resitev

January 28, 2024

- Domača naloga je nadaljevanje prejšnje.
- Ker smo se ta teden učili o izpeljanih seznamih, množicah in slovarjih ter generatorjih, je pri reševanju potrebno uporabljati le-te. Da boste to res počeli, morajo biti vse funkcije dolge eno samo vrstico - en sam `return`. Testi bodo to preverjali. Da si ne bi pomagali z dodatnimi funkcijami, ki bodo v resnici opravile vse delo, je tokrat **prepovedano pisanje funkcij**, ki jih naloga ne zahteva.
- Naloga je sestavljena tako, da je mogoče vse funkcije napisati v eni vrstici - z izpeljenim seznamom, slovarjem ali množico - brez večjega truda, vendar le, če programirate pametno in zgledno. Zato: pomagajte si z objavljenimi rešitvami prejšnje naloge. (S tem vas tudi prisilim, da jih preberete in razumete. :)) Vaša tokratna naloga je v bistvu predelati prejšnje rešitve iz "imperativne" oblike v "deklarativno".
- Naloga se začne z nekaj ogrevalnimi funkcijami, ki niso povezane z glavnim delom naloge in vam pri njem ne bodo prišle prav.
- Miklavž je letos nekam zgoden. Preden začnete reševati, poskrolajte do konca, kjer vas čaka nekaj daril.

0.1 Obvezno ogrevanje

Napiši naslednje funkcije (vse prejmejo *neprazen* seznam krogov, opisanih s trojko (x, y, r); funkcija veliki poleg tega prejme še nek polmer);

- `polmeri(krogi)` vrne množico, ki vsebuje vse različne polmere krogov iz seznama,
- `veliki(r0, krogi)` vrne množico koordinat krogov, katerih polmer je enak ali večji od `r0`,
- `obsegi(krogi)` vrne vsoto obsegov krogov,
- `najlevo(krogi)` vrne koordinato x najbolj levo točko, ki se je dotikajo krogi (namig: od koordinate središča je potrebno odšteti polmer!),
- `povrsina(krogi)` vrne površino pravokotnika, ki se razteza med najbolj levo in najbolj desno ter najbolj zgornjo in najbolj spodnjo točko, ki jo pokrivajo krogi. Z drugimi besedami, če bi morali kroge pokriti s pravokotnikom, ki bi ga postavili vzporedno z osmi (torej: brez vrtenja) - kako velik bi bil?

0.1.1 Rešitev

Naloga zahteva le preprosto ponavljanje stvari, kakršne smo počeli na predavanjih.

[1]:
```
def polmeri(krogi):
    return {r for *_, r in krogi}

def veliki(r0, krogi):
    return {(x, y) for x, y, r in krogi if r >= r0}
```

```
def obsegi(krogi):
    return 2 * pi * sum(r for *_, r in krogi)

def najlevo(krogi):
    return min(x - r for x, y, r in krogi)

def povrsina(krogi):
    return (max(x + r for x, y, r in krogi)
            - min(x - r for x, y, r in krogi)) \
           * (max(y + r for x, y, r in krogi)
              - min(y - r for x, y, r in krogi))
```

Pri obsegi lahko ločeno računamo obseg vsakega kroga, lahko pa izpostavimo `2 * pi` in seštevamo le polmere. Vseeno je.

Namen funkcije `najlevo` je predvsem mentalna priprava na funkcijo `povrsina`, kjer je potrebno na enak način pridobiti še najbolj desno, najbolj spodnjo in najbolj zgornjo točko.

Nekateri so poskušali stvari v slogu

```
def povrsina(krogi):
    return ((max(x + r) - min(x - r)) * (max(y + r) - min(y - r))
            for x, y, r in krogi)
```

To ne deluje. Kot prvo, to ni število temveč generator, saj je, če odmislimo tiste max-e in min-e, oblike `return (... for x, y, r in krogi)`. Drugo, tu v, recimo, prvem klicu funkcije `max` le-tej damo en sam argument `x + r`; celo če bi bil naš namen sestaviti generator, to ne bi delovalo, ker `max` pričakuje bodisi neko zaporedje (na primer seznam, množico, generator…) ali pa več reči, med katerimi vrne največjo.

