Белорусско-Российский университет
Кафедра «Программное обеспечение информационных технологий»

Информатика. Программирование на Python

Тема: Python. Основы. Массивы / Списки (Библиотека NumPy)

КУТУЗОВ Виктор Владимирович

Могилев, 2021

NumPy

- **NumPy** библиотека для Python
- Возможности библиотеки:
  - поддержка многомерных массивов (включая матрицы);
  - поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами.
- NumPy можно рассматривать как свободную альтернативу MatLab.

- https://numpy.org/ - Официальный сайт библиотеки NumPy
- https://numpy.org/devdocs/user/index.html - Руководство пользователя NumPy

NumPy – подключение

- Самый простой вариант подключения

```
import numpy
```

- Тем не менее, для большого количества вызовов функций `numpy`, становиться утомительно писать `numpy.X` снова и снова.
- Вместо этого намного легче сделать это так:

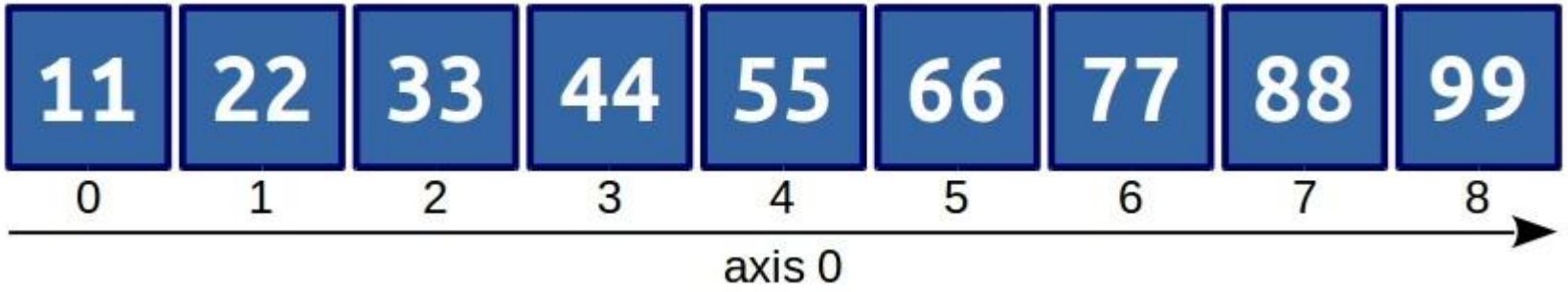
```
import numpy as np
```

NumPy

- Главный объект NumPy - это однородный многомерный массив.
- Чаще всего это одномерная последовательность или двумерная таблица, заполненные элементами одного типа, как правило числами, которые проиндексированы кортежем положительных целых чисел. В NumPy, элементы этого кортежа называются осями, а число осей рангом.

NumPy. Массив

```
import numpy as np
a = np.array([11, 22, 33, 44, 55, 66, 77, 88, 99])
print(a)
```




```
print(a[0])
print(a[1])
print(a[2])
print(a[-1])
print(a[-2])
print(a[-3])
```

```
11
22
33
99
88
77
```

NumPy. Массив

```
import numpy as np
a = np.arange(12)
print(a)
```

```
[ 0 1 2 3 4 5 6 7 8 9 10 11]
```

Функция **arange()** возвращает одномерный массив с равномерно разнесенными значениями внутри заданного интервала

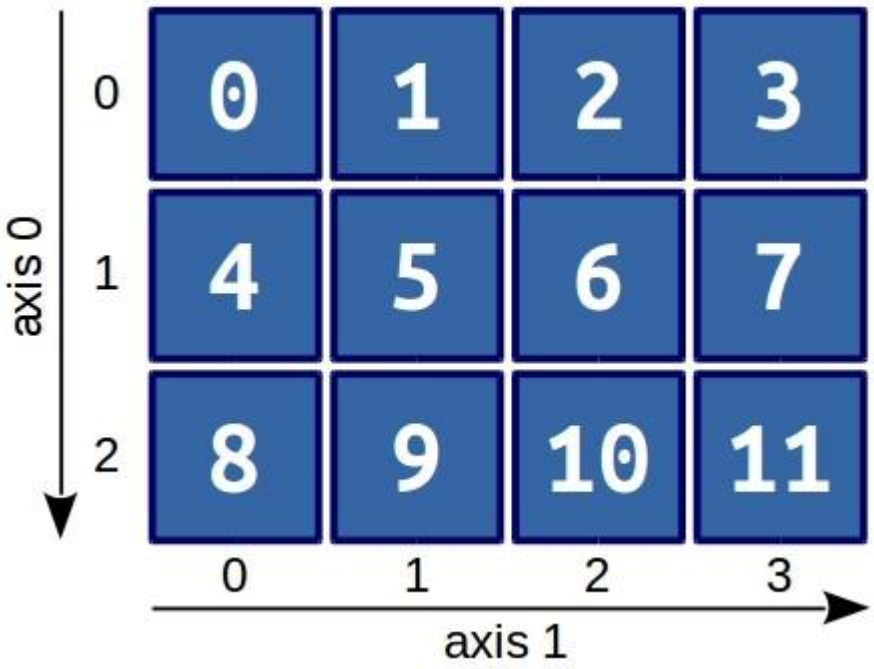
```
a = a.reshape(3, 4)
print(a)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

Функция **reshape()** изменяет форму массива без изменения его данных.

NumPy. Двумерный массив

```
import numpy as np
a = np.arange(12).reshape(3, 4)
```




```
print(a[0][0])
print(a[1][1])

print(a[2][0])
print(a[2][1])
print(a[2][2])
print(a[2][3])
```

```
0
5

8
9
10
11
```

NumPy. Двумерный массив

- Создадим двумерный массив 4 на 6 из случайных чисел от 0 до 15

```
import numpy as np
a = np.random.randint(0, 15, size = (4, 6))
print(a)
```

```
[[ 5 11 11  0  0  7]
 [ 5  7  1  3  3  0]
 [ 8 12  5  6  8  4]
 [ 7  4  2  4 14 13]]
```

Функция **numpy.random.randint** – это массив случайных целых чисел из интервала [low; high). Если параметр high не указан, то значения берутся из интервала [0, low).
Числа берутся из дискретного равномерного распределения.

NumPy. Двумерный массив

```
import numpy as np
a = np.random.randint(0, 15, size = (4, 6))
print(a)

# минимальный элемент массива
array_min = a.min()
print(array_min)

# максимальный элемент массива
array_max = a.max()
print(array_max)

# среднее значение всех элементов массива
array_mean =a.mean()
print(array_mean)

# сумма всех элементов массива
array_sum =a.sum()
print(array_sum)
```

```
[[14 12 11  8  0  9]
 [ 6  6  3  3  2 10]
 [ 7  1  0  3  5  2]
 [ 0  7  8  4  2 13]]

0

14

5.666666666666667

136
```

NumPy. Двумерный массив

```
import numpy as np
a = np.random.randint(0, 15, size = (4, 6))
print(a)

#  минимальные элементы по столбцам
print(a.min(axis = 0))
#  максимальные элементы по столбцам
print(a.max(axis = 0))
#  среднее по столбцам
print(a.mean(axis = 0))

#  минимальные элементы по строкам
print(a.min(axis = 1))
#  максимальные элементы по строкам
print(a.max(axis = 1))
#  среднее по строкам
print(a.mean(axis = 1))
```

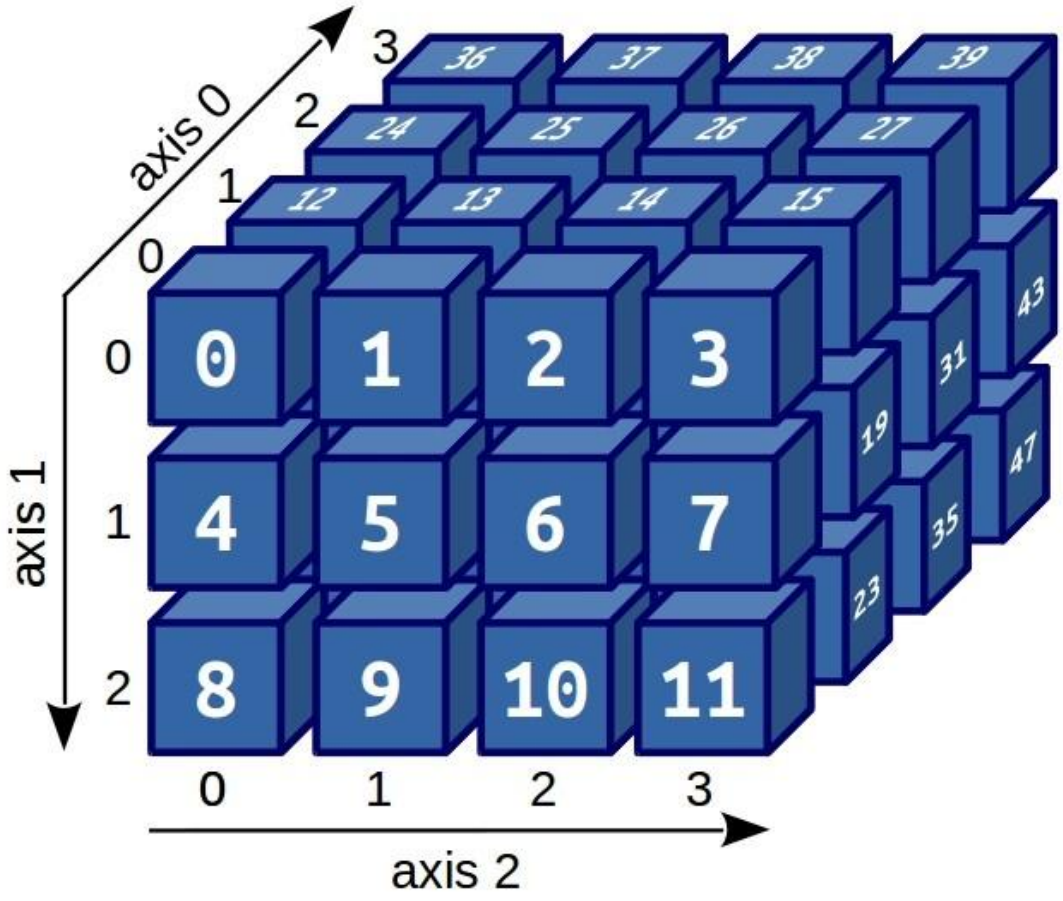
```
[[14 11  9  2  6  2]
 [ 8  7 12  1  4  8]
 [ 6 11  8  8  4  5]
 [10  9  7  1 11  3]]

[6 7 7 1 4 2]
[14 11 12 8 11 8]
[9.5  9.5  9. 3.  6.25 4.5 ]

[2 1 4 1]
[14 12 11 11]
[7.333333 6.666667 7. 6.833333]
```

NumPy. Трехмерный массив

