

Язык программирования PYTHON

5



- высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение кроссплатформенности написанных на нём программ.

Разработчик языка Python – голландский программист Гвидо ван Россум



Язык программирования PYTHON

6

Установка

1. Установить Python (<https://www.python.org/>). В OS X и Unix уже установлен.
2. Только для Windows. При установке выбрать опцию с заданием переменных окружения.

Среда разработки

1. [Jupyter Notebook](#) из пакета Anaconda (<https://www.anaconda.com/products/individual>).
2. Текстовые редакторы: VS Code, Sublime, Atom, Notepad++ и др.
3. IDE: PyCharm, Visual Studio, Spyder и др.

Почему PYTHON?

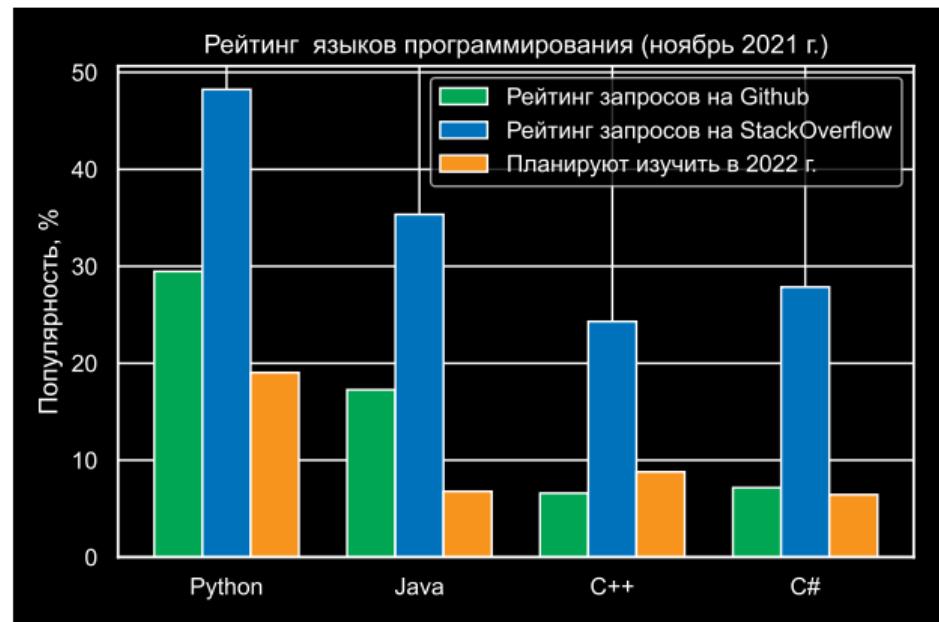
7

Кто использует Python?

- Google
- YouTube
- Dropbox
- BitTorrent
- iRobot
- Netflix и Yelp
- Intel, Cisco
- NASA

Сильные стороны

- Качество программного кода
- Продуктивность труда
- Кроссплатформенность
- Библиотеки



Простейшая программа на Python

11

Простейшей программой на языке Python является пустой файл с расширением .py, однако рассмотрим традиционную реализацию программы, выводящей на экран сообщение «Привет Мир!»:

```
1 | print("Привет Мир!")
2 |
```

Для вывода на экран сообщения «Привет Мир!» достаточно обратиться к стандартной функции `print()` и передать ей в качестве параметра строку "Привет Мир!".

- Переменные – это частный случай идентификаторов.
- Идентификаторы – это имена, присвоенные чему-то для его обозначения.

При выборе имен идентификаторов необходимо соблюдать следующие правила:

1. Первым символом идентификатора должна быть буква из алфавита (символ ASCII в верхнем или нижнем регистре, или символ Unicode), а также символ подчеркивания («_»);
2. Остальная часть идентификатора может состоять из букв (символы ASCII в верхнем или нижнем регистре, а также символы Unicode), знаков подчеркивания «_» или цифр (0 – 9);
3. Имена идентификаторов **чувствительны** к регистру!

Допустимые имена: i, __my_name, name_23, a1b2_c3

Недопустимые имена: 2things, my-name, >a1b2_c3

Определяемые Вами имена не могут совпадать с ключевыми словами в Python. К примеру, если Вы попытаетесь выбрать для переменной имя `class`, то Python сообщит об ошибке.

