Looping & Joining

Table of contents

01

Looping Tuple

Introduction

- Tuples are **iterable**, allowing iteration using loops like **for** and **while**.
- Looping through a **tuple** in Python is very similar to looping through a **list**.
- Since **tuples** are **ordered collections** like **lists**, the same techniques and principles can be applied to iterate through them.

Let's go through these methods one by one.

Note: If you have already watched the **Looping Through a List** video, you may find this video quite similar.

Looping Tuple: For Loop

- The most straightforward way to loop through a **tuple** is using a **for** loop.

Python

```
fruits = ("apple", "banana", "cherry")
for fruit in fruits:
    print(fruit)
```

Output

```
apple
banana
Cherry
```

Looping Tuple: For Loop

- Use the **range()** function along with **len()** to loop through the index numbers of the tuple.

Python

```
fruits = ("apple", "banana", "cherry")
for i in range(len(fruits)):
    print(fruits[i])
```

Output

```
apple
banana
cherry
```

Looping Tuple: enumerate() Function

- The **enumerate()** function allows you to loop through both the elements and their indices simultaneously.

Python

```
fruits = ("apple", "banana", "cherry")
for index, fruit in enumerate(fruits):
    print(f"Index: {index}, Fruit: {fruit}")
```

Output

```
Index: 0, Fruit: apple
Index: 1, Fruit: banana
Index: 2, Fruit: cherry
```

Looping Tuple: while Loop

- Similar to **lists**, use a **while** loop to iterate through a **tuple** by manually handling the index.

Python

```
fruits = ("apple", "banana", "cherry")
index = 0
while index < len(fruits):
    print(fruits[index])
    index += 1
```

Output

```
apple
banana
cherry
```

Looping Tuple: Loop backward

- Iterate through a tuple in **reverse** order using the **reversed()** function or by using **slicing**.

Python

```
fruits = ("apple", "banana", "cherry")
```

```
for fruit in reversed(fruits):
    print(fruit)
```

```
for fruit in fruits[::-1]:
    print(fruit)
```

Output

```
cherry
banana
apple
```

02

Joining Tuple

Introduction

- **Tuples** in Python are **immutable**, meaning their elements **cannot** be changed after creation.
- However, you can **join** (concatenate) two or more tuples to **create a new tuple**.
- This operation does **not** modify the original **tuples** but returns a **new one**.

Joining Tuple: Concatenation (+)

- The most common way to join tuples is using the **+ operator**, which concatenates two or more tuples into a new one.

Python

```
tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)
tuple3 = tuple1 + tuple2
print(tuple3)
```

Output

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

Joining Tuple: Repetition(*)

- The * **operator** allows repeating a tuple multiple times to create a new tuple.

Python

```
tuple1 = ("A", "B", "C")
tuple2 = tuple1 * 3
print(tuple2)
```

Output

```
('A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C')
```

Joining Tuple: sum() Function

- The **sum()** function can be used to concatenate multiple **tuples** together.
- **sum()** starts with an empty **tuple ()** and adds **tuples one by one.**

Python

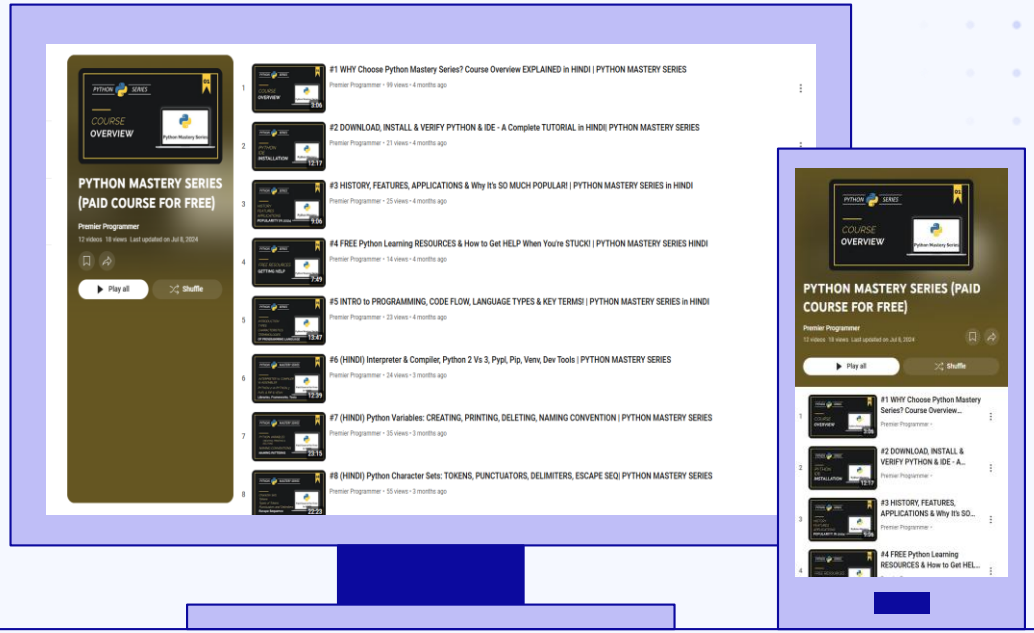
```
tuple1 = (1, 2)
tuple2 = (3, 4)
tuple3 = (5, 6)

result = sum((tuple1, tuple2, tuple3), ())
print(result)
```

Output

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

WATCH

Level up your coding with each episode in this focused Python series.



Next Video!

Tuple Methods