```
a = np.arange(48).reshape(4, 3, 4)
```

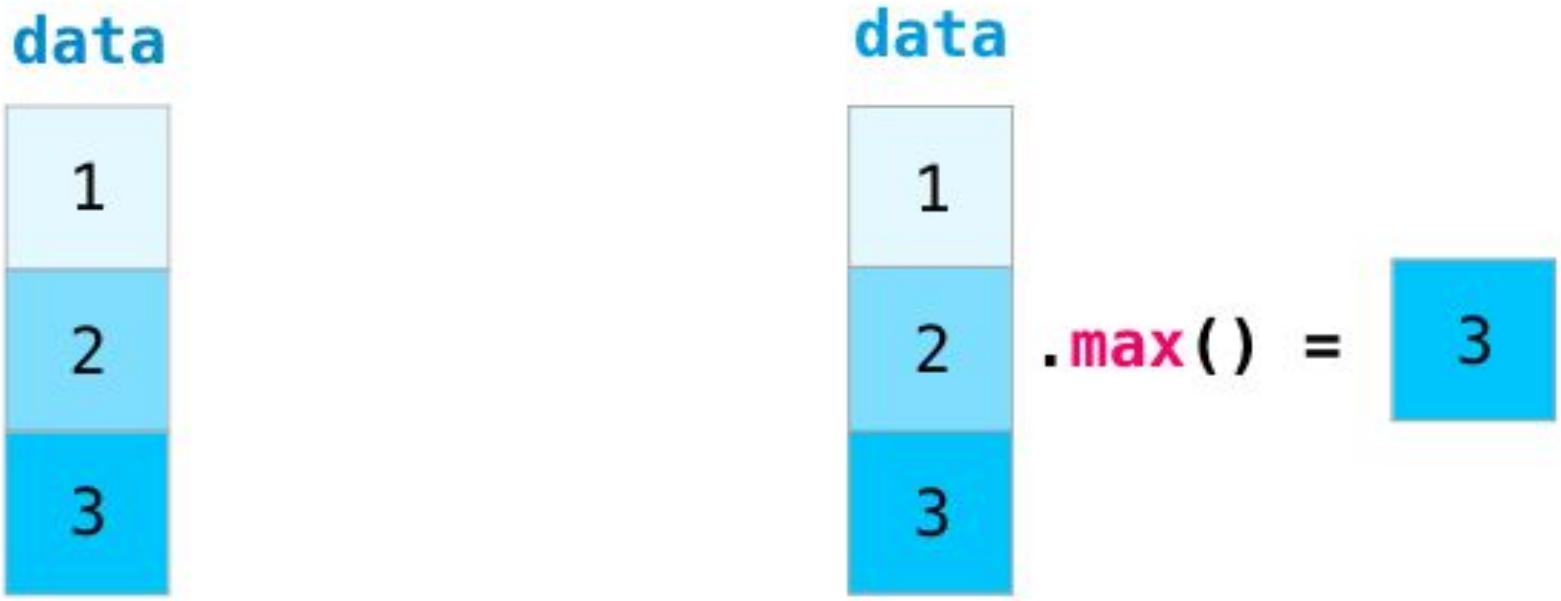


Базовые математические функции

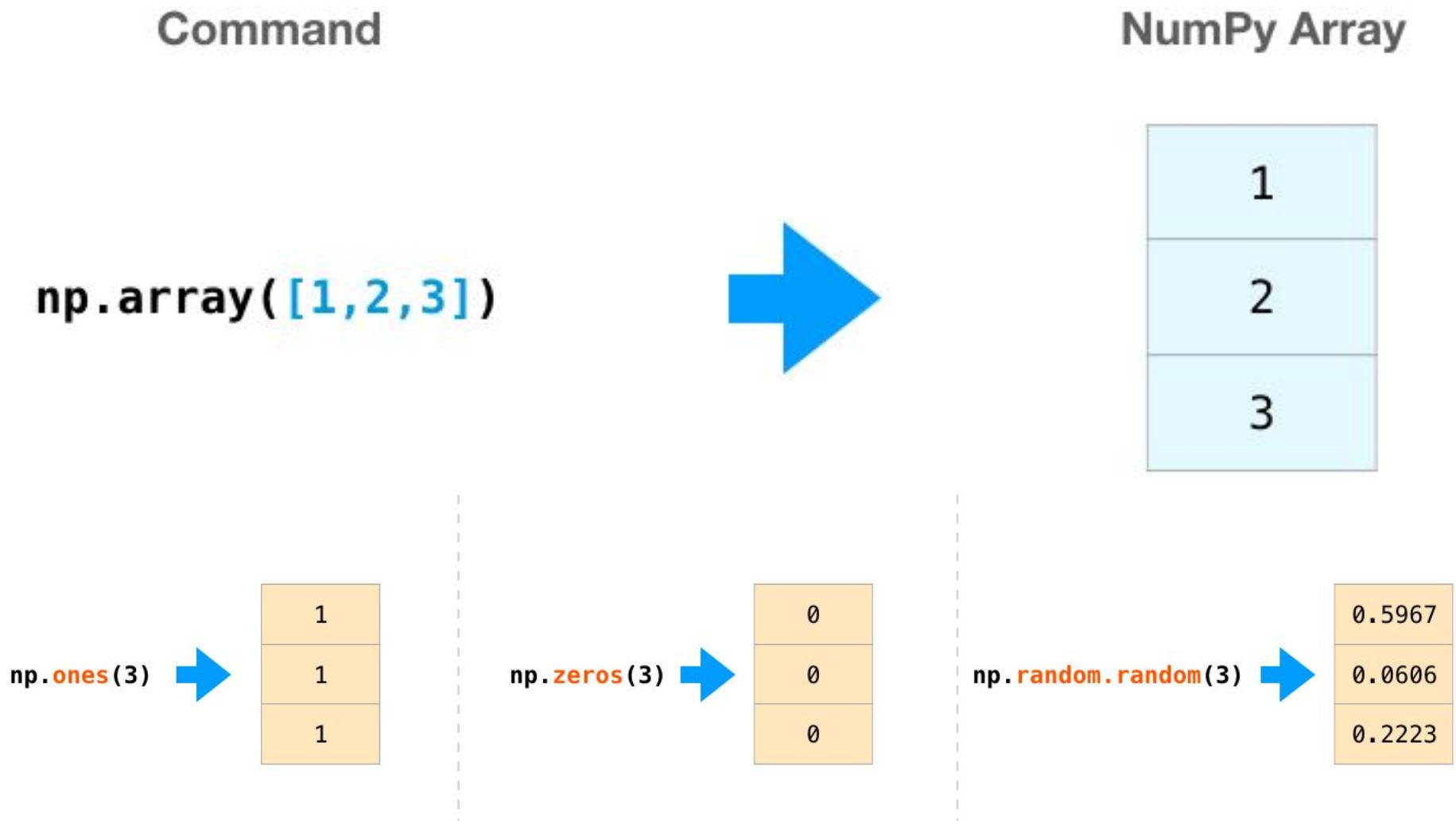
Название	Описание
np.abs(x)	Вычисление модуля от аргумента(ов)x; xможет быть числом, списком или массивом.
np.amax(x)	Нахождение максимального значения от аргумента(ов)x
np.amin(x)	Нахождение минимального значения от аргумента(ов)x
np.argmax(x)	Нахождение индекса максимального значения длях.
np.argmin(x)	Нахождение индекса минимального значения длях.
np.around(x)	Округление до ближайшего целого.
np.mean(x)	Вычисление среднего значения.
np.log(x)	Вычисление натурального логарифма.
np.log2(x)	Вычисление логарифма по основанию 2.

NumPy

```
data = np.array([1,2,3])
```



NumPy


Command
NumPy Array
np.array([1,2,3])
1
2
3
np.ones(3)
1
1
1
np.zeros(3)
0
0
0
np.random.random(3)
0.5967
0.0606
0.2223

NumPy

```
data = np.array([1,2])
```

data

1
2

```
ones = np.ones(2)
```

ones

1
1

data + ones =

data

1
2

+

ones

1
1

=

2
3

NumPy

```
data = np.array([1,2])
```

data

1
2

```
ones = np.ones(2)
```

ones

1
1

data

1
2

\-

ones

1
1

=

0
1

data

1
2

*

data

1
2

=

1
4

data

1
2

/

data

1
2

=

1
1

1
2

* 1.6 =

1
2

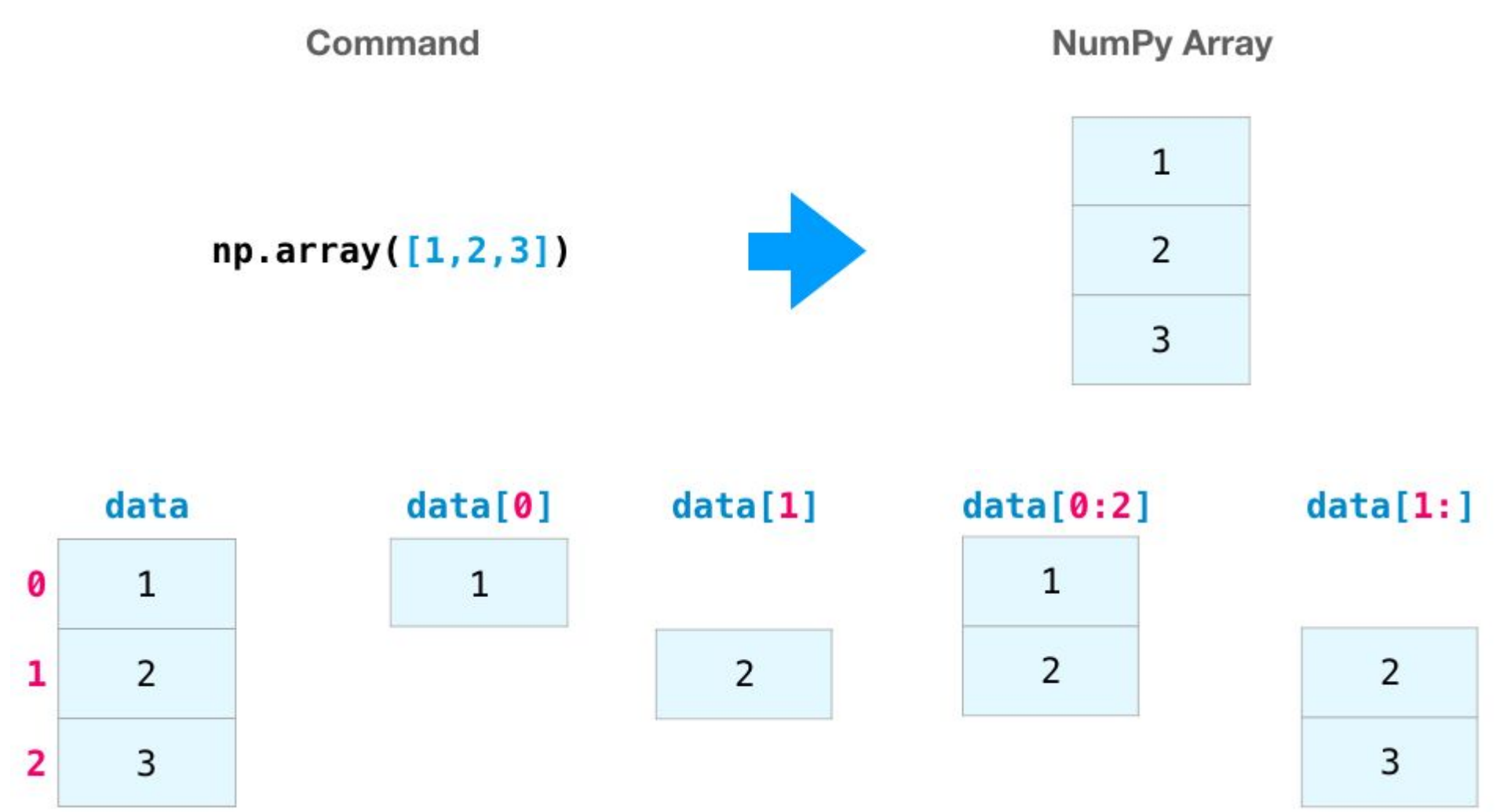
*

1.6
1.6

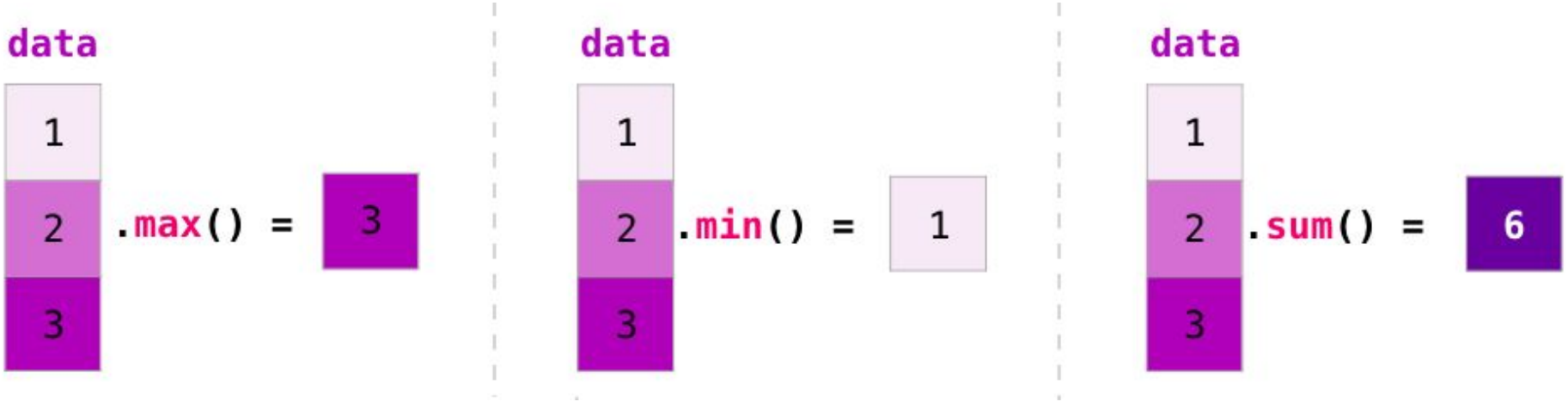
=

1.6
3.2

NumPy

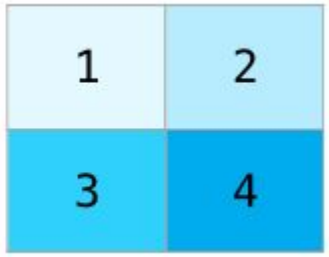

Command
NumPy Array
np.array([1,2,3])
1
2
3
data
0
1
2
1
2
3
data[0]
1
data[1]
2
data[0:2]
1
2
data[1:]
2
3

NumPy

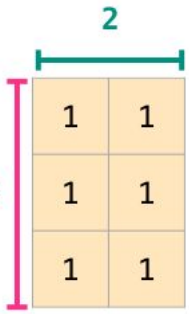


NumPy

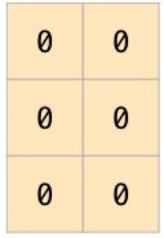
```
np.array([[1,2],[3,4]])
```



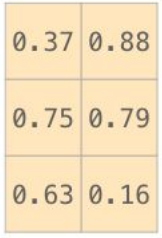