0.2 Obvezna naloga

Napiši naslednje funkcije:

- `ocisti_slovar(d)` prejme nek slovar `d` in vrne nov slovar, ki vsebuje samo tiste pare (ključ, vrednost), pri katerih je vrednost resnična. "Neresnične" vrednosti so `False`, `0` in `0.0` ter prazen seznam, množica, niz ali terka.
- `znotraj(x0, y0, r0, krogi)` vrne seznam tistih krogov iz seznama `krogi`, ki se nahajajo znotraj kroga s središčem `(x0, y0)` in polmerom `r0`. (Krog mora biti strogo znotraj - ni dovolj, da sovpada. Krog `(0, 0, 5)` ne leži znotraj kroga `(0, 0, 5)`.)
- `vsebovanost(krogi)` vrne slovar, katerega ključi so krogi, vrednosti pa so seznami krogov, ki so znotraj teh krogov, ki jih opisuje ključ. Če kak krog ne vsebuje nobenega drugega kroga, naj se tudi med ključi ne pojavi; z drugimi besedami, med vrednostmi naj ne bo nobenega praznega seznama. (Pomemben namig: uporabljaj funkcije, ki jih že imaš!)
- `notranjost(krogi)` vrne množico vseh krogov, ki so znotraj katerega izmed drugih krogov.
- `ptici0(vsebovani, notranji)` prejme slovar, kakršnega vrača funkcija `vsebovanost` in množico, kakršno vrača `notranjost`. Vrniti mora množico vseh ptičev.
- `ptici(krogi)` prejme kroge in vrne ptiče.

Rešitev Funkcija `ocisti_slovar` (glej spodaj) je bila že podarjena, napisal jo je Miklavž.

[2]:
```
def ocisti_slovar(d):
    return {k: v for k, v in d.items() if v}
```

Vseeno se spodobi, da jo pogledamo in razumemo, saj je trivialna. `{k: v for k, v in d.items()}` bi naredilo slovar, ki vsebuje enake pare (ključ, vrednost) kot orignalni slovar `d`; z drugimi besedami, sestavilo bi kopijo. Če dodamo še `if v`, pa odstrani vse pare, pri katerih vrednost ni resnična.

Funkcijo `znotraj` moramo napisati sami.

[3]:
```
def znotraj(x0, y0, r0, krogi):
    return [(x1, y1, r1)
            for x1, y1, r1 in krogi
            if r0 > r1 and (x1 - x0) ** 2 + (y1 - y0) ** 2 < r0 ** 2]
```

nek krog `(x1, y1, r1)` je znotraj `(x0, y0, r0)`, če velja pogoj, ki smo ga zapisali v `if` - in je enak kot je bil pogoj znotraj dvojnih zank v funkciji `vsebovanost` iz prejšnje naloge.

`vsebovanost` je najpomembnejša - ali vsaj najbolj poučna funkcija v celi domači nalogi. Rešitev je takšna

[4]:
```
def vsebovanost(krogi):
    return ocisti_slovar({(x, y, r): znotraj(x, y, r, krogi)
                          for x, y, r in krogi}
    )
```

Ali, če znamo malo več

[5]:
```
def vsebovanost(krogi):
    return ocisti_slovar({krog1: znotraj(*krog1, krogi)
                          for krog1 in krogi})
```

Kako, točno, dela drugo, ni pomembno. Kdor ve, ve.

Pomembno je razmišljanje ob sestavljanju slovarja. Sestaviti moramo torej slovar. Dobili ga bomo tako, da bomo šli prek vseh krogov (`for ... in krogi`). Ključi bodo podatki o krogu, `(x, y, r)`, pripadajoče vrednosti pa vsi krogi, ki so znotraj tega kroga. Le-te bomo dobili s klicem funkcije `znotraj`. (Če se tega ne bi domislili, pa bi pač zamenjali klic funkcije `znotraj` z njeno kodo.)

Na koncu pokličemo `ocisti_slovar`, da odstrani kroge, ki ne vsebujejo nobenih krogov.

Kar smo napisali, je "prevod" tega:

[6]:
```
def vsebovanost(krogi):
    d = {}
    for x, y, r in krogi:
        d[(x, y, r)] = znotraj(x, y, r, krogi)
    return ocisti_slovar(d)
```

Nekateri (mnogi? večina?) so (najprej?) poskušali preobrniti tole:

[7]:
```
def vsebovanost(krogi):
    d = {}
    for x, y, r in krogi:
        for x1, y1, r1 in krogi:
            if r < r1 and (x1 - x) ** 2 + (y1 - y) ** 2 < r1 ** 2:
                if (x1, y1, r1) not in d:
                    d[(x1, y1, r1)] = set()
                d[(x1, y1, r1)].add((x, y, r))
    return d
```

To ne gre. Problem je `add`, ki dodaja malo v to, malo v ono množico. Ko sestavljamo izpeljan seznam, množico ali slovar, si lahko predstavljamo, da se le-ta sestavi "paralelno", vsi elementi naenkrat. Ali, še boljše in bolj prav: v tak seznam, množico ali slovar, "*definiramo*", povemo, kaj vsebuje. Ne moremo najprej nečesa sestaviti, na primer prazne ali še ne čisto dokončane množice in potem dodajati vanjo.

