

Za radoznale: Uslovna grananja

Ovaj fajl nije obavezan. Sa naredbama `if`, `elif` i `else` može da se napiše svako grananje koje nam treba i na času smo radili baš tako. Ovde za nekoliko primera sa časa pokazujemo kraće ili drugačije zapise koje koriste programeri sa više iskustva. Ako vam nešto nije jasno, slobodno preskočite, jer deo ovih stvari dolazi u kasnijim temama. Svaki primer je i u folderu `primeri/za_radoznale`, pa ga možete pokrenuti i menjati.

1. Ulančani uslovni izraz

Na času smo poredili dva broja lancem `if`, `elif` i `else`, sa tri grane.

Na času:

```
a, b = 33, 33
if a > b:
    print("A je veće")
elif a == b:
    print("A je jednako B")
else:
    print("B je veće")
```

Ispis:

```
A je jednako B
```

Uslovni izraz bira jednu od dve vrednosti. Ako iza `else` stavimo još jedan uslovni izraz, dobijamo izbor između tri vrednosti, u jednom redu:

```
a, b = 33, 33
ispis = "A je veće" if a > b else "A je jednako B" if a == b else "B je veće"
print(ispis)
```

Ispis:

```
A je jednako B
```

Red se čita sleva nadesno: „A je veće“ ako je `a > b`, a inače „A je jednako B“ ako je `a == b`, a inače „B je veće“. U starim beleškama sa časova isti primer je napisan sa `print` u svakom delu, što je još teže za čitanje.

Zašto ovo nismo radili na času: ovakav red je teško pročitati čak i iskusnim programerima, a greška u njemu se teško nalazi. Lanac `if`, `elif` i `else` radi isto i mnogo je jasniji, pa njega koristimo.

Fajl: `primeri/za_radoznale/07_r1_ulancani_uslovni_izraz.py`

2. in umesto više uslova sa `or`

Na času smo više uslova spajali operatorom `or`: dovoljno je da jedan bude tačan.

Na času:

```
boja = "plava"
if boja == "crvena" or boja == "žuta" or boja == "plava":
    print("osnovna boja")
```

Ispis:

```
osnovna boja
```

Kada istu promenljivu poredimo sa više vrednosti, kraće je da te vrednosti stavimo u spisak i pitamo da li je promenljiva u njemu. Sa stringom, in pita da li je znak deo stringa:

```
boja = "plava"
if boja in ["crvena", "žuta", "plava"]:
    print("osnovna boja")

slovo = "e"
if slovo in "aeiou":
    print(slovo, "je samoglasnik")
```

Ispis:

```
osnovna boja
e je samoglasnik
```

Operator in daje True ako se vrednost nalazi u spisku. Spisak u uglastim zagradama je lista. U drugom delu in proverava da li se slovo nalazi u stringu "aeiou"; to smo već videli u temi 06, kod metoda stringa.

Zašto ovo nismo radili na času: spisak vrednosti je lista, a liste radimo u temi 09. Na času smo hteli da dobro uvežbamo operator or, jer je on osnova za sve složene uslove.

Fajl: `primeri/za_radoznale/07_r2_in_umesto_or.py`

3. Naredba match kao zanimljivost

Meni u igri bira šta će se desiti prema dugmetu koje je pritisnuto. Na času bismo to napisali lancem elif.

Na času:

```
dugme = "Podešavanja"
if dugme == "Igraj":
    print("Počinje igra")
elif dugme == "Podešavanja":
    print("Otvaram podešavanja")
elif dugme == "Izlaz":
    print("Doviđenja")
else:
    print("Nepoznato dugme")
```

Ispis:

```
Otvaram podešavanja
```

Od verzije 3.10 Python ima i naredbu match. Ona jednu vrednost redom poredi sa svakim slučajem (case):

```
dugme = "Podešavanja"
match dugme:
    case "Igraj":
        print("Počinje igra")
    case "Podešavanja":
        print("Otvaram podešavanja")
    case "Izlaz":
        print("Doviđenja")
    case _:
        print("Nepoznato dugme")
```

Ispis:

```
Otvaram podešavanja
```

Izvršava se samo prvi slučaj koji odgovara, kao kod lanca elif. Znak _ (podvlaka) odgovara bilo čemu, pa taj poslednji slučaj radi isto što i else.

Zašto ovo nismo radili na času: match je novija naredba i ne radi u starijim verzijama Pythona. Ume mnogo više od ovoga, ali sve što nama treba radimo sa if, elif i else.

Fajl: `primeri/za_radoznale/07_r3_match_case.py`

4. Prazne kolekcije su takođe neistinite

Na času smo videli da su None, nula i prazan string neistiniti, pa se izvršava grana else.

Na času:

```
if None or 0 or 0.0 or "":
    print("Ovo se ne ispisuje")
else:
    print("Sve ove vrednosti su neistinite")
```

Ispis:

```
Sve ove vrednosti su neistinite
```

Isto važi i za prazne kolekcije, vrednosti koje čuvaju više drugih vrednosti. Ovo je ceo primer iz starih beleški, sa praznom listom, praznim rečnikom i praznim skupom:

```
if None or 0 or 0.0 or "" or [] or {} or set():
    print("Ovo se ne ispisuje")
else:
    print("Sve ove vrednosti su neistinite")

boje = []
if boje:
    print("ima boja")
else:
    print("paleta je prazna")
```

Ispis:

```
Sve ove vrednosti su neistinite
paleta je prazna
```

[] je prazna lista, {} prazan rečnik, a set() prazan skup. Sve prazno Python tumači kao neistinu. Zato se često piše if boje: kada se pita „ima li nečega u paleti?“.

Zašto ovo nismo radili na času: liste radimo u temi 09, a rečnike i skupove u temi 10. Tada se vraćamo i na ovo pravilo.

Fajl: `primeri/za_radoznale/07_r4_prazne_kolekcije.py`

5. max i min umesto grananja

Na času smo najveći od tri broja tražili grananjem, uz operator and.

Na času:

```
a, b, c = 10, 14, 12
if a >= b and a >= c:
    najveći = a
elif b >= a and b >= c:
    najveći = b
else:
    najveći = c
print("Najveći broj je", najveći)
```

Ispis:

```
Najveći broj je 14
```

Python ima ugrađene funkcije max i min, koje to rade same. Sa njima možemo i da „stegnemo“ vrednost boje da ostane između 0 i 255:

```
a, b, c = 10, 14, 12
print("Najveći broj je", max(a, b, c))

r = 300
r = min(max(r, 0), 255)
print("r =", r)
```

Ispis:

```
Najveći broj je 14
r = 255
```

max vraća najveću, a min najmanju od vrednosti u zagradi. U izrazu min(max(r, 0), 255) prvo max ne da da r bude manje od 0, a zatim min ne da da bude veće od 255. Isto bismo dobili sa if i elif, u četiri reda.

Zašto ovo nismo radili na času: na času smo hteli da vežbamo pisanje uslova, a max i min sakrivaju grananje u sebi. Funkciju max smo ipak iskoristili u primeru RGB u HSV, gde nije bila glavna tema.

Fajl: `primeri/za_radoznale/07_r5_max_i_min.py`