Liste

Igor Buzov, dipl. inf., viši predavač
Centar umjetne inteligencije Lipik

Pregled lekcije

- Cilj ove lekcije je upoznati se sa osnovom podatkovnom strukturom u Pythonu - listama i objasniti čemu služe
- Također, pokazat će se na koje sve načine se liste mogu modificirati (dodavati elemente, micati elemente, brisati elemente)
- Na kraju ćete se upoznati s jednim od automatskih procesa upravljanja memorijom u Pythonu – garbage collectorom

Ponavljanje – što će biti rezultat ovog koda?

```
def pozdrav():
print ("Dobro došli!")

pozdrav()
```

- Ako postoji greška, kako biste je popravili?

Ponavljanje – što će biti rezultat ovog koda?

```
def potpis():
    print ("Trgovac d.o.o")
    print ("Ilica 1")
    print ("tel: 01/ 1234 567")
    print ("OIB: 514824578")

print ("Srdačan pozdrav!")
print ("Perić Pero, voditelj nabave")

potpis()
```

Ponavljanje – koja je razlika između ova dva koda?

```
def zanimanje (naziv):
    print ("Vi ste po zanimanju", naziv)

zanimanje ("kuhar")
```

```
def zanimanje (naziv):
    print ("Vi ste po zanimanju", naziv)

moje_zanimanje = "kuhar"
zanimanje (moje_zanimanje)
```

Ponavljanje – što će biti rezultat ovog koda?

```
def proizvod (naziv, cijena):
    print ("Naziv proizvoda:", naziv)
    print ("Cijena proizvoda:" , cijena)

proizvod ("Ljetne gume", 800)
```

Ponavljanje – što će biti rezultat ovog koda?

```
def obracun(quantity, money = 10):
    return quantity * money

br_sati = int(input ("Koliko ste sati radili ovaj mjesec?") )
ukupno = obracun (br_sati)

print ("Ovaj mjesec ste zaradili", ukupno)
```

Zašto koristimo liste?

- Dosad smo se susreli s varijablama u koju smo pohranjivali jednu vrijednost
- Primjeri: ime, grad, marka_vozila, cijena, visina itd.
- Što napraviti ako trebamo pohraniti više podataka, tipa imena učenika, gradovi u RH, cijene proizvoda u trgovini, popis za kupnju?
- Kreiranje više varijabli kako bi pohranili svaki podatak je neučinkovito (tipa grad1, grad2, grad3 … grad250)
- U Pythonu za rješavanje ovog problema koristimo liste

Primjena listi

- Skladištenje više vrijednosti pod jednim imenom
  - Umjesto da za svaki broj, riječ ili drugi podatak imamo posebnu varijablu, možemo sve to pohraniti u jednu listu
- Organiziranje podataka
  - Kad su podaci organizirani u jednom objektu, lakše je pratiti sve podatke
- Iteriranje kroz podatke
  - Python omogućuje izvršavanje jedne naredbe na više podataka (popularno zvane petlje) – podaci moraju biti organizirani u listi
- Liste su temelj za stvaranje drugih složenijih struktura kao što su tablice, matrice i nizovi

Što je lista

Liste se koriste za pohranu više elemenata u jedan objekt

Liste su jedan od 4 predefinirana tipa podataka u Pythonu koji se koriste za pohranu kolekcija podataka

Ostala 3 tipa podataka su tuplei, setovi i dictionary, svaki sa svojom primjenom

Elementi liste su poredani po redu, mogu se mijenjati te su dozvoljeni duplikati

Kako napraviti listu

- Sintaksa

ime_liste = ["element1", "element2", "element3", ...]

