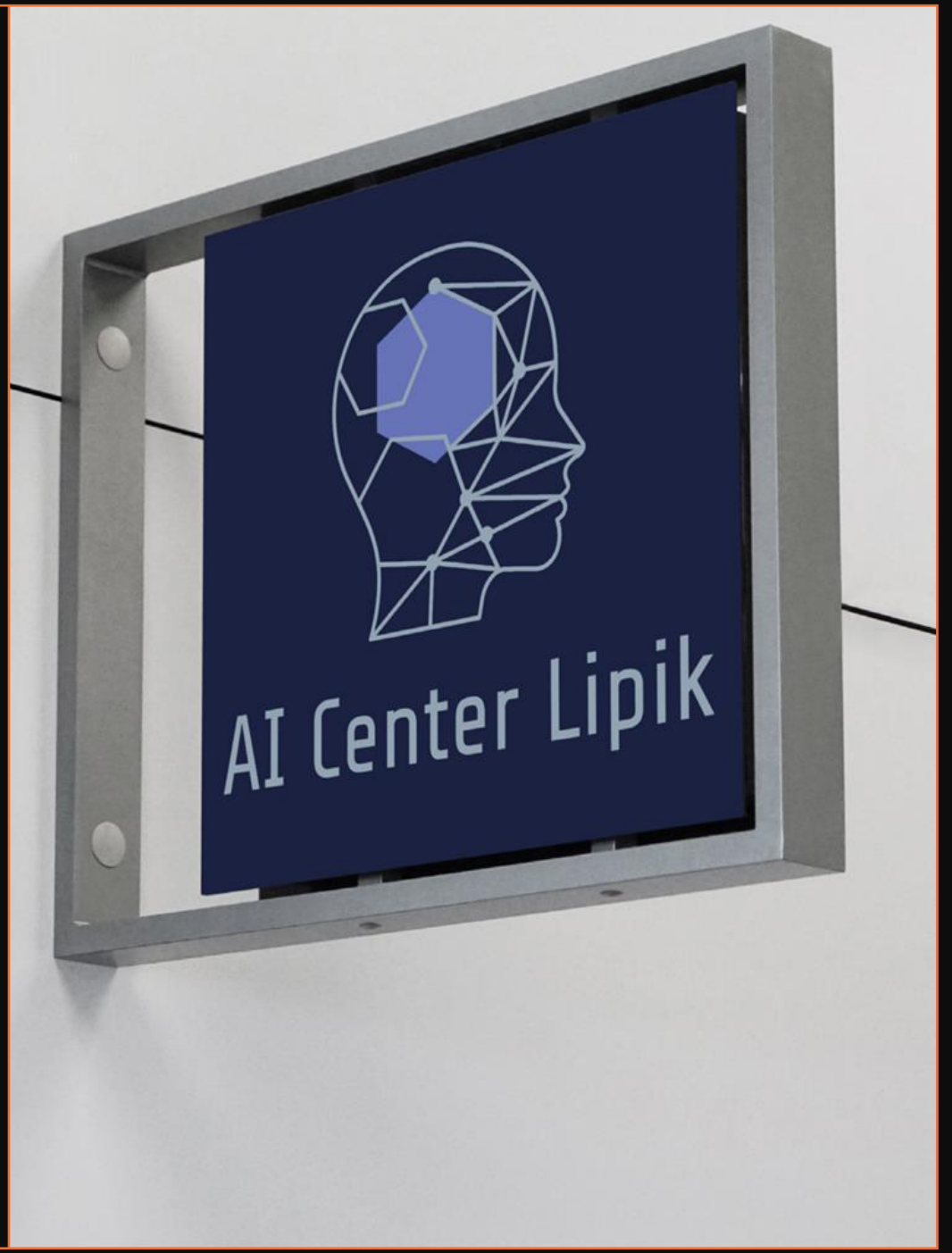
Izrada vlastitih funkcija

Igor Buzov, dipl. inf., viši predavač

Centar umjetne inteligencije Lipik



Pregled lekcije

- Osim već postojećih funkcija u Pythonu (print, type, int, float itd) programer ima mogućnost izrade i vlastitih funkcija
- U ovome poglavlju pokazat ćemo kako definirati i pozvati vlastitu funkciju
- Upoznat ćemo se s parametrima i argumentima funkcije
- Osim prikaza teksta na ekranu, funkcije se mogu koristiti i za obradu podataka i vraćanje rezultata korisniku
  - Rezultat se može dalje koristiti

