Компьютерная математика II

Дисциплина для студентов 1-го курса специальности «Компьютерная математика и системный анализ» и профилизации "Искусственный интеллект и математическая экономика" ММФ БГУ

Тема 1. Язык Python. Синтаксические особенности. Среда разработки JupyterLab

доц. Лаврова О.А., кафедра дифференциальных уравнений и системного анализа (ауд. 329)

февраль, 2025

Организация учебного процесса

- 1 лекция в неделю (среда 9:45, ауд. 606)
- 1 лабораторное занятие в неделю
- консультация (среда 14:30, 16:00)

Формы контроля знаний:

- Отчет по лабораторной работе с устной защитой — 10-11
- Управляемая самостоятельная работа — 2
- Письменный опрос — 3-4

<отметка текущей аттестации> = 0.4 <самостоятельные работы/письменные опросы> + 0.6 <защита ЛБ>

Пропуски занятий по неуважительной причине также учитываются при выставлении отметки текущей успеваемости.

Форма текущей аттестации: зачет и экзамен.

<итоговая отметка> = 0.4 <отметка текущей аттестации> + 0.6 <отметка на экзамене>

Рекомендуемая литература

1. Лутц М., **Изучаем Python**, 2 тома
2. Лутц М., **Программируем на Python**, 2 тома
3. Бизли Д. **Python. Исчерпывающее руководство** — СПб.: Питер, 2023.
4. Майер К. **Однострочники Python: лаконичный и содержательный код** — СПб.: Питер, 2022.

5. Мэтиз Э. **Изучаем Python: программирование игр, визуализация данных, веб-приложения** — СПб.: Питер, 2020.
6. Любанович Б. **Простой Python. Современный стиль программирования** (2-е изд.) — СПб.: Питер, 2021.
7. Хиллард Д. **Секреты Python Pro** — СПб.: Питер, 2021.
8. Rougier N., **Scientific Visualization: Python + Matplotlib**, 2021 https://github.com/rougier/scientific-visualization-book
9. Rougier N., **From Python to Numpy**, 2017 https://www.labri.fr/perso/nrougier/from-python-to-numpy/
10. Стандарты стиля кодирования на Python PEP 8, 2001 (изменения в 2013) https://www.python.org/dev/peps/pep-0008

1.1 Основные характеристики языка программирования Python

Создателем языка Python является голландский программист **Гвидо Ван Россум** (Guido van Rossum).

Язык Python назван в честь британской комик-группы "Монти Пайтон" (Monty Python).

Первая версия языка Python 0.9.0 появилась в 1991 году (C в 1972, C++ в 1983, Java в 1995). Текущая версия 3.13.1 (декабрь 2024). Поддержка версии Python 2.x закончилась в 2020 году, тем не менее остается популярной версия Python 2.7.

In [11]:
```
import sys
sys.version
```

Out[11]:
```
'3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.192
9 64 bit (AMD64)]'
```

Python является одним из наиболее широко используемых языков программирования на сегодняшний день:

RedMonk метрика, июнь 2024: JavaScript, **Python**, Java, PHP, C#, TypeScript

TIOBE метрика, январь 2025: **Python**, C++, Java, C, C#

PYPL метрика, январь 2025: **Python**, Java, JavaScript, C/C++, C#

Первая книга про Python в 1996; >7000 на amazon.com (январь 2025).

Python является бесплатным языком программирования *с открытым кодом*

Python является высокоуровневым языком программирования

Python поддерживает несколько парадигм программирования

Процедурная парадигма: программирование на основе использования операторов

Функциональная парадигма: программирование на основе комбинирования функций без побочных эффектов

Объектно-ориентированная парадигма: программирование с использованием пользовательских классов, экземпляров классов

Обширная функциональность

Вместе с базовым языком Python поставляется большая коллекция модулей, которая называется **стандартной библиотекой**. Стандартная библиотека состоит из более чем 300 модулей. Программирование прикладных задач осуществляется с применением инструментов из *стандартной библиотеки*.

Существует множество дополнительных модулей и расширений языка, не входящих в стандартную библиотеку. Например, `sympy` , `numpy` , `matplotlib` и др.

Python является интерпретируемым языком программирования

Код, написанный на Python, *транслируется* в байт-код, а затем выполняется интерпретатором. Этап компиляции исходного кода до машинного кода, например как в C и C++, отсутствует.

Кроссплатформенность

Большинство программ Python выполняются без изменений на всех основных компьютерных платформах.

