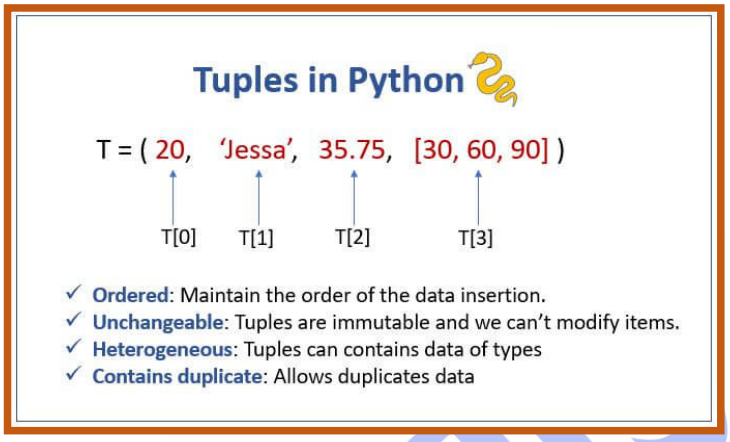
Tuples in Python

Data Structures - Tuples in Python

A tuple is a data type in Python that is used to store multiple items in a single variable. It is similar to a list, but tuples are immutable, meaning you cannot change, add, or remove items after the tuple is created.



Why do we use Tuple?

- Tuples cannot be changed (they are fixed).
- Tuples are faster than lists.
- They use less memory.
- You can use them as keys in dictionaries.
- Tuples help keep data safe from changes.
- Good for storing fixed data (like days of the week).
- You can return multiple values from a function using a tuple.

Tuples in Python

Types of Tuples in Python

Type of Tuple	Description	Example
Empty Tuple	Tuple with no elements	t = ()
Single Element Tuple	Tuple with one item (comma is needed)	t = (5,)
Homogeneous Tuple	All elements are of the same type	t = (1, 2, 3, 4)
Heterogeneous Tuple	Elements of different data types	t = (1, "hello", 3.14)
Nested Tuple	Tuple containing other tuples	t = (1, (2, 3), (4, 5))
Tuple with Collections	Tuple containing lists or sets, etc.	t = ([1, 2], {3, 4})

How to Create a Tuple

1. With Parentheses:
numbers = (10, 20, 30)

2. Without Parentheses (comma is enough):
colors = "red", "green", "blue"

3. Single Element Tuple:
Be careful! A single element tuple must have a comma, otherwise Python treats it as a normal value.

```
single = ("hello",)
print(type(single))  # <class 'tuple'>

not_tuple = ("hello")
print(type(not_tuple))  # <class 'str'>
```

Accessing Tuple Elements

Just like lists, you can access tuple items using **indexing**.

```
animals = ("cat", "dog", "rabbit", "horse")
print(animals[0])    # first element
print(animals[-1])   # last element
```

Output:

```
cat
horse
```

Tuples in Python

Slicing Tuples

You can get a range of elements using slicing.

```
numbers = (1, 2, 3, 4, 5, 6)
print(numbers[1:4])   # from index 1 to 3
print(numbers[:3])    # first three elements
print(numbers[-3:])   # last three elements
```

Output:

```
(2, 3, 4)
(1, 2, 3)
(4, 5, 6)
```

Tuple Immutability (You cannot change it)

You cannot modify a tuple once it's created.

```
fruits = ("apple", "banana", "cherry")
fruits[1] = "mango"   # ❌ Error
```

Output:

```
TypeError: 'tuple' object does not support item assignment
```

Looping through a Tuple

You can use a for loop to print all elements.

```
colors = ("red", "green", "blue")
for c in colors:
    print(c)
```

Output:

```
red
green
blue
```

Tuples in Python

Tuple Functions and Methods:

Function / Method	Description	Example
len()	Returns the number of items	len(fruits)
count(value)	Returns how many times a value appears	fruits.count("apple")
index(value)	Returns the index of the first occurrence	fruits.index("banana")
max()	Returns largest value (if numbers)	max(nums)
min()	Returns smallest value (if numbers)	min(nums)
sum()	Adds all numeric values	sum(nums)

Solved Example:

```
nums = (5, 10, 15, 20, 15)
print(len(nums))
print(nums.count(15))
print(nums.index(10))
print(max(nums))
print(sum(nums))
```

Output:

```
5
2
1
20
65
```

Tuple with Mixed Data Types

Tuples can hold different data types.

```
student = ("Mehul", 25, 9.3, True)
print(student)
```

Output:

('Abhyas', 25, 9.3, True)

Nested Tuples

Tuples can contain other tuples.

```
nested = (("apple", "banana"), ("red", "yellow"))

print(nested[0])      # ('apple', 'banana')
print(nested[0][1])   # banana
```

Concatenating Two Tuples

In Python, **tuples can be joined (concatenated)** using the + operator.

