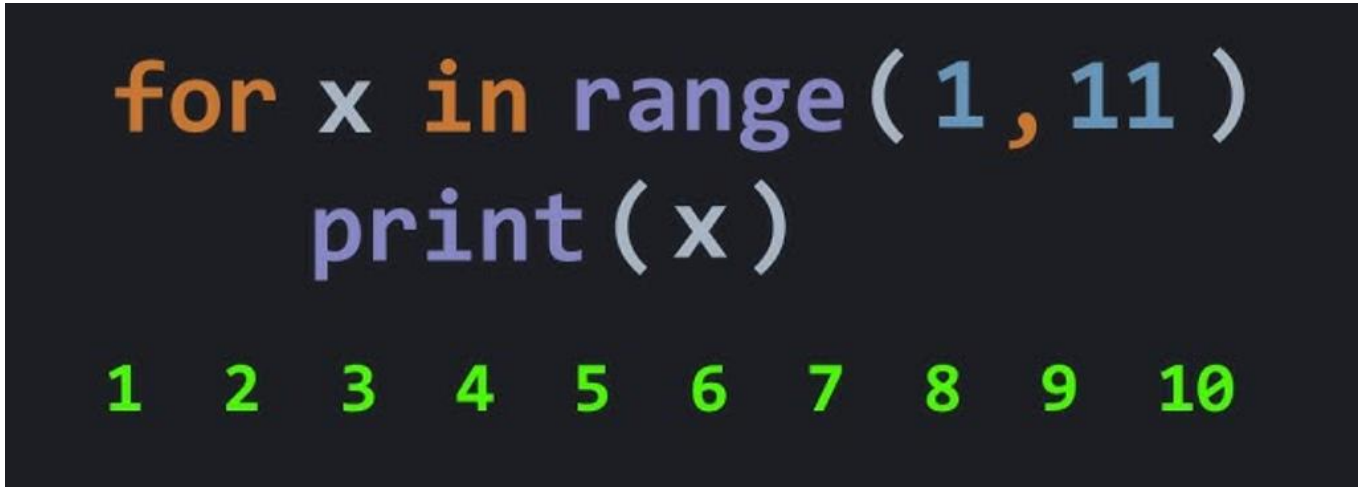



# For petlje

Igor Buzov, dipl. inf., viši predavač

Centar umjetne inteligencije Lipik

# Pregled lekcije

- Cilj ovog poglavlja je upoznati se s novim konceptom, iteracijama
- Za iteracije koristimo for petlju
- Upoznat ćemo se i s kontrolnim izjavama kao što su break, continue i else koje nam omogućuju fleksibilnost u for petljama
- Pokazat ćemo kako se može koristiti range() funkcija u kombinaciji s for petljama

# Ponavljanje - što će biti rezultat ovog koda?

```
cijena = 100

if cijena > 50:
    print ("Jako skupo")
else:
    print ("povoljno")
```

```
cijena = 75

if cijena >= 100:
    print ("Jako skupo")
elif cijena >= 75:
    print ("Povoljno")
else:
    print ("Jeftino")
```

# Ponavljanje – popravi kod!

```
cijena = 75

if cijena >= 100
print ("Jako skupo")
else:
print ("Jeftino")
```

```
auto = "BMW"

if auto = "BMW":
    print ("Voliš prave aute!")
else:
    print ("Jesi razmišljao o kupovini pravog auta?")
```

# Ponavljanje - što će biti rezultat ovog koda?

```
gradovi = ["Milano", "Rim", "Venecija", "Napulj"]

destinacija ="RIM"

if destinacija in gradovi:
    print ("Italija je predivna!")
else:
    print ("Možda želite ljetovanje negdje drugdje!")
```

# Ponavljanje - što će biti rezultat ovog koda?

```
broj = 25

if broj > 10:
    print("Broj je veći od 10.")
elif broj > 20:
    print("Broj je veći od 20.")
else:
    print("Broj je 10 ili manji.")
```

# Otvorite novi file po nazivom for_petlje.py

# For petlje - uvod

- For petlja se koristi za iteraciju kroz elemente u nekom nizu (sekvenci), kao što su liste, rasponi brojeva, tuplei itd.
- Omogućuje izvođenje bloka koda više puta, po jednom , za svaki element u sekvenci

```
voce = ["jabuka", "banana", "višnja", "lubenica", "kruška", "naranča"]
print (voce[0])
```

