



UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

Računarstvo u oblaku

ms Helena Anišić

Zimski semester 2022/2023.

Studijski program: Računarstvo i automatika

Modul: Računarstvo visokih performansi

Osnovne informacije

Materijali za vežbe i informacije o predmetu: <http://www.acs.uns.ac.rs/>

Pitanja i konsultacije: hanisic@uns.ac.rs

Sadržaj kursa

- Razvoj male veb aplikacije (Python+Django)
- Razlika između virtuelne mašine i kontejnera (Vagrant)
- Docker
- Analiza Docker Compose fajla za Hyperledger Fabric
- Docker Swarm, Kubernetes
- Osnove CI/CD (github actions)
- Cloud platforme (AWS)

Način polaganja



Python

- Razvoj je započet 1989. godine u Holandiji kao hobi projekat Gvida Van Rosuma.
- Interpretiran dinamički jezik visokog nivoa.
- Obuhvata više paradigmi: imperativno, proceduralno, objektno, funkcionalno,...
- Dolazi sa “baterijama” jer sadrži sveobuhvatnu standardnu biblioteku.
- Akcenat na efikasnosti programera i čitkosti koda
- Dizajn filozofija podržava čitkost upotrebom uvlačenja bloka koda umesto uporebe vitičastih zagrada { i }
- Poslednja verzija Python-a 2 je 2.7, a danas se koriste različite verzije Python 3
- Godinama najpopularniji programski jezik

Upotreba Python-a

- Web Development
- Data Science
- Machine Learning
- Artificial Intelligence
- Web Scraping
- Automation