```
np.ones((3,2))
```



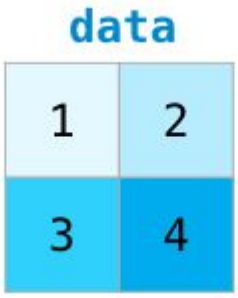

```
np.zeros((3,2))
```



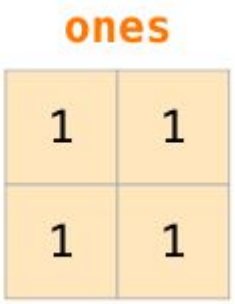

```
np.random.random((3,2))
```




```
data + ones =
```

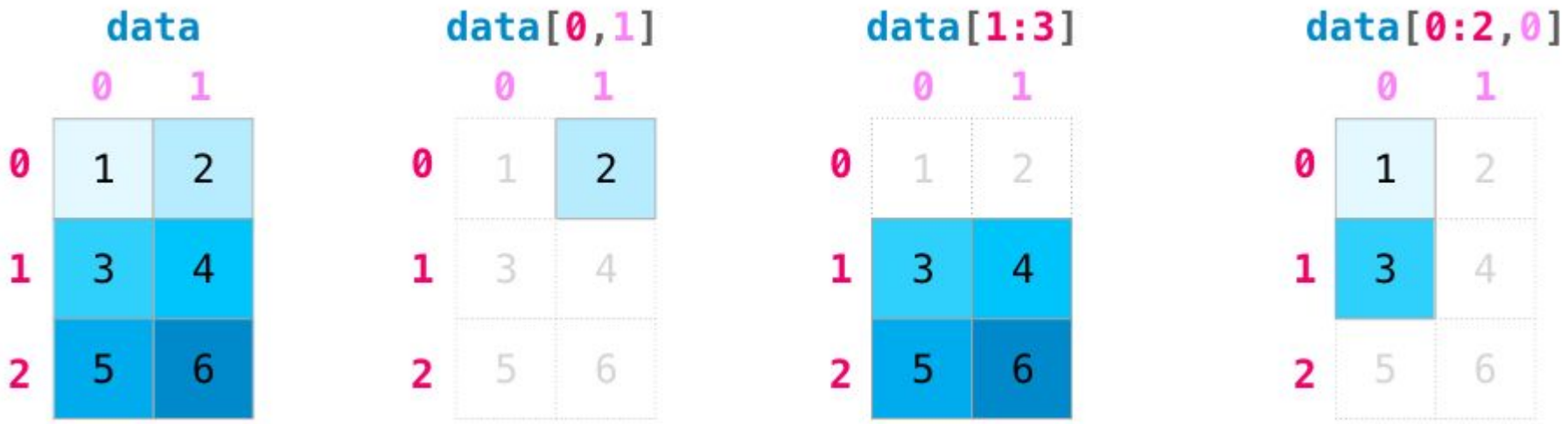


+

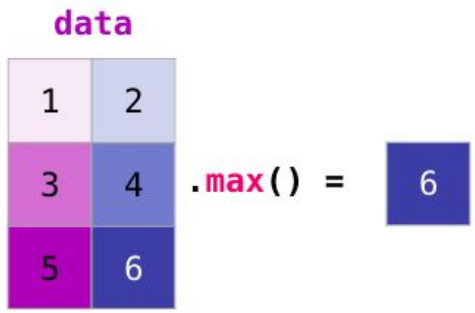


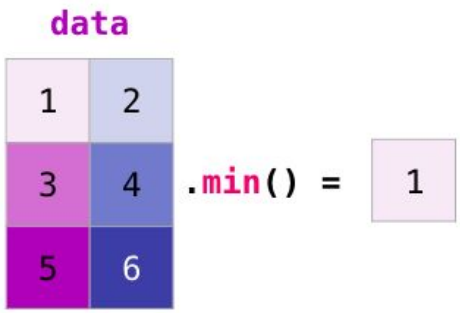
=

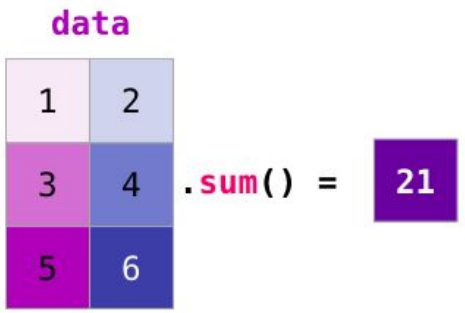
NumPy


data
0 1
0 1 2
1 3 4
2 5 6
data[0,1]
0 1
0 1 2
1 3 4
2 5 6
data[1:3]
0 1
0 1 2
1 3 4
2 5 6
data[0:2,0]
0 1
0 1 2
1 3 4
2 5 6

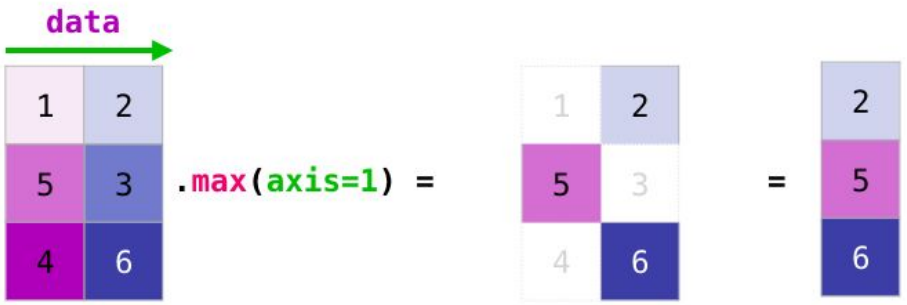
NumPy


data
1 2
3 4
5 6
.max() = 6

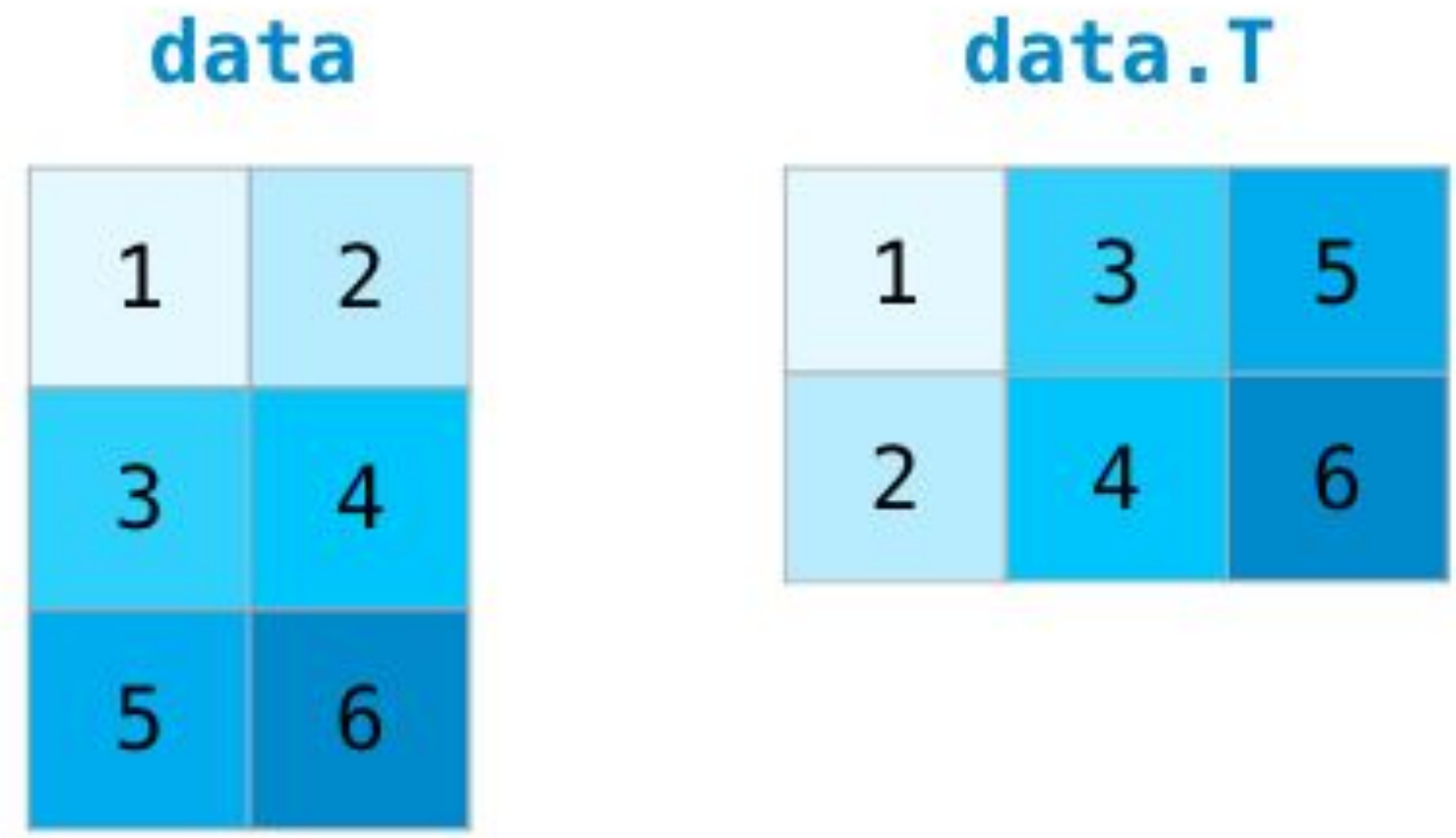

data
1 2
3 4
5 6
.min() = 1


data
1 2
3 4
5 6
.sum() = 21

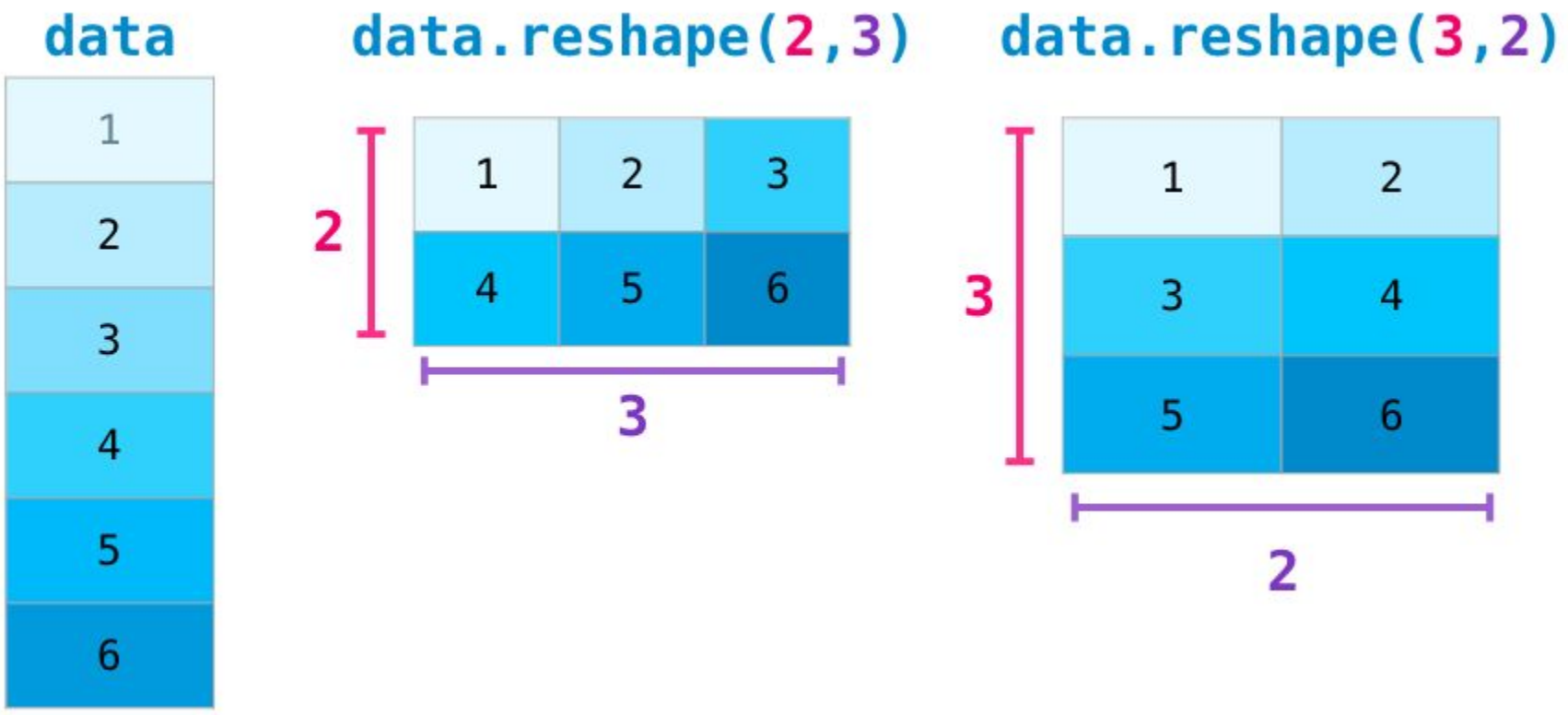

data
1 2
5 3
4 6
.max(axis=0) =
1 2
5 3
4 6
=
5 6


data
1 2
5 3
4 6
.max(axis=1) =
1 2
5 3
4 6
=
2
5
6

NumPy

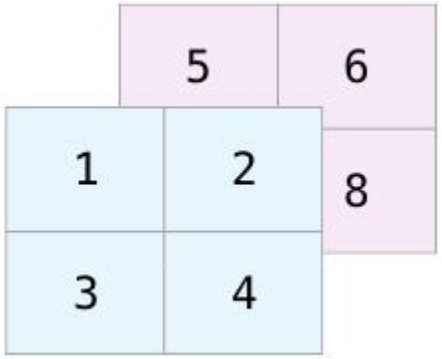


NumPy

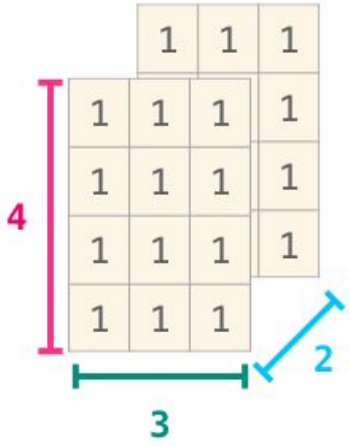

data
1
2
3
4
5
6
data.reshape(2,3)
2
1
2
3
4
5
6
3
data.reshape(3,2)
3
1
2
3
4
5
6
2

NumPy

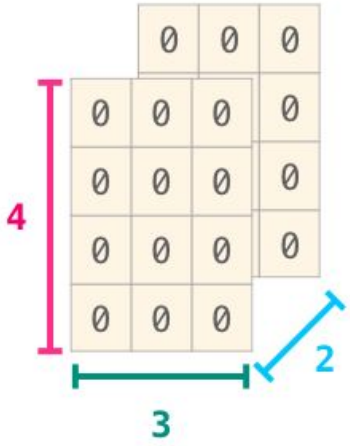
```
np.array([ [[1,2],[3,4]],
           [[5,6],[7,8]] ])
```



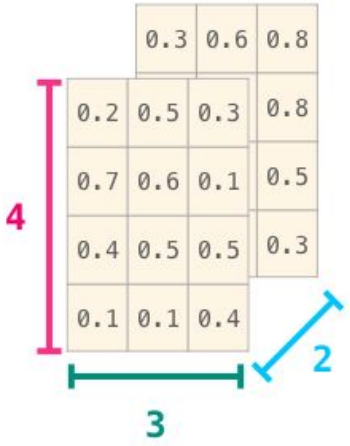