Ключевые слова в Python

<code>False</code>	<code>class</code>	<code>finally</code>	<code>is</code>	<code>return</code>
<code>None</code>	<code>continue</code>	<code>for</code>	<code>lambda</code>	<code>try</code>
<code>True</code>	<code>def</code>	<code>from</code>	<code>nonlocal</code>	<code>while</code>
<code>and</code>	<code>del</code>	<code>global</code>	<code>not</code>	<code>with</code>
<code>as</code>	<code>elif</code>	<code>if</code>	<code>or</code>	<code>yield</code>
<code>assert</code>	<code>else</code>	<code>import</code>	<code>pass</code>	<code>break</code>
<code>except</code>	<code>in</code>	<code>raise</code>		

Многократное присваивание переменной

16

- Переменные никогда не располагают какой-либо информацией о типах или связанных с ними ограничениях.
- Понятие типа связано с объектами, а не именами.
- Переменные являются обобщенными по своей сути и просто ссылаются на определенные объекты в конкретный момент времени.

```
1 | x = 1
2 | x = "Hello"
3 | x = 1.2
4 |
```

- Если переменная используется в каком-то выражении, то при его вычислении она заменяется объектом, ссылкой на который она является.
- Все переменные должны быть присвоены чему либо до того, как их можно будет использовать; использование неприсвоенных переменных приведет к ошибке!

Сылочная модель данных

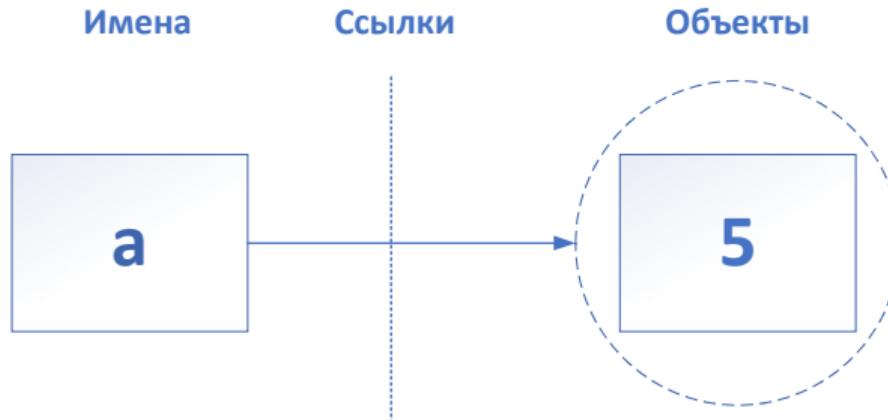
17

В момент, когда происходит присваивание значений переменной:

`a = 5`

фактически происходит выполнение трех шагов:

1. Создание объекта для представления значения 5.
2. Создание переменной a.
3. Связывание переменной a с новым объектом 5.



Сборка мусора

18

В Python при присваивании имени нового объекта происходит освобождение области памяти, занимаемой предыдущим объектом, если на него не ссылается другое имя или объект. Данная операция известна как *сборка мусора*:

```
1 >>> x = 32
2 >>> x = "hello"    #Освободить память, занимаемую 32
3                                #(если нет других ссылок)
4 >>> x = 3.1245      #Освободить память, занимаемую "hello"
5 >>> x = [0, 1, 2, 3] #Освободить память, занимаемую 3.1245
6
```

Неоспоримое преимущество сборки мусора заключается в том, что она позволяет свободно использовать объекты без необходимости выделения и освобождения памяти вручную. Python будет автоматически выделять или очищать пространство для хранения объектов при выполнении программы.

Разделяемые ссылки

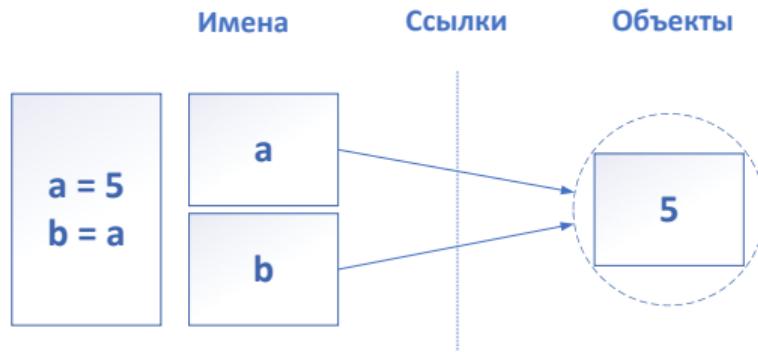
19

Рассмотрим создание разделяемых ссылок:

```
1 |     >>> a = 5  
2 |     >>> b = a  
3 |
```

Такой контекст в Python со множеством переменных, ссылающихся на один и тот же объект, называется *разделяемой ссылкой* (иногда говорят *разделяемый объект*).