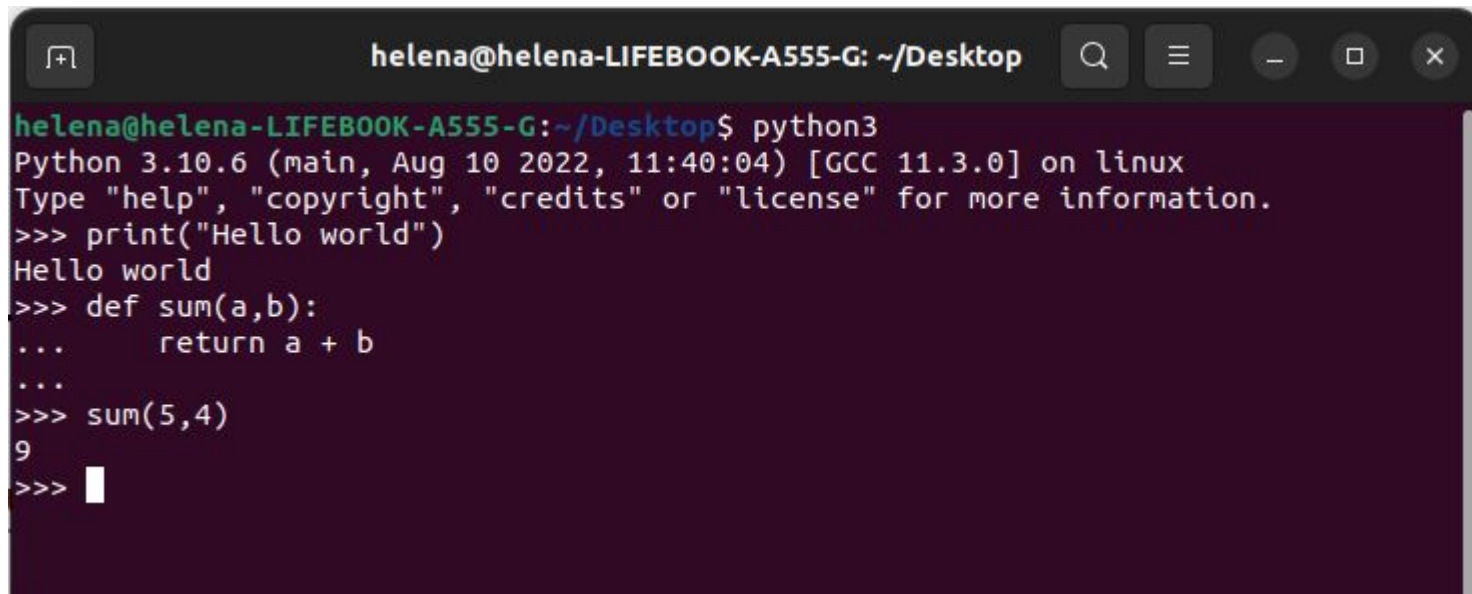


Pokretanje Python programa

Nakon instaliranja Python 3, programi se mogu pokretati na dva načina:

- Interaktivni prevodilac (Python konzola)
- pisanje programa u tekstualne fajlove sa ekstenzijom **.py** i njihovim pokretanjem kucanjem komande `python` i navođenjem imena datog fajla u terminal računara

Python konzola



```
helena@helena-LIFEB00K-A555-G: ~/Desktop
helena@helena-LIFEB00K-A555-G:~/Desktop$ python3
Python 3.10.6 (main, Aug 10 2022, 11:40:04) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world")
Hello world
>>> def sum(a,b):
...     return a + b
...
>>> sum(5,4)
9
>>> 
```

The Zen of Python (Python filozofija)

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Indentacija u Python-u

- Indentacija u Python-u je obavezna i koristi se kao zamena za vitičaste zagrade prilikom pisanja bloka koda u programu
- Nije propisana širina uvlačenja (minimum 1 space), ali mora biti konzistentna
- Preporučeno je 4 [space](#) karaktera za uvlačenje
- Preporuka je da se koriste [space](#) karakteri umesto [tab](#)
- Primer nekonzistentne indentacije:

```
if a:  
    statement1  
    statement2  
else:  
    statement3  
    statement4
```

Python User Input/Output

- Python omogućava korisnicima da unose podatke preko standardnog ulaza

```
username = input("Enter username")  
print("Username is: " + username)
```

- Python omogućava ispis podataka pomoću print() funkcije

```
print("Hello World")  
print("Number of students:", 28)
```

Komentari

- Jednolinijski komentari - #
- Višelinijijski komentari - # (na početku svakog reda) ili trostruki navodnici

ovo je komentar

"""

Ovo je višelinijijski
komentar.

"""

Osnovni tipovi podataka

- Bulov tip (ima vrednost **True** ili **False** - **bool**)
- Stringovi (nizovi tekstualnih karaktera - **str**)
- Numerički
 - Celi broj - **int**
 - Broj u pokretnom zarezu - **float**
 - Kompleksni brojevi - **complex**

Deklaracija promenljivih

- Za deklarisanje promenljivih nije potrebna nikakva specijalna komanda poput var
- Deklarisanje promenljivih ne zahteva definisanje tipa podataka
- Promenljiva je kreirana onog momenta kada joj se dodeli neka vrednost

```
a = 10  
name = "John"  
a = "Sam"
```

Složeni tipovi za podatke

- Sekvence: str, list, tuple
- Mape: dict
- Skupovi: set, frozenset

Sekvence

- Sekvence: **str**, **unicode**, **list**, **tuple**
- Sekvence je uređena kolekcija objekata indeksirana nenegativnim rednim brojem.
- **String** i **N-torka (Tuple)** - nepromenljiva sekvenca karaktera
- **List** - promenljiva sekvenca proizvoljnih objekata
- Sve sekvence podržavaju slicing i iteracije

```
tuple = ("apple", "banana", "cherry")  
print(tuple[0])
```

```
list = ["apple", "banana", "cherry"]  
print(list[0])
```

Skupovi

- Elementima u setu se ne može pristupiti preko indeksa ili ključa
- Jedini način pristupa elementima u setu jeste iteracijom pomoću for petlje ili upitom da li neka vrednost postoji uz upotrebu **in** ključne reči

```
thisset = {"apple", "banana", "cherry"}
```

```
for x in thisset:  
    print(x)
```

```
print("banana" in thisset)
```

