

Тема 2.

Основные элементы языка. Управляющие операторы языка. Структурированные типы данных. Символьные типы данных

<https://pythonworld.ru/samouchitel-python>

<https://younglinux.info/python/input>

Основные элементы языка

https://studref.com/516540/informatika/elementy_yazykov_programmirovaniya

Элементы ЯП могут рассматриваться на следующих уровнях:

- *алфавит* — совокупность символов, отображаемых на устройствах печати и экранах и/или вводимых с клавиатуры терминала. Обычно это набор символов Latin-1 с исключением управляющих символов. Иногда в это множество включаются неотображаемые символы с указанием правил их записи (комбинирование в лексемы);
- *лексика* — совокупность правил образования цепочек символов (лексем), образующих идентификаторы (переменные и метки), операторы, операции и другие лексические компоненты языка. Сюда же включаются зарезервированные (ключевые) слова ЯП, предназначенные для обозначения операторов, встроенных функций и пр.

Иногда эквивалентные лексемы, в зависимости от ЯП, могут обозначаться как одним символом алфавита, так и несколькими. Например, операция присваивания значения в ЯП Basic обозначается как «=», а в языке Pascal — «:=». Операторные скобки в Си задаются символами «{» и «}», а в Pascal — **Begin** и **End**. Граница между лексикой и алфавитом, таким образом, является весьма условной, тем более что компилятор обычно на фазе лексического анализа заменяет распознанные ключевые слова внутренним кодом (например, **Begin** — 512, **End** — 513) и в дальнейшем рассматривает их как отдельные символы;

- *морфология* — совокупность правил и возможностей варьирования написания лексических единиц, в пределах которых они будут правильно распознаны компилятором. Например, для оператора перехода в некоторых реализациях Basic записи **go**, **goto**, **do to** эквивалентны, а в ЯП FoxPro наименования команд и ключевые слова могут быть сокращены вплоть до 4-х начальных букв (например, эквивалентны команды **report**, **repor**, **repo**);

- *синтаксис* — совокупность правил образования языковых конструкций или предложений ЯП — блоков, процедур, составных операторов, условных операторов, операторов цикла и пр. Особенностью синтаксиса является принцип вложенности (рекурсивность) правил построения конструкций. Это значит, что элемент синтаксиса языка в своем определении прямо или косвенно в одной из его частей содержит сам себя. Например, в определении оператора цикла телом цикла является оператор, частным случаем которого является все тот же оператор цикла;
- *семантика* — смысловое содержание конструкций, предложений языка. Семантический анализ — это проверка смысловой правильности конструкции. Например, если мы в выражении используем переменную, то она должна быть определена ранее по тексту программы, а из этого определения может быть получен ее тип. Исходя из типа переменной, можно говорить о допустимости операции с данной переменной. Семантические ошибки возникают при недопустимом использовании операций, массивов, функций, операторов и пр.
- *зарезервированные (служебные) слова*, имеющиеся в каждом ЯП, могут быть использованы только по своему специальному назначению (как имена функций, операторов, операций и пр.).

Операторы языка

<https://pythonru.com/osnovy/operatory-python>

Оператором можно считать символ, который выполняет операцию над одним или несколькими операндами.

Операндом выступает переменная или значение, над которыми проводится операция.

Операторы бывают 7 типов:

- Арифметические операторы
- Операторы сравнения
- Операторы присваивания
- Логические операторы
- Операторы принадлежности
- Операторы тождественности
- Битовые операторы

Подробнее:

Арифметические операторы Python

Сложение (+)

Вычитание (-)

Умножение (*)

Деление (/)

Возведение в степень (**)

Деление без остатка (//)

Деление по модулю (остаток от деления) (%)

Операторы сравнения

Меньше (<)

Больше (>)

Меньше или равно (<=)

Больше или равно (>=)

Равно (==)

Не равно (!=)

Операторы присваивания

Присваивание (=)

Сложение и присваивание (+=)

Вычитание и присваивание (-=)

Деление и присваивание (/=)

Умножение и присваивание (*=)

Деление по модулю и присваивание (%=)

Возведение в степень и присваивание (**=)

Деление с остатком и присваивание (//=)

Логические операторы Python

И (and)

Или (or)

Не (not)

Операторы принадлежности

В (in)

Нет в (not in)

Операторы тождественности

Это (is)

Это не (is not)

Битовые операторы Python

Бинарное И (&)

Бинарное ИЛИ (|)

Бинарное ИЛИ НЕТ (^)

Инвертирующий оператор (~)

Бинарный сдвиг влево (<<)

Бинарный сдвиг вправо (>>)

Ввод/вывод данных

При написании программы часто возникает необходимость во вводе своих данных, чисел, строк и т. д. И соответственно вывести полученный результат на экран.

Разберем эту ситуацию на примере языка Python 3.X

Вывод данных. Функция print()

Функция `print()` – это команда языка Python, которая выводит то, что в ее скобках на экран.

Пример:

```
>>> print(255)
```

```
255
```

```
>>> print("Hello")
Hello
```

В `print()` предусмотрены дополнительные параметры. Например, через параметр `sep` можно указать отличный от пробела разделитель объектов:

```
>>> print(1, 2, 3, sep="//")
1//2//3
```

Параметр `end` позволяет указывать, что делать, после вывода строки.

```
>>> print(5, end='\n')
10
>>>
```

В данном примере произойдет переход на одну строку, в случае когда надо перейти на две строки ниже можно использовать следующий метод:

```
>>> print(10, end='\n\n')
10
>>>
```

Форматирование может выполняться в так называемом старом стиле или с помощью строкового метода `format`. Старый стиль также называют Си-стилем, так как он схож с тем, как происходит вывод на экран в языке С. Рассмотрим пример:

```
student = "Ben"
age = 16
grade = 9.2
print("It's %s, %d. Level: %f" %(student, age, grade))
```

Результат

```
It's Ben, 16. Level: 9.200000
```

Здесь вместо трех комбинаций символов `%s`, `%d`, `%f` подставляются значения переменных `student`, `age`, `grade`. Буквы `s`, `d`, `f` обозначают типы данных – строку, целое число, вещественное число.

Ввод данных. Функция `input()`

За ввод в программу данных с клавиатуры в Python отвечает функция `input`. Когда вызывается эта функция, программа останавливает свое выполнение и ждет, когда пользователь введет

текст. После этого, когда он нажмет Enter, функция *input()* возьмет введенный текст и передаст его программе, которая уже будет обрабатывать его согласно своим алгоритмам.

Введенные данные можно присваивать переменной. Команда будет иметь следующий вид:

```
a = input()
print(a)
```

и следом можно добавить команду вывода. Попробуйте самостоятельно выполнить данный код в интерпретаторе Python.

Если нам необходимо указать тип вводимой переменной, то можно использовать команду следующего вида:

```
a = int(input())
```

где int означает, что будет произведен ввод целого числа

Можно выводить диалоговые сообщения при 'общении' с пользователем. Но не отправляйте в проверочную систему программы, содержащие лишний вывод

Ссылка на онлайн интерпретатор <https://www.programiz.com/python-programming/online-compiler/>

Попробуйте исполнить следующий код:

```
name = input('Enter your name: ')
print('Hello, ', name)
```

Результат:

```
>>Enter your name: Nikolas
>>Hello, Nikolas
```