* [ОБЗОР КУРСА](https://lyceum.yandex.ru/courses/123/groups/568)

[Урок Словари](https://lyceum.yandex.ru/courses/123/groups/568/lessons/662)

**Знакомство со словарями**

**План урока**

1

[Знакомство со словарями](https://lyceum.yandex.ru/courses/123/groups/568/lessons/662/materials/1060#1)

2

[Создание словаря](https://lyceum.yandex.ru/courses/123/groups/568/lessons/662/materials/1060#2)

3

[Обращение к элементу словаря](https://lyceum.yandex.ru/courses/123/groups/568/lessons/662/materials/1060#3)

4

[Добавление и удаление элементов](https://lyceum.yandex.ru/courses/123/groups/568/lessons/662/materials/1060#4)

5

[Проверка наличия элемента в словаре](https://lyceum.yandex.ru/courses/123/groups/568/lessons/662/materials/1060#5)

6

[Нестроковые ключи](https://lyceum.yandex.ru/courses/123/groups/568/lessons/662/materials/1060#6)

7

[Методы словарей](https://lyceum.yandex.ru/courses/123/groups/568/lessons/662/materials/1060#7)

8

[Допустимые типы ключей](https://lyceum.yandex.ru/courses/123/groups/568/lessons/662/materials/1060#8)

**Аннотация**

*В этом уроке рассказывается о словарях — встроенной в Python мощной структуре данных. В других языках аналогичная структура называется map, HashMap, Dictionary.*

*Базовые функции работы со словарями показаны на простых примерах хранения библиотеки знаний о фильмах и актёрах.*

**1. Знакомство со словарями**

Списки — это удобный и самый популярный способ сохранить большое количество данных в одной переменной. Списки индексируют все хранящиеся в них элементы. Первый элемент, как мы помним, лежит по индексу 0, второй по индексу 1 и так далее. Такой способ хранения позволяет быстро обращаться к элементу списка, зная его индекс.

actors = ['Джонни Депп', 'Эмма Уотсон', 'Билли Пайпер']

**print**(actors[1])

Представим, что мы делаем свою онлайн-энциклопедию (наподобие Википедии) об актёрах мирового кино. Для каждого актёра нужно сохранить текст статьи о нём. Названием статьи будет строка, состоящая из фамилии и имени актёра. Как правильно хранить такие данные?

Можно создать список кортежей. Каждый кортеж будет состоять из двух строк — названия и текста статьи.

actors = [  
 ('Джонни Депп', 'Джон Кристофер Депп Второй родился 9 июня   
 1963 года в Овенсборо, Кентукки..'),

('Сильвестр Сталлоне', 'Сильвестр Гарденцио Сталлоне родился   
 в Нью-Йорке. Его отец — иммигрант из Сицилии, ...'),

('Эмма Уотсон', 'Эмма Шарлотта Дуерр Уотсон родилась в семье  
 английских адвокатов. В пять лет переехала вместе с семьей   
 из Парижа в Англию...'),

*# ...*

]

Со временем количество статей значительно вырастет. Чтобы найти нужную статью по названию, нам придётся написать цикл for, который пройдёт по всем элементам списка **actors** и найдёт в нём кортеж, первый элемент которого равен искомому названию. В приведённом выше примере, чтобы найти статью об Эмме Уотсон, нам придётся в цикле пройти мимо Джонни Деппа и Сильвестра Сталлоне. Угадать заранее, что статья об Эмме Уотсон лежит после них, не получится.

Корень этой проблемы в том, что списки индексируются целыми числами. Мы же хотим находить информацию не по числу, а по строке — названию статьи. Было бы здорово, если бы индексами могли быть не числа, а строки. В списках это невозможно, однако возможно в словарях!

**Словарь**

Словарь (в Python он называется **dict**) — это тип данных, позволяющий, как и список, хранить много данных. В отличие от списка в словаре для каждого элемента можно самому определить «индекс», по которому он будет доступен. Этот индекс называется **ключом**.

**2. Создание словаря**

Вот пример создания словаря для энциклопедии об актёрах мирового кино:

actors = {

'Джонни Депп': 'Джон Кристофер Депп Второй родился 9 июня 1963 года

в Овенсборо, Кентукки..',

'Сильвестр Сталлоне': 'Сильвестр Гарденцио Сталлоне родился в Нью-Йорке.

Его отец, парикмахер Фрэнк Сталлоне — иммигрант из Сицилии, ...',

'Эмма Уотсон': 'Эмма Шарлотта Дуерр Уотсон родилась в семье английских   
 адвокатов. В пять лет переехала вместе с семьей из Парижа в Англию...',

*# ...*

}

**Важно**

Элементы словаря перечисляются в фигурных скобках (как и элементы множества!) и разделяются запятой. До двоеточия указывается ключ, а после двоеточия — значение, доступное в словаре по этому ключу.

