Лекция 3. Инструменти за Data Science с езика Python - numpy, cupy, scipy, matplotlib.

DTSC001 „Прогнозиране чрез анализ на данни I“

ИНСТИТУТ *за* СЪВРЕМЕННИ ФИЗИЧЕСКИ ИЗСЛЕДВАНИЯ

НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ

Стоян Мишев, Лъчезар Петров, Петър Христов

`https://facebook.com/nbudatascience`

План

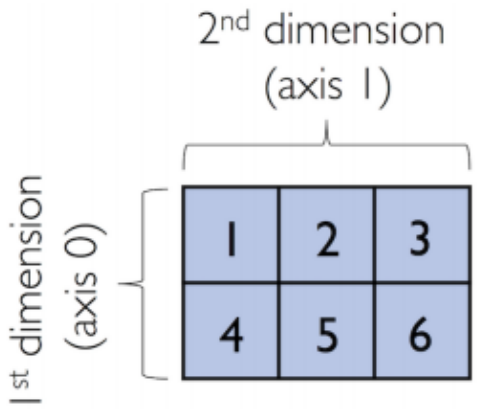
Numpy

CuPy

Matplotlib

Numpy

```
>>> import numpy as np
>>> lst = [[1, 2, 3], [4, 5, 6]]
>>> ary2d = np.array(lst)
>>> ary2d
array([[1, 2, 3],
       [4, 5, 6]])
```




```
>>> ary2d.dtype
dtype('int64')
```

```
>>> float32_ary = ary2d.astype(np.float32)
>>> float32_ary
array([[1., 2., 3.],
[4., 5., 6.]], dtype=float32)
>>> float32_ary.dtype
dtype('float32')

>>> ary2d.size
6
>>> ary2d.ndim
2
>>> ary2d.shape
(2, 3)
>>> np.array([1, 2, 3]).shape
(3,)
```

Numpy - dtype, size, ndim, shape, nbytes

```
1 arr = np.arange(8).reshape(2,4) #  the total number of
      elements is unchanged
2 print(arr)
3
4 print('Data type                    :', arr.dtype)
5 print('Total number of elements     :', arr.size)
6 print('Number of dimensions         :', arr.ndim)
7 print('Shape (dimensionality)       :', arr.shape)
8 print('Memory used (in bytes)       :', arr.nbytes)
```

```
>>> np.ones((3, 3))
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
>>> np.zeros((3, 3))
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
>>> np.eye(3)
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
>>> np.diag((3, 3, 3))
array([[3, 0, 0],
       [0, 3, 0],
       [0, 0, 3]])
```

Създаване на NumPy масиви 2

```
1 np.zeros(5, float)
2 np.zeros(3, int)
3 np.zeros(3, complex)
4 print('5 ones:', np.ones(5))
5 a = np.empty(4) # random
6 a.fill(5.5)
```

```
>>> np.arange(4., 10.)
array([4., 5., 6., 7., 8., 9.])

>>> np.arange(5)
array([0, 1, 2, 3, 4])

>>> np.arange(1., 11., 2)
array([1., 3., 5., 7., 9.])

>>> np.linspace(0., 1., num=5)
array([0. , 0.25, 0.5 , 0.75, 1. ])
```

