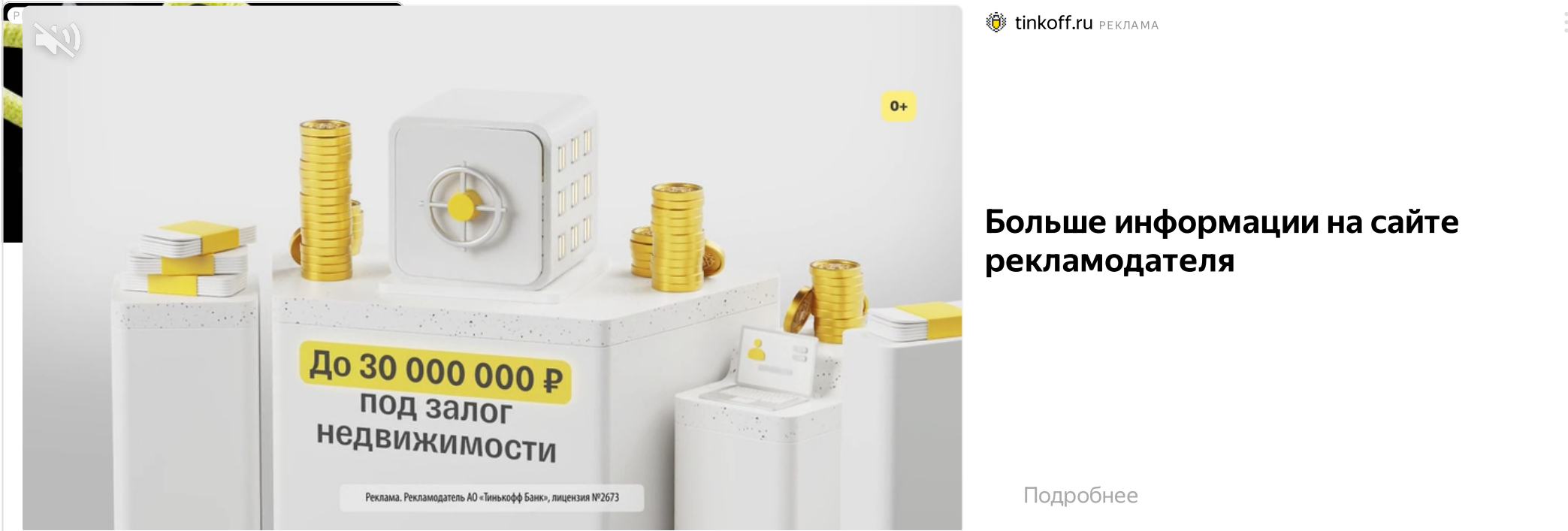


ХОЧУ ПОМОЧЬ
ПРОЕКТУ

Класс defaultdict() модуля collections в Python



Поддержка 24/7

Реальные кейсы

500 часов обучения

Узнать больше

модуль collections в Python, контейнерные типы данных

по умолчанию

ollections ничем не отличается от обычного словаря за исключением того, что по умолчанию всегда возвращает значение по умолчанию для новых значений. Другими словами Класс defaultdict() значениями по умолчанию.

СИНТАКСИС:

```
import collections

d = collections.defaultdict([default_factory[, ...]])
```

Параметры:

- default_factory - тип данных или функция, которая возвращает значение по умолчанию для новых значений.

Возвращаемое значение:

- словарь-подобный объект.

Описание:

Класс defaultdict() модуля collections возвращает новый словарь-подобный объект. Defaultdict является подклассом встроенного класса dict(). Он переопределяет один метод и добавляет одну доступную для записи переменную экземпляра. Остальная функциональность такая же, как и для класса dict().

Первый аргумент предоставляет начальное значение для атрибута default_factory. По умолчанию None. Все остальные аргументы обрабатываются так же, как если бы они были переданы конструктору dict(), включая ключевые аргументы.

Новое в Python 3.9: Добавлена поддержка операторов слияния словарей (|.) и обновления словарей (|=).

Дополнительный метод класса defaultdict():

__missing__(key):

Если атрибут default_factory равен None, то это вызывает исключение KeyError с ключом key в качестве аргумента.

Если default_factory не равен None, то метод __missing__() вызывается без аргументов для предоставления значения по умолчанию для данного ключа, это значение вставляется в словарь для ключа key.

Если вызов default_factory вызывает исключение, это исключение распространяется без изменений.

Метод __missing__() вызывается методом getitem() класса dict(), когда запрошенный ключ key не найден. Все, что он возвращает или поднимает, затем возвращается или вызывается методом __getitem__().