```
np.ones((4,3,2))
```




```
np.zeros((4,3,2))
```




```
np.random.random((4,3,2))
```



NumPy – Сложение матриц

```
import numpy as np
A = np.array([[3, 4, 5],
              [1, 2, 3],
              [8, 9, 3]])

B = np.array([[2, 5, 9],
              [8, 0, 4],
              [12, 0, 3]])
C = A + B

print(A)
print(B)
print(C)
```

$$\begin{bmatrix} 3 & 4 & 5 \\ 1 & 2 & 3 \\ 8 & 9 & 3 \end{bmatrix} + \begin{bmatrix} 2 & 5 & 9 \\ 8 & 0 & 4 \\ 12 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 5 & 9 & 14 \\ 9 & 2 & 7 \\ 20 & 9 & 6 \end{bmatrix}$$

$$A + B = C$$

```
[[3 4 5]
 [1 2 3]
 [8 9 3]]

[[ 2  5  9]
 [ 8  0  4]
 [12  0  3]]

[[ 5  9 14]
 [ 9  2  7]
 [20  9  6]]
```

NumPy – Сумма элементов массива

```
import numpy as np
A = np.array([[3, 4, 5],
              [1, 2, 3],
              [8, 9, 3]])

total = sum(sum(A))

print(total)
```

38

$$A = \begin{bmatrix} 3 & 4 & 5 \\ 1 & 2 & 3 \\ 8 & 9 & 3 \end{bmatrix}$$

$$\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A_{(i,j)} = 38$$

NumPy - Вычитание

```
import numpy as np
A = np.array([[3, 4, 5],
              [1, 2, 3],
              [8, 9, 3]])

B = np.array([[2, 5, 9],
              [8, 0, 4],
              [12, 0, 3]])

C = A - B

print(A)
print(B)
print(C)
```

$$\begin{bmatrix}3 & 4 & 5\\1 & 2 & 3\\8 & 9 & 3\end{bmatrix} - \begin{bmatrix}2 & 5 & 9\\8 & 0 & 4\\12 & 0 & 3\end{bmatrix} = \begin{bmatrix}1 & -1 & -4\\-7 & 2 & -1\\-4 & 9 & 6\end{bmatrix}$$

$$A - B = C$$

```
[[3 4 5]
 [1 2 3]
 [8 9 3]]

[[ 2  5  9]
 [ 8  0  4]
 [12  0  3]]

[[ 1 -1 -4]
 [-7  2 -1]
 [-4  9  0]]
```

NumPy - Умножение

```
import numpy as np
A = np.array([[3, 4, 5],
              [1, 2, 3],
              [8, 9, 3]])

B = np.array([[2, 5, 9],
              [8, 0, 4],
              [12, 0, 3]])

C = A * B

print(A)
print(B)
print(C)
```

$$\begin{bmatrix}3 & 4 & 5\\1 & 2 & 3\\8 & 9 & 3\end{bmatrix} * \begin{bmatrix}2 & 5 & 9\\8 & 0 & 4\\12 & 0 & 3\end{bmatrix} = \begin{bmatrix}6 & 20 & 45\\8 & 0 & 12\\96 & 0 & 9\end{bmatrix}$$

$$A * B = C$$

```
[[3 4 5]
 [1 2 3]
 [8 9 3]]

[[ 2  5  9]
 [ 8  0  4]
 [12  0  3]]

[[ 6 20 45]
 [ 8  0 12]
 [96  0  9]]
```

NumPy – Скалярное умножение

```
import numpy as np
A = np.array([[3, 4, 5],
              [1, 2, 3],
              [8, 9, 3]])

C = A * 3

print(A)
print(C)
```

```
[[3 4 5]
 [1 2 3]
 [8 9 3]]

[[ 9 12 15]
 [ 3  6  9]
 [24 27  9]]
```

$$\begin{bmatrix} 3 & 4 & 5 \\ 1 & 2 & 3 \\ 8 & 9 & 3 \end{bmatrix} * 3 = \begin{bmatrix} 9 & 12 & 15 \\ 3 & 6 & 9 \\ 24 & 27 & 9 \end{bmatrix}$$

$$A * 3 = C$$

NumPy – Умножение

```
import numpy as np
A = np.array([[3, 4, 5],
              [1, 2, 3],
              [8, 9, 3]])

B = np.array([[2, 5, 9],
              [8, 0, 4],
              [12, 0, 3]])

C = np.dot(A, B)

print(A)
print(B)
print(C)
```

$$(AB)_{ij} = \sum_{k=0}^{m-1} A_{ik} B_{kj}$$

$$\begin{bmatrix} 3 & 4 & 5 \\ 1 & 2 & 3 \\ 8 & 9 & 3 \end{bmatrix} \text{x} \begin{bmatrix} 2 & 5 & 9 \\ 8 & 0 & 4 \\ 12 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 98 & 15 & 58 \\ 54 & 5 & 26 \\ 124 & 40 & 117 \end{bmatrix}$$

$$A \text{x} B = C \quad \text{or} \quad AB = C$$

```
[[3 4 5]
 [1 2 3]
 [8 9 3]]

[[ 2  5  9]
 [ 8  0  4]
 [12  0  3]]

[[ 98  15  58]
 [ 54   5  26]
 [124  40 117]]
```

NumPy – Транспонирование

```
import numpy as np
A = np.array([[3, 4, 5],
              [1, 2, 3],
              [8, 9, 3]])

D = np.transpose(A)

print(A)
print(D)
```

```
[[3 4 5]
 [1 2 3]
 [8 9 3]]

[[3 1 8]
 [4 2 9]
 [5 3 3]]
```

$$\begin{bmatrix} 3 & 4 & 5 \\ 1 & 2 & 3 \\ 8 & 9 & 3 \end{bmatrix}^T = \begin{bmatrix} 3 & 1 & 8 \\ 4 & 2 & 9 \\ 5 & 3 & 3 \end{bmatrix}$$

$$D = A^T$$

NumPy – Обратная матрица

```
import numpy as np
A = np.array([[3, 4, 5],
              [1, 2, 3],
              [8, 9, 3]])

D = np.linalg.inv(A)

print(A)
print(D)
```

```
[[3 4 5]
 [1 2 3]
 [8 9 3]]

