

# Теоретический материал: словари

Сайт: [Дистанционная подготовка](#)

Курс: Д. П. Кириенко. Программирование на языке Python (школа 179 г. Москвы)

Book: Теоретический материал: словари

Printed by: maung myo

Date: Воскресенье 4 Март 2018, 01:22

# Table of Contents

---

[Словари](#)

# Словари

---

Обычные списки (массивы) представляют собой набор пронумерованных элементов, то есть для обращения к какому-либо элементу списка необходимо указать его номер. Номер элемента в списке однозначно идентифицирует сам элемент. Но идентифицировать данные по числовым номерам не всегда оказывается удобно. Например, маршруты поездов в России идентифицируются численно-буквенным кодом (число и одна цифра), также численно-буквенным кодом идентифицируются авиарейсы, то есть для хранения информации о рейсах поездов или самолетов в качестве идентификатора удобно было бы использовать не число, а текстовую строку.

Структура данных, позволяющая идентифицировать ее элементы не по числовому индексу, а по произвольному, называется *словарем* или *ассоциативным массивом*. Соответствующая структура данных в языке Питон называется dict.

Рассмотрим простой пример использования словаря. Заведем словарь Capitals, где индексом является название страны, а значением — название столицы этой страны. Это позволит легко определять по строке с названием страны ее столицу.

```
# Создадим пустой словарь Capitals
Capitals = dict()

# Заполним его несколькими значениями
Capitals['Russia'] = 'Moscow'
Capitals['Ukraine'] = 'Kiev'
Capitals['USA'] = 'Washington'

# Считаем название страны
print('В какой стране вы живете?')
country = input()

# Проверим, есть ли такая страна в словаре Capitals
if country in Capitals:
    # Если есть - выведем ее столицу
    print('Столица вашей страны', Capitals[country])
else:
    # Запросим название столицы и добавив его в словарь
    print('Как называется столица вашей страны?')
    city = input()
    Capitals[country] = city
```

Итак, каждый элемент словаря состоит из двух объектов: *ключа* и *значения*. В нашем примере ключом является название страны, значением является название столицы. Ключ идентифицирует элемент словаря, значение является данными, которые соответствуют данному ключу. Значения ключей — уникальны, двух одинаковых ключей в словаре быть не может.

В жизни широко распространены словари, например, привычные бумажные словари (толковые, орфографические, лингвистические). В них ключом является слово-заголовок статьи, а значением — сама статья. Для того, чтобы получить доступ к статье, необходимо указать слово-ключ.

Другой пример словаря, как структуры данных — телефонный справочник. В нем ключом является имя, а значением — номер телефона. И словарь, и телефонный справочник хранятся так, что легко найти элемент словаря по известному ключу (например, если записи хранятся в алфавитном порядке ключей, то легко можно найти известный ключ, например, бинарным поиском), но если ключ неизвестен, а известно лишь значение, то поиск элемента с данным значением может потребовать последовательного просмотра всех элементов словаря.

Особенностью ассоциативного массива является его динамичность: в него можно добавлять новые элементы с произвольными ключами и удалять уже существующие элементы. При этом размер используемой памяти пропорционален размеру ассоциативного массива. Доступ к элементам ассоциативного массива выполняется хоть и медленнее, чем к обычным массивам, но в целом довольно быстро.

В языке Питон как ключом может быть произвольный неизменяемый тип данных: целые и действительные числа, строки, кортежи. Ключом в словаре не может быть множество, но может быть элемент типа `frozenset`: специальный тип данных, являющийся аналогом типа `set`, который нельзя изменять после создания. Значением элемента словаря может быть любой тип данных, в том числе и изменяемый.

## Когда нужно использовать словари

Словари нужно использовать в следующих случаях:

- Подсчет числа каких-то объектов. В этом случае нужно завести словарь, в котором ключами являются объекты, а значениями — их количество.
- Хранение каких-либо данных, связанных с объектом. Ключи — объекты, значения — связанные с ними данные. Например, если нужно по названию месяца определить его порядковый номер, то это можно сделать при помощи словаря `Num['January'] = 1; Num['February'] = 2; ...`
- Установка соответствия между объектами (например, “родитель—потомок”). Ключ — объект, значение — соответствующий ему объект.
- Если нужен обычный массив, но при этом максимальное значение индекса элемента очень велико, но при этом будут использоваться не все возможные индексы (так называемый “разреженный массив”), то можно использовать ассоциативный массив для экономии памяти.

