Python Tuple

- A tuple is a sequence of values , Values can be any **type**, are **indexed** by integers
- **immutable** objects; cannot be changed
- Used to create **write-protected data.**

  - are not dynamic, faster than lists

- Tuple is enclosed between parenthesis
- **Tuple Vs List**

  - Tuple is **like list**
  - Difference is, List have **mutable** objects whereas Tuple have **immutable** objects.
  - List is enclosed between square bracket, tuple between parenthesis

Creating Tuple

- Tuples are defined in parentheses () , values are separated by commas
- Contain values of different data types.
- Can be an empty
- A single valued tuple,
    - must be a comma at the end of the value
- Can also be nested.
- If a Tuple does not enclose with parenthesis, still it will be considered as tuple

Creating Tuple example

```
t1=(10,20.50,"python",'p',True)

t2=  10,20.50,"python",'p',True

t3 =  (10,)

t4 =  ()

t5 =  (10,20.50,"python")

t6=   (t5,'p',True)

t7 = tuple()
```

Accessing Tuple values

- Use square bracket **[],** to slice along the **index** or **indices** and access the values of a tuple

- Tuple elements can be accessed like String and List

  - Forward Indexing , indexing start with 0 to n-1 (Reading from Left to Right)

  - Backward Indexing, indexing start with -1 to –n (Reading from Right to Left)

Accessing Tuple elements example

t1=(10,20.50,"python",'p',True)

t1[0]

t1[4]

t1[-1]

t1[-5]

t1[5]

t1[-6]

IndexError: tuple index out of range

Tuple Slicing

- Slicing is used to select range of values from tuple object

  syn: [start_index : end_index : step].

  - start_index is the beginning index of the slice; default value is 0.

  - end_index is the end index of the slice; default value is the len(sequence).

  - step is the amount by which the index increases, the default value is 1.

Tuple slicing example

```
t1 = ("python","tuples","are","immutable","write","protected")

print(t1[1:4])

print(t1[:4])

print(t1[:])

print(t1[::2])

print(t1[::-1])
```

Tuple basic operations

- **Membership operators**

  - in returns True if an item is present in sequence else False

  - not in returns True if an item is not present in sequence else False

- **Addition Tuple**

  - Tuple can be added by using the concatenation operator(+) to join two tuples.

- **Replicating Tuple:**

  - Replicating can be performed by using '*' operator by a specific number of time.

Tuple basic operations example

```
t1 = (10,20.50,"python",'p',True)

'p' in t1

20.50 not in t1
```

```
t1 = ("python","tuples","are")

t2 = ("and","immutable","write","protected")

t3 = t1 + t2

print(t3)

t4 = ("Immutable " * 3)
print(t4)
```

Tuple basic operations

- **Updating elements in a List:**
  - Elements of the Tuple cannot be **updated**. since Tuples are **immutable**.

- **Deleting elements from Tuple:**
  - Deleting individual element from a tuple is not supported.
  - Whole of the tuple can be deleted using the **del** statement

```
t1=(10,20,'rahul',40.6,'z')
print (t1)
del (t1)
```

Index and Count

- index() : searches an element in a tuple and returns its index.
    - returns its position
    - if the same element is present more than once, the first position is returneda
    - If no element is found, a ValueError exception is raised indicating the element is not found.

```
t2=(10,'savik',40.6,'z')

print(t2.index(40.6))

# print(t2.index('a'))
raises     ValueError: tuple.index(x): x not in tuple
```

```
t2=(10,'savik',40.6,10,'z')

print(t2.count(10))
```

Tuple operations

- **min(), max() and len()**

  built-in function to get the maximum value, minimum values and the length of a sequence.

```
t1=(100, 255.55, True)

print(len(t1))

print(min(t1))

print(max(t1))

print()
```

unpacking

- **unpack** : tuple into variables.
  - when unpacking a tuple the number of variables on the

    left side should be equal to the number of the values in the tuple

    Otherwise, error such as **ValueError**: too many values to unpack

```
a, b, c = (10, 20, 30)

print(a)

print(b)

print(c)
```