- Kada želimo napraviti listu, potrebno je:
  - Navesti njeno ime, malim slovima, opisno ime, ne koristiti rezervirane riječi
  - Nakon toga dolazi znak =
  - Elementi liste se sa svake strane omeđuju uglatim zagradama [ ]
  - Elementi liste unutar uglatih zagrada se odvajaju zarezima
  - Elementi liste koji su stringovi su omeđeni navodnicima, brojevi nemaju navodnike

Primjeri i zadaci

- Otvorite novi file pod nazivom liste.py
- Nekoliko primjera dobro napravljenih listi:

```
gradovi = ["Zagreb", "Split", "Osijek", "Pula", "Rijeka"]
cijene = [5, 9, 12, 15.5, -7.3]
odgovori = [5, 3, "dobro", 1, "dobro", "odlično"]
```

- **Zadaci:**
- Napravite listu u kojoj ćete navesti barem 5 imena vašim omiljenih sportaša
- Napravite listu u kojoj ćete navesti nekoliko modela vaših omiljenih auta te njihovu cijenu
- Napravite listu u kojoj ćete navesti ocjene iz testa za barem 5 učenika

Ispisivanje liste

- Ako zatražite od Pythona da vam ispiše sadržaj liste, on će je ispisati, uključujući i uglate zagrade

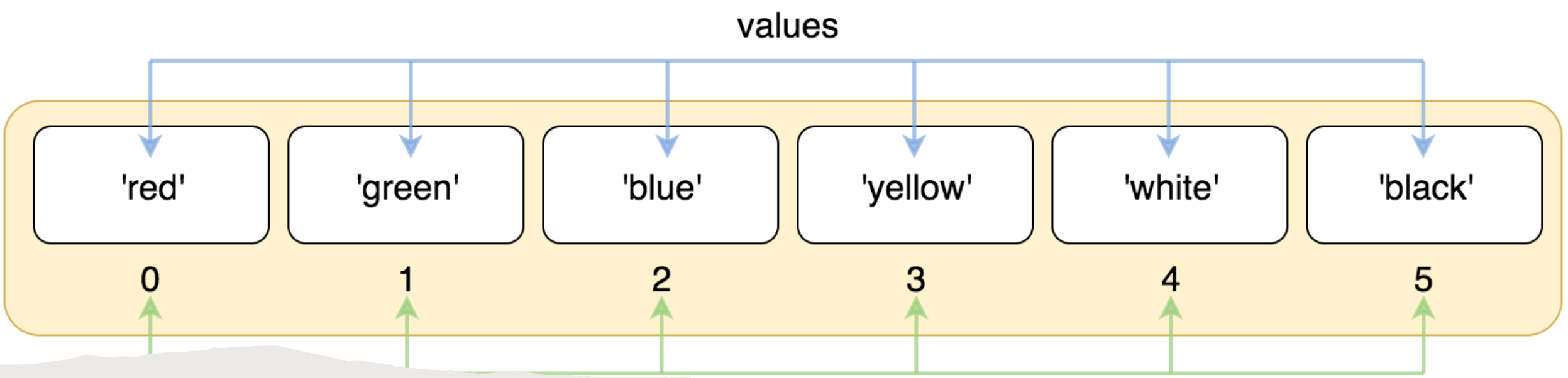
```
gradovi = ["Zagreb", "Split", "Osijek", "Pula", "Rijeka"]

print (gradovi)
```

- Rezultat:

```
['Zagreb', 'Split', 'Osijek', 'Pula', 'Rijeka']
```

- Što ako želimo ispisati (ili raditi) samo s pojedinim elementom liste (recimo Osijek)?



Pristupanje elementima liste (indeksiranje)

- Liste su uređeni nizovi podataka, redoslijed elemenata u listi je važan
- Ovo nisu dvije iste liste:

['Zagreb', 'Split', 'Osijek', 'Pula', 'Rijeka']

['Split', 'Zagreb', 'Osijek', 'Pula', 'Rijeka']

- Liste imaju svoje indekse (redne brojeve), uz pomoć kojih im pristupamo

Pristupanje elementima liste (indeksiranje)

- Elementu liste se pritupa na način da se Pythonu kaže indeks elementa koji nam treba
- Python smatra da se prvi lemement liste nalazi na poziciji 0, a ne 1
- Ako želimo element liste koji se nalazi na četvrtoj poziciji, tražit će se od Pythona element s indeksom 3
- Primjeri:

```
gradovi = ["Zagreb", "Split", "Osijek", "Pula", "Rijeka"]
print (gradovi[0])
print (gradovi[3])
```

