Словарь (dict)

Словари в Python – неупорядоченный набор объектов с доступом по ключу. Еще одно название словаря – **ассоциативный массив**. Словарь предназначен для работы с данными в формате «ключ: значение». Ключами словаря могут быть только неизменяемые объекты: числа, строки, кортежи, frozenset, а значение может быть любым.

Если сравнить словарь со списком, то доступ к его элементам осуществляется по индексу, который представляет собой целое неотрицательное число. В словаре аналогом индекса является ключи, их значения мы задаем сами.

Когда нужно использовать словарь

- 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.
- 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

`d = dict()` `d = {}`	*Создание пустого словаря*
`d1 = {1:'one', 2:'two', 3:'three'}` `d2 = {'one':1, 'two':2, 'three':3}`	*Создание словаря перечислением его элементов. Первый элемент каждой пары ключ, второй значение.*

`d[1] = 'on'`
`d[2] = 'two'`
`d[1] = 'one'`
`d = {1:'one', 2:'two'}`
	Создание слова путем явного объявления пар ключ-значение. В этом случае если в словаре не было такого ключа, он создается, если был, то значение меняется на новое.

`a = [(1, 10), (2, 20)]`
`d = dict(a)`
`d = {1: 10, 2: 20}`
	Создание словаря путем преобразования списка пар.

`a = [1, 2, 3]`
`b = ['one', 'two', 'three']`
`d = dict(zip(a, b))`
`d = {1:'one', 2:'two', 3:'three'}`
	Создание словаря путем склеивания попарно элементов 2-х списков. В этом примере используется функция zip, которая принимает на вход два списка и выдает новый итерируемый объект из пар элементов.

`key = [1, 2, 3]`
`d = dict.fromkeys(key, 0)`
`d = {1: 0, 2: 0, 3: 0}`
	Создание словаря по списку ключей. Для всех ключей значение устанавливается одинаковое. В данном случае 0.

Работа с элементами словаря

Операция	*Сложность*	*Описание*
`len(d)`	`O(1)`	*Определение количества элементов словаря*
`key in d`	`O(1)`	*Проверка наличия в словаре ключа key*
`d[key] = value`	`O(1)`	*Добавление пары* `key:'value` *в словарь*
`d.update(d2)`	`O(len(d2))`	*Добавление в текущий словарь d элементов словаря d2. Если ключи совпадают, то произойдёт замена значений.*
`x = d[key]`	`O(1)`	*Возвращает значение по ключу key. Если ключа нет, то выдает ошибку.*
`del d[key]`	`O(1)`	*Удаление из словаря пары с ключом key. Если ключа нет, то выдает ошибку.*
`d.pop(key, None)`	`O(1)`	*Безопасное удаление элемента из словаря. Возвращает значение по ключу key и удаляет элемент из словаря.*
`x = d.get(key, value)`	`O(1)`	*Возвращает значение по ключу, если ключ в словаре отсутствует, то возвращает значение переменной value. Этот метод позволяет избавиться от проверки условия наличия ключа key в словаре.*

`d.keys()`
`d.values()`
`d.items()`
	`O(len(d))`	*Возвращает итерируемые объекты из ключей словаря, значений словаря, пар ключ-значение.*

`for k in d:`
`  print(k)`
`for k in d.keys():`
`  print(k)`
	`O(len(d))`	*Перебор ключей словаря.*

`for v in d.values():` `  print(v)`	`O(len(d))`	*Перебор значений словаря.*

`for k, v in d.items():`
`  print(k, v)`
`for k in d:`
`  print(k, d[k])`
	`O(len(d))`	*Перебор пар ключ-значение.*

Примеры задач

Задача №1. На вход программе подается текст из n строк. Посчитайте частотность слов текста. Выведите слова в порядке убывания количества их встречаемости. Если слова имеют одинаковую частоту, то выведите из в алфавитном порядке

Входные данные	Выходные данные
2 AA aa BB bb BB aa BB BB	BB 4 aa 2 AA 1 bb 1

Решение.

Для решения этой задачи будем поддерживать структуру словарь, в которую будем заносить каждое новое слово в ключ, а в значении хранить сколько раз встретилось это слово.

```
n = int(input())
d = dict()
for i in range(n):
    st = input().split()
    for x in st:
        d[x] = d.get(x, 0) + 1

a = sorted(d.items(), key = lambda x: (-x[1], x[0]))
for k, v in a:
    print(k, v)
```

Словарь не поддерживает сортировку элементов, поэтому для правильного вывода результатов содержимое словаря нужно преобразовать в двумерный список, и отсортировать его по возрастанию второго параметра (значения в словаре) и после этого по возрастанию первого параметра (ключа словаря).

Используемые ресурсы:

1. Гутман Г.Н. Языки программирования: Python 3.1, учебное пособие.- Самара, Самарский государственный технический университет, 2011
2. Кириенко Д.П., Cbook
3. Саммерфилд М. Программирование на Python 3. Подробное руководство,
4. Любанович Билл, Простой Python. Современный стиль программирования. - СПб.: Питер, 2016.
5. Лутс М. Python, 4-е издание. – Пер. с англ. – Спб.: Символ-Плюс, 2010