Двумерен масив

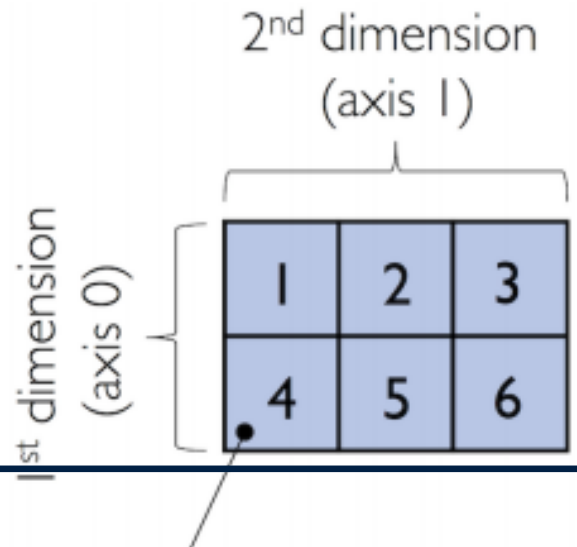
```
1 np.arange(1, 100, 5)
2 print("A linear grid between 0 and 1:", np.linspace(0,
      1, 5))
3 print("A logarithmic grid between 10**1 and 10**4: ",
      np.logspace(1, 4, 4))
4 np.random.randn(5) # standard normal distribution
5 np.random.normal(10, 3, 5)
6
7 lst2 = [[1, 2], [3, 4]]
8 arr2 = np.array([[1, 2], [3, 4]])
9 print(lst2[0][1])
10 print(arr2[0,1])
11
12 np.array([[1,2,3],[4,5,6]], order='F')
13 np.zeros((2,3))
14 np.random.normal(10, 3, (2, 4))
```

Намерете сумата на матриците

$$\begin{pmatrix} 1 & 2 & 3 \\ 6 & 8 & 9 \end{pmatrix} + \begin{pmatrix} 3 & 5 & 6 \\ 7 & 4 & 2 \end{pmatrix}$$

използвайки `numpy`.

```
>>> ary = np.array([1, 2, 3])
>>> ary[0]
1
>>> ary[:2] # equivalent to ary[0:2]
array([1, 2])
>>> ary = np.array([[1, 2, 3],
... [4, 5, 6]])
>>> ary[0, 0] # upper left
1
```



```
>>> ary[-1, -1] # lower right
6
>>> ary[0, 1] # first row, second column
2
>>> ary[0] # entire first row
array([1, 2, 3])
>>> ary[:, 0] # entire first column
array([1, 4])
>>> ary[:, :2] # first two columns
array([[1, 2],
[4, 5]])
>>> ary[0, 0]
1
```

```
print('Slicing in the second row:', arr[1, 2:4])
print('All rows, third column   :', arr[:, 2])
print('First row:  ', arr[0])
print('Second row: ', arr[1])
```

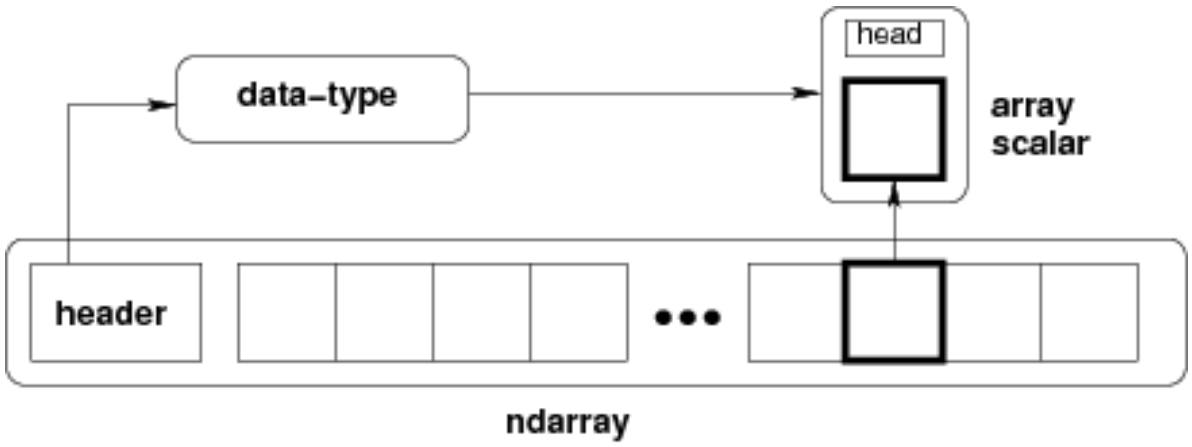


Рис.: https://docs.scipy.org/doc/numpy-1.13.0/reference/arrays.html

B Python:

```
>>> lst = [[1, 2, 3], [4, 5, 6]]
>>>   for row_idx, row_val in enumerate(lst):
...     for col_idx, col_val in enumerate(row_val):
...         lst[row_idx][col_idx] += 1
>>> lst
[[2, 3, 4], [5, 6, 7]]
>>> lst = [[1, 2, 3], [4, 5, 6]]
>>> [[cell + 1 for cell in row] for row in lst]
```

