

# Лекция 1. Введение в программирование на Python

А. Н. Лата

МГИМО(У) МИД России

07 февраля 2024 г.



# Содержание

- 1 Элементы теории алгоритмов
- 2 Основы программирования на языке Python
- 3 Ввод / Вывод
- 4 Переменные в языке Python
- 5 Встроенные типы данных в языке Python
- 6 Основные операторы в Python
- 7 Управляемые блоки кода в Python
- 8 Исключения в языке Python
- 9 Работа с библиотеками

# Неформальное (интуитивное) определение алгоритма

## Алгоритм

**Алгоритм<sup>a</sup> (или эффективная процедура)** — *точное предписание* возможному исполнителю совершить определенную последовательность действий, ведущих от *начальных данных*, которые могут варьироваться, к *искомому результату (решению задачи)* за конечное число шагов <sup>b</sup>.

---

<sup>a</sup>Мухаммеда ибн Муса ал-Хорезми (Alhorithmi)

<sup>b</sup>Неформальное определение алгоритма, так как слова, выделенные курсивом еще точно не определены.

# Основные свойства интуитивно определенного алгоритма

Каждый алгоритм должен обладать следующими свойствами:

- ❶ **Конечность** (результативность). Алгоритм должен заканчиваться за конечное (хотя и не ограниченное сверху) число шагов и получать результат, обеспечивая решение тех задач, для которых он и создавался.
- ❷ **Определенность** (детерминированность). Каждый шаг алгоритма и переход от шага к шагу должны быть точно определены и каждое применение алгоритма к одним и тем же исходным данным должно приводить к одинаковому результату.
- ❸ **Простота и понятность**. Каждый шаг алгоритма должен быть четко и ясно определен, чтобы выполнение алгоритма можно было «поручить» любому исполнителю (человеку или механическому устройству).
- ❹ **Массовость**. Алгоритм задает процесс вычисления для множества исходных данных (чисел, строк букв и т.п.), он представляет общий метод решения класса задач.

# Пример интуитивно определенного алгоритма

## Алгоритм Евклида нахождения наибольшего общего делителя двух целых положительных чисел

Даны два целых числа  $a$  и  $b$ , найти  $\text{НОД}(a, b)$ .

Выполнить следующие шаги:

- 1 Если  $a < b$ , то поменять их местами.
- 2 Разделить нацело  $a$  на  $b$ ; получить остаток  $r$ .
- 3 Если  $r = 0$ , то  $\text{НОД}(a, b) = b$ .
- 4 Если  $r \neq 0$ , заменить:  $a$  на  $b$ ,  $b$  на  $r$  и вернуться к шагу 2.

# Необходимость в формализации определения алгоритма

## Почему необходимо формальное определение алгоритма?

Не имея такого определения, **невозможно** доказать, что задача алгоритмически неразрешима, т.е. алгоритм ее решения никогда не удастся построить.

## Тезис Тьюринга–Чёрча

Для любой интуитивно вычислимой функции существует вычисляющая её значения машина Тьюринга.

## Замечание

Тезис Тьюринга–Чёрча невозможно строго доказать или опровергнуть, так как он устанавливает эквивалентность между строго формализованным понятием частично вычислимой функции и неформальным понятием вычислимости.

# Исполнитель алгоритма

## Исполнитель алгоритма

**Исполнитель алгоритма** — это некоторая абстрактная или реальная (техническая, биологическая или биотехническая) система, способная выполнить действия, предписываемые алгоритмом.

Исполнителя характеризуют:

- среда (или обстановка)
- элементарные действия
- система команд
- отказы

## Замечание

Обычно исполнитель ничего не знает о цели алгоритма. Он выполняет все полученные команды, не задавая вопросов «почему» и «зачем».

На практике наиболее распространены следующие формы представления алгоритмов:

- словесная (запись на естественном языке)
- графическая (изображения из графических символов)
- псевдокоды (полуформализованные описания алгоритмов на условном алгоритмическом языке)
- программная (тексты на языках программирования)



- ❶ Объясните, почему для алгоритма Евклида выполняется свойство конечности.
- ❷ Приведите словесное описание алгоритмов для решения следующих задач:
  - Дана последовательность из  $n$  ( $n > 0$ ) чисел. Найти максимальное (минимальное) число в последовательности.
  - Дано целое неотрицательное число  $n$ . Найти сумму цифр в десятичной записи этого числа.