Автоматическое управление памятью

Python *автоматически* выделяет память под объекты и *автоматически* очищает память, когда объекты больше не используются (на объекты нет ссылок). Сборщик мусора работает, отслеживая ссылки на объекты в памяти, используя механизм подсчета ссылок. Каждый раз, когда создается новая ссылка на объект, счетчик ссылок для этого объекта увеличивается. Точно так же, когда ссылка удаляется, счетчик ссылок уменьшается. Python очищает неиспользуемую память во время выполнения программы.

Код, необходимый для управления памятью (выделение памяти, очищение памяти), существующий в таких языках, как C и C++, отсутствует в языке Python.

Python ориентирован на высокую скорость разработки

Код Python обычно занимает от 1/3 до 1/4 части размера эквивалентного кода на C++ или Java за счет, в частности, простого синтаксиса, автоматического управления памятью, динамической типизации переменных.

Язык Python оптимизирован для скорости разработки (development speed), а не для скорости выполнения кода (runtime speed).

Недостаток языка Python: скорость выполнения кода

Скорость выполнения кода на Python может не всегда быть такой же, как у полностью компилируемых языков, так как Python является интерпретируемым языком программирования. Однако выигрыш в скорости разработки, обеспечиваемый Python, зачастую важнее, чем потери в скорости выполнения кода.

1.2 Применение языка программирования Python

Python является языком общего назначения

Python задействован в разных предметных областях

- системное программирование
- программирование GUI
- программирование для баз данных
- в веб-программировании на Python пишут в основном бэкенд
- в программировании для мобильных устройств Python используется редко

- **численное и научное программирование**
  - визуализация данных
  - обработка изображений
  - обработка естественного языка
  - анализ данных
  - разработка алгоритмов искусственного интеллекта
  - разработка компьютерных игр

Цель дисциплины "Компьютерная математика"

Изучить основы программирования на Python (базовый Python + `numpy` + `matplotlib`) для проведения исследований (*exploratory programming*) в математике, для численного и научного программирования.

Python в реальных продуктах

- поисковая система *Google*
- служба *YouTube* почти полностью написана на Python
- серверная часть *Instagram* написана на Python
- служба хранилища *Dropbox* почти полностью написана на Python
- в производстве анимационных фильмов *Pixar*
- серверная часть *Netflix*

1.3 Структура программы на Python

Концептуальная иерархия программы

Программа на Python состоит из *модулей*. Код программы всегда находится внутри какого-то модуля.

Один из модулей предназначен быть главным файлом либо *файлом верхнего уровня*, или сценарием — файлом, запускаемым для старта программы, которая выполняется строка за строкой обычным образом. Модуль верхнего уровня является основным документом программы.

Модуль — это текстовый файл, который состоит из *выражений* и *операторов*. Операторы и выражения используются для создания и обработки *объектов*. Группируя операторы и выражения специальным образом, можно создавать *функции* и *классы* для эффективного программирования.

Байт-код. Виртуальная машина Python

Код, написанный на Python, *транслируется* в байт-код, а затем выполняется интерпретатором. Этап компиляции исходного кода до машинного кода отсутствует.

Байт-код — низкоуровневое и независимое от платформы представление исходного кода, специфичное для Python. Интерпретатором байт-кода является **виртуальная машина Python** (Python Virtual Machine) для различных реализаций Python: CPython, PyPy, Jython, Pyston, Pyjion, Cinder, GraalIPython и др.

Модуль, написанный на Python, имеет расширение *.py*. Байт-код сохраняется в файлах с расширением *.pyc*.

Имена файлов с *интерактивными* документами Jupyter Notebook имеют расширение *.ipynb*. Файлы имеют формат json. Блокнот Jupyter Notebook всегда является модулем верхнего уровня.

Способы выполнения программ, написанных на Python

Anaconda — это наиболее широко используемый дистрибутив Python; поставляется с предустановленными наиболее популярными модулями и пакетами Python; содержит установщик *conda* для модулей и пакетов; содержит среду разработки **JupyterLab**.

PyCharm — очень популярная кроссплатформенная среда разработки для программ на Python.

Colab — это бесплатная веб-IDE для работы с блокнотами Jupyter Notebook, которая работает в облаке и хранит блокноты на Google Диске.

1.4 Синтаксис Python: особенности и отличия от других языков программирования

Особенность языка Python — это концентрация на *читабельности* кода, которая реализуется за счет синтаксиса языка. Синтаксис языка Python является простым, ясным и традиционным. Python считается самым легким в обучении языком программирования.

1. Базовый язык Python основан на использовании зарезервированных слов, которые являются частью синтаксиса. Они все приведены в модуле `builtins`

In [57]:
```
import builtins

# количество зарезервированных слов
len(dir(builtins))
```

Out[57]: 161

2. Python является динамически типизируемым языком программирования. Объявление типа для переменной перед началом ее использования не требуется. Понятие типа связано с объектом, на который ссылается переменная, а не с переменной. Типы объектов отслеживаются *во время выполнения кода*.

