



NumPy

ЛЕКЦИЯ 9



Содержание

- ❑ **SciPy и NumPy**
- ❑ Типы в **NumPy**
- ❑ N-мерный массив **ndarray**
- ❑ Инициализация N-мерных массивов
- ❑ Свойства N-мерных массивов
- ❑ Специальные методы N-мерных массивов



SciPy



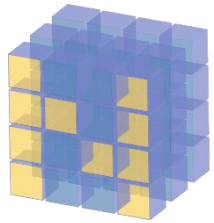
SciPy – библиотека готовых решений с открытым кодом реализованная на **Python**. Данный пакет был разработан для выполнения научных и инженерных расчётов.

Пакет **SciPy** включает:

- ☐ **NumPy** – модуль работы с N-мерными массивами
- ☐ **Matplotlib** – модуль для визуального представления результатов (графики, диаграммы и т.п.)
- ☐ **Sympy** – модуль реализации символьной математики
- ☐ И т.д.



Numpy



NumPy

NumPy – модуль работы с N – мерными массивами.

Данный модуль включает последовательность нового типа-
ndarray

ndarray – это последовательность элементы которой строго типизированны и относятся к классу чисел. Данная последовательность относится к изменяемым.

ndarray может быть получена с помощью метода модуля numpy – `array()` из любого списка содержащего элементы одного **типа** (см. далее)



Типы в NumPy

NumPy поддерживает числовые объекты различных типов, аналогичных с++.

Примечание: дело в том что множество готовых решений numpy используют скомпилированный код C++

К ним относятся:

bool_	логическое (True или False) записывается как байт
intc	int аналогичный int в Си
int8	байт (от -128 до 127)
int32	целое (от -2147483648 до 2147483647)
int64	целое (от -9223372036854775808 до 9223372036854775807)
uint8	цело без знаковое (от 0 до 255)
uint32	цело без знаковое (от 0 до 4294967295)



Типы в NumPy

uint64	целое без знаковое (от 0 до 18446744073709551615)
float16	float длиной в 16 бит
float32	float длиной в 32 бита
float64	float длиной в 64 бита
complex64	комплексное число, состоящее из двух float32
complex128	комплексное число, состоящее из двух float64

Int_, float_, complex_ представляют собой максимально длинные типы из своего вида.

Для использования специальных типов следует вызывать их из **numpy**:

```
1 import numpy as np
2
3 a = np.int_(6)
4 print(type(a))
```

```
<class 'numpy.int32'>
```

```
Process finished with exit code 0
```

Инициализация N-мерного массива (ndarray)



Метод `numpy.array()` позволяет преобразовать список из элементов одного типа в **ndarray**

```
import numpy as np

z = np.array([0,1,2])
print(z)
```

```
[0 1 2]
```

```
Process finished with exit code 0
```

Если преобразовать список с помощью любого метода приведения **numpy** то так же получится **ndarray** из элементов нужного типа:

```
import numpy as np

z = np.float32([0,1,2])
print(type(z), z)
```

```
<class 'numpy.ndarray'> [0. 1. 2.]
```

```
Process finished with exit code 0
```



Инициализация N-мерного массива (ndarray)

Метод **ndarray.astype()** позволяет конвертировать элементы массива в элементы указанного типа. При этом старый массив останется без изменения, а новый можно получить путем присвоения

```
z = np.array([0., 1., 2.])
c = z.astype(np.int8)
print(type(c), c)
```

```
<class 'numpy.ndarray'> [0 1 2]

Process finished with exit code 0
```

Поле **ndarray.dtype** содержит информацию о типах элемента массива. Указать тип массива можно в методе **numpy.array()** передав его для поля **dtype**:

```
z = np.array([0., 1., 2.])
print(z.dtype)
```

```
float64

Process finished with exit code 0
```

```
z = np.array([0., 1., 2.], dtype=np.int_)
print(z, z.dtype)
```

```
[0 1 2] int32

Process finished with exit code 0
```




numpy.arange

Данный метод аналогичен стандартному генератору `range()`, однако может работать с float аргументами:

```
z = np.arange(1, 600, 0.1)
print(z, z.dtype)
```

```
[ 1.    1.1   1.2 ... 599.7 599.8 599.9] float64
```

```
Process finished with exit code 0
```