# Графический способ записи алгоритмов

## Блок-схема алгоритма

При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий. Такое графическое представление называется **схемой алгоритма** или **блок-схемой**.

## Упражнение

Выпишите правила изображения блок-схем алгоритма. Разберите (письменно) примеры.

# Программный способ записи алгоритмов

## Язык программирования

**Язык программирования** — это формальный язык, используемый для формулирования структур данных и алгоритмов, то есть вычислительных правил, которые могут быть выполнены компьютером. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (обычно — ЭВМ) под её управлением.

## Компьютерная программа

**Компьютерная программа** — запись алгоритма на языке программирования.

# Содержание

- 1 Элементы теории алгоритмов
- 2 Основы программирования на языке Python**
- 3 Ввод / Вывод
- 4 Переменные в языке Python
- 5 Встроенные типы данных в языке Python
- 6 Основные операторы в Python
- 7 Управляемые блоки кода в Python
- 8 Исключения в языке Python
- 9 Работа с библиотеками

# Базовые сведения о Python

Python<sup>3</sup> является высокоуровневым языком программирования.



<sup>3</sup>был разработан в конце 1980-х годов. Его разработчик — Гвидо Ван Россум

# Основные особенности языка программирования Python I

- Python поддерживает структурное, функциональное и объектно-ориентированное программирование (**парадигмы программирования**).
- Python — **интерпретируемый** язык программирования. Это означает, что программный код на языке Python не компилируется перед его исполнением в отличие от таких языков, как C и Fortran. **Код выполняется специальной программой-интерпретатором**. Интерпретатор считывает высокоуровневую программу — исходный код — и, напрямую взаимодействуя с операционной системой, выполняет программу. **Преобразование и выполнение программы осуществляется построчно**.
- Python поддерживает **динамическую типизацию** — связывание переменной с типом в момент присваивания ей значения.
- В Python реализовано автоматическое управление памятью.

# Основные особенности языка программирования Python II

- Python поддерживает интерактивный режим работы (Jupyter Notebook / Lab).
- Для Python написано большое количество свободно распространяемых библиотек, в том числе библиотек для научных расчётов.
- Синтаксис языка Python является легко читаемым и неперегруженным.
- **Кроссплатформенность**: программный код, написанный на Python, может выполняться на любой платформе, на которой установлен интерпретатор Python.

# Установка Python

В качестве версии языка используем Python 3.x



[Русскоязычная документация по Python](#) (ссылка)



# Интерактивный режим работы: Jupyter Notebook или Jupyter Lab

- В интерактивном режиме работы введенные пользователем с клавиатуры команды сразу же выполняются интерпретатором, а результат выводится на экран.
- Работа с интерпретатором осуществляется в окне оболочки Jupyter Notebook (или Jupyter Lab).
- Каждая строка ввода/вывода в Jupyter Notebook (или Jupyter Lab) последовательно нумеруется.
- Порядковый номер, указываемый в квадратных скобках после слова Out, позволяет обращаться к ранее полученному результату.
- Справку на английском языке по любому объекту можно получить, набрав в окне IPython имя объекта со знаком ?
- Для вывода на экран значения переменной в интерактивном режиме достаточно написать в консоли имя переменной и нажать Enter.

Синтаксис Python отличается своей простотой и ясностью. Рассмотрим основные элементы:

- Вместо фигурных скобок или ключевых слов для обозначения блоков кода используются отступы. Отступ должен быть одинаковым и состоять из пробелов или табуляции.
- Комментарии начинаются с символа `#` и необходимы для пояснения кода или временного отключения определенных участков кода.
- Переменные объявляются присваиванием значения. Тип переменной определяется автоматически во время выполнения программы.

