

Занятие 1: Введение в Python

Практикум на ЭВМ 2017/2018

Попов Артём Сергеевич
Кропотов Дмитрий Александрович

МГУ имени М. В. Ломоносова, факультет ВМК, кафедра ММП

Организационные вопросы

- Страница курса на сайте machinlearning.ru
- Репозиторий на [github](https://github.com) со всеми материалами
- Сдача домашних заданий в систему Anytask
Инвайт у преподавателя
- Для общения с группой telegram-канал
Всем необходимо зарегаться в telegram!
- Критерии оценивания на странице курса, правила сдачи заданий на странице курса!

Зачем использовать Python?

Особенности Python:

- + Open source
- + Понятность кода
- + Высокая скорость разработки
- + Огромное число библиотек поддержки
- + Универсальность
- Низкая эффективность

Для исследовательского кода эффективность не так важна!

Если какая-то часть кода работает медленно, её можно переписать на другом языке (например, на C).

Реализации Python

Некоторые из реализаций:

- ❶ CPython — основная реализация Python, написанная на C
- ❷ IronPython — реализация, написанная на C# под платформу Microsoft.NET
- ❸ Jython — реализация, написанная на Java
- ❹ CLPython — реализация, написанная на Common Lisp
- ❺ PyPy — ускорение Python за счёт JIT-компиляции
 - Нет полной поддержки некоторых библиотек
- ❻ Stackless Python — разновидность реализации CPython, не использующая стек вызовов языка C

Мы будем использовать CPython

Как Python запускает программы?

Python — не только язык программирования, но и интерпретатор

Традиционная модель выполнения программ на Python:



байт-код \neq машинный код \Rightarrow Python медленнее C и C++

Версии Python

Есть две официальных несовместимых версии Python

Python 2.x:

- Находится в фазе поддержки — не появляется новых фич
- + Всё ещё используется в большинстве IT компаний

Python 3.x:

- + Исправлены многие ошибочные архитектурные решения Python 2.x

Мы будем использовать Python 3.5

Установка Python под Windows

Простой способ:

скачать дистрибутив, содержащий интерпретатор и предустановленные модули, например Anaconda

Сложный способ (зато более гибкий):

- 1 Скачать и установить интерпретатор Python 3 с сайта www.python.org
- 2 Установить нужные пакеты из подборки Unofficial Windows Binaries for Python Extension Packages.
Не перепутайте версию Python и разрядность. В этой подборке NumPy собран с использованием библиотеки Intel MKL, что может в разы ускорить операции линейной алгебры (в Anaconda такая опция доступна лишь за плату).

Установка Python под Mac/Linux

Простой способ (более гибкий):

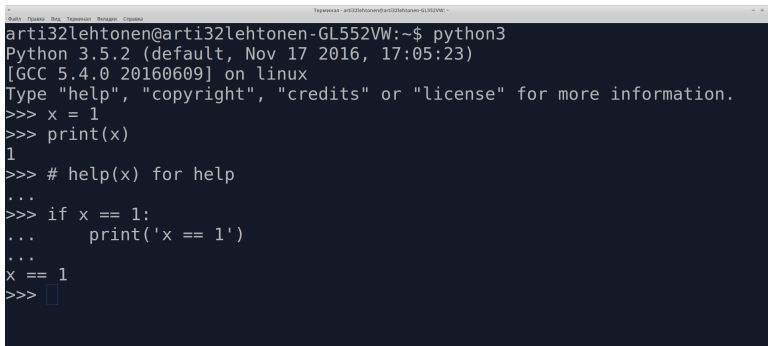
- 1 В большинстве систем Python уже предустановлен. Установить его можно с помощью любого менеджера пакетов.
- 2 Если не установлен, установить pip
- 3 Установить нужные пакеты с помощью pip
`pip install <название пакета>`

Другой простой способ:

скачать дистрибутив, содержащий интерпретатор и предустановленные модули, например Anaconda

Работа с интерпретатором в терминале

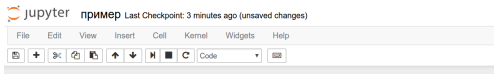
Самый простой способ работы — запуск интерпретатора в терминале

A screenshot of a terminal window titled "Терминал - arti32lehtonen@arti32lehtonen-GL552VW". The terminal shows the command "python3" being executed, which starts the Python 3.5.2 interpreter. The prompt is "arti32lehtonen@arti32lehtonen-GL552VW:~\$". The output shows the Python version and compiler information: "Python 3.5.2 (default, Nov 17 2016, 17:05:23) [GCC 5.4.0 20160609] on linux". The user then enters several commands: "Type 'help', 'copyright', 'credits' or 'license' for more information.", ">>> x = 1", ">>> print(x)", ">>> # help(x) for help", ">>> ...", ">>> if x == 1:", "... print('x == 1')", "...", "x == 1", and ">>> ". The cursor is at the end of the last line.

```
arti32lehtonen@arti32lehtonen-GL552VW:~$ python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 1
>>> print(x)
1
>>> # help(x) for help
...
>>> if x == 1:
...     print('x == 1')
...
x == 1
>>> 
```

Интерактивная среда — Jupyter notebook

Интерактивный «терминал», легко работать с графикой/таблицами, код постоянно сохраняется:



