16. Python – Tuple Data Structure

Table of Contents

16. Python – Tuple Data Structure

1. Tuple data Structure

- ✓ We can **create** tuple data structure by using,
  - Parenthesis () symbol.
  - Predefined tuple(p) function.

- ✓ A tuple can **store** group of objects or elements.
  - A tuple can store **same** (Homogeneous) type of elements.
  - A tuple can store **different** (Heterogeneous) type of elements.

- ✓ In tuple insertion **order** is preserved or **fixed**.
  - If we insert elements into 10, 20, 30 then output also will display as 10, 20, 30 then this is called as insertion order is preserved or fixed
  - Example
    - Input => (10, 20, 30)
    - Output => (10, 20, 30)

- ✓ **Duplicate** elements are **allowed**.
- ✓ Tuple having **immutable** nature.
  - Immutable means once we create a tuple object then we cannot change or modify the content of tuple object.

- ✓ Store elements by using **index**.
  - A tuple data structure supports both positive and negative indexes.
  - Positive index means from left to right
  - Negative index means right to left

Note:

- ✓ tuple is a predefined class in python.
- ✓ Once if we create tuple object means internally object is creating for tuple class.

Note:

- ✓ Inside tuple every object can be separated by comma separator.

2. When should we go for tuple data structure?

- ✓ If we are going to define a data which never change over all the period, then we should go for tuple data structure.

Example:

1. Week days names
2. Month names
3. Year names

Program Name	Tuple having same type of objects demo1.py
	employee_ids = (10, 20, 30, 40, 50) print(employee_ids) print(type(employee_ids))
Output	(10, 20, 30, 40, 50) <class 'tuple'>

3. Syntax Surprise 1: Single value tuple

- ✓ If tuple having only one object, then that object should end with comma separator otherwise python internally not considered as it is tuple.

Program Name: A single value with tuple syntax, but it's not tuple
demo2.py

```
number = (9)

print(number)
print(type(number))
```

Output

```
(9)
<class 'int'>
```

Program Name: A single value with tuple syntax, but it's not tuple
demo3.py

```
name = ("Daniel")

print(name)
print(type(name))
```

Output

```
Daniel
<class 'str'>
```

Program Name	Tuple single value ends with comma separator then it's tuple demo4.py
	name = ("Daniel",) print(name) print(type(name))
Output	('Daniel') <class 'tuple'>

4. Syntax Surprise 2. Parenthesis is optional for tuple

- ✓ While creating a tuple parenthesis is optional

Program Name	Parenthesis symbol is optional while creating tuple demo5.py
	emp_ids = 10, 20, 30, 40 print(emp_ids)
output	(10, 20, 30, 40)

5. Different ways to create a tuple

1. Empty tuple

- ✓ We can create an empty tuple by using empty parenthesis.

Program Name: empty tuple
demo6.py

```
emp_id = ()
print(emp_id)
print(type(emp_id))
```

output

```
()
<class 'tuple'>
```

2. Tuple with group of values

- ✓ Tuple can contain group of objects; those objects can be same type or different type.

Program Name: Tuple example
demo7.py

```
emp_id = (11, 12, 13)
std_id = 120, 130, 140
print(emp_id)
print(std_id)
```

output

```
(11, 12, 13)
(120, 130, 140)
```

Program Name: Tuple example
demo8.py

```
t = (11, 12, 13, "daniel")
print(t)
```

output

```
(11, 12, 13, "daniel")
```

3. By using tuple(p) function

- ✓ We can create tuple by using tuple(p) function.

Program Name: Creating tuple by using tuple function
demo9.py

```
a = [11, 22, 33]
t = tuple(a)
print(t)
```

output

```
(11, 22, 33)
```

6. Accessing elements of tuple:

- ✓ We can access tuple elements by using,
  - Index
  - Slice operator

6.1 Index

- ✓ Index means position where element stores

Program Name: Accessing tuple by using index
demo10.py

```
t = (10, 20, 30, 40, 50, 60)

print(t[0])        #    10
print(t[-1])       #    60
```

Output

```
10
60
```

6.2. Slice operator:

- ✓ A group of objects from starting point to ending point

Program Name: Accessing tuple by using slice
demo11.py

```
t = (10, 20, 30, 40, 50, 60)

print(t[2:5])
print(t[2:100])
print(t[::2])
```

Output

```
30, 40, 50)
(30, 40, 50, 60)
(10, 30, 50)
```

7. Tuple vs immutability:

- ✓ Tuple having immutable nature.
- ✓ If we create a tuple then we cannot modify the elements of existing tuple.