[[ 1.5  -2.35714286 -0.14285714]
 [-1.5   2.21428571  0.28571429]
 [ 0.5  -0.35714286 -0.14285714]]
```

$$\begin{bmatrix} 3 & 4 & 5 \\ 1 & 2 & 3 \\ 8 & 9 & 3 \end{bmatrix}^{-1} \approx \begin{bmatrix} 1.5 & -2.35 & -0.14 \\ -1.5 & 2.21 & 0.28 \\ 0.5 & -0.36 & -0.14 \end{bmatrix}$$

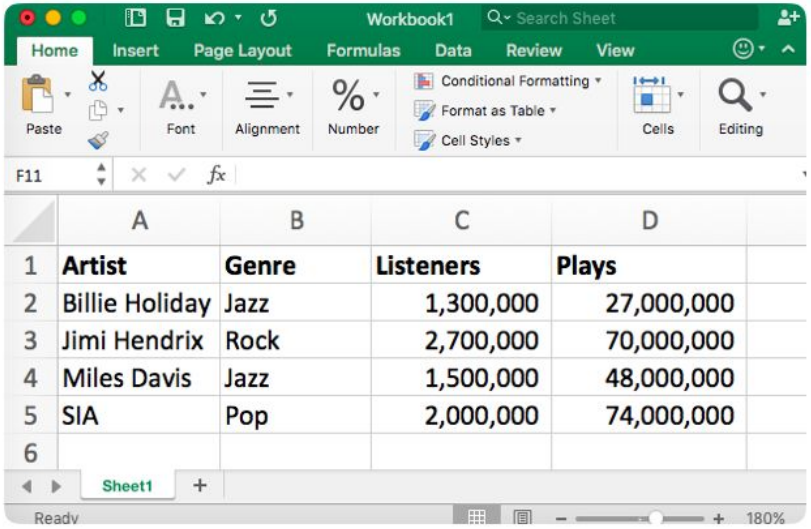
$$D = A^{-1}$$

NumPy

Дополнительные примеры практического использования

Пример

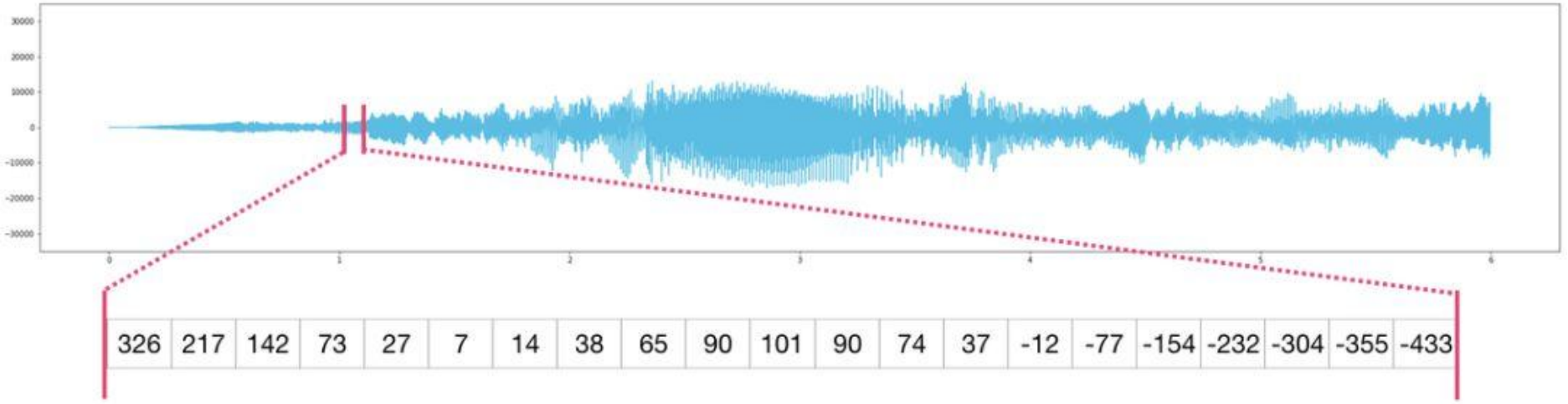
music.csv



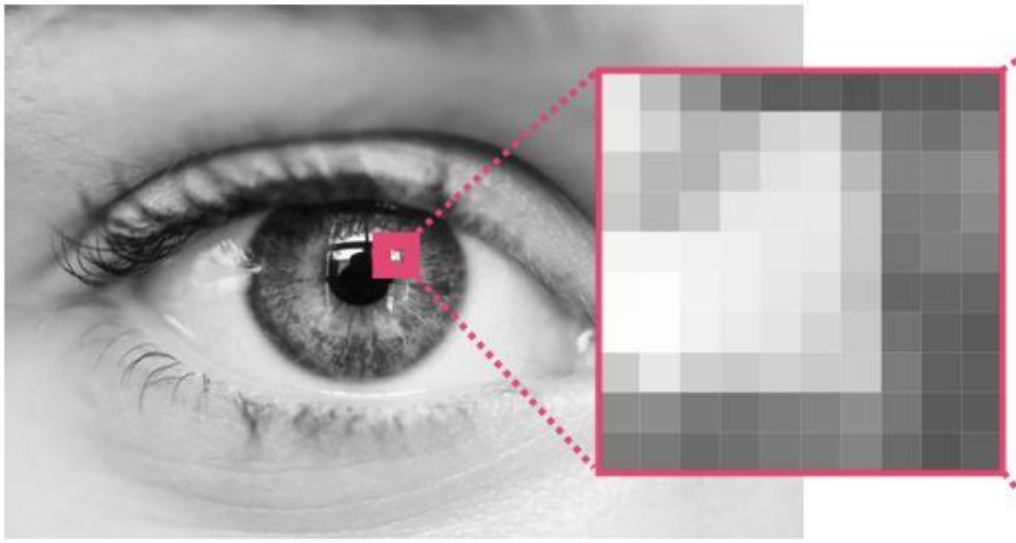
pandas.read_csv('music.csv')

	Artist	Genre	Listeners	Plays
0	Billie Holiday	Jazz	1,300,000	27,000,000
1	Jimi Hendrix	Rock	2,700,000	70,000,000
2	Miles Davis	Jazz	1,500,000	48,000,000
3	SIA	Pop	2,000,000	74,000,000

Пример

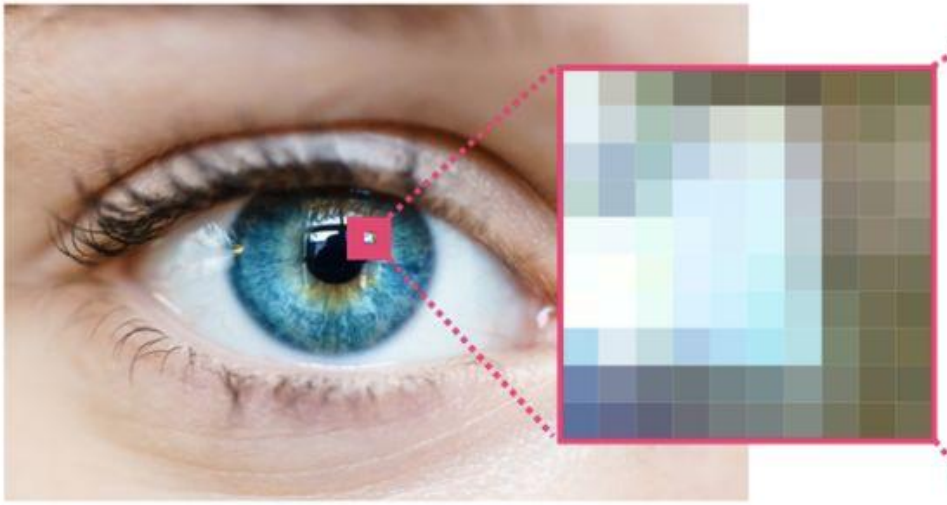

326 217 142 73 27 7 14 38 65 90 101 90 74 37 -12 -77 -154 -232 -304 -355 -433

Пример



230	194	147	108	90	98	84	96	91	101
237	206	188	195	207	213	163	123	116	128
210	183	180	205	224	234	188	122	134	147
198	189	201	227	229	232	200	125	127	135
249	241	237	244	232	226	202	116	125	126
251	254	241	239	230	217	196	102	103	99
243	255	240	231	227	214	203	116	95	91
204	231	208	200	207	201	200	121	95	95
144	140	120	115	125	127	143	118	92	91
121	121	108	109	122	121	134	106	86	97

Пример



	233	188	137	96	90	95	63	73	73	82
237	202	159	120	105	110	88	107	112	121	109
226	191	147	110	101	112	98	123	110	119	142	131
221	191	176	182	203	214	169	144	133	145	155	122
185	160	161	184	205	223	186	137	147	161	140	115
181	174	189	207	206	215	194	136	142	151	133	87
246	237	237	231	208	206	192	122	143	144	111	74
254	254	241	224	199	192	181	99	122	117	107	74
239	248	232	207	187	182	184	110	114	110	113	74
193	215	193	167	158	164	181	114	112	111	105	82
113	119	110	111	113	123	135	120	108	106	113	
93	97	91	103	107	111	122	112	104	114		

Пример

Model Vocabulary

#	
0	the
1	of
2	and
…	…
71,289	dolophine

have	the	bards	who	preceded	me	left	any	theme	unsung
38	0	29104	56	7027	745	225	104	2211	66609

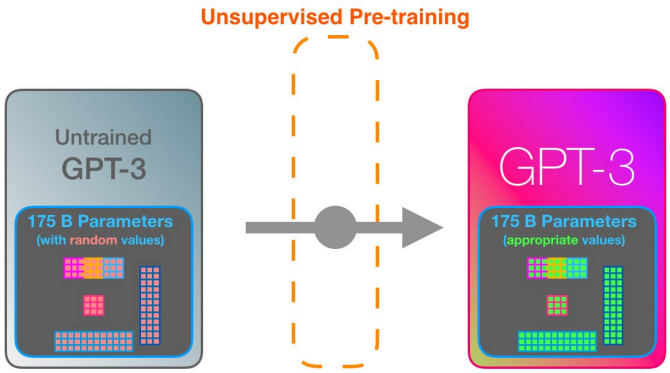
	have	the	bards	who	preceded	me	left	any	theme	unsung
0	-1.621	-0.634	-0.324	-1.357	-0.261	4.632	1.236	-0.046	-1.518	-0.082
1	-1.401	0.683	-0.114	1.891	0.544	-1.487	1.247	-0.408	0.202	-0.022
	…	…	…	…	…	…	…	…	…	…
49	-0.647	0.200	-0.203	1.573	-0.520	-0.355	-1.491	1.325	2.550	-0.010

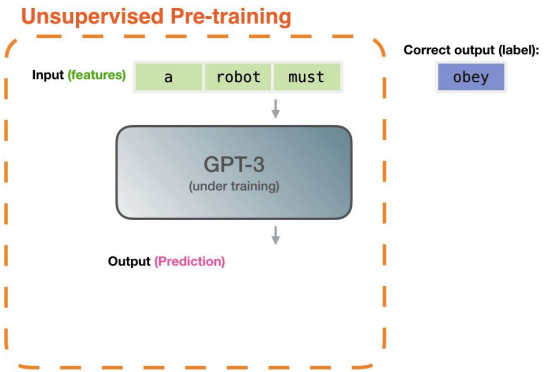
Пример

Text: Second Law of Robotics: A robot must obey the orders given it by human beings

Generated training examples

Example #	Input (features)	Correct output (labels)
1	Second law of robotics :	a
2	Second law of robotics : a	robot
3	Second law of robotics : a robot	must
...		





Справочник функций

Математические функции

Тригонометрические функции

- `sin(x)` - Тригонометрический синус.
- `cos(x)` - Тригонометрический косинус.
- `tan(x)` - Тригонометрический тангенс.
- `arcsin(x)` - Обратный тригонометрический синус.
- `arccos(x)` - Обратный тригонометрический косинус.
- `arctan(x)` - Обратный тригонометрический тангенс.
- `hypot(x1, x2)` - Вычисляет длину гипотенузы по указанным длинам катетов.
- `arctan2(x1, x2)` - Обратный тригонометрический тангенс угла где x1 - противолежащий катет, x2 - прилежащий катет. В отличие от arctan (x) функция arctan2 (y, x) справедлива для всех углов и поэтому может быть использована для преобразования вектора в угол без риска деления на ноль, а также возвращает результат в правильном квадранте.

Тригонометрические функции

- `degrees(x)` - Преобразует радианную меру угла в градусную.
- `radians(x)` - Преобразует градусную меру угла в радианную.
- `unwrap(p[, discont, axis])` - Корректировка фазовых углов при переходе через значение *pi*.
- `deg2rad(x)` - Преобразует градусную меру угла в радианную.
- `rad2deg(x)` - Преобразует радианную меру угла в градусную.

Гиперболические функции

- `sinh(x)` - Гиперболический синус.
- `cosh(x)` - Гиперболический косинус.
- `tanh(x)` - Гиперболический тангенс.
- `arcsinh(x)` - Обратный гиперболический синус.
- `arccosh(x)` - Обратный гиперболический косинус.
- `arctanh(x)` - Обратный гиперболический тангенс.

Округление

- `around(a[, decimals, out])` - Равномерное (банковское) округление до указанной позиции к ближайшему четному числу.
- `round_(a[, decimals, out])` - Эквивалентна around().
- `rint(x)` - Округляет до ближайшего целого.
- `fix(x[, out])` - Округляет до ближайшего к нулю целого числа.
- `floor(x)` - Округление к меньшему ("пол").
- `ceil(x)` - Округление к большему ("потолок").
- `trunc(x)` - Отбрасывает дробную часть числа.

Суммы, разности, произведения

- `prod(a[, axis, dtype, out, keepdims])` - Произведение элементов массива по заданной оси.
- `sum(a[, axis, dtype, out, keepdims])` - Сумма элементов массива по заданной оси.
- `nanprod(a[, axis, dtype, out, keepdims])` - Произведение элементов массива по заданной оси в котором элементы *NaN* учитываются как 1.
- `nansum(a[, axis, dtype, out, keepdims])` - Сумма элементов массива по заданной оси в котором элементы *NaN* учитываются как 0.
- `cumprod(a[, axis, dtype, out])` - Возвращает накопление произведения элементов по заданной оси, т.е. массив в котором каждый элемент является произведением предшествующих ему элементов по заданной оси в исходном массиве.
- `cumsum(a[, axis, dtype, out])` - Возвращает накопление суммы элементов по заданной оси, т.е. массив в котором каждый элемент является суммой предшествующих ему элементов по заданной оси в исходном массиве.
- `nancumprod(a[, axis, dtype, out])` - Возвращает накопление произведения элементов по заданной оси, т.е. массив в котором каждый элемент является произведением предшествующих ему элементов по заданной оси в исходном массиве. Элементы *NaN* в исходном массиве при произведении учитываются как 1.

Суммы, разности, произведения

- `nancumsum(a[, axis, dtype, out])` - Возвращает накопление суммы элементов по заданной оси, т.е. массив в котором каждый элемент является суммой предшествующих ему элементов по заданной оси в исходном массиве. Элементы *NaN* в исходном массиве при суммировании учитываются как 0.
- `diff(a[, n, axis])` - Возвращает *n*-ю разность вдоль указанной оси.
- `ediff1d(ary[, to_end, to_begin])` - Разность между последовательными элементами массива.
- `gradient(f, *varargs, **kwargs)` - Дискретный градиент (конечные разности вдоль осей) массива f.
- `cross(a, b[, axisa, axisb, axisc, axis])` - Векторное произведение двух векторов.
- `trapz(y[, x, dx, axis])` - Интегрирование массива вдоль указанной оси методом трапеций.

Экспоненцирование и логарифмирование

- `exp(x, /[, out, where, casting, order, …])` - Экспонента всех элементов массива.
- `expm1(x, /[, out, where, casting, order, …])` - Вычисляет exp(x)-1 всех элементов массива.
- `exp2(x, /[, out, where, casting, order, …])` - Вычисляет 2**x для всех *x* входного массива.
- `log(x, /[, out, where, casting, order, …])` - Натуральный логарифм элементов массива.
- `log10(x, /[, out, where, casting, order, …])` - Десятичный логарифм элементов массива.
- `log2(x, /[, out, where, casting, order, …])` - Логарифм элементов массива по основанию 2.
- `log1p(x, /[, out, where, casting, order, …])` - Вычисляет log(x+1) для всех *x* входного массива.
- `logaddexp(x1, x2, /[, out, where, casting, …])` - Натуральный логарифм суммы экспонент элементов входных массивов.
- `logaddexp2(x1, x2, /[, out, where, casting, …])` - Двоичный логарифм от *2**x1 + 2**x2* для всех элементов входных массивов.

Другие специальные функции

- `i0(x)` - Модифицированная функция Бесселя первого рода нулевого порядка.
- `sinc(x)` - Вычисляет нормированный кардинальный синусс элементов массива.

Операции с плавающей точкой

- `signbit(x, /[, out, where, casting, order, …])` - Возвращает *True* для всех элементов массива у которых знаковый бит установлен в отрицательное значение.
- `copysign(x1, x2, /[, out, where, casting, …])` - Изменяет знак элементов из массива *x1* на знак элементов из массива *x2*.
- `frexp(x[, out1, out2], / [, out, where, …])` - Разложение элементов массива в показатель мантиссы и двойки.
- `ldexp(x1, x2, /[, out, where, casting, …])` - Вычисляет x1*2**x2.
- `nextafter(x1, x2, /[, out, where, casting, …])` - Возвращает значение с плавающей точкой следующее за элементом из *x1* в направлении элемента из *x2*.
- `spacing(x, /[, out, where, casting, order, …])` - Поэлементно вычисляет расстояние между значением из массива *x* и ближайшим соседним числом.

Арифметические операции

- `lcm(x1, x2, /[, out, where, casting, order, …])` - Поэлементно вычисляет наименьшее общее кратное массивов x1 и x2.
- `gcd(x1, x2, /[, out, where, casting, order, …])` - Поэлементно вычисляет наибольший общий делитель массивов *x1* и *x2*.
- `add(x1, x2, /[, out, where, casting, order, …])` - Поэлементная сумма значений массивов.
- `reciprocal(x, /[, out, where, casting, …])` - Вычисляет обратное значение (1/x) каждого элемента массива.
- `positive(x, /[, out, where, casting, order, …])` - Эквивалентно простому копированию (numpy.copy) элементов массива, но только для массивов поддерживающих математические операции. Формально соответствует математической записи b = +a.
- `negative(x, /[, out, where, casting, order, …])` - Отрицательное значение элементов массива.
- `multiply(x1, x2, /[, out, where, casting, …])` - Поэлементное умножение значений массива *x1* на значения массива *x2*.

Арифметические операции

- `divide(x1, x2, /[, out, where, casting, …])` - Поэлементное деление значений массива *x1* на значения массива *x2*.
- `power(x1, x2, /[, out, where, casting, …])` - Поэлементное возведение значений массива *x1* в степень равную значениям из массива *x2*.
- `subtract(x1, x2, /[, out, where, casting, …])` - Поэлементная разность значений массива *x1* и *x2*.
- `true_divide(x1, x2, /[, out, where, …])` - Поэлементное истинное деление значений массива *x1* на значения массива *x2*.
- `floor_divide(x1, x2, /[, out, where, …])` - Поэлементное целочисленное деление значений массива *x1* на значения массива *x2*.
- `float_power(x1, x2, /[, out, where, …])` - Поэлементное возведение значений массива *x1* в степень равную значениям из массива *x2*, адаптированное для чисел с плавающей точкой.
- `fmod(x1, x2, /[, out, where, casting, …])` - Поэлементный остаток от деления значений массива *x1* на значения массива *x2*.