- Ako biste htjeli ispisati svaki element liste zasebno, u ovom slučaju bi vam trebalo 6 naredbi print
- Što ako imate zapise o 200 zaposlenika neke tvrtke? Odjedanput pisanje printa nije više jednostavno ni lagano – uvodimo for petlju

# For petlje - uvod

- Primjena for petlji:
- Kada želimo izvršiti istu operaciju na svakom elementu liste
  - Game development – želimo pomaknuti svaki element određene skupine za isti broj polja
  - Bankarstvo – želimo dodati kamatu na štednju za sve štediše koji su oročili štednju na isti rok
  - Data science – želimo izvršiti istu statističku operaciju na svakom elementu liste
  - Ili, kao što smo već rekli, želimo samo ispisati sadržaj određene liste (sekvence)
- Korištenje for petlji je važno jer je to jedan od najčešćih načina kako računala rješavaju repetitivne zadatke

# For petlje - uvod

```
voce = ["jabuka", "banana", "višnja", "jabuka", "kruška", "naranča"]

for vocka in voce:

    print (vocka)
```

- Za svaki element (ovdje ga zovemo vocka) liste voce, napravi neku akciju (ovdje printamo)

- voce (lista): Ovo je sekvenca kroz koju iteriramo. Može biti lista, tuple, string ili bilo koji iterabilni objekt.
- vocka (privremena varijabla): Ovo je varijabla koja uzima vrijednost svakog elementa u sekvenci, jedan po jedan, tijekom svake iteracije.
- print(vocka): Ovo je blok koda koji se izvršava za svaki element u sekvenci. U ovom slučaju, ispisuje se svaki element liste.

# Primjer 1

```
carobnjaci = ["merlin", "gandalf", "dumbledore"]


for carobnjak in carobnjaci:
    print (carobnjak.title(), ", to je bio super trik!")
```

- Kada kreirate svoje vlastite for petlje, uzmite u obzir da za naziv privremene varijable možete uzeti bilo koji izraz (ovdje je to carobnjak)
- Preporuča se za naziv privremene varijable uzeti neko smisleno ime

# Imenovanje privremene varijable – naming conventions

- Primjeri dobre prakse

```
for cat in cats:
    print (cat)
for dog in dogs:
    print (dog)
for item in list_of_items:
    print (item)
```

# Zadatak 1

- Kreirajte listu automobili i stavite unutra nekoliko članova liste
- Korištenjem for petlje ispišite sadržaj liste
- U ispisu, svi automobili koje ispišete trebaju bit napisani malim slovima (ne želimo BMW, Ford, šKODA i sl.)

# Imenovanje privremene varijable – naming conventions

- Vrlo često ćete naići na sljedeće primjere:

```
for i in lista:
    print (i)
```

- Korištenje i kao varijable u for petljama dolazi iz matematičke tradicije, gdje se i često koristi kao indeks ili brojač. To je postalo uobičajeno u programiranju, posebno u kontekstu petlji
- Kada varijabla predstavlja nešto konkretno, koristimo opisno ime, inače se koristi "i"

# Primjer 2

```
carobnjaci = ["merlin", "gandalf", "dumbledore"]

for carobnjak in carobnjaci:
    print (carobnjak.title(), ", to je bio super trik!")

print ("Hvala svima na ugodnoj zabavi, vidimo se na sljedećem nastupu")
```

# Zadatak 2

- Kreirajte listu:

rijeci = ["dobro", "jutro", "svima"]

- Napisati program koji uzima listu riječi i ispisuje svaku riječ s dodanim uskličnikom na kraju.

# Zadatak 3

- Kreirajte neki string i pohranite ga u varijablu
- Korištenjem for petlje, ispišite svaki znak tog stringa

# Primjer 3

- Što će biti rezultat ovog koda?

```
for i in [0, 1, 2, 3]:
    print ("Wuf, wuf, mijau, mijau!")
```

- Kako modificirati kod ako bismo htjeli ispis 100 puta? A 10 000 puta?

- Naravno da postoji i ljepši način!

