

Domača naloga: Vreme po Sloveniji

Tokratna naloga se je nadaljevala iz nalog z vaj. Zato so tule najprej rešitve teh, nato še rešitev domače naloge.

Naloge z vaj

V eni od nalog smo se ukvarjali z resničnimi podatki z vremenskih postaj v Sloveniji. Podatki so s strani <https://www.ncei.noaa.gov/data/daily-summaries/access/>. Ta vsebuje datoteke za vse(?) postaje na svetu; arhiv z vsemi datotekami je na <https://www.ncei.noaa.gov/data/daily-summaries/archive/> (in je velik 7 GB). Da ne bi pobrali vsega, so v priloženi datoteki .zip samo datoteke, ki se nanašajo na Slovenijo.

Kot boste videli, podatki niso v tako prijazni obliki, kot so bili tisti, s katerimi ste delali vi. Za to vajo boste delali s podatki v obliki, kakršno bi dobili na spletu. Kot boste videli, to ne bo zgolj vaja iz programiranja, temveč tudi iz brskanja po dokumentaciji in splošne iznajdljivosti. Narobe lahko gre marsikaj, sitnosti lahko povzroča karkoli. Vaje iz resničnega sveta, torej.

Datoteko s podatki lahko odzipaš v direktorij, v katerem bo program, lahko drugam (če jih znaš kljub temu brati).

1. Kodirnik postaj

Vsaka postaja ima svojo kodo. Kredarica, recimo, je SIE00105938. Napiši funkcijo `kodirnik_postaj`, ki ne prejme nobenih argumentov, vrne pa takšen slovar:

```
{'Bilje': 'SIE00115106',  
'Celje Medlog': 'SIE00115176',  
'Crnomelj Doblice': 'SIE00114856',  
'Kocevje': 'SIE00114956',  
'Kredarica': 'SIE00105938',  
'Lesce': 'SIE00114966',  
'Letalisce Edvarda Rusjana Mari': 'SIE00115156',  
'Letalisce Jozeta Pucnika Ljubl': 'SIE00115146',  
'Lisca': 'SIE00115186',  
'Ljubljana Bežigrad': 'SIM00014015',  
'Murska Sobota Rakican': 'SIE00115196',  
'Nova Vas Na Blokah': 'SIE00115066',  
'Novo Mesto': 'SIE00115126',  
'Portoroz Letalisce': 'SIE00115166',  
'Postojna': 'SIE00115076',  
'Ratece Planica': 'SIE00115206',  
'Smartno Pri Slovenj Gradcu': 'SIE00115136',  
'Topol Pri Medvodah': 'SIE00115006',  
'Veliki Dolenci': 'SIE00115096',
```

```
'Vojsko': 'SIE00115016']}
```

Bodi pozoren na velike in male črke. Slovar mora seveda razbrati iz datotek in/ali njihovih imen. (Teste bi prestala tudi, če bi ta slovar preprosto prekopirali v funkcijo. :)

Rešitev Neka zelo kratka različica bi bila

```
import os
import csv

def kodirnik_postaj():
    kodirnik = {}
    for fn in os.listdir("podatki"):
        for vrstica in csv.DictReader(open("podatki/" + fn)):
            kodirnik[vrstica["NAME"][:4].title()] = vrstica["STATION"]
    return kodirnik
```

Funkcija predpostavlja, da so datoteke v poddirektoriju "podatki" in da v tem direktoriju ni ničesar drugega kot te datoteke. Poleg tega prebere vse vrstice datoteke, čeprav je ime postaje znano že po prvi vrstici in bi lahko branje prekinili.

Primerna rešitev je

```
def kodirnik_postaj():
    kodirnik = {}
    for fn in os.listdir():
        if not fn.endswith(".csv"):
            continue
        for vrstica in csv.DictReader(open(fn)):
            kodirnik[vrstica["NAME"][:4].title()] = vrstica["STATION"]
            break
    return kodirnik
```

Zdaj imamo pogoj, ki preverja, ali se datoteka konča s ".csv". Če ni tako, s `continue` preskočimo ostanek zanke in nadaljujemo z naslednjo datoteko. Poleg tega pa prekinemo drugo zanko takoj po branju prve vrstice (`break`).

