# Кортежи (tuple) в python

Данный урок посвящен кортежам `(tuple)` в Python. Основное внимание уделено вопросу использования кортежей, почему иногда лучше применять их, а не списки, рассмотрены способы создания и основные приемы работы с кортежами. Также затронем тему преобразования кортежа в список и обратно.

## Что такое кортеж (tuple) в Python?

Кортеж (tuple) – это неизменяемая структура данных, которая по своему подобию очень похожа на список. Как вы наверное знаете, а если нет, то, пожалуйста, ознакомьтесь с седьмым уроком, список – это изменяемый тип данных. Т.е. если у нас есть список `a = [1, 2, 3]` и мы хотим заменить второй элемент с 2 на 15, то мы может это сделать, напрямую обратившись к элементу списка.

```
a = [1, 2, 3]
print(a)
# >>> [1, 2, 3]
a[1] = 15
print(a)
# >>> [1, 15, 3]
```

С кортежем мы не можем производить такие операции, т.к. элементы его изменять нельзя.

```
b = (1, 2, 3)
print(b)
# (1, 2, 3)
b[1] = 15
# Traceback (most recent call last):
#   File "<pyshell#6>", line 1, in <module>
#     b[1] = 15
# TypeError: 'tuple' object does not support item assignment
```

## Зачем нужны кортежи в Python?

Существует несколько причин, по которым стоит использовать кортежи вместо списков. Одна из них – это обезопасить данные от случайного изменения. Если мы получили откуда-то массив данных, и у нас есть желание поработать с ним, но при этом непосредственно менять данные мы не собираемся, тогда, это как раз тот случай, когда кортежи придутся как нельзя кстати. Используя их в данной задаче, мы дополнительно получаем сразу несколько бонусов – во-первых, это экономия места. Дело в том, что кортежи в памяти занимают меньший объем по сравнению со списками.

```
>>> lst = [10, 20, 30]
>>> tpl = (10, 20, 30)
>>> print(lst.__sizeof__())
32
>>> print(tpl.__sizeof__())
24
```

Во-вторых – прирост производительности, который связан с тем, что кортежи работают быстрее, чем списки (т.е. на операции перебора элементов и т.п. будет тратиться меньше времени). Важно также отметить, что кортежи можно использовать в качестве ключа у словаря.

Создание, удаление кортежей и работа с его элементами Создание кортежей Для создания пустого кортежа можно воспользоваться одной из следующих команд.

```
>>> a = ()
>>> print(type(a))
<class 'tuple'>
>>> b = tuple()
>>> print(type(b))
<class 'tuple'>
```

Кортеж с заданным содержанием создается также как список, только вместо квадратных скобок используются круглые.

```
>>> a = (1, 2, 3, 4, 5)
>>> print(type(a))
<class 'tuple'>
>>> print(a)
(1, 2, 3, 4, 5)
```

При желании можно воспользоваться функцией `tuple()`.

```
>>> a = tuple((1, 2, 3, 4))
>>> print(a)
(1, 2, 3, 4)
```

Доступ к элементам кортежа Доступ к элементам кортежа осуществляется также как к элементам списка – через указание индекса. Но, как уже было сказано – изменять элементы кортежа нельзя!

```
>>> a = (1, 2, 3, 4, 5)
>>> print(a[0])
1
>>> print(a[1:3])
(2, 3)
>>> a[1] = 3
Traceback (most recent call last):
  File "<pyshell#24>", line 1, in <module>
    a[1] = 3
TypeError: 'tuple' object does not support item assignment
```

## Удаление кортежей

Удалить отдельные элементы из кортежа невозможно.

```
>>> a = (1, 2, 3, 4, 5)
>>> del a[0]
Traceback (most recent call last):
  File "<pyshell#26>", line 1, in <module>
    del a[0]
TypeError: 'tuple' object doesn't support item deletion
```

Но можно удалить кортеж целиком.

```
>>> del a
>>> print(a)
Traceback (most recent call last):
  File "<pyshell#28>", line 1, in <module>
    print(a)
NameError: name 'a' is not defined
```

# Преобразование кортежа в список и обратно

На базе кортежа можно создать список, верно и обратное утверждение. Для превращения списка в кортеж достаточно передать его в качестве аргумента функции `tuple()`.

## tuple(lst)

```
>>> lst = [1, 2, 3, 4, 5]
>>> print(type(lst))
<class 'list'>
>>> print(lst)
[1, 2, 3, 4, 5]
>>> tpl = tuple(lst)
>>> print(type(tpl))
<class 'tuple'>
>>> print(tpl)
(1, 2, 3, 4, 5)
```

Обратная операция также является корректной.

## list(tpl)

```
>>> tpl = (2, 4, 6, 8, 10)
>>> print(type(tpl))
<class 'tuple'>
>>> print(tpl)
(2, 4, 6, 8, 10)
>>> lst = list(tpl)
>>> print(type(lst))
<class 'list'>
>>> print(lst)
[2, 4, 6, 8, 10]
```

**Теги:** python types tuple

✎ Отредактировать эту страницу

*Последнее обновление **4 сент. 2023 г.** от **Stavis***