Pristupanje elementima liste (indeksiranje)

- Osim što se elementima liste može doći s jedne strane, može se pristupiti i sa stražnje strane
- Ovaj način je zgodan ako je duljina liste nije poznata ili želimo pristupiti stražnjim elementima

```
gradovi = ["Zagreb", "Split", "Osijek", "Pula", "Rijeka"]

print (gradovi[-1])

print (gradovi[-3])
```

- Gledajući sa stražnje strane, indeksi počinju od broja -1

Pristupanje elementima liste (indeksiranje)

- Osim manipulacije jednog elementa liste, moguće je naravno raditi istovremeno i s više elemenata liste
- Primjeri:

```
gradovi = ["Zagreb", "Split", "Osijek", "Pula", "Rijeka"]
print (gradovi[1], gradovi [3])
print (gradovi[-1], gradovi[-2])
```

- Prvi primjer: Ispisat će se elementi liste s indeksom 1 te s indeksom 3 (gradovi Split i Pula)

Pristupanje elementima liste (indeksiranje)

- Ako želimo raditi s više elemenata liste, koji su u određenom rasponu, možemo koristiti znak ":" (dvotočka)
  - Ako koristimo dvotočku u radu s listama, stručni izraz glasi "slice notation"

```
gradovi = ["Zagreb", "Split", "Osijek", "Pula", "Rijeka"]
print(gradovi[0:3])
```

- Napomena: pri korištenju znaka ":", prvi indeks označava element koji je uključiv, drugi indeks označava element nakon posljednjeg elementa kojeg želimo uključiti

- Ispisat će se elementi s indeksom 0, 1 i 2 (Zagreb, Split, Osijek), element s indeksom 3 se ne ispisuje

Zadatak 1

- Radite rpg igricu u kojoj želite popis stvari koje nosi vaš heroj pohraniti u listu zvanu inventory
- Napravite listu inventory s barem 7 stvari koje nosi vaš heroj
- Ispišite sadržaj čitavog inventory-a
- Ispišite prva četiri elementa vašeg inventory-a
- Ispišite zadnja dva elementa vašeg inventory-a

Dodavanje elemenata u listu

- Većina lista koje se kreiraju su dinamičke – elementi se dodavaju i brišu po potrebi
- Na primjer:
  - Dodavanje novog registriranog korisnika na web site
  - Košarica u web shopu u koju korisnik dodaje nove stvari (ili ih briše)
  - Popis zadataka koje trebate obaviti taj dan
  - Popis putnika koji idu na organizirani izlet
  - Sadržaj inventory-a vašeg heroja u rpg igrici
- Postoji više načina za dodavanje elementa u listu

Dodavanje elementa u listu - append

- Prvi način odavanja elemenata u listu je korištenjem append metode
- Element se dodaje na kraj liste, jedan po jedan
- Na ovaj način se ne utječe na ostale elemente liste

```
gradovi = ["Zagreb", "Split", "Osijek", "Pula", "Rijeka"]
gradovi.append ("Karlovac")
gradovi.append ("Sisak")
print (gradovi)
```

Dodavanje elementa u listu - append

- Append metoda omogućuje jednostavno kreiranje dinamičkih listi
- Kreiramo na početku praznu listu i dodajemo jedan po jedan element

```
popis_kupovina = []
popis_kupovina.append ("kruh")
popis_kupovina.append ("novine")
popis_kupovina.append ("kava")
print (popis_kupovina)
```

- Dobili smo listu koja nam služi kao popis za kupovinu, s tri elementa

Zadatak 2

- Kreirajte praznu listu pod nazivom "slatkisi"
- Korištenjem odgovarajuće metode, ubacite unutra nekoliko elemenata
- Ispišite sadržaj liste slatkisi

Zadatak 3

- Kreirajte praznu listu pod nazivom "destinacije"
- Od korisnika zatražite da unese prvo mjesto koje bi htio posjetiti na ljetovanju i pohranite vrijednost
- Od korisnika zatražite da unese drugo mjesto koje bi htio posjetiti i pohranite vrijednost
- Unešene vrijednosti dodajte u vašu listu destinacije
- Ispišite poruku tipa:
- "Tokom vašeg odmora posjetit ćete ova mjesta: Brač i Delnice"

