

Za radoznale: Petlje

Ovaj fajl nije obavezan. Sa petljama while i for i funkcijom range može da se napravi svako ponavljanje koje nam treba i na času smo radili baš tako. Ovde za nekoliko primera sa časa pokazujemo kraće zapise koje koriste programeri sa više iskustva. Ako vam nešto nije jasno, slobodno preskočite, jer većina ovih stvari dolazi uz liste i kolekcije, u temama 09 i 10. Svaki primer je i u folderu `primeri/za_radoznale`, pa ga možete pokrenuti i menjati.

1. Redni broj uz svaki element: enumerate

Na času je petlja for prolazila kroz spisak boja, jednu po jednu.

Na času:

```
for boja in ["crvena", "zelena", "plava"]:  
    print("Slikam", boja)
```

Ispis:

```
Slikam crvena  
Slikam zelena  
Slikam plava
```

Ako uz svaku boju hoćemo i redni broj, funkcija enumerate daje oba odjednom, broj i boju:

```
for broj, boja in enumerate(["crvena", "zelena", "plava"], 1):  
    print(broj, "Slikam", boja)
```

Ispis:

```
1 Slikam crvena  
2 Slikam zelena  
3 Slikam plava
```

U svakom krugu petlje enumerate daje par: redni broj i element. Broj 1 na kraju kaže da brojanje počne od 1, a ne od 0. Bez enumerate bismo morali sami da vodimo brojač, na primer `i = 1` pre petlje i `i += 1` u telu.

Zašto ovo nismo radili na času: enumerate daje parove vrednosti (to su torke, tema 10), a petlja ih raspakuje u dve promenljive. Prvo treba dobro razumeti običnu petlju for.

Fajl: `primeri/za_radoznale/08_r1_enumerate.py`

2. Dva spiska u isto vreme: zip

Kada imamo dva spiska iste dužine, na času bismo prolazili kroz indekse pomoću range i len.

Na času:

```
predmeti = ["nebo", "trava", "sunce"]
boje = ["plava", "zelena", "žuta"]
for i in range(len(predmeti)):
    print(predmeti[i] + ":", boje[i])
```

Ispis:

```
nebo: plava
trava: zelena
sunce: žuta
```

Funkcija zip spaja dva spiska kao parove: prvi sa prvim, drugi sa drugim i tako redom. Indeksi nam više ne trebaju:

```
predmeti = ["nebo", "trava", "sunce"]
boje = ["plava", "zelena", "žuta"]
for predmet, boja in zip(predmeti, boje):
    print(predmet + ":", boja)
```

Ispis:

```
nebo: plava
trava: zelena
sunce: žuta
```

U svakom krugu zip daje po jedan element iz oba spiska. Red sa for se čita kao rečenica: „za svaki predmet i njegovu boju“. Ako spiskovi nisu iste dužine, zip staje kod kraćeg.

Zašto ovo nismo radili na času: zip radi sa listama, koje dolaze u temi 09. Kao i enumerate, daje parove vrednosti, a njih upoznajemo u temi 10.

Fajl: `primeri/za_radoznale/08_r2_zip.py`

3. Petlja kroz rečnik

Na času smo petljom prolazili kroz spisak imena boja.

Na času:

```
for boja in ["plava", "zelena", "žuta"]:
    print(boja)
```

Ispis:

```
plava
zelena
žuta
```

Paleta u igri obično čuva i ime dela scene i kod njegove boje. Za takve parove postoji rečnik, a petlja može da prođe kroz sve njegove parove:

```
paleta = {"nebo": "#87CEEB", "trava": "#7CB342", "sunce": "#FFC857"}
for deo, kod in paleta.items():
    print(f"{deo}: {kod}")
```

Ispis:

```
nebo: #87CEEB
trava: #7CB342
sunce: #FFC857
```

Rečnik (u vitičastim zagradama) čuva parove ključ: vrednost, ovde ime dela scene i kod boje. Metoda items daje sve parove, a petlja svaki par raspakuje u dve promenljive, deo i kod.

Zašto ovo nismo radili na času: rečnike radimo u temi 10. Ovaj primer je tu da vidite da petlja for ume da prođe kroz mnogo više od spiska brojeva.

Fajl: `primeri/za_radoznale/08_r3_petlja_kroz_recnik.py`

4. Lista u jednom redu (comprehension)

Imena fajlova za kadrove sprajta na času bismo ispisali petljom, red po red.

Na času:

```
for kadar in range(1, 5):
    print(f"heroj_{kadar:03}.png")
```

Ispis:

```
heroj_001.png
heroj_002.png
heroj_003.png
heroj_004.png
```

Ako ta imena ne želimo samo da ispišemo, nego da ih sačuvamo u jednoj listi, Python ima kratak zapis koji spaja petlju i pravljenje liste:

```
kadrovi = [f"heroj_{k:03}.png" for k in range(1, 5)]
print(kadrovi)
print(len(kadrovi), "kadra")
```

Ispis:

```
['heroj_001.png', 'heroj_002.png', 'heroj_003.png', 'heroj_004.png']
4 kadra
```

Zapis u uglastim zagradama čita se kao „napravi listu od f"heroj_{k:03}.png" za svako k iz range(1, 5)". Na engleskom se zove list comprehension. Rezultat je lista od četiri stringa, koju posle možemo da koristimo u programu.

Zašto ovo nismo radili na času: ovo je lista i petlja u jednom redu, pa prvo treba dobro znati oba. Vraćamo se na ovaj zapis posle teme 09.

Fajl: `primeri/za_radoznale/08_r4_lista_u_jednom_redu.py`

5. Romb bez računanja razmaka: center

U vežbi Level 3 smo za svaki red romba sami računali koliko razmaka ide ispred znakova +.

Na času:

```
n = 7 # neparan broj
sredina = n // 2
for red in range(n):
    rastojanje = abs(sredina - red)
    print(" " * rastojanje + "+" * (n - 2 * rastojanje))
```

Ispis:

```
  +
 +++
+++++
+++++++
+++++
  +++
   +
```

Metoda center stavlja string na sredinu reda zadate širine, pa razmake ne moramo da računamo. Dovoljne su dve petlje, jedna za gornju i jedna za donju polovinu:

```
n = 7 # neparan broj
for k in range(1, n + 1, 2):
    print(("+" * k).center(n))
for k in range(n - 2, 0, -2):
    print(("+" * k).center(n))
```

Ispis:

```
  +
 +++
+++++
+++++++
+++++
  +++
   +
```

Prva petlja ide 1, 3, 5, 7, a druga 5, 3, 1, jer range sa korakom -2 broji unazad. ("+" * k).center(n) napravi k znakova + i stavi ih na sredinu reda širokog n znakova.

Zašto ovo nismo radili na času: metodu center nismo radili, a računanje razmaka je dobra vežba razmišljanja: tako se vidi kako red i broj znakova zavise jedan od drugog.

Fajl: `primeri/za_radoznale/08_r5_romb_sa_center.py`