Tuples

Like lists, tuple is another data structure supported by Python. It is very similar to lists but differs in two things.

• First, a tuple is a sequence of *immutable* objects. This means that while you can change the value of one or more items in a list, you cannot change the values in a tuple.

• Second, tuples use parentheses to define its elements whereas lists use square brackets.

Advantages of Tuple over List

- Tuples are used to store values of different data types. Lists can however, store data of similar data types.
- Since tuples are immutable, iterating through tuples is faster than iterating over a list. This means that a tuple performs better than a list.
- Tuples can be used as key for a dictionary but lists cannot be used as keys.
- Tuples are best suited for storing data that is write-protected.
- Tuples can be used in place of lists where the number of values is known and small.
- Multiple values from a function can be returned using a tuple.

Creating a Tuple

A tuple is created by placing all the items (elements) inside a parentheses (), separated by comma. The parentheses are optional but is a good practice to write it. A tuple can have any number of items and they may be of different types (integer, float, list, string etc.).

Example

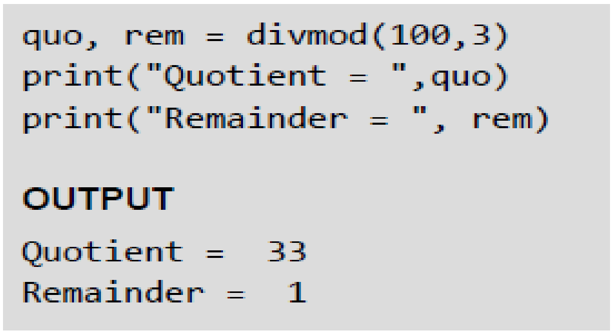
my_tuple = (1,3.2,"mouse", [8, 4, 6], (1, 2, 3))

print(my_tuple)

Utility of tuple

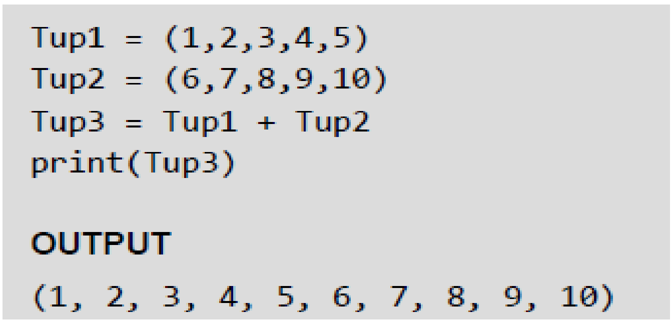
In real-world applications, tuples are extremely useful for representing records or structures as we call in other programming languages. These structures store related information about a subject together.The information belongs to different data types.

For example, a tuple that stores information about a student can have elements like roll_no, name, course, total marks, avg, etc. Some built-in functions return a tuple. For example, the divmod() function returns two values—quotient as well as the remainder after performing the divide operation.



```
quo, rem = divmod(100,3)
print("Quotient = ",quo)
print("Remainder = ", rem)

OUTPUT
Quotient =  33
Remainder =  1
```



```
Tup1 = (1,2,3,4,5)
Tup2 = (6,7,8,9,10)
Tup3 = Tup1 + Tup2
print(Tup3)

OUTPUT
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

Accessing Elements in a Tuple

We can use the index operator [] to access an item in a tuple where the index starts from 0. So, a tuple having 6 elements will have index from 0 to 5. Trying to access an element other that (6, 7,...) will raise an IndexError. The index must be an integer, so we cannot use float or other types. This will result into TypeError. Likewise, nested tuple are accessed using nested indexing.as shown in the example below.Python allows negative indexing for its sequences.

Example

```
n_tuple = (1,"mouse", [8, 4, 6], (1, 2, 3))

# Output: 's'

print(n_tuple[1][3])

# Output: 4

print(n_tuple[2][1])

# Output: 1

print(n_tuple[0])

# Output: [8, 4, 6]

print(n_tuple[-2])
```

Slicing

We can access a range of items in a tuple by using the slicing operator - colon ":".

Example

```
my_tuple = ('p','r','o','g','r','a','m')

# Output: ('r', 'o', 'g')

print(my_tuple[1:4])

# Output: ('p', 'r')

print(my_tuple[:-5])

# Output: ('a', 'm')

print(my_tuple[5:])

# Output: ('p', 'r', 'o', 'g', 'r', 'a', 'm')

print(my_tuple[:])

#('m', 'a', 'r', 'g', 'o', 'r', 'p')
```

```
print (my_tuple[::-1])

#('r', 'p')

print (my_tuple[1::-1])
```

Changing a Tuple

Unlike lists, tuples are immutable.This means that elements of a tuple cannot be changed once it has been assigned. But, if the element is itself a mutable datatype like list, its nested items can be changed. We can also assign a tuple to different values (reassignment).

Example

```
my_tuple = (4, 2, 3, [6, 5])

my_tuple[3][0] = 9

print(my_tuple)#(4, 2, 3, [9, 5])

my_tuple[1]=40#TypeError: 'tuple' object does not support item assignment

print(my_tuple)
```

We can use + operator to combine two tuples. This is also called concatenation.

We can also repeat the elements in a tuple for a given number of times using the * operator.
Both + and * operations result into a new tuple.
Example

```
# Concatenation

# Output: (1, 2, 3, 4, 5, 6)

print((1, 2, 3) + (4, 5, 6))