Ponavljanje – što će biti rezultat ovog koda?

```
broj1 = 527.56
broj2 = int(broj1)
print (broj1)
print (broj2)
```

```
text1 = "200.50"
print (text1 * 3)
broj1 = float (text1)
print (broj1 * 3)
```

Ponavljanje – što će biti rezultat ovog koda?

```
broj1 = 100
broj2 = "petnaest"
print (type(broj1))
print (type(broj2))
```

```
mjesto = input ("Gdje si rodjen/a? ")
print ("Mjesto rođenja: ", mjesto)
```

Ponavljanje – što će biti rezultat ovog koda?

```
cijena = input ("Koja je cijena proizvoda? ")
dostava = 25
ukupno = cijena + dostava
print ("ukupni iznos je" + ukupno)
```

Ako imate grešku, kako biste popravili vaš kod?

Ponavljanje – što će biti rezultat ovog koda?

```
sifra = "Lipik_online_2025_17"
duljina_sifre = len(sifra)
print (duljina_sifre)
```

Što omogućuju vlastite funkcije?

- Organizacija koda: Razbijanje velikog programa na manje, lakše razumljive dijelove
- Smanjenje ponavljanja koda: Ako se neki dio koda koristi više puta, može ga se staviti u funkciju i pozvati tu funkciju kad god zatreba
- Poboljšavanje čitljivosti: Kada se kod podijeli u u funkcije, lakše je prati tijek izvršavanja programa
- Olakšano održavanje: Ako treba promijeniti neki dio koda, to će se učiniti samo jednom, unutar funkcije, a promjena će se odraziti svugdje gdje se ta funkcija koristi

Što su funkcije

- Funkcija je blok koda koji se izvršava samo kad se pozove
- Postoje dvije vrste:
  - Ugrađene u Python, dio izvornog koda
  - Korisnički napravljene, vlastite
- Funkciji se mogu proslijediti podaci. Te podatke zovemo parametri
- Funkcija ima mogućnost vraćanja podataka kao rezultat (return value)

Kada koristimo vlastite funkcije

- U situaciji gdje više puta moramo koristiti jedan te isti kod, zašto taj dio koda ne bi stabili u funkciju i onda nju pozivali?
  - Primjer: željeli bismo pozvati funkciju koja ispisuje automatski "Hello world!"
  - Primjer: u igrici vaš heroj puca iz određenog oružja. Radi li se o pištolju, vatrenim kuglama ili samostrelu, isti je princip. Nećete svaki put pisati isti kod, pozvat ćete već definiranu funkciju
- Možemo napraviti vlastite funkcije i pozivati ih kad nam trebaju
- Za izradu vlastitih funkcija koristi se ključna riječ

def

Izrada vlastite funkcije - sintaksa

```
def hello():
    print ("Hello world!")
    print ("AI centar Lipik)


hello()
```

- Izrada vlastite funkcije počinje korištenjem ključne riječi def
- Nakon toga odabiremo naziv funkcije (preporuča se korištenje naziva koji nam olakšava razumijevanje što funkcija radi)
- Otvaramo i zatvaramo zagradu te slijedi znak dvotočke
- Što će raditi funkcija se nalazi UVUČENO ispod naziva
- Pozivamo funkciju ispisivanjem njenog naziva sa zagradama

Dijelovi funkcije

Definiranje funkcije
(Function definition)
def naziv ():
tijelo funkcije
Jedanput

Pozivanje funkcije
(Function call)
Jedanput ili više puta
naziv()

Zadatak 1

Otvorite novi file koji ćete nazvati "vlastite_funkcije.py"