- `mod(x1, x2, /[, out, where, casting, order, …])` - Поэлементно вычисляет остаток от деления значений массива *x1* на значения массива *x2*.

Арифметические операции

- `modf(x[, out1, out2], / [, out, where, …])` - Дробная и целая часть элементов массива.
- `remainder(x1, x2, /[, out, where, casting, …])` - Элементарный остаток от деления значений массива *x1* на значения массива *x2*.
- `divmod(x1, x2[, out1, out2], / [[, out, …])` - Результат истинного деления и остаток от деления значений массива *x1* на значения массива *x2*.

Операции с комплексными числами

- `angle(z[, deg])` - Вычисляет угол каждого комплексного числа в массиве.
- `real(val)` - Действительная часть комплексного числа.
- `imag(val)` - Мнимая часть комплексного числа.
- `conj(x, /[, out, where, casting, order, …])` - Комплексно-сопряженный элемент.

Прочие математические функции

- `convolve(a, v[, mode])` - Дискретная линейная свертка.
- `clip(a, a_min, a_max[, out])` - Ограничение значений массивов указанным интервалом допустимых значений.
- `sqrt(x, /[, out, where, casting, order, …])` - Квадратный корень элементов массива.
- `cbrt(x, /[, out, where, casting, order, …])` - Кубический корень элементов массива.
- `square(x, /[, out, where, casting, order, …])` - Квадрат элементов массива.
- `absolute(x, /[, out, where, casting, order, …])` - Абсолютное значение (модуль) элементов массива.
- `fabs(x, /[, out, where, casting, order, …])` - Возвращает абсолютное значение (модуль) элементов массива в виде чисел с плавающей точкой.
- `sign(x, /[, out, where, casting, order, …])` - Элементарный указатель на знак числа.

Прочие математические функции

- `heaviside(x1, x2, /[, out, where, casting, …])` - Ступенчатая функция Хевисайда.
- `maximum(x1, x2, /[, out, where, casting, …])` - Наибольшие значения после поэлементного сравнения значений массивов.
- `minimum(x1, x2, /[, out, where, casting, …])` - Наименьшие значения после поэлементного сравнения значений массивов.
- `fmax(x1, x2, /[, out, where, casting, …])` - Наибольшие значения после поэлементного сравнения значений массивов в виде чисел с плавающей точкой.
- `fmin(x1, x2, /[, out, where, casting, …])` - Наименьшие значения после поэлементного сравнения значений массивов в виде чисел с плавающей точкой.
- `nan_to_num(x[, copy])` - Заменяет *nan* на 0, бесконечность и минус-бесконечность заменяются на наибольшее и наименьшее доступное число с плавающей точкой соответственно.
- `real_if_close(a[, tol])` - Переводит комплексные числа в вещественные если мнимая часть комплексного числа меньше машинной эпсилон.
- `interp(x, xp, fp[, left, right, period])` - Одномерная линейная интерполяция.

Создание массивов

Автозаполнение массивов

- `empty()` - Возвращает новый массив заданной формы и типа без инициированных записей.
- `empty_like()` - Возвращает новый массив с формой и типом данных указанного массива без инициированных записей.
- `eye()` - Возвращает новый массив в котором диагональные элементы равны единице, а все остальные равны нулю.
- `identity()` - Возвращает новый квадратный массив с единицами по главной диагонали.
- `ones()` - Возвращает новый массив заданной формы и типа, заполненный единицами.
- `ones_like()` - Возвращает новый массив с формой и типом данных указанного массива, заполненный единицами.
- `zeros()` - Возвращает новый массив заданной формы и типа, заполненный нулями.
- `zeros_like()` - Возвращает новый массив с формой и типом данных указанного массива, заполненный нулями.
- `full()` - Возвращает новый массив заданной формы и типа все элементы которого равны указанному значению.
- `full_like()` - Возвращает новый массив с формой и типом данных указанного массива, все элементы которого равны указанному значению.

Заполнение данными

- `array()` - Создает массив NumPy.
- `asarray()` - Преобразует последовательность в массив NumPy.
- `asanyarray()` - Преобразует последовательность в массив NumPy, пропуская подклассы ndarray.
- `ascontiguousarray()` - Возвращает непрерывный массив в памяти с организацией порядка элементов в C-стиле.
- `asmatrix()` - Интерпретирует входные данные как матрицу.
- `copy()` - Возвращает копию массива.
- `frombuffer()` - Преобразует буфер в одномерный массив.
- `fromfile()` - Создает массив из текстового или двоичного файла.
- `fromfunction()` - Создает массив с выполнением указанной функции над каждым элементом.
- `fromiter()` - Создает одномерный массив из итерируемого объекта.
- `fromstring()` - Создает одномерный массив из строки.
- `loadtxt()` - Создает массив из данных в текстовом файле.

Создание массивов записей

- `core.records.array()` - Создает массив записей из указанного объекта.
- `core.records.fromarrays()` - Создает массив записей из одномерных массивов.
- `core.records.fromrecords()` - Создает массив записей из списка записей в текстовой форме.
- `core.records.fromstring()` - Создает (только для чтения) массив записей из двоичных данных, находящихся в строке.
- `core.records.fromfile()` - Создает массив записей из файла с двоичными данными.

Создание массивов символов

- `core.defchararray.array()` - Создает массив символов chararray.
- `core.defchararray.asarray()` - Преобразует последовательность в массив символов chararray.

Числовые диапазоны

- `arange()` - Возвращает одномерный массив с равномерно распределенными значениями внутри указанного интервала.
- `linspace()` - Возвращает одномерный массив с равномерно распределенными значениями внутри указанного интервала.
- `logspace()` - Возвращает одномерный массив значений, равномерно распределенных по логарифмической шкале.
- `geomspace()` - Возвращает одномерный массив значений, которые представляют собой геометрическую прогрессию.
- `meshgrid()` - Создает список массивов координатных сеток.
- `mgrid[]` - Возвращает массив плотных координатных сеток
- `ogrid[]` - Возвращает открытую сетку значений.

Создание матриц

- `mat()` - Преобразует входную последовательность в матрицу.
- `bmat()` - Создает матрицу из строки, вложенной последовательности или массива.
- `diag()` - Извлекает диагональ из массива, а так же позволяет строить диагональные массивы.
- `diagflat()` - Создает диагональный массив из элементов указанного массива.
- `tri()` - Создает массив у которого все элементы ниже указанной диагонали равны 1, а все остальные равны 0.
- `tril()` - Заменяет на 0 все элементы массива, которые оказались выше указанной диагонали.
- `triu()` - Заменяет на 0 все элементы массива, которые оказались ниже указанной диагонали.
- `vander()` - Создает матрицу Вандермонда.

Операции над массивами

Основные операции

- `copyto(dst, src[, casting, where])` - Копирует данные из одного массива в другой с выполнением транслирования если это необходимо.

Изменение формы массива

- `reshape(a, newshape[, order])` - Изменяет форму массива без изменения его данных.
- `ndarray.reshape()` - Изменяет форму массива без изменения его данных.
- `ravel(a[, order])` - Возвращает сжатый до одной оси массив.
- `ndarray.flat` - Одномерный итератор массива.
- `ndarray.flatten([order])` - Возвращает копию массива сжатую до одного измерения.

Транспозиционные операции

- `moveaxis(a, source, destination)` - Перемещает оси массива на указанные позиции.
- `rollaxis(a, axis[, start])` - Ставит указанную ось самой первой или в указанное положение.
- `swapaxes(a, axis1, axis2)` - Меняет местами две указанные оси массива.
- `ndarray.T` - Транспонирует массив.
- `transpose(a[, axes])` - Транспонирует или перемещает оси массива в указанные положения.

Изменение количества измерений

- `atleast_1d(*arys)` - Преобразует входные данные в одномерный массив.
- `atleast_2d(*arys)` - Преобразует входные данные в двумерный массив.
- `atleast_3d(*arys)` - Преобразует входные данные в трехмерный массив.
- `broadcast` - Создает объект, который имитирует транслирование массивов.
- `broadcast_to(array, shape[, subok])` - Транслирует массив по указанной форме.
- `broadcast_arrays(*args, **kwargs)` - Транслирует указанные массивы относительно друг друга.
- `expand_dims(a, axis)` - Добавляет новую ось к массиву.
- `squeeze(a[, axis])` - Удаляет оси с одним элементом, но не сами элементы массива.

Изменение типа массива

- `asarray(a[, dtype, order])` - Интерпретирует входные данные как массив.
- `asanyarray(a[, dtype, order])` - Интерпретирует входные данные как массив, но пропускает подклассы *ndarray*.
- `asmatrix(data[, dtype])` - Интерпретирует входные данные как матрицу.
- `asfarray(a[, dtype])` - Интерпретирует входные данные как массив типа *float*.
- `asfortranarray(a[, dtype])` - Интерпретирует входные данные как массив, с организацией в памяти в стиле *Fortran*.
- `ascontiguousarray(a[, dtype])` - Интерпретирует входные данные как массив, с организацией в памяти в стиле *C*.
- `asarray_chkfinite(a[, dtype, order])` - Интерпретирует входные данные как массив, с предварительной их проверкой на наличие элементов *Nan* и *inf*.
- `asscalar(a)` - Преобразует массив с одним элементом в его скалярный эквивалент.
- `require(a[, dtype, requirements])` - Интерпретирует входные данные как массив, который соответствует указанному типу и флагам.

Соединение массивов

- `concatenate((a1, a2, ...)[, axis, out])` - Соединяет массивы вдоль указанной оси.
- `stack(arrays[, axis, out])` - Соединяет массивы вдоль новой оси.
- `column_stack((a1, a2, ..., aN))` - Соединяет массивы по вертикали, т. е. вдоль первой (не нулевой) оси.
- `row_stack((a1, a2, ..., aN))` - Соединяет массивы по горизонтали, т.е. вдоль нулевой оси.
- `dstack((a1, a2, ..., aN))` - Соединяет массивы вдоль третьей оси.
- `hstack((a1, a2, ..., aN))` - Соединяет массивы по горизонтали.
- `vstack((a1, a2, ..., aN))` - Соединяет массивы по вертикали.
- `block([a1, a2, ..., aN])` - Создает массив из указанной последовательности блоков.

Разбиение массивов

- `split(a, indices_or_sections[, axis])` - Разбивает массив на несколько подмассивов.
- `array_split(a, indices_or_sections[, axis])` - Разбивает массив на несколько подмассивов.
- `dsplit(a, indices_or_sections)` - Разбивает массив на несколько подмассивов вдоль третьей оси.
- `hsplit(a, indices_or_sections)` - Разбивает массив по горизонтали.
- `vsplit(a, indices_or_sections)` - Разбивает массив по вертикали.

Циклические массивы

- `tile(A, reps)` - Создает массив повторением указанного массива *A* заданным количеством раз.
- `repeat(a, repeats[, axis])` - Повторяет элементы массива.

