

Beleške sa časa – 9.nedelja

Terminologija

- ugnježdene petlje = umetnute petlje = petlja u petlji
- format string/niska = string koji opisuje strukturu sadržaja stringa
- omotač funkcija = funkcija koja samo poziva druge funkcije uz malo ili bez ikakvog dodatnog izračunavanja

Ugnježdene petlje

- petlje mogu sadržati u sebi druge petlje (misli se unutar bloka naredbi koje čine telo petlje) i takav način korišćenja petlji naziva se *ugnježdavanje petlji*
- petlje se mogu ugnježdavati do proizvoljne dubine, tj. može se umetati proizvoljan broj petlji jedne u drugu
- takođe, prilikom ugnježdavanja moguće je kombinovati i različite vrste petlji (**for** i **while** petlje)
- prilikom ugnježdavanja petlji mora da se vodi računa o uvlačenju koda, tako da naredbe na svakom narednom nivou ugnježdavanja budu uvučene za jedan više tabulator
- kada su dve petlje umetnute jedna u drugu, nazivamo ih još *dvostrukom petljom*, tri ugnježdene petlje nazivamo *trostrukom petljom*, itd.
- kod dvostruke petlje se petlja koja je umetnuta naziva *unutrašnjom petljom*, dok se petlja u kojoj je ona umetnuta naziva *spoljašnjom petljom*
- u svakoj iteraciji spoljašnje petlje izvršavaju se sve iteracije unutrašnje petlje pre nego što se pređe na izvršavanje sledeće iteracije spoljašnje petlje

PRIMER: blok naredbi unutrašnje petlje se izvršava ukupno $n \cdot m$ puta

```
for i in range(n):  
    for j in range(m):  
        blok-naredbi-unutrasnje-petlje
```

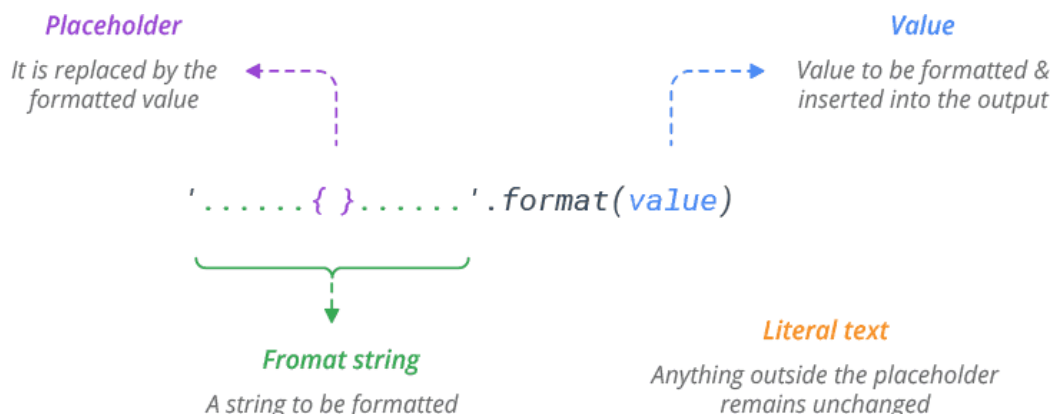
- dvostruke petlje se najčešće koriste kada postoji neka vrsta dvodimenzionalnosti u strukturi izračunavanja programa (npr. sa dva parametra koji uzimaju vrednosti iz nekih opsega) ili u podacima koje program obrađuje (npr. tabela, matrica)
- pri analiziranju izvršavanja ugnježđenih petlji često je lakše da se krene od unutrašnje petlje
- naredbe **break** i **continue** kada se koriste u ugnježđenim petljama imaju efekat samo na onu petlju u čijem telu su navedene

PRIMER: naredba **break** koja se nalazi u telu unutrašnje petlje prekinuće samo izvršavanje unutrašnje petlje, a ne i cele ugnježdene petlje

```
while (uslov-spoljasnje-petlje):  
    while (uslov-unutrasnje-petlje):  
        if uslov-prekida:  
            break
```

