Язык программирования Python

Основы языка Python

Линейные программы

С. В. Чайченков, МБОУ Грушевская СОШ

Язык Python

Языки программирования – это формальные языки, предназначенные для записи алгоритмов, исполнителем которых является компьютер. Алгоритмы, записанные на этих языках, называют **программами**.

Гвидо ван Россум

Одним из самых популярных современных языков программирования является Python (произносится «па́йтон» или просто «питон»). Его разработал в 1991 году нидерландский программист Гвидо ван Россум. Этот язык непрерывно совершенствуется, сейчас используется версия Python 3.

Язык Python применяется для обработки различных данных, математических вычислений, создания изображений, работы с базами данных, разработки веб-сайтов.

Thonny

Относительно молодая IDE, которая подходит для обучения.
«+»

- не требует дополнительного скачивания Python;
- упрощённая установка дополнительных опций и плагинов;
- встроенный ассистент, подсказки по исправлению ошибок;
- небольшой вес.

- https://www.python.org/downloads/

Общие сведения о языке программирования Python

Алфавит языка Python (набор допустимых символов) состоит из букв латинского алфавита (причём *заглавные и строчные буквы различаются*), цифр и специальных знаков (знаков препинания, арифметических и других). Русские буквы могут использоваться только при выводе текста на экран и в комментариях к программе.

Служебные слова – цепочки символов, имеющие фиксированное смысловое значение.

Величины в программе представлены в виде констант и переменных.

Константы – величины, не изменяющие своего значения при выполнении программы.

Переменные – величины, которые могут изменять свое значение при выполнении программы. Каждая переменная имеет имя, тип и значение.

Имя переменной (идентификатор) – любая отличная от служебных слов последовательность латинских букв, цифр и символа подчеркивания "_", не может начинаться с цифры.

N, N1, massa, massa_tela – правильно;
1N, масса, massa tela – неправильно.

Общие сведения о языке программирования Python

Значения переменных хранятся в ячейках оперативной памяти.

Тип переменной определяет способ хранения данных в памяти компьютера и допустимые операции над ними.

Основные типы данных в языке Python

Название	Обозначение	Допустимые значения
Целочисленный	`int` *(«integer»)*	Сколь угодно большие целые числа, размер ограничен оперативной памятью
Вещественный	`float` *(«floating point»)*	Любые числа с дробной частью (с плавающей точкой)
Строковый	`str` *(«string»)*	Произвольная последовательность символов из таблицы Unicode
Логический	`bool` *(«boolean»)*	False («Ложь») или True («Истина»)

Целая часть числа от дробной отделяется точкой.

Строковое значение заключается в двойные или одинарные кавычки.

Тип переменной определяется автоматически в момент присваивания ей значения и может изменяться по ходу выполнения программы.

Выражения и операции

Выражение – это конструкция, возвращающая значение некоторого типа.

Простыми выражениями являются переменные и константы.

Сложные выражения строятся из простых с помощью операций, функций и скобок. Данные, к которым применяются операции, называются **операндами**.

Используется линейная форма записи выражений (в одну строку).

Арифметические операции

Операция	Обозначение	Пример
Сложение	+	`3 + 4 = 7`
Вычитание	-	`7 - 2 = 5`
Умножение	*	`2 * 2 = 4`
Деление	/	`8 / 2 = 4`
Целочисленное деление	//	`9 // 2 = 4`
Остаток от деления	%	`9 % 2 = 1`
Возведение в степень	**	`2 ** 3 = 8`

Выражения и операции

Логические выражения могут содержать величины или выражения, которые сравниваются между собой с помощью операций сравнения.

Логическое выражение может принимать лишь два значения: «истина» или «ложь».

Операции сравнения

Операция	Символы	Пример
равно	==	`x == 0`
не равно	!=	`x != 0`
больше	>	`x > 0`
меньше	<	`x < 0`
больше или равно	>=	`x >= 0`
меньше или равно	<=	`x <= 0`

Выражения и операции

Приоритет выполнения операций:

1) операции в скобках;
2) возведение в степень;
3) умножение и деление (в том числе // и %);
4) сложение и вычитание.

Операции одинакового приоритета выполняются в порядке записи слева направо.

Если выражение слишком длинное и не помещается в одной строке, необходимо заключить всё выражение в скобки (перенос внутри скобок разрешён).

Например:

$$\frac{(a+b)h}{2}$$

1 2 3

```
(a+b)*h/2
```

$$v+\frac{at^2}{2}$$

4 2 1 3

```
v+a*t**2/2
```

Оператор (команда) присваивания

Оператор присваивания записывает в переменную, имя которой находится слева от знака «=» значение выражения, находящегося справа. Старое значение переменной при этом стирается.

