# Словари

Словарь – набор объектов, каждый из которых является парой вида ***ключ : значение***. Ключи используются для доступа к значениям элементов словаря так же, как индексы для доступа к элементам списка. В качестве ключа чаще всего используется строка, но можно использовать и любые другие неизменяемые объекты, например, числа или кортежи. Значения могут быть любого типа. Все ключи в словаре уникальные. Порядок элементов в словаре не определен, поэтому они не являются последовательностями. В терминологии Python словари являются отображениями.

Словари удобны для хранения объектов, имеющих именованные поля.

Главное достоинство словарей – быстрый поиск среди неупорядоченных данных.

***Создание словаря***

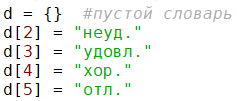
Создать словарь можно различными способами:

* явно указав все элементы словаря в тексте. Элементы словаря перечисляются в фигурных скобках через запятую. Для каждого элемента указываются ключ и значение, которые друг от друга отделяются двоеточием.





* заполнив словарь поэлементно:

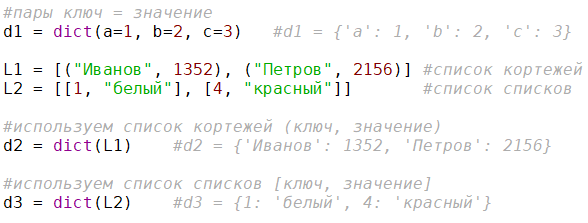


В правой части оператора присваивания указывается значение элемента. В левой – имя словаря и в квадратных скобках ключ. В данном случае в качестве ключей используются числа 2, 3, 4, 5. Словарь d будет иметь значение:



* с помощью функции dict().

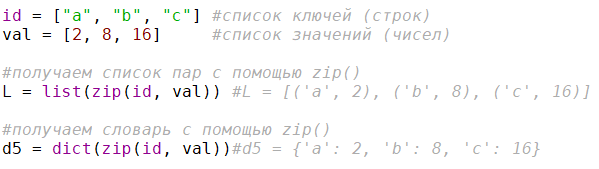
Функция обычно используется для создания словаря из данных других типов. В этом случае данные, как правило, являются последовательностью пар значений. Один элемент пары становится ключом, другой – значением элемента словаря:



Функция может применяться и для создания поверхностной копии словаря.



Если данных в виде пар нет, то их можно создать, используя функцию zip(). Вообще-то zip() умеет создавать не только пары, но и тройки, четверки и т.д. значений, но в данном случае нам нужны именно пары. Функция zip() возвращает последовательность, которую можно преобразовать в список или словарь:

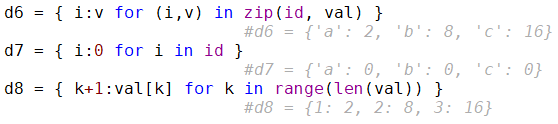


* с помощью генератора.

Генераторы словарей отличаются от генераторов списков тем, что:

* выражение заключается в фигурные скобки, а не в квадратные;
* внутри фигурных скобок перед for указываются два значения через двоеточие (слева от двоеточия ключ, справа – значение).

Примеры:



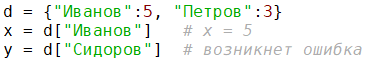
Как и в случае списков генераторы словарей могут содержать несколько вложенных циклов for и инструкцию ветвления if после циклов:



***Операции над словарями***

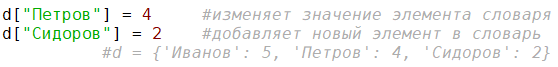
* доступ по ключу ( [] )

Для получения значения элемента словаря с заданным ключом нужно указать имя словаря и значение ключа в квадратных скобках. Если в словаре нет элемента с таким ключом, возникнет ошибка:



Чтобы подобные ошибки не возникали, можно либо вначале проверять наличие элемента в словаре, либо использовать метод get(), либо перехватывать эти ошибки (обрабатывать исключение KeyError).

Словарь – изменяемый тип данных, поэтому можно не только получить значение по ключу, но и изменить его. Для этого имя словаря и ключ должны быть указаны в левой части оператора присваивания. В правой части указывается новое значение элемента. Если элемента с нужным ключом в словаре нет, то он создается. Этим часто пользуются, чтобы добавлять элементы в словарь. Пример:



* проверка наличия элемента с заданным ключом ( in )

Оператор in проверяет наличие элемента с указанным ключом в словаре. Возвращает значения True или False:



* удаление ( del )

Удаляет из словаря элемент с заданным ключом. Если элемент с таким ключом отсутствует, то возникает ошибка (возбуждается исключение KeyError). Пример:



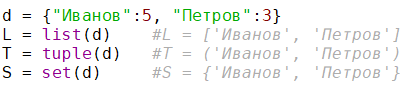
Словари не являются последовательностями, поэтому операции получения среза, конкатенации, повторения для них недопустимы.

Многие действия над словарями выполняются с помощью функций и методов.

