Tuples

Tuples

Кортежи используются для хранения нескольких элементов в одной переменной.
Кортеж - это один из 4 встроенных типов данных в Python, используемых для хранения коллекций данных, остальные 3 типа - это список, набор и словарь, все они имеют различные свойства и способы использования.
Кортеж - это упорядоченная и неизменяемая коллекция. Кортежи записываются с круглыми скобками.

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Tuple items are ordered, unchangeable, and allow duplicate values.
Tuple items are indexed, the first item has index `[0]`, the second item has index `[1]` etc.

Когда мы говорим, что кортежи упорядочены, это означает, что элементы имеют определенный порядок, и этот порядок не изменится.

Кортежи неизменяемы, то есть мы не можем изменять, добавлять или удалять элементы после создания кортежа.

To determine how many items a tuple has, use the `len()` function:

To create a tuple with only one item, you have to add a comma after the item, otherwise Python will not recognize it as a tuple.

```
thistuple = ("apple",)
print(type(thistuple))

#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
```

A tuple can contain different data types:

```
tuple1 = ("abc", 34, True, 40, "male")
```

From Python's perspective, tuples are defined as objects with the data type 'tuple':

```
mytuple = ("apple", "banana", "cherry")
print(type(mytuple))

<class 'tuple'>
```

В языке программирования Python существует четыре типа данных коллекций:

- **[Список]** представляет собой набор, который упорядочен и может быть изменен. Позволяет дублировать участников.
- **(Кортеж)** — это упорядоченная и неизменяемая коллекция. Позволяет дублировать участников.
- **{Набор}** — это неупорядоченная, неизменяемая* и неиндексированная коллекция. Нет повторяющихся членов.
- **Словарь** представляет собой сборник упорядоченный** и изменяемый. Нет повторяющихся членов.

Change Tuple Values:

Once a tuple is created, you cannot change its values. Tuples are **unchangeable**, or **immutable** as it also is called.
But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)
print(x)
```

Since tuples are immutable, they do not have a build-in `append()` method, but there are other ways to add items to a tuple.

1. **Convert into a list**: Just like the workaround for *changing* a tuple, you can convert it into a list, add your item(s), and convert it back into a tuple.

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)
```

2. **Add tuple to a tuple**. You are allowed to add tuples to tuples, so if you want to add one item, (or many), create a new tuple with the item(s), and add it to the existing tuple:

```
thistuple = ("apple", "banana", "cherry")
y = ("orange",)
thistuple += y
```

```
print(thistuple)
```

Tuples are **unchangeable**, so you cannot remove items from it, but you can use the same workaround as we used for changing and adding tuple items:

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
```

We can allowed to extract the values back into variables. This is called "unpacking":

```
fruits = ("apple", "banana", "cherry")
(green, yellow, red) = fruits
print(green)
print(yellow)
print(red)
```

Если звездочка добавлена к другому имени переменной, чем последнее, Python будет присваивать значения переменной до тех пор, пока количество оставшихся значений не совпадет с количеством оставшихся переменных:

```
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry", "Andrii_Beats")
(green, *yellow, red) = fruits
print(green)
print(yellow)
print(red)

apple
['banana', 'cherry', 'strawberry', 'raspberry']
Andrii_Beats
```

Loop Tuples:

1. You can loop through the tuple items by using a `for` loop.

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)
```

2. You can also loop through the tuple items by referring to their index number.

   Use the `range()` and `len()` functions to create a suitable iterable.

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

3. You can loop through the list items by using a `while` loop.

   Use the `len()` function to determine the length of the tuple, then start at 0 and loop your way through the tuple items by refering to their indexes.

   Remember to increase the index by 1 after each iteration.

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

Join Two Tuples:

1. To join two or more tuples you can use the `+` operator:

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)
tuple3 = tuple1 + tuple2
print(tuple3)
```

2. If you want to multiply the content of a tuple a given number of times, you can use the `*` operator:

```
fruits = ("apple", "banana", "cherry")
mytuple = fruits * 2
print(mytuple)
```

Tuple Metods:

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found