12. Zanka for na splošno

Zanko for smo spoznali že pri nizih. Tudi za sezname in druge sorodne tipe deluje na enak način. Z njo pregledamo vse elemente izbranega seznama.

```
for element in seznam:
      stavek1
      stavek2
```

Seznama, ki ga pregledujemo, v zanki ni varno spreminjati.

12.1. Funkcija range

Funkcija range vrne objekt, ki zna šteti. Pravimo, da je iterabilen. Uporabimo ga lahko namesto seznama v zanki for.

- **`range(10)`** ... šteje od 0 do 9
- **`range(1, 11)`** ... šteje od 1 do 10
- **`range(6, 30, 2)`** ... šteje od 6 do 29 s korakom 2
- **`range(30, 6, -2)`** ... šteje od 30 do 7 s korakom -2

Primer: Spodnji zanki naredita isto (izpišeta števila od 0 do 4). Razlika je v tem, da v prvem primeru sestavimo seznam, po katerem se nato premikamo, v drugem primeru pa samo štejemo, zato je druga možnost hitrejša.

```
for x in [0,1,2,3,4]: print(x)
for x in range(5): print(x)
```

Pozor: **`range`** ne vrne seznama!

`list(range(...))` vrne seznam števil, ki jih dobimo pri štetju.

12.2. Zanke - še enkrat

S stavkom **`break`** prekinemo izvajanje zanke. Program se nadaljuje s prvim stavkom, ki sledi zanki.
S stavkom **`continue`** prekinemo tekočo ponovitev zanke. Zanka for se ponovi z naslednjim el.ementom seznama (če je še kakšen), zanka while pa se ponovi, če je pogoj še vedno izpolnjen.
S stavkom **`else`** takoj za zanko določimo, kaj naj se zgodi, če se stavek break v zanki ni izvedel.

12.3. Gnezdenje zank

Zanke lahko tudi gnezdimo. Paziti moramo, kako uporabljamo števce.

12.4. Nekaj uporabnih funkcij: npr. zip in enumerate.

V globalnem paketu imamo na razpolago še nekaj uporabnih funkcij:

- **`input(s)`** ... izpiše niz s, počaka, da uporabnik nekaj vnese (vnos zaključi s tipko ENTER), ter vrne vnesen niz
- **`eval(s)`** ... izračuna in vrne vrednost izraza, zapisanega v nizu s
- **`zip(a, b)`** ... vrne objekt, sestavljen iz parov istoležnih elementov iz seznamov a in b
- **`enumerate(a)`** ... vrne objekt, sestavljen iz parov (indeks, element) po vseh elementih seznama a

12.5. Naloge

1. Prvi člen zaporedja je poljubno naravno število, vsak naslednji člen pa dobimo iz prejšnjega tako, da mu prištejo produkt njegovih neničelnih števk. Sestavi funkcijo, ki bo za dani prvi člen zaporedja izračunala in izpisala vse člene, ki so manjši ali enaki dani zgornji meji.
2. Sestavite funkcijo, ki izračuna in vrne seznam vseh primitivnih pitagorejskih trojk.
3. Sestavite funkcijo, ki iz danega naravnega števila sestavi niz, ki vsebuje število zapisano v rimski obliki.
4. Sestavite funkcijo, ki si bo izmislila celo število med 0 in 99 ter od uporabnika zahtevala, naj ga ugane. Po vsakem neuspešnem poskusu naj izpiše, ali je njegovo število premajhno ali preveliko. Število poskusov naj bo omejeno.