Mape

- Rečnici (**dict**) su ugrađeni tip i predstavljaju implementaciju hash tabela ili asocijativnih nizova
- Rečnici služe za skladištenje podataka kao **ključ:vrednost**
- Dict kao ograničenje za ključeve zahteva nepromenljivost (immutability) jer hash vrednost mora biti konstantna

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}
```

```
thisdict["color"] = "red"
```

Callables

- Objekti koji podržavaju semantiku poziva
- Funkcije, klase, metode
- Tretiraju se kao i svi drugi objekti - mogu biti elementi kolekcija, mogu se prosleđivati kao parametri, biti povratne vrednosti drugih callables itd...

```
def foo(x,y):  
    return x + y
```

```
bar = lambda x,y: x + y
```

```
funkcije = [foo, bar]  
for f in funkcije:  
    print(f(2,3))
```

Naredbe grananja

- Uslov se ne mora da se stavlja unutar () kao u nekim drugim programskim jezicima
- Blok koda za izvršavanje u okviru naredbe grananja se ne naznačava { } niti rečim begin i end, već samo uvlačenjem (4 space-a ili 1 tab)

```
if <condition>:  
    <statement>  
elif <condition>:  
    <statement>  
else:  
    <statement>
```

Naredbe ciklusa - for

- **for** petlja se koristi za iteraciju preko niza elemenata (može da bude lista, set, rečnik, string,...)
- **for** nije poput **for** petlje u drugim jezicima (primer C), već je više kao iterator koji se koristi u objektno-orijentisanim jezicima

```
fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
    print(x)
```

- **break** naredba omogućava da se napusti for petlja
- **continue** naredba omogućava prelazak na narednu iteraciju pre nego što se sve naredbe za trenutnu iteraciju izvrše
- **for** petlja može imati opcioni **else** blok koji se izvršava ukoliko se petlja nije završila prevremeno (**break**)

Naredbe ciklusa - for

- Ukoliko je potrebna for petlja koja će iterirati tačno određeni broj puta, možemo koristiti `range()` funkciju
- Funkcija `range()` vraća niz brojeva počevši od 0, sa inkrementom od 1 i završno sa specificiranim brojem
- `range(6)` -> 0,1,2,3,4,5,6

```
for x in range(2,6):  
    print(x)
```

Python pass izraz

- **Pass** izraz se koristi kao placeholder za budući kod
- Kada se pass izraz izvrši, ništa se ne desi, ali se izbegne moguća greške u situacijama kada prazan kod nije dozvoljen
- Prazan kod nije dozvoljen u petljama, definicijama funkcija, definicijama klasa ili u naredbi grananja

```
for x in [0,1,2]:  
    pass
```

```
class Person:  
    pass
```


Naredba ciklusa - while

- While petlja omogućava izvršavanje niza naredbi sve dok je uslov tačan
- Break i continue se koriste isto kao i kod for petlje
- Specijalna vrsta while petlje sa else naredbom koja će se izvršiti samo jednom kada se while završi

```
i = 1
while i < 6:
    print(i)
    i += 1
else:
    print("i is no longer less than 6")
```

Funkcija

Definisanje funkcije i pozivanje funkcije:

```
def add(x, y):  
    return x + y  
  
add(5, 4)
```

Definisanje lambda funkcije:

```
lambda args : expression  
  
my_list = [1,2,3,4]  
new_list = list(filter(lambda x: (x%2 == 0), my_list))
```

Argumenti i parametri funkcije

- **Pozicioni argumenti** - vrednosti se kopiraju po redu u njihove odgovarajuće parametre
- **Imenovani argumenti** - određivanje argumenata prema imenima njihovih odgovarajućih parametara, čak i u drugačijem redosledu od njihovih definicija
- **Podrazumevana vrednost parametara**