Стоит обратить внимание, что при этом имена *a* и *b* не связаны друг с другом напрямую.



Разделяемые ссылки

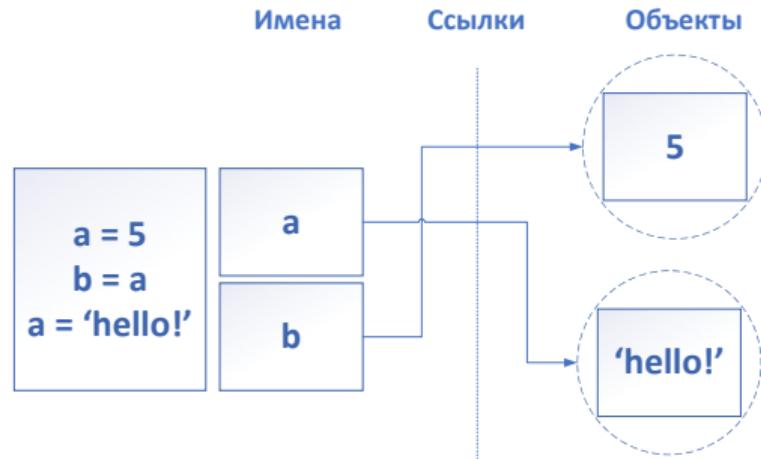
20

Добавим еще один оператор:

```
1 >>> a = 5  
2 >>> b = a  
3 >>> a = "hello!"  
4
```

Последний оператор просто создает новый объект для представления строкового значения "hello!" и переопределяет ссылку a на этот новый объект. Но все это не отражается на значении b, она по-

прежнему продолжает ссылаться на первоначальный объект, целое число 5.



Комментарии – это то, что пишется после # и представляет интерес лишь как заметка для читающего программу. Текст программы говорит о том, **КАК**, а комментарии должны объяснять, **ПОЧЕМУ**.

1 | print("Привет Мир!") #print - это функция
2 |

- Комментарии могут встречаться в тексте программы как отдельные строки или могут быть размещены справа от операторов в строке.
- Текст после символов # пропускается интерпретатором как примечание, предназначенное для человека, читающего данную программу.
- При копировании кода Вы можете игнорировать комментарии, т.к. они носят исключительно информативный характер.
- Грамотное использование комментариев может в значительной степени повысить читаемость, облегчить понимание и, как следствие, улучшить поддерживаемость Вашего программного кода.

Классификация объектов

24

Основные типы объектов в Python:

Тип объекта	Категория	Изменяемый?
Числа (все)	Числовые	Нет
Строки	Последовательности	Нет
Списки	Последовательности	Да
Словари	Отображения	Да
Кортежи	Последовательности	Нет
Файлы	Расширения	-
Множества	Множества	Да

Числа в Python могут быть трех типов:

1. Целые числа (`int`);
2. Числа с плавающей точкой (`float`);
3. Комплексные числа (`complex`).

Литерал	Расшифровка
1234, -24, 0, 9999999999999999	Целые числа (неограниченный размер)
1.23, 1., 3.17e-10, 4E210, 4.0e+210	Числа с плавающей точкой
3+4j, 3.0+4.0j, 3J	Литералы комплексных чисел

Целые числа и числа с плавающей точкой

27

- Целые числа записываются в виде строк десятичных цифр.
- Числа с плавающей точкой имеют десятичную точку и/или необязательный показатель степени со знаком, вводимый посредством нотации e или E.
- Целые числа можно создавать, используя встроенную функцию `int`, а числа с плавающей точкой – при помощи встроенной функции `float`:

```
1  >>> x = 2.5
2  >>> y = 1
3  >>> z = 2.
4  >>> type(x) #Число с плавающей точкой
5  <class 'float'>
6  >>> type(y) #Целое число
7  <class 'int'>
8  >>> type(z) #Тоже число с плавающей точкой
9  <class 'float'>
10 >>> int(x) #Получение целого числа отбрасыванием дробной части
11 2
12 >>> float(y) #Приведение целого числа к типу float
13 1.0
```

Комплексные числа

28

Комплексные числа в Python записываются как

действительная_часть + мнимая_часть

где мнимая_часть заканчивается символом *j* или *J*.

