

Лекция 5 NumPy

Давыдов Виталий Валерьевич (ГАИШ МГУ, Постгрес-ПРО)

Проект Jupyter (jupyter.org)

- Инструменты и стандарты для интерактивных вычислений с использованием интерактивных блокнотов (computational notebooks)
- Computational Notebook – документ, включающий компьютерный код, данные, визуализации (изображения и графики), интерактивные элементы управления, форматированный текст (формулы).
- Включает множество проектов
 - Jupyter User Interfaces (JupyterLab, Jupyter Notebook, Jupyter Desktop, Jupyter Console...)
 - Kernels (процессы, реализующие интерпретаторы языков программирования)
 - IPython
 - <https://jupyter4edu.github.io/jupyter-edu-book/>
- <https://nbviewer.org/>

Computational Notebook

- Последовательность ячеек (cells)
- Есть различные типы ячеек:
 - Code (исполняемый код)
 - Markdown (форматированный текст)
 - Raw (неформатированный текст)
- Код выполняется в Kernel
 - Результат отображается в ячейке под кодом

```
[5]: import matplotlib.pyplot as plt
plt.style.use('classic')
%matplotlib inline
import numpy as np
import pandas as pd
import seaborn as sns
sns.set()
```

```
[6]: rng = np.random.RandomState(0)
x = np.linspace(0, 10, 500)
y = np.cumsum(rng.randn(500, 6), 0)
```

Next step

Now, create a graph.

```
[7]: plt.plot(x, y)
plt.legend('ABCDEF', ncol=2, loc='upper left');
```



Возможности Jupyter

- Отображение кода и результата в одном окне с возможностью интерактивного редактирования кода
- Отображение визуализаций (графиков, изображений), интерактивных элементов управления
- Форматированный текст (markdown) и структурирование
- Отображение формул (latex)
- Экспорт в другие форматы (pdf, html)
- Документация:
 - <https://jupyterlab.readthedocs.io>
 - <https://jupyter-notebook.readthedocs.io/>

