



## CS324 - SKRIPTING JEZICI

Uvod u predmet

Lekcija 01

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 01

### *UVOD U PREDMET*

- ✓ Uvod u predmet
- ✓ Poglavlje 1: Teorija jezika i koncept programiranja
- ✓ Poglavlje 2: Imperativno programiranje
- ✓ Poglavlje 3: Deklarativno programiranje
- ✓ Poglavlje 4: Pojam skripting jezika
- ✓ Poglavlje 5: Specifični jezici za aplikacije
- ✓ Poglavlje 6: Pokazna vežba #1
- ✓ Poglavlje 7: Individualna vežba #1
- ✓ Poglavlje 8: Domaći zadatak #1
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ❖ Uvod

### UVOD

*Ova lekcija predstavlja uvod u oblast skripting jezika. Studenti se upoznaju sa pojmom skripting jezika, osnovnim konceptima kao i vrstama i tipovima skripting programiranja.*

#### **Dobrodošli na predmet CS324 - Skripting jezici!**

Cilj predmeta CS324 jeste da se student upozna sa skripting jezikom Python, njegovu prirodu, sintaksu i semantiku, strukture, i dinamičke karakteristike.

Ranije su pored jezika Python na ovom predmetu radili i jezici Perl i Ruby. Međutim, Python je toliko popularan i tražen programski jezik da se ceo predmet fokusira isključivo na primenama ovog programskog jezika.

Akcentat je najpre dat na savladavanju proceduralnog programiranja u Python 3 jeziku, koje se koristi pri mnogim inženjerskim aplikacijama i u aplikacijama u analizi podataka, pa tek onda se nastavlja ka objektno-orientisanim paradigmama. Obrađuju se najpopularnije biblioteke i moduli koje Python 3 koristi, radno okruženje Flask, kao i razvoj RESTful web servisa.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### PRAVILA VEZANA ZA PREDMET

*U ovom delu opisana su pravila vezana za predmet, kao i način polaganja ispita.*

Student se ocenjuje u toku celog semestra. Ocenuju se njegovi domaći zadaci, rad na projektu, testovi i aktivnost u nastavi. Na kraju u ispitnom roku, ocenjuje se i pismeni ispit. Ocene se daju u poenima.

Maksimalni broj poena je **100** (uključujući i pismeni ispit). Na pismenom ispitnu student može dobiti do **30** poena, a aktivnosti u toku semestra (predispitne obaveze) mogu mu doneti do **70** poena, po sledećoj strukturi:

1. 15 domaćih zadataka, svaki donosi po **1.5** poen (ukupno **22.5** poena),
2. 5 testova, svaki donosi po **2.5** poena (ukupno **12.5** poena),
3. 1 Projektni zadatak, donosi **25** poena,
4. Zalaganje na nastavi, do **10** poena.

### Forum

Cilj foruma je da podstiče proaktivni rad studenata i njihovu saradnju tokom nastave na predmetu. Svaki forum omogućava da student postavi problem, vezan za lekciju, i da potraži pomoć svojih kolega. Bilo da se problem odnosi na deo predavanja, bilo na deo vežbi. Za postavljanje problema na forum, studentu se priznaje aktivnost na forumu. Isto se priznaje aktivnost na forumu i studentu koji prvi reaguje na ovaj poziv studenta za pomoć, pod uslovom da je njegov savet uspešno rešio problem koji je student postavio.

### **Domaći zadaci**

Posle izučavanja određene nastavne jedinice, odnosno, posle vežbi u okviru jednog predavanja (jedna nedelja) predviđeno je da studenti dobiju zadatak koji treba samostalno da reše.

*Svaki student dobija različit zadatak i kao preduslov za izlazak na ispit u obavezi je da uradi i pošalje sve zadatke.*

Predviđeno je ukupno 15 zadataka, a svaki uspešno rešen zadatak predat u zadatom roku obezbeđuje studentu 1,5 poena, pod uslovom uspešne odbrane urađenog zadatka.

Za studente tradicionalne (u Beogradu) ili hibridne nastave (Centar u Nišu) rok za predaju zadataka je **7 dana nakon izdavanja**, a posle tog roka umanjuje ostvaren broj poena za 50%.

***Krajnji rok za predaju domaćeg zadatka i za studente tradicionalne nastave i za studente onlajn nastave je 10 (deset) dana pre ispitnog roka u kome student polaže ispit.***

Studenti online nastave brane domaće zadatke u dogovoru sa predmetnim asistentom ili nastavnikom.

## **NAČINI OCENJIVANJA PREDISPITNIH OBAVEZA**

*Student dobija poene tokom semestra na predispitnim obavezama.*

### **Testovi**

U skladu sa Planom nastave predviđeno je da student u naznačenim nedeljama treba da uradi ukupno 5 testova, čime može da obezbedi najviše 10 poena, jer svaki test može obezbediti najviše 2 poena.

Student bi trebalo da uradi test u predviđenoj nedelji, odnosno u roku od 7 dana od aktivacije. Za studente tradicionalne i hibridne nastave (centri u Beogradu i u Nišu), za svaki test urađen van datog termina, poen dobijene testom se umanjuju za 50%, s tim što je krajnji rok za polaganje testa - 10 dana pre ispita.

Studenti onlajn nastave testove moraju uraditi najkasnije 10 dana pre ispita (bez umanjenja broja poena).

### **Zalaganje**

Aktivnost u nastavi (određivanje poena za zalaganje) se delom ocenjuje drugačije za studente tradicionalne/hibridne nastave (centri u Beogradu i u Nišu), i za online studente.

Kod studenata tradicionalnog i hibridnog oblika nastave (centri u Beogradu i u Nišu) primenjuju se sledeći kriterijumi:

- Redovnost u pohađanju nastave. Student tradicionalne ili hibridne nastave koji, iz bilo kog razloga, nije pohađao nastavu na više od 30% časova predavanja i vežbanja (**pet izostanaka**), dobija automatski 0 poena na zalaganje.
- Dolazi pripremljen za nastavu.
- Redovnost i kvalitet ispunjenja predispitnih obaveza. **Student koji ne uradi sve svoje predispitne obaveze najkasnije 15 dana po završetku nastave na predmetu, dobija nula (0) poena za zalaganje.**

Kod online studenata (studije preko Interneta)

- Ranije (u odnosu na ispit) ispunjenje predispitnih obaveza
- Konsultacije tokom semestra u vezi rada na predispitnim obavezama.

## NAČINI OCENJIVANJA PROJEKTNOG ZADATKA

*Projekat je najvažnija aktivnost studenta tokom semestra.*

### Projekat

Zahtevan rok za predaju Izveštaja o urađenom projektnom zadatku je do kraja 14. nedelje nastave za studente tradicionalne studente nastave (Beograd i Niš), a za studente online nastave najkasnije 10 dana pre ispita. **Studentima tradicionalne i hibridne nastave koji predaju Izveštaj o urađenom projektnom zadatku 15 i više dana po završenoj nastavi na predmetu, umanjuje se broj ostvarenih poena za 30%.**

Obrana projekta za studente internet nastave obavlja se online, a ako je za studenta prihvatljivo, u prostorijama Univerziteta.

*Krajnji rok za predaju projekta, za sve studente, je 10 dana pre ispitnog roka u kome žele da polažu ispit. Studentima online nastave se ne umanjuje broj poena, jer su oni, po pravilu zaposleni, ili sprečeni da redovno studiraju.*

Ako student prilikom ocenjivanja projekta ne dobije najmanje 50% predviđenih poena (15 poena), mora da doradi projekat. U suprotnom, dobija 0 poena. **Student koji ne dobije više od 50% predviđenih poena ne može izaći na ispit.** Student je dužan da projekat odbrani kod asistenta ili nastavnika.

Obrana projekata studenata tradicionalne nastave i hibridne nastave vrši se u 15. nedelji jesenjeg semestra za vreme vežbi, a ako je potrebno, i van vežbi – na dodatnim časovima. Van ovog termina, student može da brani projekat u posebnom terminu koji odredi asistent pred svaki ispitni rok.

Cilj odbrane projekta je da asistent ustanovi da li je student samostalno radio projekat i u tom cilju odbrana projekta podrazumeva da student demonstrira znanje koje je iskoristio za izradu projekta.

## ▼ Poglavlje 1

# Teorija jezika i koncept programiranja

## UVOD U TEORIJU JEZIKA

*Teorija programskega jezika spada v discipline informatike, zavisi ali utiče na matematiko, softversko inženjerstvo in lingvistiku.*

**Teorija programskega jezika** je grana informatike, ki se bavi dizajnom, implementacijom, analizom in klasifikacijom programskega jezika in njegovimi pojedinačnimi karakteristikami.

Postoje različni tipovi jezika, kot so:

- Funkcionalni jeziki in proceduralni jeziki
- Imperativni jeziki in deklarativni jeziki
- Dinamični jeziki in statični jeziki
- Jaki jeziki in slabni jeziki

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ŠTA PREDSTAVLJA PROGRAMSKI JEZIK?

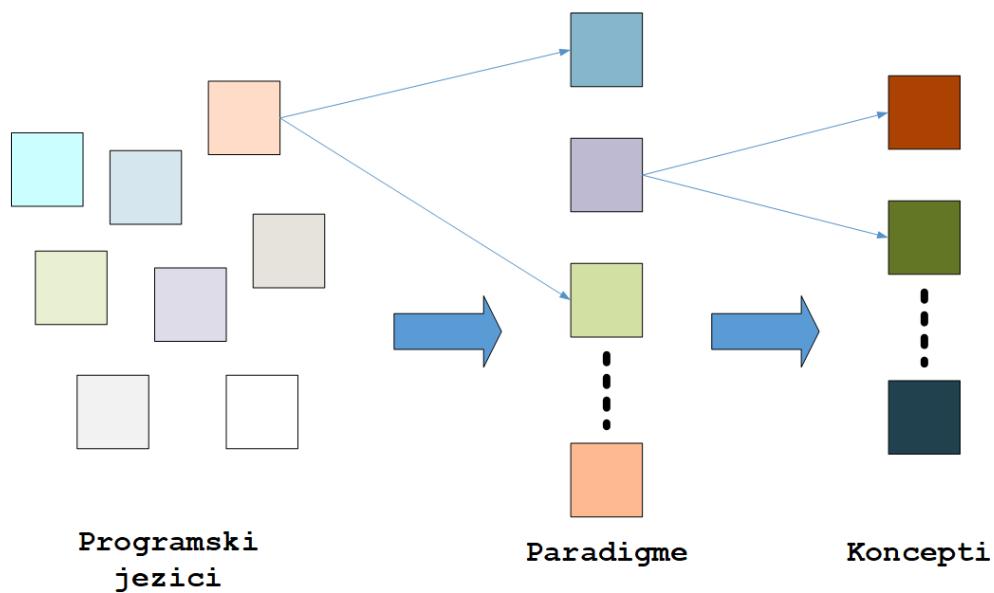
*Programski jezik je veštački jezik, ki se lahko uporablja za kontrolu ponašanja računalnika.*

**Programski jezik** (en. **programming language**) je veštački jezik, ki se lahko uporablja za kontrolu ponašanja računalnika.

Definisan je preko sintaksnih in semantičnih pravila, ki opisuje njegov strukturo.

Ski jezik realizira eno ali več paradigme (en. **paradigm**), in vsaka **paradigma** ima eno **koncept** (en. **concept**), ali češče, sestoji iz skupine konceptov.

Programski jezik je način, da se abstraktne karakteristike problema odvoje od realizacije rešenja tog problema.



Slika 1.1 Odnos programskih jezika, paradigma, i koncepta. [Izvor: Autor]

## KONCEPTI PROGRAMIRANJA

*Koncept programskog jezika je način da se oblikuje apstrakcija tako da odgovara određenom domenu problema.*

Koncept programiranja predstavlja način da se oblikuje apstrakcija tako da odgovara određenom domenu problema.

**Koncepti programiranja se dele na:**

- Imperativno programiranje
- Deklarativno programiranje.

Imperativno programiranje se dalje deli na **proceduralno** i **objektno-orientisano**, dok se deklarativno deli na **logičko** i **funkcionalno**.

**Podjela programskih jezika prema oblastima primene:**

- Jezici za naučne aplikacije
- Jezici za poslovne aplikacije
- Jezici veštačke inteligencije
- Jezici za razvoj sistemskog softvera
- Jezici za računarske komunikacije
- Jezici specijalne namene

[https://en.wikipedia.org/wiki/List\\_of\\_programming\\_languages\\_by\\_type](https://en.wikipedia.org/wiki/List_of_programming_languages_by_type)

**Karakteristike programskega jezika uključuju:**

- Formalno definisanu sintaksu programskog jezika
- Jake tipove podataka

- Strukturne tipove podataka
- Upravljačke strukture
- Potprograme
- Module
- Mehanizme za konkurentno programiranje
- Mehanizme niskog nivoa
- Mehanizme za obradu grešaka
- Standardni skup I/O procedura

**Faze u razvoju programskega jezika su:**

- Mašinski kod
- Heksadecimalni zapis mašinskog koda
- Asemblerski jezici
- Makroassemblerijski jezici
- Viši programske jezici (algoritamski ili proceduralni, strukturni jezici)
- Problemu orijentisani programske jezici

## ▼ Poglavlje 2

# Imperativno programiranje

## DEFINICIJA IMPERATIVNOG PROGRAMIRANJA

*Imperativno programiranje je programska paradigma koja opisuje računanje kao izraze koji menjaju stanje programa.*

Imperativno programiranje (en. imperative programming) je koncept programiranja koji opisuje process računanja uz pomoć sekvene izjava kojima se menja stanje programa.

*Kod imperativnog programiranja, program čini niz instrukcija računara.*

Osnovni alat koji omogućava ovaj način programiranja je uslovni skok. Imperativno programiranje često koristi dijagram toka (en. flow diagram).

Imperativno programiranje opisuje računarski proces kao stanja programa i naredbi koje menjaju stanje. Stanje računarskog sistema je definisano sadržajem memorije i naredbama mašinskog jezika.

*Programi predstavljaju skup naredbi koje računar treba da izvrši.*

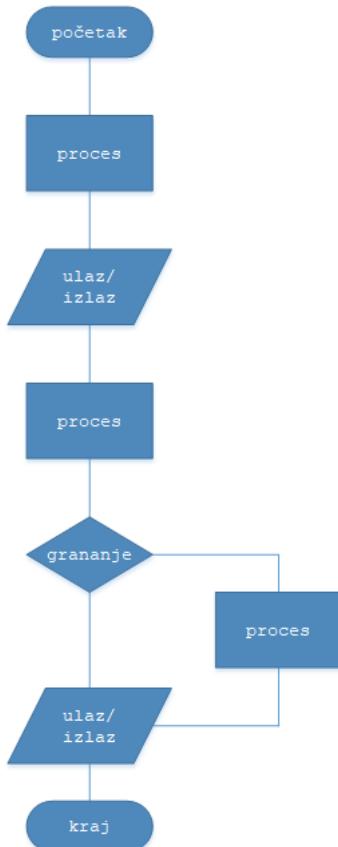
Hardverske implementacije računara su većinom imperativne, to znači da je hardver napravljen da izvršava mašinski jezik napisan u imperativnom stilu.

***Imperativno programiranje se deli na proceduralno i objektno-orientisano kao dve najznačajnije paradigme.***

Programski jezici koji podržavaju imperativnu programiranje jesu BASIC, C/C++, C#, COBOL, Java, JavaScript, FORTRAN, Python, Ruby, Rust, Perl, i mnogi drugi.

<https://en.wikipedia.org/wiki/>

[List\\_of\\_programming\\_languages\\_by\\_type#Imperative\\_languages](#)



Slika 2.1 Primer dijagrama toka. [Izvor: Autor]

## DEFINICIJA PROCEDURALNOG PROGRAMIRANJA

*Proceduralno programiranje je lista ili skup instrukcija koje definišu računaru šta treba uraditi metodom korak po korak i kako se to obavlja od prvog koda na drugom koda.*

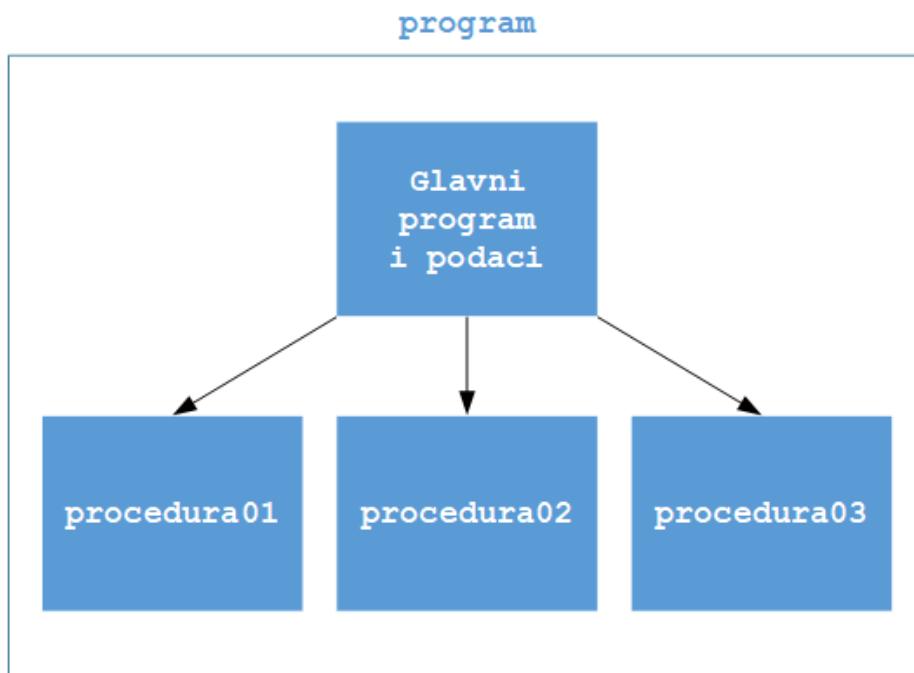
Proceduralno programiranje (en. **procedural programming**) je programska paradigma koja se zasniva na procedurama i njihovom izvršavanju, i to na principu *pozivanja procedure*.

Procedure se sastoje od sekvence računarskih koraka koji se mogu izvršavati pozivom u bilo kojoj tački izvršavanja celog programa, uključujući slučaj izvršavanja i u drugim procedurama.

*Sam program time postaje niz poziva procedura, i koda koji povezuje te pozive.*

Jezici koji podržavaju proceduralnu paradigmu su C/C++, Java, Fortran, Pascal, BASIC, i mnogi drugi.

[https://en.wikipedia.org/wiki/Category:Procedural\\_programming\\_languages](https://en.wikipedia.org/wiki/Category:Procedural_programming_languages)



Slika 2.2 Proceduralno programiranje. [Izvor: Autor]

## DEFINICIJA OBJEKTNO-ORIJENTISANOG PROGRAMIRANJA

*Objektno-orientisano programiranje je paradigma programiranja, koja koristi objekte kao osnovu za projektovanje računarskih programa i različitih aplikacija softvera.*

Objektno-orientisano programiranje (en. **object-oriented programming**, OOP) je koncept programiranja koji koristi **objekte** tj. instance **klasa**.

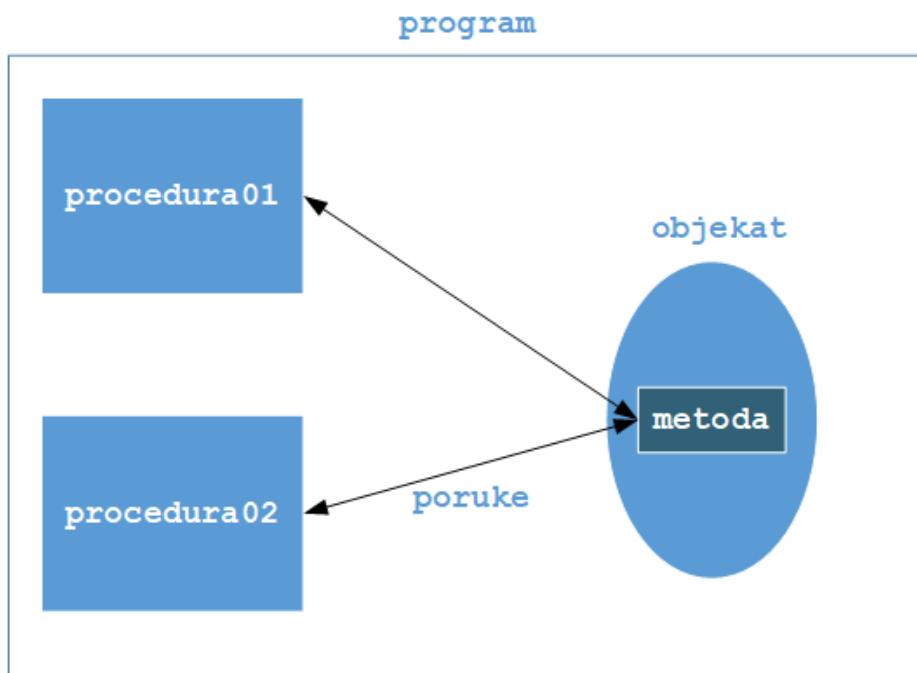
Klase se sastoje od polja podataka i metoda zajedno sa njihovom interakcijom i služe za kreiranje aplikacija i računarskih programa.

Tehnike programiranja uključuju karakteristike kao što su:

- Abstrakcija podataka (en. **data abstraction**)
- Enkapsulacija (en. **encapsulation**)
- Razmena poruka (en. **messaging**)
- Modularnost (en. **modularity**)
- Polimorfizam (en. **polymorphism**)
- Nasleđivanje (en. **inheritance**)

Najbitniji jezici koji podržavaju OOP jesu Java, C++, C#, Python, R, PHP, Visual Basic.NET, JavaScript, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp, MATLAB i mnogi drugi.

[https://en.wikipedia.org/wiki/Category:Object-oriented\\_programming\\_languages](https://en.wikipedia.org/wiki/Category:Object-oriented_programming_languages)



Slika 2.3 Objektno-orientisano programiranje. [Izvor: Autor]

## ✓ Poglavlje 3

# Deklarativno programiranje

## DEFINICIJA DEKLARATIVNOG PROGRAMIRANJA

*Deklarativno programiranje je programska paradigma koja opisuje računanje bez opisivanja kontrole toka.*

Deklarativno programiranje (en. **declarative programming**) je paradigma programiranja, stil izgradnje struktura i elemenata računarskih programa, koji izražavaju logiku računanja bez opisivanja njenog kontrolnog toka.

Mnogi jezici primenjuju ovu paradigmu tako što opisuju **šta program treba da postigne** (u smislu domena problema), umesto da opisuje kako ostvaruje niz koraka ka rešavanju problema.

*Ovo je u suprotnosti sa imperativnim programiranjem, u kojoj se algoritmi sprovode u smislu eksplicitnih koraka.*

Deklarativno programiranje može u velikoj meri pojednostaviti pisanje paralelnih programa.

Deklarativni jezici često uključuju jezika upita sistema kao što su SQL, XQuery, ali i Prolog, LISP, Miranda, Haskell.

### Razlika između imperativnog i deklarativnog programiranja

Prednost deklarativnog programiranja jeste mogućnost da kraće opiše problem u odnosu na imperativno programiranje.

Sledi primer upita za dobijanje ličnih imena u PHP dat je kroz obe paradigme

#### Imperativno programiranje

```
$participantlist = [1 => 'Peter', 2 => 'Henry', 3 => 'Sarah'];
$firstnames= [];
foreach ($participantlist as $id => $name) {
    $firstnames[] = $name;
}
```

#### Deklarativno programiranje

```
$firstnames = array_values($participantlist);
```

# LOGIČKO I FUNKCIONALNO PROGRAMIRANJE

*Logičko programiranje se zasniva na formalnoj logici. Funkcionalno programiranje tretira program kao skup funkcija i njihove primene za dobijanje rezultata.*

## Logičko programiranje

Logičko programiranje (en. **logic programming**) predstavlja programsku paradigmu koja se dosta zasniva na formalnog logici. Svaki program napisan na jeziku logičkog programiranja predstavlja skup izraza u logičkoj formi koji daju činjenice i pravila za programski domen.

Najpopularniji logički programske jezike jesu Absys, CHIP, HiLog, PROLOG i drugi.

Primer logičkog programiranja u Prolog jeziku:

```
mortal(X) :- man(X).  
man(socrates).  
  
?- mortal(socrates).  
true.
```

[https://en.wikipedia.org/wiki/Category:Logic\\_programming\\_languages](https://en.wikipedia.org/wiki/Category:Logic_programming_languages)

## Funkcionalno programiranje

Funkcionalno programiranje (en. **functional programming**) tretira program kao skup funkcija i njihove primene za dobijanje rezultata. Definicije funkcija predstavljaju stabla izraza (en. *trees of expressions*), od koje svaki može vratiti povratnu vrednost.

Najpopularniji funkcionalni programske jezike jesu Lisp, Wolfram Mathematica, Haskell, ali jezici kao što su C++, Perl, PHP, Python, Scala i SQL koriste elemente funkcionalnog programiranja.

<https://maryrosecook.com/blog/post/a-practical-introduction-to-functional-programming>

Primer funkcionalnog programiranja u Wolfram Mathematica:

```
Clear[f];  
f[x_] := 1 /; x < 1  
f[x_] := x * f[x - 1]
```

# STRUKTURNO I DINAMIČKO PROGRAMIRANJE

*Strukturno programiranje sa svojim konstrukcijama omogućava da programer uvede i određeni red (hijerarhiju) unutar programa, i tako omogućava lakše održavanje i pisanje.*

## Strukturno programiranje

Strukturno programiranje (en. **structured programming**) je koncept programiranja s ciljem da se poboljša jasnoća, kvalitet i vreme razvoja računarskog programa uz često korišćenje podprograma, blokova i "for" i "while" petlji umesto korišćenja jednostavnih testova i skokova u programu kao što su "go to" naredbe.

Procedura koja se takođe naziva i rutina (en. **routine**), potprogram (en. **subroutines**), metodom, ili funkcija (koju ne treba mešati sa matematičkim funkcijama koje se koriste u funkcionalnom programiranju), je serija koraka koje zahtevaju operacije računanja.

Bilo koja procedura može biti pozvana iz bilo kojeg dela programa u toku njegovog izvršavanja, pozivom druge procedure i pozivom samog sebe. Ovde se uvodi ideja poziva procedure sa definisanjem lokalnih varijabli u ovim procedurama. Time se ograničava sposobnost drugih procedura da pristupe lokalnim varijablama.

## Dinamičko programiranje

Dinamičko programiranje (en. **dynamic programming**) je termin koji se koristi za opisivanje klasa programskih jezika visokog nivoa (en. **high-level programming languages**) koji u toku svog izvršavanja (en. **runtime**) ispoljavaju mnogo osobina koje drugi jezici mogu obavljati u toku svog prevodenja ili kompilacije.

Ovo ponašanja može uključivati proširenje datog programa, dodavanje novog koda, širenje sadržaja objekata i definicije, ili menjanjem sistema tipiziranja i to sve u vreme izvršavanja programa.

Ovo se ponašanja može emulirati u bilo kojem jeziku dovoljne kompleksnosti, ali dinamički jezici pružaju direktnе alate za neposredno korišćenje.

Dinamičko programiranje se može koristiti za rešavanje složenih problema tako da se razbiju na jednostavnije potprobleme. To se odnosi na probleme koji pokazuju svojstva preklapanja podproblema (en. **overlapping subproblems**) i optimalno podstrukturu (en. **optimal substructure**).

Pojam optimalna podstruktura označava traženje optimalnih rešenja potproblema i njihovo korišćenje u cilju traženja optimalnog rešenja celokupnog problema.