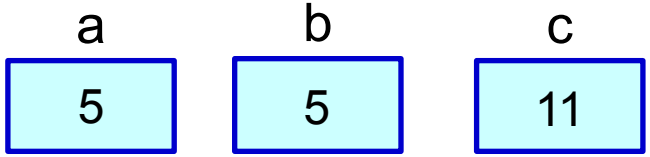
Общий вид оператора:

```
<имя переменной> = <выражение>;
```

Например:

```
a = 5
b = a
c = a+b
c = c+1
```

В памяти:



В языке Python допускается множественное присваивание:

Запись оператора:	*Равносильная запись:*
`a, b = 0, 1`	`a = 0` `b = 1`
`a = b = 0`	`a = 0` `b = 0`

Допускается запись нескольких операторов в одной строке через символ «;».

Оператор вывода

Вывод данных из оперативной памяти на экран осуществляется с помощью оператора (функции) вывода **print** («печатать»):

```
print(<выражение1>, <выражение2>, ..., <выражениеN>)
```

- На экран выводятся значения переменных и выражений, строковые значения выводится на экран без кавычек.
- Выводимые значения разделяются пробелом (по умолчанию).
- После выполнения оператора происходит автоматический переход на новую строку.

Например:

```
print ("Масса равна", m, "кг");
```

Для m=15 на экране появится:

```
Масса равна 15 кг
```

Здесь и далее символом □ обозначен пробел.

Оператор вывода

- Вместо пробела можно использовать другие символы в качестве разделителя, указав их после слова **sep** («separator»).
- Чтобы убрать переход на новую строку после выполнения оператора, используется параметр **end**.

Нужный вариант вывода	Оператор	На экране
По умолчанию	`print (1, 20, 300)`	1 20 300
Без разделителя	`print (1, 20, 300, sep="")`	120300
Через запятую и пробел	`print (1, 20, 300, sep=", ")`	1, 20, 300
Каждое с новой строки	`print (1, 20, 300, sep="\n")`	1 20 300
Без перехода на новую строку	`print (1, end="")` `print (20)`	120

Оператор вывода

Форматный вывод с помощью символьной строки позволяет задать количество позиций на экране, занимаемых выводимой величиной.

- В фигурных скобках задается формат вывода очередного элемента.
- Для целых чисел указывается количество позиций, отводимых на число.
- Если цифр в числе меньше, слева от числа выводятся пробелы.
- Для вещественного числа указывается общее количество позиций, через точку количество позиций в дробной части и буквы: **`d`** (целое число), **`f`** (вещественное) или **`e`** (экспоненциальный формат).

Фрагмент программы	На экране
`print ("{:3}{:3}{:3}".format(13, 7, 22))`	□13□□7□22
`a = 7` `print ("{:4d}{:4d}".format(a, a*a))`	□□□7□□49
`a = 1/3; b = 1/9` `print ("{:7.3f}{:7.4f}".format(a, b))`	□□0.333□0.1111
`a = 1/3` `print ("{:10.3e}".format(a))`	□3.333e-01

Оператор ввода

Для ввода значений переменных с клавиатуры в процессе выполнения программы используется оператор (функция) ввода **input** («ввод»):

```
<имя_переменной> = input()
```

При выполнении оператора:

- компьютер переходит в режим ожидания данных;
- пользователь вводит с клавиатуры данные в виде строки символов;
- для завершения ввода пользователь нажимает клавишу Enter;
- введенная строка записывается в указанную переменную.

Если вводится не строка, а число, необходимо выполнить преобразование типов с помощью функций **int** (для целых) и **float** (для вещественных).

Например:

```
print("Введите слово и два числа:")
x = input()
y = int(input())
z = float(input())
print(x, y, z)
```

На экране:

```
Введите слово и два числа:
ноль
1
2
ноль 1 2.0
```

Оператор ввода

Можно в скобках указать текст подсказки для пользователя.

Например:

```
x = input("Введите слово: ")
y = int(input("Введите целое число: "))
z = float(input("Введите вещественное число: "))
print (x, y, z)
```

На экране:

```
Введите слово: ноль
Введите целое число: 1
Введите вещественное число: 2
ноль 1 2.0
```

Оператор ввода

Можно в одной строке ввести несколько значений через пробел. Для этого используется функция **split** («расщепить»). Затем данные необходимо преобразовать к нужному типу по отдельности.

Например:

```
a, b, c = input("Введите a,b,c через пробел: ").split()
a, b, c = int(a), int(b), int(c)
print (a, b, c)
```

На экране:

```
Введите a,b,c через пробел: 1 2 3
1 2 3
```

Оператор комментария

Используется для включения в программу любых пояснений, предназначенных человеку.

Комментариями считается любой текст после символа **#** до конца строки. При выполнении программы комментарии игнорируются.

