6)

if 33 in t:

    print("Yes 33 is avaliable...")

    pass

Ex: 11

# Tuple supports loops

t = (11,2,33,4,55,6)

for i in t:

    print(i)

    pass

Ex: 12

# Tuple supports slicing

t = (11,2,33,4,55,6)

print(t[:])

print(t[::])

print(t[2:])

print(t[:2])

print(t[-2:])

print(t[:-2])

print(t[2:3])

print(t[1:4:2])

Ex: 13

# max and min functions on tuple

t = (11,2,33,4,55,6)

print(max(t))

print(min(t))

print(len(t))

Ex: 14

# To convert a tuple to list

t = (11,2,33,4,55,6)

print(t)

l = list(t)

print(l)

Ex: 15

# Nested tuples

t = (11,2,33,('a','b','c',100,200,300),4,55,6)

print(t)

# List in a tuple and tuple in a list

t1 = (11,2,33,['a','b','c',100,200,300],4,55,6)

t2 = [11,2,33,('a','b','c',100,200,300),4,55,6]

print(t1)

print(t2)

Ex: 16

# Note: All list functions and modifications are possible on a list even though the list is in a tuple.

t = (11,2,33,['a','b','c',100,200,300],4,55,6)

t[3].append(2000)

print(t)

t[3][3] = 1000

print(t)

Ex: 17

# Sorting a tuple using function sorted()

# We cannot use sort() function on a tuple

l = [1,2,3,4,5]

t = (1,22,3,24,5)

print(sorted(t))

Ex: 18

# To get no of occurences of an element in a tuple

my\_tuple = ('a', 'p', 'p', 'l', 'e',)

print(my\_tuple.count('p'))  # Output: 2

Ex: 19

# Membership test in tuple

my\_tuple = ('a', 'p', 'p', 'l', 'e',)

# In operation

print('a' in my\_tuple)

print('b' in my\_tuple)

# Not in operation

print('g' not in my\_tuple)

Ex: 21

# Using a for loop to iterate through a tuple

for name in ('John', 'Ryan', 'Tom', 'Nancy'):

    print("Hello", name)

### Advantages of Tuple over List

Since tuples are quite similar to lists, both of them are used in similar situations. However, there are certain advantages of implementing a tuple over a list. Below listed are some of the main advantages:

* We generally use tuples for heterogeneous (different) data types and lists for homogeneous (similar) data types.
* Since tuples are immutable, iterating through a tuple is faster than with list. So there is a slight performance boost.
* Tuples that contain immutable elements can be used as a key for a dictionary. With lists, this is not possible.
* If you have data that doesn't change, implementing it as tuple will guarantee that it remains write-protected.