## ▼ Poglavlje 4

# Pojam skripting jezika

## UVOD U SKRIPTING JEZIKE

*Jednostavan program se pomoću skripting jezika obično može brže napisati.*

### Definicija

Skripting jezik (en. *scripting language*) je programski jezik čiji se kod najčešće izvršava interpretiranjem. Pored toga, skriptni jezici se koriste najčešće za pisanje manjih programa (*skripti*) koji se brzo pišu i služe za obavljanje manjih poslova.

Skripting jezici su postali popularni u web programiranju, zbog svoje prenosivosti i jednostavnosti u pisanju - u njima najčešće ne postoje napredne mogućnosti koje imaju drugi programski jezici, poput pokazivača, direktnog pristupa memoriji, sistemskih funkcija i sl.

Okruženja koje se mogu automatizovati uz pomoć skripti uključuju softverske aplikacije, web stranice unutar web pretraživača, ljudske operacionih sistema i nekoliko programskih jezika opšte namene.

Skripta se može biti napisati i odmah izvršiti, tj. "*on-the-fly*", bez eksplisitnih koraka prevođenje (en. *compile*) i povezivanja (en. *link*). Pojam *skripta* obično je rezervisan za male programe (do nekoliko hiljada linija koda).

Scripting jezici se razlikuju od (standardnih) programskim jezika po tome što su napravljeni za "lepljenje" aplikacija zajedno. Oni koriste tzv. *typeless* pristup kako bi se postigao veći nivo programiranja i brži razvoj aplikacija u odnosu na programske jezike.

Početkom 21. veka skripting jezik bio je promatran kao pomoćni alat koji nije bio pogodan za opšte programiranje. Danas je teško izbeći korišćenje skripting jezika, jer je sadržan gotovo u svakoj web stranici a takođe je i deo većine softverskih okruženja.

*Korišćenje skripting jezika lakše je nego korišćenje (standardnog) programskog jezika, jer omogućava lakši uvid u greške.*

Zbog činjenice da je Internet postao dostupan velikom broju ljudi nije neobično da se programiranjem bave i korisnici kojima to nije primarno zanimanje. Većina njih se odluči koristiti skriptni jezik jer ga je lakše i brže naučiti u poređenju sa programskim jezikom.

## PREDNOSTI I MANE SKRIPTING JEZIKA

*Jedna od velikih prednosti skripting jezika u odnosu na (standardni) programski jezik je nizak stepen tipiziranja.*

Jedna od velikih *prednosti* većine skripting jezika u odnosu na (standardni) programski jezik je slab stepen tipiziranja (en. **weak typing**), dok (standardni) programski jezik ima jak stepen tipiziranja (en. **strong typing**). To ga čini mnogo lakšim za korišćenje.

Programerima je nekad teško da se nose sa jakim stepenom tipiziranja zbog toga što program može sadržavati na hiljade promenljivih, a kod jakog stepena tipiziranja svaka promenljiva mora biti unapred deklarisana tipom promenljive kako bi program zauzeo određenu memoriju i kako bi podaci koje koristi bili prepoznatljivi.

Snaga slabog stepena tipiziranja leži u tome što se promenljiva ne mora unapred deklarisati odnosno navoditi tip promenljive, smanjujući nastale greške, pa samim tim programer štedi na vremenu potrebnom za pisanje programa.

Glavni *nedostatak* skripting jezika je da se izvršni kod skripte može i slučajno preuzeti s udaljenog servera na računaru web pretraživača, i nakon toga instalirati i pokrenuti pomoću interpretora lokalnog pretraživača.

To se može lako dogoditi (i najčešće i događa) posetama sumnjivim web stranicama ili preuzimanjem programa bez autentifikacije, što predstavlja ozbiljan sigurnosni problem.

Skripting jezici su od druge decenije 21. veka postali napredniji i bolji od (standardnih) programskih jezika skoro u svim područjima. Skripting jezici izbegavaju upravljanje ručno memorijom, a to uspevaju zahvaljujući memoriskom upravljačkom sistemu u delu za izvršavanje (en. **runtime**), tj. programskom interpretatoru.

*Više programa može deliti jedan interpretator, čime se smanjuje potrošnja memorije.*

U savremenom pristupu razvoja softvera korisno je i smisleno pisati programe na nekoliko jezika, nakon čega se može odabrati najbolji jezik za određene pod-zadatke.

Na primer, delovi koji zavise od vremena i vremenski su kritični, mogu se napisati u C-u, pristup podacima se može odraditi preko SQL-a, pa se sve zajedno može spojiti u jednu celinu. Konačno korisnički interfejs može se napisati Python jeziku.

## OSOBINE SKRIPTING JEZIKA

*Skripting jezik se interpretira, ili interpretira instrukcije, on se ne kompajlira u izvorni kod*

### **Glavne osobine skripting jezika**

- Skripting jezik se interpretira, ili interpretira instrukcije, on se *ne kompajlira* u izvorni kod,
- O upravljanju memorijom brine sakupljač smeća (en. **garbage collector**), pa sam programer ne treba da brine o memoriji,

- Skripting jezik uključuje tipove podataka viših nivoa, kao što su liste, asocijativna polja, itd.
- Okvir izvršenja skripting programskog jezika može biti integriran u program koji je već napisan,
- Skript program može pristupiti modulima napisanim u nižem programskom jeziku (na primer C),
- Koriste se obično za administraciju sistema i brzu izradu prototipova (engl. **rapid prototyping**),
- Dozvoljava grupisanje tipično korišćenih komandi u batch datoteku za procesiranje,
- Dozvoljava kreiranje novih fleksibilnih i konfigurabilnih alata koji „razumeju“ skripte i alate drugih korisnika
- Neformalan način što se tiče tipiziranja promenljivih (nema razlike između tipova kao što su integer, float ili string)
- Funkcije mogu vratiti ne-skalarne vrednosti, kao i nizove
- Ponovljeni zadatak može se obaviti mnogo brže
- Skripte izbegavaju manje greške nedoslednosti grešaka u procesiranju

U savremenom pristupu razvoja softvera korisno je i smisleno pisati programe na nekoliko jezika, nakon čega se može odabratи najbolji jezik za određene pod-zadatke.

Na primer, delovi koji zavise od vremena i vremenski su kritični, mogu se napisati u C-u, pristup podacima se može odraditi preko SQL-a, pa se sve zajedno može spojiti u jednu celinu. Konačno korisnički interfejs može se napisati Python jeziku.

Skripting jezik ili skript jezik je programski jezik koji podržava skripte, programe pisane za posebna okruženja za izvršavanja (en. **runtime environment**) koja mogu interpretirati (*još jednom - ne prevoditi*) i automatski izvršavati zadatke koje bi alternativno bili izvršavani jedan po jedan od strane operatera.

## KADA (NE)KORISTITI SKRIPTING JEZIKE?

*Navedeni su slučajevi kada treba, a kada ne koristiti skripting jezike.*

### **Treba koristiti skripting jezike kada:**

- Se vrši spajanje postojećih programskih komponenti,
- Su česte promene u aplikaciji,
- Je prisutan grafički korisnički interfejs,
- Se funkcije aplikacije često menjaju,
- Je aplikacija proširiva,
- Aplikacija manipuliše stringovima.

### **Ne treba koristiti skripting jezike kada:**

- Postoje kompleksni algoritmi i strukture podataka,
- Se procesiraju velike količine podataka,
- Su funkcije strogo definisane i stalne.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## JAKO I DINAMIČKO TIPIZIRANJE PODATAKA

*Skripting jezik, Python koristi dinamičko tipiziranje (en. dynamic typing) i jako tipiziranje (en. strong typing).*

Podsetnik: Tip podataka (en. data type) je klasifikacija koja identificuje jednu vrstu podataka, kao što je realni broj (en. **real number**, **floating point number**), ceo broj (en. **integer**), ili Boolean, koji određuje moguće vrednosti za taj tip, i operacije koje se mogu obaviti na vrednosti tog tipa.

Osnovni tipovi podataka u većini programskih jezika mogu biti:

- Boolean
- Integer
- Floating-point number
- Character
- Alphanumeric string

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

Skripting jezik Python koristi dinamičko tipiziranje (en. **dynamic typing**) i jako tipiziranje (en. **strong typing**).

### **Jako tipiziranje**

Kod ovoga metoda, promenljive imaju samo jednu vrednost. Programski jezik ne dozvoljava da operacija bude uspešna ako su tipovi pogrešni.

### **Dinamičko tipiziranje**

Kod dinamičkih tipiziranih programskih jezika ne treba promenljivu definisati pre nego što ona bude korišćena. Kod ove metode, promenljive mogu da imaju različite tipove podataka. Prevodilac zna koji je korektan tip podatka.

<https://stackoverflow.com/questions/11328920/is-python-strongly-typed>

## PYTHON TIPIZIRANJE: PRIMERI

*Kombinacija jakog i dinamičkog tipiziranja u Pythonu dovoljava promenljive vraćaju različite vrednosti o tipu.*

### **Primer #1:**

Promeniti tip podataka promenljive c, da bude ceo broj, string, i lista.

### Rešenje:

```
c = 1 # an integer
c = "a string" # a string
c = [a, b, c] # a list
```

*Kombinacija jakog i dinamičkog tipiziranja u Pythonu dovoljava promenljive vraćaju različite vrednosti o tipu.*

*Na taj način isto ime promenljiva može ukazivati na različite tipove podataka.*

### Objašnjenje koda:

U prvoj liniji koda promenljivoj `c` dodeljuje se vrednost 1.

Nakon toga, korišćenjem ključnog simbola `#` pišemo *komentar*. Interpreter će ignorisati sve što je nakon početka komentara.

U drugoj liniji koda, promenljivoj `c` dodelujemo vrednost `"a string"`, i upisujemo sledeći komentar.

Stringovi u Python jeziku se označavaju unutar navodnika `" "` ili apostrofa `' '`.

Konačno, promenljivoj `c` dodelujemo vrednosti liste (lista može sadržati više vrednosti) i komentarišemo.

Svi elementi liste se označavaju unutar srednjih zagrada `[]`, a elementi liste se odvajaju zapetama.

### Primer #2

Promenljivoj `bob` dodati vrednost 1, i ispitati kojeg je tipa promenljiva. Nakon toga, dodeliti vrednost `"bob"` i ponovo ispitati kojeg je tipa promenljiva.

### Rešenje:

```
bob = 1
type(bob)
bob = "bob"
type(bob)
```

### Objašnjenje koda:

Ubacivanje imena promenljive u komandu `type()` dobija se kojeg je tipa promenljiva.

## ❖ 4.1 Podela skripting jezika

### VRSTE SKRIPTING JEZIKA

*Postoje različite vrste skripting jezika za domene specifične za određena okruženja.*

Skripting jezik može se posmatrati kao jezik specifičnog domena za određeno okruženje; u slučaju skripting aplikacije, ovaj slučaj je poznat kao jezik proširenja (en. *extension language*).

Skripting jezici se takođe ponekad nazivaju i *programski jezici visokog nivoa*, jer rade na visokom nivou apstrakcije, ili kao *kontrolni jezici*, na primer jezik za upravljanje poslovima (en. job *control language*) za mainframe računare.

Postoji više vrsta skripting jezika a neki od skripting jezika *po nameni* su:

- Jezik za upravljanje poslovima (en. *job control languages*) i ljudske (en. *shell*),
- GUI skripting jezici,
- Specifični jezici za aplikacije,
- Jezici za procesiranje teksta,
- Dinamički jezici opšte namene,
- Ugrađeni jezici (en. *embeddable languages*).

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### JEZICI ZA UPRAVLJANJE POSLOVIMA I LJUSKE

*Jezici za upravljanje poslovima jesu skripting jezici koji se koriste za IBM Mainframe OS da uputi sistem kako da pokrene paketni posao ili podsistem.*

*Glavni razlog nastanka skripting jezika je automatizacija upravljanja poslovima što se odnosi na pokretanje i kontrolu ponašanja sistemskih programa.*

Job Control Language (JCL) je skripting jezik za IBM mainframe operativni sistem koji upućuje sistem kako da pokrene paketni posao (en. *batch job*) ili da pokrene podsistem.

Mnogi od interpretera tih jezika koriste se kao interpreteri komandne linije (en. *command-line interpreters*) kao što su UNIX shell ljudska ili MS-DOS *COMMAND.COM*.

Drugi skripting jezici kao što je AppleScript koriste komande slične engleskom jeziku za izgradnju skripti.

JCL je imao veliki uticaj na UNIX ljudske, *bash*, *csh*, *ksh*, *sh*, kao i na Apple *MPW*.

Korisnici ovih ljudi mogu izdavati komande sistemu za premeštanje datoteka, kompajliranje programa i sl.

**Primer:**

Korisnik UNIX OS može izdati ove naredbe:

```
% mv foo.c bar.c      #rename "foo.c" to "bar.c"
% cc -o bar bar.c    #compile "bar.c" into "bar"
% bar
```

U većini modernih računarskih sistema proces snimanja skripti korisničnih komandi je trivijalno. Ovaj se skript može menjati i ponovno koristiti.

**Primer:**

Tipični UNIX *shell script* za sortiranje:

```
T = /tmp/wl.$$  #temp file
wc -l $* > ST  #get line count for files
lpr $T $*        #print line counts, then files
rm $T            #remove temp file
```

Skript ljudske se mogu lakše pisati, razumeti i modifikovati. Mnogi od standardnih UNIX alata su pisani kao skript ljudske.

## GUI SKRIPTING JEZICI

*Od pojave grafičkih korisničkih interfejsa (GUI), pojavila se specijalizovana vrsta skripting jezika, GUI skripting jezik, za upravljanje računarima.*

GUI (en. **Graphical User Interface**) jezici su u interakciji za grafičkim elementima aplikacija. To mogu biti prozori (en. **graphic windows**), meniji, ali i tasteri i dugmad (na mišu ili tastaturi), itd.

Interakcija sa grafičkim korisničkim interfejsom

- Grafički prozori
- Meniji
- Dugmad

*GUI skripting jezici se tipično koriste za simulaciju i automatizaciju korisničkih komandi.*

Ovakva skripta se obično zove **makro** (en. **macro**). Kontrola makroa se sprovodi putem pritiskanja dugmića ili na akciju miša.

GUI skriptni jezici se teoretski mogu koristiti za upravljanje GUI aplikacijama, ali u praksi je njihovo korišćenje limitirano radi toga što zahtevaju podršku i aplikacije i operativnog sistema.

Međutim, postoje nekoliko izuzetaka. Neki GUI skripting jezici se baziraju na prepoznavanju grafičkih objekata na ekranu. Ovi GUI skripting jezici obično ne zavise od podrške operativnog sistema ili aplikacije.

**Primer:**

SAP GUI skripting

SAP GUI skripting je interfejs sa automatizacijom koji poboljšava mogućnosti SAP GUI platforme. Koristeći ovaj interfejs, krajnji korisnici mogu da automatizuju ponavljajuće zadatke snimanjem i izvršavanjem skripti koje izgledaju kao makroi.

Sa druge strane, administratori i programeri mogu da grade alate za testiranje aplikacija na strani servera ili integrisanih aplikacija na strani klijenta.

## PRIMER: GUI SKRIPTING U VIDEO IGRAMA

*GUI skripting se gotovo svuda koristi, pa i u video igramu.*

**Primer:**

U MMORPG igri World of Warcraft korišćenjem makroa mogu se automatizovati više komandi koje bi korisnici (igrači) inače moralo pojedinačno da pritisnu (bilo na tastaturi ili klikom miša).



Slika 4.1.1 Izgled makro menija u igri World of Warcraft. [Izvor: wowhead.com]

<https://www.wowhead.com/making-a-macro-commands-modifiers-warcraft-guide>

```
/cancelaura Ice Block  
/cast Ice Block
```



Slika 4.1.2 Primer korišćenja makro u igri World of Warcraft. [Izvor: wowhead.com]

## LEPLJIVI (GLUE) I UGRADIVI (EMBEDDABLE) SKRIPTING JEZICI

*Pojedini skripting jezici se tretiraju kao lepljivi (Glue) jezici.*

Pojedini skripting jezici se tretiraju kao lepljivi jezici (en. glue languages) , zbog toga što povezuju, tj. lepi softverske komponente.

Kod napisan u ove svrhe se često naziva lepljiv kod (en. glue code)

Najčešći primer lepljivog koda jeste povezivanje baze podataka sa serverom.

Gotovo svi skripting programski jezici (JavaScript, PHP, Python, Ruby) se mogu koristiti kao lepljivi jezici.

[https://en.wikipedia.org/wiki/Glue\\_code](https://en.wikipedia.org/wiki/Glue_code)

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

Ugradivi jezici (en.embeddable languages) jesu jezici koji se mogu lako ugraditi (en. embed) u drugi program.

Najčešće se koriste u razvoju video igara.

Najčešći jezici koji se u ove svrhe koriste jesu Lua, Python, Gravity.

<https://www.pc当地.com/encyclopedia/term/embedded-language>

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 5

# Specifični jezici za aplikacije

## SPECIFIČNI SKRIPTING JEZICI ZA KONKRETNE APLIKACIJE

*Mnogi veliki aplikacioni paketi programa uključuju posebne skripting jezike prilagođene potrebama korisnika aplikacija.*

***Aplikaciono specifični skripting jezik može da se posmatra kao programski jezik za domen specijalizovan za jednu aplikaciju.***

### Specifični skripting jezici za aplikacije

Mnogi računari su prilagođeni računarskim igrama i koriste posebno kreirane skripting jezike.

Jezici ove vrste su kreirani za jednu aplikaciju mada površno gledano liče na neke jezike opšte namene (primer: QuakeC kreiran po uzoru na C), ali oni imaju posebne karakteristike koje ih razlikuju od drugih jezika.

Emacs Lisp, potpuno formirani dijalekt Lisp jezika, sadrži mnoge posebne karakteristike koje ga čine veoma korisnim za radu i ažuriranju Emacs-a .

U ovakve jezike spadaju i *dinamički jezici opšte namene* - Neki jezici, kao što su Perl, počeli su kao skripting jezici, ali su brzo razvili u programske jezike pogodne za šire namene. Ostali slični jezici su opisani kao "skripting jezici" zbog tih sličnosti i pored toga što se češće koriste za programiranje aplikacija.

Web pretraživači, jezici za procesiranje teksta, dinamički jezici opšte namene, ekstenzije/ugrađeni jezici predstavljaju primere ovakvih skripting jezika.

### Primeri:

Web pretraživači su aplikacije za prikazivanje web stranica. Skripte mogu pokrenuti web pretraživače za promenu izgleda ili ponašanja web stranice, npr. mogu promeniti sadržaj web stranice za specifičnog korisnika. Host ovih jezika specijalne namene je razvijen da upravlja radom web pretraživača. To uključuje JavaScript i VBScript, koji radi samo u Internet Explorer okruženju. Xul firme Mozilla koji radi samo u Firefox okruženju. XSLT jezik za prezentacije pretvara XML sadržaj u nove formate.

Primer skripte klijenta je JavaScript upozorenje (en. *alert box*) koje iskače kada korisnik klikne negde na web stranici.

*Jezici za procesiranje teksta* isto predstavljaju ovakav jezik. Obrada tekstualnih zapisa je jedan od najstarijih korišćenja skripting jezika. Skripte pisane za UNIX alate *awk*, *sed*, i

*and* automatizuju zadatke koji uključuju tekstualne datoteke, na primer, konfiguracija i log datoteke.

## PRIMERI SPECIFIČNIH SKRIPTING JEZIKA - JAVASCRIPT, TLC, MEL

*JavaScript i Tcl su primeri aplikacijski specifičnih jezika.*

JavaScript se počeo primarno koristiti kao jezik za skripting unutar web pretraživača, međutim, standardizacija jezika kao ECMAScript ga je napravio toliko popularnim da je postao jezik za ugradnju opšte namene.

Konkretno, Mozilla implementacija SpiderMonkey je ugrađen u nekoliko okruženja, kao što su Yahoo! Widget Engine. Ostali programi sa ugrađenim ECMAScript uključuju Adobe proizvode Adobe Flash (ActionScript) i Adobe Acrobat.

Tcl (en. **T**ool **C**ommand **L**anguage) je nastao kao proširenje jezika, ali se počeo više koristiti kao jezik opšte namene u ulogama sličnim jezicima Python, Perl i Ruby. S druge strane, REXX izvorno nastao kao jezik kontrole poslova (engl. job control language) se često koristi i kao proširenje jezika i kao jezik opšte namene. Tcl uključuje web i desktop aplikacije, umrežavanje, administraciju i ispitivanje.

Program Autodesk Maya 3D autorski alat koristi ugrađeni jezik MEL skripting jezik, Blender koristi Python za ovu ulogu. Neke druge vrste aplikacija koje trebaju veću brzinu (primer predstavlja mehanizam igrara) takođe koristi ugrađene jezike.

U toku razvoja, to im omogućava da brže razviju prototip bez potrebe da korisnik poznaje unutrašnju strukturu aplikacije. Skripting jezici koji se koriste za ove namene su u rasponu od poznatijih Lua i Python do manje poznatih poput AngelScript i Squirrel. Ch je još jedna C kompatibilnih skripting opcija za industriju za ugradnju u C/C++ aplikacione programe.

## ▼ Poglavlje 6

### Pokazna vežba #1

#### UVOD U POKAZNU VEŽBU #1

*Python 3 se lako instalira u sistemsku pitanju, i još lakše koristi.*

U pokaznoj vežbi izvršiće se instalacija Python 3 programskega jezika na računar sa Windows operativnim sistemom. Nakon instalacije, biće dostupna komandna linija Python 3 jezika.

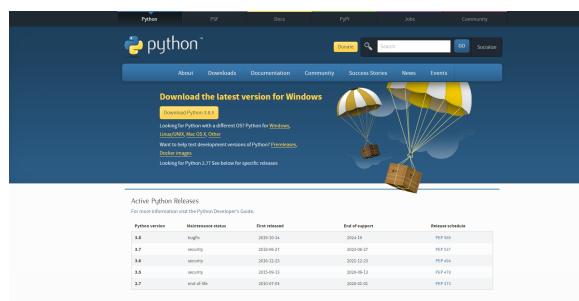
Ovo predstavlja osnovnu instalaciju, dok će u kasnijim lekcijama biti obrađeni konkretno razvojno okruženje za razvoj aplikacija u Python 3 programskom jeziku.

Procenjeno trajanje pokazne vežbe iznosi 45 minuta.

#### INSTALACIJA PYTHON 3 SKRIPTING JEZIKA

*Python 3 se lako instalira, ali treba obratiti pažnju da se ubaci u sistemsku putanju.*

Pokazna vežba #1 počinje tako što se poseti sajt <https://www.python.org/downloads/> i preuzme najnovija (3.x) verzija programa.



Slika 6.1 Python 3 stranica za preuzimanje. [Izvor: python.com]

Nakon preuzimanja, pokrenuti instalaciju.

Slika 6.2 Smeštanje instalacione datoteke. [Izvor: Autor]

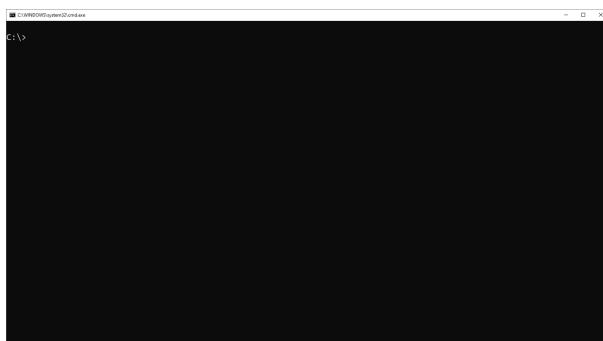
Treba obratiti pažnju na **Add Python 3 to PATH** koje treba štiklirati. Ostala podešavanja mogu ostati kako jesu. Nakon toga, može se kliknuti na **Install Now**, ili ukoliko želite podesiti dodatne opcije, onda na **Customize installation**

Slika 6.3 Instalacija Python-a na Windows Operativni sistem. [Izvor: Autor]

## POKRETANJE PYTHON 3 JEZIKU PREKO KOMANDNE LINIJE

*Osnovne operacije se mogu direktno kucati u prompt-u, ali i sam kod.*

Kada se Python instalira, može se direktno pokrenuti iz komandne linije pokretanjem **cmd** komande kroz **run**.



Slika 6.4 Komandni prozor pre pokretanja Python. [Izvor: Autor]

Komandom **python** iz prompt-a DOS prozora prelazi se na Python prompt.

Slika 6.5 Komandni prozor posle pokretanja Python-a. [Izvor: Autor]

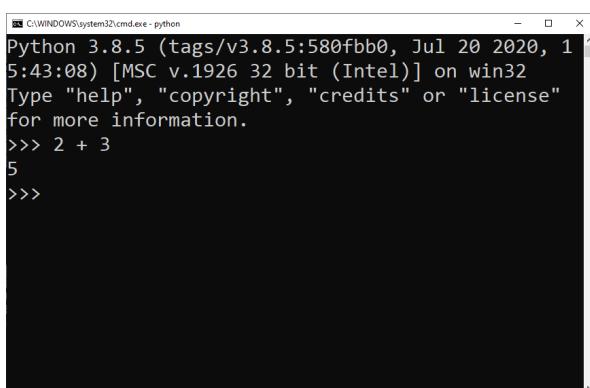
Odmah prilikom pokretanja može se videti koja je i trenutna verzija Python jezika, a i prepoznaje se prompt koji je (po default-u) tri znaka veće: **>>>**

Sada se može direktno pisati u Python prozoru. Kako je Python skripting jezik koji se interpretira, mogu se i direktno ubaciti komande koje će se izvršiti pritiskom na Enter.

### Primer #1 - Sabiranje (5 minuta)

Ukucati sledeće direktno u liniju

2 + 3

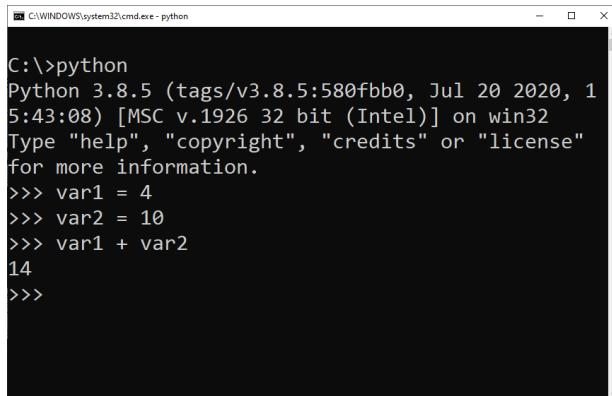


Slika 6.6 Direktno izvršavanje komandi prompt-u. [Izvor: Autor]

### Primer #2 - Dodela vrednosti promenljivima (7 minuta)

Direktno u komandnoj liniji se mogu definisati promenljive i dodeliti im vrednosti.

```
var1 = 4
var2 = 10
var1 + var2
```



Slika 6.7 Sabiranje preko promenljivih. [Izvor: Autor]

## POKRETANJE PROGRAMA PISANOG U PYTHON JEZIKU PREKO KOMANDNE LINIJE

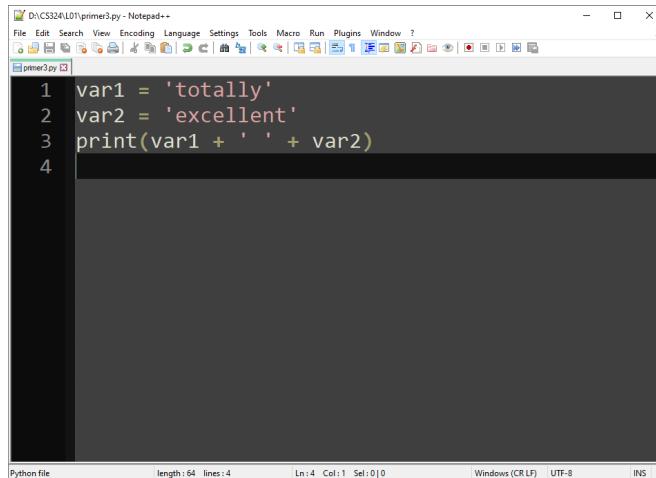
*Program koji je već napisan može se direktno pozvati na izvršenje preko komandne linije.*

Program napisan u Python jeziku (sa ekstenzijom .py), možemo takođe pokrenuti direktno iz komandne linije.

