

# Справочник по языку Python с примерами

2012-2018

# Оглавление

<b>0</b>	<b>Об этом справочнике</b>	<b>3</b>
<b>1</b>	<b>Обзор основных свойств языка</b>	<b>4</b>
1.1	Какие бывают языки	4
1.2	Значения в языке Python	5
1.3	Идентификаторы	6
1.4	Иерархия типов в Питоне	6
1.5	Операции и выражения	6
1.6	Приведение числовых типов	8
<b>2</b>	<b>Основные операторы</b>	<b>9</b>
2.1	Оператор присваивания	9
2.2	Функция <code>print</code>	10
2.3	Форматированный вывод	10
2.4	Функция ввода: <code>input()</code>	13
2.5	Условный оператор	13
2.6	Оператор цикла <code>while</code>	15
2.7	Операторы <code>break</code> и <code>continue</code>	16
2.8	Комментарии	17
<b>3</b>	<b>Строки, оператор цикла <code>for</code></b>	<b>18</b>
3.1	Строковые значения	18
3.2	Операции со строками	18
3.3	Оператор цикла <code>for</code>	21
<b>4</b>	<b>Списки, функция <code>range</code>, оператор цикла <code>for</code> (продолжение)</b>	<b>22</b>
4.1	Значения списков	22
4.2	Операции со списками	22
4.3	Присваивание и копирование списков	24
4.4	Функция <code>range</code>	25
4.5	Оператор цикла <code>for</code>	25
4.6	Функция <code>random</code>	27
4.7	Операции с вложенными списками	27
<b>5</b>	<b>Кортежи</b>	<b>30</b>
5.1	Свойства кортежей	30
5.2	Значения кортежей	30
5.3	Операции с кортежами	31
5.4	Применение кортежей	31
<b>6</b>	<b>Словари</b>	<b>32</b>
6.1	Создание словаря	32
6.2	Получение значений словаря, изменение словаря	32
6.3	Слияние словарей	33
<b>7</b>	<b>Множества</b>	<b>35</b>
7.1	Создание множества и логические операции с ними	35
7.1.1	Операции над множествами, не приводящие к их изменению	36
7.1.2	Изменение множеств	36

<b>8</b>	<b>Встроенные функции и методы</b>	<b>38</b>
8.1	Числовые функции . . . . .	38
8.2	Обработка последовательностей . . . . .	39
8.2.1	Общие функции . . . . .	39
8.2.2	Методы работы со списками . . . . .	39
8.2.3	Методы работы со строками . . . . .	42
<b>9</b>	<b>Работа с файлами</b>	<b>45</b>
9.1	Режимы работы с файлами. . . . .	45
9.2	Чтение из файла . . . . .	45
9.3	Запись в файл . . . . .	47
<b>10</b>	<b>Динамическая типизация</b>	<b>48</b>
10.1	Переменные, объекты и ссылки . . . . .	48
10.2	Автоматическая сборка мусора . . . . .	48
10.3	Разделяемые ссылки, изменяемые и неизменяемые типы . . . . .	49
<b>11</b>	<b>Функции, параметры, вызовы функций.</b>	<b>51</b>
11.1	Определение функции . . . . .	51
11.2	Вызов функции, значение None . . . . .	52
11.3	Запуск программы, содержащей функции . . . . .	54
11.4	Блоки, области видимости имён, локальные и глобальные переменные . . . . .	55
11.5	Передача переменных изменяемых и неизменяемых типов в качестве параметров функций . . . . .	57
11.6	Рекурсивные функции . . . . .	59

# Глава 0

## Зачем это было написано

Этот текст задумывался как краткий справочник по языку Python, снабжённый большим количеством примеров. Несмотря на явное указание того, что интерпретатор Python выдаёт в качестве ответа, имеет смысл выполнять указанные фрагменты кода в интерпретаторе. Во-первых, вы попытаетесь проверить собственные гипотезы, которые не нашли отражения в тексте, во-вторых — отыщите опечатки.

Этот справочник не предназначен для знакомства с программированием "с нуля". Скорее может помочь записать на языке Python уже понятные конструкции и познакомить с основными отличиями от Паскаля или Си.