Formatiranje stringova

- za formatiranje stringova u Python-u koristi se metod **str** klase **format**
- metod **format** se poziva nad tzv. *format stringom*, stringom koji definiše strukturu (format) stringa
- format string sadrži dve vrste komponenti:
 - 1) doslovni tekst, tj. obične karaktere koji doslovno određuju sadržaj stringa
 - 2) placeholder komponente **{ }** koje određuju pozicije u stringu na kojima će biti umetnute neke vrednosti



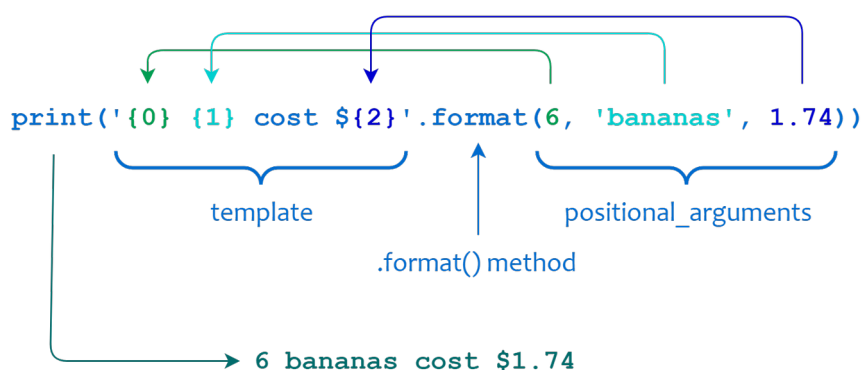
- kao argumenti metoda **format** zadaju se vrednosti koje treba umetnuti na odgovarajuće pozicije u format stringu, pri čemu broj argumenata metoda **format** mora da odgovara boju placeholder komponenti koje su zadate format stringom

- unutar vitičastih zagrada placeholder komponenti moguće je navesti specifikacije za format u kojem želimo da odgovarajuće vrednosti budu umetnute u string

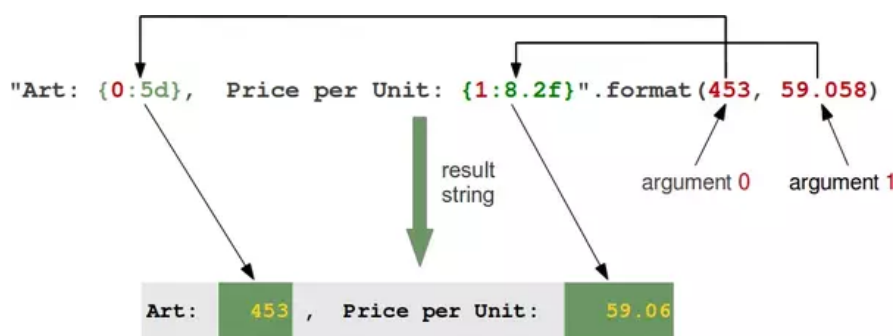
- struktura specifikacija za format umetnutih vrednosti: **{[index]:[width][.precision][type]}**

- **index** specifikuje redni broj argumenta **format** metoda koji se pridružuje datoj placeholder komponenti; ukoliko **index** nije naveden, podrazumevano se placeholder komponentama redom pridružuju argumenti **format** metoda
- **width** specifikuje širinu polja za umetanje tj. minimalan broj pozicija predviđenih za odgovarajuću umetnutu vrednost u okviru datog stringa
- **precision** specifikuje broj pozicija predviđenih za cifre iza decimalne tačke (preciznost) kod umetnutih vrednosti numeričkog tipa
- **type** specifikuje tip umetnute vrednosti (npr. **d** je oznaka za celobrojni tip, **f** je oznaka za decimalni tip, a **s** je oznaka za tip string)

PRIMER: zadavanje redosleda za umetnute vrednosti



PRIMER: zadavanje formata za umetnute vrednosti



- u strukturi specifikacije za format umetnutih vrednosti svaka od prethodno pobrojanih komponenti je navedena u okviru uglastih zagrada, što označava da je svaka od ovih komponenti opcionalna, tj. da neke od njih ili čak sve komponente mogu biti izostavljene