### Primer #3 - Spajanje stringova (10 minuta)

Sledeći program možemo napisati u bilo kom tekstualnom editoru i sačuvati kao .py datoteku.

```
var1 = 'totally'
var2 = 'excellent'
print(var1 + ' ' + var2)
```



```
1 var1 = 'totally'
2 var2 = 'excellent'
3 print(var1 + ' ' + var2)
4
```

Slika 6.8 Program napisan u eksternom editoru teksta. [Izvor: Autor]

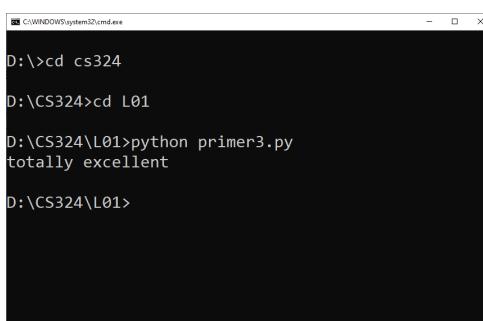
Nakon toga, možemo direktno iz komandnog prozora (bez ulaženja u Python) pokrenuti program tako što ćemo doći do putanje gde se nalazi program (ili ukucati putanju), i pre pokretanja samog programa dodati reč python.

Prethodni program sačuvan je kao primer3.py, i kada smo došli do direktorijuma gde se nalazi, smo treba ukucati sledeće:

```
python primer3.py
```

Alternativno, možemo ukucati i celu putanju relativno od direktorijuma gde se nalazimo:

```
D:\>python CS324\L01\primer3.py
```



```
D:\>cd cs324
D:\CS324>cd L01
D:\CS324\>python primer3.py
totally excellent
D:\CS324\>
```

Slika 6.9 Pozivanje napisanog programa direktno iz komandne linije. [Izvor: Autor]

## PISANJE PRVOG PROGRAMA U PYTHON 3 JEZIKU - HELLO WORLD!

*Vrlo jednostavnii programi se mogu napisati direktno iz komandne linije, ali i pokrenuti eksterno.*

### Primer #4 - Hello world! (5 minuta)

Napisati program koji ispisuje tekst Hello World! preko komandne linije.

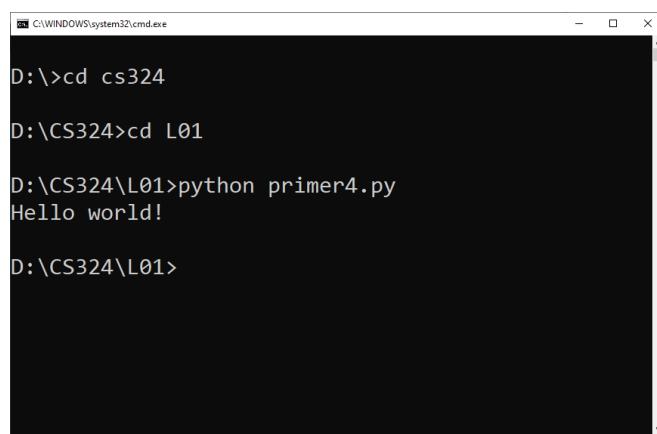
Zatim, isti program napisati preko eksternog tekstu editora, sačuvati kao primer4.py, i pozvati na izvršenje direktno iz komande linije.

**Rešenje za program:**

```
print('Hello world!')
```

**Rešenje za izvršenje preko komandne linije:**

```
python primer4.py
```



```
C:\WINDOWS\system32\cmd.exe
D:\>cd cs324
D:\CS324>cd L01
D:\CS324\L01>python primer4.py
Hello world!
D:\CS324\L01>
```

Slika 6.10 Rešenje za primer4.py Izvor: Autor

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 7

### Individualna vežba #1

#### UVOD U INDIVIDUALNU VEŽBU #1

*Python 3 se lako instalira u sistemsku pitanju, i još lakše koristi.*

U individualnoj vežbi proći će se kroz najosnovnije tipove programa, koji se odmah mogu pisati u *prompt* liniji Python-a, ili se mogu napisati u eksternom editoru, pa se pokrenuti iz komandne linije.

Individualne vežbe će se dotaći sličnosti i razlika Python programskog jezika, ali i ostalih skripting jezika (JavaScript, Perl, Ruby) kako međusobno, tako i u odnosu na programske jezike nižeg nivoa.

Procenjeno trajanje individualnih vežbi iznosi 90 minuta

#### INDIVIDUALNA VEŽBA #1 - RAZLIKE I SLIČNOSTI U PROGRAMSKIM JEZICIMA

*Kroz jednostavne zadatke uočiti sličnosti i razlike između pojedinih programskih jezika*

Individualna vežba #1 prolazi kroz osnove sintakse u različitim (standardnim i skripting) programskim jezicima.

Na osnovu toga treba zaključiti koje su sličnosti, a koje su razlike pojedinih skripting jezika u odnosu na druge programske jezike nižeg nivoa.

##### ***Uvod u individualnu vežbu #1***

Napisati program koristeći neke od sledećih jezika (barem tri):

- C
- C++
- Java
- JavaScript
- Ruby
- Perl
- Python

Bez instaliranja dodatnih okruženja za razvoj softvera, koristiti <https://www.tutorialspoint.com/codingground.htm>

**Zadatak #1** (3x 5 minuta)

Napisati "Hello World" program, odnosno program koji će kao izlaz štampati tekst *Hello World* na ekranu.

**Zadatak #2** (3x 10 minuta)

Napisati program kod kojeg unosite proizvoljnu vrednost u promenljivu *var*, i kao izlaz ispisuje koji je tip podataka.

**Zadatak #3** (3x 15 minuta)

Napisati funkciju koja izračunava i štampa vrednosti *n* članova Fibonačijevog niza. Funkciju napisati u rekurzivnoj i iterativnoj formi. Vrednost broja Fibonačijevog niza računa se po sledećoj formuli:

$$F_n = F_{n-1} + F_{n-2}$$

## ▼ Poglavlje 8

### Domaći zadatak #1

#### DOMAĆI ZADATAK

*Domaći zadatak #1 se okvirno radi 1.5h*

##### **Domaći zadatak #1**

Na kućnom računaru instalirati Python 3, i ubaciti u sistemsku putanju.

Instalacija se može naći na:

<https://www.python.org/downloads/>

Screenshot-ovati komandnu liniju gde se vidi da je Python instaliran.

Zatim, koristeći text editor (Notepad++, online editor i sl.) napisati program u Python-u koji određuje da je uneti broj n stepen broja 2.

Napisati isti program u nekom nižem programskom jeziku (C/C++/Java) i opisati sličnosti i razlike.

Možete koristiti **online compiler** (<https://www.tutorialspoint.com/codingground.htm>) za pisanje na nižem programskom jeziku.

*Koristiti uputstvo za izradu domaćeg zadatka.*

##### **Predaja domaćeg zadatka:**

##### **Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije nedelju dana nakon predavanja za 100% poena. Nakon toga poeni se umanjuju za 50%.

##### **Online studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganja ispita. **Domaći zadaci se brane!**

Svi studenti domaći zadatak poslati dr Nemanji Zdravkoviću:  
[nemanja.zdravkovic@metropolitan.ac.rs](mailto:nemanja.zdravkovic@metropolitan.ac.rs)

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Zaključak lekcije #1*

#### **Rezime:**

U ovoj lekciji objašnjena su najpre pravila vezana za predmet i načini ocenjivanja.

Zatim, posle podsetnika o teoriji programskog jezika, uveden je pojam skripting programskog jezika.

Opisane su razlike i sličnosti skripting programskog jezika i standardnih, tj. programskih jezika nižeg nivoa.

Opisane su situacija kada treba koristiti skripting programske jezike, a kada ne, kao i sama podela skripting jezika prema nameni.

U vežbi opisan je proces instalacije Python 3 programskog jezika u putanju na Windows operativnom sistemu, i nekoliko osnovnih vrsta programa je predstavljeno kroz primere.

#### **Literatura:**

1. David Beazley, Brian Jones, *Python Cookbook: Recipes for Mastering Python 3*, 3rd edition, O'Reilly Press, 2013.
2. Mark Lutz, *Learning Python*, 5th Edition, O'Reilly Press, 2013.
3. Andrew Bird, Lau Cher Han, et. al, *The Python Workshop*, Packt Publishing, 2019.
4. Al Sweigart, *Automate the boring stuff with Python*, 2nd Edition, No Starch Press, 2020.





## CS324 - SKRIPTING JEZICI

### Uvod u Python 3

Lekcija 02

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 02

### *UVOD U PYTHON 3*

- ✓ Uvod u Python 3
- ✓ Poglavlje 1: Uvod u Python 3
- ✓ Poglavlje 2: Istorija Python programskog jezika
- ✓ Poglavlje 3: Razlika između verzija Python programskog jezika
- ✓ Poglavlje 4: Primene Python programskog jezika
- ✓ Poglavlje 5: Python 3 razvojno okruženje (IDE)
- ✓ Poglavlje 6: Pokazna vežba #2
- ✓ Poglavlje 7: Individualna vežba #2
- ✓ Poglavlje 8: Domaći zadatak #2
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ✓ Uvod

### UVOD

*U drugoj lekciji govorićemo o Python programskom jeziku kao jednom od najpopularnijih skripting jeziku današnjice.*

U drugoj lekciji govorice se o Python programskom jeziku kao jednom od najpopularnijih skripting jeziku današnjice. Upravo zbog toga se samo Python obraduje na ovom predmetu.

O popularnosti Python jezika govori činjenica da su sajtovi poput **Google**-a i **reddit**-a pisani u Python-u, a ali i **Netflix**, **Spotify**, **Pinterest**, **Uber** i **Lyft**, kao i **Instagram**.

*Python se trenutno koristi u trećoj verziji, dok je prethodna verzija, python 2, i dalje zastupljena.*

Najbitnije razlike jesu te da Python 3 ima jednostavniju sintaksu. Srećom, sintaksa je dovoljno slična i kod se može lako protumačiti.

O drugim razlikama između verzija Python-a biće reči u nastavku lekcije.

Rečeno je da se Python, kao i svi skripting jezici, primenjuje najviše u razvoju web aplikacija, ali to nije jedina primena.

Pored Web-a, Python se dosta koristi i u Data Science, u algoritmima veštačke inteligencije, prvenstveno u mašinskom učenju, u NLP-u, u razvoju video igara, ali i u inženjerskim i naučnim primenama.

Na kraju lekcije biće reči kako o najčešće korišćenim okruženjima za razvoj aplikacija u Python verzije 3.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 1

### Uvod u Python 3

## UVOD U PYTHON 3 PROGRAMSKI JEZIK

*Primenom Python programskog jezika, programer se fokusira na rešavanje problema, a ne na sintaksu.*

Python programski jezik je programski jezik otvorenog koda (en.[open-source](#)) koji se najviše koristi u [web programiranju](#), u [data science](#)-u, [veštačkoj inteligenciji](#), i u mnogim [naučnim primenama](#).

*Učenje Python programskog jezika omogućava programeru za se fokusira na rešavanje problema, a ne da se fokusira na samu sintaksu jezika.*

Jednostavna sintaksa daje Python programskom jeziku prednost u odnosu na druge jezike kao što su C++ ili Java, dok sa druge strane, pregršt biblioteka omogućava Python jeziku da rešava mnoge probleme.

#### Opis Python programskog jezika

*Python predstavlja interpretirani, objektno-orientisani programski jezik visokog nivoa sa dinamičkim sistemom tipiziranja podataka.*

Zbog osobina visokog nivoa koje su ugrađene u strukture podataka, zajedno sa dinamičkim sistemom tipiziranja i dinamičkim povezivanjem, veoma je pogodan za brz razvoj aplikacija (en. [Rapid Application Development](#)), za primene kao skripting jezik, ali i za primene kao lepljiv jezik koji povezuje postojeće komponente zajedno.

Sintaksa Python jezika koja je jednostavna i laka za korišćenje povećava samu čitljivost programa, i na taj način smanjuje cenu održavanja programa pisana u Python jeziku.

Python interpreter i bogata biblioteka standardnih funkcija dostupni su u izvornom kodu ili u binarnom obliku besplatno za sve (glavne) računarske platforme, i mogu se besplatno distribuirati.

## PRODUKTIVNOST U PYTHON PROGRAMSKOM JEZIKU

*Produktivnost Python programskog jezika ogleda se u brzom ažuriraj-testiraj-debuguj ciklusu*

Programeri često biraju, i ostaju sa Python programskim jezikom zbog povećanja produktivnosti koji Python pruža. Budući da ne postoji korak kompajliranja, ciklus ažuriraj-

testiraj-debaguj (en. [edit-test-debug](#)) je neverovatno brz, pogotovo u porečenju sa programskim jezicima koje sadrže kompjajler.

[Debagovanje](#) u Python programskom jeziku je lako: [bag](#) ili loš unos neće izazvati grešku segmentacije.

**Podsetnik:**

*Greška segmentacije dešava se kada program pokuša da pristupi memorijskoj lokaciji kojoj ne treba da pristupi, ili da pokuša da pristupi memorijskoj lokaciji na način na koji ne treba da pristupi (primer: da piše na read-only lokaciji, ili da upiše preko dela operativnog sistema).*

Kada Python interpreter pronađe grešku, podiće će izuzetak. Kada program ne uhvati izuzetak, interpreter štampa [trag steka](#) (en. [stack trace](#)).

Debager na izvornom nivou (en. [source level debugger](#)) dozvoljava pregled lokalnih i globalnih promenljivih, evaluaciju proizvoljnih izraza, postavljanja [breakpoint](#)-ova, kao i pregled koda liniju po liniju. Sam debager je pisan takođe u Python jeziku.

Često je najkraći način da se debaguje program tako što se dodaju nekoliko print izraza u izvornom kodu - zbog brzog [ažuriraj-testiraj-debaguj](#) ciklusa, ovaj jednostavni pristup je jako efektivan.

## ▼ Poglavlje 2

# Istorija Python programskog jezika

## POČETAK PYTHON JEZIKA

*Tvorac Python jezika, Guido van Rossum, razvojao je Pyzhon jezik od druge polovine 80tih godina prošlog veka.*

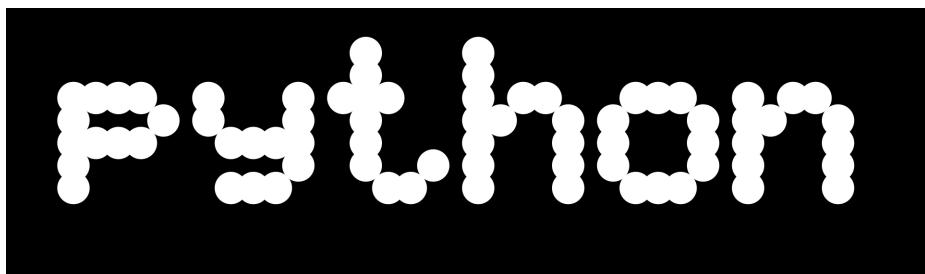
Programski jezik Python nastao je kao ideja kasnih 80tih godina 20. veka, a njegova prva implementacija počela je 1989. godine.

Tvorac Python jezika je Guido van Rossum, koji je radio u CWI (nl. [Centrum Wiskunde & Informatica](#)), razvojnom centru za matematiku i teoretsku računarsku nauku, koji pripada Holandskoj Organizaciji za Naučna Istraživanja (en. [Netherlands Organization for Scientific Research](#), NWO) u Amsterdamu.

Python predstavlja naslednika programskog jezika ABC, ali koji je mogao da obrađuje izuzetke i da ima interfejs ka Amoeba operativnim sistemom (koji je razvio Andrew Tanenbaum u Vrije univerzitetu u Amsterdamu). Van Rossum je nastavio da razvija Python programski jezik sve do 2018. godine.

### **Zanimljivost:**

Python je dobio ime po seriji Leteći Cirkus Montija Pajtona (en. *Monty Python's Flying Circus*)



Slika 2.1 Logo Python jezika iz devedesetih godina prošlog veka. [Izvor: [wikipedia.com/python](https://en.wikipedia.org/wiki/Python_(programming_language))]

### Rana istorija Python programskog jezika

Februara 1991. godine, van Rossum je objavio kod (v0.9.9) koji je već u toj fazi razvoja imao funkcionalnosti kao što su podrška za klase sa naslednošću, obrada izuzetaka, podrška za funkcije, ali i tipove podataka kao što su *list*, *dict*, *str*.

U ovom, inicijalnom izdanju, postojao je i sistem modula po ugledu na Modula-3 programski jezik. Van Rossum je opisao modul kao "jedan od glavnih osobina Python jezika".

## PYTHON PROGRAMSKI JEZIK V1

*Van Rossum je nastavio razvoj Python programskog jezika i nakon odlaska iz CWI.*

Već januara 1994. godine Python je izdat u verziji 1.0.

Glavne osobine koje su bile uključene u ovom izdanju jesu vili alati za funkcionalno programiranje (konkretno *lambda*, *map*, *filter*, *reduce*).

### **Zanimljivost:**

*Van Rossum je izjavio da je "Python 'peuzeo' funkcije *lambda()*, *filter()*, i *map()* zahvaljujući Lisp kajeru koji ih je zaboravio i poslao kao zakrpe".*

Poslednja verzija koja je izdata dok je van Rossum radio u CWI bila je Python 1.2.

Nakon toga, van Rossum se seli u Reston, Virdžinija, SAD, gde dobija posao u Korporaciji za Nacionalna Istraživanja (en. **Corporation for National Research Initiatives**, CNRI). Odatle je izdao još nekoliko verzija Python jezika.

Verzija 1.4 donela je nove opcije, za koje je Modula-3 jezik ponovo bio inspiracija.

Najbitniji noviteti bili su imenovani parametri (en. **named parameters**, **named arguments** ili **keyword arguments**), kao i podršku za kompleksne brojeve.

### **Podsetnik:**

*Imenovani parametri opisuju podršku za pozive funkcije koji jasno opisuju ime svakog parametra unutar poziva funkcije.*

Razvojni tim Python programskog jezika se 2000. godine preselio BeOpen.com da bi oformili BeOpen PythonLabs razvojni tim. Poslednja verzija pri CNRI bila je Python 1.6

Pošto je Python 1.6 bio pod licencom CNRI, klauzula u licenci je opisivala da je zakon o licenciranju bio podložan zakonima u saveznoj državi Virdžiniji. Zbog toga, Python verzija 1.6 nije bila kompatibilna sa *GNU General Public Licence*, pa je zbog toga naknadno izdata verzija 1.6.1, koja je suštinski bila ista kao 1.6, iz manje korekcije grešaka, ali bitnije, sa novom *GNU GPL* licencom.

## PYTHON PROGRAMSKI JEZIK V2

*Python v2 je imao životni vek od 20 godina, zaključno sa aprilom 2020. godine*

Python 2.0 je objavljen oktobra 2000. godine, i uveo je nove mogućnosti, prvenstveno razumevanje lista (en. [list comprehensions](#)), osobina koja se koristi u funkcionalnom programiranju, prvenstveno u jezicima SETL i Haskell.

Python 2.0 je takođe uvek i sistem za skupljanje otpada (en. [garbage collection system](#)).

### **Podsetnik:**

*Razumevanje lista je konstrukt u pojedinim programskim jezicima koji služi za pravljenje listi na osnovu postojećih listi.*

### **Podsetnik:**

*Sistem sakupljanja otpada jeste oblike automatskog upravljanja memorijom. Sakupljač otpada (collector), pokušava da povrati otpad, tj. memoriju koju su zauzeli objekti koje program više ne koristi.*

Python 2.1 je izašao pod licencom Python Software Foundation Licence, po ugledu na Apache Software Foundation.

Python 2.1 je uveo podršku za ugnježdene oblasti vidljivosti (en. [nested scope](#))

Python 2.2 je izdat decembra 2001. godine, i glavni novitet bio je objedinjenje tipova podataka i klasa u jedinstvenu hijerarhiju. Na ovaj način Python model objekata je postao zaista objektno-orientisan.

Novitet koji je uveo Python 2.5 (izdat septembra 2006. godine) jeste uvođenje ključne reči [with](#), koja enkapsulira deo koda unutar kontekstnog menadžera.

Od Python 2.6 krenula su paralelna izdanja 2.x i 3.x verzija

Python 2.6 je trebalo da bude izdat paralelno sa Python 3, sa nekim funkcionalnostima koje je trebalo da budu u novom izdanju.

Slično se desilo i sa 2.7, koja je izdata paralelno sa 3.1, u junu 2009. godine.

Od tada, prestala su paralelna izdanja dve verzije, i trenutno je Python 2.7 poslednja verzija 2.x

Novembra 2014. najavljeno je da će podrška za Python 2.x prestati 2020. godine uz preporuku korisnicima da pređu na Python v3.

Poslednja verzija Python v2, verzija 2.7.18 je izdata aprila 2020 godine i predstavlja kraj životnog ciklusa Python 2.x programskog jezika.

## PYTHON PROGRAMSKI JEZIK V3

*Python 3 je projektovan u cilju da popravi fundamentalne greške u dizajnu jezika.*

Python 3.0 izdat je decembra 2008. godine, uz kodne nazine Python 3000 ili Py3K.

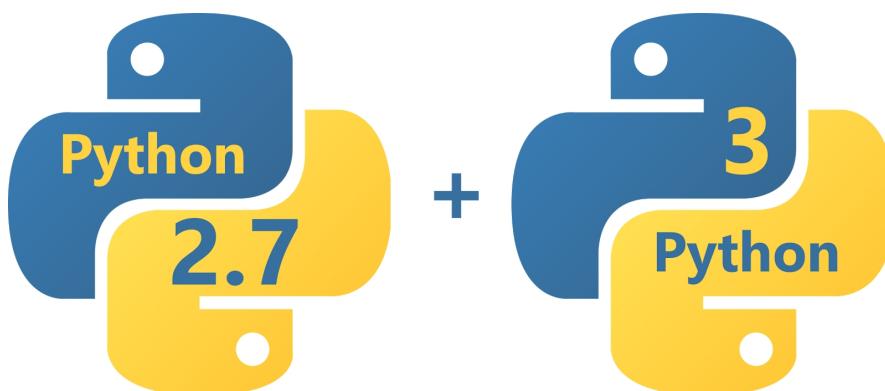
Python 3 je projektovan u cilju da popravi fundamentalne greške u dizajnu jezika.

Ove promene s nisu mogle uraditi a da se zadrži kompatibilnost unazad (en. **backwards compatibility**) sa izdanjima iz serije 2.x.

To je jedan od razloga zašto se sa pojavom Python 2.6 javio i Python 3.

*Razvojni princip pri projektovanju Python jezika 3 je bio "ukloniti duplikaciju osobina uklanjanjem starih načina rada"-*

Python 3 je razvijen sa istom filozofijom kao i u prethodnim izdanjima. Međutim, u skladu sa razvojnim principom, Python 3.0 je stavio akcenat na uklanjanje više istih modula, sa ciljem da "treba da bude jedan i samo jedan način da se nešto uradi, i da taj način bude očigledan".



Slika 2.2 Python 2.7 naspram Python 3. [Izvor: wikipedia.com/python]

Python 3.0 je ostao programski jezik sa više paradigm.

Programeri su i dalje mogli da pišu programe koji iz proceduralne, objektno-orientisane, ali i funkcionalne paradigm. Python 3.0 je nastojao da odabir paradigmne pri pisanju bude još jasniji u odnosu na prethodne 2.x verzije.

Kao što je rečeno, Python 3.0 je prestao sa kompatibilnošću unazad, i programi koji su napisani u Python 2.x programskom jeziku ne mogu raditi u Python 3.0, ukoliko se prvenstveno ne modifikuju.

Dinamičko tipiziranje podataka u kombinaciji sa promenama semantike pojedinih metoda učinilo je da je prevođenje (ne u smislu kompajliranje) koda iz Python 2.x u Python 3 jako teško.

## GLAVNE OSOBINE PYTHON 3.0

*Python 3.0 programski jezik je uvek mnoge funkcionalnosti, i doveo do prestanka razvoja 2.x verzija*

Glavne osobine Python 3.0 jezika jesu sledeće:

- Menjanje print komande da sada postaje funkcija, a ne izjava. Na ovaj način lakše se mogu koristiti druge funkcije za štampanje ukoliko je potrebno,
- Uklanjanje `input` funkcije (iz prethodnih verzija Pythona), i preimenovanje `raw_input` (iz prethodnih verzija) u `input` (u verziji 3.x),
- Dodata podrška za opcionu anotaciju funkcija koje se mogu koristiti za deklaraciju neformalnih tipova,
- Unificiranje `str` / `unicode` tipova podataka koji predstavljaju tekst, i dodavanje `bytes` i `bytearray` tipova, koji predstavljaju nizove bajtova,
- Uklanjanje kompatibilnost unazad,
- Promena u deljenju celih brojeva.

Sa svakom novom verzijom nakon 3.0, Python je uveo nove funkcionalnosti

Verzija Python programskog jezika u vreme pisanja lekcije je 3.8, i puna dokumentacija se može naći na:

<https://docs.python.org/3/whatsnew/3.8.html>

### Primer: Deljenje celih brojeva

**Napomena:** simulira se izlaz konzole, gde je `>>>` prompt koji Python izbacuje.

```
#Python v2.X
>>> 5/2
>>> 2
```

```
#Python 3.0
>>> 5 / 2
>>> 2.5
```

Za celobrojni rezultat, koristi se isključivo `//`

```
#Python 3.0
>>> 5 // 2
>>> 2
```

## ▼ Poglavlje 3

# Razlika između verzija Python programskog jezika

## DA LI TREBA I DALJE KORISTITI PYTHON 2.X?

*Od aprila 2020. godine Python 2.x ne dobija podršku od razvojnog tima.*

Prema preporukama razvojnog tima Python programskog jezika, svaki novi razvoj softvera koji se piše u Python-u, treba pisati u Python-u 3.x.

Kao što je već rečeno, najavljeno je da će početkom 2020. godine Python 2.x izgubiti podršku razvojnog tima, tj. doći će do kraja svog životnog ciklusa (en. [End of Life](#)). Ovo se desilo aprila 2020. godine, i od tada Python 2.x ne dobija nova ažuriranja.

Ažuriranja se odnose na bagove, ali i na bezbednosne ispravke. Zbog nedostatka podrške, moguće je naći nove bezbednosne ranjivosti u programiranim napisanim u Python 2.X verziji, što predstavlja veliki propust.

### **Zanimljivost:**

*Poslednja verzija Python 2.x programskog jezika jeste 2.7.18 iz 20. aprila 2020. godine.*

<https://www.python.org/downloads/release/python-2718/>

U nastavku objekta učenja biće dati konkretni primeri sa razlikama između [Python 2.x](#) i [Python 3.x](#)

Neke od funkcija koje se spominju još uvek nisu uvedeni kroz predmet, ali su dovoljno jednostavne da se razumeju imajući u vodu znanje iz sličnih predmeta.

## PYTHON 2.X VS 3.X - PRINT

*U Python 2.x print je ključna reč, dok je u 3.x funkcija*

Najpoznatija promena, i najtrivijalnija, ali i dalje bitna jeste promena u sintaksi print funkcije.

[Python 2.x](#) je imao print ključnu reč (en. statement), dok je u [Python 3.x](#) jeziku potrebno ubaciti zagrate, što pretvara u funkciju.

