

Banane predstavljajo eno od osnovnih dobrin sodobne družbe. Brez banan si praktično ne moremo več predstavljati moderne ekonomije, umetnosti in izobraževalnega procesa, pomembne pa so tudi v prehrani.

Zato se ne gre čuditi, da so banane tudi eden izmed najpomembnejših izvoznih, predvsem pa uvoznih artiklov. V priloženi datoteki zip so zato podatki Svetovne banke in UNCTADa o uvozu banan, spremenjeni v obliko .csv. Z datoteko ravnajte spoštljivo. Delite jo s prijatelji in znanci, ki bi jih utegnili zanimati ta pomembna tematika.

Odzipajte datoteko. V njej se nahaja datoteka testi.py, podatki pa so v poddirektoriju podatki. Podatke pustite tam, kjer so, program pa pišite v datoteko testi.py, ki je prav tako ne premikajte nikamor. Datotek ne preimenujte! Ko oddajate nalogo, oddajte samo testi.py.

V nekaterih nalogah je potrebno vrniti najcenejše, najdražje ali kaj podobnega, ali pa prvih n držav glede na nek kriterij. Ne ukvarjajte se s tem, da si morda več držav deli isto mesto, saj so številke dovolj raznolike, da se to najbrž ne zgodi.

Za določeno oceno je potrebno pravilno rešiti tudi vse naloge za nižje ocene.

Ocena 6

preberi_tezo in preberi_vrednost

1. Napišite funkcijo `preberi_tezo(leto)`, ki prebere podatke za podano leto in vrne slovar, katerega ključi so imena držav (Reporter), vrednosti pa so teže uvoženih banan, kot jih preberete iz stolpca Quantity.

Med državami je tudi "European Union". To "državo" preskočite. Prav tako preskočite vrstice, v katerih podatek manjka ("Quantity" je prazen).

Poleg tega napišite funkcijo `preberi_vrednost(leto)`, ki vrne podoben slovar, le da so vrednosti vrednost uvoženih banan v dolarjih (**številko iz tabele pomnožite s 1000). Tudi tu odstrani evropsko unijo.

Rešitev Naloga preverja, ali znamo sestaviti slovar in uporabljati `DictReader`. S `split(",")` si tu ne moremo pomagati: v to nas prisilijo vejice znotraj podatkov.

Omembe je vredno še sestavljanje imena datoteke. Navadno so ga naše funkcije dobile kot argument, tokrat pa kot argument dobijo letnico. Ime sestavimo z f-stringom `f"podatki/Banane {leto}.csv"`.

```
def preberi_tezo(leto):
    uvoz = {}
    for vrstica in csv.DictReader(open(f"podatki/Banane {leto}.csv")):
        if vrstica["Quantity"].strip() and vrstica["Reporter"] != "European Union":
```

```

        uvoz[vrstica["Reporter"]] = int(vrstica["Quantity"])
    return uvoz

def preberi_vrednost(leto):
    uvoz = {}
    for vrstica in csv.DictReader(open(f"podatki/Banane {leto}.csv")):
        if vrstica["Trade Value 1000USD"].strip() and vrstica["Reporter"] != "European Union":
            uvoz[vrstica["Reporter"]] = 1000 * float(vrstica["Trade Value 1000USD"])
    return uvoz

```

Kdor hoče, lahko piše tudi

```

def preberi_tezo(leto):
    return {vrstica["Reporter"]: int(vrstica["Quantity"])
            for vrstica in csv.DictReader(open(f"podatki/Banane {leto}.csv"))
            if vrstica["Quantity"].strip() and vrstica["Reporter"] != "European Union"}

def preberi_vrednost(leto):
    return {vrstica["Reporter"]: 1000 * float(vrstica["Trade Value 1000USD"])
            for vrstica in csv.DictReader(open(f"podatki/Banane {leto}.csv"))
            if vrstica["Trade Value 1000USD"].strip() and vrstica["Reporter"] != "European Union"}

```

vendar kakšne posebne prednosti ne vidim.

najcenejse_banane

1. Napišite funkcijo `najcenejse_banane(leto)`, ki poišče državo, ki je v podanem letu `leto` uvažala banane za najnižjo ceno (na kilogram) in vrne par (terko) z imenom te države in ceno, ki jo je plačala (v dolarjih na kilogram).

