

Библиотека numpy

numpy - библиотека языка Python ориентирована на работу с многомерными массивами и матрицами, с полиномами и с другими объектами.

Основным объектом numpy является однородный многомерный массив элементов одного типа (называется `numpy.ndarray`).

Некоторые атрибуты `ndarray`

- `ndarray.ndim` - число измерений(принято называть осями) массива.
- `ndarray.shape` - кортеж натуральных чисел, показывающий длину массива по каждой оси. Число элементов кортежа `shape` равно `ndim`.
- `ndarray.dtype` - тип элементов массива. numpy имеет собственные типы данных, например: `float32`, `complex64` и т. д.
- `ndarray.itemsize` - размер каждого элемента в байтах.

```
import numpy
import numpy as np
from numpy import *
```

Создание массивов

```
import numpy as np
```

```
# Создание массивов
```

```
# Одномерный массив
```

```
a = np.array((2, -3, 1, 9))
```

```
print(type(a))
```

```
print(a)
```

```
print(a.shape)
```

```
print(a.dtype, '\n')
```

```
b = np.array([-7, 7, 8, 4], dtype=float)
```

```
print(b)
```

```
print(b.shape)
```

```
print(b.dtype, '\n')
```

```
print('input array')
```

```
c = np.array(list(map(float, input().split())))
```

```
print(c, '\n\n\n')
```

Создание двумерного массива

Двумерный массив

```
a = np.array([[1, 2, 3], [9, 8, 7]], 'int64')
print(type(a))
print(a)
print (a.shape)
print (a.dtype,'\n\n')
```

Создание матрицы из списка и обратная операция

```
lst = [[4, 5, 6], [7, 8, 9]]
print(lst)
d = np.array(lst)
print(d)
d1 = d.tolist()
print(d1)
```

Формирование одномерных массивов. Функция `arange()`

`numpy.arange([start,]stop, [step,]dtype=None)` -- возвращает массив (`numpy.ndarray`) чисел, равномерно распределенных в заданном интервале.

`start` -- начало интервала,
`stop` -- конец интервала,
`step` -- шаг,
`dtype` -- тип данных.

```
a = np.arange(10)
print(a)
```

```
b = np.arange(2, 10)
print(b)
```

```
c = np.arange(10, 2, -2)
print(c, '\n\n')
```

Формирование одномерных массивов. Функция `linspace()`

`numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0)` -- возвращает массив (`numpy.ndarray`) N чисел, равномерно распределенных в заданном интервале.

`start` - начало интервала

`stop` - конец интервала

`num` - число элементов

`endpoint` - если `True`, то `stop` включается в массив

`retstep` - если `True`, то функция вернет кортеж вида (`values`, `step`)

```
a = np.linspace(0, 2, 5)
```

```
print(a)
```

```
b = np.linspace(0, 2, 5, endpoint=False)
```

```
print(b)
```

```
c = np.linspace(0, 5)
```

```
print(c)
```

Настройка печати

`set_printoptions(edgeitems=3, infstr='inf', linewidth=75, nanstr='nan', precision=8, suppress=False, threshold=1000, formatter=None)` -- позволяет настроить параметры вывода массивов на экран.

`edgeitems` - количество элементов в начале и в конце каждой размерности,

`infstr` - строковое представление `inf`,

`linewidth` - число символов в строке, после которых производится перенос,

`nanstr` - строковое представление `NaN`,

`precision` - число цифр после запятой,

`suppress` - если `True`, не печатает маленькие значения в научной нотации,

`threshold` - число элементов в массиве, вызывающее обрезание,

`formatter` - позволяет более тонко управлять печатью массива.

Примеры

```
np.set_printoptions(edgeitems=7, linewidth=30)
a = np.arange(1001)
print(a, '\n')
```

```
#np.set_printoptions(threshold=100000)
#b = np.arange(2001)
#print(b)
```

```
np.set_printoptions(precision=3)
c = np.linspace(1, 5, 8)
print(c, '\n')
```

```
d = np.array([2.3, 2e-29, 0])
print(d, '\n')
```

```
np.set_printoptions(suppress=True)
print(d, '\n')
```

Матрицы специального вида

```
# Пустая матрица  
a = np.empty((3, 2))  
print(a, '\n')
```

```
# Единичная матрица -- 1 на главной диагонали, 0 -- остальные.  
b = np.identity(4)  
print(b, '\n')
```

```
# Нулевая матрица  
c = np.zeros((4, 5), int)  
print(c, '\n')
```

```
# Матрица из единиц  
d = np.ones((4, 3), 'int64')  
print(d, '\n')
```

```
#  
f = np.full((3, 3), 7)  
print(f, '\n')
```