Комплексные числа можно создать при помощи встроенной функции `complex(real, imag)`:

```
1 | >>> x = 2.5
2 | >>> y = 0.85
3 | >>> complex(x, y)
4 | (2.5+0.85j)
```

Арифметические операции с числами

30

Основные арифметические операции с числами:

Приоритет	Операция	Описание
1	$x + y$	Сложение
2	$x - y$	Вычитание
3	$x * y$	Умножение
4	$x \% y$	Остаток от деления (деление по модулю)
5	x / y , $x // y$	Настоящее деление, деление нацело (с округлением в меньшую сторону)
6	$-x$, $+x$	Противоположность, идентичность
7	$x ** y$	Возведение в степень

- Если операция используется с двумя operandами, она называется **бинарной**.
- Существуют также операции с одним operandом, называемые **унарными**.

Подобно другим языкам программирования, сложные выражения в Python записываются путем объединения арифметических операций. Например, определение суммы двух произведений можно представить в виде комбинации переменных и операций:

$$a \cdot b + c \cdot d$$

Значения подобных и еще более сложных выражений интерпретатор Python будет вычислять в соответствии с *приоритетом операций*.

Круглые скобки в выражениях

32

Заключение подвыражений в круглые скобки переопределяет приоритеты операций Python; выражения в круглых скобках во всех случаях вычисляются первыми и затем их результаты используются в охватывающих выражениях.

$$\begin{aligned}(x + y) \cdot z \\ x + (y \cdot z)\end{aligned}$$

- В первом случае операция $+$ применяется к переменным x и y первой, потому что это действие помещено в круглые скобки.
- Во втором случае первой выполняется операция $*$, так же, как если бы круглые скобки отсутствовали.
- В большинстве случаев добавление скобок в объемные составные выражения является хорошим тоном, т.к. не только обеспечивает правильный порядок его вычисления, но и помогает в обеспечении лучшей читабельности.

Операции с числами разных типов

33

В Python допускается использовать в выражениях различные числовые типы. Например, можно сложить целое число и число с плавающей точкой:

$$30 + 3.14$$

Целые числа проще чисел с плавающей точкой, которые проще, чем комплексные числа. Таким образом, если операция выполняется над целым числом и числом с плавающей точкой, то целое число приводится к значению с плавающей точкой и результат вычисляется тоже как число с плавающей точкой:

```
1 >>> 30 + 3.14      #Преобразование целого числа в число с плавающей точкой  
2                                #результат вычисления тоже число с плавающей точкой  
3 33.14
```

Типы также можно преобразовать вручную, вызвав встроенные функции:

```
1 >>> int(30.145)    #Сокращение числа с плавающей точкой до целого  
2 30  
3 >>> float(6)       #Преобразование целого числа в число с плавающей точкой  
4 6.0
```

- Классическое (настоящее) деление – деление с сохранением остатка:

 x / y

- Деление с округлением в меньшую сторону – данная операция всегда усекает дробные остатки, округляя в меньшую сторону не зависимо от типов operandов. Тип результата зависит от типов operandов.

 $x // y$

```
1 >>> 100 / 40      #Сохраняет остаток
2 2.5
3 >>> 100 / 40.0
4 2.5
5 >>> 100 // 40     #Усекает остаток, возвращает целое число
6 2
7 >>> 100 // 40.0   #Округляет в меньшую сторону, возвращает число с
8 #плавающей точкой
9 2.0
```

Составные операторы присваивания

35

В Python доступен набор дополнительных форматов операторов присваивания.

Данный набор по большей части является всего лишь сокращением, совмещающим в себе бинарную операцию и присваивание.