```
def add(x,y=3):  
    return x + y
```

```
add(1,2)
```

```
add(y=2,x=1)
```

```
add(0)
```

Klase

```
class Person():  
    def __init__(self, name, age):    #konstruktor  
        self.name = name  
        self.age = age  
  
p = Person("John", 36)
```

Šta se dešava nakon linije koda `p = Person("John", 36)`:

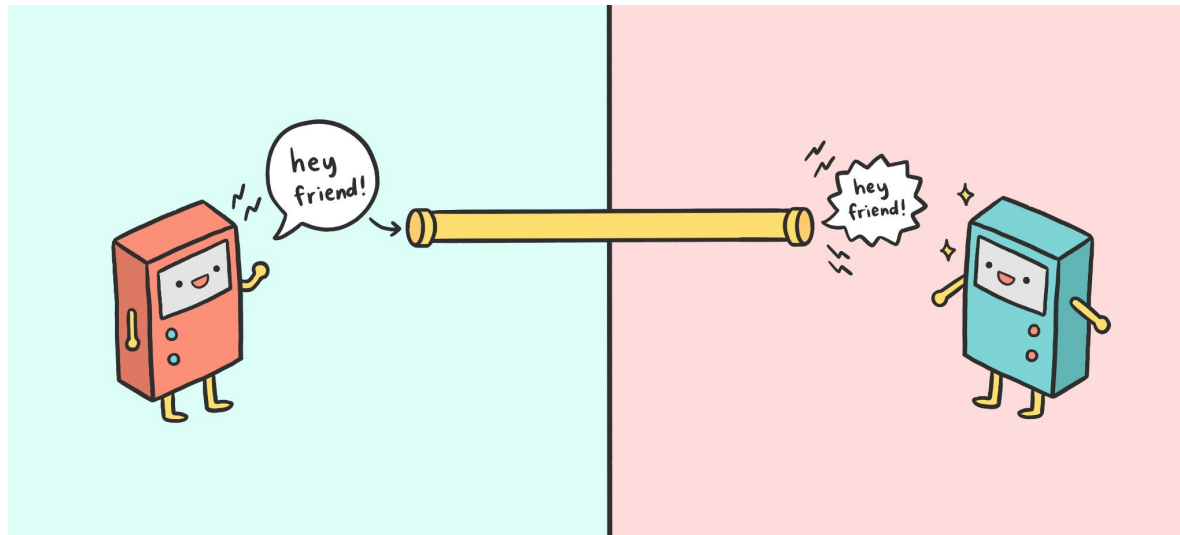
- Pretražuje definiciju klase `Person`
- Stvara novi objekat u memoriji
- Poziva metodu objekta `__init__`, prosleđujući novostvoreni objekat kao `self`, drugi argument kao ime, a treći kao broj godina
- Skladišti vrednost imena (`name`) i broja godina (`age`) objekta
- Vraća novi objekat
- Privezuje ime `p` za novi objekat

Klase

```
class Person():  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
  
    def myfunc(self):  
        print("Hello my name is " + self.name)  
  