Добавление и удаление элементов

- `delete(a, obj[, axis])` - Удаляет указанные элементы на указанной оси.
- `insert(a, obj, values[, axis])` - Вставляет указанные элементы перед указанными индексами на указанной оси.
- `append(a, values[, axis])` - Добавляет элементы в конец массива.
- `resize(a, new_shape)` - Возвращает новый массив с указанной формой.
- `trim_zeros(filt[, trim])` - Удаляет нули из начала или конца указанного одномерного массива или последовательности.
- `unique(a, return_index=False, return_inverse=False, return_counts=False, axis=None)` - Находит уникальные элементы массива.

Изменение порядка элементов

- `flip(m, axis)` - Располагает элементы в обратном порядке вдоль указанной оси.
- `fliplr(m)` - Отражает массив по горизонтали.
- `flipud(m)` - Отражает массив по вертикали.
- `reshape(a, newshape[, order])` - Изменяет форму массива без изменения его данных.
- `roll(a, shift[, axis])` - Циклическое смещение элементов массива вдоль указанной оси.
- `rot90(m[, k, axes])` - Поворачивает массив на 90 градусов в плоскости указанных осей.

Генерация случайных данных

Получение простых случайных данных

- `rand(d0, d1, …, dn)` - Массив случайных значений заданной формы.
- `randn(d0, d1, …, dn)` - Массив случайных значений с нормальным распределением.
- `randint(low[, high, size, dtype])` - Массив случайных целых чисел из интервала *[low; high)*.
- `random_integers(low[, high, size])` - Массив случайных целых чисел типа *np.int* из интервала *[low; high)*.
- `random_sample([size])` - Массив случайных чисел с плавающей точкой из интервала *[0.0; 1.0)*.
- `random([size])` - Массив случайных чисел с плавающей точкой из интервала *[0.0; 1.0)*. Псевдоним функции random_sample([size]).
- `ranf([size])` - Массив случайных чисел с плавающей точкой из интервала *[0.0; 1.0)*. Псевдоним функции random_sample([size]).
- `sample([size])` - Массив случайных чисел с плавающей точкой из интервала *[0.0; 1.0)*. Псевдоним функции random_sample([size]).
- `choice(a[, size, replace, p])` - Случайная выборка из значений заданного одномерного массива.
- `bytes(length)` - Строка случайных байт заданной длинны.

Перестановки

- `shuffle(x)` - Возвращает случайную перестановку элементов массива.
- `permutation(x)` - Возвращает случайную перестановку элементов массива или случайную последовательность заданной длинны из его элементов.

Генератор псевдо-случайных чисел

- `RandomState([seed])` - Класс-контейнер для генератора псевдослучайных чисел.
- `seed([seed])` - Задает начальные условия для генератора случайных чисел.
- `get_state()` - Возвращает кортеж с данными о внутреннем состоянии генератора.
- `set_state(state)` - Установка внутреннего состояния генератора.

Статистика

Основные статистические характеристики

- `amin(a[, axis, out, keepdims, initial])` - Минимальное значение элементов массива. Параметр axis позволяет указывать оси, вдоль которых необходим поиск минимальных значений.
- `amax(a[, axis, out, keepdims, initial])` - Максимальное значение в массиве. Параметр axis позволяет указывать оси, вдоль которых необходим поиск максимальных значений.
- `nanmin(a[, axis, out, keepdims])` - Минимальное значение массива или минимальное значение вдоль указанной оси. Элементы с значением *np.nan* игнорируются.
- `nanmax(a[, axis, out, keepdims])` - Максимальное значение массива или максимальное значение вдоль указанной оси. Элементы с значением *np.nan* игнорируются.
- `numpy.ptp(a[, axis, out, keepdims])` - Возвращает диапазон значений ([*max* - *min*]) массива или указанной оси массива.
- `numpy.percentile(a, q[, axis, out, overwrite_input, interpolation, keepdims])` - Вычисляет q-й процентиль (перцентиль) значений элементов массива или элементов вдоль указанной оси.

Основные статистические характеристики

- `numpy.nanpercentile(a, q[, axis, out, overwrite_input, interpolation, keepdims])` - Вычисление q-го процентиля (перцентиля) значений вдоль указанной оси массива. Элементы с значением *np.nan* игнорируются.
- `numpy.quantile(a, q[, axis, out, overwrite_input, interpolation, keepdims])` - Вычисление q-го процентиля (перцентиля) значений вдоль указанной оси массива. Элементы с значением *np.nan* игнорируются.
- `numpy.nanquantile(a, q[, axis, out, overwrite_input, interpolation, keepdims])` - Вычисление q-го процентиля (перцентиля) значений вдоль указанной оси массива. Элементы с значением *np.nan* игнорируются.

Средние и отклонения

- `median(a[, axis, out, overwrite_input, keepdims])` - Медиана значений элементов массива, расположенных вдоль указанной оси.
- `average(a[, axis, weights, returned])` - Средневзвешенное значений элементов массива расположенных вдоль указанной оси.
- `mean(a[, axis, dtype, out, keepdims])` - Среднее арифметическое значений элементов массива расположенных вдоль указанной оси.
- `std(a[, axis, dtype, out, ddof, keepdims])` - Стандартное отклонение значений элементов массива расположенных вдоль указанной оси.
- `var(a[, axis, dtype, out, ddof, keepdims])` - Дисперсия значений элементов массива расположенных вдоль указанной оси.
- `nanmedian(a[, axis, out, overwrite_input, …])` - Медиана значений элементов массива расположенных вдоль указанной оси. Элементы с значением *NaN* игнорируются.
- `nanmean(a[, axis, dtype, out, keepdims])` - Среднее арифметическое значений элементов массива расположенных вдоль указанной оси. Элементы с значением *NaN* игнорируются.

Средние и отклонения

- `nanstd(a[, axis, dtype, out, ddof, keepdims])` - Стандартное отклонение значений элементов массива расположенных вдоль указанной оси. Элементы с значением *NaN* игнорируются.
- `nanvar(a[, axis, dtype, out, ddof, keepdims])` - Дисперсия значений элементов массива расположенных вдоль указанной оси. Элементы с значением *NaN* игнорируются.

Корреляции

- `corrcoef(x[, y, rowvar, bias, ddof])` - Коэфициент корреляции Пирсона.
- `correlate(a, v[, mode])` - Взаимнокорреляционная функция двух одномерных последовательностей.
- `cov(m[, y, rowvar, bias, ddof, fweights, …])` - Ковариационная матрица.

Гистограммы

- `histogram(a[, bins, range, normed, weights, …])` - Вычисление гистограммы набора данных.
- `histogram2d(x, y[, bins, range, normed, weights])` - Вычисление двумерной гистограммы двух наборов данных.
- `histogramdd(sample[, bins, range, normed, …])` - Вычисление N-мерной гистограммы N-го количества наборов данных.
- `bincount(x[, weights, minlength])` - Количество вхождений значений в массиве.
- `numpy.histogram_bin_edges(a, bins=10, range=None, weights=None)` - Значения для прямоугольников гистограммы.
- `digitize(x, bins[, right])` - Вычисление индексов числовых интервалов массива *bins* в которые входит каждое последующее значение элемента массива *x*.

Линейная алгебра

Произведение векторов и матриц

- `dot(a, b[, out])` - Скалярное произведение двух массивов.
- `linalg.multi_dot(arrays)` - Скалярное произведение двух и более массивов с автоматической оптимизацией расстановки множителей.
- `vdot(a, b)` - Скалярное произведение двух векторов в том числе и комплексных.
- `inner(a, b)` - Обычное скалярное произведение.
- `outer(a, b[, out])` - Внешнее произведение двух векторов.
- `matmul(a, b[, out])` - Матричное произведение двух массивов.
- `tensordot(a, b[, axes])` - Тензорное скалярное произведение двух массивов вдоль указанных осей.
- `einsum(subscripts, *operands[, out, dtype, …])` - Применение правила суммирования Эйнштейна к указанным массивам.
- `einsum_path(subscripts, *operands[, optimize])` - Оценивает самый оптимальный порядок сокращений для правила суммирования Эйнштейна путем создания промежуточных массивов.
- `linalg.matrix_power(M, n)` - Возведение матрицы степень указанного целого числа.
- `kron(a, b)` - Произведение Кронекера двух массивов.

Разложения матриц

- `linalg.cholesky(a)` - Разложение Холецкого.
- `linalg.qr(a[, mode])` - QR-разложение матрицы.
- `linalg.svd(a[, full_matrices, compute_uv])` - Сингулярное (SVD) разложение матрицы.

Нормы и прочие числовые характеристики матриц

- `linalg.norm(x[, ord, axis, keepdims])` - Норма матрицы или вектора.
- `linalg.cond(x[, p])` - Число обусловленности матрицы.
- `linalg.det(a)` - Определитель (детерминант) матрицы.
- `linalg.matrix_rank(M[, tol, hermitian])` - Вычисление ранга матрицы на основе метода SVD.
- `linalg.slogdet(a)` - Вычисление знака и натурального логарифма определителя (детерминанта) матрицы.
- `trace(a[, offset, axis1, axis2, dtype, out])` - Суммы диагональных элементов массива.

Собственные значения матриц

- `linalg.eig(a)` - Вычисление собственных значений и правых собственных векторов.
- `linalg.eigh(a[, UPLO])` - Вычисление собственных значений и собственных векторов эрмитовой или вещественной симметричной матрицы.
- `linalg.eigvals(a)` - Вычисление собственных значений матрицы.
- `linalg.eigvalsh(a[, UPLO])` - Вычисление собственных значений матрицы эрмитовой или вещественной симметричной матрицы.

Системы уравнений и обратные матрицы

- `linalg.solve(a, b)` - Решение линейного матричного уравнения.
- `linalg.tensorsolve(a, b[, axes])` - Решение линейного тензорного уравнения.
- `linalg.lstsq(a, b[, rcond])` - Решает задачу поиска наименьших квадратов для линейного матричного уравнения.
- `linalg.inv(a)` - Вычисление обратной матрицы.
- `linalg.pinv(a[, rcond])` - Вычисление псевдообратной (Мура-Пенроуза) матрицы.
- `linalg.tensorinv(a[, ind])` - Вычисление обратного тензора (N-мерного массива).

- **Исключения**
- `linalg.LinAlgError` - Генерация исключений Python вызванные функциями *linalg*.

Ввод и вывод данных

Файлы NumPy (npy, npz)

- `load(file[, mmap_mode, allow_pickle, …])` - Загрузка массивов из файлов формата .npy, .npz, а также pickle объектов и pickle файлов.
- `save(file, arr[, allow_pickle, fix_imports])` - Сохранение массива в .npy двоичном файле.
- `savez(file, *args, **kwds)` - Сохранение нескольких массивов в несжатом .npz файле.
- `savez_compressed(file, *args, **kwds)` - Сохранение нескольких массивов в сжатом .npz файле.

Текстовые файлы

- `loadtxt(fname[, dtype, comments, delimiter, …])` - Загрузка данных из текстового файла.
- `savetxt(fname, X[, fmt, delimiter, newline, …])` - Сохранение массива в текстовом файле.
- `genfromtxt(fname[, dtype, comments, …])` - Загрузка данных из текстового файла с указанием правил обработки отсутствующих и прочих значений.
