

Za radoznale: Tipovi, promenljive i stringovi

Ovaj fajl nije obavezan. Na času smo naučili osnovne alate za rad sa brojevima i stringovima i to je sasvim dovoljno za sve naše programe. Ovde za nekoliko primera sa časa pokazujemo kako bi isto napisao neko ko programira duže. Ako vam nešto nije jasno, slobodno preskočite, jer se na deo ovih stvari vraćamo u kasnijim temama. Svaki primer je i u folderu `primeri/za_radoznale`, pa ga možete pokrenuti i menjati.

1. Reči iz rečenice: split i join

Na času smo stringove spajali znakom `+`, a razmak smo dodavali sami.

Na času:

```
print("na" + " " + "primer")
print("-" * 20)
```

Ispis:

```
na primer
-----
```

Metoda `split` radi obrnuto: deli string na reči. Metoda `join` ih ponovo spaja, sa stringom po izboru između njih. Ovaj primer je iz starih beleški sa časova:

```
s = "Programiranje za umetnike 1"
reci = s.split()
print(reci)
print(reci[-1] + " " + reci[-3] + " " + reci[2])
print("**".join(["Ovo", "je", "primer"]))
print("-".join(reci))
```

Ispis:

```
['Programiranje', 'za', 'umetnike', '1']
1 za umetnike
Ovo**je**primer
Programiranje-za-umetnike-1
```

`split` vraća spisak reči u uglastim zagradama. Takav spisak se zove lista i, kao string, ima indekse, pa je `reci[-1]` poslednja reč. `join` se piše malo neobično: ispred tačke je string koji ide između reči, a u zagradi lista reči.

Zašto ovo nismo radili na času: `split` vraća listu, a liste radimo tek u temi 09. Kad budemo znali liste, ovaj primer će biti sasvim jednostavan.

Fajl: `primeri/za_radoznale/06_r1_split_join.py`

2. Naslov na sredini: center, ljust i rjust

Na kartici lika smo ukrasnu liniju pravili množenjem stringa, a ime ispisivali u posebnom redu.

Na času:

```
ime = "Zmaj"  
print("=" * 28)  
print(f"Lik: {ime.upper()}")  
print("=" * 28)
```

Ispis:

```
=====  
Lik: ZMAJ  
=====
```

Metoda center stavlja string na sredinu reda zadate širine, a prazno mesto popunjava znakom po izboru. Metode ljust i rjust guraju tekst levo, odnosno desno:

```
ime = "Zmaj"  
print(ime.upper().center(28, "="))  
print("Boja:".ljust(12) + "#78C850")  
print("Kadrova:".ljust(12) + "12")  
print("kraj".rjust(28, "."))
```

Ispis:

```
=====ZMAJ=====  
Boja:      #78C850  
Kadrova:   12  
.....kraj
```

center(28, "=") pravi string dug tačno 28 znakova, sa imenom u sredini i znakovima = sa obe strane. ljust(12) dopunjava tekst razmacima do 12 znakova, pa vrednosti u drugoj koloni počinju na istom mestu i izgledaju kao tabela.

Zašto ovo nismo radili na času: na času smo imali mnogo novih metoda, a ove samo ulepšavaju ispis. Isti izgled se dobija i množenjem stringa, kao što smo radili.

Fajl: `primeri/za_radoznale/06_r2_center_ljust_rjust.py`

3. Poravnanje u f-stringu

Na času smo videli dva zapisa posle dvotačke u f-stringu: `:.2f` za dve decimale i `:03` za vodeće nule.

Na času:

```
kadar = 7  
print(f"{7 / 3:.2f}")  
print(f"heroj_{kadar:03}.png")
```

Ispis:

```
2.33  
heroj_007.png
```

Posle dvotačke može da stoji i širina polja sa smerom poravnanja: `<` levo, `>` desno i `^` na sredini. Tako se lako pravi tabela:

```
naslov1, naslov2 = "LIK", "SNAGA"
print(f"{naslov1:<10}{naslov2:>6}")
print(f"{'Zmaj':<10}{87:>6}")
print(f"{'Vila':<10}{9:>6}")
print(f"{'kraj':*^16}")
```

Ispis:

```
LIK      SNAGA
Zmaj      87
Vila      9
*****kraj*****
```

{naslov1:<10} znači: ispiši vrednost u polju od 10 znakova, priljubljenu uz levu ivicu. {87:>6} stavlja broj uz desnu ivicu polja od 6 znakova, pa se jedinice svih brojeva nađu jedna ispod druge. U poslednjem redu znak * ispred ^ kaže čime se popunjava prazno mesto oko reči.

Zašto ovo nismo radili na času: zapisa posle dvotačke ima mnogo i bilo bi ih previše odjednom. Na času smo uzeli dva koja nam trebaju za imena fajlova i decimale.

Fajl: `primeri/za_radoznale/06_r3_poravnanje_u_f_stringu.py`

4. Iz heksadecimalnog zapisa nazad u R, G i B

Na času smo od tri broja napravili zapis boje kakav koriste programi za crtanje i veb stranice.

Na času:

```
r, g, b = 255, 200, 87
print(f"#{r:02X}{g:02X}{b:02X}")
```

Ispis:

```
#FFC857
```

Ponekad nam treba obrnuto: dobili smo boju kao #FFC857, a treba nam R, G i B. Isecanjem uzmemo po dve cifre, a int ih pretvori u broj:

```
boja = "#FFC857"
r = int(boja[1:3], 16)
g = int(boja[3:5], 16)
b = int(boja[5:7], 16)
print(r, g, b)
```

Ispis:

```
255 200 87
```

Drugi argument 16 kaže funkciji int da string čita kao broj u heksadecimalnom sistemu, sistemu sa šesnaest cifara (od 0 do 9 i od A do F). Tako "FF" postaje 255. Isečak boja[1:3] preskače znak # i uzima prve dve cifre.

Zašto ovo nismo radili na času: ovde se odjednom spajaju isecanje, konverzija i drugi brojni sistem. Svaka od tih stvari je bila nova na času, a spajanje sve tri traži malo više vežbe.

Fajl: `primeri/za_radoznale/06_r4_hex_u_rgb.py`

5. Lepše ime: title, count i startswith

U vežbi Level 2 smo od unetog imena pravili inicijale: uzeli smo prvo slovo i pretvorili ga u veliko.

Na času:

```
ime = " mila"
print(ime.strip()[0].upper() + ".")
```

Ispis:

```
M.
```

String ima još mnogo metoda. Metoda `title` svaku reč počinje velikim slovom, `count` broji koliko puta se nešto pojavljuje, a `startswith` proverava kako string počinje:

```
ime = " mila jovanović "
cisto = ime.strip().title()
print(cisto)
print(cisto.count("a"))
print(cisto.startswith("Mila"))
print(cisto.startswith("mila"))
```

Ispis:

```
Mila Jovanović
2
True
False
```

Metode se nižu jedna za drugom i izvršavaju se sleva nadesno: prvo `strip` skine razmake, pa `title` popravi velika slova. `count("a")` broji samo malo slovo a, a `startswith("mila")` daje `False`, jer posle `title` ime počinje velikim M.

Zašto ovo nismo radili na času: metoda ima nekoliko desetina i nema smisla učiti ih sve. Na času smo uzeli one koje se najčešće koriste, a ostale se lako nađu u dokumentaciji kada zatrebaju.

Fajl: `primeri/za_radoznale/06_r5_title_count_startswith.py`