

Занятие 1.

Правила игры. Основы Python.

Практикум на ЭВМ, осень 2021

Находнов Максим Сергеевич

кафедра ММП, ВМК МГУ

02 сентября 2021

Преподаватели курса

- ▶ Находнов Максим Сергеевич
- ▶ Кропотов Дмитрий Александрович
- ▶ Бобров Евгений Александрович
- ▶ Чернышёв Александр Владиславович
- ▶ Охотин Андрей Сергеевич
- ▶ Тыцкий Владислав Игоревич
- ▶ Елистратов Семен Юрьевич
- ▶ Афанасьев Глеб Ильич
- ▶ Васильев Руслан Леонидович
- ▶ Никоров Кирилл Николаевич

Экосистема курса

- ▶ Страница курса на github: ссылка
- ▶ Сдача заданий в систему anytask: ссылка
При регистрации указывайте ваши реальные имя и фамилию!
- ▶ Автоматическая проверка заданий в системах ejudge/яндекс.контекст
- ▶ Телеграм-чат для всех вопросов и ответов
Писать ВСЕ вопросы следует в телеграм-чат.

О чём наш курс?

- ▶ Программирование на языке Python. Numpy,...
- ▶ Реализация базовых алгоритмов машинного обучения
- ▶ Использование системы LaTeX для вёрстки документов
- ▶ Проведение исследований и написание отчётов по их результатам
- ▶ Базовые навыки создания индустриальных ML систем. Git, Docker, Flask

Виды домашних заданий и правила их сдачи

Задания с автоматической проверкой (5 контестов):

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

Задания на исследование (2 задания):

- ▶ 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. Отдельно оценивается качество оформления отчёта.

Небольшой финальный проект (1 проект):

- ▶ Всё, что было в предыдущем пункте + реализовать небольшую ML-систему для решения реальной задачи

Дедлайны

- ▶ По всем заданиям на автопроверку — жёсткий дедлайн через неделю после выдачи задания
- ▶ По всем остальным заданиям — две недели на сдачу без штрафа. Затем неделя после мягкого дедлайна со штрафом 3 балла в день. Затем жёсткий дедлайн.

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

Плагии

При обнаружении плагии 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

Форма курса — зачёт с оценкой. Оценка складывается из вашей работы в семестре.

- ▶ 8-26 баллов за каждый из контестов
- ▶ 50 баллов за «большие» задания
- ▶ Сумма баллов ≈ 231 балл

Предварительные критерии оценки:

- ▶ отлично — 185 баллов ($\approx 80\%$),
3 «больших» задания зачтены
- ▶ хорошо — 139 балла ($\approx 60\%$),
2 «больших» задания зачтены
- ▶ удовлетворительно — 93 балла ($\approx 40\%$),
1 «большое» задание зачтено

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, автоматическое распараллеливание с помощью Numba)
- ▶ Некоторые фрагменты кода можно переписать на другом языке (например, C)

Реализации Python

Python == язык + интерпретатор.

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 выходит в октябре 2021)
- + Исправлены многие ошибочные архитектурные решения Python 2.x

Задания в систему будут приниматься на Python 3.6.

Рекомендуется установить именно эту версию.

Установка Python

Простой и рекомендуемый способ (для всех ОС).

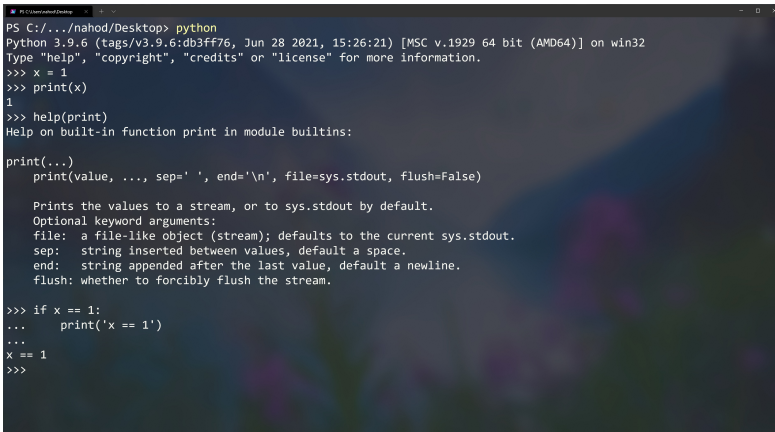
Скачать дистрибутив Anaconda, содержащий интерпретатор и предустановленные модули.

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):



```
PS C:\Users\johnd\Desktop> python
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 1
>>> print(x)
1
>>> help(print)
Help on built-in function print in module builtins:

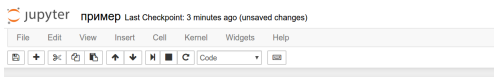
print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep: string inserted between values, default a space.
    end: string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.

>>> 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/динамическим содержимым, код постоянно сохраняется:



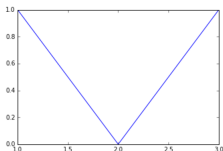
Пример

```
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])

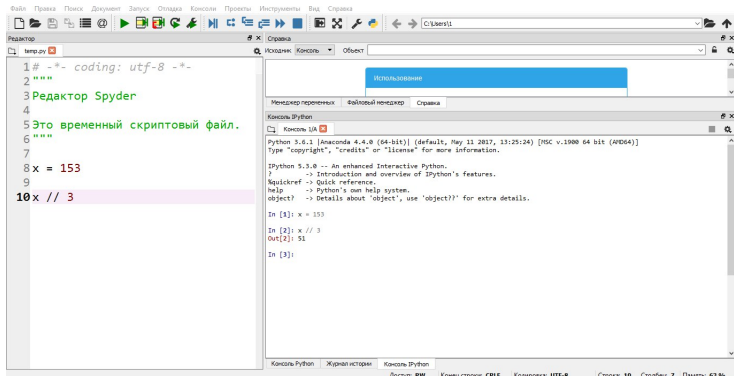
Out[3]: [<matplotlib.lines.Line2D at 0x7f7b67fa0590>]
```



Работа в IDE

- ▶ PyCharm (свободно доступна Community Edition)
- ▶ VSCode (свободно доступна)
- ▶ 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



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>