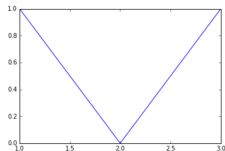
Пример

```
In [1]: 1 x = 1

In [2]: 1 %matplotlib inline
        2 import matplotlib.pyplot as plt

In [3]: 1 plt.plot([1, 2, 3], [1, 0, 1])

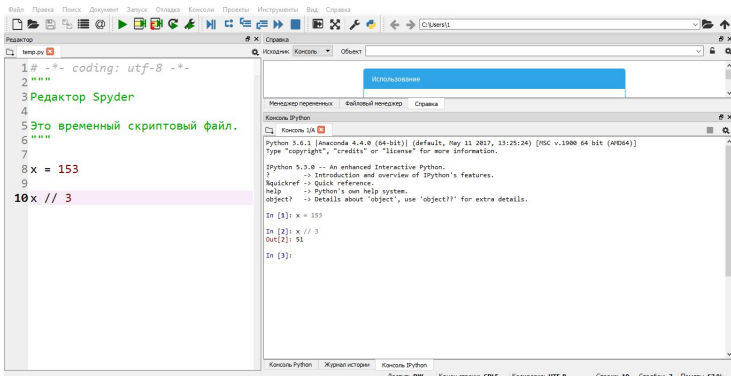
Out[3]: [<matplotlib.lines.Line2D at 0x7f7b67fa0590>]
```



Работа в IDE

- PyCharm (свободно доступна Community Edition)
- Spyder (входит в Anaconda)

Много возможностей: отладчик, автоматическая проверка стиля, встроенный терминал



Динамическая типизация

В Python не требуется объявлять типы переменных:

```
>>> x = 317
>>> type(x)
int
```

Можно делать и так:

```
>>> x = 317
>>> type(x)
int
>>> x = "mmp"
>>> type(x)
str
```

Пример сложной ситуации

Подробно разберём пример программы:

```
>>> x = 317
>>> x = ['m', 'm', 'p']
>>> y = x
>>> x[0] = "c"
>>> x = ['m', 's', 'u']
>>> print(x)
['m', 's', 'u']
>>> print(y)
['c', 'm', 'p']
```

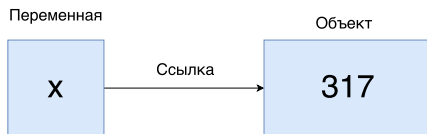
Как это работает?

Создание нового объекта

```
>>> x = 317
```

Что концептуально происходит:

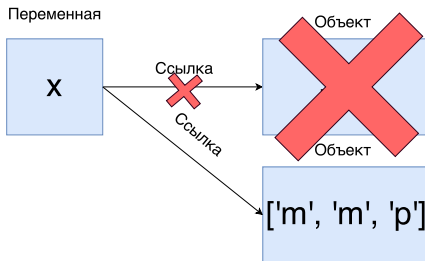
- 1 Создается *объект*, представляющий число 317
- 2 Создается *переменная* *x*, если она еще отсутствует
- 3 В переменную *x* записывается *ссылка* на созданный объект



Изменение ссылки, уничтожение старого объекта

```
>>> x = ['m', 'm', 'p']
```

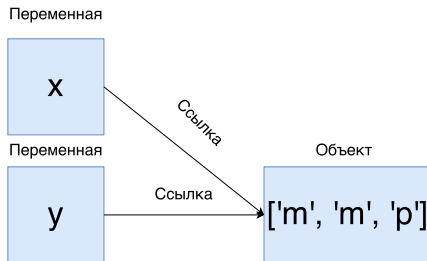
- ④ Создается *объект*, представляющий список `['m', 'm', 'p']`
- ⑤ В переменную `x` записывается *ссылка* на новый объект
- ⑥ Объект, представляющий `317`, уничтожается (сборка мусора)



Новая переменная ссылается на старый объект

```
>>> y = x
```

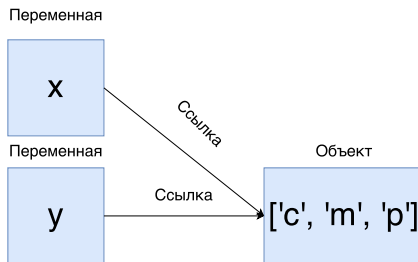
- 7 Создаётся переменная *y*
- 8 В переменную *y* записывается *ссылка* на уже существующий объект



Изменение объекта

```
>>> x[0] = "c"
```

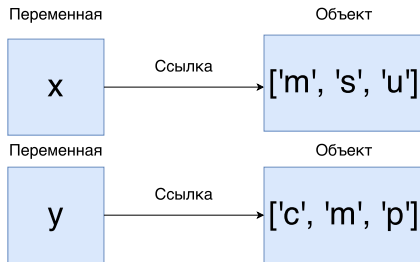
- 9 Меняется содержимое объекта с ['m', 'm', 'p'] на ['c', 'm', 'p']



Объект не уничтожается, если на него что-то ссылается

```
>>> x = ['m', 's', 'u']
```

- 10 Создаётся объект, соответствующий `['m', 's', 'u']`
- 11 В переменную `x` записывается *ссылка* на новый объект
- 12 Старый объект не удаляется, так как на него ссылается `y`



Что надо запомнить?

- 1 Одной переменной в разное время могут соответствовать объекты разных типов
- 2 При присваивании не происходит копирование объекта
- 3 Есть некоторые тонкости, о которых мы поговорим позднее

Список литературы по Python



The Python Tutorial

<https://docs.python.org/3/tutorial/>



Учебник Python 3.1

https://ru.wikibooks.org/wiki/Python/Учебник_Python_3.1



Лутц М. — Изучаем Python (4-е издание и выше)
(легко найти в интернете)



Курс CSC «Программирование на Python» (видеолекции)
<https://compscicenter.ru/courses/python/2015-autumn/classes/>

Список материалов по занятию



Почему существует так много Питонов?

<https://habrahabr.ru/post/209812/>



Беглый обзор внутренностей интерпретатора Python

<https://www.youtube.com/watch?v=zOuxxnUY4lg>



Code Like a Pythonista: Idiomatic Python

<http://python.net/~goodger/projects/pycon/2007/idiomatic/handout.html>