B numpy

```
>>> ary = np.array([[1, 2, 3], [4, 5, 6]])
>>> ary = np.add(ary, 1)
>>> ary
array([[2, 3, 4],
       [5, 6, 7]])
```

add, subtract, divide, multiply, and exp

Кратък запис

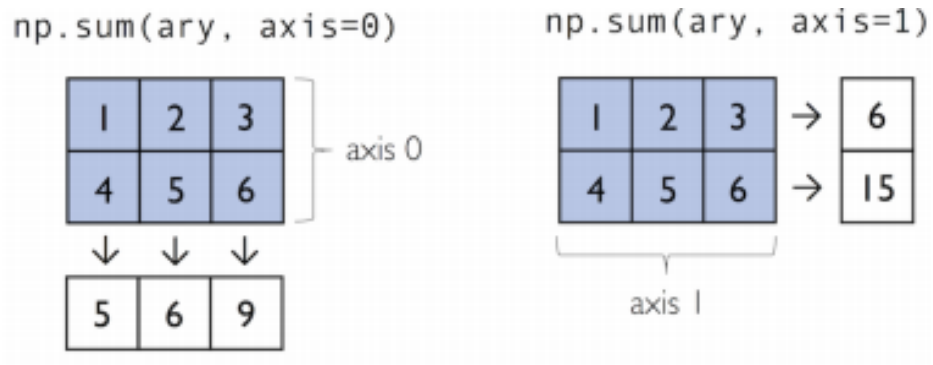
```
>>> ary + 1
array([[3, 4, 5],
       [6, 7, 8]])
>>> ary**2
array([[ 4, 9, 16],
       [25, 36, 49]])
```

Сумиране по колони и редове

```
>>> ary = np.array([[1, 2, 3],
...                 [4, 5, 6]])
>>> np.add.reduce(ary) # column sumns
array([5, 7, 9])
>>> np.add.reduce(ary, axis=1) # row sums
array([ 6, 15])
```

Кратък запис

```
>>> ary.sum(axis=0) # column sums
array([5, 7, 9])
>>> ary.sum()
21
```



Numpy - axis

```
1 print('For the following array:\n', arr)
2 print('The sum of elements along the rows is:', arr.sum
      (axis=1))
3 print('The sum of elements along the columns is:', arr.
      sum(axis=0))
4
5 np.zeros((3,4,5,6)).sum(2).shape
6
7 print('Array:\n', arr)
8 print('Transpose:\n', arr.T)
```

- mean (computes arithmetic average)
- std (computes the standard deviation)
- var (computes variance)
- sort (sorts an array)
- argsort (returns indices that would sort an array)
- min (returns the minimum value of an array)
- max (returns the maximum value of an array)
- argmin (returns the index of the minimum value)
- argmax (returns the index of the maximum value)
- array equal (checks if two arrays have the same shape and elements)

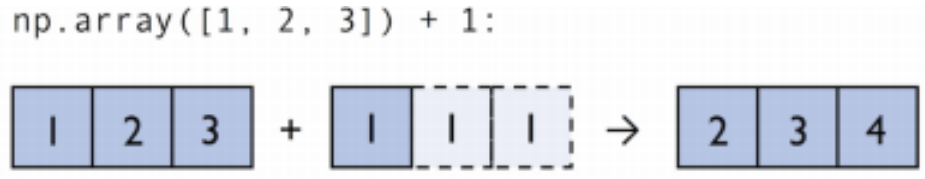
Numpy - min, max, prod, mean, std

```
print('Minimum and maximum :', arr.min(), arr.max())
print('Sum and product of all elements :', arr.sum(),
    arr.prod())
print('Mean and standard deviation  :', arr.mean(), arr
    .std())
```