- pored metoda **format**, postoji još jedan način za formatiranje stringova, a to su tzv. *F-strings*¹, međutim ovaj koncept neće biti objašnjavan na ovom mestu

Imenovani argumenti funkcija

- u Python-u postoje dva načina za prosleđivanje argumenata funkciji:

- 1) u *pozicionom (neimenovanom) redosledu* koji podrazumeva da se argumenti prosleđuju funkciji u onom redosledu u kojem ih ona očekuje, tj. u redosledu u kojem su oni navedeni u definiciji te funkcije
- 2) u *nepozicionom (imenovanom) redosledu* koji podrazumeva da se prilikom poziva te funkcije argumenti mogu prosleđivati u proizvoljnom redosledu, uz pridruživanje vrednosti svakom od argumenata prema njihovom imenu (**ar_ime = vrednost**)

¹ Više o F-strings konceptu možete pročitati na sledećim linku: https://www.w3schools.com/python/python_string_formatting.asp

- argumenti funkcija kako smo ih do sada navodili su uvek bili u neimenovanom, pozicionom obliku
- da bi bilo moguće koristiti jedan i/ili drugi način za zadavanje argumenata funkciji, potrebno je da u definiciji te funkcije bude dopušteno (omogućeno) korišćenje jednog od ili oba načina za zadavanje argumenata funkciji (više o tome na sledećem [linku](#))
- najčešći slučaj je da funkcije dozvoljavaju oba načina za zadavanje argumenata, tako da se funkcija može pozvati na više načina koji su međusobno ekvivalentni
- moguće je kombinovati ova dva načina zadavanja argumenata, tako da neki od argumenata jednog istog poziva funkcije budu zadati na jedan način a neki argumenti na drugi način, pri čemu se prvo moraju navesti svi neimenovani argumenti u redosledu koji odgovara definiciji funkcije, a nakon njih imenovani argumenti u proizvoljnom redosledu
- u funkcijama standardne biblioteke često neki od argumenata zahtevaju isključivo imenovano zadavanje, i to su obično parametri koji predstavljaju nekakva podešavanja funkcije, kojima se modifikuje ponašanje (način izvršavanja) funkcije, i ti argumenti uobičajeno imaju pridružene podrazumevane vrednosti koje određuju podrazumevano ponašanje te funkcije

Imenovani argumenti funkcije **print**

- kao što već znamo, funkcija **print** može prihvatiti neograničen broj argumenata za ispis
- svi argumenti funkcije **print** se prilikom ispisa podrazumevano odvajaju sa po jednim razmakom (tzv. *separator*), a nakon poslednjeg ispisanog argumenta podrazumevano ispisuje se prelazak u novi red (tzv. *sekvenca za kraj naredbe ispisa*)
- ovakvo podrazumevano ponašanje funkcije **print** može se izmeniti zadavanjem imenovanih argumenata **sep** i **end** čije podrazumevane vrednosti su, redom, ' ' i '\n'
- zadavanje imenovani argumenti **sep** i/ili **end** se pri pozivu funkcije **print** se obavezno navode na kraju, nakon svih ostalih (neimenovanih) argumenata

Biblioteka **termcolor**

- biblioteka za rad sa obojenim ispisom u terminalu
- da bi se mogle koristiti funkcije i konstante iz **termcolor** biblioteke (modula) potrebno ih je uvesti u program pomoću naredbe **import**
- biblioteka sadrži svega dve funkcije:
 - 1) **colored** - priprema (formatira) tekst za ispis u obojenoj formi; omogućava bojenje teksta i pozadine, kao i podešavanje stila teksta, kao što su podebljano, podvučeno i iskošeno
 - 2) **cprint** - ispisuje u terminal tekst u obojenoj formi; to je omotač funkcija funkcije **colored** koja formatirani tekst prosledjuje funkciji **print** da ga ispiše u terminal
- pored ove dve funkcije, biblioteka takođe definiše razne konstante koje se tiču *ANSI* kodiranja boja