

Константин Леонидович Белянский

Программирование на языке Python

Факультет проблем физики и энергетики

Кафедра "Фундаментальные взаимодействия и космология"

Осенний семестр 2019-2020 учебного года

<http://python.inr.ru>

E-Mail: python@inr.ru

Основная литература:

1. Марк Лутц. "Изучаем Python", 4-е издание. Пер. с англ. СПб., Символ-Плюс, 2011. 1280 с, ил. ISBN 978-5-93286-159-2

Дополнительная литература:

1. Марк Лутц. "Python. Карманный справочник", Пер. с англ. М, Вильямс, 2015, 320 с, ил. ISBN 978-5-8459-1965-6
2. Марк Лутц. "Программирование на Python", 4-е издание, в двух томах. Пер. с англ. СПб., Символ-Плюс, 2011, 992 с, ил. ISBN т.1 978-5-93286-211-7, т.2 978-0-596-15810-1
3. Марк Саммерфилд. "Программирование на Python 3. Подробное руководство", Пер. с англ. СПб., Символ-Плюс, 2009. - 608 с, ил. ISBN: 978-5-93286-161-5
4. Стив Макконнелл. "Совершенный код. Мастер-Класс", Пер. с англ. М. "Русская Редакция", 2010, 896 с, ил. ISBN 978-5-7502-0064-1

Технические средства курса

- Слайды лекций, примеры к лекциям и домашние задания можно найти на сайте

`http://python.inr.ru/`

- Для выполнения домашних заданий будет использоваться учебный сервер

`pitomnik.inr.ru`

- доступ к серверу осуществляется по протоколу ssh

- Простой способ скопировать примеры к лекции на учебный сервер:

`wget http://python.inr.ru/e1901.py`

- Со всеми вопросами можно обращаться по электронной почте

`python@inr.ru`

Лекция 1

Введение

**Python и его место среди языков
программирования**

Запуск Python программ

Ресурсы в сети Internet

Официальный сайт языка Python

<https://www.python.org/>

Здесь находятся:

- Официальная документация по языку и стандартной библиотеке
 - отличный справочный материал, не столько для изучения, скорее для поиска ответа на любой вопрос
- Документы PEP (Python Enhancement Proposals)
 - фактически это изложение стандарта языка
- Инсталляционные пакеты для Windows, Mac OS X и Linux
 - Все дистрибутивы Linux уже содержат интерпретатор Python
 - При установке Python для Windows следует разрешить включение Python в System Path

Ресурсы в сети Internet

Специальные сборки Python, включающие библиотеки для научных расчетов

- Anaconda
 - <https://www.anaconda.com/distribution/>
 - Есть пакеты для Windows, Mac OS X и Linux
 - Лицензия BSD

Текстовые редакторы, требования:

- Поддержка отступов
- Подсветка синтаксиса
- Автодополнение
- Запуск разрабатываемой программы из редактора

Текстовые редакторы, примеры

- SciTE (Linux, Windows)
 - <http://www.scintilla.org/SciTE.html>
 - Простой редактор в виде единственного .exe файла
- Geany (Linux, Mac OS X, Windows)
 - <https://www.geany.org/>
 - Более развитый редактор с поддержкой проектов
- Sublime Text 3 (демо-версия, Linux, Mac OS X, Windows)
 - <https://www.sublimetext.com/3>
 - Редактор популярный среди Web-разработчиков
- PyCharm Community Edition (Linux, Mac OS X, Windows)
 - <https://www.jetbrains.com/pycharm/>
 - Бесплатная версия интегрированной среды разработки на Питоне, есть базовая поддержка других языков
 - Запускается как `charm`

Web-ориентированная среда Jupyter

- Среда имитирует рабочую тетрадь исследователя. Заметки, расчеты и результаты отображаются в общем пространстве.
- В качестве пользовательского приложения используется Web-браузер
- Имеет модульную архитектуру и в дополнение к Питону поддерживает ряд других языков программирования, в частности:
 - C#
 - Fortran
 - Mathematica
 - MATLAB
 - Perl
 - PHP
 - R

Среда исполнения программ

Операционные системы

- Семейство UNIX
- Семейство Windows
- ?