Tuples in Python

Example:

```
# Two tuples
fruits = ("apple", "banana", "mango")
vegetables = ("carrot", "potato", "tomato")

# Concatenate tuples
food_items = fruits + vegetables

print(food_items)
```

Output:

```
('apple', 'banana', 'mango', 'carrot', 'potato', 'tomato')
```

Explanation:

- + joins the two tuples together.
- The result is a **new tuple** because tuples are **immutable** (cannot be changed once created).

Converting a Tuple into a List

Sometimes you may need to **modify** a tuple — but since tuples are **immutable**, you can't directly change them.
So, we **convert** the tuple into a **list**, make changes, and convert it back to a tuple.

Example:

```
# Original tuple
fruits = ("apple", "banana", "mango")

# Convert tuple to list
fruits_list = list(fruits)

# Add a new element
fruits_list.append("orange")

# Convert list back to tuple
updated_fruits = tuple(fruits_list)

print(updated_fruits)
```

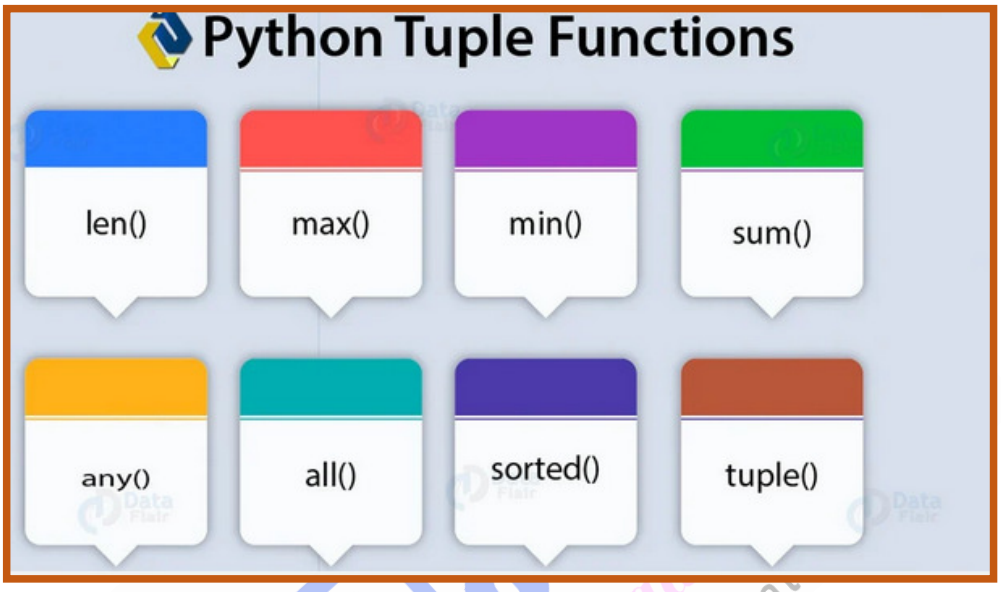
Output:

```
('apple', 'banana', 'mango', 'orange')
```

Tuples in Python

Explanation:
1. list(fruits) → converts tuple to list.
2. .append("orange") → adds a new element.
3. tuple(fruits_list) → converts it back to a tuple.

Basic Functions of Tuple



Difference between List and Tuple

Aspect	List	Tuple
Syntax	[] (square brackets)	() (parentheses)
Changeable	Yes (mutable)	No (immutable)
Speed	Slower	Faster
Methods	More methods	Fewer methods
Use	For changing data	For fixed data
Memory	Uses more memory	Uses less memory

Tuples in Python

Assignment

Ques 1: **Create a tuple of five fruits** and print the second and last fruit from it. *(Hint: Use indexing)*
Ques 2: **Write a Python program** to find the **length** of a tuple.
Ques 3: **Create a tuple of numbers** and find the **maximum and minimum value** using built-in functions.
Ques 4: **Concatenate two tuples** - one with fruits and one with vegetables - and print the combined tuple.