

Rešitev

Najprej preberimo podatke; ločimo jih na koordinate in navodila za prepogibanje.

```
import numpy as np
```

```
xy, instructions = open("example.txt").read().split("\n\n")
```

Spisek koordinat razdelimo na vrstice (`xy.splitlines()`). Za vsako vrstico (`for v in xy.splitlines()`) sestavimo seznam, ki vsebuje števila iz vrstice, ki jo razbijemo z vejico kot ločilom (`[int(x) for x in v.split(",")]`). Ta seznam seznamov spremenimo v tabelo

```
xy = np.array([[int(x) for x in v.split(",")]
               for v in xy.splitlines()]])
```

```
xy
```

```
array([[ 6, 10],
       [ 0, 14],
       [ 9, 10],
       [ 0,  3],
       [10,  4],
       [ 4, 11],
       [ 6,  0],
       [ 6, 12],
       [ 4,  1],
       [ 0, 13],
       [10, 12],
       [ 3,  4],
       [ 3,  0],
       [ 8,  4],
       [ 1, 10],
       [ 2, 14],
       [ 8, 10],
       [ 9,  0]])
```

V prvem stolpcu so koordinate `x`, v drugem `y`. Poiskati moramo največjo in prišteti 1 (ker začnemo z 0); bo bodo dimenzije naše tabele.

```
np.max(xy, axis=0) + 1
```

```
array([11, 15])
```

Imeli bomo 15 vrstic in 11 stolpcev, torej moramo to še obrniti. Tako dobimo obliko, ki jo podamo `np.zeros`. Recimo, da bomo uporabljali tabelo `bool`-ov.

```
dots = np.zeros(np.max(xy, axis=0)[::-1] + 1, dtype=bool)
```

Zdaj pa na mesta, ki ju določajo koordinate `y` in `x` (dobimo jih z `xy[:, 1]` in `xy[:, 0]`) zapišemo enice.

Na hitro izpišimo, da vidimo, kaj imamo: gremo čez vse vrstice (`for v in dots`), za vsako gremo čez vse vrednosti `for i in v`, in to vrednost, spremenjeno v `int`, uporabimo kot indeks v niz `"#"`; kadar je `i` enak 0, bomo izpisali `.`, kadar je 1, pa `#`. Znakce združimo z `"".join`, vrstice pa z `"\n".join`.

. . . # . # . # .
 . . . #

 #
 . . . # . . . # . #

 . # . . . # . # # .
 . . . #
 # . # .
 #
 # . #

```
xy, instructions = open("example.txt").read().split("\n\n")
```

```
xy = np.array([[int(x) for x in v.split(",")] for v in xy.splitlines())]
dots = np.zeros(np.max(xy, axis=0)[::-1] + 1, dtype=bool)
dots[xy[:, 1], xy[:, 0]] = 1
```

```
xy, instructions = open("example.txt").read().split("\n\n")
```

```
x, y = zip(*(map(int, v.split(", ")) for v in xy.splitlines()))
dots = np.zeros((max(y) + 1, max(x) + 1), dtype=bool)
dots[y, x] = 1
```

Zdaj gremo čez navodila. Če vrstica navodil vsebuje `y`, prepogibamo tako, da seštevamo spodnje in zgornje vrstice, sicer seštevamo levi in desni del. Če sta `h` in `w` trenutni dimenziji (`h, w = dots.shape`), je `dots[:h // 2]` zgornja polovica vrstic, `dots[:h // 2:-1]` pa prezrcaljena spodnja polovica. Podobno

```

for instruction in instructions.splitlines():
    h, w = dots.shape
    if "y" in instruction:
        dots = dots[:h // 2] + dots[h // 2:-1]
    else:
        dots = dots[:, :w // 2] + dots[:, :w // 2:-1]

print("\n".join("".join("#"[int(i)] for i in v) for v in dots))

#####
#...#
#...#
#...#
#####
.....
.....

```

```
xy, instructions = open("input.txt").read().split("\n\n")

xy = np.array([[int(x) for x in v.split(",")] for v in xy.splitlines()]])
dots = np.zeros(np.max(xy, axis=0)[::-1] + 1, dtype=bool)
dots[xy[:, 1], xy[:, 0]] = 1

for i, instruction in enumerate(instructions.splitlines()):
    h, w = dots.shape
    if "x" in instruction:
        dots = dots[:, :w // 2] + dots[:, :w // 2:-1]
    else:
        dots = dots[:h // 2] + dots[:h // 2:-1]
    if not i:
        print(np.sum(dots))

print("\n".join("".join("#"[int(i)] for i in v) for v in dots))
```

712

3

#.#.#...#.#.#...#.#.#...#.#.#...
###.####.#.#.#...##.#...##.#...