Архитектуры процессоров

- Intel / AMD : x86 / x86_64
- ARM
- MIPS
- IBM POWER
- Множество архитектур микроконтроллеров

Сетевые утилиты:

- Для запуска программ на удаленных Linux серверах используется ssh протокол с шифрованием трафика
 - утилита ssh обеспечивает терминальный доступ
 - утилита scp копирует файлы
- Все дистрибутивы Linux уже содержат утилиты ssh и scp
- MobaXterm - один из лучших ssh / scp пакетов для Windows
 - <http://mobaxterm.mobatek.net/>
 - есть бесплатная версия
 - обеспечивает запуск приложений с графическим интерфейсом пользователя (включает в себя X-server)
- PuTTY - Простейший ssh / scp клиент для Windows
 - <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
 - Открытое ПО, лицензия MIT

Локальное редактирование + удаленное исполнение

При работе на удаленном сервере часто используется следующая модель поведения:

- Запуск терминала на удаленном сервере с возможностью работы с графическими приложениями

```
ssh -Y username@remote.server.com или ssh сессия в MobaXterm  
cd working/directory
```

- Редактирование файлов программы на компьютере пользователя
- Копирование файлов программы на удаленный сервер

```
scp -p *.py username@remote.server.com:working/directory
```

- Запуск программы в окне терминала удаленного сервера

```
python3 my_program.py
```

Запуск Python программы

- Программа не имеет специального оформления, строки кода помещенные в файл с расширением .py исполняются последовательно.
- Windows
 - Python устанавливается как обычное Windows приложение
 - Файлы с расширением .py запускаются двойным щелчком мыши
 - Запуск из командной строки: `python имя_файла.py`
Для этого нужно разрешить включение Python в System Path
- Linux
 - Python входит в дистрибутив Linux как пакет
 - Запуск из командной строки: `python3 имя_файла.py`
- Запуск без имени файла - интерактивный режим, выход: `exit()`
а также: `Ctrl-Z + Enter` в Windows или `Ctrl-D` в Linux
- Среда IDLE - интерактивный интерпретатор. В Linux запускается как `idle3`, содержит отладчик
- Редактор Geany: программа запускается нажатием клавиши `F5`

Формат текстового файла

- Текстовый файл это последовательность строк с разделителями между ними
- В UNIX, Linux и MAC OS X разделителем является символ перевода строки "\n" (код 10)
- В MS DOS и MS Windows разделителем является последовательность из *двух символов*: возврат каретки и перевод строки "\r\n" (коды 13, 10)
- Несовпадение формата файла с форматом ожидаемым операционной системой может привести к проблемам
- Преобразование формата файла из DOS (Windows) в UNIX (Linux)

```
dos2unix file.txt
```

- Преобразование формата файла из UNIX (Linux) в DOS (Windows)

```
unix2dos file.txt
```

- Содержимое файла в виде последовательности байт можно посмотреть используя утилиты *od* или *hexdump*:

```
hexdump -C file.txt  
od -tx1 -a file.txt
```

Первые две строки программы

- Файл программы на Питоне как правило начинается со следующих двух строк:

```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-
```

- Первая строка сообщает исполняющей системе какую программу следует запустить для интерпретации данного файла
 - В Linux файл имеющий такую первую строку может быть запущен следующим образом:

```
chmod 755 program.py # придание файлу разрешения на исполнение  
./program.py        # исполнение программы
```
 - В Windows эта строка считывается как комментарий
- Вторая строка определяет в какой кодировке находятся литералы строк

Python и другие языки программирования

- компилируемые, статическая типизация
 - C/C++, C# (Microsoft), Objective-C (Apple), Pascal / Delphi
 - Fortran, Algol, PL/I, Modula 2, Ada
 - Go, Rust - новые языки, поддержка параллелизма, автоматическое управление памятью
- интерпретируемые, динамическая типизация
 - Basic (VB), Tcl/Tk, Perl, Python, Ruby - универсальные
 - Javascript (ECMAScript) - интерпретируется в браузере
 - Dart - разрабатывается в Google, компилируется в Javascript
 - PHP - интерпретируется на Web-сервере
 - Интерпретаторы команд - sh, bash, csh, Microsoft PowerShell
- Java
 - корпоративные системы, программы для Android