# Содержание

- 1 Элементы теории алгоритмов
- 2 Основы программирования на языке Python
- 3 Ввод / Вывод**
- 4 Переменные в языке Python
- 5 Встроенные типы данных в языке Python
- 6 Основные операторы в Python
- 7 Управляемые блоки кода в Python
- 8 Исключения в языке Python
- 9 Работа с библиотеками

# Вывод информации на экран

Для вывода информации на экран в программах на языке Python используется инструкция `print()`. В скобках указывается то, что нужно вывести.

Что указано в скобках	Пример	На экран будет выведено	Пример
1. Текст	<code>print('Привет!')</code>	Текст без кавычек, включая возможные начальные и конечные пробелы	Привет!
2. Число	<code>print(-2)</code>	Соответствующее число	-2
3. Имя переменной величины	<code>print(x1)</code>	Значение величины	273
4. Выражение	<code>print(a * b)</code>	Значение выражения	1024
5. Метод	<code>print(famil.upper())</code>	Результат применения метода	ЛУКИН

# Вывод информации на экран

Можно указывать несколько значений, в том числе разного типа, разделяя их запятой.

Между указанными в инструкции `print()` значениями (будем называть их «список вывода») выводится также один пробел.

Этот разделитель можно изменить на любой другой символ (или последовательность символов).

Для этого после списка вывода нужный символ-разделитель указывается как параметр `sep` инструкции.

```
print(<список вывода>, sep = ', ')
```

```
print(<список вывода>, sep = ', ', end = ") или
```

```
print(<список вывода>, end = ", sep = ', ')
```

# Ввод данных с клавиатуры

Для ввода данных с клавиатуры используется функция `input()` из стандартной библиотеки Python.

После срабатывания команды `input()` поток выполнения программы останавливается в ожидании данных, которые пользователь должен ввести с помощью клавиатуры.

После ввода данных и нажатия Enter, функция `input()` завершает своё выполнение и возвращает результат, который представляет собой строку символов, введённых пользователем.

# Содержание

- 1 Элементы теории алгоритмов
- 2 Основы программирования на языке Python
- 3 Ввод / Вывод
- 4 Переменные в языке Python**
- 5 Встроенные типы данных в языке Python
- 6 Основные операторы в Python
- 7 Управляемые блоки кода в Python
- 8 Исключения в языке Python
- 9 Работа с библиотеками

# Переменные величины

## Переменные

Переменные величины (или, короче, переменные) – величины, которые при выполнении программы могут принимать различные значения.

## Константы

Существуют также так называемые «константы» – так в программировании называют величины, которые во время выполнения программы не меняются. В языке Python имеются также «встроенные» константы, например константы логического типа `True` и `False`.

Каждая переменная характеризуется **именем** и **типом**.



## Ключевые слова

Есть набор слов, которые нельзя использовать в качестве имен переменных, так как эти слова «зарезервированы» в языке Python для определенных целей (эти слова называют «зарезервированными», или «служебными», или «ключевыми»).

Тип величины определяет:

- какие значения может принимать величина (область допустимых значений переменной)
- какие операции можно проводить над переменной (множество допустимых операций с величиной)
- какой объем памяти компьютера требуется для хранения значения данной переменной и в каком формате будут храниться данные

# Содержание

- 1 Элементы теории алгоритмов
- 2 Основы программирования на языке Python
- 3 Ввод / Вывод
- 4 Переменные в языке Python
- 5 Встроенные типы данных в языке Python**
- 6 Основные операторы в Python
- 7 Управляемые блоки кода в Python
- 8 Исключения в языке Python
- 9 Работа с библиотеками

# Встроенные типы данных в языке Python

Типы данных в Python можно разделить на два класса:

- атомарные
- структурные

## Атомарные типы данных

- число с плавающей точкой (`float`)
- целое число (`int`)
- логический тип (`bool`)
- комплексное число (`complex`)

## Структурные типы данных

- списки (в частности, строки `str`)
- кортежи (`tuple`)
- словари, классы, функции и т. д.

# Встроенные типы данных в языке Python

## Строки

Строки — это наборы символов. Для записи строк в Python используются как символы апострофа, так и кавычки: `'label'`, `"label"`

## Список

Список — упорядоченная последовательность элементов, пронумерованных с 0. Элементы списка перечисляются в квадратных скобках через запятую и могут иметь разные типы.

