numpy

January 28, 2024

0.1 Preoblikovanje tabel

Tabele imajo metodo `reshape`: podamo ji terko z novo obliko tabele, pa vrnejo matriko v tej, novi obliki. Nova oblika mora imeti enako število elementov kot stara. (Obe tabeli si v resnici delita isti pomnilnik, le različen pogled imata nanj).

[1]:
```
import numpy as np

a = np.arange(12)

a
```

[1]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

[2]:
```
a.reshape(4, 3)
```

[2]:
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

[3]:
```
a.reshape(2, 3, 2)
```

[3]:
```
array([[[ 0,  1],
        [ 2,  3],
        [ 4,  5]],

       [[ 6,  7],
        [ 8,  9],
        [10, 11]]])
```

[4]:
```
a.reshape((6, 2))
```

[4]:
```
array([[ 0,  1],
       [ 2,  3],
       [ 4,  5],
       [ 6,  7],
       [ 8,  9],
```

```
        [10, 11]])
```

Seveda lahko preoblikujemo tudi tabele, ki niso enodimenzionalne.

[5]:
```
a = np.array([[5, 4, 1, 8],
              [3, 0, 7, 0],
              [9, 6, 3, 4]])
```

[6]:
```
a.reshape((4, 3))
```

[6]:
```
array([[5, 4, 1],
       [8, 3, 0],
       [7, 0, 9],
       [6, 3, 4]])
```

Eno dimenzijo lahko `numpy` izračuna sam. Katero, povemo tako, da na onem mestu podamo `-1`.

[7]:
```
b = a.reshape((2, -1, 2))
b
```

[7]:
```
array([[[5, 4],
        [1, 8],
        [3, 0]],

       [[7, 0],
        [9, 6],
        [3, 4]]])
```

[8]:
```
b.shape
```

[8]:
```
(2, 3, 2)
```

To je posebej uporabno, kadar za matriko ne vemo vnaprej, kako velika bo. Če vemo, da bo imela štiri stolpce in bi radi združili ničtega in drugega v en stolpec, prvega in tretjega pa v drugi stolpec, napišemo

[9]:
```
a.reshape((-1, 2))
```

[9]:
```
array([[5, 4],
       [1, 8],
       [3, 0],
       [7, 0],
       [9, 6],
       [3, 4]])
```

Primerjajte to z `a`, ki ima v ničtem in drugem stolpcu (slučajno) soda, v prvem in tretjem pa liha števila.

[10]:
```
a
```

[10]:
```
array([[5, 4, 1, 8],
       [3, 0, 7, 0],
       [9, 6, 3, 4]])
```

Mimogrede in malenkost povezano s tem: število elementov matrike izvemo s

[11]:
```
a.size
```

[11]:
```
12
```

Seveda je `a.size` enak

[12]:
```
np.prod(a.shape)
```

[12]:
```
12
```

0.2 Naloga

Naloga bode naloga 5, Hydrothermal Venture.

Prijetno lahka je. Da bo res tako, vam podarjam funkcijo `anyrange`.

[13]:
```
def anyrange(a, b):
    return np.arange(a, b + 1) if b >= a else np.arange(a, b - 1, -1)
```

Podobna je funkciji `range`, vendar vključuje obe meji in ji je vseeno, katera je večja.

[14]:
```
anyrange(5, 10)
```

[14]:
```
array([ 5,  6,  7,  8,  9, 10])
```

[15]:
```
anyrange(10, 5)
```

[15]:
```
array([10,  9,  8,  7,  6,  5])
```

Tule je še branje datoteke.

[16]:
```
import re

lines = np.array([[int(x) for x in re.findall(r"\d+", v)]
                  for v in open("example.txt")])
```

Kdor pozna regularne izraze, to itak zna; kdor jih ne, naj mu bo tole prihranjeno.

Zdaj pa uživajte v preprosti nalogi. Edina zanka, ki jo še napišete, naj gre čez `lines`. Vse ostalo bodo učinkovito uporabljeni indeksi (ne pozabite na funkcijo `anyrange`!), seštevanje in vsote.

Kako dobimo elemente večje od 1? Ne pozabite, da je `True` isto kot `1` in `False` isto kot `0`.

Da ne bo koga zmedlo: `reshape` v resnici skoraj ne potrebujemo. Samo jagoda na vrhu kupe je.