Примечание: Если элементов слишком много то numpy выводит только первые 3 и последние 3



numpy.linspace

метод `numpy.linspace(start, end, size)` позволяет генерировать **size** элементов включая **start** и **end** значения

```
z = np.linspace(1, 6, 6000)
print(z, z.dtype)
```

```
[1.          1.00083347 1.00166694 ... 5.99833306 5.99916653 6.          ] float64
```

```
Process finished with exit code 0
```



N – мерные массивы

Конструкция N – мерного массива аналогична вложенным спискам, однако все элементы массива должны иметь схожую длину.

```
z = np.array([[1, 3, 5], [1, 2, 1]])  
print(z, z.dtype)
```

```
[[1 3 5]  
 [1 2 1]] int32
```

Process finished with exit code 0

Примечание: В противном случае элементы такого массива будут приведены к типу object, и дальнейшее использование методов numpy станет не корректно

```
z = np.array([[1, 3, 5], [1, 2]])  
print(z, z.dtype)
```

```
[list([1, 3, 5]) list([1, 2])] object
```

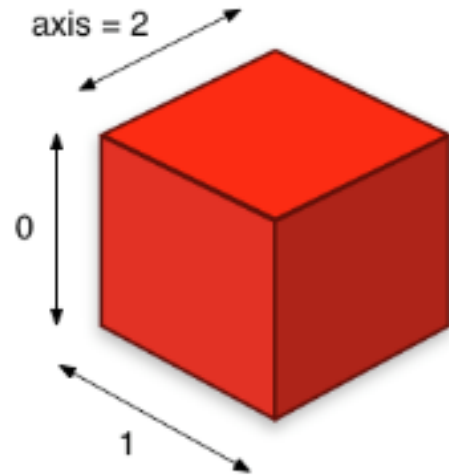
Process finished with exit code 0



N – мерные массивы. Оси и форма

Оси (axes) характеризуют размер каждого измерения в массиве. Кроме того они описывают порядок индексирования в массиве.

axis = 0 характеризует первый индекс, axis = 1 – второй и т.д.



Форма (shape) – поле, в виде кортежа, описывающее размер в каждом измерении у массива, в зависимости от количества элементов по каждой оси



N – мерные массивы. Свойства

- ❑ все элементы должны иметь один и тот же тип (**dtype**)
- ❑ по умолчанию **dtype** – float
- ❑ массив, конструируемый из списков смешенных типов, приводится к максимальному из ЭТИХ ТИПОВ:

```
z = np.array([[1, 3., 5e1], [1., 2, 1 + 2j]])  
print(z, z.dtype)
```

```
[[ 1.+0.j  3.+0.j 50.+0.j]  
 [ 1.+0.j  2.+0.j  1.+2.j]] complex128
```

```
Process finished with exit code 0
```



numpy.ones, numpy.zeros, numpy.eye

numpy.ones, **numpy.zeros** позволяют создать массив произвольной формы, заполненный единицами и нулями соответственно. В качестве аргумента методы принимают кортеж формы:

```
z = np.zeros((2, 3))  
print(z, z.dtype)
```

```
y = np.ones((3, 2))  
print(y, y.dtype)
```

```
[[0. 0. 0.]  
 [0. 0. 0.]] float64  
[[1. 1.]  
 [1. 1.]  
 [1. 1.]] float64
```

Process finished with exit code 0

numpy.eye позволяет создать единичную матрицу размера **n**

```
z = np.eye(4)  
print(z, z.dtype)
```

```
[[1. 0. 0. 0.]  
 [0. 1. 0. 0.]  
 [0. 0. 1. 0.]  
 [0. 0. 0. 1.]] float64
```

Process finished with exit code 0