- `fromregex(file, regexp, dtype[, encoding])` - Создание массива из данных текстового файла с использованием регулярных выражений.
- `fromstring(string[, dtype, count, sep])` - Создание одномерного массива из данных в строке.
- `ndarray.tofile(fid[, sep, format])` - Запись массива в текстовый или двоичный (по умолчанию) файл.
- `ndarray.tolist()` - Представление массива NumPy в виде списка Python.

Необработанные бинарные файлы

- `fromfile(file[, dtype, count, sep])` - Создание массива из данных в текстовом или двоичном файле.
- `ndarray.tofile(fid[, sep, format])` - Запись массива в текстовый или двоичный (по умолчанию) файл.

Строковое представление

- `array2string(a[, max_line_width, precision, …])` - Возвращает строковое представление массива.
- `array_repr(arr[, max_line_width, precision, …])` - Возвращает строковое представление массива вместе с информацией о его типе и типе его данных.
- `array_str(a[, max_line_width, precision, …])` - Возвращает строковое представление массива.
- `format_float_positional(x[, precision, …])` - Возвращает строковое представление числа с плавающей точкой.
- `format_float_scientific(x[, precision, …])` - Возвращает строковое представление числа с плавающей точкой в научной нотации.

Ввод и вывод данных

- **Параметры текстового представления**

- `set_printoptions([precision, threshold, …])` - Позволяет настроить параметры вывода массивов на экран.
- `get_printoptions()` - Возвращает текущие параметры вывода массивов на экран.
- `set_string_function(f[, repr])` - Позволяет задать собственную функцию для вывода массивов на экран.

- **Источники данных**

- `DataSource([destpath])` - Использование локальных и сетевых источников данных по указанному пути или URL-адресу.

- **Системы счисления**

- `binary_repr(num[, width])` - Возвращает строковое представление числа в двоичной системе счисления.
- `base_repr(number[, base, padding])` - Возвращает строковое представление числа в заданной системе счисления.

Работа с индексом

Создание индексных массивов

- `c_` - Преобразует срез в столбец, а последовательности соединяет вдоль второй оси.
- `r_` - Преобразует срез в строку, а последовательности соединяет вдоль первой оси
- `s_` - Создает объект среза.
- `nonzero(a)` - Возвращает индексы ненулевых элементов массива.
- `where(condition, [x, y])` - Возвращает элементы, которые могут выбираться из двух массивов в зависимости от условия.
- `indices(dimensions[, dtype])` - Возвращает координатную сетку для пространства заданной размерности и заданного размера.
- `ix_(*args)` - Возвращает открытую координатную сетку созданную из одномерных массивов.
- `ogrid` - Возвращает открытую координатную сетку значений.
- `ravel_multi_index(multi_index, dims[, mode, …])` - Преобразует кортеж массивов индексов в плоский массив индексов.
- `unravel_index(indices, shape[, order])` - Преобразует плоский массив индексов в кортеж массивов индексов.

Создание индексных массивов

- `diag_indices(n[, ndim])` - Возвращает индексы элементов главной диагонали квадратного массива заданного размера и размерности.
- `diag_indices_from(arr)` - Возвращает индексы элементов главной диагонали указанного массива.
- `mask_indices(n, mask_func[, k])` - Индексы элементов квадратного массива выбранные по некоторому условию (маске массива).
- `tril_indices(n[, k, m])` - Возвращает индексы элементов нижнего треугольника массива указанного размера.
- `tril_indices_from(arr[, k])` - Возвращает индексы элементов нижнего треугольника указанного массива.
- `triu_indices(n[, k, m])` - Возвращает индексы элементов верхнего треугольника массива указанного размера.
- `triu_indices_from(arr[, k])` - Возвращает индексы элементов верхнего треугольника указанного массива.

Индексоподобные операции

- `take(a, indices[, axis, out, mode])` - Возвращает элементы массива с указанными индексами вдоль указанной оси.
- `take_along_axis(arr, indices, axis)` - Сопоставляет одномерные массивы индексов с соответствующими полными срезами исходного массива вдоль указанной оси и возвращает найденные элементы.
- `choose(a, choices[, out, mode])` - Создает массив на основе индексного массива из набора массивов.
- `compress(condition, a[, axis, out])` - Возвращает указанные срезы массива вдоль заданной оси.
- `diag(v[, k])` - Извлекает диагональ из массива, а так же позволяет строить диагональные массивы.
- `diagonal(a[, offset, axis1, axis2])` - Возвращает указанные диагонали массива.
- `select(condlist, choicelist[, default])` - Возвращает массив составленный из элементовдругих массивов, которые выбираются из них в зависимости от указанного условия.
- `lib.stride_tricks.as_strided(x[, shape, …])` - Возвращает представление массива с указанной формой и смещением байтов между элементами для перехода по ним вдоль разных осей.

Вставка данных в массив

- `place(arr, mask, vals)` - Изменяет элементы массива на указанные значения в зависимости от выполнения заданных условий.
- `put(a, ind, v[, mode])` - Заменяет элементы массива по указанному индексу заданными значениями.
- `put_along_axis(arr, indices, values, axis)` - Сопоставляет одномерные массивы индексов с соответствующими полными срезами исходного массива вдоль указанной оси и вставляет туда указанные значения.
- `putmask(a, mask, values)` - Изменяет элементы массива на указанные значения в зависимости от выполнения заданных условий.
- `fill_diagonal(a, val[, wrap])` - Изменяет элементы главной диагонали на указанное значение в массиве любой формы и любой размерности.

Итерирование массивов

- `ndenumerate(arr)` - Итератор, возвращающий пары с индексом элементов в массиве и их значением.
- `ndindex(*shape)` - Итератор, возвращающий индексы элементов в массиве заданной формы.
- `flatiter` - Итератор, возвращающий элементы сжатого до одной оси массива.

Сортировка, поиск, подсчет

Сортировка

- `sort(a[, axis, kind, order])` - Возвращает отсортированную копию массива.
- `lexsort(keys[, axis])` - Выполняет обратную устойчивую сортировку на основе указанных ключей (лексикографическая сортировка).
- `argsort(a[, axis, kind, order])` - Возвращает индексы, сортирующие элементы исходного массива.
- `ndarray.sort([axis, kind, order])` - Функция sort() реализованная в виде метода базового класса ndarray.
- `msort(a)` - Возвращает отсортированную по первой оси копию массива.
- `sort_complex(a)` - Возвращает отсортированную по последней оси копию массива комплексных чисел.
- `partition(a, kth[, axis, kind, order])` - Возвращает копию исходного массива элементы которого разделены по указанному значению.
- `argpartition(a, kth[, axis, kind, order])` - Возвращает массив индексов элементов исходного массива в их разбиении по указанному значению.

Поиск

- `argmax(a[, axis, out])` - Возвращает индекс максимального значения вдоль указанной оси.
- `nanargmax(a[, axis])` - Возвращает индекс максимального значения вдоль указанной оси с игнорированием значений *nan*
- `argmin(a[, axis, out])` - Возвращает индекс минимального значения вдоль указанной оси.
- `nanargmin(a[, axis])` - Возвращает индекс минимального значения вдоль указанной оси с игнорированием значений *nan*.
- `argwhere(a)` - Возвращает индексы ненулевых элементов указанного массива.
- `nonzero(a)` - Возвращает индексы ненулевых элементов массива.
- `flatnonzero(a)` - Возвращает индексы ненулевых элементов в сжатом до одной оси представлении указанного массива.
- `where(condition, [x, y])` - Возвращает элементы, которые могут выбираться из двух массивов в зависимости от условия.

Поиск, Подсчет

- `searchsorted(a, v[, side, sorter])` - Возвращает индексы в которые должны быть вставлены указанные элементы, что бы порядок сортировки был сохранен.
- `extract(condition, arr)` - Возвращает элементы массива, которые удовлетворяют указанному условию.

- **Подсчет**
- `count_nonzero(a[, axis])` - Возвращает количество ненулевых элементов массива вдоль указанной оси.

Белорусско-Российский университет
Кафедра «Программное обеспечение информационных технологий»

Информатика. Программирование на Python
Тема: Python. Основы. Массивы / Списки. Библиотека NumPy

Благодарю за внимание

КУТУЗОВ Виктор Владимирович

Список использованных источников

1. Python
   https://www.python.org/
2. Google Colaboratory
   https://colab.research.google.com/
3. Официальный сайт библиотеки NumPy
   https://numpy.org/
4. Руководство пользователя NumPy
   https://numpy.org/devdocs/user/index.html
5. NumPy Справочное руководство
   https://pyprog.pro/reference_manual.html
6. NumPy справочное руководство. Математические функции
   https://pyprog.pro/mathematical_functions/mathematical_functions.html
7. NumPy справочное руководство. Создание массивов
   https://pyprog.pro/array_creation/array_creation_functions.html
8. NumPy справочное руководство. Операции над массивами
   https://pyprog.pro/array_manipulation/array_manipulation_functions.html
9. NumPy справочное руководство. Генерация случайных данных
   https://pyprog.pro/random_sampling_functions/random_sampling_functions.html

Список использованных источников

10. NumPy справочное руководство. Статистика
https://pyprog.pro/statistics_functions/statistics_function.html

11. NumPy справочное руководство. Ввод и вывод данных
https://pyprog.pro/io_functions/io_functions.html

12. NumPy справочное руководство. Работа с индексом
https://pyprog.pro/indexing_routines/indexing_routines.html

13. NumPy справочное руководство. Сортировка, поиск, подсчет
https://pyprog.pro/sort/sort_func.html

14. NumPy Manual
https://numpy.org/doc/stable/

15. An introduction to Numpy and Scipy
https://sites.engineering.ucsb.edu/~shell/che210d/numpy.pdf

16. Хабр. NumPy в Python. Часть 1
https://habr.com/ru/post/352678/

17. Базовые математические функции
https://proproprogs.ru/modules/numpy-bazovye-matematicheskie-funkcii

18. 图解NumPy，这是理解数组最形象的一份教程了 // Graphic NumPy, это самый яркий туториал по пониманию массивов https://blog.csdn.net/zouxiaolv/article/details/95617537

19. Python数据分析：Numpy // Анализ данных Python: Numpy
https://blog.csdn.net/pcn01/article/details/103575974

Список использованных источников

20. A Visual Intro to NumPy and Data Representation
https://jalammar.github.io/visual-numpy/

21. Matrix Operations with Python and Numpy
https://www.nebomusic.net/perception/Matrix_Operations_Python_Numpy.pdf