Rešitev Naloga od reševalca zahteva klasično reč, ki smo jo programirali že dvajsetkrat - iščemo minimum glede na določen kriterij. Enkrat smo napisali celo splošno funkcijo `argmax`, ki dela nekaj takšnega.

Najbolj klasična rešitev je

```

def najcenejse_banane(leto):
    teza = preberi_tezo(leto)
    vrednost = preberi_vrednost(leto)
    naj_drzava = naj_cena = None
    for drzava in teza:
        cena = vrednost[drzava] / teza[drzava]
        if naj_cena == None or cena < naj_cena:
            naj_cena = cena
            naj_drzava = drzava
    return naj_drzava, naj_cena

```

Gremo čez vse države - se pravi, čez vse ključe v enem slovarju, ob čemer hrabro predpostavljamo, da se bo ta ključ našel tudi v drugem. Za vsako državo izračunamo ceno ter si zapomnimo najmanjšo ceno in pripadajočo državo.

Nekoliko drugačna rešitev je tale:

```
def najcenejse_banane(leto):
    teza = preberi_tezo(leto)
    vrednost = preberi_vrednost(leto)
    cene = []
    for drzava in teza:
        cene.append((vrednost[drzava] / teza[drzava],
                    drzava))
    naj_cena, naj_drzava = min(cene)
    return naj_drzava, naj_cena
```

V teoriji je to slabše, saj zahteva več pomnilnika. Naših podatkov je malo, torej nam je vseeno.

Tu zložimo vse par (cena, država) v seznam terk `cene`. Po zanki z `min(cene)` poiščemo "najmanjšo" terko. Terke se najprej primerjajo po prvem elementu, tisti, pri katerih je ta enak, pa po drugem. Dobili bomo torej terko z najnižjo ceno (če bi bili dve ceni enaki, pa tisto, pri kateri je država prej po abecednem redu). Razpakiramo ju v `naj_cena` in `naj_drzava` ter ju vrnemo.

In to nas pripelje do krajše rešitve, ki je poleg tega še pomnilniško učinkovita in hitra.

```
def najcenejse_banane(leto):
    teza = preberi_tezo(leto)
    vrednost = preberi_vrednost(leto)
    naj_cena, naj_drzava = min((vrednost[drzava] / teza[drzava], drzava)
                               for drzava in teza)
    return naj_drzava, naj_cena
```

Po ideji je podobna prejšnji, le da `min` spustimo prek generatorja, tako da se seznam nikoli ni zapiše v pomnilnik, saj se terke generirajo (in pozabljajo) sproti.

Kdor bi hotel še vrstico manj, bi pisal

```
def najcenejse_banane(leto):
    teza = preberi_tezo(leto)
    vrednost = preberi_vrednost(leto)
    return min((vrednost[drzava] / teza[drzava], drzava)
               for drzava in teza)[-1]
```

Vendar je prejšnja funkcija lepša, saj bolj jasno poimenuje, kaj dobimo iz `min-a`. Vsaj jaz tako mislim.

najblizje_sloveniji

1. Napišite funkcijo `najblizje_sloveniji(leto)`, ki vrne ime države, ki je bila glede na skupno težo uvoženih banan v podanem letu najbližje Sloveniji. Če za Slovenijo v tistem letu ni podatkov, funkcija vrne `None`.

Ne spreglejte, da gre lahko za državo, ki je uvozila malo manj ali malo več banan kot Slovenija.

