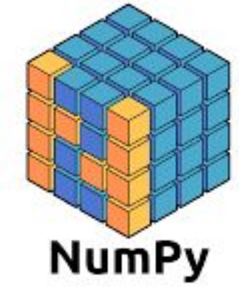
Numpy



Numpy

It is a module with a very convenient data type - array

Array characteristics:

1. Fixed size - size predetermined in the creation time
2. Homogeneity - contains values with one type
3. Memory efficiency
4. Speed efficiency
5. Vectorization - same behaviour as R vectors

Creation

We will encounter several shared arguments for many numpy functions

1. `shape` - dimensionality of desired array, e.g. 1 x 10, 5 x 5, usually passed as tuple
2. `dtype` - type of data in an array, typically includes specific allocated number of bits per value, e.g. `int`, `float`, `np.int8`, and so on

Creation from python sequences

```python
import numpy as np


# From list
xs = np.array([1, 2, 3])
xs
```

```
array([1, 2, 3])
```

`dtype` parameter

`dtype` can be passed with several ways:

1. python type - `int`, `float`
2. string name - '`int`', '`float`'
3. numpy types - `np.int`, `np.int8`, `np.int16`, `np.float64`

```python
# Now floats
fractions = np.array([1, 2, 3, 4, 5], dtype='float')
fractions
```

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

```python
# Another variant
np.array([1, 2, 3, 4, 5], dtype=float)
```

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

```
# Same
np.array([1, 2, 3, 4, 5], dtype=np.float)

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


# With specified number of bits per value - 16
floats16 = np.array([1, 2, 3, 4, 5],
dtype=np.float16)
floats16

array([1., 2., 3., 4., 5.], dtype=float16)
```

```python
import sys
```

```python
# In bytes
sys.getsizeof(fractions)
```

```
136
```

```python
sys.getsizeof(floats16)
```

```
106
```

Other types of initialization

```
# Shape is equal to 3 here
# With specific value - 5
np.full(3, 5)

array([5, 5, 5])



# Another shape - 2 x 2
np.full((2, 2), 5)

array([[5, 5],
       [5, 5]])
```

```
# Another type
np.full((2, 2), 5, dtype='str')

array([['5', '5'],
       ['5', '5']], dtype='<U1')
```

Some edge cases

```
# All 0
np.zeros(3)

array([0., 0., 0.])
```

```
# All 0
np.zeros((3, 3))

array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

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

array([[1., 1., 1.],
       [1., 1., 1.]])


# Identity matrix
# Only for 2-dimensional arrays
np.eye(3)

array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

```
np.eye(3, 4)

array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.]])
```

```
# Ranges
np.arange(5)

array([0, 1, 2, 3, 4])


# From 3 to 12
np.arange(3, 12)

array([ 3,  4,  5,  6,  7,  8,  9, 10, 11])


# With step equal to 2
np.arange(3, 12, 2)

array([ 3,  5,  7,  9, 11])
```

Vectorization

```
a = np.arange(3)
b = np.ones(3)
a, b
```

```
(array([0, 1, 2]), array([1., 1., 1.]))
```

```
a + 3
```

```
array([3, 4, 5])
```

```
a * 3

array([0, 3, 6])


a + b

array([1., 2., 3.])


a < b

array([ True, False, False])
```

```
a * b

array([0., 1., 2.])


# Vector multiplication
a.dot(b)

3.0
```

```
a = np.array(((1, 2),
             (3, 4)))
b = np.arange(2)
```

```
a + b
```

```
array([[1, 3],
       [3, 5]])
```

```
a * b
```

```
array([[0, 2],
       [0, 4]])
```

```
a.dot(b)

array([2, 4])


b.dot(a)

array([3, 4])


a @ b

array([2, 4])
```

Useful attributes

1. `shape` - lengths of dimensions, tuple, first rows
2. `size` - number of elements in the array, same as product of `shape`

```
a.shape
```

```
(2, 2)
```

```
a.size
```

```
4
```

```
# Don't use it
len(a)
```

```
2
```

Indexing

Vast topic. A couple of methods to start with:

1. `a[start:stop:step]` - same as with list, just extended to multiple dimensions
2. `a[[1, 2, 3]]` - "fancy" indexing, will get elements with indices 1, 2 and 3

```
# Don't pay attention to reshape now, will cover it
 later
matrix = np.arange(3, 15).reshape(3, -1)
matrix
```

```
array([[ 3,  4,  5,  6],
       [ 7,  8,  9, 10],
       [11, 12, 13, 14]])
```

```
matrix[0]
```

```
array([3, 4, 5, 6])
```

```
matrix[-1]

array([11, 12, 13, 14])
```

```
matrix[::2]

array([[ 3,  4,  5,  6],
       [11, 12, 13, 14]])
```

```
matrix[0, 0]

3
```

```
# 1st column
matrix[:, 0]

array([ 3,  7, 11])


matrix[:2, 1:3]

array([[4, 5],
       [8, 9]])


matrix[::2, ::2]

array([[ 3,  5],
       [11, 13]])
```