### Python 2.x:

```
print 'Hello, World!'
print('Hello, World!')
print "text", ; print 'print more text on the same line'
```

```
Hello, World!
Hello, World!
text print more text on the same line
```

### Python 3.x:

```
print('Hello, World!')
print("some text,", end="")
print(' print more text on the same line')
```

```
Hello, World!
some text, print more text on the same line
```

Ukoliko bismo hteli da koristimo sintaksu 2.x u 3.x, dobili bismo sledeće:

### Python 3.x:

```
print 'Hello World!'
```

```
File "<stdin>", line 1
    print 'Hello World!'
    ^

```

```
SyntaxError: Missing parentheses in call to 'print'. Did you mean print('Hello
World!')?
```

## PYTHON 2.X VS 3.X - DELJENJE CELIH BROJEVA

*Kada se izvršava Python 3.x kod koji ima deljenje celih brojeva u 2.x interpretalu, može doći do bitnih promena i grešaka.*

Jedna od bitnijih promena kod [Python 3.x](#) u odnosu na [Python 2.x](#) jeste rezultat deljenja celih brojeva.

Ova promena se često previdi, i ukoliko se ne vodi računa može izazvati velike greške (dok se ne pronađe i ispravi).

### Python 2.x:

```
print '3 / 2 =', 3 / 2
print '3 // 2 =', 3 // 2
print '3 / 2.0 =', 3 / 2.0
print '3 // 2.0 =', 3 // 2.0
```

```
3 / 2 = 1
3 // 2 = 1
3 / 2.0 = 1.5
3 // 2.0 = 1.0
```

### Python 3.x:

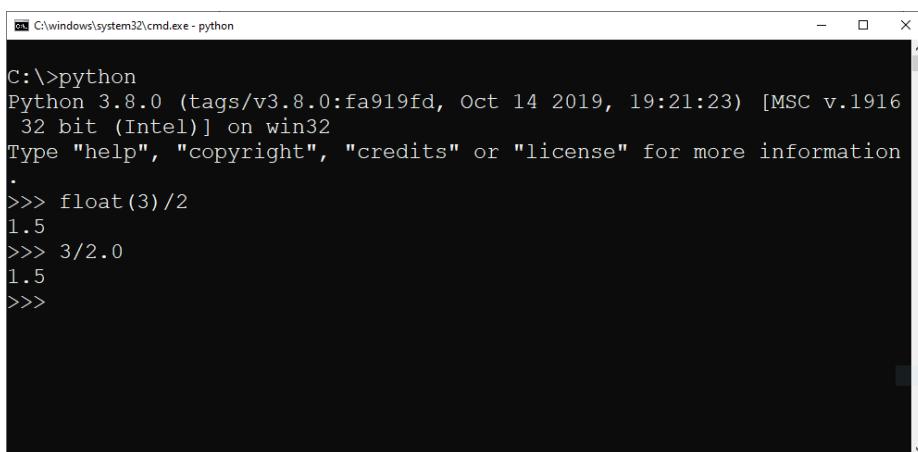
```
print('3 / 2 =', 3 / 2)
print('3 // 2 =', 3 // 2)
print('3 / 2.0 =', 3 / 2.0)
print('3 // 2.0 =', 3 // 2.0)
```

```
3 / 2 = 1.5
3 // 2 = 1
3 / 2.0 = 1.5
3 // 2.0 = 1.0
```

U Python 3.x se može koristiti `float(3)/2` ili `3/2.0` da bi se kod mogao koristiti i u Python 2.0 jeziku.

```
float(3)/2
3/2.0
```

```
1.5
1.5
```



Slika 3.1 Ukoliko se program koristi i u Python 2.x, sigurnije je koristiti float(). [Izvor: Autor]

## PYTHON 2.X VS 3.X - UNICODE FORMATIRANJE I BYTE TIP PODATAKA

*U Python 2.x jeziku razdvojeni su str i unicode tipovi podataka, dok su u Python 3.x jeziku oba formata u str tipu.*

U Python 2.x jeziku razdvojeni su str i unicode tipovi podataka, dok su u Python 3.x jeziku oba formata u str tipu.

### Python 2.x:

```
print type(unicode('this is like a python3 str type'))
```

```
<type 'unicode'>
```

### Python 3.x:

```
print('strings are now utf-8 \u03BCnico\u0394Irish!')
```

```
strings are now utf-8 μnicoΔIrish!
```

Takođe, ne postoji tip podataka byte i bytearray u Python 2.x jeziku

### Python 2.x:

```
print type(b'byte type does not exist')
```

```
<type 'str'>
```

### Python 3.x:

```
print('Python 3.x has', type(b' bytes for storing data'))
```

```
Python 3.x has <class 'bytes'>
```

## PYTHON 2.X VS 3.X - OBRADA IZUZETAKA

*U Python 3.x programskom jeziku ubačena je ključna reč as pri obradi izuzetaka.*

Obrada izuzetaka je promenjena u Python 3 programskom jeziku. Za razliku od [Python 2.x](#), u [Python 3.x](#) jeziku treba da se koristi ključna reč [as](#).

### Python 2.x:

```
try:  
    let_us_cause_a_NameError  
except NameError, err:  
    print err, '--> our error message'
```

```
name 'let_us_cause_a_NameError' is not defined --> our error message
```

### Python 3.x:

```
try:  
    let_us_cause_a_NameError  
except NameError as err:  
    print(err, '--> our error message')
```

```
name 'let_us_cause_a_NameError' is not defined --> our error message
```

## PYTHON 2.X VS 3.X - PARSIRANJE PODATAKA U INPUT() FUNKCIJI

*U Python 3.x jeziku `input()` funkcija uvek parsira u `str` tip potadaka*

U [Python 2.x](#) jeziku funkcija [input\(\)](#) je parsirala ulaz u različite tipove podataka, dok se za parsiranje u [str](#) tipove podataka sada koristila [raw\\_input\(\)](#) funkcija. U [Python 3.x](#) jeziku, [input\(\)](#) uvek parsira ulaz kao [str](#) tip podataka.

### Python 2.x:

```
>>> my_input = input('enter a number: ')  
enter a number: 123
```

```
>>> type(my_input)
<type 'int'>

>>> my_input = raw_input('enter a number: ')

enter a number: 123

>>> type(my_input)
<type 'str'>
```

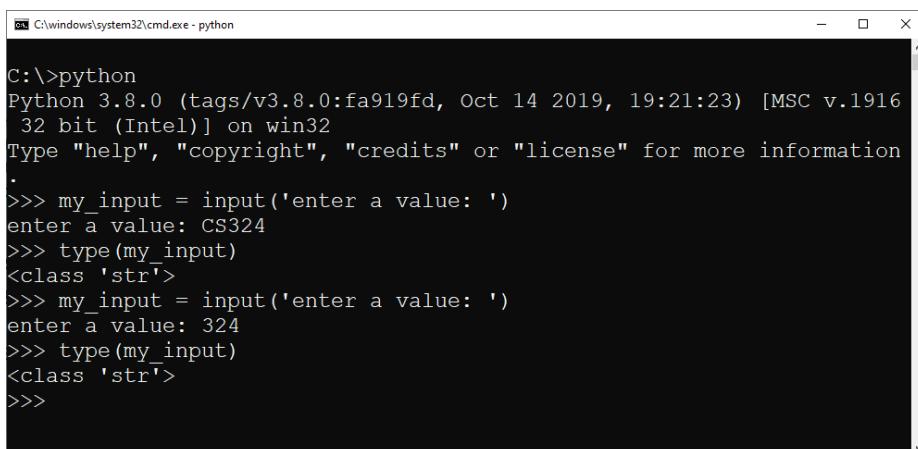
### Python 3.x:

```
>>> my_input = input('enter a number: ')

enter a number: 123

>>> type(my_input)
<class 'str'>
```

Možemo primetiti da se bilo koji ulaz parsira kao string.



```
C:\>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916
 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information
.
>>> my_input = input('enter a value: ')
enter a value: CS324
>>> type(my_input)
<class 'str'>
>>> my_input = input('enter a value: ')
enter a value: 324
>>> type(my_input)
<class 'str'>
>>>
```

Slika 3.2 Parsitanje pomoću input() funkcije. [Izvor: Autor]

## ▼ Poglavlje 4

# Primene Python programskog jezika

## PRIMENE PYTHON PROGRAMSKOG JEZIKA - WEB I GAME DEVELOPMENT

*Python programski jezik je poznat po svojoj opštoj primeni u svakom domenu razvoja softvera.*

Python programski jezik je poznat po svojoj opštoj primeni u gotovo svakom domenu razvoja softvera. Python kao takav predstavlja koji je doživeo najveći uspon za razvoj aplikacija.

U nastavku su date najpoznatije primene Python programskoj jeziku

### **Web development**

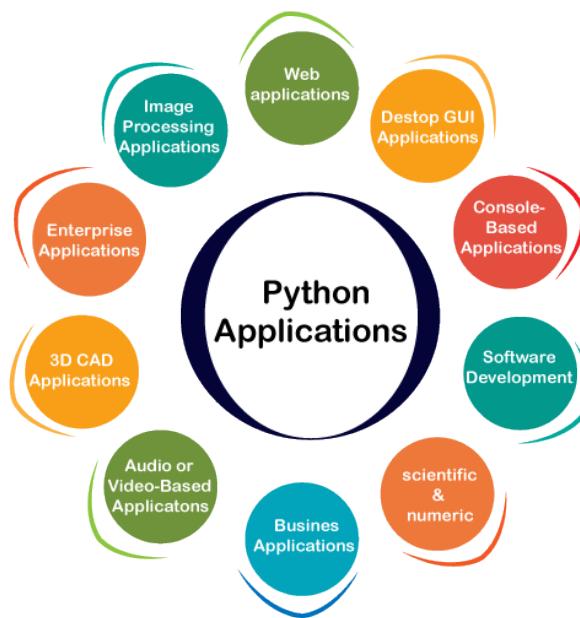
Python programski jezik se može koristiti za brz razvoj web aplikacija. Python koristi framework-ove i za *common-backend* logiku i biblioteke koji mogu pomoći za integriranje protokola kao što su HTTPS, FTP, SSL i drugi. Takođe može procesuirati JSON, XML, e-mail i druge tipove.

Najpoznatiji framework-ovi za Python jesu Django i Pyramid za "teže" aplikacije, dok se Flask koristi kao mikro-framework. Korišćenjem ovih framework-a dobija se na sigurnosti, skalabilnosti i ujedno i na jednostavnosti, pogotovo u poređenju sa razvojem web-stranice od nule.

### **Razvoj igara**

Python se koristi u razvoju interaktivnih video igara. Korišćenjem biblioteka kao što su PySoy koji predstavlja 3D endžin (en. *engine*) sa podrškom za Python 3, PyGame koji pruža funkcionalnost. Poznate igre koje su pisane (bilo u celosti ili delom) u Python jeziku:

- Battlefield 2
- Civilization IV
- Sims 4
- World of Tanks
- EVE Online



Slika 4.1 Primene Python programskog jezika. [Izvor: <https://www.javatpoint.com/python-applications>]

## PRIMENE PYTHON PROGRAMSKOG JEZIKA - AI/MAŠINSKO UČENJE, DATA SCIENCE, KONZOLNE APLIKACIJE

*Python koristi mnoge biblioteke za konkrene primene: Pandas, Scikit-Learn, NumPy, Matplotlib, REPL*

### **Veštačka inteligencija i mašinsko učenje**

Mnoge Python biblioteke kao što su Pandas, Scikit-Learn i NumPy se koriste u oblasti veštačke inteligencije, pogotovo u mašinskom učenju. Posebne lekcije obrađuju ovu temu, kao i predmet Mašinsko učenje.

Računar koristi algoritme mašinskog učenja da "uči" na osnovu svog iskustva sa prethodnim skupom podataka, ili da samostalno uči bez poznavanja ranijih izlaza.

### **Data Science i vizuelizacija podataka**

Data Science postaje jedna od vodećih oblasti istraživanja kako u akademskim krugovima tako i u privredi. Mogućnost obrade velike količine podataka postao je imperativ u današnjem svetu.

Python nudi biblioteke za laku obratu i vizuelno predstavljanje podataka. Najčešće korišćene biblioteke jesu Matplotlib i Seaborn. Sve što mogu konkretne aplikacije za vizuelno predstavljanje podataka (uključujući i Excel), može se postići i sa Matplotlib bibliotekom za iscrtavanje grafova.

### Konzolne aplikacije

Konzolne aplikacije mogu se pokrenuti sa komandne linije ili shell-a. Python može razviti ovakav tip aplikacija vrlo lako i brzo. **REPL** (en. **Real-Eval-Print-Loop**) interakcijom sa Python interpretatom mogu se odmah videti izlazi programa.



Slika 4.2 Primene Python programskog jezika. [Izvor: <https://www.javatpoint.com/python-applications>]

## PRIMENE PYTHON PROGRAMSKOG JEZIKA - DESKTOP GUI, WEB SCRAPING, BIZNIS, AUDIO/ VIDEO APLIKACIJE

*Python koristi mnoge biblioteke za konkrene primene: Tkinter, BeautifulSoup, Tryton*

### Desktop GUI

Python se može koristiti i za razvoj desktop aplikacija i korisničkih interfejsa. Za razvoj desktop grafičkog korisničkog interfejsa (en. **Graphics User Interface**, GUI) najpopularnija je **Tkinter** biblioteka. Ostali alati uključuju wxWidgets, Kivy, PYQT, koji se mogu koristiti za razvoj aplikacija za različite platforme.

### Aplikacije za preuzimanje podataka sa Interneta

Python se može koristiti za preuzme (en. **pull**) veliku količinu podataka s web stranica koje se nakon toga mogu koristiti u primenama kao što su poređenje cena, oglasa, razvoj i sl. Ovakav

tip aplikacija nazivaju se aplikacije za preuzimanje podataka sa Interneta (en. **Web Scraping Applications**). Python biblioteka koja se najčešće koristi za ovu primenu jeste [BeautifulSoup](#).

### Biznis aplikacije

Biznis aplikacije su nešto različite u odnosu na standardne aplikacije, i uglavnom se bave e-commerce-om, planiranjem resursima u korporacijama (en. Enterprise Resource Planning, ERP), i sl. Aplikacije ovakvog tipa moraju biti skalabilne, proširive, i čitljive. Sve ovo Python programski jezik može da pruži kroz platformu za razvojanje biznis aplikacija kao što je [Tryton](#).

### Audio & Video aplikacije

Python programski jezik se može koristiti u razvoju aplikacija koje podržavaju multitasking i na svojim izlazima imaju audio i/ili video formate datoteka.

Primeri uspešnih Audio & Video aplikacija napisanih u Python programskom jeziku uključuju TimPlayer i Cplay.



Slika 4.3 Primene Python programskog jezika. [Izvor: <https://www.javatpoint.com/python-applications>]

## PRIMENE PYTHON PROGRAMSKOG JEZIKA - CAD

*Python koristi mnoge biblioteke za konkrene primene: Pillow, OpenCV, SimpleITK*

### CAD Aplikacije

CAD (en. **Computer Aided Desing**) aplikacije mogu biti veoma komplikovane zbog mnogih komponente koje sadrže i o kojima stalno treba voditi računa (objekti i njihova reprezentacija, funkcije i sl.)

Najpoznatija aplikacija CAD tipa razvijena u Python programskom jeziku jeste Fandango.

### **Embedded aplikacije**

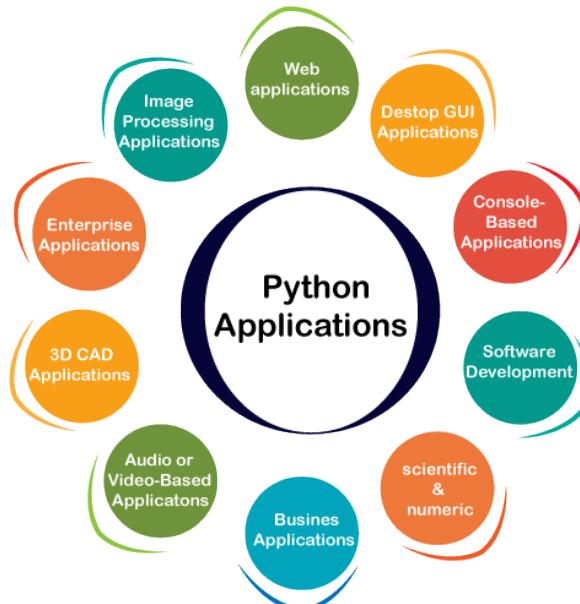
Python jezik je baziran na C jeziku pa se može iskoristiti za se napravi Embedded C softver za embedded aplikacije. Ovo omogućuje primenu aplikacija visokog nivoa na manjim uređajima koje mogu da vrše izračunavanja na Python jeziku.

Najpoznatija embedded platforma, Raspberry Pi, upravo koristi Python za svoja izračunavanja. Može se koristiti kao računar, ali i kao mikrokontroler, tj. embedded ploča za izračunavanje (en. high-level computations)

### **Aplikacije za procesiranje slika**

Python programski jezik sadrži mnoge biblioteke koje služe za procesiranje i obradu slika. Slike se mogu manipulisati prema zahtevima korisnika.

Najpoznatije biblioteke za rad sa slikama jesu Pillow, OpenCV, SimpleITK.



Slika 4.4 Primene Python programskog jezika. [Izvor: <https://www.javatpoint.com/python-applications>]

## ▼ Poglavlje 5

# Python 3 razvojno okruženje (IDE)

## INTEGRISANO RAZVOJNO OKRUŽENJE (IDE) I EDITORI KODA

*Za razvoj ozbiljnih aplikacija, potrebno je koristiti integrisano razvojno okruženje - IDE*

Do sada je bilo reči o razvoju Python aplikacija preko komandne linije, tj. o pisanju programa direktno u komandnu liniju iz prozora terminala.

Ovaj pristup jeste dovoljan za vrlo jednostavne aplikacije, ali ukoliko želite da se bavite ozbiljnijim razvojem, potrebno je koristiti [integrisano razvojno okruženje](#) (en.[Integrated Development Environment](#), IDE).

### **Osobine IDE okruženja**

IDE predstavlja skup softvera koji sadrži sve komponente za razvoj i testiranje softvera. Programer može koristiti neke ili sve alate u okviru jednog IDE okruženja.

Alati IDE okruženja uključuju editore teksta, biblioteke, kao i platforme za kompajliranje, debagovanje i testiranje.

IDE okruženje pomaže da se automatizuju zadaci programera tako što smanjuju broj zadataka koje bi inače ručno programer ratio, i ujedno spaja sve alate u jedan zajednički framework.

Da ne postoji IDE okruženje, programer bi morao ručno da obavi procese selekcije, integracije i postave (en. [deployment](#)) programa.

### **Editor koda**

Za razliku od IDE okruženja, neki programeri preferiraju [editor koda](#) (en. [code editor](#)), koji predstavlja editor teksta u koje programer može pisati kod za razvoj bilo kog softvera. Editor obično dozvoljavaju programeru da sačuvaju te tekstualne datoteke.

### **Razlike između IDE okruženja i editora koda**

I IDE i editor koda se mogu koristiti za razvoj softvera. Editor teksta koji sadrže obe varijante mogu pomoći programeru u pisanju skripti, modifikovanju koda i drugih operacija nad tekstrom koda.

Međutim, ukoliko programer koristi IDE, onda može obavljati i druge funkcije koje editor koda uglavnom ne poseduje.

Ove funkcije koda obično uključuju izvršenje koda, kontrolu verzije, debagovanje, interprataciju i kompajlovanje, funkcije auto-complete, auto-linting, i sl.

IDE obično poseduje i integrisani sistem za upravljanje datotekama.

Editor koda se može smatrati kao običan tekstualni editor sa nekim dodatim funkcijama (može prepoznati programski jezik).

## PREGLED NAJŠEŠĆE KORIŠĆENIH EDITORA KODA: NOTEPAD++

*Notepad++ je jedan od najprepoznatljivijih editora koda, nastao 2003. godine, i još uvek je aktivan, prvenstveno među korisnicima Windows operativnih sistema.*

Kao što je rečeno, editor koda je tekstualni editor koji uglavnom ima neke dodatne funkcionalnosti koje olakšavaju programerima da pišu izvorni kod softvera.

### **Zanimljivost:**

*Kada bi se sve dodatne funkcionalnosti uklonile sa editora koda, dobio bi se Notepad.*

Na tržištu komercijalnog softvera postoji mnogo rešenja za editore koda, od kojih su neke besplatne opcije, a neke se plaćaju, ili na bazi preplate, ili jednokratno.

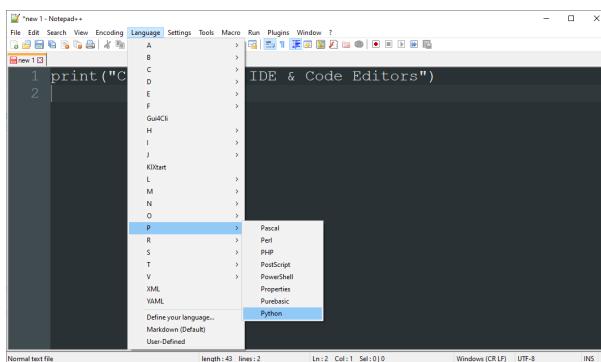
U nastavku objekta učenja biće reči o najpopularnijim editorima koda koji su besplatni.

### **Notepad++**

Notepad++ je jedan od najprepoznatljivijih editora koda, nastao 2003. godine, i još uvek je aktivan, prvenstveno među korisnicima Windows operativnih sistema.

Notepad++ zaista podseća na Notepad, ali podržava više tabova. Najbitnija opcija jeste prepoznavanje sintakse mnogih programskeh jezika, (ključne reči ili neke standardne funkcije).

<https://notepad-plus-plus.org>



Slika 5.1 Odabir programskog jezika u Notepad++ [Izvor: Autor]

Slika 5.2 Izgled Notepad++ kada se ne selektuje jezik. [Izvor: Autor]

Slika 5.3 Izgled Notepad++ kada se selektuje jezik. [Izvor: Autor]

## PREGLED NAJŠEŠĆE KORIŠĆENIH EDITORA KODA: VISUAL STUDIO CODE

*Visual Studio Code predstavlja editor koda koji ima ugrađene napredne funkcije i čini ga bliži pravom IDE okruženju.*

### Visual Studio Code

Visual Studio Code je besplatan editor koda kojeg je razvio Microsoft 2015. godine i koji pruža pregršt mogućnosti u odnosu na mnoge editora koda (poput Notepad++), ali nije potpuni IDE kao što je puni Visual Studio IDE.

Visual Studio Code dolazi sa ugrađenom podrškom za JavaScript, TzpeScript i Node.js jezike, dok se mogu instalirati ekstencije za druge programske jezike poput C++, C#, Java, Python, PHP, Go) ali i za runtime-ove (kao što su .NET i Unity).

Visual Studio Code takođe pored prepoznavanja sintakse sadrži i IntelliSense, opciju za (polu)automatsko kompletiranje koda, koje uključuje promenljive, metode ali i uvežene module.

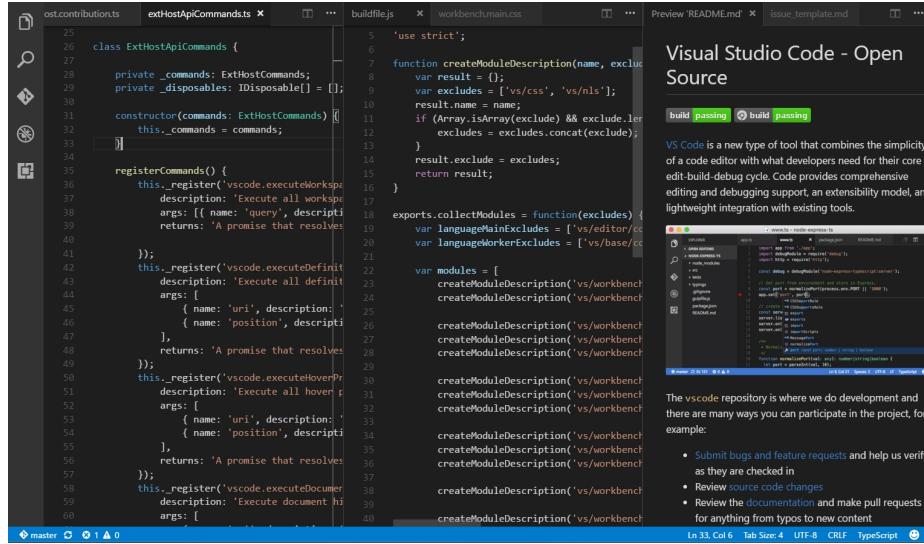
Pruža i opciju navigacije do tražene reči, preimenovanje reči odjednom na više mestu, opcije lintovanja i višestruke selekcije.

Pored osnovnih opcija editora koda, Visual Studio Code pruža i opcije debagovanja sa alatima terminala.

Konačno, Visual Studio Code nudi integraciju sa softverom za kontrolu verzije poput Git-a.

<https://code.visualstudio.com>

Treba napomenuti da Visual Studio Code, kao i ostali editori koda ne sadrži kompjajlere i interpretare, i treba njih providno instalirati i integrisati sa ovim editorom.



Slika 5.4 Izgled Visual Studio Code prozora. [Izvor: Autor]

## PREGLED NAJŠEŠĆE KORIŠĆENIH IDE: SPYDER

*Python Spyder IDE predstavlja jedan od najčešće korišćenih okruženja, prvenstveno za naučno programiranje.*

Spyder, kao deo Anaconda distribucije Python i R programske jezike, predstavlja IDE okruženje, prvenstveno za naučno programiranje, i projektovano je u vidu za inženjere, naučnike i analitičare podataka.

Spyder je IDE otvorenog koda tj. besplatan je za preuzimanje. Može se koristiti na svim popularnim operativnim sistemima, tj. Windows, Linux i macOS.

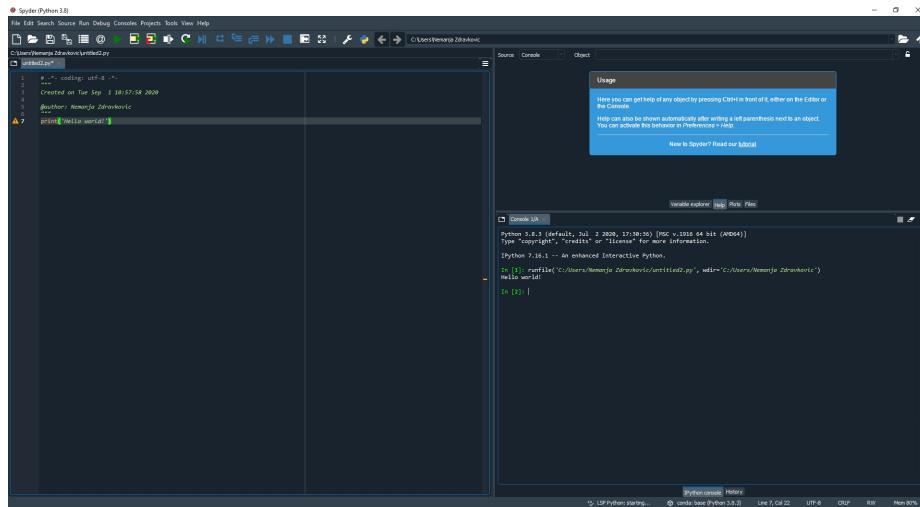
Kao pravo IDE okruženje, Spyder poseduje alate za ažuriranje koda, analizu, debagovanje i profilisanje funkcionalnosti, uz jasan pregled i korisnički interfejs.

Spyder se može preuzeti sa <https://www.spyder-ide.org>, ali i sa Anaconda distribucijom <https://www.anaconda.com/products/individual>

Spyder se može isprobati i online na sajtu MyBinder.

Izgled Spyder IDE je dat na slici.

Pored samog editora koji zauzima najveći deo prostora, konzola je uvek vidljiva i uvek se može direktno kucati u konzolu, ali takođe se interfejs može i personalizovati.