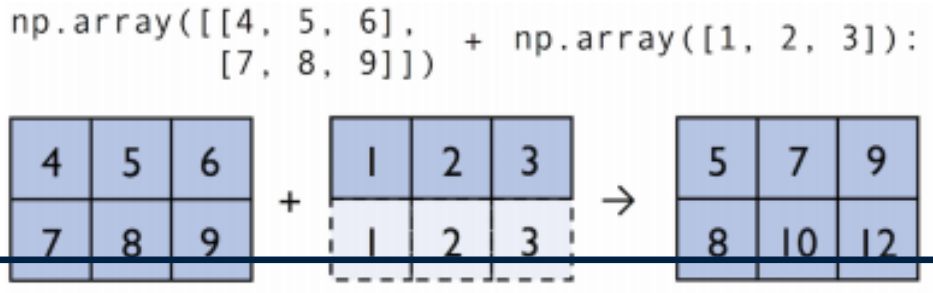
Чрез numpy да се създаде матрицата
$\begin{pmatrix} 3 & 5 & 6 & 9 & 0 & 15 \\ 7 & 4 & 2 & 6 & 21 & 5 \end{pmatrix}$
и да се намерят:

- най-малките елементи от всеки ред, както и техните индекси ;
- дисперсията на всеки ред

Broadcasting




```
>>> ary1 = np.array([1, 2, 3])
>>> ary2 = np.array([4, 5, 6])
>>> ary1 + ary2
array([5, 7, 9])
>>> ary3 = np.array([[4, 5, 6],
...                  [7, 8, 9]])
>>> ary3 + ary1 # similarly, ary1 + ary3
array([[ 5, 7, 9],
       [ 8, 10, 12]])
```



Views

```
>>> ary = np.array([[1, 2, 3],
...                 [4, 5, 6]])
>>> first_row = ary[0]
>>> first_row += 99
>>> ary
array([[100, 101, 102],
       [ 4, 5, 6]])
```

Views

```
>>> ary = np.array([[1, 2, 3],
... [4, 5, 6]])
>>> center_col = ary[:, 1]
>>> center_col += 99
>>> ary
array([[ 1, 101, 3],
[ 4, 104, 6]])
```

Copy

```
>>> ary = np.array([[1, 2, 3],
... [4, 5, 6]])
>>> second_row = ary[1].copy()
>>> second_row += 99
>>> ary
array([[1, 2, 3],
[4, 5, 6]])
```

Индексиране

```
>>> ary = np.array([[1, 2, 3],
...                 [4, 5, 6]])
>>> ary[:, [0, 2]] # first and and last column
array([[1, 3],
       [4, 6]])
```

```
>>> this_is_a_copy = ary[:, [0, 2]]
>>> this_is_a_copy += 99
>>> ary
array([[1, 2, 3],
       [4, 5, 6]])
>>> ary[:, [2, 0]] # first and and last column
array([[3, 1],
       [6, 4]])
```

```
>>> ary = np.array([[1, 2, 3],
... [4, 5, 6]])
>>> greater3_mask = ary > 3
>>> greater3_mask
array([[False, False, False],
[ True, True, True]])

>>> ary[greater3_mask]
array([4, 5, 6])

>>> ary[(ary > 3) & (ary % 2 == 0)]
array([4, 6])
```

```
>>> np.random.seed(123)
>>> np.random.rand(3)
array([0.69646919, 0.28613933, 0.22685145])

>>> rng1 = np.random.RandomState(seed=123)
>>> rng1.rand(3)
array([0.69646919, 0.28613933, 0.22685145])
```

```
>>> ary1d = np.array([1, 2, 3, 4, 5, 6])
>>> ary2d_view = ary1d.reshape(2, 3)
>>> ary2d_view
array([[1, 2, 3],
       [4, 5, 6]])
>>> np.may_share_memory(ary2d_view, ary1d)
True

>>> ary1d.reshape(2, -1)
array([[1, 2, 3],
[4, 5, 6]])
>>> ary1d.reshape(-1, 2)
array([[1, 2],
[3, 4],
[5, 6]])
```

```
>>> ary = np.array([[[1, 2, 3],
...                  [4, 5, 6]]])
>>> ary.reshape(-1)
array([1, 2, 3, 4, 5, 6])

>>> ary = np.array([1, 2, 3])
>>> # stack along the first axis
>>> np.concatenate((ary, ary))
array([1, 2, 3, 1, 2, 3])

>>> ary = np.array([[1, 2, 3]])
>>> # stack along the first axis (here: rows)
>>> np.concatenate((ary, ary), axis=0)
array([[1, 2, 3],
[1, 2, 3]])
```

```
>>> # stack along the second axis (here: column)
>>> np.concatenate((ary, ary), axis=1)
array([[1, 2, 3, 1, 2, 3]])
>>> ary = np.array([[1, 2, 3]])

>>> # stack along the first axis (here: rows)
>>> np.concatenate((ary, ary), axis=0)
array([[1, 2, 3],
       [1, 2, 3]])
```

```
>>> ary = np.array([1, 2, 3, 4])
>>> mask = ary > 2
>>> mask
array([False, False, True, True])
>>> ary[mask]
array([3, 4])
>>> mask
array([False, False, True, True])
>>> mask.sum()
2

>>> np.where(ary > 2, 1, 0)
array([0, 0, 1, 1])
```