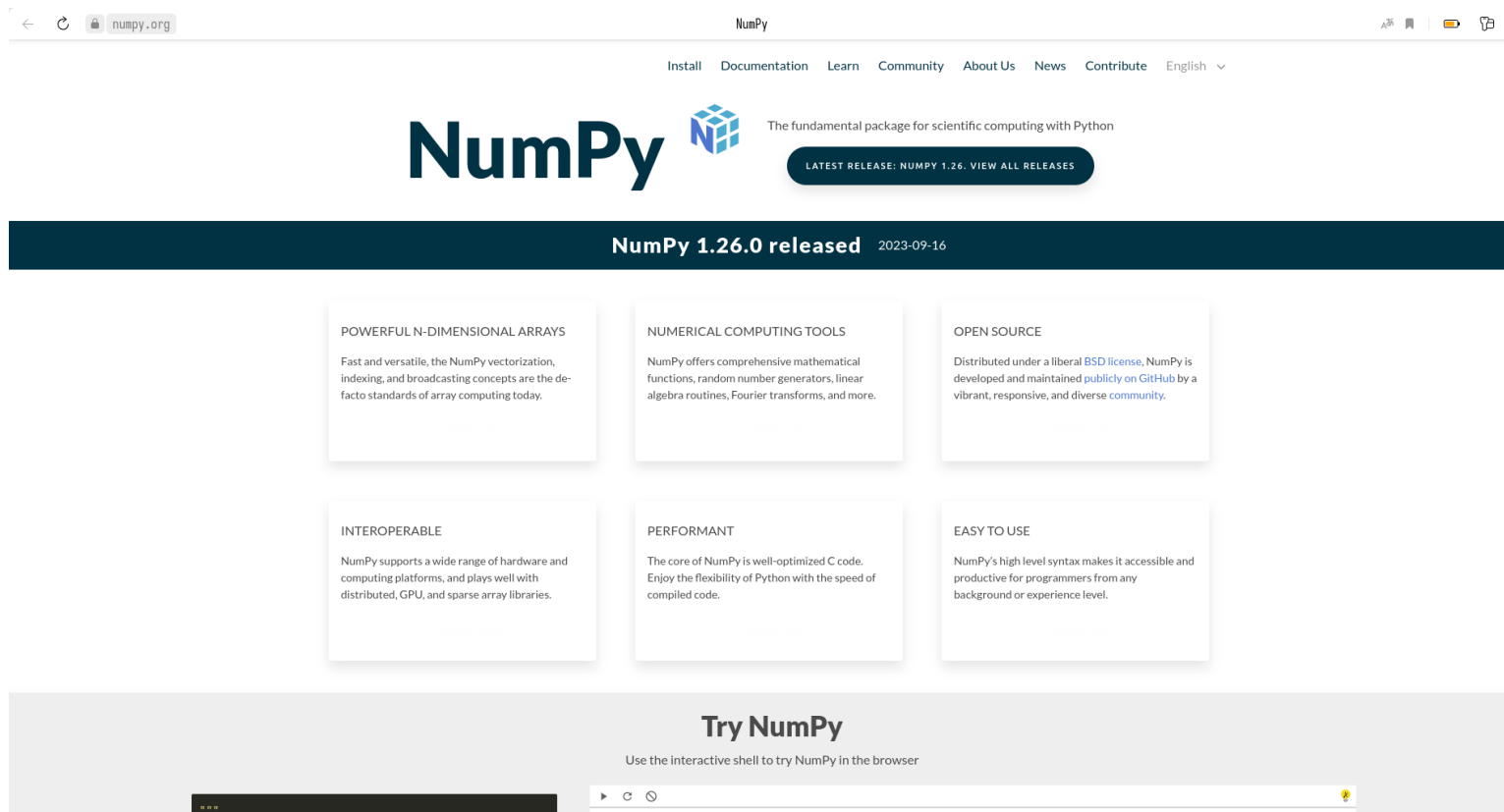
Полезные фишки

- **Клавиатурные комбинации:**
 - Выполнить одну ячейку : CTRL + ENTER
 - Добавить новую ячейку: ESC a (above), ESC b (below)
 - Удалить текущую ячейку: ESC d d
- Тайминг выполнения одной строки, ячейки: `%timeit`, `%%timeit`
- Выполнить скрипт: `%%bash`
- Исполнить ячейку как latex, html: `%%latex`, `%%html`
- Установить точность вывода FP (не в print): `%precision`
- Информация по "магическим" командам: `%magic`

Пакет NumPy

- **Библиотека для работы с многомерными массивами**
 - Многомерный массив (`numpy.ndarray`)
 - Новые скалярные типы, расширяющие систему типов Python (знаковые и беззнаковые)
 - Masked array (`numpy.ma`)
 - Типы для работы с временем (`numpy.datetime64`)
- **Набор операций над массивами**
 - Создание и изменение многомерных массивов
 - Математические функции
 - Загрузка и сохранение данных на диск
 - Операции линейной алгебры
 - Решение линейных уравнений
 - Операции с маскированными массивами
 - Математические функции (применяемые поэлементно)
 - Полиномы
 - Статистики
 - Дискретное Фурье-преобразование

Сайт numpy.org



The screenshot shows the NumPy website homepage. At the top, there's a navigation bar with links: Install, Documentation, Learn, Community, About Us, News, Contribute, and a language dropdown set to English. The main header features the NumPy logo (a blue cube) and the tagline "The fundamental package for scientific computing with Python". Below this, a dark blue banner announces "NumPy 1.26.0 released" with the date "2023-09-16". A button next to it says "LATEST RELEASE: NUMPY 1.26. VIEW ALL RELEASES". The page is divided into six feature boxes arranged in a 2x3 grid:

- POWERFUL N-DIMENSIONAL ARRAYS**: Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.
- NUMERICAL COMPUTING TOOLS**: NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.
- OPEN SOURCE**: Distributed under a liberal [BSD license](#), NumPy is developed and maintained [publicly on GitHub](#) by a vibrant, responsive, and diverse [community](#).
- INTEROPERABLE**: NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.
- PERFORMANT**: The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.
- EASY TO USE**: NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

At the bottom, a grey section titled "Try NumPy" invites users to "Use the interactive shell to try NumPy in the browser". Below the text is a simulated terminal window with a dark background and a light-colored prompt.

Скалярные типы

Basic Type	Available NumPy types	Comments
Boolean	<code>bool</code>	Elements are 1 byte in size
Integer	<code>int8, int16, int32, int64, int128, int</code>	<code>int</code> defaults to the size of <code>int</code> in C for the platform
Unsigned Integer	<code>uint8, uint16, uint32, uint64, uint128, uint</code>	<code>uint</code> defaults to the size of unsigned <code>int</code> in C for the platform
Float	<code>float32, float64, float, longfloat,</code>	Float is always a double precision floating point value (64 bits). <code>longfloat</code> represents large precision floats. Its size is platform dependent.
Complex	<code>complex64, complex128, complex</code>	The real and complex elements of a <code>complex64</code> are each represented by a single precision (32 bit) value for a total size of 64 bits.
Strings	<code>str, unicode</code>	Unicode is always UTF32 (UCS4)
Object	<code>object</code>	Represent items in array as Python objects.
Records	<code>void</code>	Used for arbitrary data structures in record arrays.