Обратите внимание, что метод __missing__() не вызывается ни для каких операций, кроме как __getitem__(). Это означает, что [метод defaultdict.get()], как и обычные словари, будет возвращать None - как значение по умолчанию, а не использовать default_factory.

Вверх

Переменная экземпляра класса defaultdict():

default_factory:

Этот атрибут используется методом [__missing__\(\)](#). Он инициализируется из первого аргумента, переданного в конструктор, если он не None, иначе устанавливается в None. Если default_factory не None, то defaultdict будет возвращать значение, которое возвращает default_factory, если он отсутствует.

Использование defaultdict() для группировки значений

С помощью функции [default_factory](#), легко сгруппировать последовательность [кортежей](#) ключ-значение в словарь defaultdict.

practicum.yandex.ru

Курс «Мидл python-разработчик». Яндекс Практикум

5,0 ★ Рейтинг организации

Вырасти в backend разработке на Python - 6 месяцев практики на реальных проектах!

Поддержка 24/7

Реальные кейсы

500 часов обучения

Узнать больше

```
d = defaultdict(list)

s = [('yellow', 3), ('blue', 4), ('red', 1)]

for k, v in s:
    d[k].append(v)

sorted(d.items())
# [('blue', [2, 4]), ('red', [1]), ('yellow', [1, 3])]
```

Когда встречается в первый раз, то его еще нет в словаре d и запись d[k] создается автоматически с помощью функции [list.append\(\)](#), которая возвращает пустой [список](#). Затем операция [list.append\(\)](#) присоединяет значение к списку. Если ключ k в d встречается снова, то возвращая список для этого ключа и операция [list.append\(\)](#) добавляет значение v к списку. Это проще и быстрее, чем эквивалентный метод словаря [dict.setdefault\(\)](#):

```
s = [('yellow', 3), ('blue', 4), ('yellow', 3), ('blue', 4), ('red', 1)]
d = {}
for k, v in s:
    d.setdefault(k, []).append(v)

sorted(d.items())
# [('blue', [2, 4]), ('red', [1]), ('yellow', [1, 3])]
```

Установка функции [int\(\)](#) в качестве функции default_factory, генерирующей значений по умолчанию, делает defaultdict() полезным для подсчета чего либо:

```
from collections import defaultdict

s = 'mississippi'
d = defaultdict(int)
for k in s:
    d[k] += 1

sorted(d.items())
# [('i', 4), ('m', 1), ('p', 2), ('s', 4)]
```

Когда буква встречается впервые, она отсутствует в словаре d, поэтому [функция default_factory](#) вызывает [функцию int\(\)](#), чтобы предоставить значение счетчика по умолчанию, которое будет равно нулю.

[Функция int\(\)](#), которая всегда возвращает ноль, является просто частным случаем постоянных [функций](#). Более быстрый и гибкий способ создания постоянных функций заключается в использовании [лямбда-функции](#), которая может предоставлять любое постоянное значение, не только равное 0:

```
from collections import defaultdict

def constant_factory(value):
    return lambda: value

d = defaultdict(constant_factory('<missing>'))
d.update(name='John', action='ran')

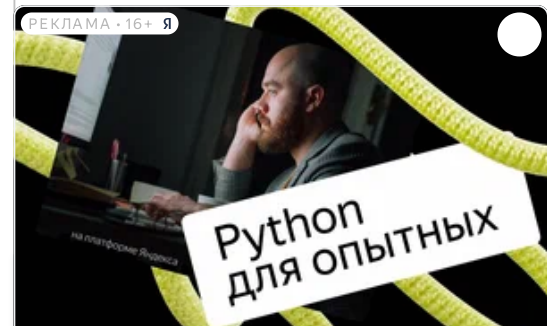
'%(name)s %(action)s to %(object)s' % d
# 'John ran to <missing>'
```


Установка значения default_factory делает defaultdict полезным для создания словаря [Множеств](#):

Вверх

```
from collections import defaultdict

s = [('red', 1), ('blue', 2), ('red', 3), ('blue', 4), ('red', 1), ('blue', 4)]
d = defaultdict(set)
```




 practicum.yandex.ru

Курс «Мидл python-разработчик». Яндекс Практикум

- 5,0

★

Рейтинг организации


- Вырасти в backend разработке на Python - 6 месяцев практики на реальных проектах!
- Поддержка 24/7

>
- Реальные кейсы

>
- 500 часов обучения

>

[DOCS-Python.ru](#)™, 2023 г.

Содержание раздела:

[ns](#)

[s](#)

[tions](#)

[ions](#)

[tions](#)

[ions](#)

(Внимание! При копировании материала ссылка на источник обязательна)

[@docs_python_ru](#)

Вверх