05 Zanka ‘while‘

January 28, 2024

0.1 “Ta druga zanka”

Zanka `for` zna iti prek vrstic datoteke, znakov niza, ključev slovarja, elementov seznama. In še prek česa. Značilnost zanke `for` je, da “gre”. Zahteva neko zaporedje, pa bo ponavljala nek blok kode in pred vsako ponovitvijo neki spremenljivki (ali večim) priredila naslednjo vrednost (ali vrednosti) iz zaporedja.

So pa situacije, ko bi radi nekaj ponavljali, vendar ne tako, da bi pri tem šli prek “nečesa”. Vsaj ne prek nekega zaporedja.

0.2 Kaj je domneval Collatz

Vzemimo poljubno število in z njim počnimo tole: če je sodo, ga delimo z 2, če je liho, pa ga pomnožimo s 3 in prištejmo 1. To ponavljamo, dokler ne dobimo 1.

Za primer vzemimo 12. Ker je sodo, ga delimo z 2 in dobimo 6. 6 je sodo, torej ga delimo z 2 in dobimo 3. 3 je liho, torej ga množimo s 3 in prištejemo 1 - rezultat je 10. 10 je sodo, zato ga delimo z 2 in dobimo 5… Celotno zaporedje je 12, 6, 3, 10, 5, 16, 8, 4, 2, 1. Ko pridemo do 1, se ustavimo.

Matematiki, ki vedno radi počnejo koristne reči, si belijo glavo z vprašanjem, ali se reč vedno izteče ali pa se lahko kdaj tudi zacikla tako, da se zaporedje ponavlja in ponavlja v nedogled. Lothar Collatz, konkretno je že od leta 1937 domneval, da se zaporedje vedno izteče. Kot pravi njegov življenjepis, se je njegovo domnevanje končalo leta 1990, matematiki pa domnevajo domnevo še naprej. Eden sicer najbolj neustrašnih matematikov 20. stoletja, couch surfer Erdos, je na to temo rekel, da matematika za takšna vprašanja še ni zrela.

Naši cilji bodo skromnejši: napisali bomo program, ki mu bo uporabnik vpisal število in program bo izpisal zaporedje, kot je bilo gornje.

Najprej povejmo kar po slovensko:

[ ]:
```
stevilo = kar vpiše uporabnik
dokler stevilo != 1:
    če je število sodo:
        deli število z 2
    sicer:
        pomnoži število s 3 in prištej 1
```

Tole skoraj že znamo prevesti v Python (prevod bo pravzaprav dobeseden), manjka nam le vrstica “dokler je število sodo”. Ta je slišati sumljivo podobno kot “če je število sodo”, le namesto “če” imamo “dokler”. Poskusimo.

[1]:
```
n = int(input("Vnesi število: "))
while n != 1:
    print(n)
    if n % 2 == 0:
        n //= 2
    else:
        n = n * 3 + 1
print(1)
```

```
Vnesi število: 42
42
21
64
32
16
8
4
2
1
```

Ne spreglejte, da smo uporabili celoštevilsko deljenje. Če bi uporabili navadnega, `n /= 2`, bi se `n` spremenil v necelo število (`float`) in v izpisu bi se pojavile zoprne decimalke (za vsakim številom bi pisalo še .0).

Popaziti je potrebno, da izpišemo tako prvo kot zadnje število, zato bomo potrebovali še en `print` izven zanke: po zanki izpišemo še zadnji člen zaporedja, 1. Namesto `print(1)` bi lahko napisali tudi `print(n)` - tako ali tako vemo, da je `n` enak 1, saj program sicer ne bi prišel do tam, do koder je prišel, namreč onstran zanke, ki teče, dokler je `n` različen od 1.

Zanka `while` torej po svoje podobna stavku `if`. Medtem ko se stavki znotraj `if` izvedejo, *če* je pogoj resničen, se stavki znotraj `while` ponavljajo, *dokler* je pogoj resničen.