3. Выравнивание кода в соответствии с его структурой с применением отступа (например, 4 пробела или табуляция в начале строки кода) является частью синтаксиса языка Python для обозначения блока кода (например, тела оператора). Эта особенность языка Python заимствована из языка ABC.

In [60]:
```
if 'KM2':
    print(True)
else:
    print(False)
```

```
True
```

Совокупный эффект заключается в том, что код на Python становится более согласованным и читабельным за счет формирования кодовой структуры. В известной степени Python является языком WYSIWYG (what you see is what you get — что видишь, то и получаешь), поскольку внешний вид кода определяет его структуру.

Тело оператора выделяется только отступом, дополнительного синтаксиса не нужно:

In [63]:
```
if True:
    s1 = 'KM' + '2'
    print(s1)
```

```
KM2
```

4. Символ двоеточие (:) является частью синтаксиса языка Python.

5. Символ ; в конце строки кода не обязателен.

6. Круглые скобки вокруг выражений проверки не обязательны

In [67]:
```
if 1!=1+1.e-16:
    print(True)
else:
    print(False)
```

```
False
```

Рекомендации:

1. Для обучения синтаксису языка набирайте код, не копируя его.

2. Пишите код, заботясь в первую очередь о читабельности кода. Читабельность кода важнее, чем его краткость.

3. При написании кода следуйте руководству по стандартам стиля кодирования на Python **PEP 8**, см. https://www.python.org/dev/peps/pep-0008/ (2001, изменения в 2013).

Язык Python предлагает основные принципы написания кода (всего 19 принципов).Принципы подчеркивают важность простоты кода, ясности кода, удобочитаемости кода и согласованности кода.

