

Izpit 31. januar 2024

1. Knjigovodstvo

Napišite funkcijo `knjigovodstvo(ime_datoteke)`, ki prejme ime datoteke s prihodki (vrstice s pozitivnimi števili) in odhodki (negativna števila) v obliki, kot jo kaže okvirček na desni. Funkcija mora vrniti trojko: seznam s pari imen dohodkov in pripadajočih zneskov, seznam s pari imen odhodkov in pripadajočih cen (kot pozitivna števila) ter skupno vsoto. Za datoteko

```
slika: 50
slika: 100
tempera: -3
stol: -20
kip: 20
zrak: 0
torba: 12

vrne

([('slika', 50), ('slika', 100), ('kip', 20), ('torba', 12)],
 [('tempera', 3), ('stol', 20)],
 159)
```

pri čemer je 159 preprosto vsota vseh števil.

Rešitev

Najprej pripravimo, kar bomo vrnili - seznama `prihodki` in `odhodki` ter `bilanca`.

Nato gremo po datoteki: vsako vrstico razdelimo glede na `:` in pretvorimo znesek v število. Če je število negativno, dodamo par s stvarjo in zneskom v `odhodki` (pri čemer mora biti število `-znesek`), sicer v `prihodki`. Pa `bilanco` povečamo za `znesek`.

```
def knjigovodstvo(ime_datoteke):
    prihodki = []
    odhodki = []
    bilanca = 0
    for vrstica in open(ime_datoteke):
        stvar, znesek = vrstica.split(":")
        znesek = int(znesek)
        if znesek < 0:
            odhodki.append((stvar, -znesek))
        elif znesek > 0:
            prihodki.append((stvar, znesek))
        bilanca += znesek
    return prihodki, odhodki, bilanca
```

Nekateri radi pišejo

```
if znesek > 0:
    bilanca += znesek
else:
    bilanca -= abs(znesek)
```

To seveda ni potrebno. Python zna prištevati negativne vrednosti. :)

Prav tako je lepo, da bilanco spreminjamo izven `if` in ne znotraj, ločeno za odhodke in za prihodke.

Eden od študentov je poskusil nalogo rešiti z `numpy`-jem. Sicer se mu ni čisto posrečilo, vendar gre in je kar lepo. Dobra ideja.

```
def knjigovodstvo(ime_datoteke):
    podatki = np.genfromtxt(ime_datoteke, delimiter=":", dtype=str)
    stvari = podatki[:, 0]
    cene = podatki[:, 1].astype(int)
    prihodki = cene > 0
    odhodki = cene < 0
    return (list(zip(stvari[prihodki], cene[prihodki])),
            list(zip(stvari[odhodki], -cene[odhodki])),
            np.sum(cene))
```

Zoprna stvar je tisto na koncu: z `zip` poberemo skupaj stvari in cene, da nam sestavi pare, potem pa pokličemo še `list`, da jih zložimo v seznam.

2. Draginja

Napiši funkcijo `draginja(odhodki)`, ki vrne seznam parov imen stvari in cen. Ista stvar se lahko pojavi večkrat, a morda z različnimi cenami. Funkcija mora vrniti ime stvari, katere povprečna cena je bila največja. Če je takšnih stvari več, lahko vrne poljubno izmed njih.

Klic `draginja([('stol', 20), ('torba', 12), ('tempera', 3), ('miza', 50), ('stol', 30), ('stol', 60), ('miza', 40), ('torba', 5)])` vrne `'miza'`, saj je povprečna cena miz enaka $(50 + 40) / 2 = 45$, medtem ko je povprečna cena, recimo, stolov $(20 + 30 + 60) / 3 = 36,66$, torbe in tempere pa so še cenejše.

Rešitev

Tole je naloga iz slovarjev. Da bomo lahko računali povprečne cene, bomo potrebovali dva: v enem beležimo vsote cen vsake stvari, v drugem število takšnih stvari.

