

Beleške sa časa – 8.nedelja

Terminologija

- konsekutivni = uzastopni (vrednosti, elementi)
- sukcesivni = sekvencijalni, u određenom redosledu, jedan za drugim ali ne nužno uzastopni (vrednosti, elementi)
- pomeraj/korak = razlika između vrednosti dva susedna elementa
- predefinisano = unapred definisano, zadato

Kolekcijski tipovi podataka

- **int**, **float**, **complex**, **bool** i **str**¹ su tzv. *primitivni tipovi podataka* (osnovni, elementarni, jednostavni tipovi podataka) jer se njihove vrednosti (instance) ne mogu raščlaniti na komponente jednostavnijeg tipa
- preostali ugrađeni tipovi podataka (**range**, **list**, **tuple**, **set** i **dict**) su kolekcijski tipovi podataka
- tipovi podataka koji se sastoje od više objekata jednostavnijeg tipa nazivaju se *kolekcijski tipovi podataka* ili kraće *kolekcije*
- kolekcijski tipovi podataka grupišu veći broj objekata jednostavnijeg tipa
- objekti od kojih se kolekcija sastoji nazivaju se *elementi kolekcije*

Tip podataka **range**

- tip **range** podpredstavlja opseg (raspon, segment, interval) celobrojnih vrednosti sa određenim pomerajem (korakom odabiranja)
- drugim rečima, objekat tipa **range** predstavlja kolekciju sukcesivnih celobrojnih vrednosti koje se nalaze u određenim granicama i čije se susedne vrednosti podjednako razlikuju jedne od drugih
- dakle, **range** objekti zapravo definišu konačne aritmetičke nizove²
- objekti tipa **range** se definišu pomoću istoimene funkcije (tzv. *konstruktorska funkcija*)
- postoje tri varijante konstruktorske funkcije za **range** objekte, sa 1, 2 ili 3 argumenta:

range(stop)
range(start, stop)
range(start, stop, step)

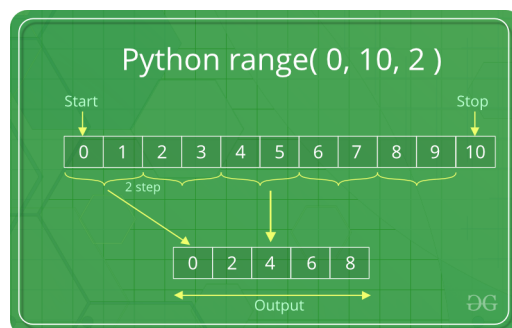
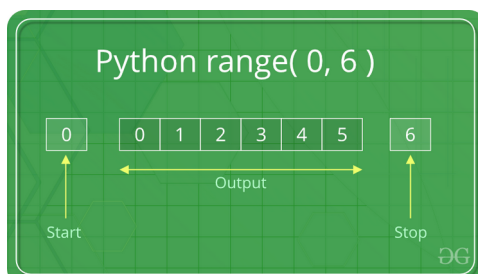
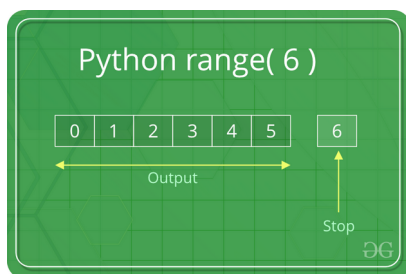
pri čemu navedeni argumenti imaju sledeće značenje:

start – početna vrednost opsega, tj. leva granica opsega (ceo broj, podrazumevano **0**)

stop – krajnja vrednost opsega³, tj. desna granica opsega (ceo broj)

step – pomeraj, korak odabiranja vrednosti za opseg (ceo broj različit od **0**, podrazumevano **1**)

- argumenti **start** i **step** imaju svoje podrazumevane vrednosti koje se koriste kod varijanti konstruktorske funkcije koje izostavljaju navođenje jednog ili oba ta parametra



- za bilo koji od argumenata **start**, **stop** i **step** mogu biti navedene i negativne celobrojne vrednosti

¹ Tip **str** se nekada smatra primitivnim a nekada kolekcijskim tipom podataka, s obzirom da ima neke karakteristike kolekcijskog tipa, u smislu da se može posmatrati kao kolekcija svojih karaktera, ali nije kolekcijski tip podataka u pravom smislu zato što ne postoji jednostavniji tip podataka kojim bi se predstavili pojedinačni karakteri stringa.

² Nizovi u kojima je razlika između bilo koja dva uzastopna člana konstantna nazivaju se aritmetički nizovi (progresije).

³ Krajnja vrednost opsega se nikada ne uključuje u opseg, tako da za argument **stop** uvek treba navesti za jedan veću vrednost od one koju želimo da bude uključena u opseg, kada je **step** pozitivan, odnosno za jedan manju vrednost ukoliko je **step** negativan.