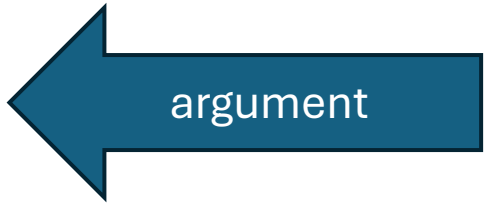
Napravite funkciju u kojoj ćete, nakon što je pozovete, ispisati nekoliko redova teksta s ponudom ljetovanja u Zanzibaru

Prosljeđivanje podataka funkciji (parametri i argumenti)

- Pri pozivanju funkcije možete proslijediti podatke funkciji koja će ona dalje koristiti

```
def prebivaliste (grad):
        print ("Prebivaliste zaposlenika glasi " + grad)

prebivaliste ("Zagreb")
prebivaliste ("Split")
```



Zadatak 2

- Napravite funkciju koja se zove "favorite_book", koja ima jedan parametar pod nazivom "title"
- Funkcija bi trebala ispisati tekst kao što je:
  - Jedna od mojih najdražih knjiga je Hobbit
- Pozovite funkciju u kojoj ćete kao argument staviti naziv vaše najdraže knjige

Pozicijski argumenti

- Pri pozivanju funkcije možemo koristiti više argumenata, potrebno ih je odvojiti samo zarezom
- Ovaj način korištenja argumenata zove se pozicijski argumenti

```
def describe_pet (animal, animal_name):
    print ("Moj ljubimac je " + animal)
    print ("Zove se " + animal_name)

describe_pet ("hrčak", "Harry")
```

Oprez!

- Što će biti rezultat ovo koda?

```
def describe_pet (animal, animal_name):
    print ("Moj ljubimac je " + animal)
    print ("Zove se " + animal_name)

describe_pet ("Harry", "hrčak")
```

- Kao što i sam naziv kaže, pozicija argumenta je važna. Ako im zamijenite raspored, može doći do logičke greške!

Zadatak 3

- Napravite funkciju koja se zove "auti" koja ima dva parametra: marka_auta i zemlja_porijekla
- Funkcija bi trebala ispisati tekst kao što je:
  - Moj najdraži auto je BMW
  - Zemlja porijekla je Njemačka
- Pozovite funkciju u kojoj ćete kao argumente staviti vašu najdražu marku automobila i zemlju iz koje potiče taj automobil

Zadatak 4

- Napravite funkciju koja se zove "informacije" koja ima tri parametra: prezime, ime i godinu rođenja
- Funkcija bi trebala ispisati tekst kao što je:
  - Prezime i ime korisnika: Perić Pero
  - Godina rođenja: 1993
  - Ogovarajućom metodom na imenu i prezimenu namjestite obavezno veliko početno slovo
- Pozovite funkciju u kojoj ćete kao argumente staviti prezime, ime i godinu rođenja

Primjer – objasnite kod, što će biti rezultat?

```
def ime_heroja(hero):
        print ("Hello " + hero)
        print ("Welcome to the valley! This is where your adventure begins!")

name = input ("What is the name of your hero?")
ime_heroja(name)
```

Zadatak 5

- Napišite kod u kojem korisnika pitate za njegov nadimak, odgovor pohranite u odgovarajuću varijablu
- Napravite funkciju koja kao rezultat ispisuje tekst kao što je:
  - Pozdrav Visoki
- Pri pozivanju funkcije, u argument stavite nadimak koji je unio korisnik

Zadatak 6

- Napišite kod u kojem korisnika pitate za naziv omiljenog filma i odgovor pohranite u odgovarajuću varijablu
- Nakon toga pitajte korisnika za ime glavnog glumca iz tog filma i odgovor isto pohranite u odgovarajuću varijablu
- Pozovite vlastitu funkciju u kojoj kao argumente stavite ime filma i ime glumca koje je unio korisnik
- Pozvana funkcija bi trebala ispisivati tekst tipa:

"Vaš omiljeni film je Terminator, u kome glumi Arnold Schwarzneger!"

