resitev

January 28, 2024

0.1 Rešitev

Branje bomo tokrat napisali v funkciji, saj ga bomo poklicali dvakrat.

[1]:
```
import numpy as np

def read():
    numbers, *boards = open("example.txt").read().split("\n\n")
    numbers = map(int, numbers.split(","))
    boards = np.array([[[int(x) for x in line.split()]
                        for line in board.splitlines()]
                        for board in boards])
    return numbers, boards
```

Tule ni nič zanimivega v zvezi z `numpy`-jem, morda pa bo za koga kaj zanimivega v zvezi s Pythonom.

Datoteka vsebuje najprej spisek števil, nato dve prazni vrstici in nato listke, ki so prav tako ločeni z dvojnimi praznimi vrsticami.

`open("example.txt").read().split("\n\n")`

bo vrnil seznam, katerega prvi element so številke, ostali elementi listki. Razpakiramo ga v `numbers, *boards`, tako da bo `boards` terka z listki.

Druga zanimiva stvar je `map(int, numbers.split(","))`. To preslika vse `number.split(",")` čez funkcijo `int`. Rezultat ni seznam `int`-ov temveč generator. Razlika je v tem, da moremo čez generator le enkrat. Tu nam je za to vseeno, saj bomo dejansko šli čez števila le enkrat, z zanko `for`.

Zdaj pa se lotimo prvega dela.

[2]:
```
numbers, boards = read()
```

Zato, da bomo lažje videli, kaj se dogaja, izžrebajmo vsa števila do 24 - tam namreč dobimo prvi zmagovalni listek.

[3]:
```
for number in numbers:
    boards[boards == number] = 0
    if number == 24:
        break
```

[4]:
```
boards
```

[4]:
```
array([[[22, 13,  0,  0,  0],
        [ 8,  0,  0,  0,  0],
        [ 0,  0,  0, 16,  0],
        [ 6, 10,  3, 18,  0],
        [ 1, 12, 20, 15, 19]],

       [[ 3, 15,  0,  0, 22],
        [ 0, 18, 13,  0,  0],
        [19,  8,  0, 25,  0],
        [20,  0, 10,  0,  0],
        [ 0,  0, 16, 12,  6]],

       [[ 0,  0,  0,  0,  0],
        [10, 16, 15,  0, 19],
        [18,  8,  0, 26, 20],
        [22,  0, 13,  6,  0],
        [ 0,  0, 12,  3,  0]]])
```

Vidimo, da je zmagal zadnji listek, saj ima prečrtano celo prvo vrstico. Kako to izvemo? Narediti moramo `np.any` po vrsticah.

[5]:
```
np.any(boards, axis=2)
```

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

V tabeli, ki smo jo dobili zdaj, vsakemu listku ustreza ena vrstica, stolpci te matrika pa so vrstice originalnih listkov. Če ima listek kakšen `False`, je zmagal. To najpreprosteje preverimo z `np.all`:

[6]:
```
np.all(np.any(boards, axis=2), axis=1)
```

[6]:
```
array([ True,  True, False])
```

Zmagal je tretji listek, ker ima `False`. Ker bi radi listke, ki so zmagali in ne listkov, ki niso zmagali, to še negiramo.

[7]:
```
~np.all(np.any(boards, axis=2), axis=1)
```

[7]:
```
array([False, False,  True])
```

(De Morgan nas uči, da namesto lahko namesto `~np.all(nekaj)` pišemo tudi `np.any(~nekaj)`. Kar vam bolj sede.)

Listek zmaga tudi, če ima kakšen stolpec, v katerem so prečrtani vsi elementi. Pogoljufajmo in izžžrebajmo še 19.

[8]:
```
boards[boards == 19] = 0

boards
```

[8]:
```
array([[[22, 13,  0,  0,  0],
        [ 8,  0,  0,  0,  0],
        [ 0,  0,  0, 16,  0],
        [ 6, 10,  3, 18,  0],
        [ 1, 12, 20, 15,  0]],

       [[ 3, 15,  0,  0, 22],
        [ 0, 18, 13,  0,  0],
        [ 0,  8,  0, 25,  0],
        [20,  0, 10,  0,  0],
        [ 0,  0, 16, 12,  6]],

       [[ 0,  0,  0,  0,  0],
        [10, 16, 15,  0,  0],
        [18,  8,  0, 26, 20],
        [22,  0, 13,  6,  0],
        [ 0,  0, 12,  3,  0]]])
```

Zdaj mora iti notranji `np.any` po osi 1, zunanji pa še vedno po osi 1, saj os 0 še vedno predstavlja listke.

[9]:
```
~np.all(np.any(boards, axis=1), axis=1)
```

[9]:
```
array([ True, False, False])
```

Vsi zmagovalni listki so disjunkcija obeh; zmagaš lahko po z vrstico ali s stolpcem.

[10]:
```
~np.all(np.any(boards, axis=1), axis=1) | ~np.all(np.any(boards, axis=2),␣
↪axis=1)
```

[10]:
```
array([ True, False,  True])
```

Zdaj vemo vse, kar je treba in znamo napisati funkcijo, ki reši prvi del naloge.

[11]:
```
def winning():
    numbers, boards = read()
    for number in numbers:
        boards[boards == number] = 0
        won = ~np.all(np.any(boards, axis=2), axis=1) \
              | ~np.all(np.any(boards, axis=1), axis=1)
        if np.any(won):
            return number * np.sum(boards[won])
```

Funkcija v zanki prečrta število, poišče vse zmagovalne listke (`bool`-ova tabela `won`) in če je dejansko kdo zmagal vrnemo, kot zahteva naloga, to je, zadnje izžrebano število, pomnoženo z vsoto števil

na vseh zmagovalnih listkih - te dobimo z `boards[won]`.

[12]:
```
winning()
```

[12]:
```
4512
```

Funkcija, ki reši drugi del naloge, je čisto podobna.

[13]:
```
def losing():
    numbers, boards = read()
    for number in numbers:
        boards[boards == number] = 0
        won = ~np.all(np.any(boards, axis=2), axis=1) \
              | ~np.all(np.any(boards, axis=1), axis=1)
        if np.all(won):
            return number * np.sum(boards)
        boards = boards[~won]
```

Ker naloga zahteva, da vrnemo izžrebano število, pomnoženo z vsoto števil na vseh listkih, ki so bili prečrtani v zadnjem koraku, na koncu zanke vedno izfiltriramo listke, ki so "zmagali". Tako imamo takrat, ko ugotovimo, da bi zdaj končali igro vsi listki, pri roki tabelo s temi, ki so bili v zadnjem koraku še v igri.