Модуль collections

Модуль **collections** - предоставляет специализированные типы данных, на основе словарей, кортежей, множеств, списков.

Первым рассматриваемым типом данных будет Counter.

collections.Counter

collections.Counter - вид словаря, который позволяет нам считать количество неизменяемых объектов (в большинстве случаев, строк). Пример:

```
>>> import collections
>>> c = collections.Counter()
>>> for word in ['spam', 'egg', 'spam', 'counter', 'counter',
...     c[word] += 1
...
>>> print(c)
Counter({'counter': 3, 'spam': 2, 'egg': 1})
>>> print(c['counter'])
3
>>> print(c['collections'])
0
```

Но возможности Counter на этом не заканчиваются. У него есть несколько специальных методов:

elements() - возвращает список элементов в лексикографическом порядке.

```
>>> c = Counter(a=4, b=2, c=0, d=-2)
>>> list(c.elements())
['a', 'a', 'a', 'a', 'b', 'b']
```

most_common([n]) - возвращает n наиболее часто встречающихся элементов, в порядке убывания встречаемости. Если n не указано, возвращаются все элементы.

```
>>> Counter('abracadabra').most_common(3)
[('a', 5), ('r', 2), ('b', 2)]
```

subtract([iterable-or-mapping]) - вычитание

```
>>>
```

Поиск …

Свежее

- Модуль csv - чтение и запись CSV файлов
- Создаём сайт на Django, используя хорошие практики. Часть 1: создаём проект
- Онлайн-обучение Python: сравнение популярных программ

Категории

- Книги о Python
- GUI (графический интерфейс пользователя)
- Курсы Python
- Модули
- Новости мира Python
- NumPy
- Обработка данных
- Основы программирования
- Примеры программ
- Типы данных в Python
- Видео
- Python для Web
- Работа для Python-программистов

Полезные материалы

- Сделай свой вклад в развитие сайта!
- Самоучитель Python
- Карта сайта
- Отзывы на книги по Python
- Реклама на сайте

Мы в соцсетях

```
>>> c = Counter(a=4, b=2, c=0, d=-2)
>>> d = Counter(a=1, b=2, c=3, d=4)
>>> c.subtract(d)
Counter({'a': 3, 'b': 0, 'c': -3, 'd': -6})
```

Наиболее часто употребляемые шаблоны для работы с Counter:

- sum(c.values()) - общее количество.
- c.clear() - очистить счётчик.
- list(c) - список уникальных элементов.
- set(c) - преобразовать в множество.
- dict(c) - преобразовать в словарь.
- c.most_common()[:-n:-1] - n наименее часто встречающихся элементов.
- c += Counter() - удалить элементы, встречающиеся менее одного раза.

Counter также поддерживает сложение, вычитание, пересечение и объединение:

>>>

```
>>> c = Counter(a=3, b=1)
>>> d = Counter(a=1, b=2)
>>> c + d
Counter({'a': 4, 'b': 3})
>>> c - d
Counter({'a': 2})
>>> c & d
Counter({'a': 1, 'b': 1})
>>> c | d
Counter({'a': 3, 'b': 2})
```

Следующими на очереди у нас очереди (deque)

collections.deque

collections.deque(iterable, [maxlen]) - создаёт очередь из итерируемого объекта с максимальной длиной maxlen. Очереди очень похожи на списки, за исключением того, что добавлять и удалять элементы можно либо справа, либо слева.

Методы, определённые в deque:

append(x) - добавляет x в конец.

appendleft(x) - добавляет x в начало.

clear() - очищает очередь.

count(x) - количество элементов, равных x.

extend(iterable) - добавляет в конец все элементы iterable.

extendleft(iterable) - добавляет в начало все элементы iterable (начиная с последнего элемента iterable).

pop() - удаляет и возвращает последний элемент очереди.

popleft() - удаляет и возвращает первый элемент очереди.

remove(value) - удаляет первое вхождение value.

reverse() - разворачивает очередь.

rotate(n) - последовательно переносит n элементов из начала в конец (если n отрицательно, то с конца в начало).

collections.defaultdict

collections.defaultdict ничем не отличается от обычного словаря за исключением того, что по умолчанию всегда вызывается функция,

