resitev

January 28, 2024

0.1 Janez reče

Na prvem izpitnem roku 2020/21 je bila naloga Simon reče. Zaploskajte, primite se za nosove in tako naprej. To je v času korone popolnoma neaktualno (razen, če je avtor naloge hotel nakazati, da se da to igrati tudi po Zoomu). Tu se bomo šli igro "Janez reče".

Vse funkcije, ki jih boste pisali, bodo dobile seznam nizov tega, kar je v nekih zaporednih tednih, dneh ali urah rekel Janez. Na primer

[1]:
```
odredbe = [
    "Prepovedano: gostilne, fakultete",
    "Prepovedano: prehajanje med regijami, nočni sprehodi, javni promet",
    "Dovoljeno: fakultete",
    "Prepovedano: tek na prostem",
    "Dovoljeno: fakultete",
    "Dovoljeno: nočni sprehodi, tek na prostem",
    "Prepovedano: fakultete, nočni sprehodi, frizer",
    "Dovoljeno: frizer",
    "Prepovedano: fakultete, zobozdravnik, tek na prostem",
    "Dovoljeno: tek na prostem, javni promet"]
```

Vsak niz se torej začne z besedo `"Prepovedano"` ali `"Dovoljeno"`, ki ji sledi dvopičje, nato pa seznam sveže prepovedanih ali sveže dovoljenih dejavnosti. Posamezne dejavnosti so ločene z vejicami (in same ne vsebujejo vejic - med dejavnostmi ne boste našli `"Pouk v šolah, srednjih šolah in fakultetah"`, ker bi vmesna vejica povzročala nerešljive sitnosti).

Žal moramo računati s tem, da lahko Janez v kakem trenutku prepove tudi, kar je v tem trenutku itak že prepovedano, ali dovoli kaj, kar je že dovoljeno. Vem, da to zaplete nalogo, a takšna je pač realnost.

Napiši funkcijo `je_dovoljeno(odredbe, dejavnost)`, ki prejme takšen seznam odredb in ime neke dejavnosti, na primer `"javni promet"`. Funkcija mora vrniti `True`, če je dejavnost v tem trenutku dovoljena in `False`, če ni.

Napiši funkcijo `najnevarnejse(odredbe)`, ki vrne ime dejavnosti, ki je bila največkrat prepovedana. Kot prepoved štejemo tudi prepoved, izrečeno takrat, ko je dejavnost sicer že itak prepovedana. Če je enakokrat prepovedanih več dejavnosti, naj vrne poljubno izmed njih. V gornjem primeru so to fakultete, ki se na seznamu prepovedanih reči znajdejo trikrat.

Napiši funkcijo `tezki_casi(odredbe)`, ki pove, koliko je bilo največ hkrati prepovedanih aktivnosti. V gornjem primeru vrne `7`, saj je bilo v nekem trenutku (točneje: predzadnjem koraku) hkrati

prepovedanih sedem stvari.

Mogoče se ti splača predtem napisati pomožno funkcijo, ki prejme posamično vrstico in vrne podatke, ki jih le-ta vsebuje, v primernejši obliki. Oblikuj in imenuj jo po lastni presoji, ali pa je ne piši. Kokr čš.

Disclaimer: avtor naloge nima nobenega posebnega odnosa ne do Janeza ne do Alenke, temveč podpira samo Ano, Berto, Cilko, Dani, Emo in Fanči, pa vse do Olge iz tretjega letnika.

0.2 Rešitev

Najprej napišimo funkcijo `preberi(odredba)`, ki prejme odredbo (posamična vrstica iz seznama odredb, kot je gornji) in vrne "smer", ki bo `True`, če gre za dovoljenje in `False`, če za prepoved, ter seznam dejavnosti, na katere se nanaša odredba. Naloga takšne funkcije sicer ne zahteva, vendar nam bo prišla prav pri prav vseh funkcijah, ki jih bomo morali napisati.

[2]:
```
def preberi(odredba):
    smer, dejavnosti = odredba.split(": ")
    return smer == "Dovoljeno", dejavnosti.split(", ")
```

Vrstico najprej razdelimo glede na `": "`. Potem vrnemo dve reči: prva pove, ali je "smer" enaka `"Dovoljeno"`, druga pa bo seznam dejavnosti, ki jih dobimo tako, da vse, kar je bilo desno od dvopičja razdelimo glede na vejico.

Na hitro preskusimo.

[3]:
```
preberi("Prepovedano: prehajanje med regijami, nočni sprehodi, javni promet")
```

[3]:
```
(False, ['prehajanje med regijami', 'nočni sprehodi', 'javni promet'])
```

[4]:
```
preberi("Dovoljeno: nočni sprehodi, tek na prostem")
```

[4]:
```
(True, ['nočni sprehodi', 'tek na prostem'])
```

0.2.1 `je_dovoljeno`

Tale funkcija ni nič posebnega. Ne zahteva slovarjev, še sezname komaj.