Rešitev Spet naloga na isto vižo. Ideja je, da morate sestaviti nekoliko bolj zapleten pogoj, poleg tega pa še manjši hec s tem, da Slovenije morda sploh ni v seznamu za tisto leto.

```
def najblizje_sloveniji(leto):
    podatki = preberi_tezo(leto)
    slovenija = podatki.get("Slovenia")
    if slovenija is None:
        return None
    naj_drzava = None
    for drzava, kolicina in podatki.items():
        if drzava != "Slovenia" and (
            naj_drzava is None
            or abs(kolicina - slovenija) < abs(podatki[naj_drzava] - slovenija)):
            naj_drzava = drzava
    return naj_drzava
```

Tu je primerno uporabiti metodo `get`, saj nismo prepričani, ali je Slovenija res v slovarju. Če je ni, takoj vrnemo `None`. Sicer je reč podobna kot prej, le da smo tokrat obrnili še malo drugače: namesto da bi si zapomnili najbližjo državo in najmanjšo razliko, si zapomnimo le najbližjo državo, razliko pa računamo kar vsakič sproti, z `abs(podatki[naj_drzava] - slovenija)`. Stvar sloga.

Glavna reč te funkcije je pogoj. Najprej preverimo, ali država ni Slovenija. Sledi `and` in - zelo pomembno - `or` v oklepaju. Brez tega oklepaja je `and` močnejši kot `or`, tako da bi dobili, v bistvu (`drzava != "Slovenia" and naj_drzava is None`) or

razpon_cen

1. Napišite funkcijo `razpon_cen(leto)`, ki vrne par (terko) z najnižjo in najvišjo ceno banan v podanem letu (v dolarjih na kilogram).

Rešitev Ker gre za naloge za oceno 6, ni tako grozno, če so malo dolgočasne. Prej smo iskali minimume, zdaj iščemo minimum in maksimum hkrati. Za popestritev pa bo tokrat začetna vrednost neskončna, ne `None`.

```
def razpon_cen(leto):
    teza = preberi_tezo(leto)
    vrednost = preberi_vrednost(leto)
```

```

najm_cena = math.inf
najv_cena = -math.inf
for drzava in teza:
    cena = vrednost[drzava] / teza[drzava]
    if cena < najm_cena:
        najm_cena = cena
    if cena > najv_cena:
        najv_cena = cena
return najm_cena, najv_cena

```

Ocena 7

1. Napišite funkcijo `naj_uvoznice(leto, n)`, ki vrne prvih `n` držav po teži uvoza banan v podanem letu. Seznam naj bo urejen padajoče po teži.

Rešitev

Če hočemo prvih `k`, bo potrebno urejati. (Obstajajo učinkovitejše rešitve brez popolnega urejanja, vendar jih pri tem predmetu ne razumemo.) Kaj pa bomo urejali? V bistvu teže. Vendar poleg tež potrebujemo države. Torej bomo sestavili seznam terk (teža, država) in ga uredili. Potem bomo iz zadnjih `k` terk pobrali samo države.

```

def naj_uvoznice(leto, k):
    teza = preberi_tezo(leto)

    uvoznice = []
    for drzava, teza in teza.items():
        uvoznice.append((teza, drzava))
    uvoznice.sort()

    prvih_k = []
    for _, drzava in uvoznice[::-1][:k]:
        prvih_k.append(drzava)
    return prvih_k

```

Reč je nekoliko zamotana, tule: `uvoznice[::-1][:k]`. Seznam je urejen po naraščajočih težah. Z `[::-1]` ga obrnemo, s `[:k]` pa poberemo prvih `k`. Šlo bi tudi z `uvoznice[-1:-k-1:-1]`.

Vendar je boljše, če že `sort`-u naročimo urejati padajoče. Skok na v Pythonovo dokumentacijo pove, kako.

```

def naj_uvoznice(leto, k):
    teza = preberi_tezo(leto)

    uvoznice = []
    for drzava, teza in teza.items():

```

```

        uvoznice.append((teza, drzava))
    uvoznice.sort(reverse=True)

    prvih_k = []
    for _, drzava in uvoznice[:k]:
        prvih_k.append(drzava)
    return prvih_k

```

Funkcija se še vedno prav nemarno vleče. Spomnimo se, kaj smo se učili prejšnji teden, pa bo postala prijaznejša.

```

def naj_uvoznice(leto, k):
    teza = preberi_tezo(leto)
    uvoznice = [(teza, drzava) for drzava, teza in teza.items()]
    uvoznice.sort(reverse=True)
    return [drzava for _, drzava in uvoznice[:k]]