Program Name	Prove tuple having immutable nature demo12.py
	t = (10, 20, 30, 40) print(t[1]) t[1] = 70
Output	
	20 TypeError: 'tuple' object does not support item assignment

8. Mathematical operators on tuple:

- ✓ We can apply plus (+) and Multiplication (*) operators on tuple.
- ✓ + Operator works as concatenation.
- ✓ * Operator works as multiplication.

8.1. Concatenation operator (+):

- ✓ + operator concatenates two tuples and returns single tuple

Program Name	Concatenation operator on tuple demo13.py

```
t1 = (10,20,30)
t2 = (40,50,60)
t3 = t1 + t2

print(t3)
```

Output

```
(10, 20, 30, 40, 50, 60)
```

8.2 Multiplication operator (*)

- ✓ Multiplication operator works as repetition operator

Program Name: Repetition operator on tuple
demo14.py

```
t1 = (10,20,30)
t2 = t1*3
print(t2)
```

Output

```
(10, 20, 30, 10, 20, 30, 10, 20, 30)
```

9. len(p) function

- ✓ To return number of elements present in the tuple

Program Name: len(p) function
demo15.py

```
t = (10,20,30,40)
print(len(t))
```

Output

```
4
```

10. Method in tuple data structure

- ✓ As discussed, tuple is a predefined class.
- ✓ So, tuple class can contain methods because methods can be created inside of class only.
- ✓ We can check these methods by using dir(p) predefined function.

- ✓ So, internally tuple class contains two types of methods,

    - With underscore symbol methods.
        - We no need to focus

    - Without underscore symbol methods.
        - We need to focus much on these

Program Name	Printing tuple data structure methods by using dir(p) function demo16.py print(dir(tuple))
output	['__add__', ... , '__subclasshook__', 'count', 'index']

Important point

- ✓ As per object-oriented principle,
  - If we want to access instance method then we should access by using object name.
- ✓ So, all tuple methods we can access by using tuple object.

Methods in tuple

1. count(parameter1) method
2. index(parameter1) method

10.1. count(p) method

- ✓ count(p) is a method, we should access this method by using tuple object.
- ✓ This method returns the number of occurrences of specified item in the tuple

Program Name	count(p) method demo17.py
	t = (10, 20, 10, 10, 20) print(t.count(10))
output	
	3

10.2. index(p) method

- ✓ returns index of first occurrence of the given element.
- ✓ If the specified element is not available, then we will get ValueError.

Program Name	index(p) method demo18.py
	t = (10, 20, 30) print(t.index(30))
Output	
	2

Program Name	index(p) method demo19.py
	t = (10, 20, 30) print(t.index(88))
Output	
	ValueError: tuple.index(x): x not in tuple

12. Differences between List and Tuple:

- ✓ List and Tuple are exactly same except small difference:
  - List objects are mutable
  - Tuple objects are immutable.
- ✓ In both cases,
  - Insertion order is preserved.
  - Duplicate objects are allowed
  - Heterogeneous objects are allowed
  - Index and slicing are supported.

List	Tuple
✓ List is a Group of Comma separated Values within Square Brackets and Square Brackets are mandatory. ✓ Example: i = [10, 20, 30, 40]	✓ Tuple is a Group of Comma separated Values within Parenthesis and Parenthesis are optional. ✓ Example: t = (10, 20, 30, 40) ✓ Example: t = 10, 20, 30, 40
✓ List Objects are Mutable i.e. once we create List object we can perform any changes in that Object. ✓ Example: i[1] = 70	✓ Tuple Objects are Immutable i.e. once we create Tuple object we cannot change its content. ✓ Example: t [1] = 70 ✓ TypeError: tuple object does not support item assignment.
✓ If the Content is not fixed and keep on changing, then we should go for List.	✓ If the content is fixed and never changes then we should go for Tuple.

13. Can I add elements to this tuple t = (11, 22, [33, 44], 55, 66)?

- ✓ Yes we can add elements to list in tuple.
- ✓ In second index position list is available, to that we can add

Program Name: tuple data structure can store any data
demo23.py

```
t = (11, 22, [33, 44], 55, 66)

t[2].append(77)
print(t)
```

output

```
(11, 22, [33, 44, 77], 55, 66)
```