**Пустой словарь** можно создать двумя способами:

d = dict()

*# или так*

d = {}

Вспомните, что создать пустое множество можно только используя функцию **set()**. Теперь понятно, почему это так — пустые фигурные скобки зарезервированы для создания словаря.

**3. Обращение к элементу словаря**

После инициализации словаря мы можем быстро получать статью про конкретного актёра:

**print**(actors['Эмма Уотсон'])

**Важно**

Обращение к элементу словаря выглядит как обращение к элементу списка, только вместо целочисленного индекса используется ключ. В качестве ключа можно указать выражение — Python вычислит его значение, прежде чем обратится к искомому элементу.

first\_name = 'Сильвестр'

last\_name = 'Сталлоне'

**print**(actors[first\_name + ' ' + last\_name])

Если ключа в словаре нет, возникнет ошибка:

**print**(actors['Несуществующий ключ'])

------------------------------------------

KeyError: 'Несуществующий ключ'

**4. Добавление и удаление элементов**

Важная особенность словаря — его динамичность. Мы можем добавлять новые элементы, изменять их или удалять. Изменяются элементы точно так же, как в списках, только вместо целочисленного индекса в квадратных скобках указывается ключ:

actors['Эмма Уотсон'] = 'Новый текст статьи об Эмме Уотсон'

Также в словари можно добавлять и удалять новые элементы.

**Добавление элемента**

Добавление синтаксически выглядит так же, как и изменение:

actors['Брэд Питт'] = 'Уильям Брэдли Питт, более известный как   
 Брэд Питт — американский актёр и продюсер.   
 Лауреат премии «Золотой глобус» за 1995 год   
 ...'

**Удаление элемента**

Для удаления можно использовать инструкцию **del** (как и в списках):

**del** actors['Джонни Депп']

*# больше в словаре нет ни ключа 'Джонни Депп',   
# ни соответствующего ему значения*

**print**(actors['Джонни Депп'])

KeyError: 'Джонни Депп'

**Удаление элемента**

Удалять элемент можно и по-другому:

actors.pop('Джонни Депп')

Единственное отличие этого способа от вызова **del** — он возвращает удалённое значение. Можно написать так:

deleted\_value = actors.pop('Джонни Депп')

В переменную deleted\_value положится значение, которое хранилось в словаре по ключу ’Джонни Депп’. В остальном этот способ идентичен оператору **del**. В частности, если ключа ’Джонни Депп’ в словаре нет, возникнет ошибка **KeyError**.

**Важно**

Чтобы ошибка не появлялась, этому методу можно передать второй аргумент. Он будет возвращён, если указанного ключа в словаре нет. Это позволяет реализовать безопасное удаление элемента из словаря:

deleted\_value = actors.pop('Джонни Депп', None)

Если ключа ’Джонни Депп’ в словаре нет, в deleted\_value попадёт **None**.

**5. Проверка наличия элемента в словаре**

Оператор **in** позволяет проверить, есть ли ключ в словаре:

**if** 'Джонни Депп' **in** actors:

**print**('У нас есть статья про Джонни Деппа')

Проверить, что ключа нет, можно с помощью аналогичного оператора **not in**:

**if** 'Сергей Безруков' **not** **in** actors:

**print**('У нас нет статьи о Сергее Безрукове')

**6. Нестроковые ключи**

Решим следующую задачу. Пусть дан длинный список целых чисел numbers. Мы знаем, что некоторые числа встречаются в этом списке несколько раз. Нужно узнать, сколько именно раз встречается каждое из чисел.

numbers = [1, 10, 1, 6, 4, 10, 4, 2, 2, 1, 10, 1]

counts = {}

**for** number **in** numbers:

**if** number **not** **in** counts:

counts[number] = 1

**else**:

counts[number] += 1

Просто так сделать counts[number] += 1 нельзя: если ключа number в словаре нет, возникнет ошибка **KeyError**.

В результате работы этой программы все элементы из списка numbers окажутся ключами словаря counts. Значением counts[x] будет количество раз, которое число x встретилось в списке numbers. Как это работает?

Цикл for перебирает все элементы списка numbers и для каждого проверяет, присутствует ли он уже в качестве ключа в counts. Если нет — значит, число встретилось нам впервые, и мы инициализируем значение counts[numbers] = 1. Иначе увеличим counts[number] на единицу, поскольку число number встретилось нам повторно.

Почему для этой задачи не стоит использовать список, хотя ключи — обычные целые числа? Потому что используя словарь, мы можем решить эту задачу и для вещественных чисел, и для очень больших целых чисел, и вообще для любых объектов, которые можно сравнивать.