# Funkcija range()

```
for i in range (4):
    print ("Wuf, wuf, miaju, mijau!")
```

- Funkcija range() vraća niz brojeva, počevši od 0 po defaultu, i povećava se za 1 (po defaultu) te se zaustavlja prije određenog broja
  - U našem primjeru kreira niz brojeva koji počinju od 0 i zaustavljaju se na 3 što je ukupno 4
  - Kao rezultat ispisuje 4 puta tekst
- Ako bismo htjeli ispisati 500 puta tekst, samo upišemo broj 500 unutar zagrada funkcije range

# Funkcija range() - sintaksa

## Range (start, stop, step)

- Start – opcionalno, integer koji nam govori na kojem broju se počinje, ako se izostavim kreće se od nule
- Stop – obavezno, integer koji nam govori na kojem broju ste staje (isključivo taj broj)
- Step – opcionalno, iteger koji nam gvori za koliko se povećava, ako se izostavi, koristi se 1

# Primjeri

```
x = range(3, 6)
for i in x:
  print(i)
```

- Kreiraj niz brojeva od 3 do 5 i ispiši svaki od njih

```
x = range(3, 20, 2)
for i in x:
  print(i)
```

- Kreiraj niz brojeva od 3 do 19 i ispiši ih, svaki drugi

# Korištenje range() funkcije

- Uz pomoć range() funkcije može se kreirati niz brojeva koji se odmah može pohraniti u listu

```
numbers = list(range(1, 6))
print(numbers)
print (type(numbers))
```

- Ako se koristi treći parameter u funkciji range(), može se napraviti lista u kojoj se određeni brojevi preskaču

```
even_numbers = list(range(2, 11, 2))
print(even_numbers)
print (type(even_numbers))
```

# Primjer 4

```
squares = []
for value in range(1,11):
    squares.append(value**2)
print(squares)
```

```
brojevi = list(range(1,6))
suma = 0
for broj in brojevi:
  suma += broj
print("Suma brojeva je: ", suma)
```

- Varijabla suma je counter, koristimo je kao pomoćnu varijablu i na početku je inicijaliziramo na nulu

# Zadatak 4

- Kreirajte samostalno, bez korištenja range funkcije, listu s barem 5 cijelih brojeva
- Napišite kod s kojim biste prošli kroz listu i provjerili je li broj u listi paran ili ne
- Ako je paran, ispisuje se: "Paran broj!"
- Ako nije paran broj, ispisuje se: "Nije paran broj!"

# Zadatak 5

Kreirajte listu pod nazivom ocjene i stavite u nju nekoliko ocjena (od 1 do 5)

Prođite kroz vašu listu for petljom i prebrojite koliko ima negativnih ocjena

HINT: na početku definirajte neku proizvoljnu varijablu u koju ćete pohranivati broj negativnih ocjena (counter)

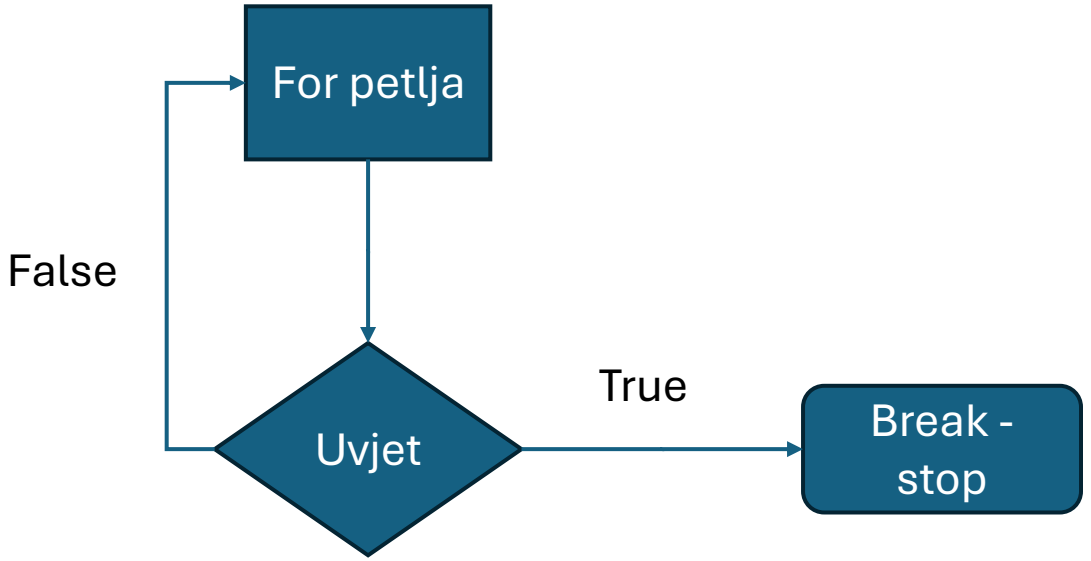
Ispišite na kraju koliko ima negativnih ocjena

