# Today I'll Cover :

1. Tuple Data Structure
2. Creation of Tuple
3. Accessing of Tuple
4. Traversing in Tuple
5. Methods of Tuple
6. Operator that can be applied on Tuple .
7. Nested Tuple & Tuple Comprehension

Mohammad Altaf Hussain


In the name of ALLAH,
the Most Beneficent, the Most Merciful

Mohammad Altaf Hussain

# Tuple ()

If we want to represent a group of individual objects as a single entity where :-

- ✓ insertion order preserved.
- ✓ duplicate objects are allowed
- ✓ heterogeneous objects are allowed
- ✓ Modification are not allowed, once object is created. Then we should go for Tupple. The elements are placed within parenthesis and with comma separator. Where parenthesis is Optional.

Mohammad Altaf Hussain

# Creation of Tuple

- Empty Tuple:-
  1. ()
  2. tuple()

- Tuple with element:-

  ( element1, element2, elementN )

  element1, elementN

- tuple() function used to cast other data type into tuple type.

# Accessing of Tuple

- By using indexing we can access single element :-
  - +(ve) index means left to right (forward).
  - -(ve) index means right to left (backward).
  - Indexing start from 0 in forward direction -1 in backward direction.
  - If we are trying to access characters of a string with out of range index then we will get IndexError.

- **By using string slicing, we can access one or more elements.**

**Syntax** : **any_tuple [bIndex:eIndex:step]**

Where,

**bIndex**: slice has to start.

**eIndex**: slice has to end.

**step**: increment value.

**Note :-**

- Default value of begin index is 0.
- Default value of ending index is max allowed index of tuple ie. length of the tuple.
- Default value for step is 1.

**Note :-**

- if step value is +ve then it should be traverse in forward direction(left to right) and we have to consider begin to end -1.
- if step value is -ve then it should be traverse in backward direction(right to left) and we have to consider begin to end +1

# Traversing in Tuple

We can traverse in list using:-

- For loop
- While loop

# Methods of Tuple

**index()** :- It return the index of the specified element. If element is not available then we will get ValueError. So first We should check the availability of element in the tuple.

**Syntax: any_tuple.index(element, begin, end)**

Where, **begin** and **end** indicates beginning index position and ending index position respectively. It is optional.

## ❖ Counting element in given Tuple

- **count() :-** It is used to find the number occurrence of specified element.

**Syntax: any_tuple.count(element)**

# ❖ Tuple packing and unpacking

➢ **We can create a tuple by packing a group of variables.**

**Syntax:  t = a, b, c, d**

Here a, b, c, d are packed into a tuple t. This is nothing but tuple packing

➢ **Tuple unpacking is the reverse process of tuple packing. We can unpack a tuple and assign its values to different variables.**

**Syntax: a, b, c, d = t**

**Note:-** In tuple unpacking the number of variables and number of values should be same. Otherwise we will get ValueError.

# Operator that can be applied on tuple

| | | | | | |
|---|---|---|---|---|---|
| + | | | Concatenation | | |
| * | | | Repetition | | |
| < | <= | == | != | >= | > |
| in | not in | | Membership | | |

1. To use + operator for tuple, compulsory both arguments should be tuple type
2. To use * operator for tuple, compulsory one argument should be tuple and other argument should be int
3. We can compare tuple, comparison is based on how the letters are arranged in dictionary that we use in real life.

Mohammad Altaf Hussain

# Nested Tuple

- Tuple within tuple is known as nested Tuple.
- We can access nested tuple elements by using index just like accessing multi dimensional array elements.

# Tuple Comprehension

Python does not support tuple comprehension.