Еще немного функций

```
import numpy as np
```

```
a = np.arange(12)  
print('Формирование одномерного массива\n',a,'\n')
```

```
b = np.reshape(a,(2,6))  
print('Изменение формы массива\n',b,'\n')
```

```
c = np.resize(a,(2,7))  
print('Изменение формы массива при',  
      'несовпадении числа элементов\n',c,'\n')
```

```
print('Копирование массивов')
d1 = np.copy(a)
print(d1)
d2 = np.copy(b)
print(d2, '\n')
```

```
d2.shape=(2,2,3)
print('Формирование трёхмерного массива из двухмерного\n',d2, '\n')
```

```
print('Печать одномерного массива')
for i in a:
    print (i)
```

```
print('\n')
print('Печать трёхмерного массива')
for i in d2:
    for j in i:
        print (j)
```

```
a = np.arange(12)
a1 = np.copy(a)
print('Исходная матрица')
a2 = np.reshape(a1,(3,4))
print(a2,\n')
```

```
a2 = a2.T
print('Транспонированная матрица')
print(a2,\n')
```

```
# min, max, sum, сортировка
b = np.array([[2, 8, 0], [6, 1, 3], [4, 7, 5]])
print('Новая исходная матрица\n',b, '\n')
```

```
dsum = b.sum()
dmin = b.min()
dmax = b.max()
print('Некоторые значения для всей матрицы')
print('sum=', dsum, ' min=', dmin, ' max=',dmax,\n')
```

```
mincol = b.min(axis=0)
maxrow = b.max(axis=1)
print('Значения min и max для столбцов и строк')
print('min в столбцах = ', mincol, ' max в строках = ',maxrow, '\n')
```

Сортировка

```
sort(axis=-1, kind='quicksort', order=None)
```

axis - ось, по которой идёт сортировка. По умолчанию по последней оси

kind - тип сортировки. Возможные значения 'quicksort', 'mergesort', 'heapsort'

```
c = b.copy()
```

```
c.sort(axis=0, kind='mergesort')
```

```
print('сортировка столбцов\n', c, '\n')
```

```
c = b.copy()
```

```
c.sort(axis=1)
```

```
print('сортировка строк\n', c, '\n')
```

```
dtype = [('weight', int), ('height', float), ('age', int)]
```

```
v = [(70, 1.75, 15), (65, 1.8, 19), (45, 1.85, 16)]
```

```
c = np.array(v, dtype=dtype)
```

```
c.sort(axis=0, order = ['weight'])
```

```
print(c)
```

Матричные операции

```
for i in range(1,15):  
    print('\n')
```

```
print('Формирование исходных массивов')  
a = np.arange(1, 5).reshape(2, 2)  
b = np.arange(5, 9).reshape(2, 2)  
print(a)  
print(b)
```

```
print('\nСложение')  
c_plus = a+b  
print(c_plus)
```

```
print('\nВычитание')  
c_minus = a-b  
print(c_minus)
```

```
print('\nУмножение поэлементное')  
c_mult = a*b  
print(c_mult)
```

```
print('\nВозведение в степень')  
c_power = a**b  
print(c_power)
```

```
print('\nУмножение на число')  
c_num = a*5  
print(c_num)
```

```
print('\nМатричное умножение 2*2')  
c_multmat = np.dot(a, b)  
print(c_multmat)
```

```
print('\nМатричное умножение 2*3')  
x = np.arange(1, 7).reshape(2,3)  
a_x = np.dot(a, x)  
print(a_x)
```

```
print('деление на ноль')  
a0 = np.arange(0, 4).reshape(2, 2)  
a_div = a/a0  
print(a_div)
```

```
print('Остаток от деления')
a_remainder = a%a0
print(a_remainder)
print('\n\n')
```

```
# Другой способ
d=np.add(a,b)
print(d)
```

```
print('Вычитание subtract')
e = np.subtract(a,b)
print(e,'\n')
```

```
print('Умножение поэлементное multiply')
f = np.multiply(a,b)
print(f,'\n')
```

```
print('Деление divide')
g = np.divide(a,b)
print(g,'\n')
```

```
print('Матричное умножение dot')
h = np.dot(a,b)
```

```
print(h,'\n')
```

```
print('Смена знака')  
m = np.negative(a)  
print(m,'\n')
```

```
print('Транспонирование')  
n = np.transpose(a)  
print(n,'\n')
```