Python, краткая история

- Создатель: Гвидо Ван Россум (Guido van Rossum)
 - 1991 г. февраль - первая публикация
 - 1994 г. январь - версия 1.0
 - 2000 г. 16 октября - версия 2.0
 - 2008 г. 3 декабря - версия 3.0 (Python 3000)
 - 2019 г. 8 июля - версия 3.7.4
 - 2019 г. 21 октября - версия 3.8.0
- Формат номера версии: MAJOR.MINOR.PATCH
- CPython - эталонная реализация
- Другие реализации:
 - Jython - реализация Python на языке Java
 - PyPy - реализация Python на языке ... Python - для прототипирования новых возможностей языка
 - Несколько реализаций Питона для Android

Версии языка Python

- Версия 1 уже давно не используется
- Версия 2
 - Все еще используется в ряде проектов
 - Будет поддерживаться до 2020 г.
- Версия 3, изменения:
 - Эффективная работа с большими объемами данных
 - Пример: результат функции `range(0, 100)`
 - Строгая реализация работы с Unicode
 - Последовательности байт выделены в отдельный тип данных
 - инструкция `print` заменена функцией `print()`
 - Увеличена гибкость операции вывода на консоль
 - В версию 2 введен модуль `future` для обеспечения обратной совместимости

Python это открытое программное обеспечение

- Интерпретатор
 - Собственная лицензия, совместимая с GPL
 - Нет ограничений на интеграцию с коммерческими продуктами. Сделанные изменения не обязательно должны иметь открытую лицензию.
- Программы
 - Реализация программы, как интерпретируемого текста препятствует ее "закрытию"
 - Есть возможность распространения программ в виде байт-кода или в виде монолитного исполняемого файла, однако такой подход встречается достаточно редко
- Открытое и бесплатное программное обеспечение это не одно и то же

Почему Python ?

- Используется в CERN
 - наличие стандартных библиотек, ориентированных на научные вычисления
 - комплексный тип данных, как один из базовых числовых типов
- Используется в Google
 - Google App Engine
 - решение ряда внутренних инженерных задач
- Фактический стандарт интерпретируемого языка программирования в Linux, постепенно заменяет собой Perl
- Высокая производительность труда программиста
- Программы [почти] независимы от платформы исполнения
- Один из наиболее востребованных на рынке труда интерпретируемый язык программирования

Недостатки Python

- медленнее чем C/C++
- распараллеливание программ сильно затруднено
 - параллельные процессы достаточно эффективны
 - мультитредовые приложения ограничены GIL
- много возможностей, необходимость самодисциплины
- часть ошибок выявляется только на этапе исполнения, это общий недостаток интерпретируемых языков
 - юнит-тестирование приобретает особую важность
 - непрерывная интеграция позволяет обнаруживать ошибки на ранних этапах разработки

Терминология

- В лекциях использована терминология русского перевода книги Марка Лутца "Изучаем Python" [1]
- *Программа* на Питоне это последовательность инструкций
- *Инструкция* это
 - инструкция управления, например цикл или условие
 - инструкция присваивания: *переменная = выражение*
 - одиночное выражение записанное в строке текста программы, в том числе элементарное выражение
- *Оператор* - символ операции допустимой в выражении. Полный список операторов будет представлен на лекции 2.
- Вызов функции является оператором
- *Выражение* это последовательность операндов соединенных операторами. *Выражение не может содержать инструкций.*
- *Операнд* это выражение, в том числе элементарное выражение
- *Элементарное выражение* это литерал или переменная
- *Литерал* это буквальная запись данных, например число или строка
- Следует различать строку текста программы (line) и строку-литерал (string)

Особенности синтаксиса и терминологии

- Строка текста программы это инструкция
- Отступы это часть синтаксиса
- *Составная конструкция языка* состоит из заголовка и блока
 - Заголовок это строка, завершающаяся двоеточием
 - Блок это последовательность строк, отступ которых на единицу больше, чем у строки-заголовка

```
if apples > pears:  
    eat_apples(apples - pears)  
    print('We ate extra apples')  
get_more_fruits()
```

- Массив это *list*, неизменяемый массив это *tuple* (кортеж)
- Хеш или map (ассоциативный массив) это *dict* (словарь)
- Язык поддерживает тип данных *set* (множество) к которому применимы теоретико-множественные операции
- Python создан математиком, а Perl лингвистом