***Функции для работы со словарями***

**dict()** – создает словарь (примеры приведены выше при описании способов создания словарей);

**list(), tuple(), set()** – преобразуют словарь в объект соответствующего типа (список, кортеж или множество ключей словаря):



**len()** – возвращает количество элементов в словаре.

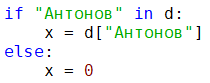
***Методы словарей***

Здесь рассматриваются только основные методы словарей. Полное описание следует смотреть в справочной литературе.

**get()** – возвращает значение элемента словаря. Ключ указывается в первом параметре метода. Если в словаре нет такого ключа, то возвращается значение второго (необязательного) параметра. По умолчанию значение второго параметра равно None. В отличие от операции [] здесь ошибка не возникает даже при отсутствии элемента в словаре. Например, инструкция



делает то же самое, что



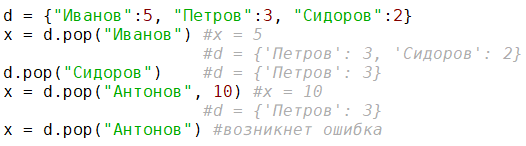
Другие примеры:



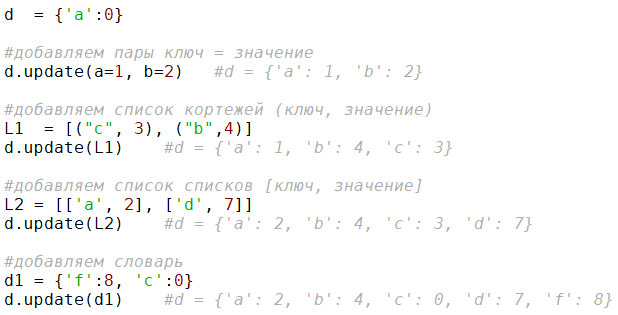
**clear()** – удаляет все элементы словаря. Метод ничего не возвращает:



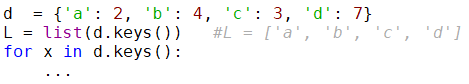
**pop()** – удаляет элемент с ключом, указанном в первом параметре, и возвращает значение, соответствующее этому ключу. Если ключ отсутствует, то возвращается значение второго (необязательного) параметра. Если второй параметр не задан, то возникнет ошибка (исключение KeyError):



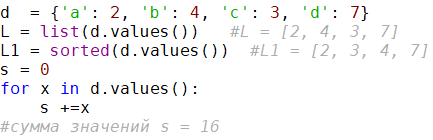
**update()** – добавляет элементы в словарь. Метод изменяет текущий словарь и ничего не возвращает. Параметры могут быть такими же, как у функции dict(). Если элемент с указанным ключом уже есть в словаре, то его значение заменяется на новое:



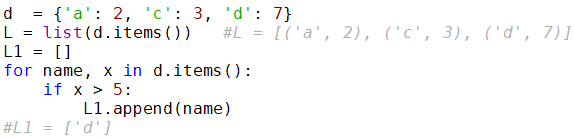
**keys()** – возвращает объект, содержащий все ключи словаря. Этот объект поддерживает итерации (можно использовать в цикле for) и операции над множествами:



**values()** – возвращает объект, содержащий все значения словаря. Объект поддерживает итерации (можно использовать в цикле for):



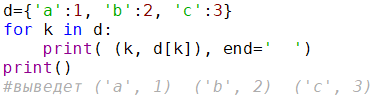
**items()** – возвращает объект, содержащий все ключи и значения в виде кортежей. Объект поддерживает итерации (можно использовать в цикле for):



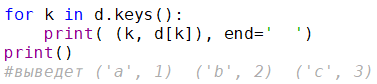
***Перебор элементов словаря***

Перебор всех элементов словаря выполняется с помощью цикла for.

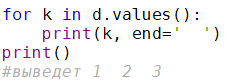
Если в заголовке цикла указать просто имя словаря, то переменная цикла на каждой итерации будет равна ключу очередного элемента словаря. Доступ к значению при необходимости выполняется стандартным образом. Учтите, что порядок элементов в словарях не определен, поэтому ключи могут выдаваться совсем не в том порядке, в котором элементы добавлялись в словарь:



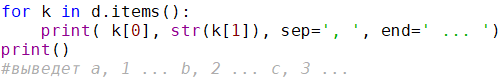
Если использовать в заголовке цикла метод keys(), то результат будет точно такой же:



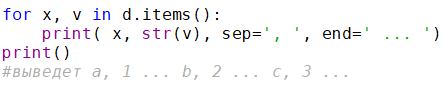
Если использовать в заголовке цикла метод values(), то переменная цикла на каждой итерации будет равна значению очередного элемента словаря. Ключи в таком варианте цикла будут недоступны. Пример:



Если использовать в заголовке цикла метод items(), то значением переменной цикла k будет кортеж из двух элементов. Тогда k[0] – это ключ очередного элемента словаря, а k[1] – значение этого элемента:



Чтобы не использовать индексы кортежа, удобно его распаковать:

