Tuples

Tuple:

* A tuple is an inbuilt data type which is an ordered, immutable and lightweight collection of elements in python.
* The term immutable means once the tuple is created you can’t modify the size or the elements which are present in the tuple.
* The term ordered means the position of each element is fixed in the way that they are inserted.
* Tuples are considered as light weight because they consume less memory when compared with the sequence data types like lists.
* Tuples are enclosed in parentheses, and their elements can be accessed using index positions, starting from 0.

**Creation of tuples**

You can create a tuple by placing all sequence elements in between parentheses.

Syntax: Tuple\_name= (sequence of elements )

Examples:

Tuple\_1= ( ) # empty Tuple

Tuple\_2= (20) # Tuple with single element

Tuple\_3= (10,20,30,40,50) # Tuple with sequence of elements

Tuple\_4= (9.0,20,”python”,”True”) # Tuple with multiple data types

Tuple\_5= (10,20,30,40,(2.3,9.0,5,7),18) # nested tuple

**Accessing the tuples**

The elements in tuples can be accessed through indexing.

Tuples support both positive( starts with zero) and negative indexing (start with -1) which is from the back of the tuple i.e. 90 in the below example.

Examples:

Tuple\_1=(10,30,40,50,60,80,90)

Tuple\_1[0] #10

Tuple\_1[-2] #80

Tuple\_1[3] #50

Tuple\_1[-1] #90

**Slicing the tuples**

It is a way of extracting a part of the tuple with specified index ranges.

You can perform slicing with both the positive and negative index.

**Syntax: Tuple\_name[start :end :difference]**

**start** : index from where you want to start slicing, it is inclusive

**end** : index before where slicing ends, it is exclusive

**difference** : it is the interval between the elements.By default the value will be 1.

Examples:

names=("anil","akash”,"arjun","akhil","bhupal","dheeraj","mahesh")

If you want all the names starts with “a” i.e from index 0 to index 3, then use names[0:4]

If you want names alternatively from begining to till the end, then use names[0:6:2]

**Deletion of tuple**

You can’t delete the elements in tuple but you can delete the entire tuple using del keyword.

Example:

tuple\_1=(10,20,30,40,90,89,109)

del tuple\_1

print(tuple\_1) # when you try to print you will get a nameerror because that name does not exist.

============================

Build in methods for tuples

1. len ( ):

This method is used to get the length of the tuple i.e total number of elements in the tuple.

If the tuple has one more tuple inside it, then the inside tuple is considered as a single element irrespective of the number of elements in it.

Examples:

1. numbers=(1,2,3,4,5,6,7,8,9,10)

print (len (numbers) ) # prints 10 in the output.

1. numbers=(1,2,3,4,5,6,7,8,9,10,(2,3,7,9))

print (len (numbers) ) # prints 11 in the output.

1. count ( ):

Count method is used to find the number of occurrences of a specified element in the given tuple.

The element can be of numbers or strings.

Examples:

1. numbers=(4,5,6,7,8,9,8,9,5)

print(numbers.count(9)) # prints 2 because 9 appeared two times.

1. tuple\_1=(4,5,6,7,8,"i","i","am",9,"am",90,"am",9,8)

print(tuple\_1.count("am")) # prints 3 because “am” is repeated 3 times.

3. index ( ):

This method is used to get the index of the first occurrence of the specified element in the tuple.

It will raise a value error when the specified value is not present in the given tuple.

Examples:

1. numbers=(4,5,6,7,8,9,8,9,5,4)

print(numbers.index(4)) #prints 0 because it is the first occurrence of 4.

names=("anil","ganesh","ram","lokesh","uday","ganesh")

print(names.index("ganesh")) #prints 1 as output.

4. Tuple concatenation

It is the concatenation of 2 or more tuples into one single tuple.

You have to use the plus operator ( + ) to perform concatenation.

The order of the elements will stay in the way you add them.

Examples:

1. numbers\_1= (1,2,3,4,5)

numbers\_2=(99,105,110,114,117)

numbers\_3= numbers\_1+numbers\_2

print(numbers\_3) # prints (1, 2, 3, 4, 5, 99, 105, 110, 114, 117)

1. numbers\_1= (1,2,3,4,5)

numbers\_2=(99,105,110,114,117)

numbers\_3= numbers\_2+numbers\_1

print(numbers\_3) # prints (99, 105, 110, 114, 117, 1, 2, 3, 4, 5)

5. Tuple repetition

It is the way of cloning the data of a given tuple a number of times.

You have to use the repetition operator ( \* ) to perform repetition.

Examples:

1. numbers\_1=(1,2,3,4)

print(numbers\_1 \* 2) #prints (1, 2, 3, 4, 1, 2, 3, 4)

1. names=("anil","mahesh","arjun","vijay","sri")

print(names\*2) #prints ('anil', 'mahesh', 'arjun', 'vijay', 'sri', 'anil', 'mahesh', 'arjun', 'vijay', 'sri')

6. sorted

It is a built-in function that is used to sort the elements of the same datatype.

If the elements in the tuple are not of the same data type while using a sorted function then it will raise an error called TypeError.

This function works for both the numbers and strings as well.

Numbers are sorted in ascending order and Strings will be sorted in the alphabetical order.

As the tuples are immutable they give a list as the output for this function.

Examples:

1. numbers=(2,6,3,4,8,7,9,1)

print(sorted(numbers)) #prints [1, 2, 3, 4, 6, 7, 8, 9]

1. names = ('bipin','shree','pooja','akhil','arjun')

print(sorted(names)) #prints ['akhil', 'arjun', 'bipin', 'pooja', 'shree']

7. Math functions

As the tuple elements can be iterable you can use the functions like min(),max(),sum() on tuples.

Example:

nums=(90,40,50,20,45)

print(max(nums)) #prints 90

print(min(nums)) #prints 20

print(sum(nums)) #prints 245