UNIT-II

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

Mutable V/s Immutable Data Types

Mutable Data Type	Immutable Data Type
Sequences can be modified after creation	Sequences cannot be modified after creation
Ex: Lists, Dictionary, Sets	Ex: Strings, Tuples
Operations like add, delete and update can be performed	Operations like add, delete and update cannot be performed

Tuples

- An ordered group of sequence seperated by symbol, and enclosed inside the parenthesis
- A tuple is a sequence of immutable objects, therefore tuple cannot be changed
- In Python, tuple is similar to a list. Only the difference is that list is enclosed between square bracket, tuple between parenthesis and List have mutable objects whereas Tuple have immutable objects.
- NOTE: If Parenthesis is not given with a sequence, it is by default treated as Tuple.

Advantages of Tuple over List

- Since tuple are immutable, iterating through tuple is faster than with list. So there is a slight performance boost.
- It makes the data safe as Tuples are immutable and hence cannot be changed.
- Tuples that contain immutable elements can be used as key for a dictionary. With list, this is not possible.
- Tuples are used for String formatting.

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

- **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.).
- Syntax: tuple1= ( ) # creation of empty tuple

  tuple2 = (Sequence 1,) # , symbol is mandatory with out which it becomes just a string assignment operator

  tuple3= (Sequence1,Sequence2)
- **Example**

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

- Tuples can also be nested.

Ex-

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

Output:

```
>>>  ('a', 'mahesh', 10.56)
     (('a', 'mahesh', 10.56), (10, 20, 30))
```


K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT-Basara, T.S, IND.

- **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])

Tuple Operations : Slicing

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

A subpart of a tuple can be retrieved on the basis of index. This subpart is known as tuple slice.

: If the index provided in the Tuple slice is outside the list, then it raises an IndexError exception

- **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[:])
- # Output: ('m') print(my_tuple[-1])
- # Output: ('p','r','o','g','r','a','m') print(my_tuple)

- **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)

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT-Basara, T.S, IND.

- **Deleting a Tuple**
- we cannot delete or remove items or elements 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 # will delete the tuple data
- print(my_tuple) # will show an error sice tuple data is already deleted

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

- **Tuple Methods**
- Methods that add items or remove items are not available with tuple. Only the following two methods are available.
- count(x) Return the number of items that is equal to x
- index(x) Return index of first item that is equal to x
- **Example**
- my_tuple = ('a','p','p','l','e',)
- print(my_tuple.count('p'))#Output:2
- print(my_tuple.index('l'))#Output:3

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

- **Tuple functions**
- all (T) - Return True if all elements of the tuple are true (or if the tuple is empty).
- any(T) - Return True if any element of the tuple is true. If the tuple is empty, return False.
- enumerate(T) - Return an enumerate object. It contains the index and value of all the items of tuple as pairs.
- len(T) - Return the length (the number of items) in the tuple.
- max(T) -Return the largest item in the tuple.
- min(T) - Return the smallest item in the tuple
- sorted(T) - Take elements in the tuple and return a new sorted list (does not sort the tuple itself).
- sum(T) - Retrun the sum of all elements in the tuple.
- tuple(T) - Convert an iterable (list, string, set, dictionary) to a tuple.
- cmp(t1,t2) – To compare the two given tuples
- tuple(sequence) – Converts the sequence into tuple

- **cmp(tuple1,tuple2)**
- **Explanation:**If elements are of the same type, perform the comparison and return the result. If elements are different types, check whether they are numbers.
- If numbers, perform comparison.
- If either element is a number, then the other element is returned.
- Otherwise, types are sorted alphabetically .
- If we reached the end of one of the lists, the longer list is "larger." If both list are same it returns 0.

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

- **Eg:**
- data1=(10,20,'rahul',40.6,'z')
- data2=(20,30,'sachin',50.2)
- print cmp(data1,data2)
- print cmp(data2,data1)
- data3=(20,30,'sachin',50.2)
- print cmp(data2,data3)
- **Output:**

  -1

  1

  0

K.RaviKanth, Ast.Prof., Dept. of CSE, IIIT- RGUKT- Basara, T.S, IND.

- **Example**

pyTuple =(20,55,43,22,67,90,0) # data=(10,20,'Ravi',40.6,'z')

- print all(pyTuple)

Try out

- 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)

IIIT BASARA

Infosys | Campus Connect

For Details Contact Me @ :

9247448766

ravikanth27787@gmail.com

python

Programming Language