# Zadatak 6

- Zamolite korisnika da unese neku rečenicu
- Vaš zadatak je proći for petljom kroz tu rečenicu i prebrojiti koliko ima samoglasnika
- HINT: Python je case sensitive pa su samoglasnici i slova "aeiou" kao i "AEIOU"
- Na kraju ispišite tekst:
  - Ukupan broj samoglasnika iznosi: BROJ SAMOGLASNIKA

# Kontrolne izjave u for petljama - break

- Naredba break se koristi za trenutno prekidanje petlje u kojoj se nalazi.
- Kada se break izvrši, tok programa se odmah prebacuje na prvu naredbu koja slijedi nakon petlje
- Svrha ove naredbe je učiniti for petlju efikasnijom te implementirati složenije logike iteracije
- Primjena: dostupnost proizvoda, točan login itd.



# Primjer 4

```
brojevi = [1, 2, 3, 4, 5]
trazeni_broj = 4
pronadjen = False

for broj in brojevi:
    print("Provjeravam broj: ", broj)
    if broj == trazeni_broj:
        print("Pronašao sam broj", trazeni_broj)
        pronadjen = True
        break
else: #Pazi, ovaj else nije dio if petlje, nije uvučen, o tome poslije
    print("Broj", trazeni_broj, "nije pronađen u listi.")
```