Ubacivanje elementa u listu - insert

- Korištenjem insert metode moguće je ubaciti novi element u listu na bilo koje željeno mjesto
- To postižemo na način da specificiramo indeks novog elementa kao i njegovu vrijednost

```
gradovi = ["Zagreb", "Split", "Osijek", "Pula", "Rijeka"]

gradovi.insert (1, "Zadar")

print (gradovi)
```

- Ubacili smo novu vrijednost na mjesto s indeksom 1 i ostale elemente pomakli udesno

Brisanje elemenata iz liste - remove

- Korištenjem remove metode moguće je ukloniti element liste
- Vrijednost se miče na osnovi vrijednosti, a ne indeksa

```
alieni = ["patuljak", "vilenjak", "predator"]
alieni.remove("vilenjak")
print (alieni)
```

- Dobili smo novu listu iz koje smo maknuli vilenjaka
- Vježba: iz vaše postojeće liste slatkisi, obrišite jedan element

Brisanje elemenata iz liste - pop

- Korištenjem metode pop moguće je obrisati element liste ovisno o njegovom indeksu (poziciji)

```
gradovi = ["Zagreb", "Split", "Osijek", "Pula", "Rijeka"]

gradovi.pop(2)

print (gradovi)
```

- U slučaju da se ne navede indeks, pop metoda miče zadnji element liste

```
gradovi.pop()
```

Mijenjanje sadržaja liste

- Nakon što je lista kreirana, moguće je mijenjati i sadržaj pojedinog elementa

```
narudzba = ["pljeskavica", "pohana piletina", "pomfrit"]
narudzba[1] = "Coca cola"
print (narudzba)
```

- Element liste "pohana piletina" je zamijenjen s "Coca Colom"

Zadatak 4

- Napravite listu koju ćete nazvati shopping_cart i koja ima tri odjevna predmeta

shopping_cart = ["hlače", "kaput", "remen"]

- Kupac bi htio u svoju košaricu dodati još i zimsku kapu te rukavice. Ubacite te lemente u listu
- Kupac se predomislio i ne želi uzeti remen, maknite ga iz liste
- Kupac također želi promijeniti kaput i umjesto njega staviti zimsku jaknu
- Za kraj ispišite listu korištenjem naredbe print

Spajanje listi korištenjem operatora +

- Ako želimo spojiti dvije liste, možemo to napraviti na više načina
- Jedan od njih je operator + (konkatenacija)
- Pri korištenju operatora +, obavezno mora nastati nova lista

```
gradovi_sjever = ["Zagreb", "Karlovac", "Sisak"]
gradovi_jug = ["Split", "Zadar", "Rijeka"]
hrvatska = gradovi_sjever + gradovi_jug
print (hrvatska)
```

Brisanje sadržaja čitave liste - clear

- Ako ne želite micati jedan po jedan element iz liste, nego ih želite sve obrisati, koristi se metoda clear()
- Kao rezultat dobije se prazna lista

```
biljke = ["hrast", "jagoda", "banana", "kaktus"]

print(biljke)

biljke.clear()

print(biljke)
```

Zadatak 5

- Kreirajte listu "prvi_razred" u kojoj se nalaze 3 imena đaka te je ispišite
- Đaci su nakon godine dana završili prvi razred, obrišite sadržaj liste "prvi razred", neka ostane prazna te ispišite sadržaj liste

Brisanje više elemenata liste odjedanput - del

- Ako želite obrisati više elemenata liste odjedanput, umjesto da se to radi sa svakim elementom pojedinačnom, možemo koristiti slice notation i statement del

```
cijene = [5, 10, 27, 43, 11, 4, 39, 51]

del cijene[2:5]

print (cijene)
```

- Korištenjem slice notationa brišemo sve elemente koji počinju s indeksom 2 te završavaju s indeksom 4

