


Использование кортежей tuple



Больше информации на сайте рекламодателя

Узнать больше

[Справочник по языку Python3.](#) / Использование кортежей tuple

[Списки](#) и [строки](#) имеют много общих свойств, например такие как операции [индексации](#) и [среза](#). Это два примера [типов данных последовательности](#). Поскольку Python является развивающимся языком, могут быть добавлены другие типы данных последовательности. Существует также другой стандартный тип данных последовательности - [кортеж](#).

Применение кортежей:

- Хотя кортежи могут показаться похожими на списки, но они часто [используются в разных ситуациях и для разных целей](#). Кортежи являются неизменяемыми и обычно содержат гетерогенную последовательность элементов, доступ к которым осуществляется через распаковку (см. далее в этом разделе) или [индексацию](#), или даже по атрибуту в случае [collections.namedtuple\(\)](#).
- Поскольку кортежи неизменяемы, итерация по кортежу выполняется быстрее, чем по списку. Так что есть небольшой прирост производительности.
- Кортежи, содержащие неизменяемые элементы, могут использоваться в качестве ключа для словаря. Со списками это невозможно.
- Если у вас есть данные, которые не изменяются, реализация их в виде кортежа гарантирует, что они остаются защищенными от записи.

Кортеж состоит из нескольких значений, разделенных запятыми, например:

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
# 12345

>>> t
# (12345, 54321, 'hello!')
```

кортежи могут быть вложенными

```
>>> u = t, (1, 2, 3, 4, 5)
>>> u
# ((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
```

Кортежи неизменяемы:

```
>>> t[0] = 88888
# Traceback (most recent call last):
#   File "<stdin>", line 1, in <module>
# TypeError: 'tuple' object does not support item assignment
```

Кортежи могут содержать изменяемые объекты:

```
>>> v = ([1, 2, 3], [3, 2, 1])
>>> v
# ([1, 2, 3], [3, 2, 1])
```

Кортежи всегда заключаются в круглые скобки, так что вложенные кортежи всегда интерпретируются правильно. Кортежи могут быть записаны в коде с окружающими их скобками или без них, хотя часто скобки все равно необходимы, если кортеж является частью большего выражения. Кортежи невозможно изменить или присвоить другое значения отдельным элементам, однако можно создать кортежи, которые содержат изменяемые объекты, такие как [списки](#).

Особая проблема заключается в создании кортежей, содержащих 0 или 1 элемент. В синтаксисе есть несколько причуд при их создании, к ним просто надо привыкнуть:

- Пустые кортежи состоят из пустой пары скобок.
- кортеж с одним элементом записывается как значение с запятой (недостаточно заключить одно значение в скобки).

Некрасиво, но эффективно. Например:

```
# пустой кортеж
>>> empty = ()

# 1 элемент с запятой в конце
>>> singleton = 'hello',

>>> len(empty)
# 0
>>> len(singleton)
# 1
>>> singleton
# ('hello',)
```

Утверждение `t = 12345, 54321, 'hello!'` - вот пример упаковки кортежей: значения 12345, 54321 и 'hello!' упакованы вместе в кортеж. Возможна и обратная операция - распаковка:

```
>>> x, y, z = t
```

Код выше называется [распаковкой последовательности](#) и работает для любой последовательности с правой стороны. Распаковка [последовательности](#) требует, чтобы в левой части знака равенства было столько же переменных, сколько элементов в последовательности. Обратите внимание, что множественное назначение на самом деле является просто комбинацией упаковки кортежа и распаковки последовательности.

Так-же дополнительно смотрите:

- Встроенный класс [tuple\(\)](#).
- [Тип tuple, кортежи в Python](#).
- [Общие операции с последовательностями list, tuple, str в Python](#).

ХОЧУ ПОМОЧЬ
ПРОЕКТУ