Составные операторы присваивания

<code>x += y</code>	<code>x &= y</code>	<code>x -= y</code>	<code>x = y</code>
<code>x *= y</code>	<code>x ^= y</code>	<code>x /= y</code>	<code>x >>= y</code>
<code>x %= y</code>	<code>x <<= y</code>	<code>x **= y</code>	<code>x //= y</code>

Operator	Description	Syntax
&	Bitwise AND	<code>x & y</code>
	Bitwise OR	<code>x y</code>
~	Bitwise NOT	<code>~x</code>
^	Bitwise XOR	<code>x ^ y</code>
>>	Bitwise right shift	<code>x >> y</code>
<<	Bitwise left shift	<code>x << y</code>

К примеру, рассмотрим два способа добавления 10 к имени:

```
>>> x = 1
>>> x = x + 10 #Традиционное присваивание
>>> x
11
>>> x += 10      #Составное присваивание
>>> x
21
```

Другие встроенные операции для работы с числами

36

Python предлагает встроенные функции и стандартные модули для работы с числами. Например, встроенные функции `pow` и `abs` вычисляют степень и абсолютное значение, соответственно.

```
1  >>> import math
2  >>> math.pi, math.e                         #Общие константы
3  (3.141592653589793, 2.718281828459045)
4  >>> math.sin(2 * math.pi / 180)              #Синус, косинус, тангенс
5  0.03489949670250097
6  >>> math.sqrt(121), math.sqrt(3)             #Квадратный корень
7  (11.0, 1.7320508075688772)
8  >>> pow(3, 4), 3 ** 4, 3.0 ** 4.0          #Возведение в степень
9  (81, 81, 81.0)
10 >>> abs(-24.0), sum((1, 2, 3, 4, 5, ))   #Абсолютное значение,
11                               #суммирование
12 (24.0, 15)
13 >>> min(3, 1, 2, 4, 5), max(3, 1, 2, 4, 5) #Минимум, максимум
14 (1, 5)
```

Округление и усечение чисел с плавающей точкой

37

Существуют также несколько способов отбрасывания десятичных цифр из чисел с плавающей точкой:

```
1 >>> math.floor(2.512), math.floor(-2.512)      #Округление
2                                         #до меньшего целого
3 (2, -3)
4 >>> math.trunc(2.512), math.trunc(-2.512)    #Усечение (отбрасывание
5                                         #десятичных цифр)
6 (2, -2)
7 >>> int(2.512), int(-2.512)     #Преобразование в целое число
8 (2, -2)
9 >>> round(2.512), round(2.462), round(2.516, 2) #Округление
10 (3, 2, 2.52)
```

Распространенные математические функции модуля math

38

Функция	Описание
<code>math.ceil(x)</code>	округление до ближайшего большего числа
<code>math.floor(x)</code>	округление вниз
<code>math.trunc(x)</code>	усекает значение x до целого
<code>math.exp(x)</code>	e^x
<code>math.log(x, [base])</code>	логарифм x по основанию base, если base не указан, вычисляется натуральный логарифм
<code>math.acos(x)</code>	арккосинус x в радианах
<code>math.asin(x)</code>	арксинус x в радианах
<code>math.atan(x)</code>	арктангенс x в радианах
<code>math.cos(x)</code>	косинус x (x указывается в радианах)
<code>math.sin(x)</code>	синус x (x указывается в радианах)
<code>math.tan(x)</code>	тангенс x (x указывается в радианах)

Для ввода пользовательских данных в Python предусмотрена стандартная функция `input()`:

```
1 x = input("Введите x: ")  
2 y = input("Введите y: ")  
3  
4 print(x + y)
```

Введите x: 5

Введите y: 7

57

Функция `input()`: возвращает строку, поэтому при вычислении выражения $x + y$ в результате получается 57, т.к. происходит конкатенация двух строк.

При необходимости ввода числовых значений следует явно приводить результат функции `input()` к желаемому типу:

```
1 x = int(input("Введите x: "))
2 y = int(input("Введите y: "))
3
4 print(x + y)
```

Введите x: 5

Введите y: 7

12

Пример программирования арифметического выражения

Вычислите выражение при $x = 1$:

$$(\arctan(\sqrt{x} - 2.7 \times 10^{-3}))^2 + \frac{e^{-x}}{\cos(x)}$$

```
1 import math
2
3
4 x = 1
5
6 y = math.atan(x ** 0.5 - 2.7e-3) ** 2 \
7     + math.exp(-x) / math.cos(x)
8
9 print(y) #1.2956057140344002
```