p = Person("John", 36)  
p.myfunc()
```

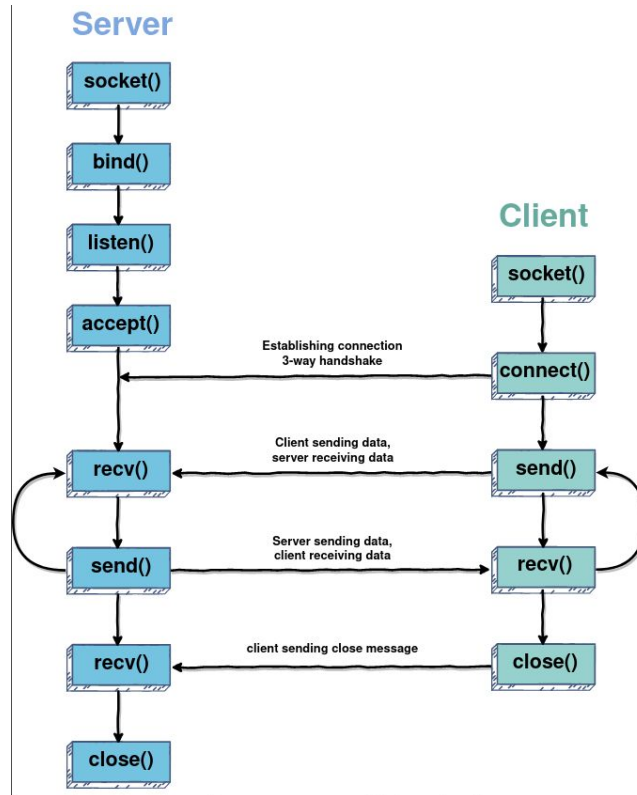
- Objekti takođe mogu da sadrže metode
- Self parametar je referenca na trenutnu instancu klase i služi za pristup promenljivima koje pripadaju datoj klasi
- Naziv parametra ne mora da bude self, ali mora da bude prvi parametar svake metode unutar klase

Veb razvoj u Python-u



OSI - TCP/IP

TCP/IP model	Protocols and services	OSI model
Application	HTTP, FTP, Telnet, NTP, DHCP, PING	Application
		Presentation
		Session
Transport	TCP, UDP	Transport
Network	IP, ARP, ICMP, IGMP	Network
Network Interface	Ethernet	Data Link
		Physical

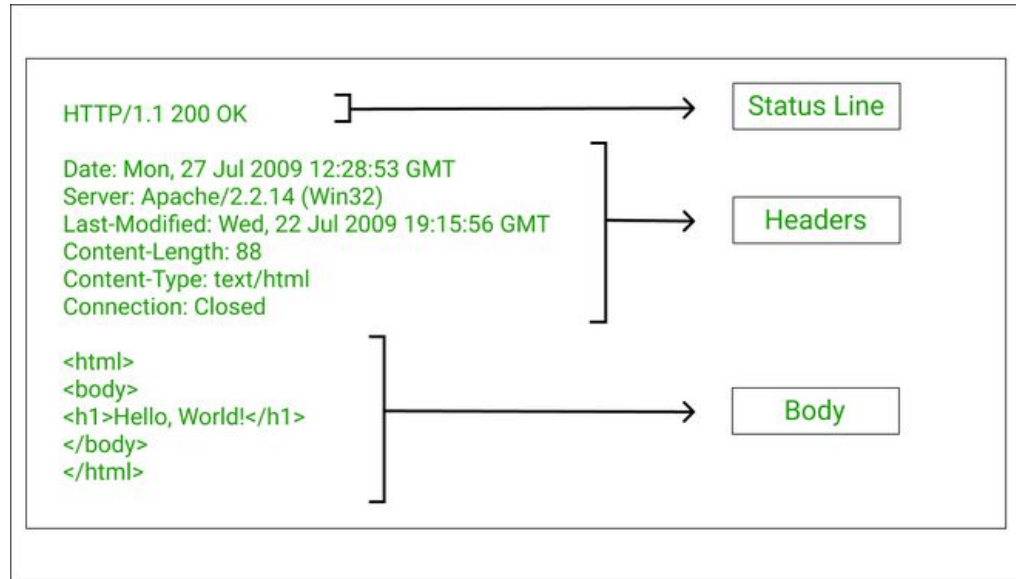


HTTP Request

HTTP Request



HTTP Response



Materijali za učenje Python-a

- Python izazov: <http://www.pythonchallenge.com/>
- Python tutor: <https://pythontutor.com/>
- Python dokumentacija: <https://docs.python.org/3/>
- Python kurs prof. Dejanovića: <https://www.igordejanovic.net/courses/tech/Python/>
- Python kurs w3schools: <https://www.w3schools.com/python/default.asp>