## Создание словаря

Пустой словарь можно создать при помощи функции `dict()` или пустой пары фигурных скобок `{}` (вот почему фигурные скобки нельзя использовать для создания пустого множества). Для создания словаря с некоторым набором начальных значений можно использовать следующие конструкции:

```
Capitals = {'Russia': 'Moscow', 'Ukraine': 'Kiev', 'USA': 'Washington'}
Capitals = dict(Russia = 'Moscow', Ukraine = 'Kiev', USA = 'Washington')
Capitals = dict([("Russia", "Moscow"), ("Ukraine", "Kiev"), ("USA", "Washington")])
Capitals = dict(zip(["Russia", "Ukraine", "USA"], ["Moscow", "Kiev", "Washington"]))
```

Первые два способа можно использовать только для создания небольших словарей, перечисляя все их элементы. Кроме того, во втором способе ключи передаются как именованные параметры функции `dict`, поэтому в этом случае ключи могут быть только строками, причем являющимися корректными идентификаторами. В третьем и четвертом случае можно создавать большие словари, если в качестве аргументов передавать уже готовые списки, которые могут быть получены не обязательно перечислением всех элементов, а любым другим способом построены по ходу

исполнения программы. В третьем способе функции `dict` нужно передать список, каждый элемент которого является кортежем из двух элементов: ключа и значения. В четвертом способе используется функция `zip`, которой передается два списка одинаковой длины: список ключей и список значений.

## Работа с элементами словаря

Основная операция: получение значения элемента по ключу, записывается так же, как и для списков: `A[key]`. Если элемента с заданным ключом не существует в словаре, то возникает исключение `KeyError`.

Другой способ определения значения по ключу — метод `get`: `A.get(key)`. Если элемента с ключом `key` нет в словаре, то возвращается значение `None`. В форме записи с двумя аргументами `A.get(key, val)` метод возвращает значение `val`, если элемент с ключом `key` отсутствует в словаре.

Проверить принадлежность элемента словарю можно операциями `in` и `not in`, как и для множеств.

Для добавления нового элемента в словарь нужно просто присвоить ему какое-то значение: `A[key] = value`.

Для удаления элемента из словаря можно использовать операцию `del A[key]` (операция возбуждает исключение `KeyError`, если такого ключа в словаре нет. Вот два безопасных способа удаления элемента из словаря. Способ с предварительной проверкой наличия элемента:

```
if key in A:
    del A[key]
```

Способ с перехватыванием и обработкой исключения:

```
try:
    del A[key]
except KeyError:
    pass
```

Еще один способ удалить элемент из словаря: использование метода `pop`: `A.pop(key)`. Этот метод возвращает значение удаляемого элемента, если элемент с данным ключом отсутствует в словаре, то возбуждается исключение. Если методу `pop` передать второй параметр, то если элемент в словаре отсутствует, то метод `pop` возвратит значение этого параметра. Это позволяет проще всего организовать безопасное удаление элемента из словаря: `A.pop(key, None)`.

## Перебор элементов словаря

Можно легко организовать перебор ключей всех элементов в словаре:

```
for key in A:
    print(key, A[key])
```

Следующие методы возвращают *представления* элементов словаря. Представления во многом похожи на множества, но они изменяются, если менять значения элементов словаря. Метод `keys` возвращает представление ключей всех элементов, метод `values` возвращает представление всех значений, а метод `items` возвращает представление всех пар (кортежей) из ключей и значений.

Соответственно, быстро проверить, если ли значение `val` среди всех значений элементов словаря `A` можно так: `val in A.values()`, а организовать цикл так, чтобы в переменной `key` был ключ элемента, а в переменной `val` было его значение можно так:

```
for key, val in A.items():  
    print(key, val)
```