Функция `print()`

- Начиная с версии 3 в языке Python *инструкция* `print` заменена *функцией* `print()`
- Большинство объектов имеют свойство "быть напечатанным", то есть быть преобразованным в текстовую строку
- Функция `print()` воспринимает любое количество аргументов и "печатает" их последовательно в стандартный вывод разделяя пробелом и завершая символом новой строки
- Именованные аргументы позволяют настроить поведение функции `print()`
 - аргумент *sep* позволяет заменить разделитель
 - аргумент *end* позволяет заменить конец строки
 - аргумент *file* позволяет направить вывод в файл
- Пример замены разделителя:

```
print(1, 2, 3) # => 1 2 3
print(1, 2, 3, sep='->') # => 1->2->3
```

Литерал строки

- Литерал строки это последовательность символов, заключенных в одинарные или двойные кавычки
- Литералы в одинарных кавычках и литералы в двойных кавычках эквивалентны и интерпретируются одинаково
- Литералы ограниченные тремя символами кавычек могут занимать несколько строк текста программы, при этом символы переноса строки входят в состав литерала как символ `\n` (10)
 - символ кавычек может быть как двойным, так и одинарным
- Внутри строк интерпретируются C Esc-последовательности
- Переменные внутри строк не интерпретируются. Для "внедрения" в строку значения переменной существует оператор `%` и синтаксис аналогичный функции `printf()` языка C
 - Пример:

```
s = '%2d часов\t%02d минут' % (16, 30)
print(s)
```

Комментарии

- Символ комментария `#`. Сам символ и все символы справа от него и до конца строки игнорируются
- Для комментирования большого фрагмента текста можно использовать строки в тройных кавычках
 - Строка это элементарное выражение, а выражение это инструкция, то есть синтаксически корректная конструкция
- Примеры:

```
a = 1
a = a * 2 # здесь a увеличивается вдвое
a = a * 2 # здесь a увеличивается вдвое еще раз
a = a * 2 # и еще раз
"""
a = a * 2 # и еще раз
a = a * 2 # и еще раз
a = a * 2 # и еще раз
"""
```

`#` После всех вычислений переменная `a` примет значение 8

Пример синтаксически корректной программы

- Одиночное выражение, записанное в строке программы является допустимой инструкцией

```
a = 3 + 2    # инструкция присваивания
3 + 2        # одиночное выражение записанное в строке
"A string"   # элементарное выражение
3            # элементарное выражение
a            # элементарное выражение
a == 12      # одиночное выражение записанное в строке
```

- Вызов функции без использования ее результата также является одиночным выражением, записанным в строке текста программы

```
exit(1)
```

Пробелы и отступы (PEP 8)

- Отступы составлены из пробелов или горизонтальных табуляций. Рекомендуемый единичный отступ 4 пробела.
- В отступах нельзя смешивать пробелы и табуляции; в Питоне версии 2 это допустимо но не рекомендуется
- За пределами отступов:
 - пробел и табуляция эквивалентны
 - пробел это разделитель, но он необходим только там, где нет другого разделителя
 - там где есть один пробел можно добавить любое количество пробелов
- Примеры:

`a=2+3*4 # Без пробелов`

`a = 2 + 3 * 4 # То же, но с пробелами`

`a=a+1 if a==14 else a+4 # Правильно`

`a=a+1 ifa==14 else a+4 # Синтаксическая ошибка`

Переносы строк (PEP 8)

- Длинную строку текста программы можно перенести используя символ `\` в точке переноса
- Конструкции, заключенные в скобки `()`, `[]`, или `{ }` можно переносить в любом месте - свободный формат
- Примеры:

```
a = 0.5 - math.sin(alpha + math.pi/2)
a = 0.5 - \
    math.sin(alpha + math.pi/2)
a = 0.5 - math.sin(alpha +
math.pi/2)
```

- Точка переноса интерпретируется как пробел:

```
b = alpha ** 2 # Правильно
b = alpha *\
* 2 # Синтаксическая ошибка, оператор ** разорван пробелом
s = 'ab\
cd' # => 'abcd' - однако пробел не вставляется в строковые литералы
```