Comparison Operators and Masks

```
>>> ary = np.array([1, 2, 3, 4])
>>> mask = ary > 2
>>> ary[mask] = 1
>>> ary[~mask] = 0
>>> ary
array([0, 0, 1, 1])
```

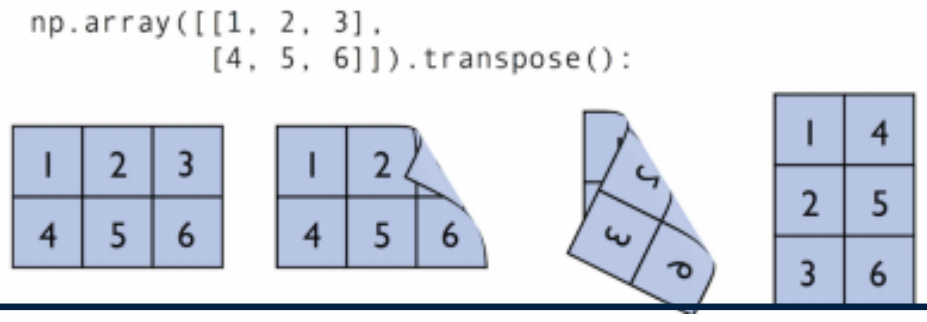
- A: & or np.bitwise and
- Or: | or np.bitwise or
- Xor: ^ or np.bitwise xor
- Not: ~ or np.bitwise not

```
>>> ary = np.array([1, 2, 3, 4])
>>> (ary > 3) | (ary < 2)

>>> ~((ary > 3) | (ary < 2))
array([False, True, True, False])
```

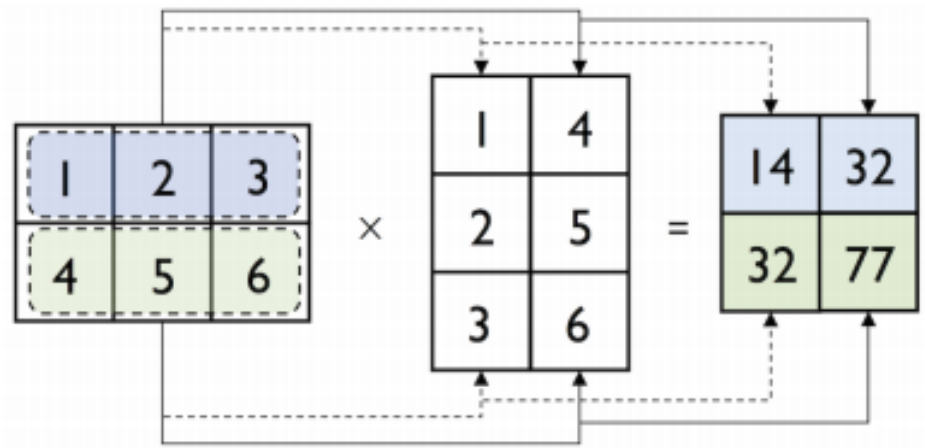
```
>>> row_vector = np.array([1, 2, 3])
>>> row_vector
array([1, 2, 3])

>>> column_vector = np.array([[1, 2, 3]]).reshape(-1, 1)
>>> column_vector
array([[1],
[2],
[3]])

>>> row_vector[:, np.newaxis]
array([[1],
[2],
[3]])
```

```
>>> matrix = np.array([[1, 2, 3],
... [4, 5, 6]])
>>> np.matmul(matrix, column_vector)
array([[14],
[32]])
>>> np.matmul(matrix, row_vector)
array([14, 32])
>>> np.matmul(row_vector, row_vector)
14

>>> np.dot(row_vector, row_vector)
14
>>> np.dot(matrix, row_vector)
array([14, 32])
```

```
>>> np.dot(matrix, column_vector)
array([[14],
       [32]])

>>> matrix = np.array([[1, 2, 3],
...                    [4, 5, 6]])
>>> matrix.transpose()
array([[1, 4],
    [2, 5],
    [3, 6]])
```



```
>>> np.matmul(matrix, matrix.transpose())
array([[14, 32],
[32, 77]])
```




```
>>> matrix.T
array([[1, 4],
[2, 5],
[3, 6]])
```