**7. Методы словарей**

Взять значение в словаре можно не только с помощью квадратных скобок, но и с помощью метода **get**:

article = actors.get('Джонни Депп')

Преимущество метода в том, что кроме ключа он может принимать и второй аргумент — значение, которое вернётся, если заданного ключа нет:

article = actors.get('Джонни Депп', 'Статья о Джонни Деппе не найдена')

Воспользуемся этим приёмом для улучшения нашей программы в задаче о повторяющихся числах:

numbers = [1, 10, 1, 6, 4, 10, 4, 2, 2, 1, 10, 1]

counts = {}

**for** number **in** numbers:

counts[number] = counts.get(number, 0) + 1

Попробуйте понять, почему это работает верно.

Все ключи словаря можно перебрать циклом **for**:

**for** actor\_name **in** actors:

**print**(actor\_name, actors[actor\_name])

Другой способ сделать то же самое — вызвать метод **.keys()**:

**for** actor\_name **in** actors.keys():

**print**(actor\_name, actors[actor\_name])

**Метод keys()**

С помощью метода .keys() можно получить список всех ключей словаря:

actors\_names = list(actors.keys())

**Метод values()**

Есть и парный метод .values(), возвращающий все значения словаря:

all\_articles = list(actors.values())

Он позволяет, например, проверить, если ли какое-нибудь значение value среди значений словаря:

value **in** d.values()

**Метод items()**

Если вы хотите перебрать элементы словаря d так, чтобы в переменной key оказывался ключ, а в value — соответствующее ему значение, то это можно сделать с помощью метода .items() и цикла for.

**for** key, val **in** d.items():

Например:

**for** actor\_name, article **in** actors.items():

**print**(actor\_name, article)

**8. Допустимые типы ключей**

Мы уже выяснили, что ключами в словарях могут быть строки и целые числа. Кроме этого, ключами могут быть вещественные числа и кортежи.

**Важно**

Ключами в словаре не могут быть другие словари. В принципе, в одном словаре могут быть ключи разных типов, однако обычно принято использовать однотипные ключи. Вообще, есть строгий способ определить, может ли объект быть ключом в словаре. Для этого объект должен быть **неизменяемым**. Неизменяемые объекты не могут поменять значение в себе во время выполнения программы. Неизменяемыми в Python являются числа, строки и кортежи. Именно их обычно и используют в качестве ключей словарей.

Вот как может выглядеть список с ключами-кортежами. В качестве ключа используются координаты, а в качестве значения — название города.

cities = {  
 (55.75, 37.5): 'Москва',   
 (59.8, 30.3): 'Санкт-Петербург',   
 (54.32, 48.39): 'Ульяновск'  
}

**print**(cities[(55.75, 37.5)])

cities[(53.2, 50.15)] = 'Самара'

Возможно, нам захочется развернуть этот словарь, то есть построить такой словарь, в котором ключами будут города, а значениями — их координаты.

coordinates = {}

**for** coordinate, city **in** cities.items():

coordinates[city] = coordinate

Если в исходном словаре были повторяющиеся значения, то некоторые значения потеряются при разворачивании словаря. Это объясняется тем, что значения в словаре могут повторяться, а вот ключи обязаны быть уникальными.

Значениями в словаре, в отличие от ключей, могут быть объекты любого типа: числа, строки, кортежи, списки и даже другие словари. Вот, например, как можно сохранить список фильмов для каждого из актёров:

films = {

'Джонни Депп':   
 [  
 'Эдвард Руки-Ножницы',   
 'Одинокий рейнджер',   
 'Чарли и шоколадная фабрика',   
 ..],

'Эмма Уотсон':   
 [  
 'Гарри Поттер и философский камень',   
 'Красавица и Чудовище',   
 ..],

*# ...*

}

*# Вывести список фильмов, в которых снималась Эмма Уотсон*

**print**(films['Эмма Уотсон'])

*# Проверить, снимался ли Джонни Депп в фильме «Чарли и шоколадная фабрика»*

**if** 'Чарли и шоколадная фабрика' **in** films['Джонни Депп']:

**print**('Снимался!')

[Справка](https://yandex.ru/support/lyceum-students)

Исключительное право на учебную программу и все сопутствующие ей учебные материалы, доступные в рамках проекта «Яндекс.Лицей», принадлежат АНО ДПО «ШАД». Воспроизведение, копирование, распространение и иное использование программы и материалов допустимо только с предварительного письменного согласия АНО ДПО «ШАД».

© 2018 – 2020  ООО «[Яндекс](https://yandex.ru/)»

Чаты