

## Beleške sa časa – 10.nedelja

### Terminologija

- ulaz programa = svi podaci i/ili akcije (npr. klik na dugme) koje korisnik zadaje tokom izvršavanja programa
- kraj ulaza programa = kada nema više podataka za čitanje

### Indikatorska promenljiva

- to je obična promenljiva koja svojom vrednošću indikuje kakvo je tekuće stanje izvršavanja programa u smislu da li se neki događaj od značaja (za dalji tok izvršavanja programa) dogodio ili ne tokom izvršavanja nekog dela programa (najčešće tokom izvršavanja petlje)
- **indikatorska promenljiva se koristi onda kada tok izvršavanja programa zavisi od vrednosti ulaznih podataka**
- dakle, osnovna karakteristika indikatorske promenljive jeste da je to *binarna promenljiva* koja indikuje realizaciju nekog relevantnog događaja tokom obrade ulaznih podataka programa, tj. prisustvo ili odsustvo neke karakteristike u ulaznim podacima, od čega zavisi tok izvršavanja ostatka programa
- indikatorska promenljiva može uzimati bilo koje vrednosti bilo kog tipa, kao što su npr. **True** ili **False**, **0** ili **1**, **“jeste”** ili **“nije”** i slično, ali tako da u okviru jednog programa to bude jednoznačno, tj. da u toku izvršavanja tog programa promenljiva može da ima samo jednu od dve unapred određene vrednosti istog tipa, kako bi se na osnovu vrednosti indikatorske promenljive pravilno moglo zaključivati o daljem toku izvršavanja programa
- ne postoji pravilo (i često zavisi od konteksta) koju od dve predviđene vrednosti treba pridružiti indikatorskoj promenljivoj kao inicijalnu, ali se uobičajeno za inicijalnu vrednost uzima vrednost sa odričnom interpretacijom (**False**, **0**, **“nije”** i sl.) kako bi se time predstavilo inicijalno stanje programa – da se događaj koji želimo da indikujemo pomoću te promenljive (još uvek) nije desio, tj. da (još uvek) nismo pronašli karakteristiku ulaznih podataka koju želimo da indikujemo pomoću te promenljive
- na kraju se proverava da li je indikatorska promenljiva zadržala inicijalnu vrednost ili ne, i na osnovu toga se odlučuje o daljem toku izvršavanja programa

### Kolekcijska **for** petlja i stringovi

- 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
- iako nisu kolekcije u pravom smislu, stringovi se mogu koristiti u kombinaciji sa kolekcijskom **for** petljom za iteriranje po karakterima stringa (u obliku jednokaracterskih stringova)

### Operator **in** i stringovi

- pomoću operatora **in** (operator pripadnosti elementa kolekciji) primenjenog nad stringovima može se proveriti da li jedan string sadrži drugi string kao podstring
- ovo znači da se stringovi u kombinaciji sa operatorom **in** ne ponašaju samo kao kolekcija svojih karaktera, tj. svojih jednokaracterskih podstringova, već kao kolekcija svih svojih podstringova (podstringova svih dužina)

### EOF

- programu se ulazni podaci mogu zadavati putem terminala ili čitati iz tekstualnog fajla, a moguće je i kombinovanje ta dva pristupa u okviru istog programa (deo podataka se unosi preko terminala a deo se čita iz tekstualnog fajla)
- kada se ulazni podaci čitaju iz fajla i prilikom čitanja se dođe do kraja fajla, automatski se generiše tzv. *EOF signal* (eng. *End Of File*) koji se dalje prosleđuje funkciji koja vrši čitanje fajla kao indikator kraja ulaza
- kada se ulazni podaci čitaju sa standardnog ulaza, tj. iz terminala, tada se EOF signal generiše zadavanjem specijalne kontrolne sekvence koja se razlikuje kod različitih operativnih sistema:
  - Unix i Mac operativni sistemi – Ctrl + D (istovremeno)
  - Windows operativni sistemi – Ctrl + Z (istovremeno) pa ENTER

- dakle, EOF je univerzalni indikator (signal, marker) za kraj ulaza programa
- kada funkcija koja vrši čitanje ulaza programa (npr. **input** funkcija) prilikom svog izvršavanja dobije EOF signal, ona generiše **EOFError** grešku koju je potrebno obraditi na odgovarajući način pomoću **try-except** bloka

### Biblioteka **random**

- biblioteka za rad sa generatorima pseudoslučajnih brojeva
- pod slučajnim (nasumičnim) odabirom brojeva iz nekog skupa (opsega) misli se na uzorkovanje brojeva zasnovano na teoriji verovatnoće, gde je svakom broju pridružena određena verovatnoća da bude odabran
- generatori pseudoslučajnih brojeva se koriste za simulaciju slučajnog odabira brojeva u programima<sup>1</sup>, pri čemu se termin “pseudoslučajni” odnosi na to da ovako generisani (odabrani) brojevi nisu zaista slučajni, već su dobijeni specifičnim postupkom (izračunavanjem) koji imitiraju slučajan odabir
- da bi se mogle koristiti funkcije iz **random** biblioteke (modula) potrebno ih je uvesti u program pomoću naredbe **import**
- spisak svih funkcija definisanih u biblioteci **random** možete naći na sledećem linku:  
[https://www.w3schools.com/python/module\\_random.asp](https://www.w3schools.com/python/module_random.asp)
- neke od najčešće korišćenih funkcija **random** biblioteke su:
  - **random.random()** – pseudoslučajan realan broj iz opsega [0,1)
  - **random.randint(a, b)** – pseudoslučajan ceo broj iz intervala [a, b]
  - **random.randrange(stop)** – pseudoslučajan element iz **range(stop)** tj. ceo broj iz intervala [0, stop)
  - **random.randrange(start, stop, step=1)** – pseudoslučajan element iz **range(start, stop, step)** tj. ceo broj iz intervala [start, stop) sa korakom odabira **step** (podrazumevano 1)
  - **random.uniform(a, b)** – pseudoslučajan realan broj iz uniformne raspodele  $U[a, b]$
  - **random.gauss(mi, sigma)** – pseudoslučajan realan broj iz normalne(Gausove) raspodele  $N(mi, sigma)$
  - **random.shuffle(kolekcija)** – mešanje, tj. promena redosleda elemenata kolekcije na pseudoslučajan način
  - **random.choice(kolekcija)** – pseudoslučajno odabran jedan element iz kolekcije
  - **random.sample(kolekcija, n)** – pseudoslučajno odabranih **n** elemenata iz kolekcije
  - **random.seed(x)** – postavljanje inicijalnog stanja (vrednosti) za generator pseudoslučajnih brojeva

---

<sup>1</sup> U računarstvu nije moguće ostvariti potpunu slučajnost, s obzirom da se sva izračunavanja vrše na egzaktn način, pa se zato pribegava rešenjima koja imitiraju (simuliraju) slučajnost.