Funkcija `notranjost` je bila podana.

[8]:
```
def notranjost(krogi):
    return set().union(*map(set, vsebovanost(krogi).values()))
```

Če koga zanima: naredi prazno množico, potem pa pokliče njeno metodo `union`, ki ji kot argument (ne prezrite `*`, če razumete, kaj pomeni `*` v klicu) poda vse vrednosti iz slovarja `values`, ki jih mimogrede (z `map`) preslika v množice.

Če tega ne razumete, nič hudega. Presega to, kar se učimo pri predmetu, zato vam je funkcija podarjena.

Zdaj pa še `ptici0`. To je množica vseh krogov, ki vsebujejo kakšen krog in ustrezajo določenim pogojem.

[9]:
```
def ptici0(vsebovani, notranji):
    return {krog[:2]
            for krog, znotraj in vsebovani.items()
            if krog not in notranji and len(znotraj) == 2
            and znotraj[0] not in vsebovani and znotraj[1] not in vsebovani}
```

Ta funkcija ni nič drugega kot prevod rešitve prejšnje naloge,

[10]:
```
def ptici(krogi):
    vsebuje, notranji = vsebovanost(krogi)
    pticji = set()
    for krog, vkrogu in vsebuje.items():
        if (krog not in notranji
                and len(vkrogu) == 2 \
                and vkrogu[0] not in vsebuje and vkrogu[1] not in vsebuje):
            pticji.add(krog[:2])
    return pticji
```

Pri čemer izpustimo prvo vrstico, ker funkcija že dobi prav tidve stvari kot argument.

0.3 Dodatna naloga

Napiši še

- `letala0(vsebovani, notranji)`
- `letala(krogi)`
- `sumljivi0(krogi, vsebovani, notranji)` (pazi: ta dobi tudi `krogi`!)
- `sumljivi(krogi)`

ki so ekvivalentne funkcijam `ptici0` in `ptici`, le da vračajo letala in sumljive leteče predmete.

0.3.1 Rešitev

Tole je bilo za dodatno nalogo, ker zahteva, da se spomnimo še na funkciji `all` in `any`.

Neka reč je letalo, če gre za zunanji krog, vsebuje vsaj kakšen krog, vendar ne dveh in **za vse** (`all`) vsebovane kroge velja, da ne vsebujejo nobenega kroga, torej da ne nastopajo kot ključi v slovarju `vsebovani`.

[11]:
```
def letala0(vsebovani, notranji):
    return {krog[:2]
            for krog, znotraj in vsebovani.items()
            if krog not in notranji and len(znotraj) != 2
            and all(krog2 not in vsebovani for krog2 in znotraj)
            }
```

Neka reč je sumljiva, če je zunanji krog, ki nima notranjih (ne nastopa kot ključ v `vsebovani` ali pa **katerikoli** (`any`) od vsebovanih krogov vsebuje kakšen ključ (torej: nastopa kot ključ v `vsebovani`).

[12]:
```
def sumljivi0(krogi, vsebovani, notranji):
    return {krog[:2]
            for krog in krogi
            if krog not in notranji and (
                    krog not in vsebovani
                    or any(krog2 in vsebovani for krog2 in vsebovani[krog]))}
```

0.4 Miklavževa darila

Sveti Miklavž se je naveličal sajastih dimnikov in razmišlja o zamenjavi službe, zato se letos uči programirati, ker bo to izgledalo dobro v CV-ju. Ker ne more iz svoje dobrohotne kože, pa vam podarja nekaj funkcij. Lahko jih uporabite; glede na to, da gre za Miklavževa darila, tega ne bomo šteli za prepisovanje.

Objavljam jih skupaj s komentarji otrok, mislim, študentov.

```
# Hvala, Miklavž, čeprav bi tole znali napisati tudi sami.
# Obljubimo, da se jo bomo potrudili razumeti
def ocisti_slovar(d):
    return {k: v for k, v in d.items() if v}
```

```python
# U, tale je pa kar zaresna. V ponedeljek bomo prosili profesorja,
# da nam jo razloži, ker smo radovedni kako deluje.
def notranjost(krogi):
    return set().union(*map(set, vsebovanost(krogi).values()))

# Eh, to bi pa res znali sami. Bomo še ostale naredili podobno.
def ptici(krogi):
    return ptici0(vsebovanost(krogi), notranjost(krogi))
```