возвращающая значение:

```
>>> import collections
>>> defdict = collections.defaultdict(list)
>>> print(defdict)
defaultdict(<class 'list'>, {})
>>> for i in range(5):
...     defdict[i].append(i)
...
>>> print(defdict)
defaultdict(<class 'list'>, {0: [0], 1: [1], 2: [2], 3: [3], 4
```

collections.OrderedDict

collections.OrderedDict - ещё один похожий на словарь объект, но он помнит порядок, в котором ему были даны ключи. Методы:

popitem(last=True) - удаляет последний элемент если last=True, и первый, если last=False.

move_to_end(key, last=True) - добавляет ключ в конец если last=True, и в начало, если last=False.

```
>>> d = {'banana': 3, 'apple':4, 'pear': 1, 'orange': 2}
>>> OrderedDict(sorted(d.items(), key=lambda t: t[0]))
OrderedDict([('apple', 4), ('banana', 3), ('orange', 2), ('pea
>>> OrderedDict(sorted(d.items(), key=lambda t: t[1]))
OrderedDict([('pear', 1), ('orange', 2), ('banana', 3), ('appl
>>> OrderedDict(sorted(d.items(), key=lambda t: len(t[0])))
OrderedDict([('pear', 1), ('apple', 4), ('orange', 2), ('banar
```

collections.namedtuple()

Класс **collections.namedtuple** позволяет создать тип данных, ведущий себя как кортеж, с тем дополнением, что каждому элементу присваивается имя, по которому можно в дальнейшем получать доступ:

```
>>> Point = namedtuple('Point', ['x', 'y'])
>>> p = Point(x=1, y=2)
>>> p
Point(x=1, y=2)
>>> p.x
1
>>> p[0]
1
```

10 Комментариев

Войти ▾

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ ИЛИ ЧЕРЕЗ DISQUS

Имя

3 Поделиться Лучшие Новые Старые

leksuss
4 года назад

>c += Counter() - удалить элементы, встречающиеся менее одного раза.

То есть, удалить элементы, которые встречаются 0 раз? :) Это как? :)

2 0 Ответить

Альберт Иванов
6 лет назад

>>> c = collections.Counter()
>>> for word in ['spam', 'egg', 'spam', 'counter', 'counter', 'counter']:
... c[word] += 1

это можно сократь до
>>> c = collections.Counter(['spam', 'egg', 'spam', 'counter', 'counter', 'counter'])

4 1 Ответить

Елена Бронированых
6 лет назад

там ошибка. Метод rotate, определённый в deque, переносит n элементов из конца в начало, а при -n - с начала в конец, а не наоборот, как указано в статье.

0 0 Ответить

ДамТеррион
10 лет назад edited

В разделе статьи "collections.defaultdict" почему-то текст (зелёное) начинается со второй кавычки, а не с первой 0_o

А для чего namedtuple нужно? Какие оно имеет преимущества по сравнению с OrdererdDict (или с обычным словарём кроме порядка элементов)?
Почему Counter и OrderedDict с заглавными буквами, а прочие - нет?..

Забавная конструкция:

```
c += Counter()
```

0 0 Ответить

Дмитрий Модератор → ДамТеррион
10 лет назад

Парсер такой. На disqus не лучше, он вообще что-то исправить пытается :(

```
>>> import collections
>>> defdict = collections.defaultdict(list)
>>> defdict
defaultdict(<class 'list'="">, {})
>>> for i in range(5):
    defdict[i].append(i)


>>> defdict
defaultdict(<class 'list'="">, {0: [0], 1: [1], 2: [2], 3: [3],
```

В namedtuple нет хэширований, они неизменяемы.
http://stackoverflow.com/qu...

Так исторически сложилось.
http://stackoverflow.com/qu...

1 0 Ответить

ДамТеррион → Дмитрий

10 лет назад

Я не понимаю, почему неизменяемость является преимуществом

0 0 Ответить

Tom Jones → ДамТеррион

6 лет назад

А вы попробуйте по списку в 10,000 значений пройтись и по кортежу. И сразу поймете разницу.

0 0 Ответить

ДамТеррион → Tom Jones

6 лет назад

Различие во времени обработки. Как это связано с неизменяемостью?

0 0 Ответить

Юрий → ДамТеррион

9 лет назад

Неизменяемые объекты работают быстрее, но я предпочитаю работать с изменяемыми - они удобнее :)