# Repeat

# Output: ('Repeat', 'Repeat', 'Repeat')

print(("Repeat",) * 3)
```

Deleting a Tuple

we cannot delete or remove items from a tuple.But deleting a tuple entirely is possible using the keyword del.

Example

```
my_tuple = ('p','r','o','g','r','a','m')

del my_tuple
```

Tuple Methods

Methods that add items or remove items are not available with tuple. Only the following two methods are available.

index() method

The index method takes a single argument (element) and finds the given element in a tuple and returns its position.However, if the same element is present more than once, index() method returns its first position.

Syntax

tuple.index(element)

count() method

The count() method takes a single argument(element) and counts how many times an element has occurred in a tuple and returns it.

Syntax

tuple.count(element)

Example

```
my_tuple = ('a','p','p','l','e',)

print(my_tuple.count('p'))#Output:2

print(my_tuple.index('l'))#Output:3
```

Tuple functions

all(): It takes any iterable as a parameter and return True if all elements of the tuple are true(tuple should not contains any 0 value or False value) or if the tuple is empty.

Syntax : all(iterable)

any(): It takes any iterable as a parameter and Return True if any element of the tuple is true. If the tuple is empty, return False.

Syntax: any(iterable)

len() : It takes the sequence as a parameter and return the length (the number of items) in the tuple.

Syntax: len(sequence)

sum(): Return the sum of all elements in the tuple.Normally, items of the iterable should be numbers.

Syntax: sum(iterable)

tuple(): It takes the sequence as an parameter and Convert an iterable (list, string, set, dictionary) to a tuple. If no parameters are passed, it creates an empty tuple

Syntax: tuple([iterable])

max() : The method max returns the elements from the *tuple* with maximum value.While giving an iterable as an argument we must make sure that all the elements in the iterable are of the same type.

Syntax:max(tuple)

Example

```
print (max((2,3)))

print (max(('A','b','c','B')))

print (max(("cat","bird","apple","cb")))#based on the ascii value of first letter

print (max(([1,2,3],[3,1,2],[3,3,2])))#comparing the first element of each list
```

min() : The method min() returns the elements from the *tuple* with minimum value.While giving an iterable as an argument we must make sure that all the elements in the iterable are of the same type.

Syntax:min(tuple)

sorted() : The sorted() method sorts the elements of a given iterable in a specific order - Ascending or Descending. Return a new sorted list(does not sort the tuple itself).

Syntax

sorted(tuple)

To print the values in Descending order Syntax is

sorted(tuple,reverse=True)

enumerate()

The enumerate() method adds counter to an iterable and returns it. The returned object is a enumerate object. It contains the index and value of all the items of list as a tuple.

Syntax

enumerate(iterable, start=0)

reversed()

The reversed() function returns an iterator object that accesses the given sequence in the reverse order.we have to convert that iterator object into tuple by using tuple function.

Syntax

reversed(seq)

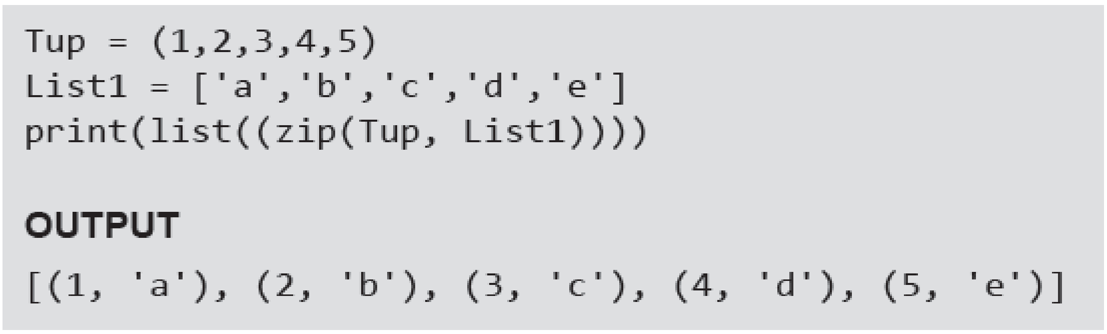
Example

```
pyTuple =(20,55,43,22,67,90,0)
print (all(pyTuple))
print (any(pyTuple))
print (len(pyTuple))
print (max(pyTuple))
print (min(pyTuple))
print (sum(pyTuple))
print (sorted(pyTuple))
a=enumerate(pyTuple)
print (tuple(a))
for item in enumerate(pyTuple):
  print(item)
mytuple=reversed(pyTuple)
print (tuple(mytuple))
```

Zip() function

The zip() is a built-in function that takes two or more sequences and "zips" them into a list of tuples. The tuple thus, formed has one element from each sequence.

Example



```
Tup = (1,2,3,4,5)
List1 = ['a','b','c','d','e']
print(list((zip(Tup, List1))))
```

OUTPUT

```
[(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd'), (5, 'e')]
```

Exercise:

1.Write a Python program to count the elements in a tuple and add the elements to a new tuple until an element is a tuple.Print the sum of nested tuple in the tuple?

2. Write a Python program to replace last value of tuples in a list with A,B,C grades?

 Example:mytuple=((1,2,70),(3,4,80),(5,6,90)

Output: ((1,2,'C'),(3,4,'B'),(5,6,'A'))

3. Write a Python program to check whether an element exists within a tuple or not without using any methods and functions?