Урок 11.  
Словари.

короткая линия

# План урока

1. Словари
2. Методы словарей
3. Генераторы словарей

# Словари

В этом модуле мы будем заниматься изучением структур данных. Некоторые из них встроены в язык Python, другие же необходимо будет реализовать самим. Более того, с простейшими структурами данных Вы уже знакомы – это кортеж (tuple) и список (list).

Структуры данных призваны упорядочить принимаемые данные в определенном виде. Кортежи и списки просто хранят данные различных типов, поэтому являются простейшими примерами. Однако есть структуры, которые представляют данные в виде графов (например, как расстояния между населенными пунктами), очереди, двоичного дерева и т. д.

Храня элемент в списке, мы можем однозначно его идентифицировать по числовому индексу. Однако это не всегда удобно: для электронной телефонной книги удобнее получать телефоны не по индексу владельца в списке, а по его имени; для электронной библиотеки к каждой книге удобнее обращаться по названию, а к списку книг – по имени автора. Такое поведение возможно благодаря словарям. Словарь хранит данные в формате “ключ-значение”. Чтобы получить нужные Вам данные, необходимо запросить их по ключу. Создать пустой словарь можно несколькими способами:

1. С помощью фигурных скобок:

|  |
| --- |
| dictionary = {} |

2. С помощью функции dict():

dictionary = dict()

Также можно создать словарь с некоторыми начальными значениями. Пары “ключ-значение” в словаре разделяются двоеточием. Создадим небольшой географический справочник, хранящий пары “страна-столица”:

|  |
| --- |
| capitals = {**"Spain"**: **"Madrid"**, **"England"**: **"London"**, **"USA"**: **"Washington"**, **"Latveria"**: **"Doomstadt"**} |

или

|  |
| --- |
| capitals = dict(Spain=**"Madrid"**, England=**"London"**, USA=**"Washington"**, Latveria=**"Doomstadt"**) |

или

|  |
| --- |
| capitals = dict([(**"Spain"**, **"Madrid"**), (**"England"**, **"London"**), (**"USA"**, **"Washington"**), (**"Latveria"**, **"Doomstadt"**)]) |

Обращаться к элементам словаря можно так же, как и в списках, кортежах, строках, только не по числовому индексу, а по ключу:

|  |
| --- |
| spain\_capital = capitals[**"Spain"**] print(**"I want to go to"**, spain\_capital) |

Имена ключей уникальны, поэтому если Вы несколько раз дадите значение одному ключу, останется только последнее значение. Переприсваивать значения, а также добавлять значения в словарь можно и намеренно, если в программе что-то меняется в ходе работы:

|  |
| --- |
| *# добавим Австралию* capitals[**"Australia"**] = **"Sydney"** *# упс, неправильно указали столицу, перезапишем* capitals[**"Australia"**] = **"Canberra"** |

Необходимо упомянуть, что ключами в словаре могут быть только неизменяемые типы. Так, Вы не сможете сделать ключом список, другой словарь, множество (тема следующего урока). Это происходит потому, что для хранения элементов Python вычисляет [хэш](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) (грубо говоря, id объекта в памяти) ключа. После изменения объекта его хэш, естественно, изменяется, что может повлечь за собой повреждение структуры словаря. Однако на значения ключа нет никаких ограничений, поэтому можно сделать так:

|  |
| --- |
| authors = {**"Лев Толстой"**: [**"Война и мир"**, **"Анна Каренина"**, **"Воскресение"**], **"Джек Лондон"**: [**"Белый клык"**, **"Мартин Иден"**, **"Сердца трех"**]} |

**Задача “Телефонная книга. Этап 1”**

# Методы словарей

Для иллюстрации методов будем использовать словарь d.

**d.get(key)**

Возвращает значение по переданному ключу key. Использовать этот метод удобнее, чем индексацию в квадратных скобках, так как, если ключа не существует, метод вернет None, а обычная индексация выбросит ошибку. Также вторым аргументом можно передать значение, которое вернется, если ключа не существует. Посчитаем количество появлений каждого слова в тексте:

|  |
| --- |
| text = input().split() *# дополнительно можно убрать знаки препинания*  counts = {} **for** word **in** text:  counts[word] = counts.get(word, 0) + 1 print(counts) |

**d.update(dict)**

Обновляет словарь d, принимая на вход другой словарь. Таким образом можно как добавить новые значения, так и перезаписать старые.

capitals = {**"Spain"**: **"Madrid"**, **"England"**: **"London"**, **"USA"**: **"Washington"**, **"Latveria"**: **"Doomstadt"**}  
capitals.update({**"Russia"**: **"Moscow"**, **"France"**: **"Paris"**, **"Latveria"**: **"No data"**})

**d.pop(key)**

Аналогично методу списков удаляет элемент словаря по ключу и возвращает соответствующее значение.

**d.keys()**

Возвращает список всех ключей словаря.

**d.values()**

Возвращает список всех значений словаря

**d.items()**

Возвращает список кортежей, каждый из которых представляет пару “ключ-значение”.

Естественно, это не полный список методов. Рекомендуем ознакомиться с документацией: <https://docs.python.org/3/library/stdtypes.html#typesmapping>

Кроме этих методов, к словарям также можно применять функцию len(), а также функции min() и max(), но только если ключи словаря представляют собой один и тот же тип данных (подумайте, из-за чего возникло такое ограничение).

По ключам словаря можно проходиться с помощью цикла for:

|  |
| --- |
| **for** capital **in** capitals:  print(**"The capital of"**, capital, **"is"**, capitals[capital]) |
|  |

Как и в строках, кортежах и списках, можно применять оператор членства **in**. В этом случае оператор будет искать переданный ключ в словаре.

# Генераторы словарей

Если Вам нравятся генераторы списков, скорее всего, Вам захочется использовать ту же конструкцию для словарей. К счастью, это возможно. Создадим словарь, сопоставив каждому числу из диапазона [1; 100] его квадрат:

numbers = {x: x \*\* 2 **for** x **in** range(1, 101)}

**Задача “Словарь программиста”**

**Задача “Маленький частотный анализ”**

**Задача “Права доступа”**

**Задача “Телефонная книга. Этап 2”**

**Задача “Инверсия справочника”**

**Задача “Азбука Морзе”**

**Задача “Жаркий аукцион”**

**Задача “Bookflix”**

**Задача “Телефонная книга. Этап 3”**

**Задача “Частотный анализ”**

**Задача “Список книг на лето”**