```

Kdor hoče narediti vse skupaj v enem zamahu, napiše

```

def naj_uvoznice(leto, k):
    return [drzava
            for _, drzava in sorted(
                ((teza, drzava) for drzava, teza in preberi_tezo(leto).items()),
                reverse=True
            )[:k]]

```

To je sicer zabavno, ni pa pregledno. V Pythonu ne programiramo tako, Python za to ni primeren. Koda je nepregledna, saj se najprej zgodi tisto, kar je najbolj znotraj, namreč `preberi_tezo`, potem pa se program izvaja "navzven" odtod. (Kdor ne razume, naj se ne vznemirja.)

Pač pa je zelo Pythonovsko narediti tole:

```

def naj_uvoznice(leto, k):
    teza = preberi_tezo(leto)
    return sorted(teza, key=teza.get, reverse=True)[:k]

```

Kdor je sprogramiral tako, zasluži bodisi pohvalo bodisi grajo. Pohvalo, če razume, kaj je naredil, in grajo, če je to nekje našel in uporabil. Slednji namreč ne more biti nikoli prepričan, da to res vedno naredi to, kar bi moralo ...

izpis

1. Napišite funkcijo `izpis(drzave, leto)`, ki prejme seznam držav in leto. Funkcija mora **vrniti niz** (ne izpisati!) takšne oblike:

| | | |
|-------------|------|------|
| Slovenia | 98 | 0.79 |
| China | 1768 | 0.61 |
| Germany | 1367 | 0.85 |
| Netherlands | 1072 | 0.85 |

| | | |
|-------|------|------|
| Japan | 1033 | 0.93 |
|-------|------|------|

Prvi stolpec je ime države, drugi teža uvoženih banan v gigagramih (kilotonah, 1000 tonah), tretji je cena v dolarjih na kilogram.

Pazi na poravnave in presledke: ti so postavljeni pravilno, če funkcija prestane teste.

Rešitev V tej nalogi očitno preverjamo, ali študentka, študent zna oblikovati nize.

```
def izpis(drzave, leto):
    teza = preberi_tezo(leto)
    vrednost = preberi_vrednost(leto)
    besedilo = ""
    for drzava in drzave:
        besedilo += f"{drzava:15} {teza[drzava] // 1_000_000:5} {vrednost[drzava] / teza[drzava]:5.2f}\n"
    return besedilo
```

prvih_n

1. Napišite funkcijo `prvih_n(leto, n)`, ki izpiše tabelo, kot je gornja, za `n` držav, ki so v podanem letu uvozile največ banan. Seznam naj bo urejen padajoče po teži.

Rešitev Ta naloga pa preverja, ali študent zna poklicati funkcijo, ki jo je sam napisal. Točneje, ali mu pride na misel, da je vse, kar potrebuje, že pripravljeno.

```
def prvih_n(leto, n):
    return izpis(naj_uvoznice(leto, n), leto)
```

Ocena 8

1. Napišite funkcijo `trend(drzava, od, do)`, ki vrne seznam s količino banan, ki jih je država uvozila v letih med `od` in `do` (vključno z `do`). Če za kako leto ni podatka, naj bo za tisto leto v seznamu `None`.
2. Napišite funkcijo `skupni_uvoz(leto)`, ki za podano leto vrne skupno težo vseh uvoženih banan v vseh državah.
3. Napišite funkcijo `rast_porabe(od, do)`, ki vrne seznam s skupnim uvozom banan po vsem svetu med podanima letoma.
4. Napišite funkcijo `inflacija(od, do)`, ki vrne seznam s povprečno ceno banan v dolarjih na kilogram med podanima letoma. Povprečna cena je enaka skupni vrednosti uvoza (vsota prek vseh držav) deljena s skupno težo uvoženih banan.