Slika 5.5 Korisnički interfejs Spyder IDE okruženja. [Izvor: Autor]

## PREGLED NAJŠEŠĆE KORIŠĆENIH IDE: JUPYTER

*Jupyter predstavlja interaktivno IDE okruženje bazirano na web stranicama, koje objedinjuje pisanje koda i dodavanje dodatnih elemenata*

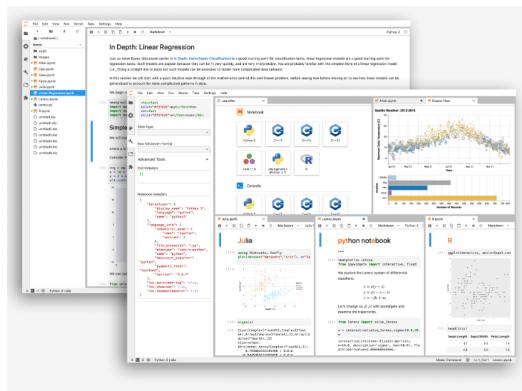
Jupyter predstavlja tzv. **notebook interface**, odnosno interativno okruženje namenjeno kako pisanju koda, tako i prikazu i vizuelizaciji podataka.

Slično kao i Spyder, Jupyter se koristi najpre u projektima naučne svrhe, najčešće za mašinsko učenje, jer podržava laku integraciju izlaznih podataka.

Jupyter je modularno okruženje koje podržava se u postojeće **sveske** lako dodaju nove komponente i da se integrišu u postojeće.

Jupyter se može isprobati i online na <https://jupyter.org>

Jupyter je deo Anaconda 3 distribucije, i automatski se instalira prilikom instalacije ove platforme.



Slika 5.6 Izgled Jupyter svezaka. [Izvor: [jupyter.com](https://jupyter.com)]

Izgled Spyder IDE je dat na slici.

Pored samog editora koji zauzima najveći deo prostora, konzola je uvek vidljiva i uvek se može direktno kucati u konzolu, ali takođe se interfejs može i personalizovati.

Prilikom pokretanja Jupyter-a pokreće se web strana gde se mogu otvoriti .py datoteke i Jupyter sveske.



Slika 5.7 Izgled Jupyter web okruženja. [Izvor: Autor]

Slika 5.8 Ulazak u .py datoteku kroz Jupyter web okruženje. [Izvor: Autor]

## PREGLED NAJŠEŠĆE KORIŠĆENIH IDE: PYCHARM

*PyCharm IDE okruženje nudi opštu funkcionalnost i bliže je okruženjima za razvoj softvera na nižim programskim jezicima.*

JetBrains distribucije za razvoj softvera su jako popularne kako među naučnim, tako i među komercijalnim krugovima programera.

JetBrains okruženje za razvoj Python aplikacija jeste PyCharm, koje nudi opštu funkcionalnost, ne samo za naučne svrhe (kao što je bio slučaj sa Spyder-om), već nudi opcije za lako postavljanje web aplikacija, integracija sa web servisima i druge opcije.

Međutim, PyCharm je licencirano okruženje, tj. plaća se.

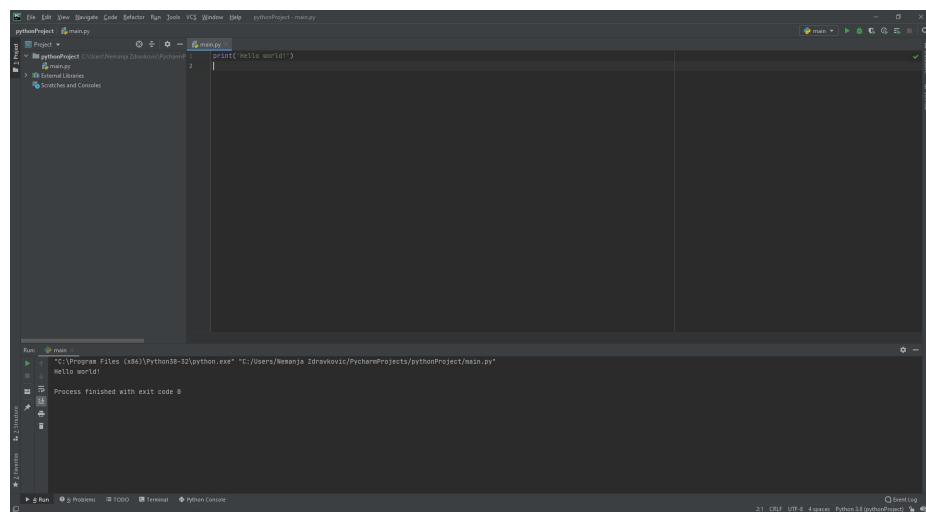
Mesečna licenca za individualnu upotrebu iznosi oko 9€, ili oko 90€ godišnje, dok za firme cena je nešto veća, oko 200€ godišnje.

Srećom, za univerzitete, odnosno studente i nastavnike, JetBrains nudi besplanu licencu sa validnom akademskom e-mail adresom, koja traje godinu dana i koja se nakon toga obnavlja.

PyCharm se može preuzeti sa sledeće stranice:

<https://www.jetbrains.com/pycharm/>

PyCharm okruženje je bliže okruženjima za razvoj softvera u nižim programskim jezicima, i takođe nudi opciju za pregled izlaza, kako kroz *run* tab, tako i kroz *terminal* tab, gde možemo kucati direktno u konzoli.



Slika 5.9 Izgled PyCharm IDE okruženja. [Izvor: Autor]

## ▼ Poglavlje 6

### Pokazna vežba #2

#### UVOD U POKAZNU VEŽBU #2

*Editori koda se vrlo lako instaliraju, jer u opštem slučaju ne zahtevaju već instaliran interpreter.*

U pokaznoj vežbi izvršiće se instalacija Notepad++ editora koda kao i Visual Studio Code editora koda na računar sa Windows operativnim sistemom.

Nakon instalacije Visual Studio Code-a, povezaće se sa već instaliranim Python interpretrom i na taj način se može ovaj editor koda koristiti kao razvojno okruženje.

Procenjeno trajanje pokazne vežbe iznosi 45 minuta.

#### INSTALACIJA NOTE PAD++

*Notepad++ se vrlo lako instalira, i lako i koristi jer se integriše u kontektsni meni.*

Notepad++ se može preuzeti sa sajta <https://notepad-plus-plus.org/>



Slika 6.1 Preuzimanje Notepad++ editor koda. [Izvor: Autor]

Nakon jednostavne instalacije, Notepad++ možete dodatno konfigurisati u pogledu izgleda interfejsa.

Najbolja opcija Notepad++ jeste što podržava prepoznavanje sintakse mnogih programskih jezika.

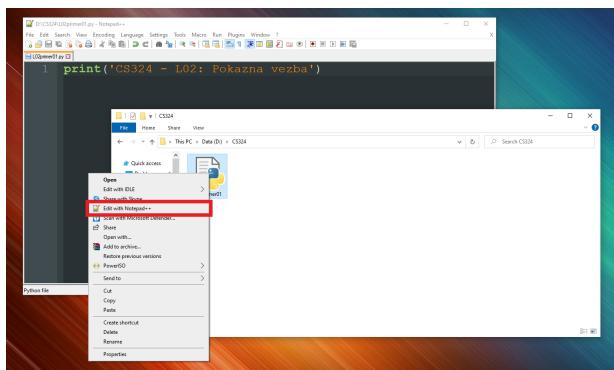
Slika 6.2 Podrška za mnoge programske jezike. [Izvor: Autor]

Nakon odabira jezika, u Notepad++ možemo da vidimo sintaksu jezika, ali ne postoje naprednije opcije kao *auto-complete* ili *IntelliSense*

Slika 6.3 Prepoznavanje sintakse u Notepad++ [Izvor: Autor]

Jedna od najčešće korišćenih "osobina" Notepad++ jeste ta da se integriše u kontekstni meni Windows-a, i bilo koja datoteka se može otvoriti i pregledati desnim klikom korišćenjem Notepad++.

Ova osobina je možda i glavni razlog zašto se Notepad++ koristi i pored postojanja ozbiljnijih editora koda.



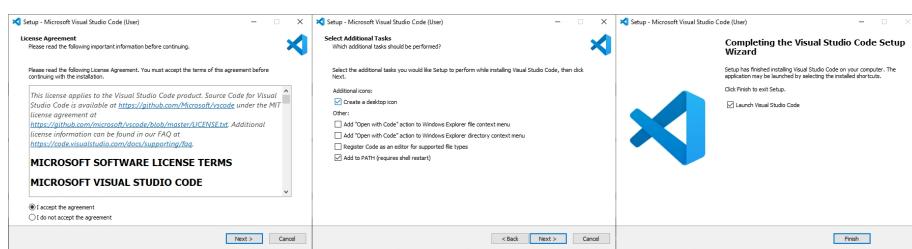
Slika 6.4 Integracija u kontekstni meni Windows-a. [Izvor: Autor]

## INSTALACIJA VISUAL STUDIO CODE EDITORA KODA

*Visual Studio Code se lako instalira, ali je neophodno dodati ekstenzije za Python.*

Visual Studio Code se može naći na <https://code.visualstudio.com>

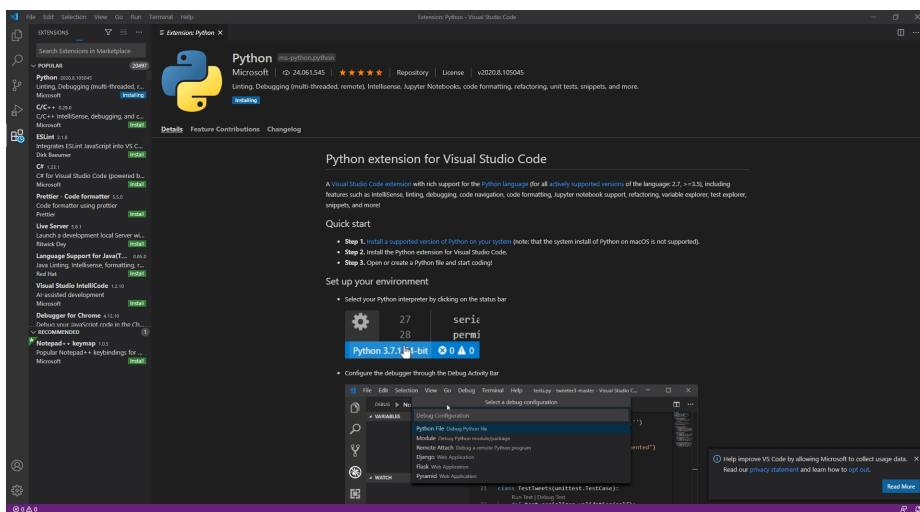
Nakon pokretanja instalacije, pratiti sve korake, i štiklirati *Add to PATH*.



Slika 6.5 Instalacija Visual Studio Code editora koda. [Izvor: Autor]

Nakon instalacije (eventualno i restarta) može se pokrenuti Visual Studio Code koji omogućava ažuriranje datoteka u raznim programskim jezicima.

Da bi Visual Studio "prepoznao" jezik, potrebno je preuzeti odgovarajuću ekstenziju sa njegove prodavnice.



Slika 6.6 Instalacija Python ekstenzije. [Izvor: Autor]

Ova ekstenzija, osim prepoznavanja sintakse jezika, ima opcije IntelliSense, linting, debagovanja, navigaciju i formatiranje koda, podršku za Jupyter sveske, pretraživač promenljivih, kao i mnoštvo drugih opcija.

## ODABIR PYTHON INTERPRETERA U VISUAL STUDIO CODE-U

*Pre pokretanja koda u Visual Studio Code-u, potrebno je ubaciti interpreter.*

Ukoliko već postoji instaliran Python interpreter (a instaliran je direktnom instalacijom Python-a u vežbi #1), dok se može direktno i pokrenuti.

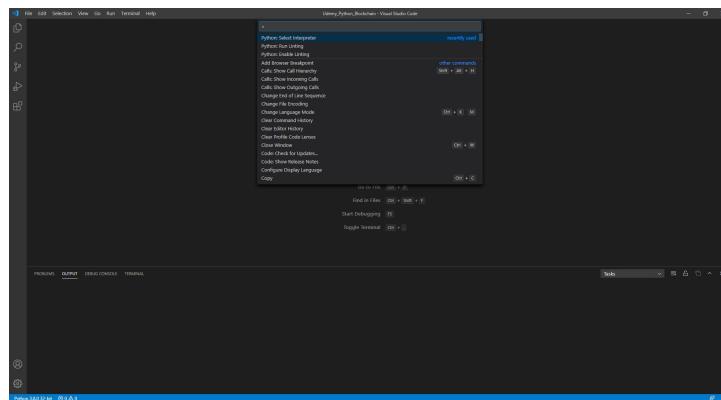
### Podsetnik:

*Editori koda obično ne sadrže interpreter/kompajler, i nije moguće pokrenuti program napisan u jeziku ukoliko ne postoji (i ukoliko nije povezan) interpreter ili kompajler.*

### Odabir Python Interpretera

Odabir Python interpretera u Visual Studio Code-u vrši se kroz sledeće korake:

1. Pokrenuti *Show All Commands* klikom na **Ctrl + Shift + P**
2. Učekati **Python: Select Interpreter**
3. Iz padajuće liste instaliranih interpretera odabratи koji interpreter želite da koristite.



Slika 6.7 Show All Commands. [Izvor: Autor]

Slika 6.8 Odabir Python Interpretera. [Izvor: Autor]

## KORIŠĆENJE VISUAL STUDIO CODE EDITORA KODA - PRIMER #1

### Visual Studio Code - Primer #1

Nakon odabira interpretera (Visual Studio Code i svoja Python ekstenzija podržava i Python 2.7 kao i Python 3.x) možete u editoru pisati programe.

#### Primer #1: Površina trougla (10 minuta)

Uneti stranice trougla a, b, i c, i onda izračunati površinu trougla po Heronovm obrascu:

$$s = \frac{a+b+c}{2}$$

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

```
a = float(input('Strana a: '))
b = float(input('Strana b: '))
c = float(input('Strana c: '))

# Poluobim
s = (a + b + c) / 2

# Povrsina
area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
print('Povrsina trougla je %.2f' %area)
```



l02\_prime0.py

```
1 #!/usr/bin/python
2
3 a = float(input("Strana a: "))
4 b = float(input("Strana b: "))
5 c = float(input("Strana c: "))
6
7 # Poluhranin
8 s = (a + b + c) / 2
9
10 # Formular
11 area = ((s-a)*(s-b)*(s-c)) ** 0.5
12 print("Povrsina trougla je", area)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\VSCode\l02 & D:\anacelci\python\www\l02\l02_prime0.py
Strana a: 3
Strana b: 5
Strana c: 5
Povrsina trougla je 6.08
PS D:\VSCode\l02
```

File Edit Selection View Go Run Terminal Help

l02\_prime0.py - l02 - Visual Studio Code

Slika 6.9 Primer 1 u Visual Studio Code-u. [Izvor: Autor]

## Objašnjenje koda:

`input()` funkcija unosi podatke sa tastature, kao string.

`float()` funkcija vrši konverziju podataka u razlomljen broj.

# KORIŠĆENJE VISUAL STUDIO CODE EDITORA KODA - PRIMER #2 & #3

Visual Studio Code - Primer #2 i Primer #3

## Primer #2: Konverzija stepeni Celzijusa u Farenhajt (10 minuta)

Uneti vrednost temperature u stepenima Celzijusa i pretvoriti u stepene Farenhajta. Konverzija se računa po formuli

$$T_F = (T_C \times 1.8) + 32$$

```
tempC = float(input('Uneti temperaturu u C: '))

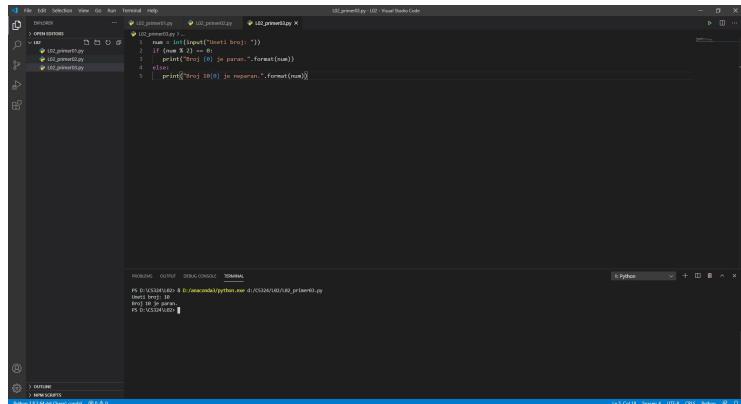
tempF = (tempC * 1.8) + 32
print('%0.1f stepeni Celzijus je %0.1f stepeni Farenhjita.' %(tempC,tempF))
```

### Primer #3: Proveriti da li je broj paran ili neparan (10 minuta)

Uneti ceo broj.

Program vraća odgovor da li je broj paran ili neparan.

```
num = int(input("Uneti broj: "))
if (num % 2) == 0:
    print("{0} je paran".format(num))
else:
    print("{0} je neparan".format(num))
```



Slika 6.10 Primer #3 u Visual Studio Code-u. [Izvor: Autor]

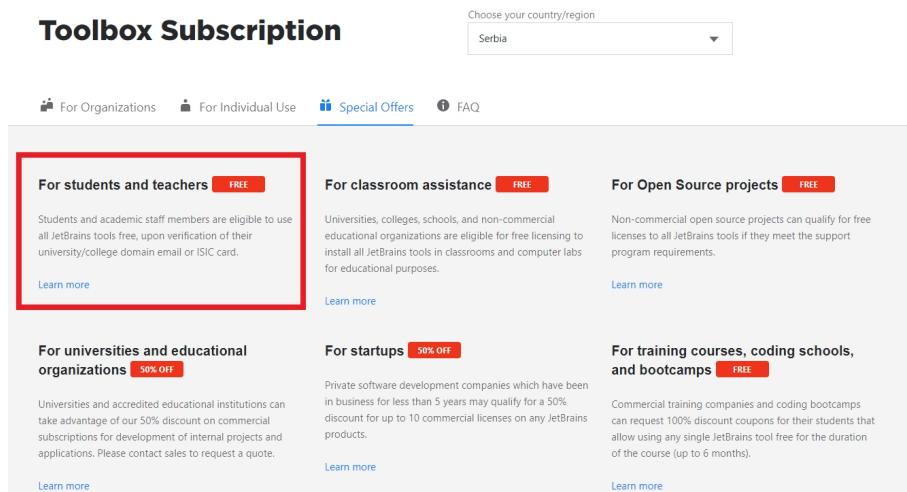


# DOBIJANJE JETBRAINS LICENCE I INSTALACIJA PYCHARM OKRUŽENJA

*Da bi se preuzeila puna verzija PyCharm okruženja, najpre treba obezbediti licencu.*

Da bi se preuzeila puna verzija PyCharm okruženja, najpre treba obezbediti licencu.

Na sajtu <https://www.jetbrains.com/pycharm/buy/#discounts?billing=yearly> nalazi se odeljak za studente i nastavnike.



Slika 6.12 JetBrain licenca za studente i nastavnike. [Izvor: [jetbrains.com](https://www.jetbrains.com/student/)]

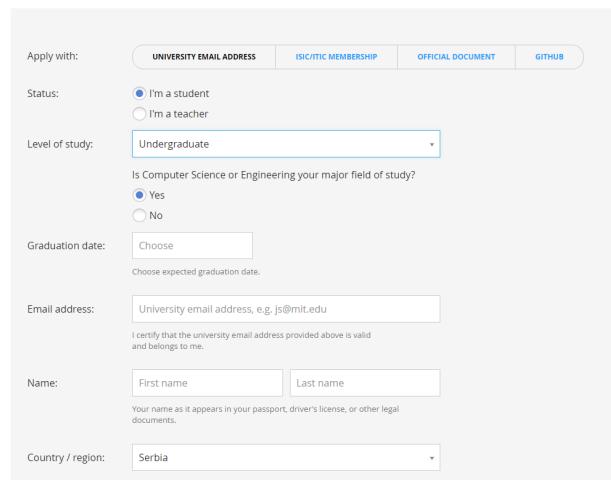
Izabratи tu opciju za studente i nastavnike, i popunitи formular korišćenjem fakultetske email adrese . Nakon toga, moguće je prijaviti se na sajt prema daljem uputstvu.

**Podsetnik:**

Za prijavu za godišnju licencu koristiti fakultetsku email adresu.

**JetBrains Products for Learning**

Before you apply, please read the [Educational Subscription Terms and FAQ](#).



The form is a web-based application for applying for a free JetBrains license. It starts with a 'Status' section where 'I'm a student' is selected. Below that is a 'Level of study' dropdown set to 'Undergraduate'. The next section is 'Is Computer Science or Engineering your major field of study?' with 'Yes' selected. The 'Graduation date' field is a 'Choose' button. The 'Email address' field is 'University email address, e.g. js@mit.edu'. Below it is a checkbox for 'I certify that the university email address provided above is valid and belongs to me.' The 'Name' section has 'First name' and 'Last name' fields. Below them is a note: 'Your name as it appears in your passport, driver's license, or other legal documents.' The 'Country / region' dropdown is set to 'Serbia'. At the top, there are tabs for 'UNIVERSITY EMAIL ADDRESS', 'ISIC/ITIC MEMBERSHIP', 'OFFICIAL DOCUMENT', and 'GITHUB', with 'UNIVERSITY EMAIL ADDRESS' being the active tab.

Slika 6.13 Prijava za dobijanje besplatne licence. [Izvor: jetbrains.com]

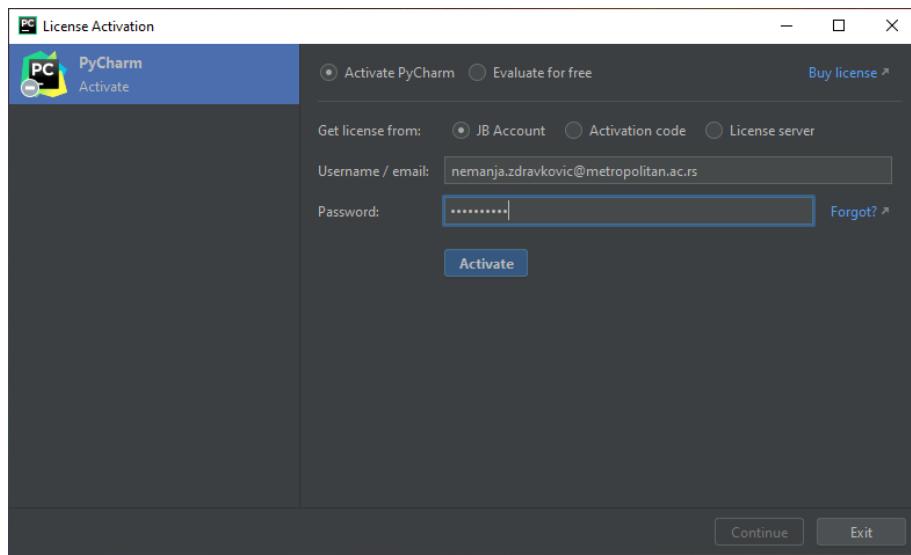
## INSTALIRANJE PYCHARM IDE

*Instalacija PyCharm IDE je jednostavna i ne bi trebalo da izazove poteškoće.*

Nakon prijave i verifikacije email adrese, preuzeti PyCharm instalaciju i pratiti uputstva.

**Podsetnik:**

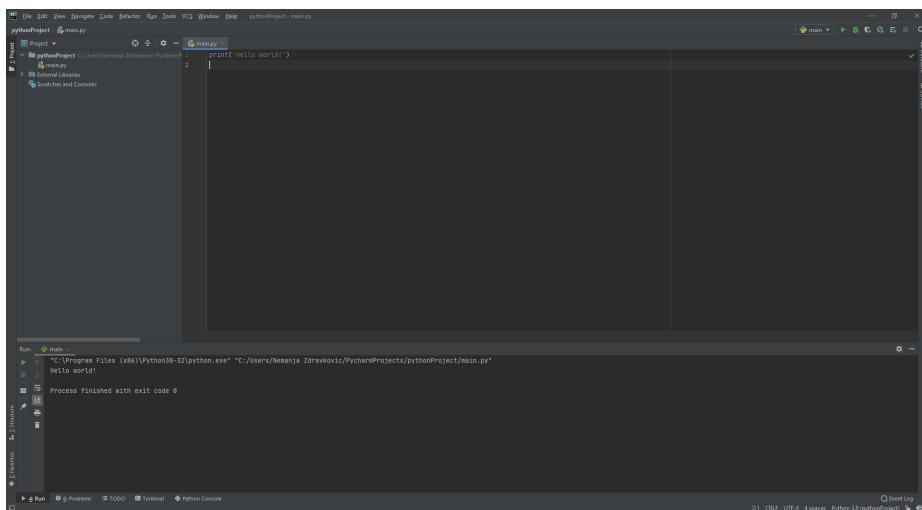
*Ukoliko je potrebno, i uPyCharm IDE podesiti Python interpreter.*



Slika 6.14 Aktivacija PyCharm IDE-a. [Izvor: Autor]

Nakon uspešne aktivacije, možete i u PyCharm-u pisati projekte u Pythonu.

PyCharm nudi veće mogućnosti nego Visual Studio Code, posebno integraciju sa drugim servisima o kojima će biti reči u daljem toku predmeta.



Slika 6.15 Prikaz PyCharm IDE interfejsa. [Izvor: Autor]

## KORIŠĆENJE PYCHARM IDE - PRIMER #4

### *PyCharm - Primer #4*

#### **Primer #4: Provera prostog broja (15 minuta)**

Napisati program u PyCharm-u koji proverava da li je broj prost.

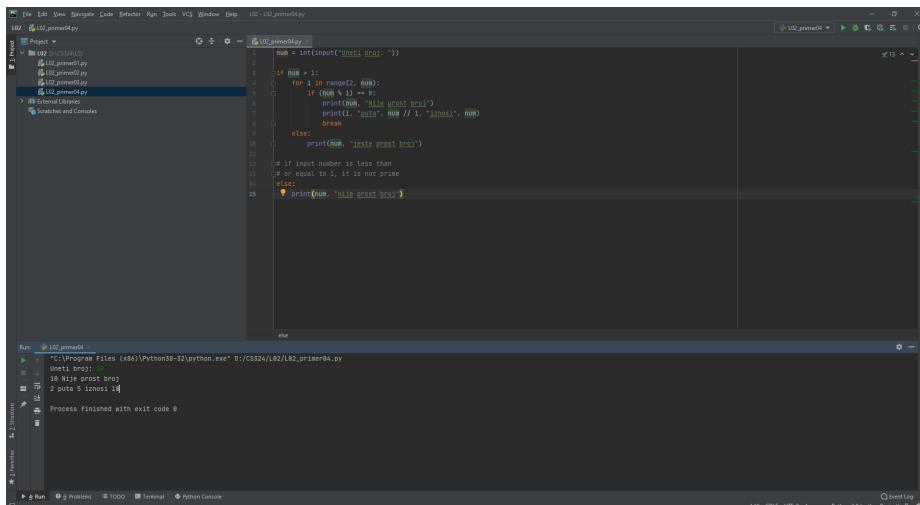
Ukoliko broj nije prost, napisati njegove sadržaoce.

```
num = int(input("Uneti broj: "))

if num > 1:
    for i in range(2, num):
        if (num % i) == 0:
            print(num, "Nije prost broj")
            print(i, "puta", num // i, "iznosi", num)
            break
    else:
        print(num, "jeste prost broj")

# if input number is less than
# or equal to 1, it is not prime
else:
    print(num, "nije prost broj")
```

Prepoznati delove koda koji se odnose na petlje i kontrolu toka. U čemu je razlika u odnosu na C familiju programskih jezika?