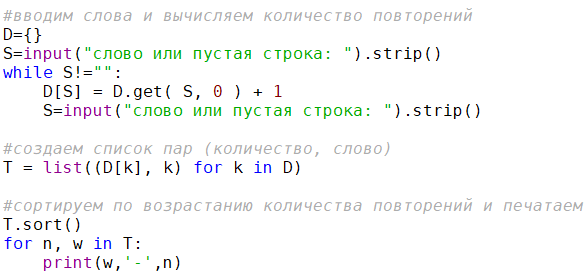


***Пример использования словаря***

Для введенных слов (каждое слово вводится отдельно, признак конца ввода – пустая строка) вычислить, сколько раз они вводились. Вывести слова и полученные значения на экран в порядке возрастания частоты встречаемости слов. Для простоты под словом будем понимать любую последовательность символов.

Для хранения информации используем словарь, в котором ключами будут введенные слова, а значениями – количество повторений слова.

Чтобы вывести информацию на экран, словарь сначала преобразуем в список пар (кортежей из двух элементов). Список отсортируем, так как словари сортировать нельзя. Используя полученный отсортированный список, выведем на экран информацию в нужном виде. Обратите внимание на элементы списка: ключ – второй элемент кортежа, значение – первый элемент кортежа. При сортировке пары сортируются сначала по первому элементу, т.е. по числу вхождений слова.



***Задания для самостоятельной работы***

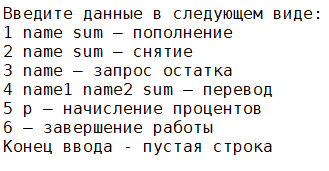
1. Дана строка, состоящая из слов, разделенных пробелами. Выведите все слова, встречающиеся в этой строке, на экран. Рядом выведите, сколько раз встречается слово в исходной строке. Слова должны быть отсортированы по убыванию их количества появления в тексте.
2. Имеется строка с названиями товаров вида «яблоки, груши, яблоки, киви, сливы, киви». Товары перечислены через запятую, товары могут повторяться. Выведите название товара, который встречается в этой строке чаще всего. Если таких товаров несколько, то названия всех товаров.
3. Число от 2 до 55, которое может содержать только цифры 2, 3, 4, 5 записано прописью, например, «тридцать три». Вычислите квадратный корень из этого числа.
4. Имеется список названий месяцев: [‘января’, ‘февраля’, ‘марта’, ‘апреля’, ’мая’, ‘июня’, ‘июля’, ‘августа’, ‘сентября’, ‘октября’, ‘ноября’, ‘декабря’]. Создайте по этому списку словарь, в котором название месяца будет ключом, а номер месяца (от 1 до 12) – значением.
5. При работе со счетами клиентов поддерживаются следующие операции:

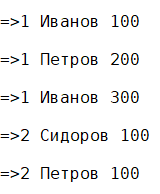
* 1 – пополнение счета клиента;
* 2 – снятие денег со счета;
* 3 – запрос остатка средств на счете;
* 4 – перевод денег между счетами клиентов;
* 5 – начисление процентов всем клиентам;
* 6 – завершение работы с программой.

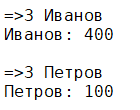
Во время работы программы информация об операциях вводится в виде:

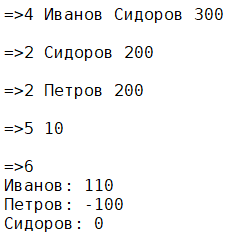
* 1 name sum – зачислить сумму sum на счет клиента name. Если у клиента нет счета, то счет создается.
* 2 name sum – снять сумму sum со счета клиента name. Если у клиента нет счета, то счет создается.
* 3 name – узнать остаток средств на счету клиента name.
* 4 name1 name2 sum - перевести сумму sum со счета клиента name1 на счет клиента name2. Если у какого-либо клиента нет счета, то ему создается счет.
* 5 p – начислить всем клиентам, у которых открыты счета, p% от суммы счета. Проценты начисляются только клиентам с положительным остатком на счете. Если у клиента остаток отрицательный, то его счет не меняется. После начисления процентов сумма на счету остается целой, то есть начисляется только целое число денежных единиц. Дробная часть начисленных процентов отбрасывается.
* 6 – завершить работу программы. Вывести на экран информацию о всех счетах.

Если символ => использовать как приглашение для ввода в функции input(), то результат выполнения программы должен выглядеть примерно так:









1. Информация о продажах вводится в виде строки «Покупатель Товар Количество», где Покупатель — имя покупателя (строка без пробелов), Товар — название товара (строка без пробелов), Количество — количество приобретенных единиц товара (целое число). Конец ввода – пустая строка.

Задание:

* выведите на экран списки покупателей и товаров;
* для каждого покупателя подсчитайте количество приобретенных им единиц каждого вида товаров;
* для каждого товара получите информацию о том, кто и сколько его купил.

Результат выполнения программы может выглядеть примерно так (как и в предыдущем задании => – приглашение для ввода):

