

Zakaj se datoteke zaprejo same (in kdaj)?

Eden hujših grehov, ki jih naredim pri Programiranju 1, je povezan z odpiranjem datotek. Odpirali smo jih z

```
v = 0
for vrstica in open("stevilke.txt"):
    v += int(vrstica)
```

Pri tem povem, da se z zapiranjem datotek v Pythonu ne obremenjujem, ker se itak zaprejo same. To ni lepo.

Ni lepo, je pa res. Python redno pospravlja pomnilnik. Ko naleti na objekt, na katerega se (ne neposredno ne posredno) ne nanaša nobeno ime več, ga odstrani iz pomnilnika (= označi pomnilnik kot prost), predtem pa pokliče njegov *destruktor*. Destruktor datotek zapre datoteko.

Če je bilo to preveč učeno, pokažimo, kako vse se to lahko zgodi.

```
f = open("stevilke.txt")
...
f = 42
```

Ker se ime `f` po prirejanju `f = 42` ne nanaša več na datoteko, do datoteke ne moremo več priti (razen, če smo jo na kakšen način privezali k življenju v delu, označenem s tremi pikicami). Zato Python v onem trenutku zapre datoteko.

```
def beri(ime_datoteke):
    f = open(ime_datoteke)
    ...
```

Podobna reč: na datoteko se nanaša le lokalno ime `f`. Ko se funkcija konča, datoteka ni več dostopna, torej se zapre.

In, končno, zgled z začetka zapiska:

```
for vrstica in open("stevilke.txt"):
    ...
```

Dokler teče zanka, je živ generator, ki vrača vrstice datoteke. (Poenostavimo, povejmo preprosteje, čeprav ne bo čisto prav: za datoteko "ve" zanka `for`). Ko se zanka `for` konča, do datoteke ne moremo več, torej se zapre.

Ker datoteke praktično nikoli ne odpiram tako, da bi ostala dostopna tudi po tem, ko je ne potrebujem več, se z zapiranjem ne obremenjujem.

In? Zakaj to ni OK?

Dobro vprašanje. Saj je. Deluje. Vendar je to zapiranje le nekako stranski učinek nedosegljivosti objekta. To je dovolj dobro za skripte, ki si jih napišemo tako, na hitro, ne pa za resne projekte.

Obstaja tudi močnejši protiargument kot zgolj "ni lepo". Python ne zagotavlja, da bo pospravil datoteko iz pomnilnika točno takrat, ko bo `f` postal 42 ali pa ko se bo iztekla funkcija ali zanka. V trenutni različici Pythona, napisanem v C-ju, je tako. Za Jython (Python v Javi) nisem prepričan, za PyPy (Python, napisan v Pythonu) pa *vem*, da se pomnilnika ne pospravlja sproti, temveč občasno. In prav nihče nam ne zagotavlja, da ne bo tudi Python v neki prihodnji različici prešel s sprotnega na občasno pospravljanje.

Ročno zapiranje datotek

V večini jezikov, ki zahtevajo ročno zapiranje datotek, to počnemo takole.

```
f = open("stevilke.txt")

v = 0
for vrstica in f:
    v += int(vrstica)

f.close()
```

`open` in `close`, logično. In? Je mar to tako težko? Zakaj se ne učimo tako?

Tu imam dilemo. Po eni strani se hočemo učiti programiranja, ne Pythona. Če se z datotekami v splošnem dela tako, je prav, da se tako tudi naučimo, ne?

Da, vendar se vseeno učimo programirati v Pythonu in ne bi vas rad učil *slabega* Pythona. V modernem Pythonu (je od leta 2008, Python 2.6 :) datoteke zapiramo drugače. Tega, drugačnega načina pa ne boste videli v drugih jezikih.

In, končno, ta, pravi način zahteva sintaktični element, ki se ga sicer ne učimo, ker za potrebe Programiranja 1 ni preveč zanimiv. Zato sodi v rubriko "tole vam bom pokazal, če bo čas". Zdaj je.

Datoteke in `with`

Prav naredimo tako:

```
with open("stevilke.txt") as f:
    v = 0
    for vrstica in f:
        v += int(vrstica)

print(v)
```

76

Temu, kar je znotraj bloka `with` rečemo *kontekst*. Idejo bomo razložili v ločenem zapisku, tule le na hitro: kontekst je nekaj, kar se vzpostavi in, ko je bloka konec, pospravimo. Tisti reči, ki sledi `with`-u (v tem primeru: datoteko) bo `with` sporočal, da se začenja izvajanje bloka in, ko je bloka konec, da se je izvajanje

bloka končalo. V tem konkretnem primeru torej bo `with` sporočil datoteki, da je bloka konec in datoteka bo vedela, da je čas, da se zapre.

`as f` služi temu, da ima datoteka znotraj bloka ime.

Namerno sem dodal `print(v)` in to, namerno, izven `with`. Življenjska doba pythonovih spremenljivk ni omejena na bloke in `v` je dostopen tudi po `with`, čeprav je nastal v njem. (Isto velja za `f`: tudi ta obstaja tudi po bloku, vendar z njim nimamo kaj početi, saj je `with` datoteko zaprl.

Če boste torej hoteli lepo programirati v Pythonu, boste datoteke uporabljali znotraj kontekstov, v blokkih `with`. Če programirate nekaj na hitro, pa bo čisto dobro tudi tako, kot smo delali doslej.