2.Tuples

A **tuple** in Python is a built-in data structure that allows you to store a collection of items. Tuples are similar to lists but have some key differences that make them unique. Here's a detailed overview of the tuple data structure in Python:

Characteristics of Tuples

1. **Immutable**:
   - Once a tuple is created, its contents cannot be changed (i.e., you cannot add, remove, or modify elements).
2. **Ordered**:
   - Tuples maintain the order of elements, meaning that the position of each element is fixed.
3. **Allow Duplicates**:
   - Tuples can contain duplicate values, just like lists.
4. **Heterogeneous**:
   - A tuple can hold items of different data types, including integers, strings, lists, and even other tuples.

Key Features of Tuples

1. **Immutable**: Once created, the contents of a tuple cannot be changed (no addition, deletion, or modification of elements).
2. **Ordered**: Tuples maintain the order of elements, which means the position of each element is fixed.
3. **Allow Duplicates**: Tuples can contain multiple occurrences of the same value.
4. **Heterogeneous**: A tuple can store items of different data types, such as integers, strings, and lists.
5. **Indexable**: Elements can be accessed using zero-based indexing.
6. **Lightweight**: Tuples are generally more memory-efficient than lists.

Syntax of Tuples

Tuples are created by placing a comma-separated sequence of items inside parentheses `()`.

```
# Basic syntax
my_tuple = (item1, item2, item3)
```

Examples

1. Creating a Tuple

```
# Creating a tuple with integers
tuple_of_numbers = (1, 2, 3, 4, 5)
print(tuple_of_numbers)  # Output: (1, 2, 3, 4, 5)
```

```python
# Creating a tuple with mixed data types
mixed_tuple = (1, "Hello", 3.14, True)
print(mixed_tuple)  # Output: (1, 'Hello', 3.14, True)

# Creating a tuple with a single element
single_element_tuple = (5,)
print(single_element_tuple)  # Output: (5,)
```

2. Accessing Elements

```python
# Accessing elements using indexing
my_tuple = (10, 20, 30, 40)
print(my_tuple[0])  # Output: 10
print(my_tuple[2])  # Output: 30
```

3. Tuple Operations

- **Concatenation**:

```python
tuple1 = (1, 2)
tuple2 = (3, 4)
combined = tuple1 + tuple2
print(combined)  # Output: (1, 2, 3, 4)
```

- **Repetition**:

```python
repeated = tuple1 * 3
print(repeated)  # Output: (1, 2, 1, 2, 1, 2)
```

- **Membership Test**:

```python
print(2 in tuple1)  # Output: True
print(5 in tuple1)  # Output: False
```

4. Tuple Methods

- **Count**:

```python
my_tuple = (1, 2, 2, 3)
print(my_tuple.count(2))  # Output: 2
```

- **Index**:

```python
print(my_tuple.index(2))  # Output: 1
```

5. Packing and Unpacking

- **Packing**:

```python
packed_tuple = (1, 2, 3)  # Packing values into a tuple
```

- **Unpacking**:

```python
a, b, c = packed_tuple
```

```python
print(a, b, c)  # Output: 1 2 3
```