# Встроенные типы данных в языке Python

## Кортеж

Кортеж — это неизменяемый список. Кортежи объявляются так же, как списки, только с использованием круглых скобок. Обращение к элементам кортежа осуществляется так же, как обращение к элементам списка: через указание индекса искомого элемента в квадратных скобках.

## Замечание

Удобство использования кортежей заключается в том, что их невозможно случайно изменить. Кроме того, кортежи занимают в оперативной памяти меньше места, чем списки той же длины.

## Словарь

Словарь — это неупорядоченный набор объектов, записанных в виде пары «ключ: значение».

# Содержание

- 1 Элементы теории алгоритмов
- 2 Основы программирования на языке Python
- 3 Ввод / Вывод
- 4 Переменные в языке Python
- 5 Встроенные типы данных в языке Python
- 6 Основные операторы в Python**
- 7 Управляемые блоки кода в Python
- 8 Исключения в языке Python
- 9 Работа с библиотеками

# Основные операторы в Python

## Оператор и Операнд

**Оператор** в Python — это символ, который выполняет операцию над одним или несколькими операндами.

**Операндом** выступает переменная или значение, над которыми проводится операция.

# Основные операторы в Python

Операторы в Python бывают следующих типов:

- процедурные операторы
- операторы функций
- операторы классов
- операторы исключений
- операторы модулей
- (по)битовые операторы
- арифметические операторы
- операторы сравнения
- операторы присваивания
- логические операторы
- операторы принадлежности
- операторы тождественности



# Операции и логические операторы в Python

Рассмотрим некоторые наиболее часто используемые в Python операторы.

- **Операция присваивания** переменной значения делается с помощью знака `=`
- Арифметические операции над числами: `+` (сложение), `-` (вычитание), `/` (деление), `*` (умножение), `/` (деление), `//` (целочисленное деление), `%` (нахождение остатка от деления)
- **Команда возведения в степень**: `**` (двойная звёздочка)
- **Команды преобразования типов**: `int(a)` — преобразование аргумента `a` к целому числу; `float(a)` — преобразование аргумента `a` к числу с плавающей точкой; `str(a)` — преобразование аргумента `a` к строке.
- **Оператор проверки равенства**: `==`. Результатом действия операции является `True`, если объекты совпадают, и `False` — в противном случае.
- **Оператор `in`** — проверка, содержится ли объект в списке, кортеже или словаре.

# Упражнение

## Упражнение

Составьте Jupyter Notebook, в который выпишите **основные операторы Python** (из второй колонки см. слайд 32) с примерами.

# Содержание

- 1 Элементы теории алгоритмов
- 2 Основы программирования на языке Python
- 3 Ввод / Вывод
- 4 Переменные в языке Python
- 5 Встроенные типы данных в языке Python
- 6 Основные операторы в Python
- 7 Управляемые блоки кода в Python**
- 8 Исключения в языке Python
- 9 Работа с библиотеками

# Управляемые блоки кода в Python: Условный оператор if

Данный оператор сообщает интерпретатору, что некоторый блок кода необходимо выполнять только при определённом условии. Синтаксис конструкции:

```
if <условие 1>:  
    <блок операторов 1>  
elif <условие 2>:  
    <блок операторов 2>  
...  
elif <условие n-1>:  
    <блок операторов n-1>  
else:  
    <блок операторов n>
```

# Управляемые блоки кода в Python: Оператор цикла for

Оператор цикла позволяет выполнять блок некоторых операций заданное число раз. Чаще всего используется перебор значений переменной цикла из заранее заданного списка значений. Синтаксис конструкции:

```
for <переменная цикла> in [список значений]:  
    <блок операторов>
```

Каждая строка в блоке кода внутри конструкции **for** должна быть выделена с помощью отступа.

Перебор значений переменной цикла можно осуществлять с помощью любой последовательности (строки, списка, словаря, строк в файле и т. п.).

# Управляемые блоки кода в Python: Оператор цикла while

Данная управляющая конструкция позволяет выполнять некоторый блок кода до тех пор, пока является истинным некоторое заданное условие.

Синтаксис конструкции:

```
while <условие>:  
    <блок операторов>
```

Каждая строка в блоке кода внутри конструкции **while** должна быть выделена с помощью отступа.