Пример программы:

```
# Длина окружности и площадь круга
r = float(input("Введите радиус: "))
c = 2*3.14*r    # длина окружности
s = 3.14*r**2      # площадь круга
print ("c=", "{:7.3f}".format (c))
print ("s=", "{:7.3f}".format (s))
```

На экране:

```
Введите радиус: 10
c=    62.80
s=   314.00
```

Стандартные функции

Функции имеют определенное *имя* и один или несколько *аргументов* в скобках. Функция возвращает свое значение в то место программы, из которого она вызывается.

Некоторые стандартные функции, встроенные в ядро языка Python

Функция	Назначение	Тип аргумента	Тип результата
`abs(x)`	абсолютная величина (модуль числа x)	`int, float`	как у аргумента
`int(x)`	преобразование вещественного числа к целому значению (отбрасывание дробной части)	`float`	`int`
`round(x)`	округление вещественного числа до заданного количества знаков после точки (по умолчанию – до ближайшего целого)	`float`	`int, float`

Стандартные функции

Большинство стандартных функций языка Python разбиты на группы по назначению, каждая группа записана в отдельном файле, который называется **модулем**. Подключение модуля осуществляется командой **`import`**.

Например:

```
# подключаем все функции из модуля math
from math import *
```

Стандартные функции модуля `math`

Функция	Назначение	Тип аргумента	Тип результата
`sqrt(x)`	квадратный корень из x	`int, float`	`float`
`sin(x)`	синус угла x в радианах	`int, float`	`float`
`cos(x)`	косинус угла x в радианах	`int, float`	`float`

Стандартные функции

После подключения модуля к его функциям можно обращаться так же, как к встроенным. Например:

$$\frac{-b+\sqrt{b^2-4ac}}{2a}$$

6 5 1 4 2 3 8 7

```
(-b + sqrt(b**2 - 4*a*c))/(2*a)
```

Можно подключать не все функции, а только необходимую. *Например:*

```
# подключаем функцию randint() из модуля random
from random import randint
```

Стандартные функции модуля `random`

Функция	Назначение	Тип аргумента	Тип результата
`random()`	случайное число из полуинтервала [0, 1)	–	`float`
`randint(a, b)`	случайное число из отрезка [a, b]	`int`	`int`

Стандартные функции

Пример со стандартными функциями:

```
# Стандартные функции
a = 3.56
print (a)
print (round(a))
print (round(a, 1))
print (int(a))
from math import *
b = 16
print (sqrt(b))
from random import randint
x = randint(1, 10)
y = randint(1, 10)
z = randint(1, 10)
print (x, y, z)
```

На экране:

```
3.56
4
3.6
3
4.0
2 10 8
```

Задача 1

Составить программу, меняющую местами значения двух переменных

```
# Обмен значений переменных
# Классическое решение
a = int(input("a= "))
b = int(input("b= "))
t = a    # временная переменная
a = b
b = t
print ("a=", a)
print ("b=", b)
```

```
# Обмен значений переменных
# Возможности языка Python
a = int(input("a= "))
b = int(input("b= "))
a, b = b, a
print ("a=", a)
print ("b=", b)
```

На экране:

```
a= 2
b= 5
a= 5
b= 2
```

Задача 2

Составить программу для вычисления площади треугольника по известным длинам его сторон.

Формула Герона: $S = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$

```
# Площадь треугольника
print ("Введите длины сторон треугольника: ")
a = float(input("a="))
b = float(input("b="))
c = float(input("c="))
p = (a+b+c)/2                       # полупериметр
from math import sqrt               # подключаем модуль math
s = sqrt(p*(p-a)*(p-b)*(p-c))       # формула Герона
print ("Площадь треугольника", "{:7.2f}".format(s))
```

На экране:

```
Введите длины сторон треугольника:
a=5
b=6
c=7
Площадь треугольника    14.70
```

Задача 3

Составить программу, вычисляющую сумму цифр введенного с клавиатуры целого трёхзначного числа

```
# Сумма цифр трехзначного числа
# Трёхзначное число x = a•100 + b•10 + c
# где a, b, c - цифры этого числа
x = int(input("Введите трехзначное число: "))
a = x // 100          # сотни
b = x % 100 // 10       # десятки
c = x % 10            # единицы
s = a + b + c
print ("Сумма цифр равна", s)
```

На экране:

```
Введите трехзначное число: 345
Сумма цифр равна 12
```

Используемые материалы:

- *Босова Л.Л. Информатика. 8-9 классы. Начала программирования на языке Python. Дополнительные главы к учебникам – М. : БИНОМ. Лаборатория знаний, 2020.*
- *Поляков К.Ю. Информатика. 10 класс. Базовый и углубленный уровни : в 2ч. Ч. 2 – М. : БИНОМ. Лаборатория знаний, 2018.*