Programování 2: Knihovna NumPy

Martin Mareš
mj@ucw.cz

Katedra Aplikované Matematiky
MFF UK Praha

2021

Jak si pořídit matici

```
>>> import numpy as np
>>> a = np.array([[1, 2, 3], [4, 5, 6]])
```
>>> a (np.array je libovolně-rozměrné homogenní pole)
```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
>>> a.ndim
```
(kolikarozměrné je naše pole)
```
2
```

```
>>> a.shape
```
(tvar pole)
```
(2, 3)
```

```
>>> a.size
```
(celkový počet prvků)
```
6
```

```
>>> a.dtype
```
(typ dat společný pro všechny prvky)
```
dtype('int64')
```
(můžeme zvolit při vytváření pole)

```
>>> a.itemsize
```
(velikost jednoho prvku v paměti)
```
8
```

Nuly a jedničky

```
>>> np.zeros((2, 2))
array([[0., 0.],
       [0., 0.]])

>>> np.zeros((2, 2), dtype=np.int8)
array([[0, 0],
       [0, 0]], dtype=int8)

>>> np.ones((2, 2))
array([[1., 1.],
       [1., 1.]])

>>> a = np.array([[1, 2, 3], [4, 5, 6]])
>>> np.zeros_like(a)
array([[0, 0, 0],
       [0, 0, 0]])

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

(nebo také `np.eye`)

Generátory polí

```
>>> np.arange(10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

>>> np.arange(1, 2, 0.1)
array([1. , 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8,
1.9])

>>> np.linspace(1, 2, 10)
array([1. , 1.11111111, 1.22222222, 1.33333333,
1.44444444, 1.55555556, 1.66666667, 1.77777778,
1.88888889, 2. ])

>>> np.arange(6).reshape((2, 3))
array([[0, 1, 2],
       [3, 4, 5]])

>>> np.random.random((2, 2))
array([[0.3530976 , 0.32314115],
       [0.04913276, 0.86769289]])
```

(mezi 0 a 1)

Aritmetika s poli

```
>>> a = np.arange(4).reshape((2,2))
>>> a
array([[0, 1],
       [2, 3]])
```

```
>>> a + a
```
(sčítání po složkách)

```
array([[0, 2],
       [4, 6]])
```

```
>>> a * a
```
(násobení po složkách)

```
array([[0, 1],
       [4, 9]])
```

```
>>> a @ a
```
(maticové násobení)

```
array([[ 2, 3],
       [ 6, 11]])
```

Rozšiřování argumentů (broadcasting)

```
>>> a = np.ones((2, 3))
>>> a*3
array([[3., 3., 3.], [3., 3., 3.]])

>>> a * np.array([1, 2, 3])
array([[1., 2., 3.],
       [1., 2., 3.]])

>>> a * np.array([[6], [7]])
array([[6., 6., 6.],
       [7., 7., 7.]])
```

Pravidla rozšiřování – pro každý index **od konce**:

- Pokud jsou rozměry stejné, nedělej nic.
- Pokud je jeden roven 1, rozšiř ho podle druhého.
- Pokud jsou jinak různé, vyhlaš chybu.
- Pokud už jednomu poli došly indexy, domysli si rozměr 1.

Další aritmetika

```
>>> a = (2**np.arange(12)%11).reshape((3,4))
>>> a
array([[ 1,  2,  4,  8],
       [ 5, 10,  9,  7],
       [ 3,  6,  1,  2]])

>>> a.max()
10

>>> a.max(axis=0)    (maximum po řádcích)
array([ 5, 10, 9, 8])

>>> a.max(axis=1)    (maximum po sloupcích)
array([ 8, 10, 6])

>>> a.sum(axis=0)    (součet po řádcích)
array([ 9, 18, 14, 17])
```

U vícerozměrných polí může být **axis** i tuple více indexů.

Indexování a řezy

```
>>> a = np.arange(12).reshape((2,2,3))
array([[[ 0,  1,  2],
        [ 3,  4,  5]],
       [[ 6,  7,  8],
        [ 9, 10, 11]]])

>>> a[0,1,2]
5

>>> a[0,1]
array([3, 4, 5])

>>> a[0,1,1:3]
array([4, 5])

>>> a[:,:,0]    (zkratka: a[..., 0])
array([[0, 3],
       [6, 9]])
```

Pozor, řezy odkazují do původního pole! Kopíruje se pomocí `copy()`.

Transpozice a spol.

```
>>> a = np.arange(6).reshape((2, 3))
>>> a
array([[0, 1, 2],
       [3, 4, 5]])
```

```
>>> a.T    (transpozice, obecněji: a.swapaxes(0, 1))
array([[0, 3],
       [1, 4],
       [2, 5]])
```

```
>>> a[:,np.newaxis,:]
array([[[0, 1, 2]],
       [[3, 4, 5]]])
```

```
>>> np.arange(1,4)[:,np.newaxis] * np.arange(1,5)
array([[ 1,  2,  3,  4],
       [ 2,  4,  6,  8],
       [ 3,  6,  9, 12]])
```

Vybírání prvků

```
>>> b = 2**np.arange(10) % 13
>>> b
array([ 1, 2, 4, 8, 3, 6, 12, 11, 9, 5])
```

```
>>> b[np.array([2, 3, 4])]
```
(indexování číselným vektorem)

```
array([4, 8, 3])
```

```
>>> b < 7
array([ True,  True,  True, False,  True,  True,
       False, False, False, True])
```

```
>>> b[b < 7]
```
(indexování booleovským vektorem)

```
array([1, 2, 4, 3, 6, 5])
```

```
>>> b[b < 7] = 0
>>> b
array([ 0, 0, 0, 8, 0, 0, 12, 11, 9, 0])
```

Pozor, `x == y` také vrací booleovský vektor. Pro porovnání celých vektorů se hodí `np.equal()` nebo `np.allclose()`.

Slepování polí

```
>>> a = np.ones((2, 2))
>>> b = np.zeros((2, 2))
>>> np.vstack((a, b))
array([[1., 1.],
       [1., 1.],
       [0., 0.],
       [0., 0.]])

>>> np.hstack((a, b))
array([[1., 1., 0., 0.],
       [1., 1., 0., 0.]])
```

Obecněji: `np.concatenate`, `np.stack`