- korak sa pozitivnom vrednošću se koristi za kreiranje sekvenci brojeva u rastućem poretku, dok se korak sa negativnom vrednošću koristi za kreiranje sekvenci brojeva u opadajućem poretku
 - pozitivna vrednost za parametar **step** znači da vrednosti koje čine opseg počinju od vrednosti **start** i da se svaka naredna vrednost *uvećava* za vrednost **step** (podrazumeva se da je zadato **start < stop**)
 - negativna vrednost za parametar **step** znači da vrednosti koje čine opseg počinju od vrednosti **start** i da se svaka naredna vrednost *umanjuje* za vrednost **step** (podrazumeva se da je zadato **start > stop**)
- nepravilno zadate granice opsega ili veličina/znak koraka rezultuje praznim opsegom, tj. kolekcijom koja ne sadrži niti jedan element
- kako opseg **range(n)** sadrži ukupno **n** vrednosti, ovako definisan opseg se često koristi za odbrojanje kada neku naredbu ili blok naredbi treba ponoviti određen broj puta, tj. **n** puta
- drugim rečima, **range** objekti se najčešće koriste za realizaciju brojačke petlje u kombinaciji sa **for** naredbom, tj. za generisanje niza vrednosti brojača koje uzima u svakoj od iteracija petlje
- objekti tipa **range** su imutabilni, tj. jednom kada se kreiraju ne može im se više menjati vrednost (sadržaj kolekcije)
- **range** objekti se interno (u memoriji računara) čuvaju u kompaktnom obliku, tako što se čuvaju samo početna i krajnja vrednost koje određuju opseg, kao i pomeraj/korak, a onda se po potrebi (prilikom pristupa pojedinačnim elementima opsega) na osnovu tih informacija sračunavaju vrednosti koje čine opseg
- iako to najčešće nije neophodno, konvertovanje (kastovanje) **range** objekta u listu može se izvršiti pomoću funkcije za kastovanje **list**
- nad objektima tipa **range** mogu se primeniti operator indeksiranja (operator **[]**), slajsing operator (operator **[:]**), kao i operatori pripadnosti (operatori **in** i **not in**)
- indeksiranje elemenata **range** objekata funkcioniše po istom principu kao indeksiranje karaktera kod stringova

Operatori pripadnosti

- u Python-u postoje dva operatora pripadnosti: **in** i **not in**
- operatori pripadnosti proveravaju pripadnost tj. nepripadnost datog elementa nekoj kolekciji
- struktura izraza sa operatorima pripadnosti:
 - elem in kolekcija**
 - elem not in kolekcija**
- dakle, to su binarni operatori gde je drugi operand kolekcija, a prvi argument je element (vrednost, objekat) koji se pretražuje u datoj kolekciji
- rezultat primene operatora pripadnosti je **True** ili **False**
- operatori pripadnosti su nižeg prioriteta u odnosu na aritmetičke i relacione operatore, a višeg prioriteta od logičkih operatora

for petlja (naredba)

- struktura for naredbe:
 - for elem in kolekcija:**
 - blok-naredbi**
- glavna komponenta **for** naredbe je kolekcija kroz koju se iterira, tj. kroz koju se u iteracijama prolazi element po element, te se otuda **for** petlja još naziva *kolekcijskom petljom*
- u svakoj iteraciji se tzv. *promenljivoj petlje* (**elem**) redom dodeljuju vrednosti elemenata kolekcije i za svaku od tih vrednosti se izvršavaju naredbe u okviru tela petlje
- dakle, **koliko iteracija (koraka) će imati for petlja zavisi od broja elemenata u kolekciji kroz koju se iterira (prolazi)**
- primetiti da ključna reč **in** kao komponenta **for** naredbe ima nešto drugačiji smisao nego istoimeni operator pripadnosti - naime, u kontekstu **for** naredbe ključna reč **in** ima smisao pridruživanja (nalik naredbe dodele) elemenata iz kolekcije sa desne strane promenljivoj koja se nalazi sa leve strane ključne reči **in**
- kada na mesto kolekcije u **for** petlji stavimo **range** objekat, dobijamo brojačku petlju, tj. brojačka **for** petlja je specijalan slučaj (opšte) kolekcijske **for** petlje
- u ovakvoj konstrukciji brojačke **for** petlje, promenljiva petlje **elem** ima ulogu brojača

- kada nam vrednosti koje uzima brojač (promenljiva petlje u opštem slučaju) u iteracijama petlje nisu od značaja, tj. ne koriste se nigde u telu petlje, tada na mesto promenljive petlje možemo staviti placeholder promenljivu `_`
- kako u Python-u ne postoji beskonačna konstanta tipa `int`⁴, a argumenti za `range` konstruktorsku funkciju moraju biti isključivo tipa `int`, nije moguće definisati beskonačnu brojačku `for` petlju
- beskonačnu `for` petlju je moguće konstruisati jedino tako što će se kolekcija kroz koju se iterira (npr. lista) u svakom koraku proširuje dodavanjem barem jednog novog elementa, čime se obezbeđuje da se nikada neće doći do kraja kolekcije, tj. da uvek postoje elementi kolekcije kroz koje se još uvek nije prošlo
- dakle, sve što može da se uradi pomoću `while` petlje postoji način da se izvede i pomoću `for` petlje, tj. moglo bi se reći da `while` i `for` petlja predstavljaju dva načina da uradimo istu stvar, s tim da je u nekada prirodnije i jednostavnije koristiti jednu, a nekada drugu vrstu petlje (u zavisnosti od konteksta problema koji se rešava prirodno se nameće koja naredba petlje je pogodnija)
- podsećanje: po prirodi svoje konstrukcije, za petlje za koje znamo unapred koliko puta će se izvršiti (brojačke petlje i petlje koje prolaze kroz neku kolekciju), uobičajeno se koristi `for` naredba, dok se za petlje gde ne znamo unapred koliko će se puta ponoviti (petlje sa uslovom) uobičajeno koristi `while` naredba, iako se pomoću i jedne i druge naredbe mogu realizovati obe vrste petlji

⁴ podsećanje - postoje beskonačne konstante tipa `float`, a to su `float('±inf')` i `math.inf`