Несколько инструкций в строке

- В одной строке можно записать несколько инструкций разделяя их символом ; (точка с запятой)

```
a = math.sin(alpha + math.pi/2) ; a = 0.5 - a ; print(a)
```

- В составной конструкции языка блок можно записать строке заголовка, сразу после двоеточия

```
if a < 0: a = -a ; print(a) ; a = a + 1 ; print(a)
```

- в этом примере все четыре инструкции следующие за двоеточием будут исполнены только в том случае если a меньше нуля
- Несколько инструкций в строке это плохой стиль, однако его можно встретить:
 - в больших группах инструкций `if`
 - в группах отладочных инструкций
 - в лекциях для экономии места на слайде

Объект

- *Объект* это композитная конструкция включающая в себя именованные элементы
- Элементы в составе объекта называют *атрибутами*
- Атрибуты могут быть объектами, функциями или данными элементарных типов
- Для атрибутов-функций введен специальный термин *метод*
- Для обращения к атрибуту (методу) используется оператор точка

```
a.data = 3  
a.add(2)
```

- Метод имеет доступ к атрибутам и методам своего объекта используя аргумент `self`

```
result = self.data + value
```

- Тип объекта хранится внутри объекта, атрибут `__class__`

Объекты, классы типы

- Почти все, что есть в языке Питон это объекты
- Объект это воплощение (конкретная реализация) класса
- Класс это набор свойств, передаваемый объекту при его создании
- В Питоне тип объекта и его класс фактически синонимы
 - Создавая новый класс мы вводим новый тип данных с именем созданного класса
 - Встроенные в язык типы данных и типы данных введенные стандартной библиотекой реализованы соответствующими классами
- Класс это тоже объект (!), атрибут объекта `__class__` это ссылка на объект, являющийся его классом
- Питон поддерживает как процедурный, так и объектно-ориентированный стили программирования
- Внутри интерпретатора программа представлена в виде ориентированного графа, исполнение программы это обход графа

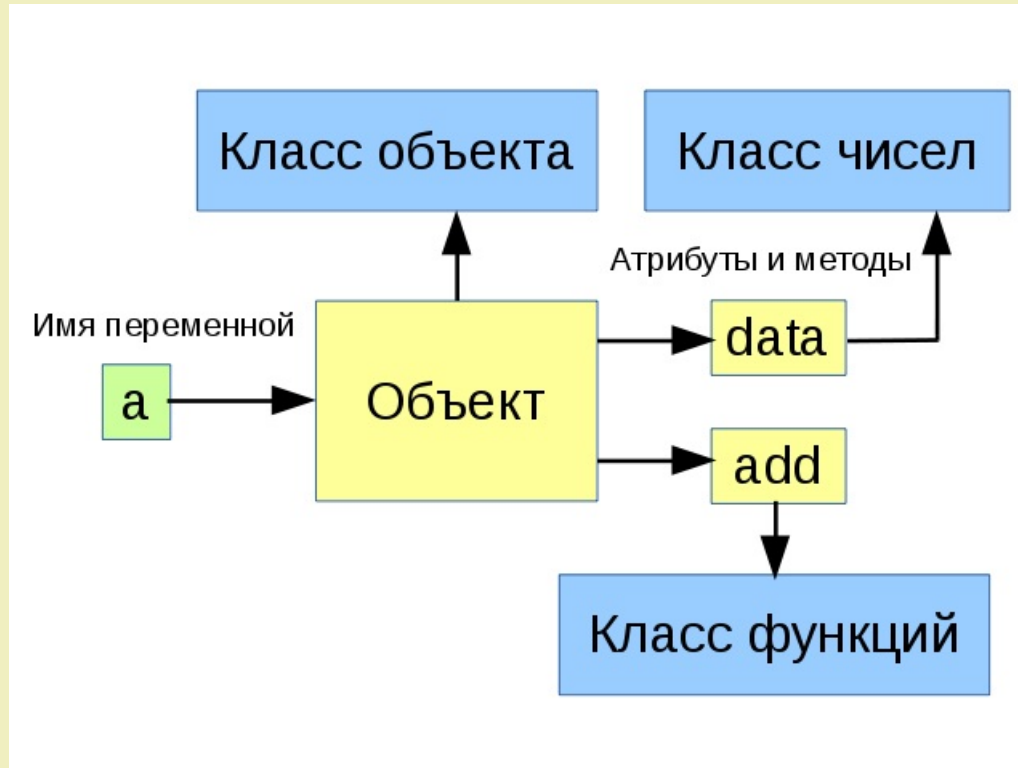
Переменные и инструкция присваивания

- Для сохранения результата вычисления выражения используется переменная

```
n = 4 + 6 * 2  
s = 'Long' + 'Name'
```