# Управляемые блоки кода в Python: Функции I

Функции в программировании нужны для повторного использования некоторого блока кода. Синтаксис объявления функции в Python:

```
def <имя функции>(<список формальных аргументов>):  
    <блок операторов>  
    return <результат>
```

## Объявление функции

Объявление функции начинается с зарезервированного слова `def`. После него указывается имя функции `<имя функции>`. Формальные аргументы функции перечисляются в круглых скобках после её имени. После скобок ставится знак двоеточия. Тело функции может состоять из произвольного числа строк кода, каждая из которых выделяется с помощью отступа. В конце функции может располагаться оператор `return` с последующим указанием значения `<результат>`, которое будет возвращать функция при её вызове. Если в теле функции отсутствует строка, начинающаяся с `return`, то при вызове такая функция возвращает значение `None` (нулевой указатель). Строка документации, описывающая то, как работает функция, оформляется с помощью тройных кавычек. Вызов функции осуществляется указанием её имени со списком фактических аргументов в круглых скобках.



# Упражнение

## Упражнение

Составьте Jupyter Notebook, в который выпишите определение и примеры **Лямбда-функций в Python**.

# Содержание

- 1 Элементы теории алгоритмов
- 2 Основы программирования на языке Python
- 3 Ввод / Вывод
- 4 Переменные в языке Python
- 5 Встроенные типы данных в языке Python
- 6 Основные операторы в Python
- 7 Управляемые блоки кода в Python
- 8 Исключения в языке Python**
- 9 Работа с библиотеками

# Исключения в языке Python I

**Исключения** в Python — тип данных, с помощью которых разработчик узнает об ошибках и необычных ситуациях.

Примеры исключений в Python:

- **TypeError** означает, что операция выполняется над объектом несовместимого типа данных.
- **ValueError** возникает, когда функция получает аргумент правильного типа, но с неправильным значением.
- **IndexError** появляется, когда происходит попытка получить доступ к элементу списка или строки с использованием недопустимого индекса.
- **KeyError** — при попытке получить доступ к элементу словаря по ключу, которого нет в словаре.
- **FileNotFoundError** — возникает, при открывании файла, которого не существует.

# Исключения в языке Python II

- `ZeroDivisionError` — деление на ноль происходит в программе.
- `IOError` — появляется, когда возникают проблемы с вводом-выводом, например, при чтении или записи файлов.

Вы можете создавать собственные исключения в Python, отталкиваясь от встроенных классов исключений.

# Содержание

- 1 Элементы теории алгоритмов
- 2 Основы программирования на языке Python
- 3 Ввод / Вывод
- 4 Переменные в языке Python
- 5 Встроенные типы данных в языке Python
- 6 Основные операторы в Python
- 7 Управляемые блоки кода в Python
- 8 Исключения в языке Python
- 9 Работа с библиотеками

# Работа с библиотеками I

Рассмотрим способы подключения внешних библиотек (модулей) в Python, на примере математической библиотеки `math`<sup>4</sup>.

- 1 Подключение библиотеки осуществляется с помощью команды `import`.  
В этом случае все функции из библиотеки `math` загружаются в текущее пространство имён интерпретатора так, что пользователь может получить доступ к каждой функции через префикс `math`.
- 2 Загрузка всех функций из некоторой библиотеки в текущее пространство имён интерпретатора делается с помощью знака звёздочка: `from math import *`. В этом случае все функции из библиотеки `math` доступны без префикса.
- 3 Загрузка в текущее пространство имён конкретных функций из заданной библиотеки: `from math import log10, fact`. В рассмотренном примере, из библиотеки `math` загружаются только функции `log10` (десятичный логарифм) и `fact` (факториал). Загруженные функции можно использовать без префикса `math`.

- 4 Подключение библиотеки с псевдонимом.  
Синтаксис конструкции: `import math as m`. В дальнейшем обращение к функциям библиотеки `math` осуществляется с помощью префикса-псевдонима, а не полного имени библиотеки.

---

<sup>4</sup>Описание полного списка функций библиотеки можно получить с помощью команды `help('math')`