

# Семинар 1.

## Правила игры. Основы Python.

### Машинное обучение, акт 1

программа AIMasters

# Познакомимся: Бугаевский Владимир

## Работа в Data Science:

- ▶ СберМаркет (2022-н.в.)
- ▶ Пульс (VK) Поиск@Mail.Ru (2017-2022)

## Преподавание:

- ▶ лектор, Введение в анализ данных на языке Python, Техносфера (2018-2022)
- ▶ семинарист, Машинное обучение, OzonMasters (2019-2022)
- ▶ лектор, Машинное обучение, Техносфера (2022)
- ▶ семинарист, Машинное обучение, AIMasters (2022-н.в.)
- ▶ лектор, Программирование на Python, ВШЭ (2022-н.в.)

# Познакомимся: Краснов Александр

## Работа в Data Science:

- ▶ Товарные рекомендации Ozon (2021-н.в.)
- ▶ Открытие (2021)
- ▶ Тинькофф (2020)

## Преподавание:

- ▶ семинарист, Машинное обучение, OzonMasters (2021-2022)
- ▶ семинарист, Машинное обучение, AIMasters (2022-н.в.)

# Экосистема курса

- ▶ Сдача заданий в систему `classroom.google.com`  
Регистрация в курсе доступна только с gmail аккаунта (не с университетского или рабочего!)  
Проверьте, что ваши имя и фамилия соответствуют реальным.
- ▶ Автоматическая проверка заданий в системах `ejudge/Яндекс.Контест`
- ▶ Телеграм-канал для всех вопросов и ответов  
Писать ВСЕ вопросы следует в телеграм-канал.

# Виды домашних заданий и правила их сдачи

## Теоретические задания:

- ▶ Решения принимаются только в формате pdf
- ▶ Можно и фотографировать решения, и оформлять на компьютере (например, при помощи системы LaTeX)

## Задания с автоматической проверкой:

- ▶ Сдаются в систему и проверяются автотестами
- ▶ Задача засчитывается, если пройдены все тесты

## Задания на исследование:

- ▶ Не только запрограммировать алгоритм, но и проверить его на некотором наборе данных и сделать выводы
- ▶ Оценивается всё: правильность написанного кода, качество проведённого исследования, адекватность сделанных выводов.

# Дедлайны

- ▶ По всем заданиям — жёсткий дедлайн сдачи!
- ▶ За небольшое опоздание ( $\leq 15$  минут) — штраф 1 балл.
- ▶ За большее опоздание — задание автоматически оценивается в 0 баллов.

Если дедлайн наслаивается на другой, если задание кажется сложным, следует сказать об этом заранее, а не в последний день сдачи.

# Плагиат

При обнаружении плагиата в задании у нескольких студентов баллы за заданием обнуляются всем студентам с найденным плагиатом, независимо от того, кто списал, а кто дал списать.

Плагиатом считается явное заимствование фрагментов кода или текстовых решений.

## Правила оценивания

За курс выставляется оценка по 10-ти балльной шкале. Оценка складывается из вашей работы в семестре.

$M$  — максимальная оценка без учёта бонусов ( $\approx 115$  баллов)

$D$  — ваша оценка за домашние задания

$I$  — итоговая оценка:

$$I = \max(\min(0, \text{round}(18D/M - 6)), 10)$$

Это значит, что за 50% выполненных заданий можно получить минимальную удовлетворительную оценку (3), а за 90% — максимальную (10).

Планируется 3 больших практических задания.



# Научные вычисления (scientific computing)

Научные вычисления — программирование математических моделей для решения прикладных задач.

**Python** — один из основных языков для научных вычислений:

- + Open source
- + Огромное число библиотек, поддерживающих самые различные математические алгоритмы
- + Понятность кода, высокая скорость разработки
- + Универсальность
- Низкая эффективность по сравнению с компилируемыми языками
- Сложен для разработки масштабных проектов

# Что делать с низкой эффективностью Python?

- ▶ Для исследовательского кода эффективность не всегда важна
- ▶ Низкая эффективность частично нивелируется использованием специальных библиотек (например, векторизация вычислений в библиотеке Numpy)
- ▶ Некоторые фрагменты кода можно переписать на другом языке (например, C)

# Реализации Python

Некоторые из реализаций:

1. CPython — основная реализация Python, написанная на C
2. IronPython — реализация, написанная на C# под платформу Microsoft.NET
3. Jython — реализация, написанная на Java
4. CLPython — реализация, написанная на Common Lisp
5. PyPy — ускорение Python за счёт JIT-компиляции
  - Нет полной поддержки некоторых библиотек
6. 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 (компилирующий)

Традиционная модель выполнения программ на Python:



байт-код  $\neq$  машинный код  $\Rightarrow$

1. Python медленнее C и C++
2. Скомпилированная программа платформонезависима

# Версии Python

Вы можете встретить две несовместимых версии Python

## Python 2.x:

- Официальная разработка и поддержка остановлены
- + Всё ещё используется в некоторых IT компаний

## Python 3.x:

- + Активно развивается (версия 3.10.7 вышла в сентябре 2022)
- + Исправлены многие ошибочные архитектурные решения Python 2.x

Задания в систему будут приниматься на Python 3.7.  
Рекомендуется установить именно эту версию.