Slika 6.16 Primer #4 u PyCharm-u. [Izvor: Autor]

## ▼ Poglavlje 7

### Individualna vežba #2

#### UVOD U INDIVIDUALNU VEŽBU #2

*Nakon dobijanja licence od JetBrains-a, PyCharm se može slobodno koristiti u edukativne svrhe*

U individualnoj vežbi #2 proći će se kroz proces dobijanja studenteske licence za PyCharm IDE, samostalnu instalaciju PyCharm IDE okruženja, kao i pisanje jednostavnih programa u Python-u koristeći ovo okruženje.

Procenjeno trajanje individualnih vežbi iznosi 90 minuta.

#### INDIVIDUALNA VEŽBA #2

*Napisati jednostavne programe u Python programskoj jeziku koristeći instalirana okruženja.*

Individualna vežba #2 prolazi kroz rad u editorima koda i IDE okruženjima.

Studenti treba samostalno da znaju kada mogu brzo iskoristiti editora koda za brz pregled i ispravke programa, a kada razvijati softver u IDE okruženju.

Na osnovu individualnih zadataka studenti treba da zaključe kada koji alat koristiti.

##### **Uvod u individualnu vežbu #2**

Instalirati sledeće editore koda:

- Notepad++
- Visual Studio Code

Nabaviti studentsku licencu JetBrains za PyCharm i instalirati ga.

<https://www.jetbrains.com/pycharm/>

Povezati Python interpretare gde je neophodno.

##### **Zadatak #1 (30 minuta)**

Napisati program u Python-u za zamenu dve vrednosti promenljiva tako da:

- Postoji treća promenljiva,
- Ne postoji treća promenljiva.

Program napisati u editoru koda i sačuvati kao L02\_zad01.py. Nakon toga, iz konzole pokrenuti program.

**Zadatak #2 (30 minuta)**

Napisati program koji rešava kvadratnu jednačinu. Koristiti Visual Studio Code ili PyCharm.

$$ax^2 + bx + c = 0$$
$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

**Zadatak #3 (30 minuta)**

Napisati program za pronalaženje faktorijela broja. Faktorijel nule je jedan.

$$n! = n \times (n - 1) \times \cdots \times 2 \times 1$$

## ✓ Poglavlje 8

### Domaći zadatak #2

#### DOMAĆI ZADATAK

*Domaći zadatak #2 se okvirno radi 2h*

##### **Domaći zadatak #2**

Na kućnom računaru instalirati Notepad++, Visual Studio Code sa Python ekstenzijom, kao i PyCharm IDE okruženje.

Povezati Python interpretare gde je neophodno.

Screenshot-ovima pokazati da su interpretari povezani i da možete pokretati pisane programe.

Napisati program koji će ispisivati sve proste brojeve u proizvoljnom intervalu.

Koristiti uputstvo za izradu domaćeg zadatka dato u prethodnoj lekciji. Uz dokument dostaviti i .py izvorni kod programa.

##### **Predaja domaćeg zadatka:**

##### **Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije nedelju dana nakon predavanja za 100% poena. Nakon toga poeni se umanjuju za 50%.

##### **Online studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganja ispita. **Domaći zadaci se brane!**

Svi studenti domaći zadatak poslati dr Nemanji Zdravkoviću:  
[nemanja.zdravkovic@metropolitan.ac.rs](mailto:nemanja.zdravkovic@metropolitan.ac.rs)

## ▼ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Zaključak lekcije #2*

#### **Rezime:**

U ovoj lekciji objašnjena je najpre istorija Python jezika, kao i dalji razvoj koji i danas traje.

Zatim objašnjene su razlike dva izdanja Python jezika koja se i dalje koriste, Python 2.x i Python 3.x, kao i razlozi zbog koje treba preći na 3.x.

Opisane su primene Python programskoj jeziku, u kojima se vidi da se Python, iako skripting jezik, može koristiti za opštu upotrebu bez velikih poteškoća.

Konačno, bilo je reči o editorima koda i IDE razvojnim okruženjima za Python.

U vežbi opisan je proces instalacije editora koda i IDE okruženja uz odgovarajuće primere.

#### **Literatura:**

1. David Beazley, Brian Jones, *Python Cookbook: Recipes for Mastering Python 3*, 3rd edition, O'Reilly Press, 2013.
2. Mark Lutz, *Learning Python*, 5th Edition, O'Reilly Press, 2013.
3. Andrew Bird, Lau Cher Han, et. al, *The Python Workshop*, Packt Publishing, 2019.
4. Al Sweigart, *Automate the boring stuff with Python*, 2nd Edition, No Starch Press, 2020.





## CS324 - SKRIPTING JEZICI

Proceduralno programiranje -  
promenljive, petlje, kontrola toka

Lekcija 03

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 03

### ***PROCEDURALNO PROGRAMIRANJE - PROMENLJIVE, PETLJE, KONTROLA TOKA***

- ✓ Proceduralno programiranje - promenljive, petlje, kontrola toka
- ✓ Poglavlje 1: Vrste promenljivih
- ✓ Poglavlje 2: Imenici i tuple-ovi
- ✓ Poglavlje 3: Tipiziranje podataka u Python 3.x jeziku
- ✓ Poglavlje 4: Kontrola toka
- ✓ Poglavlje 5: Petlje
- ✓ Poglavlje 6: Pokazna vežba #3
- ✓ Poglavlje 7: Individualna vežba #3
- ✓ Poglavlje 8: Domaći zadatak #3
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ▼ Uvod

### UVOD

*U trećoj lekciji govoriće se o proceduralnom programiranju u Python-u, sa akcentom na vrste promenljivih, kontroli toka, i o petljama.*

U trećoj lekciji govoriće se o proceduralnom programiranju u Python-u. Pre nego što se pređe na koncepte objektno-orientisanog programiranja, veoma je bitno da se savlada proceduralno programiranje.

OOP možda jeste napredniji koncept, međutim neki zadaci, pogotovo u primeni Python-a u inženjerskim aplikacijama, proceduralno napisan kod bolje rešava.

*Podsetimo se nekih od osobina proceduralnog programiranja:*

Ključni koncept jesu procedure, odnosno funkcije, koje predstavljaju instrukcije koja su definisane nazivom.

Takođe, u radu sa funkcijama treba obratiti pažnju na *lokalne* i *globalne* promenljive. Lokalne važe samo u opsegu strukture podataka u kojoj su definisane, dok globalne promenljive se deklarišu izvan ostalih funkcija.

U ovoj lekciji, osnovni rad sa promenljivima biće predstavljen, kako sa numeričkim tako i sa alfa-numeričkim. Biće reči i o kontroli toka, i petljama i kako se predstavljaju u Python 3.x jeziku.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 1

### Vrste promenljivih

## STRINGOVI U PYTHON 3.X JEZIKU

*String tip podataka predstavlja niz slova, brojeva, simbola i razmaka.*

String tip podataka predstavlja niz slova, brojeva, simbola i razmaka.

U Python 3.x programskom jeziku, stringovi mogu biti gotovo bilo koje dužine i mogu sadržati razmake (en. **space**), jer je i razmak validan karakter u Python jeziku. Stringovi mogu stajati između apostrofa ili između navodnika.

#### **Brojevi kao stringovi**

Brojevi (celi i razlomljeni) se takođe mogu definisati kao stringovi. Ukoliko je ceo, razlomljen ili kompleksni broj definisan pod apostrofima, čuva se kao string a ne kao broj.

#### **Logičke vrednosti kao stringovi**

Stringovi se mogu konvertovati u logičke promenljive korišćenjem `bool()` funkcije. Jedino prazan string `""` vraća `False`, a svi ostali vraćaju `True`.

*String koji sadrži samo razmak, " ", nije prazan string i vratiće True, jer je razmak (blank, space) validan karakter.*

#### **Indeksiranje stringova**

Indeksiranjem stringova može se izvući konkretni karakter iz stringa u određenom redosledu.

U Python programskom jeziku, stringovi se indeksiraju korišćenjem srednjih zagrada `[ ]`.

***U Python jeziku prvi indeks je 0 a poslednji je n-1.***

#### **Negativno indeksiranje stringova**

Ukoliko u srednjim zagradama stavimo negativni broj, onda će se izvući karakter stringa od poslednjeg karaktera.

#### **Opseg indeksiranja**

Može se izvući sekvenca karaktera iz stringa. Ukoliko između srednjih zagrada stavimo prvi indeks, pa dvotačku, pa krajnji indeks, onda će se izvući karakteri unuter tog opsega.

## UVOD U TIPOVE PODATAKA

*Najčešće grupe tipova podataka jesu numerički, ne-numerički i Boolean tip*

### **Podsetnik:**

*Tip podataka predstavlja klasifikaciju ili kategorizaciju podataka. Predstavlja tip vrednosti koji određuje koje operacije se mogu izvršiti nad tim podacima.*

Tip podataka, tj. vrsta promenljive, obično se grupiše u 3 ili više grupa.

### **Najčešće grupe tipova podataka.**

- Numerički tip podatak
- Ne-numerički tip podataka
- Boolean tip podataka (True/False)

U okviru svake grupe mogu postojati više tipova podataka, opet, u zavisnosti koji je programski jezik u pitanju.

Kod nekih programskih jezika (poput C jezika), pravi se razlika između tipa podataka *char* (jedan karakter) i *string* (više karaktera).

*Svaki programski jezik ima svoje klasifikacije koje opisuju filozofiju tog programskog jezika.*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

*Koristeći funkciju **type()** koja vraća tip podataka uvek možemo proveriti o kojem je tipu podataka reč.*

## ▼ 1.1 Numeričke promenljive

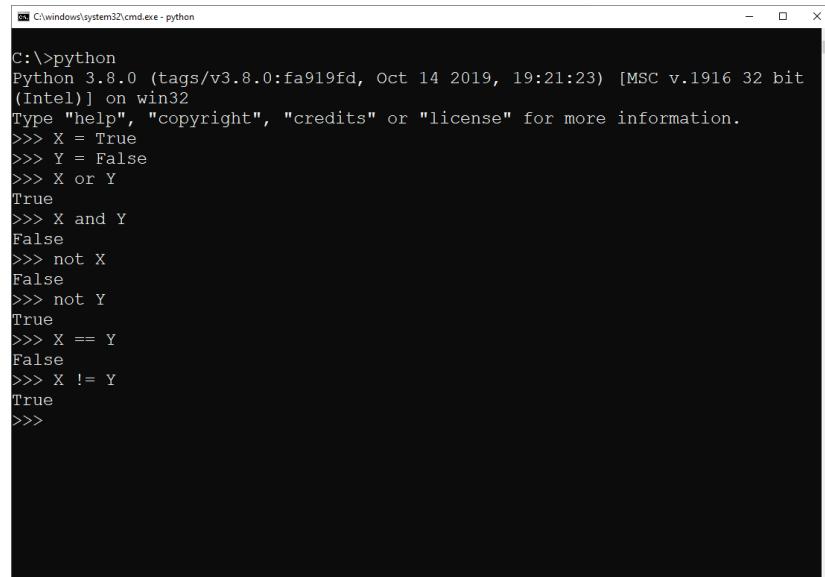
### CELOBROJNI I RAZLOMLJENI TIPOVI PODATAKA

*Python 3.x jezik podržava celobrojne i razlomljene numeričke tipove podataka.*

#### **Celobrojni tip podataka**

Celobrojni tip podataka (en. *Integer*) jeste upravo to - celi broj, koji može biti pozitivan, jednak nuli, ili negativan

```
int_a = 5
type(int_a)
int_b = -3
type(int_b)
z = 0
type(z)
```



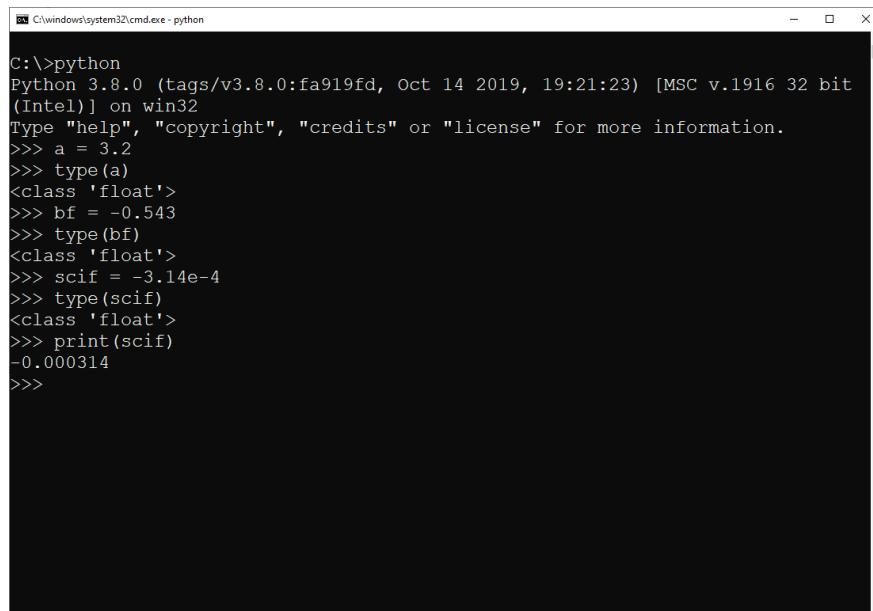
```
C:\>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> X = True
>>> Y = False
>>> X or Y
True
>>> X and Y
False
>>> not X
False
>>> not Y
True
>>> X == Y
False
>>> X != Y
True
>>>
```

Slika 1.1.1 Celobrojni tip podataka. [Izvor: Autor]

## Razlomljeni tip podataka

**Razlomljeni tip podataka** (en. **floating point**) jesu brojevi sa decimalnom tačkom, koji takođe mogu biti pozitivni, jednaki nuli, i negativni.

Pored decimalne notacije, mogu se predstaviti i u *obliku sa eksponentom* (en. **scientific notation**), u kojoj eksponent može biti označen bilo malim bilo velikim slovom e.



```
C:\>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 3.2
>>> type(a)
<class 'float'>
>>> bf = -0.543
>>> type(bf)
<class 'float'>
>>> scif = -3.14e-4
>>> type(scif)
<class 'float'>
>>> print(scif)
-0.000314
>>>
```

Slika 1.1.2 Razlomljeni tip podataka. [Izvor: Autor]

## KOMPLEKSNI TIPOVI PODATAKA

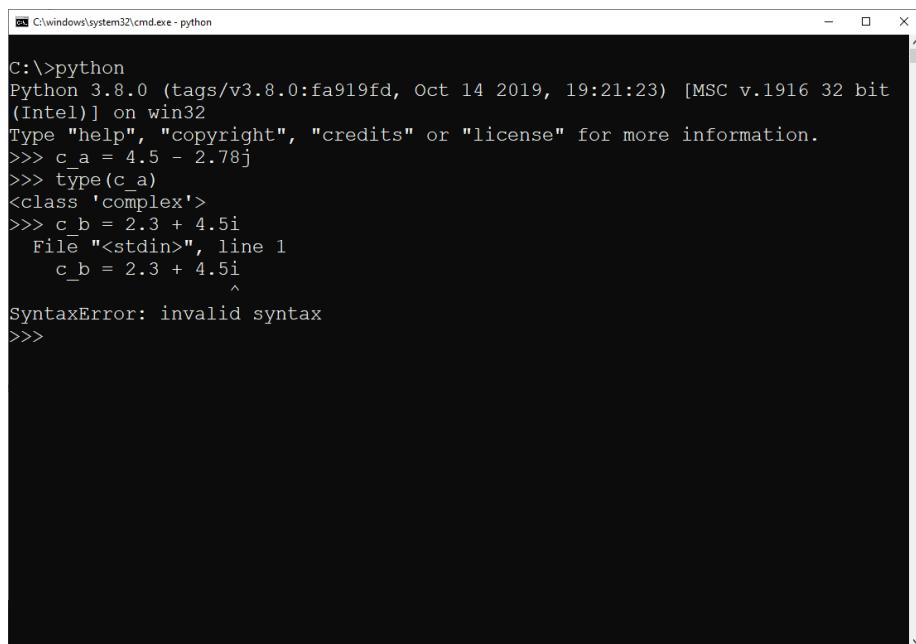
*Python 3.x jezik podržava i kompleksne numeričke tipove podataka.*

### Kompleksni tip podataka

Kompleksni tip podataka (en. **complex number**) odnosi se na kompleksni broj koji ima realni i imaginarni deo.

Kompleksni brojevi se u Python 3.x jeziku pišu kao *realni\_deo + imaginarni\_deo \* j*.

Koristi se *j* kao oznaka za imaginarnu jedinicu, dok ukoliko se koristi *i*, Python će vratiti grešku.



```
C:\>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> c_a = 4.5 - 2.78j
>>> type(c_a)
<class 'complex'>
>>> c_b = 2.3 + 4.5i
      File "<stdin>", line 1
          c_b = 2.3 + 4.5i
          ^
SyntaxError: invalid syntax
>>>
```

Slika 1.1.3 Kompleksni tip podataka. [Izvor: Autor]

## ✓ 1.2 Logičke promenljive - Boolean

### TIP PODATAKA BOOLEAN

*Svaki programski jezik, pa i Python, ima logički tip podataka Bool (Boolean)*

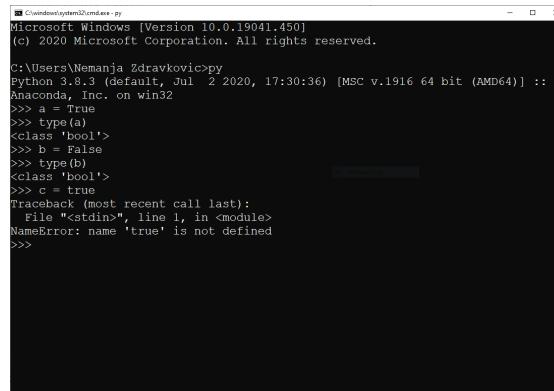
Kao i kod gotovo svih ostalih programskih jezika, Python poseduje logički tip podataka koji može imati vrednost tačno (en. **true**) i netačno (en. **false**).

U Python 3.x jeziku, dodela Boolean vrednosti razlikuje veličinu slova (en. case sensitive), tako da je prvo slovo uvek veliko.

**Primer:**

Promenljivama **a** i **b** se pravilno dodeljuje logička vrednost, dok promenljivoj **c** se loše dodeljuje, što javlja grešku.

```
a = True
type(a)
b = False
type(b)
c = true
```



```
C:\Users\Nemanja Zdravkovic>py
Microsoft Windows [Version 10.0.19041.450]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Nemanja Zdravkovic>py
Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
>>> a = True
>>> type(a)
<class 'bool'>
>>> b = False
>>> type(b)
<class 'bool'>
>>> c = true
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'true' is not defined
>>>
```

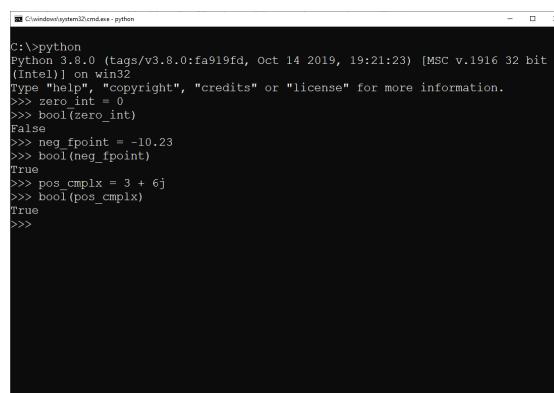
Slika 1.2.1 Primer za Boolean tipa podataka u Python 3.x. [Izvor: Autor]

### Celobrojni i razlomljeni podaci kao logičke vrednosti

Celi i razlomljeni brojevi se mogu konvertovati u logički tip podataka koristeći funkciju `bool()`. Izlaz funkcije jeste True ili False.

Celi, razlomljeni i kompleksni brojevi koji su jednaki nuli vraćaju `False`, dok ostale vrednosti (i pozitivne i negativne) vraćaju `True`.

```
zero_int = 0
bool(zero_int)
neg_fpoint = -10.23
bool(neg_fpoint)
pos_cmplx = 3 + 6j
bool(pos_cmplx)
```



```
C:\>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> zero_int = 0
>>> bool(zero_int)
False
>>> neg_fpoint = -10.23
>>> bool(neg_fpoint)
True
>>> pos_cmplx = 3 + 6j
>>> bool(pos_cmplx)
True
>>>
```

Slika 1.2.2 `bool()` funkcija koja vraća logičke vrednosti. [Izvor: Autor]

## ARITMETIKA NAD BOOLEAN TIPOM PODATAKA

*U Pythonu je moguće raditi osnovne logičke operacije (logička aritmetika) nad podacima tipa Boolean.*

Nad Boolean tipom podataka se mogu vršiti logičke operacije sabiranja, množenja i negacije.

### Operatori u Python jeziku nad Boolean tipom podataka

- Logičko ili - *or*
- Logičko i - *and*
- Logičko ne - *not*
- Jednako - *==*
- Nije jednako - *!=*

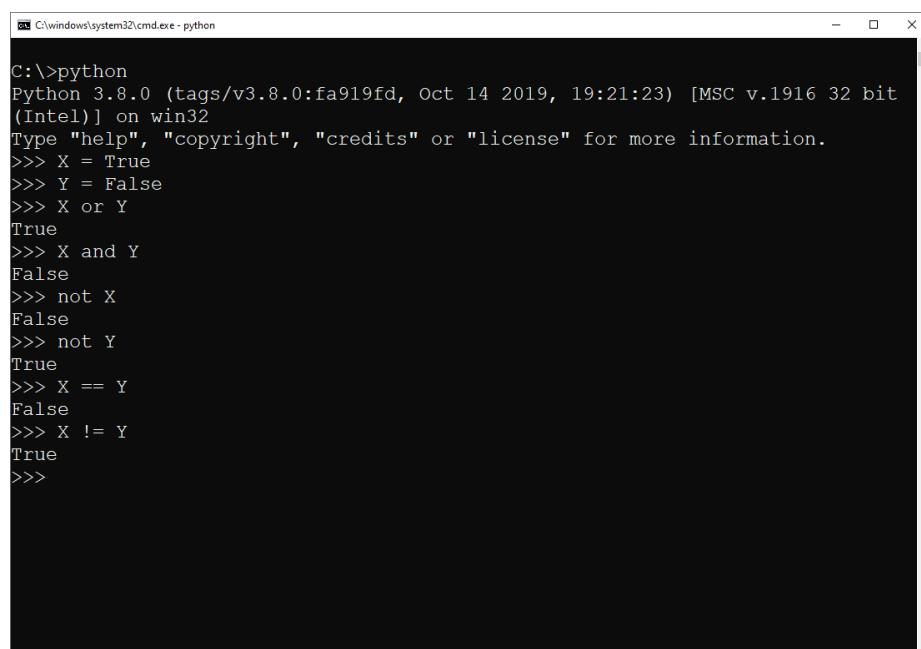
Kroz kratke primere dodelićemo logičke vrednosti promenljivima X i Y i izvršiti logičke operacije nad njima.

```
X = True
Y = False

X or Y
X and Y

not X
not Y

X == Y
X != Y
```



```
C:\>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> X = True
>>> Y = False
>>> X or Y
True
>>> X and Y
False
>>> not X
False
>>> not Y
True
>>> X == Y
False
>>> X != Y
True
>>>
```



Slika 1.2.3 Operacije nad Boolean tipovima podataka. [Izvor: Autor]

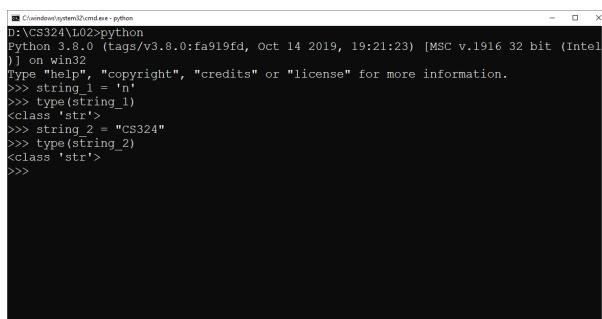
## 1.3 Stringovi i liste

### PRIMERI SA STRINGOVIMA U PYTHON 3.X JEZIKU

*Primeri u radu sa stringovima u Python jeziku*

#### Primer #1 - Definisanje stringova koji sadrže jedan i više karaktera

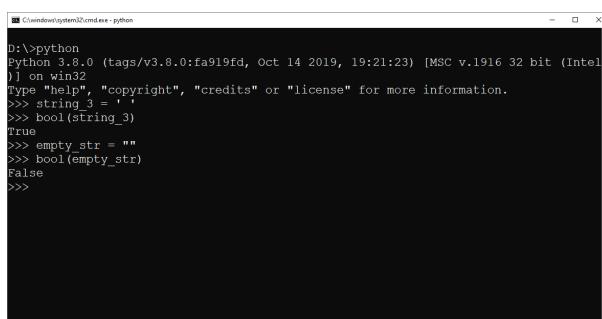
```
string_1 = 'n'  
type(string_1)  
  
string_2 = "CS324"  
type(string_2)
```



Slika 1.3.1 Deklarisanje stringova. [Izvor: Autor]

#### Primer #2 - Stringovi kao logičke vrednosti i kao brojevi

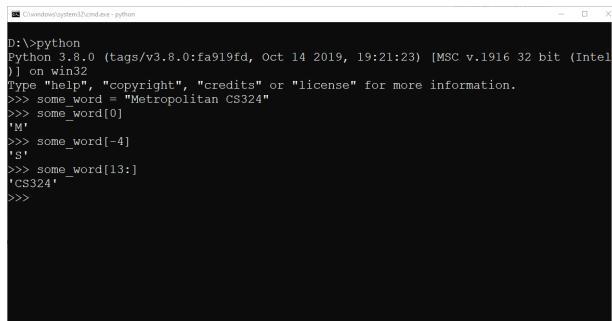
```
string_3 = ''  
bool(string_3)  
  
empty_str = ""  
bool(empty_str)
```



Slika 1.3.2 Stringovi i logičke vrednosti. [Izvor: Autor]

#### Primer #3 - Pozitivno i negativno indeksiranje stringova i opsezi indeksiranja

```
some_word = "Metropolitan CS324"  
  
some_word[0]  
some_word[-4]  
some_word[13:]
```



Slika 1.3.3 Indeksiranje stringova. [Izvor: Autor]

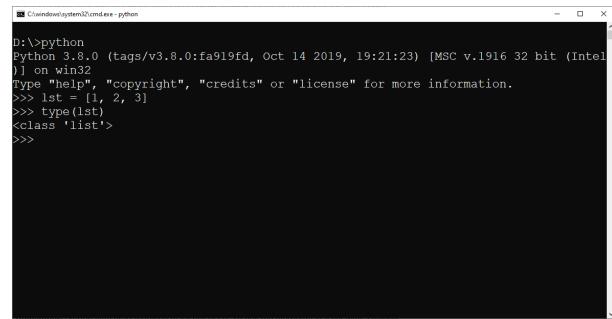
## LISTE U PYTHON 3.X JEZIKU

*Lista može sadržati više elemenata od kojih svaki element može pripadati različitom tipu podataka.*

List tip podataka jeste struktura podataka u Python programskom jeziku koja može sadržati više elemenata bilo kojeg drugog tipa podataka.