Vsaka dejavnost je na začetku dovoljena. Nato gremo prek vseh odredb. Če se dejavnost pojavi med tistimi, na katere se nanaša odredba, si zapomnimo "smer" - prepoved ali dovoljenje. Na koncu vrnemo, kar je bilo pač zadnje.

[5]:
```
def je_dovoljeno(odredbe, dejavnost):
    dovoljeno = True
    for odredba in odredbe:
        smer, dejavnosti = preberi(odredba)
        if dejavnost in dejavnosti:
            dovoljeno = smer
    return dovoljeno
```

[6]:
```
je_dovoljeno(odredbe, "nočni sprehodi")
```

[6]:
```
False
```

[7]:
```
je_dovoljeno(odredbe, "tek na prostem")
```

[7]:
```
True
```

[8]:
```
je_dovoljeno(odredbe, "marihuana")
```

[8]:
```
True
```

0.2.2 najnevarnejse

Tu pa bo brez slovarjev težko. Šteti moramo, kolikokrat se pojavi kakšna dejavnost. Za to bomo uporabili slovar, katerega ključi so dejavnosti, pripadajoče vrednosti pa število omemb te dejavnosti. Potrebovali bomo dve zanki - eno čez odredbe in znotraj te čez dejavnosti. Za vsako dejavnost povečamo števec v slovarju.

Za začetek dobljeni slovar samo izpišimo.

[9]:
```
def najnevarnejse(odredbe):
    pojavitve = {}
    for odredba in odredbe:
        _, dejavnosti = preberi(odredba)
        for dejavnost in dejavnosti:
            if dejavnost in pojavitve:
                pojavitve[dejavnost] += 1
            else:
                pojavitve[dejavnost] = 1

    print(pojavitve)
```

[10]:
```
najnevarnejse(odredbe)
```

```
{'gostilne': 1, 'fakultete': 5, 'prehajanje med regijami': 1, 'nočni sprehodi':
3, 'javni promet': 2, 'tek na prostem': 4, 'frizer': 2, 'zobozdravnik': 1}
```

Takšno preštevanje je preprostejše, če uporabimo `defaultdict`. Kot argument mu podamo `int`, tako da bodo novoizmišljene vrednosti enake `0`.

[11]:
```
from collections import defaultdict

def najnevarnejse(odredbe):
    pojavitve = defaultdict(int)
    for odredba in odredbe:
        _, dejavnosti = preberi(odredba)
        for dejavnost in dejavnosti:
            pojavitve[dejavnost] += 1
```

```
    print(pojavitve)
```

[12]:
```
najnevarnejse(odredbe)
```

```
defaultdict(<class 'int'>, {'gostilne': 1, 'fakultete': 5, 'prehajanje med
regijami': 1, 'nočni sprehodi': 3, 'javni promet': 2, 'tek na prostem': 4,
'frizer': 2, 'zobozdravnik': 1})
```

Zdaj pa moramo v slovarju poiskati ključ, ki pripada največji vrednosti. Recimo tako:

[13]:
```
from collections import defaultdict

def najnevarnejse(odredbe):
    pojavitve = defaultdict(int)
    for odredba in odredbe:
        _, dejavnosti = preberi(odredba)
        for dejavnost in dejavnosti:
            pojavitve[dejavnost] += 1

    naj_dejavnost = None
    for dejavnost, pojavitev in pojavitve.items():
        if naj_dejavnost == None or pojavitev > pojavitve[naj_dejavnost]:
            naj_dejavnost = dejavnost
    return naj_dejavnost
```

[14]:
```
najnevarnejse(odredbe)
```

[14]:
```
'fakultete'
```

V zanki gremo prek vseh parov `(dejavnost, pojavitev)` in če je število pojavitev večje od število pojavitev tiste dejavnosti, pri kateri jih je bilo doslej največ (`naj_dejavnost`), si zapomnimo to dejavnost. V začetku pa je `naj_dejavnost` enaka `None`; v prvem krogu zanke jo zamenjamo z naslednjo.

Druga pogosta rešitev (sam bom včasih napisal eno ali drugo) je ta:

[15]:
```
from collections import defaultdict

def najnevarnejse(odredbe):
    pojavitve = defaultdict(int)
    for odredba in odredbe:
        _, dejavnosti = preberi(odredba)
        for dejavnost in dejavnosti:
            pojavitve[dejavnost] += 1

    naj_dejavnost = None
    naj_pojavitev = -1
    for dejavnost, pojavitev in pojavitve.items():
```

```
        if pojavitev > naj_pojavitev:
            naj_dejavnost = dejavnost
            naj_pojavitev = pojavitev
    return naj_dejavnost
```

[16]:
```
najnevarnejse(odredbe)
```

[16]:
```
'fakultete'
```

Oboje je v redu.