- Результат вычисления выражения это объект, находящийся во внутренних структурах данных интерпретатора
- Объект имеет тип, то есть представляет собой данные определенного типа
- Переменная это имя объекта или label (ярлык). Сама по себе переменная типа аналогичного типам C++ не имеет и может быть "наклеена" на объект любого типа.
- Операция присваивания это *акт привязки* имени переменной к объекту
- Присваивание не создает объект; объекты создаются и уничтожаются интерпретатором по мере необходимости

Объект, его класс, имя и атрибуты



Три фундаментальных конструкции

- В основе реализации языка Python лежат три фундаментальных конструкции:

Присваивание

Присваивание это придание имени объекту. Присваивание является реализацией целого ряда операций языка.

Обращение к атрибуту

Обращение к атрибуту это возможность получить доступ к отдельной части составного объекта

Выражение

Важно понимать как и из чего можно составить выражение, как выражение вычисляется, и где выражение можно использовать

- *Знание этих конструкций критически важно для понимания языка*

Функции

- Определение функции это присваивание имени объекту-функции

```
def mul(a, b):  
    n = a * b  
    return n
```

- Вызов функции

```
result = mul(2, 4)  
print('product =', result) # => product = 8
```

- При определении функции указываются ее *аргументы* (a и b)
- При вызове функции ей передаются *параметры* (2 и 4)
- Передача параметра эквивалентна инструкции присваивания:

```
переменная_аргумент = передаваемый_параметр
```

- Имя функции это такая же переменная, как и любая другая
- *Функции можно определять внутри функций*

Включение файлов в текст программы

- Включение файла в виде собственного пространства имен
 - `import math`
`a = math.sin(alpha)`
- Включение имен из файла в текущее пространство имен
 - `from math import sin, cos, tan`
`a = sin(alpha)`
- Изменение имен при включении из файла
 - `from math import sin as sinus, cos as cosinus`
`a = sinus(alpha)`
- `math` может быть файлом `math.py`, файлом `math.pyс` или специально оформленным модулем, написанным на Питоне или другом языке программирования

Файл, модуль, объект

- Файл с текстом программы на Питоне называют *модулем*
- Файл переданный интерпретатору для исполнения называют главным модулем. Имя главного модуля `__main__`
- Модуль это объект. Определенные в модуле глобальные переменные это его атрибуты, а функции это его методы. Имена функций также являются глобальными переменными.
- При использовании конструкции вида

```
from math import sin, cos, tan
```

в текущем модуле создаются глобальные переменные `sin`, `cos` и `tan`, которые ссылаются на соответствующие функции из модуля `math`

- Строго говоря, "глобальная" переменная это переменная уровня модуля. Получить доступ к такой переменной можно только явным использованием импорта.

Идеология языка или Zen of Python

- Красивое лучше уродливого
- Явное лучше неявного
- Простое лучше сложного
- Сложное лучше запутанного
- Развернутое лучше вложенного
- Разреженное лучше плотного
- Читаемость имеет значение
- Особые случаи не настолько особые, чтобы нарушать правила.
При этом практичность важнее безупречности.
- Ошибки не должны замалчиваться
Если не замалчиваются явно
- Встретив двусмысленность, отбросьте искушение угадать

Идеология языка или Zen of Python

- Должен существовать один - и, желательно, только один - очевидный способ сделать это.
Хотя он поначалу может быть и не очевиден, если вы не голландец (то есть автор языка)
- Сейчас лучше, чем никогда
- Хотя никогда зачастую лучше, чем *прямо* сейчас
- Если реализацию сложно объяснить - идея плоха
- Если реализацию легко объяснить - идея, возможно, хороша
- Пространства имен - отличная штука! Будем делать их побольше!

Эти правила всегда можно прочесть набрав в командной строке интерпретатора `import this`