Keyword arguments

- Ako ne želite voditi brigu o redoslijedu argumenata kao što imamo kod pozicijskih argumenata (situacija da je životinja Harry, a ime ljubimca hrčak), koristimo keywords arguments
- Pri pozivanju funkcije, unutar zagrade pišemo parove name – value
- Korištenjem keyword argumenata ne moramo razmišljati o pravilnom rasporedu argumenata u pozivu funkcije

Keyword arguments

```
def describe_pet (animal, animal_name):
    print ("Moj ljubimac je " + animal)
    print ("Zove se " + animal_name)

describe_pet (animal_name = "Harry", animal = "hrčak")
```

- Pri definiraju funkcije nije bilo promjena
- Zamijenili smo redoslijed argumenata pri pozivanju funkcije, ali zbog keyword argumenata, sve radi po planu
- Pri pozivanju funkcije, mora se paziti da se koriste isti nazivi parametara kao i u definiciji funkcije

Zadatak 7

- Od korisnika zatražite naziv grada koji bi htio posjetiti, pohranite vrijednost
- Od korisnika zatražite naziv zemlje u kojoj se nalazi taj grad, pohranite vrijednost
- Napravite funkciju koja kao rezultat ispisuje tekst kao što je:
  - Austrija i Beč su prekrasni za posjetiti u ovo doba godine!
- Pozovite vašu funkciju, koristite keyword argumente

Zadatak 8

- Napravite funkciju koja računa kvadraturu sobe ako joj predate duljinu i širinu prostorije
- Duljina i širina su brojevi
- Kao rezultat neka se ispiše tekst kao što je:
  - Površina sobe iznosi 55 kvadrata

Primjer – objasnite kod, što će biti rezultat?

```
def describe_pet (animal, animal_name):
    print ("Moj ljubimac je " + animal)
    print ("Zove se " + animal_name)

describe_pet ()
```

Return vrijednosti

- **ImeFunkcije (argumenti) → side effect**
  - Vrijedi i za ugrađene funkcije i za vlastite funkcije
- Do sada smo kao rezultat funkcije imali ispis na ekranu – nekakav side effect
- Taj rezultat nije bilo moguće ponovo koristiti
- Ako želimo ponovo, i u drugim dijelovima programa iskoristiti rezultat funkcije, treba nam način kojim bi nam funkcija VRATILA rezultat
- Na scenu stupaju return vrijednosti

Zašto koristiti return vrijednost?

- Ponovna upotreba rezultata: Možete pohraniti rezultat funkcije u varijablu i koristiti ga kasnije u programu
- Modularnost: Funkcije postaju samostalne jedinice koje obavljaju specifične zadatke i vraćaju rezultate
- Čitljivost: Kod postaje čitljiviji jer su funkcije fokusirane na jedan zadatak
- Efikasnost: Izbjegava se ponavljanje istog koda

- Primjer – funkcija za izračun cijene s pdv-om koja vraća vrijednost korisniku

Primjer korištenja funkcije bez return-a

```
def cijena_pdv (iznos):
        print (iznos * 1.25)

cijena_pdv (150)
```

- Kao rezultat na ekranu će se ispisati 187.5
- Što ako bismo sad tu vrijednost htjeli pomnožiti s brojem komada kako bi izračunali ukupan iznos računa?
- Vrijednost od 187.5 bi trebali nekako izvući van iz funkcije (treba nam return)

Sintaksa korištenja return vrijednosti

```
def function_name (parameter1, parameter2...)
    # function_body
    return return_value

result = function_name(arg1, arg2 ...)
```

Primjer korištenja funkcije s upotrebom return-a

```
def pdv (iznos):
    izracun = iznos * 1.25
    return izracun

cijenaspdvom = pdv (150)
kolicina = 10

print ("Ukupni iznos racuna iznosi:")
ukupno = kolicina * cijenaspdvom
print (ukupno)
```