Многомерный массив (пример)

Создание многомерного массива из последовательностей Python:

```
np.array([1, 2, 3, 4, 5])
```

– одномерный массив

```
np.array([[11, 12], [21, 22]])
```

– двухмерный массив

```
np.array(range(100))
```

– одномерный массив из генератора

Операции над массивами

- Операции выполняются поэлементно
- Массивы можно передавать в математические функции `numpy`

numpy.ndarray

- **Многомерный массив значений одного типа**
 - Размер задается во время создания
 - Изменение размера приведет к удалению старого массива и созданию нового
 - Поэлементное изменение данных массива
 - Элементы массива одного типа (размера)
 - Многие операции реализованы в скомпилированном коде
 - Поддержка больших массивов
 - Другие научные пакеты для Python используют массивы numpy
 - Расширенный диалект языка Python для работы с массивами
- **Основные свойства**
 - dtype (data type, тип значений)
 - shape (форма, количество элементов по каждому измерению)
 - ndim (количество измерений)
 - size (общее количество элементов)
 - base (ссылка на исходные данные для представлений или None)
- **Полный список свойств и методов ndarray:**
 - <https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html>

Создание многомерных массивов

- **Задание формы (shape) массива (N-Dim)**

- `np.empty(10,10)`, `np.empty([10, 10], dtype=np.float64)` – неинициализированный массив
- `np.zeros(...)` – массив с нулями
- `np.ones(...)` – массив из единиц

- **Из последовательностей Python (1-Dim)**

- `np.array(<list>)`, `np.array(<tuple>)`

- **С использованием генераторов (1-Dim)**

- `np.arange([start,] stop[, step,][, dtype, like])`
- `linspace(start, stop[, num, endpoint, ...])`

- **Из внешних источников**

- `np.fromfile(...)`, `np.fromfunction(...)`, `np.fromiter(...)`, `np.fromstring`

- **Матрицы (2-Dim)**

- `np.diag`, `np.tri`, `np.vander`
- `np.mat(data, dtype)`

Доступ к элементу по индексу

- **Индекс элемента – тапл, содержащий позицию по каждой оси**
 - Общая форма: `X[<obj>]`, где `obj` – `list`, `tuple`, `slice` (срез), генератор и др
- **Одномерные массивы**
 - `x[10]` – элемент по индексу 10
 - `x[(10)]`
- **Двухмерные массивы**
 - `x[0]` – возвращает срез массива по оси 0 (`view`)
 - `x[1,3]` – возвращает значение элемента
 - `x[[0,0), (0,1), (10,10)]]`
- **Многомерные массивы**
 - `x[1,2,3,4,5]` – в случае 5 осей доступ к элементу (1,2,3,4,5)
- **Нет операции удаления элемента**

Срезы

- **Обобщенный синтаксис: `x[obj]`**
- **Расширяет концепцию срезов Python на N измерений**
- **Простое индексирование (`obj` – tuple)**
 - Срез по одному измерению: `x[begin:end:step]` (также, как и для `list`)
 - Срез по двум двух измерениям: `x[b1:e1:s1, b2:e2:s2]`
 - Ellipsis: `x` – 3d-массив, тогда `x[..., 1:3] ↔ x[:, :, 1:3]`
 - `newaxis` – добавление нового измерения
 - Результат простого индексирования – `views` (а не копии!)
- **Продвинутое (`advanced`) индексирование (`obj` – не tuple)**
 - Целочисленные последовательности: `x[[1, 1, 4, 4, 5]]` – выбор произвольных элементов
 - Булевы последовательности: `x[[True, False, True]]` – выбор элементов с `True` (по маске)

Срезы (продолжение)

- **Фильтрация элементов**

- `x = np.array([1., 2., np.nan, 3., np.nan, np.nan])`
- `y = x[~np.isnan(x)]`

- **Операции по условию**

- `x = np.array([1., -1., -2., 3])`
- `x[x < 0] += 100`

Представления (views)

- `b = a[1:5]`, где `b.base` \neq `None` (`b` – представление)
- `b = a[1:5].copy()`, где `b.base` `=` `None` (копия)
- Представления разделяют тот же блок памяти
- Могут иметь `shape`, отличный от оригинала
- Используются для уменьшения расходования памяти

Broadcasting

- Набор правил для выполнения арифметических операций с массивами разных размерностей
- Массив с меньшей размерностью "расширяется" до размерности другого массива
- Правила бродкастинга:
 - Перед выполнением бинарной операции идет сравнение размерностей двух массивов
 - Размерности сравнимы, если: (1) они одинаковы, (2) одна из них равна 1
 - Если размерности не сравнимы, выдается ошибка `ValueError: operands could not be broadcast together`
 - К-во размерностей результата – к-во размерностей "большого" массива

```
» a = np.array([1.0, 2.0, 3.0])
» b = 2.
» a * b
array([2., 4., 6.])
```

```
Image (3d array): 256 x 256 x 3
Scale (1d array): 3
Result (3d array): 256 x 256 x 3
```

Пример: Линейный МНК

- Практическая работа в JuPyter