Tuple

In Python, a tuple is an ordered, immutable collection of elements. Tuples are similar to lists, but once a tuple is created, you cannot modify its contents - you cannot add, remove, or change elements. **Advantages of Tuples:**

- Immutability: Once created, the elements of a tuple cannot be changed, making them suitable for situations where data should remain constant.
- Performance: Tuples are generally faster than lists for certain operations, especially when used as keys in dictionaries.
- Validity: Tuples can be used as dictionary keys, whereas lists cannot, due to their immutability.

Tuples are often used when the order and structure of the data should remain constant throughout the program.

Remember that tuples are immutable, so you cannot modify their elements after creation. If you need to modify elements, consider using a list instead.

Lets discover how to iterate over tuple.

In [1]:
```python
# Iteration using for loop
my_tuple = (1, 2, 3, 4, 5)

for element in my_tuple:
    print(element)
```

```
1
2
3
4
5
```

In [2]:
```python
# Iteration using enumerate()
my_tuple = (1, 2, 3, 4, 5)

for index, element in enumerate(my_tuple):
    print(f"Index: {index}, Element: {element}")
```

```
Index: 0, Element: 1
Index: 1, Element: 2
Index: 2, Element: 3
Index: 3, Element: 4
Index: 4, Element: 5
```

In [3]:
```python
# Interation with indices using range()
my_tuple = (1, 2, 3, 4, 5)
```

```
for index in range(len(my_tuple)):
    print(f"Index: {index}, Element: {my_tuple[index]}")
```

```
Index: 0, Element: 1
Index: 1, Element: 2
Index: 2, Element: 3
Index: 3, Element: 4
Index: 4, Element: 5
```

Tuple operations

1. Tuple creation
2. Accessing tuple elements
3. Concatenation of 2 tuples
4. Repetition of tuple elements
5. Element Membership testing
6. To find Length of tuple
7. To unpack tuple elements
8. To count occurnces of a value in a tuple
9. To find the index of first occurence of a value
10. To convert tuple into list & vice versa
11. Usage of built-in function

1. Tuple creation

In [9]:
```
my_tuple = (1, 2, 3, 'a', 'b')
my_tuple
```

Out[9]:
```
(1, 2, 3, 'a', 'b')
```

2. Accessing tuple elements

In [2]:
```
my_tuple = (1, 2, 3, 'a', 'b')
print(my_tuple[0])  # Output: 1
print(my_tuple[2:4])  # Output: (3, 'a')
```

```
1
(3, 'a')
```

3. Concatenation of 2 tuples

In [4]:
```
# Concatenating tuples using the + operator.
tuple1 = (1, 2, 3)
tuple2 = ('a', 'b', 'c')
```

```
result_tuple = tuple1 + tuple2
result_tuple
```

Out[4]: (1, 2, 3, 'a', 'b', 'c')

4. Repetition of tuple elements

In [5]:
```
# Repeating a tuple using the * operator
my_tuple = (1, 2)
repeated_tuple = my_tuple * 3
repeated_tuple
```

Out[5]: (1, 2, 1, 2, 1, 2)

5. Element Membership testing

In [7]:
```
# Checking if an element is present in a tuple using in.
my_tuple = (1, 2, 3)
print(2 in my_tuple)  # Output: True
```

```
True
```

6. To find Length of tuple

In [10]:
```
# Finding the length of a tuple using len().
my_tuple = (1, 2, 3, 'a', 'b')
length = len(my_tuple)
length
```

Out[10]: 5

7. To unpack tuple elements

In [15]:
```
# Assigning values of a tuple to multiple variables.
my_tuple = (1, 'apple', 3.14)
a, b, c = my_tuple
```

8. To count occurnces of a value in a tuple

In [16]:
```
# Counting occurrences of a value in a tuple.
my_tuple = (1, 2, 2, 3, 2, 4)
count_of_2 = my_tuple.count(2)
```

9. To find the index of first occurence of a value

In [18]:
```
# Finding the index of the first occurrence of a value.
my_tuple = (1, 2, 3, 4, 2)
index_of_2 = my_tuple.index(2)
index_of_2
```

Out[18]: 1

10. To convert tuple into list & vice versa

In [19]:
```
# Converting a tuple to a list or vice versa.
my_tuple = (1, 2, 3)
my_list = list(my_tuple)
```

11. Usage of built-in function

In [22]:
```
# Using built-in functions like `max()`, `min()`, `sum()`.
my_tuple = (5, 2, 8, 1)
max_value = max(my_tuple)
max_value
```

Out[22]: 8