numpy

January 28, 2024

0.1 Logične operacije

Nad matrikami dtipa `bool` lahko izvajamo logične operacije. Uporabiti moramo Pythonove operatorje za delo nad biti, `&`, `|`, `^` in `~`.

[11]:
```
import numpy as np

a = np.array([[True, True, False],
              [False, False, True]])
b = np.array([[False, True, True],
              [True, False, True]])
```

[12]:
```
a & b
```

[12]:
```
array([[False,  True, False],
       [False, False,  True]])
```

[13]:
```
a | b
```

[13]:
```
array([[ True,  True,  True],
       [ True, False,  True]])
```

[4]:
```
a ^ b
```

[4]:
```
array([[ True, False,  True],
       [ True, False, False]])
```

[5]:
```
~a
```

[5]:
```
array([[False, False,  True],
       [ True,  True, False]])
```

Isti operatorji delajo tudi s tabelami `int`-ov, kjer res delajo z biti, kot običajno v Pythonu.

[6]:
```
e = np.array([3, 7, 8])
f = np.array([5, 1, 1])
```

[7]:
```
e | f
```

[7]:
```
array([7, 7, 9])
```

[8]:
```
e & f
```

[8]:
```
array([1, 1, 0])
```

[10]:
```
e ^ f
```

[10]:
```
array([6, 6, 9])
```

Poiščimo vse elemente `e`, ki so manjši od istoležnih elementov `f` ali pa so večji od `7`.

[14]:
```
(e < f) | (e > 7)
```

[14]:
```
array([ True, False,  True])
```

Da, in kateri so? Tile:

[16]:
```
e[(e < f) | (e > 7)]
```

[16]:
```
array([3, 8])
```

Pazite na oklepaje: `|` ima višjo prioriteto kot `<`.

0.2 `any` in `all`

Funkciji `np.any` in `np.all` sta podobni Pythonovima `any` in `all`, le da znata iti še po oseh.

[18]:
```
b
```

[18]:
```
array([[False,  True,  True],
       [ True, False,  True]])
```

[19]:
```
np.all(b, axis=0)
```

[19]:
```
array([False, False,  True])
```

[21]:
```
np.any(b, axis=0)
```

[21]:
```
array([ True,  True,  True])
```

0.3 Naloga

Tole vam bo prišlo prav pri 11 Dumbo Octopus.

Glavni trik je tule. Razmislite, kaj počnemo tule:

[23]:
```
a = np.array([3, 5, 2, 1, 6])
b = np.array([False, False, True, True, False])
```

```
a[:-1] += b[1:]
a
```

[23]:
```
array([3, 6, 3, 1, 6])
```

S tem trikom boste lahko prištevali `1` k sosedom. Če torej sprogramirate pravilno, ne boste nikoli naredili zanke prek elementov tabele. Imeli boste zanko, ki se bo ponavljala, dokler se zasveti kaka nova hobotnica, in potem, najbrž, zanko ali zanki, ki bosta šli prek osmih ali devetih sosednjih polj. Najbrž si boste hoteli pripraviti še dve tabeli - eno z vsemi hobotnicami, ki so se zasvetile v tem koraku in eno z vsemi, ki so se zasvetile pravkar, znotraj "podkoraka".

Dovolj spoilerjev. :)