Обо всех опечатках, ошибках, неудачных примерах и нелогичной последовательности изложения сообщайте по адресу: [gusarere@gmail.com](mailto:gusarere@gmail.com)

# Глава 1

## Обзор основных свойств языка. Значения, типы, выражения. Приведение ТИПОВ.

### 1.1 Какие бывают языки

- Высокоуровневые и низкоуровневые  
Текст программы на языке Питон не похож на машинный код, программа не оперирует непосредственно содержимым регистров процессора. Его служебные слова — это слова английского языка; язык, с помощью которого записываются выражения почти полностью совпадает с принятым в математике алгебраическим языком.
- Общего назначения и специализированные  
На Питоне можно написать графический редактор, вебсайт, программу, обрабатывающую результаты опытов, игру в крестики-нолики и многое другое.
- Интерпретируемые и компилируемые  
Программу на Питоне выполняет программа-интерпретатор. Почти для каждой операционной системы и архитектуры процессора существует свой интерпретатор, так что текст программы не требуется модифицировать. Правда, из-за необходимости обработки интерпретатором программа на Питоне работает медленнее, чем аналогичная программа на компилируемом языке (C++, Паскаль).

Определение языка включает в себя синтаксические и семантические правила. Первые описывают структуру, вторые — смысл.

- Синтаксис — определяет разрешённые конструкции языка  
`1x = 6` # имя переменной не может начинаться с цифры  
Аткрой окно! # нет таких слов в русском языке

Синтаксические ошибки всегда обнаруживаются до выполнения программы в компилируемых языках. В интерпретируемых, как правило, на этапе выполнения, но, вообще говоря, бывают и предварительные проверки до выполнения программы интерпретатором. Эти ошибки — самые простые.

- Статическая семантика — связывает синтаксически верные конструкции общим смыслом

```
x = 6 + 'asd' # операция не определена для int и string
print(t) # имя t не определено (не имеет значения)
```

Такие ошибки обнаруживаются на этапе выполнения программы. Пример программы, которая при разных входных данных либо корректно завершается, либо выдаёт сообщение об ошибке:

↓

```
x = int(input())
if x > 1:
    print(5)
else:
    print('asd' // x) # не определена операция деления строки на число
```

- Самые сложные ошибки возникают, когда программа написана верно в предыдущих двух смыслах, но на некоторых входных данных не заканчивает работу или выдаёт неправильный ответ.

↓

```
def power(a, n):
    res = 1
    i = 1
    while i < n:
        res = res * a
    return res
```

В этом примере цикл `while` никогда не закончит работу, если переданное значение параметра `n` больше 1.

Для естественного языка примером может послужить сочинение на заданную тему, заслужившее пометку «Тема не раскрыта».

*Условия реализации основной образовательной программы среднего (полного) общего образования должны обеспечивать для участников образовательного процесса возможность... формирования у обучающихся..., готовности к защите Отечества, службе в Вооружённых Силах Российской Федерации.*

(Выдержка из Федерального Государственного Образовательного Стандарта Общего Образования)

## 1.2 Значения в языке Python

- Числа
  - Для хранения целых чисел Python в отличие от большинства языков (C++, Паскаль) использует всю доступную память. Фактически, работа с длинными целыми числами реализована в стандарте языка.
  - Вещественные числа реализованы на основе чисел с плавающей точкой двойной точности — `double` (64 бита). 1 бит на знак, 11 бит на показатель экспоненты и 52 бита на значащую часть (мантиссу).  
Примеры: `3.0`, `-123.345`, `.76543`, `23.490e23`.
- Логические: `True`, `False`. Логический тип на самом деле является лишь подтипом целого, значение `False` соответствует нулю, `True` — любому ненулевому целому числу.
- Упорядоченные последовательности
  - **строки**: последовательность литералов (символов). Строковые значения должны быть заключены в одинарные и двойные кавычки. Примеры: `'a'`, `'abc'`, `'234g 3654__'`, `"don't"`.
  - **списки**: последовательность произвольных элементов, разделяемых запятыми и взятая в квадратные скобки. Пустой список — `[]`. Примеры: `[1, 2, 3]`, `['Name', 'Surname', 5]`.
  - **кортежи**: последовательность произвольных элементов, разделяемых запятыми, которая может быть взята в круглые скобки. Пустой кортеж обязательно должен быть взят в скобки: `()`, кортеж из одного элемента обязательно должен содержать запятую после единственного элемента: `(4,)`.  
Примеры: `(2, 3)`, `('abc', 345)`





























































































