Има и специален тип matrix .

Numpy. Матрично умножение.

```
1  v1 = np.array([2, 3, 4])
2  v2 = np.array([1, 0, 1])
3  print(v1, '.', v2, '=', v1.dot(v2))
4  print(v1, '.', v2, '=', v1 @ v2)
5
6  A = np.arange(6).reshape(2, 3)
7  print(A, 'x', v1, '=', A @ v1)
8
9  print(A @ A.T)
10
11 print(A.T @ A)
12
13 print(v1, '.', v2, '=', v1 @ v2)
```

Да се умножи матрицата

$$\begin{pmatrix} 3 & 5 & 6 & 9 & 0 & 15 \\ 7 & 4 & 2 & 6 & 21 & 5 \end{pmatrix}$$

с 2×2 матрица с елементи, които са случайни числа

https://www.w3resource.com/python-exercises/numpy/linear-algebra

CuPy

https://cupy.dev/

CuPy

https://cupy.dev/
https://docs.cupy.dev/en/stable/user_guide/basic.html

https://gist.github.com/fukatani/
4702aa05aed255cd25f42e77d0a22e37

Matplotlib

Matplotlib (https://matplotlib.org/)

```
1  import matplotlib.pyplot as plt
2
3  x = np.linspace(0, 2*np.pi)
4  y = np.sin(x)
5  plt.figure()
6  plt.plot(x,y, label='sin(x)')
7  plt.legend()
8  plt.grid()
9  plt.title('Harmonic')
10 plt.xlabel('x')
11 plt.ylabel('y')
12
13 plt.plot(x, y, linewidth=2)
14
15 plt.plot(x, y, 'o', markersize=5, color='r');
```

```
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, normed=1, facecolor=
    'g', alpha=0.75)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
# This will put a text fragment at the position given:
plt.text(55, .027, r'$\mu=100,\ \sigma=15$', fontsize
    =14)
plt.axis([40, 160, 0, 0.03])
plt.grid()
```

```
1 from matplotlib import cm
2
3 plt.imshow(np.random.rand(5, 10), cmap=cm.gray,
      interpolation='nearest');
4
5 img = plt.imread('stinkbug.png')
6 print('Dimensions of the array img:', img.shape)
7 plt.imshow(img);
```

```
1 from mpl_toolkits.mplot3d.axes3d import Axes3D
2 from matplotlib import cm
3
4 fig = plt.figure()
5 ax = fig.add_subplot(1, 1, 1, projection='3d')
6 X = np.arange(-5, 5, 0.25)
7 Y = np.arange(-5, 5, 0.25)
8 X, Y = np.meshgrid(X, Y)
9 R = np.sqrt(X**2 + Y**2)
10 Z = np.sin(R)
11 surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1,
       cmap=cm.viridis,
12           linewidth=0, antialiased=False)
13 ax.set_zlim3d(-1.01, 1.01);
```