

Сообщить об ошибке.

Метод dict.get() в Python, значение по умолчанию если ключа нет

[Справочник по языку Python3.](#) / [Методы и операции над словарями dict в Python](#)

/ Метод dict.get() в Python, значение по умолчанию если ключа нет

Значение по умолчанию для отсутствующих ключей в словаре

Синтаксис:

```
dict.get(key[, default])
```

Параметры:

- key - ключ словаря
- default - значение по умолчанию

Возвращаемое значение:

- значение ключа key или default если ключа нет.

Описание:

Метод `dict.get()` возвращает значение для ключа `key`, если ключ находится в [словаре](#), если ключ отсутствует то вернет значение `default`.

Если значение `default` не задано и ключ `key` не найден, то метод вернет значение `None`.

Метод `dict.get()` никогда не вызывает исключение `KeyError`, как это происходит в операции [получения значения словаря по ключу `dict[key]`.

Примеры, как работает метод dict.get():

```
>>> x = {'one': 1, 'two': 2, 'three': 3, 'four': 4}
>>> x.get('two', 0)
# 2

>>> x.get('four', 0)
# 4

>>> x.get('ten', 0)
# 0

>>> >>> print(x.get('six'))
# None
```

Где можно применить метод dict.get().

Метод `dict.get()` можно применить, например, для подсчета количества одинаковых элементов последовательности. Допустим есть [список](#) чисел или [слов \(символов\)](#) или то и другое и необходимо узнать, сколько раз КАЖДЫЙ элемент встречается в этом списке. Для решения этой задачи создадим пустой [словарь](#), в который будем добавлять в качестве ключа - элемент списка (словари, в качестве ключа могут принимать неизменяемые значения), а в качестве значения будет количество появлений этого элемента в списке.

Смотрим пример:

```
# имеем список
lst = [9, 13, 1, 3, 7, 3, 1, 1, 7, 1, 7, 9]
# создаем пустой словарь
rez = {}
# в процессе итерации по списку
for el in lst:
    # проверяем есть ли в словаре
```

```
# ключ с элементом `el`
if rez.get(el, None):
    # если есть, то увеличиваем
    # счетчик с этим ключом не 1
    rez[el] += 1
else:
    # если нет, то создаем такой
    # ключ со значением, равным 1
    rez[el] = 1

# смотрим что получилось
print(rez)
# {9: 2, 13: 1, 1: 4, 3: 2, 7: 3}

# чтобы было нагляднее, можно отсортировать словарь
# по значению (количеству появлений элементов в списке)
rez_sorted = sorted(rez.items(), key=lambda x: x[1], reverse=True)
print(dict(rez_sorted))
# {1: 4, 7: 3, 9: 2, 3: 2, 13: 1}
```

Содержание раздела:

- [ОБЗОРНАЯ СТРАНИЦА РАЗДЕЛА](#)
- [Представления словарей dict.keys, dict.values и dict.items](#)
- [Исходный словарь для представления dictview.mapping](#)
- [Получение списка ключей словаря list\(dict\)](#)
- [Количество элементов в словаре len\(dict\)](#)
- [Доступ к значению словаря по ключу dict\[key\]](#)
- [Добавление/изменение значения словаря по ключу key](#)
- [Удаление значения словаря по ключу](#)
- [Проверка наличия/отсутствия ключа key в словаре dict](#)
- [Проверка наличия/отсутствия значения value в словаре Python](#)
- [Проверка наличия/отсутствия пары \(key, value\) в словаре dict](#)
- [Итерирование по ключам и значениям словаря Python](#)
- [Метод dict.clear\(\), Очистить словарь](#)
- [Метод dict.copy\(\), копия словаря](#)
- [Метод dict.fromkeys\(\), словарь с ключами по умолчанию](#)
- [Метод dict.get\(\), значение по умолчанию если ключа нет](#)
- [Метод dict.items\(\), список кортежей](#)
- [Метод dict.keys\(\), список ключей словаря](#)
- [Метод dict.values\(\), список значений словаря](#)
- [Метод dict.pop\(\)](#)
- [Метод dict.popitem\(\), получить пару ключ/значение](#)
- [Метод dict.setdefault\(\), получает/вставляет значение ключа](#)
- [Метод dict.update\(\), обновление/дополнение словаря](#)
- [Объединение двух словарей в новый словарь Python](#)
- [Сортировка словаря по значению и/или ключу](#)
- [Обратный порядок/реверс словаря reversed\(dict\)](#)
- [Генератор словаря и его использование](#)
- [Фильтр словаря по ключам и/или значениям](#)
- [Словарь как фабрика функций](#)

ХОЧУ ПОМОЧЬ
ПРОЕКТУ

[@docs.python.ru](#)