Študenti pa pogosto oddajo nekaj takšnega:

[17]:
```
from collections import defaultdict

def najnevarnejse(odredbe):
    pojavitve = defaultdict(int)
    for odredba in odredbe:
        _, dejavnosti = preberi(odredba)
        for dejavnost in dejavnosti:
            pojavitve[dejavnost] += 1

    return max(pojavitve, key=pojavitve.get)
```

[18]:
```
najnevarnejse(odredbe)
```

[18]:
```
'fakultete'
```

Študent, ki odda takšno rešitev, je dokazal, da zna uporabljati Google in je odkril Stack Overflow. Lepo. Vendar le, če študent potem tudi razume, kaj je napisal. Zakaj deluje tale `max` bomo razumeli, ko se bomo učili o vezanih metodah. Namig: pri Programiranju 1 se to ne bo zgodilo. Če prav vem, tudi nikoli kasneje ne.

Z Googlanjem za rešitve ni nič narobe, če potem razumete, kaj pišete.

Nalogo se da rešiti tudi tako.

[19]:
```
from collections import Counter

def najnevarnejse(odredbe):
    return Counter(d for odredba in odredbe for d in preberi(odredba)[1]).
 ↪most_common(1)[0][0]
```

[20]:
```
najnevarnejse(odredbe)
```

[20]:
```
'fakultete'
```

Poučno? Ne.

0.2.3 `tezki_casi`

To nalogo je mogoče rešiti na precej različnih načinov. V vseh je potrebno iti prek odredb, beležiti, kaj je v vsakem določenem trenutku prepovedano in si zapomniti največje število hkrati prepovedanih dejavnosti. Vprašanje je, v kakšni obliki shranjevati seznam prepovedanih reči.

Za začetek - lahko je kar seznam.

[21]:
```
def tezki_casi(odredbe):
    prepovedano = []
    naj_prepovedanih = 0
    for odredba in odredbe:
        smer, dejavnosti = preberi(odredba)
        for dejavnost in dejavnosti:
            if smer:
                if dejavnost in prepovedano:
                    prepovedano.remove(dejavnost)
            else:
                if dejavnost not in prepovedano:
                    prepovedano.append(dejavnost)

        if len(prepovedano) > naj_prepovedanih:
            naj_prepovedanih = len(prepovedano)
    return naj_prepovedanih
```

[22]:
```
tezki_casi(odredbe)
```

[22]: 7

V seznamu `prepovedno` bo vse, kar je prepovedano in v `naj_prepovedanih` bo največje število hkrati prepovedanih doslej.

Zanka gre, kot vedno, čez odredbe. Iz vsake preberemo smer in dejavnosti. Nato gremo prek vseh dejavnosti. Če gre za odredbo o dovoljenjih, jo odstranimo iz seznama prepovedanih (a le, če je v njem, torej, če je res prepovedana!). Če gre za prepovedi, jo dodano v seznam (a le, če še ni v njem).

Na koncu, torej po tem, ko smo dodali ali odvzeli vse prepovedane ali dovoljene dejavnosti, pogledamo, ali je seznam daljši od najdaljšega doslej videnega.

Namesto seznama prepovedanih dejavnosti bi lahko imeli slovar. Ključi bi bili dejavnosti, vrednosti pa preprosto `None`. Ker nas v resnici ne zanimajo.

[23]:
```
def tezki_casi(odredbe):
    prepovedano = {}
    naj_prepovedanih = 0
    for odredba in odredbe:
        smer, dejavnosti = preberi(odredba)
        for dejavnost in dejavnosti:
            if smer:
```

```
                if dejavnost in prepovedano:
                    del prepovedano[dejavnost]
            else:
                prepovedano[dejavnost] = None

        if len(prepovedano) > naj_prepovedanih:
            naj_prepovedanih = len(prepovedano)
    return naj_prepovedanih
```

[24]:
```
tezki_casi(odredbe)
```

[24]: 7

Odstranjevanje je podobno prejšnjemu, dodajanje ima en pogoj manj. Smo s tem res kaj pridobili?

Smo: če imamo seznam, morata operator `in` in metoda `remove` iskati dejavnost. Daljši ko je, počasnejši sta. Če vzamemo slovar, pa `in` in `del` vzameta čas, neodvisen od dolžine seznama. Pri tako kratkih seznamih prepovedi (stokajte, koliko hočete: za Python bi bilo celo sto prepovedi malenkst :), se to seveda ne pozna.

Če se komu zdi hecno imeti slovar, katerega vrednosti so vedno `None`: v resnici smo jih uporabljali do nekega davnega Pythona, v katerem so končno dodali nov podatkovni tip: množice. Te se bomo učili ravno danes.

[25]:
```
def tezki_casi(odredbe):
    prepovedano = set()
    naj_prepovedanih = 0
    for odredba in odredbe:
        smer, dejavnosti = preberi(odredba)
        for dejavnost in dejavnosti:
            if smer:
                prepovedano.discard(dejavnost)
            else:
                prepovedano.add(dejavnost)

        if len(prepovedano) > naj_prepovedanih:
            naj_prepovedanih = len(prepovedano)
    return naj_prepovedanih
```

[26]:
```
tezki_casi(odredbe)
```

[26]: 7

Tako se to naredi zares.