Rešitev

Ocena 8 je malo podarjena. :)

```
def trend(drzava, od, do):
    return [preberi_tezo(leto).get(drzava) for leto in range(od, do + 1)]

def skupni_uvoz(leto):
    return sum(preberi_tezo(leto).values())

def rast_porabe(od, do):
    return [skupni_uvoz(leto) for leto in range(od, do + 1)]

def inflacija(od, do):
    return [sum(preberi_vrednost(leto).values()) / sum(preberi_tezo(leto).values())
            for leto in range(od, do + 1)]
```

Rešitve smo zapisali z izpeljanimi seznammi. Kdor imaraje `append`, naj pač kliče `append`.

Ocena 9

Napišite funkcijo `tabela(od, do, drzave, ime_dat)`, ki v datoteko z imenom `ime_dat` izpiše podatke o uvozu banan v tonah za podane države v podanem intervalu let. Oblika tabela mora biti do presledka kot v spodnjem primeru in v podanih datotekah `test1.txt`, `test2.txt` in `test3.txt`. (Te datoteke se uporabljajo v testih, zato jih ne spreminjaj!!!)

Teža v tonah naj bo zaokrožena navzdol, torej `teza_v_kg // 1000`. Če teža ni znana ali pa je enaka 0, na tistem mestu ne izpišite ničesar (razen ustreznega števila presledkov).

Klic

```
tabela(1992, 2000,
       ["Slovenia", "Germany", "Japan", "United States"],
       "tabela.txt")
```

v `tabela.txt` shrani tole:

| Leto | Slovenia | Germany | Japan | United Sta |
|------|----------|---------|--------|------------|
| 1992 | | 1378870 | 777477 | 3693120 |
| 1993 | | 1222890 | 913612 | 3676980 |
| 1994 | 25227 | 1175620 | 929799 | 3862050 |
| 1995 | 31298 | 1306440 | 874108 | 3836570 |
| 1996 | 29645 | 1244570 | 819086 | 3966400 |
| 1997 | 30287 | 1133170 | 885454 | 3957720 |
| 1998 | 25391 | 1033120 | 865310 | 4113060 |
| 1999 | 28693 | 1034380 | 983504 | 4512440 |

| | | | | |
|------|-------|---------|---------|---------|
| 2000 | 26674 | 1113840 | 1079060 | 2268700 |
|------|-------|---------|---------|---------|

Ime države je skrajšano na prvih deset znakov.

Funkcija mora delovati za poljubno veliko število držav!

Rešitev

```
def tabela(od, do, drzave, ime_dat):
    f = open(ime_dat, "w")
    f.write("Leto " + "".join(f"{drzava[:10]:>12}" for drzava in drzave) + "\n")
    f.write("-" * (6 + 12 * len(drzave)) + "\n")
    for leto in range(od, do + 1):
        podatki = preberi_tezo(leto)
        f.write(f"{leto:<6}" + "".join(f"{podatki.get(drzava, 0) // 1000 or ':12}' for drzava in drzave) + "\n")
    f.write("-" * (6 + 12 * len(drzave)) + "\n")
```

Glavnina dogajanja je v join-ih.

- `drzava[:10]` je prvih deset znakov imena države.
- `{drzava[:10]:>12}` bo to ime izpisalo na 12 mest in poravnalo na desno kot hoče naloga.
- `f"{drzava[:10]:>12}" for drzava in drzave` generira takšne nize za vse države.
- `"".join(f"{drzava[:10]:>12}" for drzava in drzave)` pa to zlepi skupaj.

Kdor noče tako, lahko namesto

```
f.write("Leto " + "".join(f"{drzava[:10]:>12}" for drzava in drzave) + "\n")
```

piše v datoteko državo za državo

```
f.write("Leto ")
for drzava in drzave:
    f.write(f"{drzava[:10]:>12}")
f.write("\n")
```

Podobno se dogaja v drugem join, kjer nas potem zafrkavajo še prazne celice. Načelno bi napisali `podatki[drzava] // 1000`, vendar se lahko zgodi, da podatka za državo ni. Zato uporabimo `get`, kot privzeto vrednost pa nastavimo 0, torej `podatki.get(drzava, 0) // 1000`. Podatka ni, bo `get` vrnil 0 in deljenje bo dalo rezultat 0. Zato dodamo `or ""`. Ker je 0 neresnična vrednost, je 0 `or ""` enako `""`.