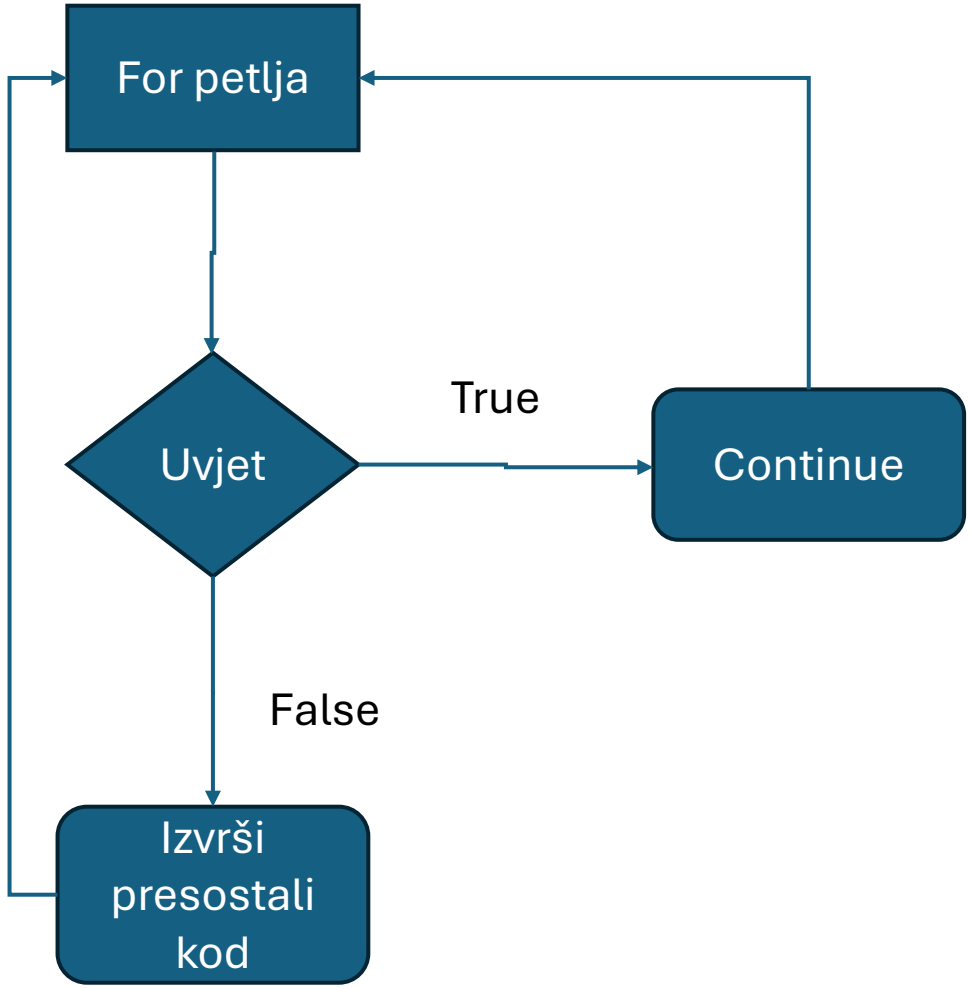
# Zadatak 7

- Napišite program koji prolazi kroz listu cijelih brojeva.
- Čim naiđe na prvi parni broj, program treba ispisati taj broj i prekinuti daljnje pretraživanje liste.
- Ako u listi nema parnih brojeva, program ispisuje prigodnu poruku
- Naša lista:

brojevi = [1, 3, 5, 2, 7, 9, 4, 6]

# Kontrolne izjave u for petljama - continue

- Naredba continue se koristi s for petljom za trenutno preskakanje postojeće iteracije i preskakanje na sljedeću iteraciju
- Postoje situacije kad ne želite napraviti neku operaciju na nekim elementima, ostaju kakvi jesu i idete dalje



# Primjer 5

```
brojevi = [200, 0, 50, 10, -20, 0, 70]
moj_broj = 10

for broj in brojevi:
    if broj == 0:
        print("Ne mogu dijeliti s nulom, preskačem.")
        continue  # Preskoči dijeljenje ako je djelitelj nula

    rezultat = broj / moj_broj    # pripaziti na uvlačenje
    print("broj / moj_broj =", rezultat)
```

# Zadatak 8

- Imate listu automobila:
  auti = ["audi", "MERCEDES", "pEUGEOT", "BMW", "fiat"]
- Prođite kroz vašu listu i ispišite naziv svakog automobila s prvim početnim velikim slovom i ostalima malim (npr. Fiat)
- Ako se radi o automobilu marke "BMW", njega preskačete

# Kontrolne izjave u for petljama - else

- Uvećini programskih jezika (java, C/C++), korištenje else statementa je ograničeno samo na if petlju
- Python omogućava korištenje else izraza i u for petlji
- Blok koda pod else izrazom u for petlji se izvršava jedino ako nije bilo izlaza pod break izrazom

- Pogledati primjere s break kontrolnom izjavom

# Ugniježđene petlje – petlja unutar petlje

- Ugniježđena petlja je petlja koja se nalazi unutar petlje
- Unutrađnja petlja će se izvršiti kompletno izvršiti za svaku iteraciju vanjske petlje

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

- Imamo dvije liste, adj i fruits
- Prvo se pokreće petlja za adj, prvi element je red
  - Ispisuje se red te se vrti čitava unutarnja petlja za voće
  - Ispisuje se red apple, red banana, red cherry
- Kreće se s drugom iteracijom iz liste colors, ovaj put big
  - Ponvo se vrti čitava unutrađnja petlja, big apple, big banana itd…

# Primjena ugnježđenih petlji

Obrada dvodimenzionalnih podataka (rad s matricama ili tablicama), gdje jedna petlja prolazi kroz redove, a druga kroz kolone

Usporedba svih parova elemenata izmedju dvije liste

# Primjer 6

```
korisnici1 = ["Ana", "Marko", "Iva"]
korisnici2 = ["Luka", "Ana", "Petra"]

print("Potencijalni parovi korisnika:")
for korisnik_prvi in korisnici1:
    for korisnik_drugi in korisnici2:

        # Želimo izbjeći usporedbu korisnika sa samim sobom

        if korisnik_prvi != korisnik_drugi:
            print(korisnik_prvi, korisnik_drugi)
```

# Zadatak 9

- Napišite program koji će generirati tablicu množenja za brojeve u dva raspona.
- Ispišite rezultat množenja u formatu: "Broj1 * Broj2 = Umnožak".

# Sažetak lekcije

- For petlja se koristi za iteraciju kroz elemente u nekom nizu (sekvenci), kao što su liste, rasponi brojeva, tuplei itd.
- Omogućuje izvođenje bloka koda više puta, po jednom , za svaki element u sekvenci
- Funkcija range() vraća niz brojeva, počevši od 0 po defaultu, i povećava se za 1 (po defaultu) te se zaustavlja prije određenog broja
  - Sintaksa: range (start, stop, step)
- Kontrolne izjave break i continue nam omogućuju izlazak iz petlje odnosno preskakanje određene iteracije