**Lista je definisana srednjim zagradama [ ], a elementi liste se razdvajaju zarezima.**

```
lst = [1, 2, 3]  
type(lst)
```

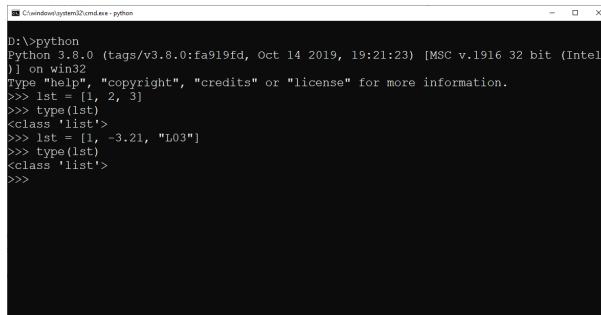


Slika 1.3.4 Pravljenje liste u Python jeziku. [Izvor: Autor]

### Indeksiranje lista

Kao i kod stringova, pojedinačni elementi liste se mogu pristupiti indeksiranjem putem srednjih zagrada.

```
lst = [1, -3.21, "L03"]
type(lst)
lst[2]
type(lst[1])
```



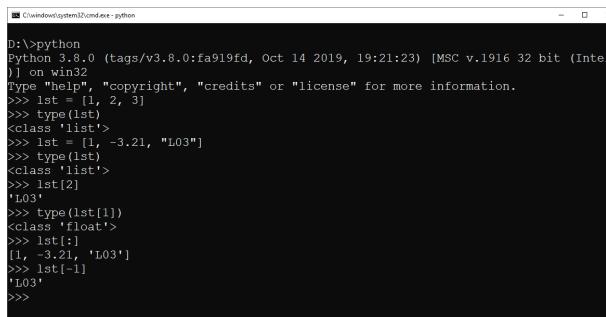
Slika 1.3.5 Indeksiranje liste. [Izvor: Autor]

### Opsezi indeksiranja lista

Moguće je indeksirati više elemenata liste korišćenjem dvotačke.

Ukoliko treba indeksirati sve elemente liste, u srednje zagrade se stavlja samo dvotačka, odnosno za listu *lst*, svi elementi prikazaće se ukoliko se napiše *lst[:]*.

Isto važi i za negativno indeksiranje kao kod stringova, ali i opseg.



Slika 1.3.6 Indeksiranje lista u određenom opsegu i negativno indeksiranje lista. [Izvor: Autor]

## ▼ Poglavlje 2

### Imenici i tuple-ovi

## TUPLE PODACI U PYTHON 3.X JEZIKU

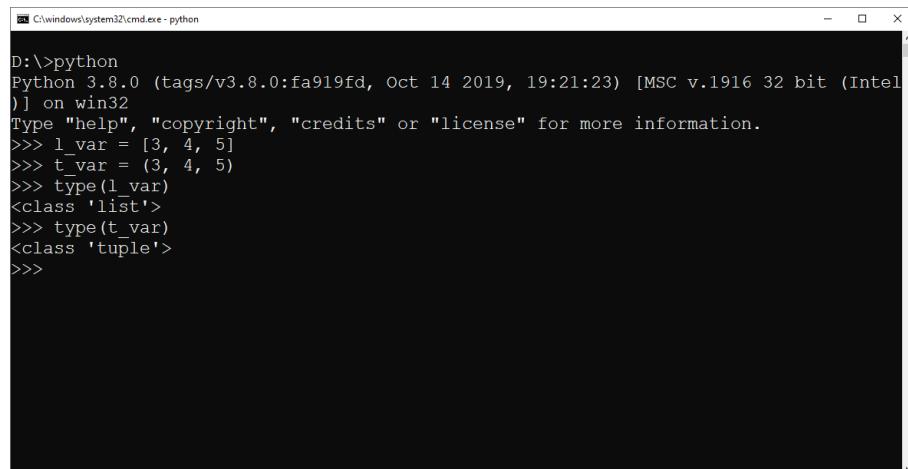
*Lista sa nepromenljivim elementima predstavlja tuple.*

Tuple tip podataka predstavlja listu koja je nepromenljiva. Za razliku od obične liste čiji se elementi mogu modifikovati, elementi u tuple tipu podataka se samo mogu pristupiti, ali ne i modifikovati. Tuple podaci se definišu u Python jeziku tako što su elementi tuple-a u malim zagradama, a pojedinačni elementi su razdvojeni zapetama.

#### Primer: Poređenje liste i tuple tipa podataka

Napomena. funkcija `type()` vraća tip podataka argumenta.

```
l_var = [3, 4, 5]
t_var = (3, 4, 5)
type(l_var)
type(t_var)
```

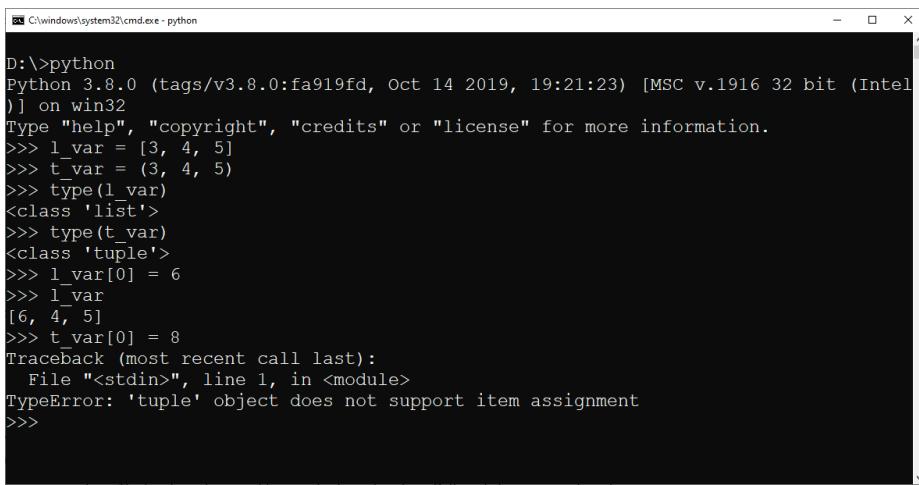


```
D:\>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)]
] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> l_var = [3, 4, 5]
>>> t_var = (3, 4, 5)
>>> type(l_var)
<class 'list'>
>>> type(t_var)
<class 'tuple'>
>>>
```

Slika 2.1 Definisanje elemenata liste i tuple-a. [Izvor: Autor]

Ukoliko želimo da promenimo jedan element tuple-a, to neće biti moguće, dok za listu neće vratiti nikakvu grešku.

```
l_var[0] = 8
l_var
t_var[0] = 8
```



```
D:\>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel
)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> l_var = [3, 4, 5]
>>> t_var = (3, 4, 5)
>>> type(l_var)
<class 'list'>
>>> type(t_var)
<class 'tuple'>
>>> l_var[0] = 6
>>> l_var
[6, 4, 5]
>>> t_var[0] = 8
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
```

Slika 2.2 Promena elemenata u listi i neuspeli pokušaj u tuple-u. [Izvor: Autor]

**Pitanje:**

Kako definisati tuple koji ima samo jedan element?

## IMENICI U PYTHON 3.X JEZIKU

*Imenik, za razliku od liste, sastoji se od para podataka: ključa i vrednosti.*

Imenik (en. **dictionary**) je tip podataka koji se sastoji od para podataka, i to po šablonu **ključ: vrednost**.

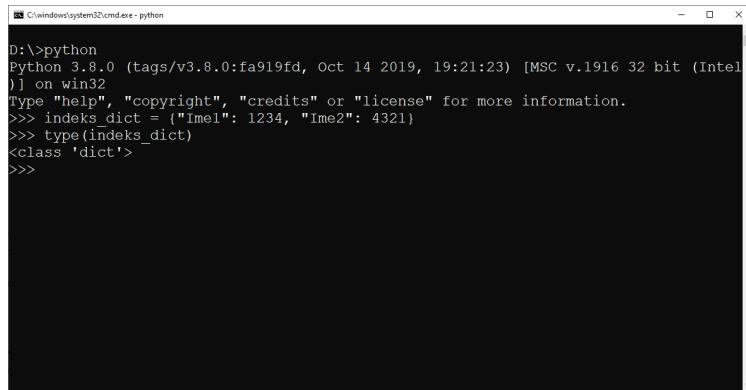
Imenicima u Python jeziku su organizovani koristeći par ključa i vrednosti, i tako im se i pristupa.

*Lokacija para ključ-vrednost u imeniku nije bitna.*

Imenici se u Python jeziku definišu u velikim zagradama `{}`. Zapere razdvajaju ove parove, a svaki par je povezan dvotačkom.

### Primer - Definisanje imenika

```
indeks_dict = {"Ime1": 1234, "Ime2": 4321}
type(indeks_dict)
```



```
D:\>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)]
) on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> indeks_dict = {"Ime1": 1234, "Ime2": 4321}
>>> type(indeks_dict)
<class 'dict'>
>>>
```

Slika 2.3 Definisanje imenika. [Izvor: Autor]

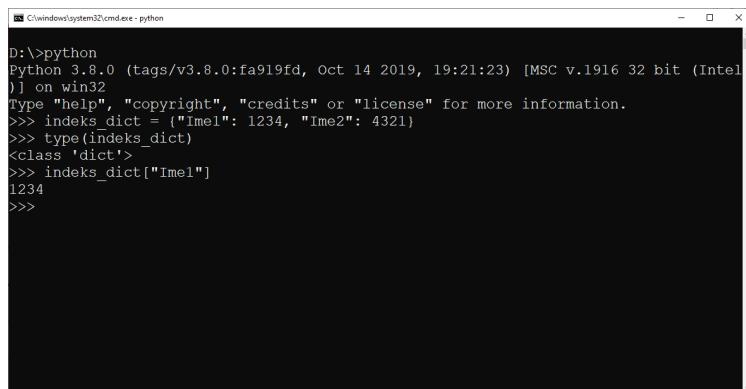
### Pristupanje vrednostima

Vrednostima unutar imenika se pristupa sledećom sintaksom

```
dict_name[key] = value
```

Primer - pristupanje vrednosti imenika

```
indeks_dict["Ime1"]
```



```
D:\>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)]
) on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> indeks_dict = {"Ime1": 1234, "Ime2": 4321}
>>> type(indeks_dict)
<class 'dict'>
>>> indeks_dict["Ime1"]
1234
>>>
```

Slika 2.4 Pristupanje vrednosti imenika. [Izvor: Autor]

### Pitanje:

Kako se imenici mogu pretvoriti u liste?

## DODAVANJE I ODUZIMANJE ELEMENTA U IMENIKU

*U Python jeziku se lako dodaju novi elementi imenika, a korišćenjem .pop() metode.*

### Dodavanje elemenata imeniku

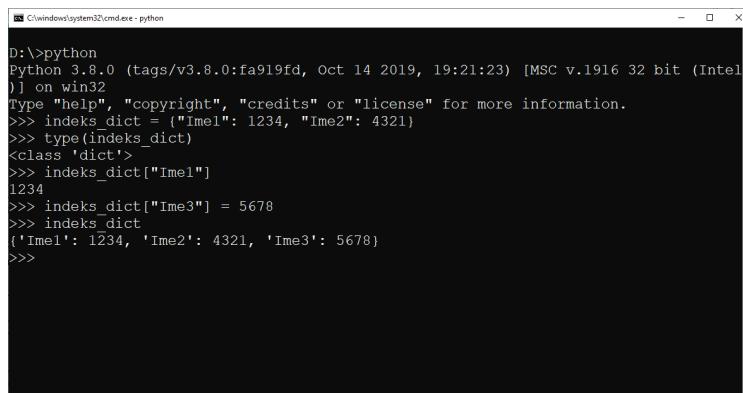
Dodavanje novog elementa postojećem imeniku može se uraditi upotrebom sledeće sintakse:

```
dict_name[new_key] = new_value
```

### Primer: Dodavanje elementa imeniku

Na primeru **indeks\_dict** dodaće se još jedan element

```
indeks_dict["Ime3"] = 5678
```



```
C:\Windows\system32\cmd.exe - python
D:\>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel
)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> indeks_dict = {"Ime1": 1234, "Ime2": 4321}
>>> type(indeks_dict)
<class 'dict'>
>>> indeks_dict["Ime1"]
1234
>>> indeks_dict["Ime3"] = 5678
>>> indeks_dict
{'Ime1': 1234, 'Ime2': 4321, 'Ime3': 5678}
>>>
```

Slika 2.5 Dodavanje elementa imeniku [Izvor: Autor]

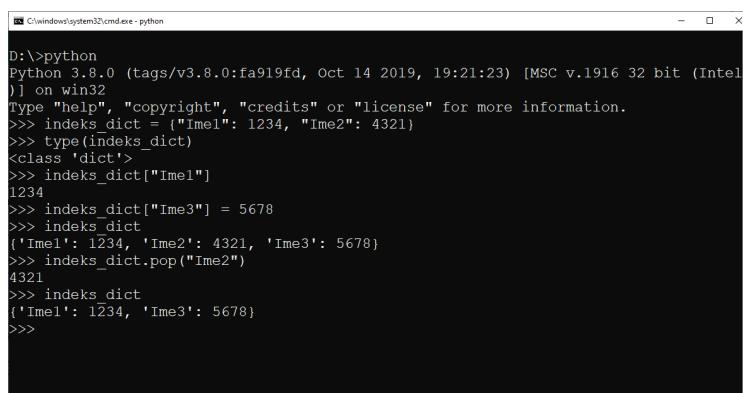
### Oduzimanje elementa imenika

Elementi imenika se mogu oduzeti pozivanjem `.pop()` metode. Vrednost ključa koja se prosleđuje metodi `.pop()` određuje koji će se par ključ-vrednost izbaciti.

### Primer: Oduzimanje elementa imeniku

Na primeru `indeks_dict` oduzeće se jedan element

```
indeks_dict.pop("Ime2")
```



```
C:\Windows\system32\cmd.exe - python
D:\>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel
)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> indeks_dict = {"Ime1": 1234, "Ime2": 4321}
>>> type(indeks_dict)
<class 'dict'>
>>> indeks_dict["Ime1"]
1234
>>> indeks_dict["Ime3"] = 5678
>>> indeks_dict
{'Ime1': 1234, 'Ime2': 4321, 'Ime3': 5678}
>>> indeks_dict.pop("Ime2")
4321
>>> indeks_dict
{'Ime1': 1234, 'Ime3': 5678}
>>>
```

Slika 2.6 Oduzimanje elementa imeniku. [Izvor: Autor]

### Pitanje:

U čemu je razlika između funkcije i metode?

## ▼ Poglavlje 3

# Tipiziranje podataka u Python 3.x jeziku

## PODELA TIPIZIRANJA

*Postoje jako i slabo tipizirani programski jezici, kao i statički i dinamički tipizirani. Svaki programski jezik spada u jednu od četiri kombinacije.*

Pri procesu izvršenja programa (bilo pri kompajliranju ili interpretiranju), vrši se provera tipa podataka (en.type checking).

***Ukoliko se proces provere tipa podataka izvršava u vremenu kompajliranja, onda se vrši statička provera.***

***Ukoliko se proces provere tipa podataka izvršava u vremenu interpretiranja, onda se vrši dinamička provera.***

Provera tipa podataka vrši se u cilju da program bude *tipski* siguran (en. type-safe), što umanjuje mogućnost za greškama.

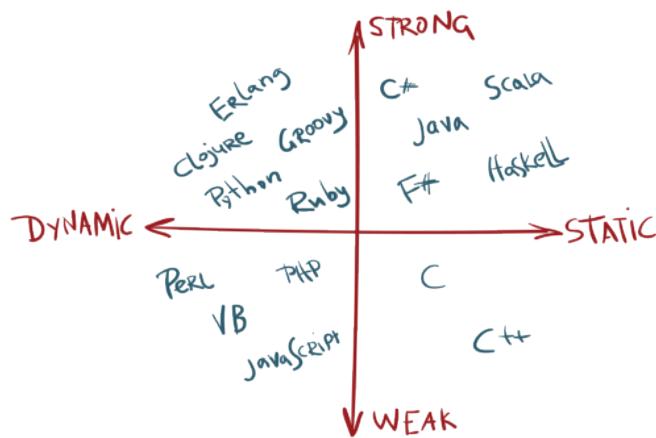
Kada program nije tipski siguran, javiće se greška u tipiziranju, i u zavisnosti koji se programski jezik (i koji kompajler ili interpreter) koristi, greška će obično zaustaviti izvršenje programa.

Postoje dve podele tipiziranja podataka.

Jedna podela jeste na *jako tipizirane* (en. **strong typed**) i *slabo tipizirane* (en. **weak typed**) programske jezike.

Druga podela jeste na *statičko tipizirane* (en.**statically typed**) i *dinamičko tipizirane* (en. **dynamically typed**) programske jezike.

Po ovoj podeli, svaki programski jezik spada u jednu od četiri moguće kombinacije.



Slika 3.1 Podela programskih jezika po načinu tipiziranja podataka. [Izvor: <https://itnext.io>]

U nastavku biće reči o svakoj od ovih kategorija tipiziranja.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## JAKO I SLABO TIPIZIRANJE

*Kod jako tipiziranih programskih jezika, vrednost promenljive mora da se poklopi sa tipom podataka koji se očekuje.*

### Jako tipiziran programski jezik

Kod programskega jezika koji spada u kategoriju jako tipiziranog jezika promenljive su vezane za konkretni tip podataka.

Prilikom kompajliranja ili interpretacije, ukoliko se vrednost promenljive ne poklopi sa tipom podataka koja se očekuje u izrazu.

**Primer: Dodavanje numeričke vrednosti ne-numeričkoj promenljivoj.**

```
temp = "Hello World!"
temp = temp + 10
```

Ukoliko pokušamo ovako nešto u programskom jeziku koji je jako tipiziran, doći će do greške.

```
C:\>python
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> temp = "Hello World"
>>> temp + 10
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
>>>
```

Slika 3.2 Greška tipiziranja u Python 3.x jeziku. [Izvor: Autor]

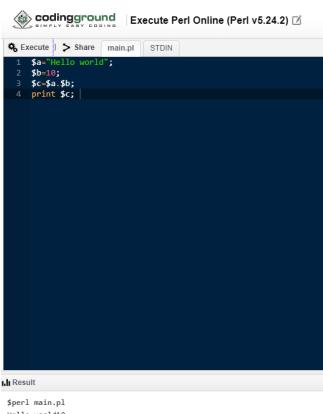
Programski jezici koji imaju jako tipiziranje jesu Java, Python, C#, Ruby, Haskell, dok slabo tipiziranje imaju C, C++, Perl, PHP, JavaScript.

### Slabo tipiziran programski jezik

Kod programske jezike koji spada u kategoriju slabo tipiziranoj jezika, promenljive nisu vezane za konkretni tip podataka. Promenljive i dalje imaju tip podataka, ali su ograničenja manja u odnosu na jako tipizirane jezike. Sledeci primer daje sintaksu u *Perl* jeziku.

#### Primer: Dodavanje numeričke vrednosti ne-numeričkoj promenljivoj.

```
$a="Hello world";
$b=10;
$c=$a.$b;
print $c;
```



The screenshot shows a web-based Perl code editor and executor. The code in the editor is:

```
1 $a="Hello world";
2 $b=10;
3 $c=$a.$b;
4 print $c;
```

The 'Result' section shows the error output:

```
$perl main.pl
Hello world10
```

Slika 3.3 Ne dolazi do greške tipiziranja u Perl jeziku. [Izvor: Autor]

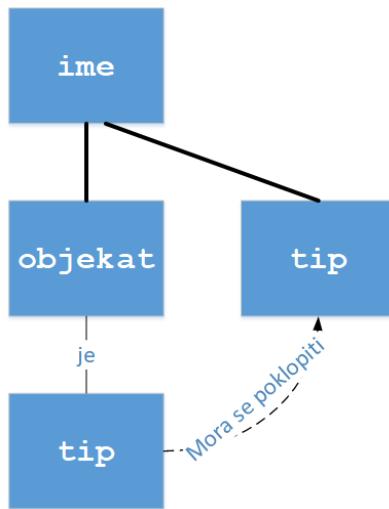
## STATIČKO TIPIZIRANJE

*Kod staticko tipiziranih programskih jezika, promenljivoj se ne može nakdnano dodeliti drugi tip podataka.*

### Statičko tipiziran programski jezik

Kod programske jezike koji spada u kategoriju staticko tipiziranog jezika, tip promenljive je poznat u vremenu kompajliranja (en. *compile-time*) umesto u vremenu izvršenja (en. *run-time*).

Ovo znači da kada se deklariše tip podataka promenljivoj, ne može se dodeliti drugom tipu podataka. Ukoliko se to desi, u vremenu kompajliranja desiće se ponovo greška u tipiziranju.



Slika 3.4 Ograničenja tipiziranja. [Izvor: Autor]

### Primer: Promena tipa podataka u Javi

```
int data;  
data = 50;  
data = "Hello World!";
```

```
codingground | Compile and Execute Java Online (JDK 1.8.0)  
Execute | Share | Source File | STDIN  
1- public class HelloWorld{  
2-  
3-     public static void main(String []args)  
4-     {  
5-         int data;  
6-         data = 50;  
7-         data = "Hello World!";  
8-     }  
9- }  
  
Result  
$javac HelloWorld.java  
HelloWorld.java:7: error: incompatible types: String cannot be converted to int  
    data = "Hello World!";  
           ^  
1 error
```

Slika 3.5 Greška u tipiziranju. [Izvor: Autor]

## DINAMIČKO TIPIZIRANJE

*Kod dinamičko tipiziranih programskih jezika, promenljivoj se može naknadno dodeliti drugi tip podataka.*

## Dinamičko tipiziran programski jezik

Kod programskog jezika koji spada u kategoriju dinamičko tipiziranog jezika, tip promenljive proverava u vremenu izvršenja programa.

Ovo efektivno znači da su promenljive vezane za tip podataka tek kada se vrši izvršenje programa, i moguće je vezati iste promenljive za različite tipove podataka.



Slika 3.6 Nepostojanje ograničenja tipiziranja. [Izvor: Autor]

Provera tipiziranja u dinamičko tipiziranom jeziku rezultuje u manje optimalnom kodu u odnosu na programske jezike sa statičkom proverom tipiziranja.

Posledica jeste mogućnost pojave grešaka tipiziranja u vremenu izvršenja, odnosno svakog tupa kada se program izvrši.

### Primer: Promena tipa podataka u Python 3.x jeziku

```
data = 10
data = "Hello World!"
```

Slika 3.7 Nema greške u tipiziranju. [Izvor: Autor]

## KADA KORISTITI STATIČKO, A KADA DINAMIČKO TIPIZIRANJE?

*Nekada je povoljnije koristiti programske jezike koji imaju statičko tipiziranje, ali nekad prevladavaju povoljnosti dinamičkog tipiziranja.*

Nekada je povoljnije koristiti programske jezike koji imaju statičko tipiziranje, ali nekad prevladavaju povoljnosti dinamičkog tipiziranja.

Prednosti statičkog tipiziranja

- Velika grupa grešaka koja se javlja može se primetiti (i otkloniti) u ranoj fazi razvoja programa.
- Kompajlirani kod se izvršava brže jer kompajler tačno zna koji se tipovi podatka koriste, i na taj način može napisati optimizovan mašinski kod.

Prednosti dinamičkog tipiziranja

- Prilikom izvršenja programa, svaki objekat ima svoj tag odnosno referencu o tipu (en.**type reference**). Ova informacija (en. **run-time type information**, RTTI) se može iskoristiti za implementaciju dodatnih funkcija.
- Nepostojanje koraka kompajliranja omogućava da se kod može izmeniti a da se ne čeka svaki put na kompajliranje. Ovo čini ciklus debagovanja kraćim.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## IME PROMENLJIVE I OBJEKTI U MEMORIJI U PYTHON JEZIKU

*Treba znati razliku između imena promenljive, objekta koji sadrži vrednost, i njihovu vezu.*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 4

### Kontrola toka

## ŠTA PREDSTAVLJA KONTROLA TOKA U PROGRAMIRANJU?

*Kontrola toka je eksplicitna u imperativnom programskom jeziku kao što je Python 3.x.*

Kontrola toka (en. **control flow**) predstavlja redosled po kojim pojedinačne naredbe, izrazi, pozivi funkcija i instrukcije se pozivaju prilikom izvršenja programa pisanog u imperativnom programskom jeziku.

#### **Podsetnik:**

*Eksplicitna kontrola toka jeste ono što razlikuje imperativni programski jezik od deklarativnog programskog jezika.*

U imperativnom programskom jeziku, naredba za kontrolu toka daje opciju o biranju dve ili više putanje.

#### **Mehanizmi kontrole toga niskog nivoa**

Prekid (en. **interrupt**) i signal predstavljaju mehanizme kontrole toga niskog nivoa koje mogu promeniti tok izvršenja programa na sličan način kao i pod-rutina (en. **subroutine**).

Za razliku od in-line naredbe za kontrolu toga, mehanizmi niskog nivoa obično se dešavaju po nekom eksternom pozivu ili događaju, koji je obično asinhron.

Na nivou mašinskog i asebmlerskog jezika, instrukcije za kontrolu toga rade tako što se menja programski brojač. Centralna procesorska jedinica (en. **central processing unit, CPU**) razume ove uslovne i bezuslovne instrukcije koje se nazivaju instrukcije skoka (en. **jump**).

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## OSNOVE KONTROLA TOKA U PYTHON 3.X JEZIKU

*U bilo kom programskom jeziku, pa i u Python 3.x, biće potrebno da kontrolišete tok programa.*

U bilo kom programskom jeziku, pa i u Python 3.x, biće potrebno da kontrolišete tok programa.

Kao što je rečeno, kontrola toka se javlja kada je potrebno da se napravi neka odluka - ukoliko je ispunjen neki uslov, izvršiće se jedan skup koda, a ako nije, izvršiće se drugi.

U Python 3.x jeziku, postoje više uslovnih naredbi (en. **conditional statements**), koje služe za kontrolu toka. To su:

- **if**
- **elif**
- **else**

Najjednostavnija naredba je, naravno **if**, nešto složeniji skup naredbi jeste **if-else**, dok **jeif-elif-else** najsloženiji od sva tri.

U nastavku lekcije biće reči i o ugnježdenim uslovnim naredbama (en. **nested conditional statements**)

### Naredba if

Najjednostavnija uslovna naredba koja proverava uslov, i vraća Boolean vrednost kao rezultat. Ukoliko je vrednost True, onda će se deo koda neposredno ispod uslova, tj. nakon ključnog simbola dvotačke **:** izvršiti.

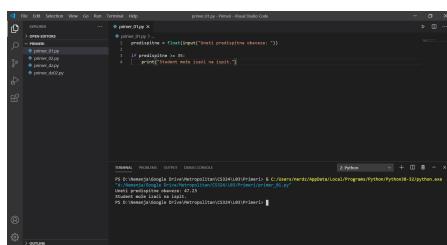
Generalna sintaksa izgleda:

```
if (uslov):  
    Deo koda ako uslov vrati True
```

### Primer - Provera uslova izlaska na ispit

Program proverava da li je unet broj predispitnih obaveza veći od 35. Ukoliko jeste, ispisuje da student može izaći na ispit.

```
predispitne = float(input("Uneti predispitne obaveze: "))  
  
if predispitne >= 35:  
    print("Student može izaći na ispit.")
```



Slika 4.1 Primer uslovne komande if. [Izvor: Autor]

# NAREDBE IF I IF-ELSE U PYTHON 3.X JEZIKU

*Kod Python 3.x jezika ključni znak dvotačka razdvaja uslov od naredbi koje treba da se izvrše posle provere uslova.*

## Specifičnosti za Python 3.x jezik

Može se primetiti da posle naredbe uslova stoji ključni znak dvotačka :

U Python 3.x jeziku dvotačka se koristi da bi razdvojila uslov od naredbi koje treba da se izvrše posle provere uslova.