- Definiramo funkciju s jednim argumentom
- Radimo izračun
- Koristimo return kako bismo dobivenu vrijednost vratili calling funkciji

- Pozivamo našu funkciju, predajemo joj vrijednost od 150
- Dobiveni rezultat se pohranjuje u varijabli

- Računamo ukupnu vrijednost

Primjer – objasnite kod, što će biti rezultat?

```
def korisnik (ime, prezime):
    puno_ime = ime + " " + prezime
    return puno_ime

ime_i_prezime = korisnik ("Nikola", "Tesla")
print (ime_i_prezime)
print (ime_i_prezime.upper())
```

Zadatak 9

- Od korisnika zatražite da unese broj koji predstavlja udaljenost u kilometrima (npr. korisnik unese broj 157)
- Napravite funkciju koja se zove **km_to_miles** koja uzima kilometre te ih pretvara u milje
- Funkcija mora vratiti (return) rezultat korisniku
- Ispišite rezultat na ekranu koji ste dobili

Zadatak 10

- Definirajte funkciju koja izračunava kvadrat broja i vraća rezultat
- Primjer:
- Pozivamo funkciju gdje kao argument stavljamo broj 5

  kvadrat (5)
- Kao rezultat dobijemo tekst

  Kvadrat ovog broja iznosi: 25

Primjer – objasnite kod, što će biti rezultat?

```
def calc (a, b):
    zbroj = a + b
    razlika = a - b
    umnozak = a * b
    return zbroj, razlika, umnozak


zbrajanje, oduzimanje, mnozenje = calc (20, 10)


print ("Zbroj ova dva broja iznosi: " + str(zbrajanje))
print ("Oduzimanje ova dva broja iznosi: " + str(oduzimanje))
print ("Množenje ova dva broja iznosi: " + str(mnozenje))
```

Vraćanje višestrukih vrijednosti iz funkcije

Ako želite vratiti više vrijednosti iz funkcije, potrebno je koristiti return te navesti vaše vrijednosti koje odvajate zarezom

Struktura podataka koju ste inače dobili zove se tuple (u ovom slučaju nam nisu potrene zagrade)

Default values u funkcijama

- Pri definiranju funkcije, može se definirati zadana (default) vrijednost za svaki parametar
- Ako se pri pozivu funkcije ipak navede vrijednost, Python koristi vrijednost koju je dobio pri pozivu. Ako se pri pozivu ne navede vrijednost, Python koristi defaultnu vrijednost
- Često se koristi kod funkcija gdje određeni parametri imaju skoro uvijek istu vrijednost

Korištenje defaultnih vrijednosti

```
def krug (r, pi =3.14):
    povrsina = 2 * r * pi
    return povrsina

pov_krug = krug (5, 3.1415)
print ("Površina kruga iznosi: " + str (pov_krug))
```

- Ne koriste se defaultne vrijednosti, pi će iznositi 3,1415, unešen je pri pozivu funkcije

```
def krug (r, pi = 3.14):
    povrsina = 2 * r * pi
    return povrsina

pov_krug = krug (5)
print ("Površina kruga iznosi: " + str (pov_krug))
```

- Koristi se defaultna vrijednost jer pri pozivu nije navedena, pi će iznositi 3.14

Zadatak 11

- Definirat ćete funkciju koja računa trošak goriva na osnovi dva parametra, broja litara i cijene jedne litre goriva te vraća rezultat
- Parametar cijena goriva ima defaultnu vrijednost od 1.5
- Pozovite funkciju s korištenjem dva argumenta
- Pozovite funkciju korištenjem samo jednog argumenta (broj litara)

Sažetak lekcije

- Ključna riječ koja se koristi za definiranje vlastite funkcije se zove def
- Funkcija se prvo definira, a onda se poziva
- Pri definiranju funkcije u zagradi definiramo parametre, a pri pozivanju u zagradi definiramo argumente
- Parametri mogu imati default vrijednosti
- Argumenti mogu biti pozicijski te keyword
- Ključna riječ koju koristimo ako želimo dobiti povrat vrijednosti iz funkcije se zove return