numpy

January 28, 2024

0.1 Tridimenzionalne tabele - in osi

O `numpy`-ju se nam za to nalogo ni potrebno naučiti ničesar novega. Samo spomnimo se: tabele imajo lahko poljubno število dimenzij. Lahko tudi tri.

[1]:
```
import numpy as np

a = np.array([[[5, 1, 8, 6],
               [4, 5, 3, 12]],

              [[5, 3, 1, 1],
               [7, 5, 5, 4]],

              [[1, 3, 6, 3],
               [9, 1, 8, 2]]
             ])
```

[2]:
```
a.shape
```

[2]:
```
(3, 2, 4)
```

Tudi takšne tabele lahko indeksiramo z vsemi triki, ki smo jih spoznali, vključno s seznami in z maskami, ki morajo biti tudi v tem primeru enakih dimenzij kot tabele.

[3]:
```
a
```

[3]:
```
array([[[ 5,  1,  8,  6],
        [ 4,  5,  3, 12]],

       [[ 5,  3,  1,  1],
        [ 7,  5,  5,  4]],

       [[ 1,  3,  6,  3],
        [ 9,  1,  8,  2]]])
```

[4]:
```
a[a == 5] -= 1

a
```

[4]:
```
array([[[ 4,  1,  8,  6],
        [ 4,  4,  3, 12]],

       [[ 4,  3,  1,  1],
        [ 7,  4,  4,  4]],

       [[ 1,  3,  6,  3],
        [ 9,  1,  8,  2]]])
```

Takšna matrika ima tri osi. Da se bomo lažje pogovarjali, recimo, da ta tabela vsebuje tri matrike; vsaka ima dve vrstici in štiri stolpce.

Ničta os so matrike (`shape` je (3, 2, 4); prva os gre torej čez tri stvari). Če seštevamo po njej, seštevamo istoležne elemente.

[5]:
```
np.sum(a, axis=0)
```

[5]:
```
array([[ 9,  7, 15, 10],
       [20,  9, 15, 18]])
```

Druga os so vrstice (torej: vrstici) znotraj teh matrik. Če seštevamo po njej, torej po vrsticah, seštejemo stolpce vsake matrike. Vsaka vrstica te vsote ustreza eni od matrik, stolpci pa seveda ustrezajo stolpcem. Vrstice (tisto, česar je 2), so izginile, ker smo jih izsešteli.

[6]:
```
np.sum(a, axis=1)
```

[6]:
```
array([[ 8,  5, 11, 18],
       [11,  7,  5,  5],
       [10,  4, 14,  5]])
```

Tretja os so stolpci. Če seštevamo po njih, seštejemo vsako vrstico vsake matrike.

[7]:
```
np.sum(a, axis=2)
```

[7]:
```
array([[19, 23],
       [ 9, 19],
       [13, 20]])
```

Enako delujejo tudi vse druge funkcije, ki jim kot argument podamo os, na primer `np.min`, `np.max`, `np.mean`, `np.all`, `np.any`… Predvsem slednji dve boste potrebovali v tej nalogi.

0.2 Naloga

Naloga, ki se je lotite, je naloga 4, Giant Squid.

- Vse listke preberite v eno samo tridimenzionalno matriko.
- Zanka bo tekla prek izžrebanih številk, očitno. Ko je številka izžrebana, to številko na vseh listkih spremenite v nič. Z enim samim prirejanjem z maskami. To bo res preprosto - nobenih komplikacij, nobenih osi.

- Prava umetnost te naloge pa je ugotoviti, katere listki so zmagali. Če boste spretni, se boste izognili vsem zankam in z nekaj klici `np.any` in, morda, `np.all` ter kakšnim `|`, `&` in `~` dobili enodimenzionalno tabelo `bool`-ov, ki bodo povedali, kateri listki so zmagali.

Nalogo jemljite kot vajo iz razmišljanja o oseh.