Brisanje čitave liste - del

- Ako ne trebamo više listu i ne želimo da nam zauzima memoriju, korištenjem satementa del je možemo potpuno obrisati

```
cijene = [5, 10, 27, 43, 11, 4, 39, 51]

del cijene

# lista cijene više ne postoji
```

- Uz pomoć del statementa mogu se brisati i drugi objekti, tipa varjabli, ne samo liste

Važnost brisanja nepotrebnih objekata

- Zašto brisati objekte?
  - **Oslobađanje memorije**: Svaki objekt koji stvorimo u Pythonu zauzima određenu količinu memorije. Ako objekte ne brišemo kada nam više ne trebaju, memorija se s vremenom može popuniti, što može dovesti do usporavanja programa ili čak do njegovog rušenja (pogotovo kod velikih programa ili obrade velikih količina podataka)
  - **Bolja organizacija koda**: Brisanje nepotrebnih objekata čini kod čitljivijim i lakšim za održavanje. Jasno je koje varijable se koriste i kada se više ne koriste

Sakupljač smeća (garbage collector)

Sakupljač smeća je automatski proces upravljanja memorijom u Pythonu. Njegov zadatak je pronaći i osloboditi memoriju koju zauzimaju objekti koji se više ne koriste u programu

U većini slučajeva, ne morate se brinuti o ručnom brisanju objekata jer Pythonov sakupljač smeća radi dobar posao. Ipak, razumijevanje del statementa i upravljanja memorijom je važno za pisanje efikasnijih programa, naročito radi bolje organizacije i preglednosti koda

Zadatak 6

- Kreirajte novu listu koju ćete nazvati vrucina. U njoj se nalaze dvije vrijednosti, maksimalna temperatura za jedan dan te minimalna temperatura za jedan dan
- Kreirajte varijablu razlika u kojoj ćete izračunati koja je razlika između te dvije vrijednosti te je ispišite

Sortiranje liste - sort

Vrlo često ne možemo kontrolirati kako nastaje lista i kojim redoslijedom se elementi zapisuju u listu

Vrlo često se želi podatke prezentirati u specifičnom redoslijedu, sortirano abecedno (ili unazad)

Primjeri: košarica na web shopu, inventory u igrici, popis polaznika edukacije

Sortiranje liste – sort

- Napravit ćemo listu s popisom stvari u košarici na webu
- Korisnik je ubacivao stvari u svoju košaricu nekim slučajnim redoslijedom
- Želio bi vidjeti popis svojih stvari abecednim redom
- Koristimo metodu sort. Metoda sort trajno mijenja raspored elemenata u listi
- Elementi u listi se mogu sortirati i obrnutim redoslijedom, abecedno unazad – koristi se parametar *reverse*

Sortiranje liste - primjer

```
kosarica = ["sol", "papar", "kim", "anis", "vlasac"]
print (kosarica)

kosarica.sort()
print (kosarica)
```

Sortiranje liste – primjer (reverse)

```
visina_sportasa = [157, 192, 148, 179, 201]
print (visina_sportasa)

visina_sportasa.sort(reverse = True)
print (visina_sportasa)
```

Proučite i objasnite kod (primjer RPG igrice)

```
def dodaj_u_inventory (inventory, predmet):
        inventory.append(predmet)
        print (predmet + " je dodan u inventory")

inventory = ["mač", "štit", "koplje"]

dodaj_u_inventory (inventory, "sjekira")
```

Zadatak 7

- Kreirajte vlastitu funkciju uz pomoć koje biste iz svoje liste za inventory mogli maknuti željeni predmet
- Ispišite sadržaj inventorya na kraju!

Proučite i objasnite kod (primjer RPG igrice)

```
def zamijeni_u_inventory(inventory, stari_predmet, novi_predmet):
        index = inventory.index(stari_predmet)
        inventory[index] = novi_predmet
        print (stari_predmet + " je zamijenjen u inventoryu s " + novi_predmet)

inventory = ["mač", "štit", "koplje"]

zamijeni_u_inventory (inventory, "mač", "čarobnjakov mač")
```

- Čemu služi metoda index?

Sažetak lekcije