Tole je en čuden hack, vendar deluje ... A ne povsem pravilno. Priznam: tako sem sprogramiral, nato pa so me študenti opozorili, da se v njihovih rešitvah ponekod pojavi 9 namesto prazne celice - in sicer pri državah, za katere podatek obstaja, vendar je manjši od 1000. Mora "rešitev" v tem primeru dobi količnik 0 in ga zamenja s praznim nizom.

Zgodba naj bo v svarilo pred takšnimi rokohitrstvi, za katere smo dovzetni vsi, tudi tisti, nekoliko bolj izkušeni. Ker so bili testi že objavljeni, sem raje "dopolnil" navodila in zahteval, da so prazne celice tudi pri državah, ki so uvozila manj kot 1000 ton banan.

```
def tabela(od, do, drzave, ime_dat):
    f = open(ime_dat, "w")
    f.write("Leto ")
    for drzava in drzave:
        f.write(f"{drzava[:10]:>12}")
    f.write("\n")
    f.write("-" * (6 + 12 * len(drzave)) + "\n")
    for leto in range(od, do + 1):
        podatki = preberi_tezo(leto)
        f.write(f"{leto:<6}")
        for drzava in drzave:
            if drzava not in podatki:
                f.write(" " * 12)
            else:
                f.write(f"{podatki[drzava] // 1000:12}")
        f.write("\n")
    f.write("-" * (6 + 12 * len(drzave)) + "\n")
```

Ta rešitev ne prestane testov, vendar dela to, kar je bilo prvotno mišljeno in je tudi smiselno. Hkrati pa je preglednejša.

Moja napaka kaže, da se učimo programirati vse življenje. Tudi po 40 letih izkušenj še vedno delamo trapaste napake.

Ocena 10

Napišite funkcijo `tabela_drzav(od, do, drzave, ime_dat)`, ki prav tako zapiše podatke v datoteko, vendar drugačni obliki:

| Država | 1992 | 1993 | 1994 | 1995 | 1996 | 1997 | 1998 | 1999 |
|---------------|---------|---------|---------|---------|---------|---------|---------|---------|
| Slovenia | | | 25227 | 31298 | 29645 | 30287 | 25391 | 28123 |
| Germany | 1378870 | 1222890 | 1175620 | 1306440 | 1244570 | 1133170 | 1033120 | 1034560 |
| Japan | 777477 | 913612 | 929799 | 874108 | 819086 | 885454 | 865310 | 983456 |
| United States | 3693120 | 3676980 | 3862050 | 3836570 | 3966400 | 3957720 | 4113060 | 4512340 |

Rešitev

Ta naloga je težja od prejšnje, ker izpisovanje teče po vrsticah. V prejšnji nalogi so v vrsticah leta, torej lahko znotraj zanke (za vsako vrstico) preberemo ustrezna podatke. Tule pa so leta v stolpcih, torej moramo bodisi stalno brati podatke,

za vsako celico posebej -- ali pa si pripravimo slovar, katerega ključi so leta, vrednosti pa slovarji s podatki za to leto,

```
podatki = {leto: preberi_tezo(leto) for leto in range(od, do + 1)}
```

Ostanek je podoben kot prej, vključno z mojo napako.

```
def tabela_drzav(od, do, drzave, ime_dat):
    f = open(ime_dat, "w")
    f.write(f"{\"Država\":15} " + ".join(f\"{leto:10}\" for leto in range(od, do + 1)) + "\n")
    f.write("-" * (15 + 10 * (do - od + 1)) + "\n")
    podatki = {leto: preberi_tezo(leto) for leto in range(od, do + 1)}
    for drzava in drzave:
        f.write(f"{drzava[:13]:15} "
                + ".join(f\"{int(podatki[leto].get(drzava, 0) // 1000) or \":10}\" for leto i
                + "\n")
    f.write("-" * (15 + 10 * (do - od + 1)) + "\n")
```