2. Popravi kodirnik

Imena krajev niso lepa. Nekatera se končajo sredi besede, druga so brez šumnikov, tretja vsebujejo preveč podroben opis krajev (namesto Celje Medlog smo zadovoljni tudi s Celje). Napiši funkcijo `popravi(kodirnik)`, ki prejme slovar, kakršnega vrača prva funkcija in ga spremeni - zamenja imena krajev (ključe) z lepšimi.

Zahtevane pretvorbe so zapisane v tem slovarju.

```
popravki = {'Murska Sobota Rakican': 'Murska Sobota',
```

```

'Crnomelj Doblice': 'Črnomelj',
'Letalisce Edvarda Rusjana Mari': 'Maribor',
'Letalisce Jozeta Pucnika Ljubl': 'Brnik',
'Ljubljana Bezigrad': 'Ljubljana',
'Kocevje': 'Kočevje',
'Smartno Pri Slovenj Gradcu': 'Smartno pri Slovenj Gradcu',
'Kredarica': 'Kredarica',
'Veliki Dolenci': 'Veliki Dolenci',
'Novo Mesto': 'Novo mesto',
'Nova Vas Na Blokah': 'Bloke',
'Celje Medlog': 'Celje',
'Portoroz Letalisce': 'Portorož',
'Topol Pri Medvodah': 'Topol pri Medvodah',
'Ratece Planica': 'Rateče'
}

```

Torej: 'Crnomelj Doblice' se mora spremeniti v 'Črnomelj'.

Ta slovar skopiraj v funkcijo in ga uporabi.

Pomoč: ključa v slovarju se ne da spremeniti. Pač pa odstraniš stari ključ (uporabiš lahko `del d[k]` ali, mogoče še boljše `d.pop(k)`, kjer je `d` slovar, `k` pa ključ, ki ga želiš odstraniti) in dodaš nov, popravljen ključ, z vrednostjo, kakršna je bila pripisana staremu ključu.

Rešitev Smiselna rešitev je

```

def popravi(kodirnik):
    for ime, menjava in popravki.items():
        kodirnik[menjava] = kodirnik.pop(ime)

```

Predpostavili smo, da se vsako ime, ki ga je potrebno popraviti (in je torej v `popravki`), dejansko pojavi v slovarju `kodirnik`. Če ni tako, bo `kodirnik.pop(ime)` javil napako.

Nekateri študenti so naredili obratno: z zanko so šli čez `kodirnik` in poskušali zamenjevati ključke, ki jih je potrebno zamenjati. To je, kot prvo, potratno: predstavljajte si, da bi imeli kodirnik z vsemi postajami na svetu, preimenovati pa bi jih bilo potrebno 100. Gornji program gre pač čez 100 postaj, ki potrebujejo preimenovanje; različica, ki gre čez vse postaje in preverja, katere preimenovati, pa gre čez, uh, vse postaje na svetu.

Še pomembneje: to ne deluje. Če greš z zanko čez slovar (ali množico), v tej zanki tega slovarja (ali množice) ne smeš spreminjati. Zanka bo to opazila in javila napako. Zakaj? Zato ker se v slovarju (predvsem pa v množici - slovar je od Pythona 3.6 bolj "stabilen") vrstni red elementov spreminja in zanka ne more vrniti vseh elementov, če se ji ti sproti mešajo.

3. Preberi meritve

Napiši funkcijo `preberi_meritve(ime_postaje, kodirnik)`, ki prejme ime postaje in kodirnik (slovar, kakršnega pridela prejšnja funkcija). Vrniti mora slovar, katerega ključi so datumi, vrednosti pa najvišje temperature na ta dan.

Ključi morajo biti terke v obliki (leto, mesec, dan). Po klicu `temperature = preberi_meritve("Kredarica", kodirnik)`, je `temperature[(2023, 8, 13)]` temperatura 13. avgusta 2023.

Temperatura je v stolpcu TMAX, podana je v desetinkah Celzijev. Deliti jo morate torej z 10. Na omenjeni datum je za temperaturo na Kredarici zapisano 144; to pomeni, da je bila temperatura 14.4 Celzijev. Shraniti morate seveda to, popravljeno temperaturo.