Ko se odločimo glede tega, je funkcija preprosta.

```
def draginja(odhodki):
    skupna_cena = defaultdict(float)
    skupno_kosov = defaultdict(int)
    for stvar, cena in odhodki:
        skupna_cena[stvar] += cena
        skupno_kosov[stvar] += 1

    naj_stvar, naj_povp = None, -1
    for stvar in skupna_cena:
        povp = skupna_cena[stvar] / skupno_kosov[stvar]
        if povp > naj_povp:
            naj_stvar, naj_povp = stvar, povp
    return naj_stvar
```

Zoprni drugi del se da skrajšati, če znamo bolj spretno uporabljati `max` in poznamo `lambde`.

```
def draginja(odhodki):
    skupna_cena = defaultdict(float)
    skupno_kosov = defaultdict(int)
    for stvar, cena in odhodki:
        skupna_cena[stvar] += cena
        skupno_kosov[stvar] += 1
    return max(skupna_cena, key=lambda stvar: skupna_cena[stvar] / skupno_kosov[stvar])
```

3. Dragocenosti

Napišite funkcijo `dragocenosti(stvari, cene, meja)`, ki prejme `numpy`-jevo tabelo z imeni stvari in dvodimenzionalno tabelo, katere vrstice ustrezajo stvarjem (v enakem vrstnem redu kot so te našteje v prvi tabeli), stolpci pa dnevom. Vrednosti v tabeli povedo, koliko je stvar stala na posamezni dan.

Funkcija mora vrniti tabelo z imeni vseh stvari, katerih cene so bile vsaj en dan višje od podane meje `meja`. Vrstni red mora biti enak vrstnemu redu v tabeli stvari.

Pri tej nalogi pričakujemo rešitev v čistem `numpy`-ju, brez zank v `Pythonu`.

Rešitev

Takole.

```
def dragocenosti(stvari, cene_po_dnevih, meja):
    return stvari[np.any(cene_po_dnevih > meja, axis=1)]
```

Ali pa takole.

```
def dragocenosti(stvari, cene_po_dnevih, meja):
    return stvari[np.max(cene_po_dnevih, axis=1) > meja]
```

4. Spletno knjigovodstvo

Ko poženete teste, se bo v direktoriju s programom pojavila datoteka knjigovodstvo.html. Lahko jo odprete v brskalniku ali urejevalniku.

Napišite funkcijo `spletno_knjigovodstvo(ime_datoteke)`, ki zna prebrati podatke iz takšne datoteke. Predpostaviti smete, da opis nakupa vedno vsebuje le eno število ter da je to število pozitivno in celo. Funkcija mora vrniti seznam s pari stvari in cen. Za podano datoteko naj vrne `[('slika', 50), ('slika', 100), ('tempera', 3), ('polomljen stol', 20), ('kip', 20), ('zrak', 0), ('torba', 12)]`.

Rešitev

Seveda bomo uporabili BeautifulSoup in regularne izraze.

Podatki so v delu HTML-ja, ki je videti tako:

```
<dt>slika</dt>
<dd>Stala je 50, lahko bi tudi malo manj.</dd>
<dt>slika</dt>
<dd>Ta je bila malo dražje, zanjo smo dali 100.</dd>
```

Poiščemo vse `dt` in potem `dd`, ki jim sledijo. Znotraj tega, kar je bilo v `dd` poiščemo zaporedja števk `\d+`. Kar najdemo, pretvorimo v število.

```
def spletno_knjigovodstvo(ime_datoteke):
    soup = BeautifulSoup(open(ime_datoteke), "html.parser")
    postavke = []
    for vrstica in soup.find_all("dt"):
        stvar = vrstica.text
        opis = vrstica.find_next_sibling("dd").text
        znesek = re.search(r"\d+", opis).group()
        postavke.append((stvar, int(znesek)))
    return postavke
```

5. Evidenca

Napišite funkcijo `evidenca(postavke, ime_datoteke)`, ki prejme postavke v obliki trojk z imenom stvarmi, ceno za kos in številom kosov, na primer `[("slika", 50, 2), ("tempera", 3, 1), ("stol", 20, 1), ("kip", 20, 12), ("zrak", 0, 141), ("torba", 12, 1)]`. Funkcija mora v datoteko `ime_datoteke` zapisati tabelo v obliki, ki jo kaže slika na desni. Točno obliko - število presledkov - razberite iz testov.

Stvar	Cena x Kosov	Skupaj
-----	-----	-----
slika	50 x 2	100
tempera	3 x 1	3
stol	20 x 1	20
kip	20 x 12	240
zrak	0 x 141	0
torba	12 x 1	12

Rešitev

Tule gre zgolj za osnovno oblikovanje nizo: prešteti moramo, na koliko znakov poravnamo kakšno stvar, pa z > in < potiskati stvari na desno in na levo.

```
def evidenca(postavke, ime_datoteke):
    f = open(ime_datoteke, "w")
    f.write(f"{'Stvar':10} {'Cena':>6} x {'Kosov':<5} {'Skupaj':>10}\n")
    f.write("-" * 36 + "\n")
    for stvar, cena, kosov in postavke:
        f.write(f"{'stvar':10} {'cena':>6} x {'kosov':<5} {'cena * kosov':10}\n")
```