In [73]:
```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

1.5 Среда разработки JupyterLab

JupyterLab — это веб-приложение, которое реализует среду разработки (IDE) с интерфейсом в отдельной вкладке браузера.

JupyterLab — это свободное программное обеспечение с открытым кодом из проекта Jupyter. Первая стабильная версия JupyterLab вышла в 2018 году.

URL-адрес JupyterLab похож на http://localhost:8888/lab. *Localhost* не является веб-сайтом, но указывает, что контент обслуживается с локального компьютера.

JupyterLab используется для работы с несколькими блокнотами Jupyter Notebook, с кодами и данными.

Рабочая область JupyterLab разбита на вкладки, в которых могут быть расположены: несколько редакторов для блокнотов Jupyter Notebook, браузер файлов, текстовые редакторы, терминалы командной строки, консоль для кода, средства просмотра файлов данных и другие компоненты.

JupyterLab поддерживает отображение и редактирование множества форматов данных: изображений, CSV, JSON, PDF и др.

Консоль (Console) можно использовать как место для интерактивной проверки различных небольших идей без изменения блокнота Jupyter Notebook.

Терминал командной строки (Terminal) запускает терминал вашей операционной системы в браузере.

Блокнот *Jupyter Notebook*

Блокнот **Jupyter Notebook** представляет собой *интерактивную вычислительную среду*, которая позволяет объединить код на Python с результатами выполнения кода, с текстом (встроенный язык разметки Markdown), с математическими формулами (встроенный язык разметки Latex), с визуализацией, с интерактивными элементами управления и др.

Блокнот Jupyter Notebook всегда является файлом верхнего уровня (основным документом), который использует инструменты из импортируемых модулей.

Интерфейс блокнота Jupyter Notebook заимствован из Mathematica.

Режимы работы с ячейками, типы ячеек

Блокнот Jupyter Notebook состоит из чередующихся ячеек. Все действия в блокноте Jupyter Notebook выполняются в какой-то из ячеек.

Одна из ячеек является *текущей ячейкой* или активной. Она выделяется слева специальным маркером в виде синей вертикальной полоски.

Существует два режима для работы с ячейкой: *режим редактирования* (**Edit mode**) и *командный режим* (**Command mode**).

Для перехода в *режим редактирования* необходимо дважды щелкнуть мышью по текущей ячейке или нажать клавишу ввода Enter. В этом случае внутри ячейки появится курсор, вокруг ячейки появится синяя рамка.

Для перехода в *командный режим* необходимо нажать клавишу Esc.

Для добавления ячейки перед текущей используйте клавишу a в *командном режиме*.

Для добавления ячейки после текущей используйте клавишу b в *командном режиме*.

Удалить ячейку можно дважды нажав клавишу d, d в *командном режиме*.

В *командном режиме* копирование ячейки — клавиша C, вставка ячейки — клавиша V, вырезание ячейки — клавиша X.

Смотрите подменю Edit горизонтального меню JupyterLab для других возможностей редактирования ячеек в блокноте Jupyter Notebook.

Ячейки могуть быть двух типов: *ячейка кода* (**Code cells**) и *текстовая ячейка* (**Markdown cells**). По умолчанию создается *ячейка кода*.

Ячейка кода содержит произвольное количество строк кода, который будет выполняться в ядре. Вывод результатов выполнения кода будет отображаться под ячейкой кода.

Текстовая ячейка содержит текст, отформатированный с использованием языков разметки Markdown и Latex. Вывод результатов выполнения ячейки отображается на месте.

Для автозавершения в *ячейке кода* используйте клавишу Tab *в режиме редактирования*.

Выбранная *ячейка кода* или *текстовая ячейка* запускается с помощью комбинации клавиш Shift+Enter. При запуске ячейки кода код из ячейки отправляется в ядро для выполнения в интерпретаторе Python. Затем курсор перемещается в ячейку, расположенную ниже. Если ниже нет ячейки, то создается новая пустая ячейка.

Смотрите подменю Run горизонтального меню JupyterLab для других возможностей выполнения ячеек в блокноте Jupyter Notebook.

Для изменения типа текущей ячейки на *текстовую ячейку* используйте клавишу m в *командном режиме*.

Для изменения типа текущей ячейки на *ячейку кода* используйте клавишу y в *командном режиме*.

Языки разметки для задания форматирования в тексте

При выполнении *текстовые ячейки* преобразуются в текст в формате HTML, применяя языки разметки **Markdown** и **Latex**.

Markdown — это язык разметки для форматирования простого текста.

Используйте символ решетка \# перед текстом для создания заголовков.

Используйте символ звездочка * *вокруг текста для выделения его курсивом*.

Используйте двойные звездочки ** **вокруг текста для выделения его жирным шрифтом**.

- Используйте символ тире с пробелом - или символ плюс с пробелом + или символ звездочка с пробелом * перед текстом для создания ненумерованного списка

Смотрите подменю Help | Markdown Reference горизонтального меню для других возможностей форматирования простого текста.

Latex — это язык разметки для форматирования математического текста.

Используйте символ доллара $ вокруг текста для создания математических формул в синтаксисе Latex: $\frac{1}{2} + \sqrt{2} - x^2$.

Используйте двойные символы доллара $$ вокруг текста для создания выносных математических формул в синтаксисе Latex:

$$\frac{1}{2} + \sqrt{2} - x^2.$$

Ядро Python

Код, содержащийся в блокнотах Jupyter Notebook, выполняется в ядре. Полученные результаты вычислений ядро передает обратно в браузер. Для каждого блокнота Jupyter Notebook запускается собственное ядро.

Существует около 50 JupyterLab-совместимых ядер для многих языков программирования, включая Python, R, Julia и Haskell. Например, ядро SoS обеспечивает многоязычную поддержку в пределах одного блокнота Jupyter Notebook.

Ядро можно перезапустить или остановить, если возникли проблемы выполнения кода.

Смотрите подменю Kernel горизонтального меню для других возможностей управления ядром.

При закрытии вкладки с блокнотом Jupyter Notebook, консоли или терминала, соответствующее ядро или терминал продолжают работу в фоновом режиме.

Смотрите вкладку Running... вертикального меню для управления запущенными ядрами и открытыми терминалами.

JupyterLab предлагает специальные команды для расширения возможностей программирования. Специальные команды начинаются с символа `%` для выполнения в строке, и с символов `%%` для выполнения в ячейке.

Например, команда `%timeit` измеряет время выполнения однострочного выражения

In [99]:
```
%timeit x = [i**2 for i in range(10)]
```

```
2.34 µs ± 120 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
```

In [100…
```
%%time
import time
x = [time.sleep(0.1) for i in range(10)]
```

```
CPU times: total: 0 ns
Wall time: 1.01 s
```

Справочную информацию о любом объекте Python можно получить с помощью средств оболочки блокнота Jupyter Notebook, вводя символ знака вопроса `?` до или после имени объекта:

In [102…

```
?range
```

```
Init signature: range(self, /, *args, **kwargs)
Docstring:
range(stop) -> range object
range(start, stop[, step]) -> range object

Return an object that produces a sequence of integers from start (inclusive)
to stop (exclusive) by step.  range(i, j) produces i, i+1, i+2, ..., j-1.
start defaults to 0, and stop is omitted!  range(4) produces 0, 1, 2, 3.
These are exactly the valid indices for a list of 4 elements.
When step is given, it specifies the increment (or decrement).
Type:           type
Subclasses:
```