Na nekatere dneve temperatura ni bila izmerjena. Vrednosti za ta dan ne shranjuj v slovar. Tako, recimo, slovar za Kredarico ne bo imel ključa `(2023, 10, 4)`, ker ta dan termometer, izgleda, ni obratoval. :)

Rešitev Uporabili bomo `csv.DictReader`. Pri sestavljanju naloge sem mislil, da bo tako lažje kot uporabiti `csv.reader` in ročno šteti stolpce. Potem se je izkazalo, da vsaj ena postaja (Bilje) ne meri globine snežne ideje (zraven Nove Gorice to res ni preveč smiselno), zato ima datoteka tam manj stolpcev in pride TMAX na vrsto že prej. Brez `DictReader`-ja bi tu torej imeli hujše težave, kot je bilo videti na prvi pogled.

```
def preberi_meritve(kraj, kodirnik):
    meritve = {}
    for vrstica in csv.DictReader(open(kodirnik[kraj] + ".csv")):
        datum = datetime.strptime(vrstica["DATE"], "%Y-%m-%d")
        temp = vrstica["TMAX"].strip()
        if temp:
            temp = int(temp) / 10
            meritve[(datum.year, datum.month, datum.day)] = temp
    return meritve
```

Datum smo razkopali z `datetime.strptime`. To se mi zdi lažje kot `strip("-")` in potem pretvarjanje vsakega kosa posebej v `int`. Komur je bolj všeč po starem, pa naj dela po starem.ž

Domača naloga

Napiši funkcijo `mrzli_silvester(podatki)`, ki prejme slovar, kakršnega vrne funkcija `preberi_podatke`. Vrniti mora leto z najhladnejšim silvestrskim dnem. Torej: med vsemi leti, za katere je znana temperatura 31. decembra, mora vrniti tisto leto, ko je bila le-ta najnižja. Če je takih let več, naj vrne najzgodnejše.

Nasvet: najstarejši podatki so iz leta 1900, torej lahko iščete podatke za 31. 12. po letih od 1900 do 2022. Ne delajte zanke čez celoten slovar (to deluje, vendar

ni zgledno), temveč le čez leta.

Rešitev

Tole niti ni bila naloga iz tekoče snovi, temveč zgolj iz slovarjev, ki jih poznamo že več kot mesec. Morda je bila ideja bolj v tem, da zagotovo rešite naloge z vaj. :)

```
import math

def mrzli_silvester(podatki):
    naj_leto = naj_mraz = math.inf
    for leto in range(1900, 2023):
        datum = (leto, 12, 31)
        if podatki.get(datum, math.inf) < naj_mraz:
            naj_mraz = podatki[datum]
            naj_leto = leto
    return naj_leto
```

No, da bo vsaj nekaj novega, smo uporabili `math.inf` za začetno "najnižjo" temperaturo.

Rešitev vsega skupaj

Za celovitejši občutek obsežnosti celotnega projekta :), je to vsa koda na kupu.

```
import os
import csv
import math
from datetime import datetime

## Naloge z vaj

def kodirnik_postaj():
    kodirnik = {}
    for fn in os.listdir():
        if not fn.endswith(".csv"):
            continue
        for vrstica in csv.DictReader(open(fn)):
            kodirnik[vrstica["NAME"][:-4].title()] = vrstica["STATION"]
            break
    return kodirnik

def popravi(kodirnik):
    for ime, menjava in popravki.items():
        kodirnik[menjava] = kodirnik.pop(ime)
```

```

def preberi_meritve(kraj, kodirnik):
    meritve = {}
    for vrstica in csv.DictReader(open(kodirnik[kraj] + ".csv")):
        datum = datetime.strptime(vrstica["DATE"], "%Y-%m-%d")
        temp = vrstica["TMAX"].strip()
        if temp:
            temp = int(temp) / 10
            meritve[(datum.year, datum.month, datum.day)] = temp
    return meritve

```

Domača naloga

```

def mrzli_silvester(podatki):
    naj_leto = naj_mraz = math.inf
    for leto in range(1900, 2023):
        datum = (leto, 12, 31)
        if podatki.get(datum, math.inf) < naj_mraz:
            naj_mraz = podatki[datum]
            naj_leto = leto
    return naj_leto

```