# Установка Python

**Простой и рекомендуемый способ (для всех ОС).**

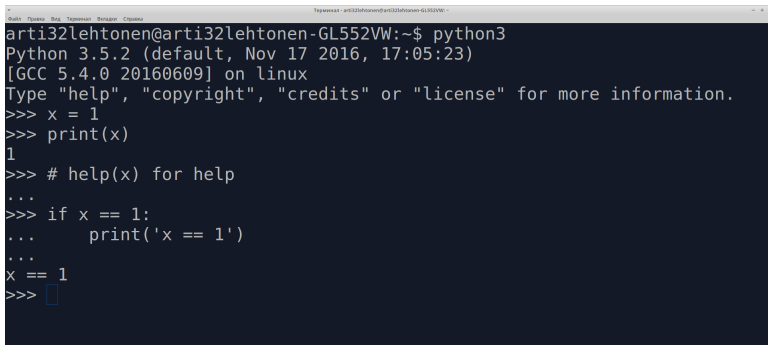
Скачать дистрибутив Anaconda, содержащий интерпретатор и предустановленные модули.

Для установки новых пакетов рекомендуется использовать систему управления пакетами `pip`.

**Для продвинутых:** можно создавать отдельные окружения (своя версия языка, свой набор библиотек) под свои нужды (например, для разных учебных курсов).

# Работа с интерпретатором в терминале

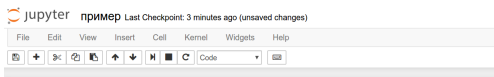
Самый простой способ работы — запуск интерпретатора в терминале (интерактивный режим):

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 interpreter displays its version and build information: "Python 3.5.2 (default, Nov 17 2016, 17:05:23) [GCC 5.4.0 20160609] on linux". It then prompts the user to type "help", "copyright", "credits" or "license" for more information. The user enters ">>> x = 1", and the interpreter responds with ">>> print(x)" and "1". The user then enters ">>> # help(x) for help", and the interpreter responds with "...". The user then enters ">>> if x == 1:", and the interpreter responds with "... print('x == 1')". The user then enters ">>> x == 1", and the interpreter responds with "...". Finally, the user enters ">>> ", and the interpreter responds with a prompt character "□".

```
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», нечто среднее между интерактивным режимом и IDE. Позволяет легко работать с графикой/таблицами, код постоянно сохраняется:



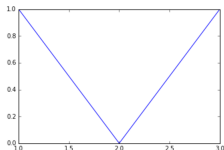
## Пример

```
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>]
```

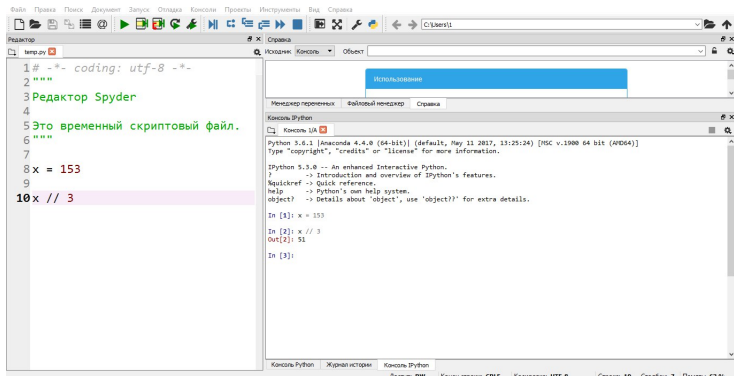




# Работа в IDE

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

Много возможностей: отладчик, автоматическая проверка стиля, встроенный терминал.



# Список литературы по Python

Рекомендуется всем начинающим ознакомиться с учебником Лутца до 7 части включительно.



The Python Tutorial

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



Учебник Python 3.1

[https://ru.wikibooks.org/wiki/Python/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



Стайлгайд языка Python PEP8

<https://www.python.org/dev/peps/pep-0008/>

<https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html>



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>