### Naredbe if-else

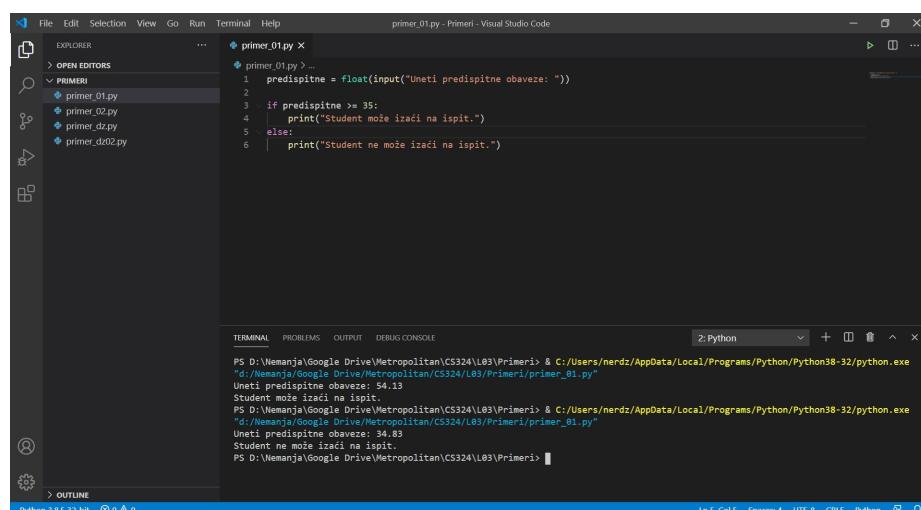
Ukoliko postoji potreba da se izvrši deo koda i kada nije ispunjen uslov (tačnije, jedan deo koda se izvršava kada uslov vrati True, a drugi deo koda se izvršava kad ulov vrati False), onda se koristi kombinacija if i else naredbi. Sintaksa je sledeća:

```
if (uslov):  
    Deo koda ako uslov vrati True  
else:  
    Deo koda ako uslov vrati False
```

### Primer - Provera uslova izlaska na ispit (nastavak)

U nastavku prilikom unošenja poena za predispitne obaveze izlaz štampa i kada student može, i kada ne može da izade na ispit.

```
predispitne = float(input("Uneti predispitne obaveze: "))  
  
if predispitne >= 35:  
    print("Student može izaći na ispit.")  
else:  
    print("Student ne može izaći na ispit.")
```



Slika 4.2 Primer uslovne komande if-else. [Izvor: Autor]

## NAREDBE IF-ELIF-ELSE

*Python podržava višestruku proveru uslova u if-elif-else skupu naredbi.*

### If-elif-else

Python podržava višestruku proveru uslova u if-elif-else skupu naredbi.

Najpre se proverava prvi uslov. Ukoliko on vrati False, proverava se uslov u elif (elif je of else-if). Ukoliko i taj uslov vrati False, onda se ide na (čistu) else granu koja izvršava liniju koda. Sintaksa je sledeća

```
if (uslov1):  
    Deo koda ako vrati True uslov1  
elif (uslov2):  
    Deo koda ako vrati False na uslov1, a True na uslov2  
else:  
    Deo koda ako vrati False na uslov1 i False na uslov2
```

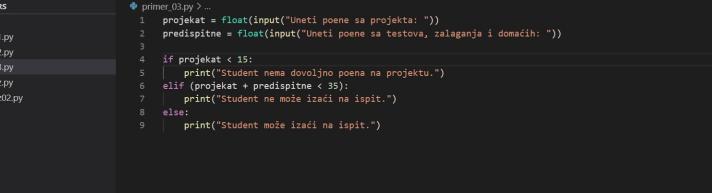
### Primer: Provera poena na predispitnim obavezama i projektu

Uneti poene sa predispitnih obaveza i projekta.

Ukoliko student nema 15 poena na projektu, izlaz javlja da nema dovoljno poena na projektu.

Ukoliko ima, provera se da li zbor poena na projektu i ostalim predispitnim obavezama iznosi barem 35. Ukoliko iznosi, izlaz javlja da student može izaći na ispit, a ukoliko nema, izlaz javlja da student ne može izaći na ispit.

```
projekat = float(input("Uneti poene sa projekta: "))  
predispitne = float(input("Uneti poene sa testova, zaloganja i domaćih: "))  
  
if projekat < 15:  
    print("Student nema dovoljno poena na projektu.")  
elif (projekat + predispitne < 35):  
    print("Student ne može izaći na ispit.")  
else:  
    print("Student može izaći na ispit.")
```



The screenshot shows the Visual Studio Code interface. The left sidebar has 'EXPLORER' open, showing a tree view of 'PRIMERI' with files 'primer\_01.py', 'primer\_02.py', 'primer\_03.py', 'primer\_dt.py', and 'primer\_d02.py'. The main area shows the code for 'primer\_03.py'. The terminal at the bottom shows the execution of the script and its output.

```
primer_03.py x
primer_03.py ...
1 projektat = float(input("Uneti poene sa projekta: "))
2 predispitne = float(input("Uneti poene sa testova, zaloganja i domaćih: "))
3
4 if projektat < 15:
5     print("Student nema dovoljno poena na projektu.")
6 elif (projektat + predispitne) < 35:
7     print("Student ne može izaći na ispit.")
8 else:
9     print("Student može izaći na ispit.")

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
2: Python
Uneti poene sa projekta: 10
Uneti poene sa testova, zaloganja i domaćih: 12
Student nema dovoljno poena na projektu.
PS D:\Nemanja\Google Drive\Metropolitan\CS324\L03\Primeri & C:/Users/nerdz/AppData/Local/Programs/Python/Python38-32/python.exe
Uneti poene sa projekta: 16
Uneti poene sa testova, zaloganja i domaćih: 12
Student ne može izaći na ispit.
PS D:\Nemanja\Google Drive\Metropolitan\CS324\L03\Primeri & C:/Users/nerdz/AppData/Local/Programs/Python/Python38-32/python.exe
Uneti poene sa projekta: 16
Uneti poene sa testova, zaloganja i domaćih: 20
Student može izaći na ispit.
PS D:\Nemanja\Google Drive\Metropolitan\CS324\L03\Primeri
```

Slika 4.3 Primer if-elif-else naredbi. [Izvor: Autor]

# PRIMER IF/ELIF/ELSE: POENI NA PREDISPITNIM OBAVEZAMA I ISPITU VIDEO

*Video na kojem je objašnjen primer korišćenja if-elif-else naredbi*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## UGNJEŽDENE USLOVNE NAREDBE

*Ukoliko želimo da imamo dodatno grananje unutar grane koda, korsitićemo ugnježđeno grananje.*

buđući da je sintaksa if-else naredbi takva:

```
if (uslov):  
    Deo koda ako uslov vrati True  
else:  
    Deo koda ako uslov vrati False
```

ništa ne sprečava programere da u *Delu koda ako uslov vrati True/False* ubaci ponovo neke uslove i if-else naredbe

Ovo se nazivaju ugnježdene uslovne naredbe. Opšti primer bio bi:

```
if (uslov1):  
    #Deo koda ako uslov1 vrati True  
    if (uslov2)  
        #Deo koda ukoliko uslov2 vrati True  
    else:  
        #Deo koda ukoliko uslov2 vrati False
```

```
else:  
    #Deo koda ako uslov1 vrati False  
    if (uslov3)  
        #Deo koda ukoliko uslov3 vrati True  
    else:  
        #Deo koda ukoliko uslov3 vrati False
```

**Podestnik:**

*Bitno je voditi računa da kod koji se piše bude pregledan za kasnije čitanje, bilo od Vas kao autora koda, bilo od strane drugih. Poštujte principe čistog koda (en. clean code).*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Pitanje:**

*Ukoliko želimo više uslova ispitati u jednoj proveri, kako je to moguće uraditi?*

## ✓ Poglavlje 5

### Petlje

## UVOD U PETLJE

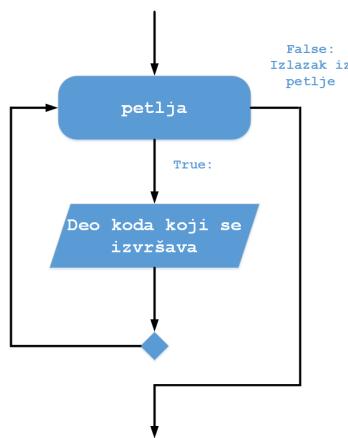
*Petlja predstavlja naredbu u programu koja ponavlja deo koda dok se ne ostvari neki uslov.*

Petlja (en. **loop**) predstavlja jedan od najosnovnijih koncepta programiranja.

Petlja predstavlja naredbu u programu koja ponavlja deo koda dok se ne ostvari neki uslov.

#### Opšta struktura petlje

Može se reći da petlja postavlja pitanje. Ukoliko odgovor na to pitanje zahteva neku radnju, ta radnja se izvršava. Isto pitanje se ponovo pita sve dok nije potrebno ništa više uraditi. Svaki put kada se postavlja pitanje čija je posledica izvršenje koda, dešava se jedna iteracija (en. **iteration**).



Slika 5.1 Opšta struktura petlje. [Izvor: Autor]

Najveća prednost korišćenja petlji jeste da programer ne mora više puta koristiti isti kod.

Svi programske jezike sadrže neki vid petlji.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PYTHON 3.X WHILE PETLJA

*While petlja u Python jeyiku identična je kao i kod drugih programske jezika.*

Python 3.x podržava dva osnovna tipa petlji: `while` i `for`.

Sintaksa while petlje je sledeća:

```
while (uslov):  
    Deo koda kada je uslov ispunjen
```

Sve dok je ispunjen uslov deo koda koji ide neposredno ispod while (tačnije, nakon znaka : ) izvršavaće se u iteracijama. Dobra (pravilna) praksa jeste unutar samog koda uneti naredbu koja će menjati neku promenljivu koja se ispituje u uslovu.

Ukoliko ne postoji naredba koja menja stanje uslova, javiće se tzv. beskonačna petlja (en. infinite loop) koja uglavnom izaziva probleme prilikom izvršenja programa.

### **Pitanje:**

*Kako najjednostavnije napraviti beskonačnu petlju?*

*Koji problemi se javljaju ukoliko postoji u programu beskonačna petlja?*

### **Primer: Ispisivanje celih brojeva**

Napisati program koji ispisuje sve cele brojeve od 0 do proizvoljnog broja n.

```
n = int(input("Uneti poslednji broj: "))  
i = 0  
while i <= n:  
    print(i)  
    i += 1
```

## BESKONAČNA PETLJA

*Kod while petlje treba voditi računa na pojavu beskonačne petlje.*

### **Primer: Beskonačna petlja**

U ovom primeru samo je neznatno promenjen uslov, koji dovodi do beskonačne petlje. Pri izvršenju ovog programa moraćemo ručno da prekinemo.

```
n = int(input("Uneti poslenji broj: "))  
i = 0  
while n:
```

```
print(i)
i += 1
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PYTHON 3.X FOR PETLJA

*For petlje u Python 3.x jeziku kreću od nultog indeksa.*

For petlje u Python 3.x jeziku nemaju sintaksu nalik na C jeziku u njemu slične jezike, već je sintaksa

```
for in n
```

For petlje se mogu koristiti za iteraciju ne samo za numeričke tipove podataka, već i za liste, stringove, tuple-ove i nizove.

Ukoliko je dat skup elemenata u listi, for petlja se može iskoristiti da izvrši iteraciju nad svakim elementom liste.

Da bi petlja prošla svaku iteraciju, koristi se funkcija *range()*.

*Funkcija range() vraća novu listu sa brojevima specifičnog opsega u odnosu na dužinu sekvence.*

Kroz iteraciju kroz sekvencu mogu se takođe iskoristiti i indeksi elemenata sekvene za iteraciju, nali najpre treba izračunati kolika je dužina liste i tek onda proći kroz iteracije unutar opsega te dužine.

*For petlje u Python 3.x jeziku kreću od nultog indeksa.*

### **Osnovni primer korišćenja for petlje**

#### **Primer: Štampanje brojača (od nule)**

```
for i in range(10):
    print(i)
```

#### **Primer: Štampanje brojača (od jedinice)**

```
for j in range(10):
    print(j+1)
```

#### **Primer: Štampanje brojača sa korakom 2**

```
for j in range(1,20,2):
    print (j)
```

#### **Primer: Štampanje karaktera i u tekstu "CS324 Skripting jezici"**

```
for k in "CS324 Skripting jezici":  
    if k == 'i':  
        print (k)
```

### Primer: Štampanje elemenata liste

```
predmeti = ["CS220", "CS225", "CS324", "IT331", "IT335"]  
for ii in predmeti:  
    print(ii)
```

### Primer: Štampanje elemenata heterogene liste

```
rnd_list = [1, 2, 20.54, 100 + 2.3j, 324, 'L03','Skripting Jezici']  
for jj in rnd_list:  
    print(jj)
```

## DODATNI PRIMERI I VIDEO SA PYTHON 3.X FOR PETLJAMA

### *Primer sa listama i petljama*

Do sada je prikazano više osnovnih primera u radu sa for petljom.

Može se primetiti da je for petlja jako korisna za dobijanje elemenata liste.

Međutim, ukoliko nekada želimo samo da saznamo koliko je lista dugačka, tj. koliko elemenata ima u sebi, možemo koristiti funkciju `len()`.

Ova funkcija vraća kao izlaz element liste.

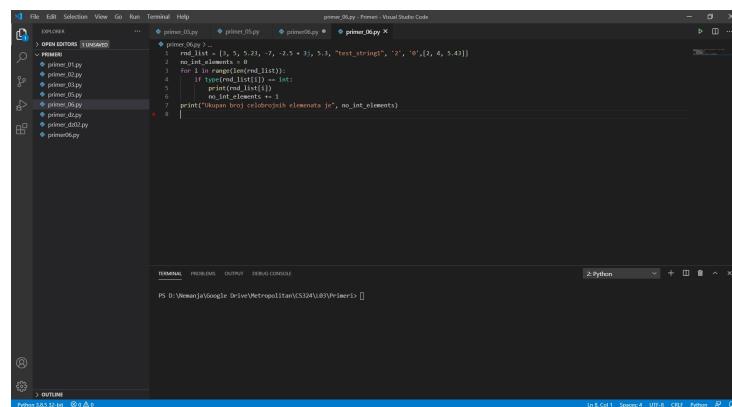
**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### Primer:

Napraviti listu koja sadrži 10 elemenata, a da sadrži celobrojne, kompleksne, razlomljene brojeve, stringove, i jednu listu kao elemente.

Napisati program koji će ispisati broj elemenata koji su celi brojevi, ali i te elemente štampati na izlazu.

```
rnd_list = [3, 5, 5.23, -7, -2.5 + 3j, 5.3, "test_string1", '2', '0',[2, 4, 5.43]]  
no_int_elements = 0  
for i in range(len(rnd_list)):  
    if type(rnd_list[i]) == int:  
        print(rnd_list[i])  
        no_int_elements += 1  
print("Ukupan broj celobrojnih elemenata je", no_int_elements)
```



```
prime05.py:1: def no_int_elements(rnd_list):  
prime05.py:2:     no_int_elements = 0  
prime05.py:3:     for i in rnd_list:  
prime05.py:4:         if type(rnd_list[i]) is int:  
prime05.py:5:             no_int_elements += 1  
prime05.py:6:     print('Input list contains', no_int_elements)
```

Slika 5.2 Primer programa koji broji celobrojne elemente liste. [Izvor: Autor]

## ▼ Poglavlje 6

### Pokazna vežba #3

#### UVOD U POKAZNU VEŽBU #3

*Izlaz, tačnije `print()` funkcija Python 3.x programskog jezika se može formatirati na više načina.*

U pokaznoj vežbi biće reči o formatiranju izlaza u različitim formatima.

Nakon toga, biće obrađeno par primera koji opisuju operacije nad numeričkim i ne-numeričkim podacima.

Procenjeno trajanje pokazne vežbe iznosi 45 minuta.

#### FORMATIRANJE IZLAZA

*Izlaz (štampa) se može različito formatirati u Python 3.x jeziku*

Formatiranje izlaza u Python 3.x jeziku može se ostvariti na klasični `%`način (poznat korisnicima C familije jezika), korišćenjem `.format()` metode, ali i i putem novog, jednostavnijeg `f-string` načina.

U nastavku vežbe biće dati primeri za svaki od tipova formatirana izlaza.

##### **Klasični način formatiranja izlaza**

U Python 3.x jeziku koristi se `%` placeholder za formatiranje izlaza, i može se iskoristiti za više tipova podataka. Ovaj način formatiranja se u literaturi naziva i *printf-stil formatiranja*, jer podseća na `printf` funkciju iz programskog jezika C.

U savremenom programiranju, ovakav način pisanja izlaza u Python 3.x jeziku je zastareo, i najviše se koriste sledeća dva načina - `.format()` metoda, i f-string formatiranje.

##### **Spisak %-formata**

- `%d` - označeni decimalni ceo broj
- `%o` - označeni oktalni broj
- `%x` - označeni heksadecimalni broj (mala slova)
- `%X` - označeni heksadecimalni broj (velika slova)
- `%e` - razlomljeni broj u eksponencijalnom obliku (mala slova)
- `%E` - razlomljeni broj u eksponencijalnom obliku (velika slova)
- `%f` - razlomljeni decimalni broj

- %g - razlomljeni decimalni broj (automatski bira da li koristi eksponencijalnu notaciju ili ne)
- %c - karakter
- %s - string
- %% - vraća karakter "%"

## FORMATIRANJE IZLAZA POMOĆU KLASIČNOG NAČINA - PRIMER

*Izlaz se može formatirati u Python 3.x jeziku na klasični način, tj. korišćenjem %*

### Primer #1: %-formatiranje stringova klasičnim načinom (15 minuta)

Napisati program koji ispisuje šifru predmeta i naziv predmeta koji se trenutno sluša, korišćenjem % formatiranja izlaza.

```
predmet_sifra = "CS324"
predmet_naziv = "Skripting jezici"
print("Slušate predmet %s - %s." % (predmet_sifra, predmet_naziv))
```

Primećuje se da u okviru izlaznog stringa koji ispisuje vrednost neke promenljive stoji karakter kojem prethodi znak procenta %. To predstavlja **placeholder** za promenljivu, koja se redom navodi nakon kraja stringa, ponovo sa % znakom pre navođenja.

```
predmet_sifra = "CS324"
predmet_naziv = "Skripting jezici"
print("Slušate predmet %s - %s." % (predmet_sifra, predmet_naziv))
```

Slika 6.1 Formatiranje izlaza % načinom. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

# FORMATIRANJE POMOĆU .FORMAT() METODE

*Metoda .format() olakšava štampanje izlaza u odnosu na % način, jer se ne mora brinuti o tome kakva je promenljiva*

## .format() način formatiranja izlaza

Umesto eksplisitnog navođenja tipa podataka kroz % placeholder, u izlaznom stringu se može navesti "mesto" za promenljivu, tako što će promenljiva biti "smeštena" u velike zagrade {}.

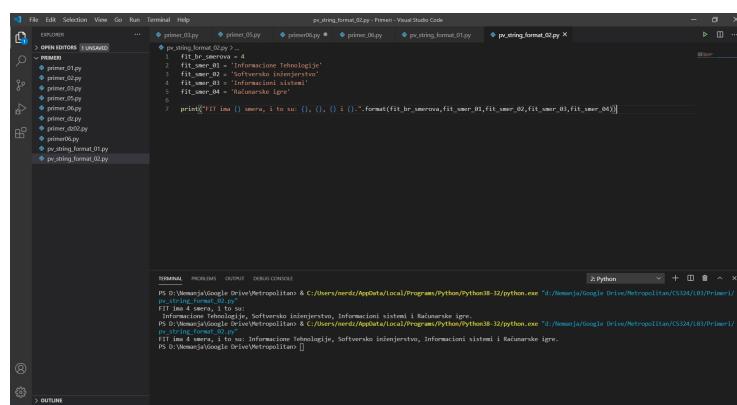
Nakon pisanja samog stringa, treba pozvati metodu .format(), koja u svojoj listi argumenata ima spisak promenljivih.

### Primer #2: formatiranje stringova .format() metodom (15 minuta)

Napisati program koji ispisuje broj i nazive smerova na Fakultetu Informacionih Tehnologija Univerziteta Metropolitan. Koristiti .format() metodu za formatiranje izlaznog stringa.

```
fit_br_smerova = 4
fit_smer_01 = 'Informacione Tehnologije'
fit_smer_02 = 'Softversko inženjerstvo'
fit_smer_03 = 'Informacioni sistemi'
fit_smer_04 = 'Računarske igre'

print("FIT ima {} smera, i to su: {}, {}, {} i {}.".format(fit_br_smerova,fit_smer_01,fit_smer_02,fit_smer_03,fit_smer_04))
```



Slika 6.2 Formatiranje izlaza .format() načinom. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

# FORMATIRANJE POMOĆU F-STRING METODE

*Najnoviji format koji podržava Python 3.x programski jezik jeste tzv. f-string formatiranje*

## f-string formatiranje izlaza

F-string formatiranje uvedeno je u Python 3.6 verziji avgusta 2015. godine.

Kod ovakvog formatiranja na početku stringa dodato je slovo `f`, a unutar velikih zagrada `{}` ubacuje se promenljiva čija se vrednost ispisuje.

### Primer #3: Računanje vrednosti broja Pi (15 minuta)

Napisati program koji računa približnu vrednost broja Pi korišćenjem Nilakanta aproksimacije:

$$\pi = 3 + \frac{4}{2 \times 3 \times 4} - \frac{4}{4 \times 5 \times 6} + \frac{4}{6 \times 7 \times 8} - \frac{4}{8 \times 9 \times 10} + \dots$$

Na izlazu koristiti f-string formatiranje.

```
var_pi_str = 'π'
pi_temp = 0

for i in range(1, 1 + int(input("Uneti željenu tačnost: "))):
    if i % 2 == 1:
        pi_temp += 4/((2*i) * ((2*i)+1) * ((2*i)+2))
    else:
        pi_temp -= 4/((2*i) * ((2*i)+1) * ((2*i)+2))
var_pi_approx = 3 + pi_temp

print(f"Vrednost broja {var_pi_str} izosi približno {var_pi_approx}.")
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 7

### Individualna vežba #3

#### UVOD U INDIVIDUALNU VEŽBU #3

*Individualna vežba #3 odnosi se na rad sa petljama i kontrolom toka.*

Individualna vežba #3 daje tri primera u kojima se radi sa petljama, kontrolom toka, i formatiranjem izlaza.

Procenjeno trajanje individualnih vežbi iznosi 90 minuta.

#### INDIVIDUALNA VEŽBA #3

*Napisati jednostavne programe u Python programskoj jeziku koristeći instalirana okruženja.*

##### **Zadatak #1** (25 minuta)

Napisati program koji ispisuje tablicu istinitosti za izraz

$x = (a \text{ OR } b) \text{ AND } \text{NOT}(c)$

u Bool / numeričkom obliku.

Koristiti:

1. Štampanje izlaza samo sa promenljivima
2. `.format()` metodu za štampanje izlaza uz pravilni alignment

##### **Zadatak #2** (30 minuta)

Napisati program koji broj četvorocifren broj indeksa pretvara u BCD format.

##### **Zadatak #3** (35 minuta)

Napisati program koji računa vrednost broja  $f_i$  po sledećem beskonačnom redu:

$$\varphi = \frac{13}{8} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1} (2n+1)!}{4^{2n+3} n! (n+2)!}$$

Umesto beskonačnosti uneti krajnji član reda.

Zatim, ispisati vrednost broja  $f_i$  ako je krajnji član 10, 100 i 1000 puta veći od unetog.

## ✓ Poglavlje 8

### Domaći zadatak #3

#### DOMAĆI ZADATAK

*Domaći zadatak #3 se okvirno radi 2h*

##### **Domaci zadatak #3**

###### **Zadatak #1**

Napisati program u kojem ubacujete broj trenutno položenih ispita. Ulazne podatke sačuvati u imenik koji sadrži šifru predmeta kao ključ predmeta, a vrednost jeste ocena na položenom predmetu.

Zatim, izvršiti upit za predmet po šifru predmeta.

Izbaciti kao izlaz da li je predmet položen ili ne, i ako jeste, sa kojom ocenom.

###### **Zadatak #2**

Napisati program koji će generisati  $n$  nasumičnih celih brojeva, gde je  $n$  broj indeksa studenta, i smestiti u listu, i sortirati listu u padajućem redosledu. Koristiti *import random* za dobijanje nasumičnih brojeva.

###### **Zadatak #3**

Napisati program koji će generisati  $m$  nasumičnih razlomljenih brojeva uniformne raspodele tako da je opseg od prve dve cifre indeksa do druge dve cifre, i  $m = \text{broj\_indeksa} // 3$ . Smestiti brojeve u imenik, tako je ključ redni broj, a vrednost sam razlomljeni broj.

Zatim, izvući *broj\_indeksa % 3* elementa imenika. Koristiti *import random*.

Primer: Ako je broj indeksa 1234, koristiti:

```
random.uniform(12, 34)
```

###### **Predaja domaćeg zadatka:**

###### **Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije nedelju dana nakon predavanja za 100% poena. Nakon toga poeni se umanjuju za 50%.

###### **Online studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganja ispita. **Domaći zadaci se  
brane!**

Svi studenti domaći zadatak poslati dr Nemanji Zdravkoviću:  
[nemanja.zdravkovic@metropolitan.ac.rs](mailto:nemanja.zdravkovic@metropolitan.ac.rs)

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Zaključak lekcije #3*

#### **Rezime:**

U ovoj lekciji bilo je reči o standardnim tipovima podataka koje Python 3.x programski jezik koristi.

Bilo je reči o logičkim, numeričkim i ne-numeričkim tipovima podataka.

Zatim, govorilo se o kontroli toka i o if-else uslovnim naredbama.

Konačno, bilo je reči o petljama while i for.

Sve ove oblasti propraćene su adekvatnim primerima.

#### **Literatura:**

1. David Beazley, Brian Jones, Python Cookbook: Recipes for Mastering Python 3, 3rd edition, O'Reilly Press, 2013.
2. Mark Lutz, Learning Python, 5th Edition, O'Reilly Press, 2013.
3. Andrew Bird, Lau Cher Han, et. al, The Python Workshop, Packt Publishing, 2019.
4. Al Sweigart, Automate the boring stuff with Python, 2nd Edition, No Starch Press, 2020.





## CS324 - SKRIPTING JEZICI

### Strukture podataka i funkcije

Lekcija 04

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 04

### *STRUKTURE PODATAKA I FUNKCIJE*

- ✓ Strukture podataka i funkcije
- ✓ Poglavlje 1: Strukture podataka
- ✓ Poglavlje 2: Funkcije
- ✓ Poglavlje 3: Lambda funkcije
- ✓ Poglavlje 4: Parametri i povratne vrednosti
- ✓ Poglavlje 5: Ugrađene funkcije
- ✓ Poglavlje 6: Pokazna vežba #4
- ✓ Poglavlje 7: Individualna vežba #4
- ✓ Poglavlje 8: Domaći zadatak #4
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ✓ Uvod

# UVOD U ČETVRTU LEKCIJU

### *Uvod u lekciju #4*

U četvrtoj lekciji nastavlja se sa proceduralnim programiranjem u Python 3.x jeziku.

Dok je prethodna lekcija opisala numeričke promenljive, stringove, imenike i druge tipove podataka, kao i kontrolu toka i petlje, u ovoj lekciji biće reči o strukturama podataka, funkcijama, njihovim parametrima i povratnim vrednostima, kao i nekim ugrađenim funkcijama u Python 3.x programskom jeziku.

U opštem slučaju, strukture podataka se mogu podeliti u postojeće (en. **built-in**) strukture, i u strukture koje sami korisnici definišu (en. **user-defined**).

Postojeće strukture podataka u Python 3.x jeziku jesu zapravo tipovi podataka koji sadrže više od jedne vrednosti, a koje su obrađivani u prethodnoj lekciji. To su liste, imenici, i Tuple-ovi. Sa druge strane, najčešće korišćene strukture podataka koje korisnik pravi jesu stekovi (en. **stack**), redovi (en. **queue**), stabla (en. **tree**), lančane liste (en. **linked lists**) i grafovi (en. **graphs**).

U ovoj lekciji detaljno će biti objašnjeni načini kreiranja korisničkih struktura podataka, a u sledećoj prelazi se na objektno-orientisano programiranje u Pythonu 3x. jeziku.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 1

# Strukture podataka

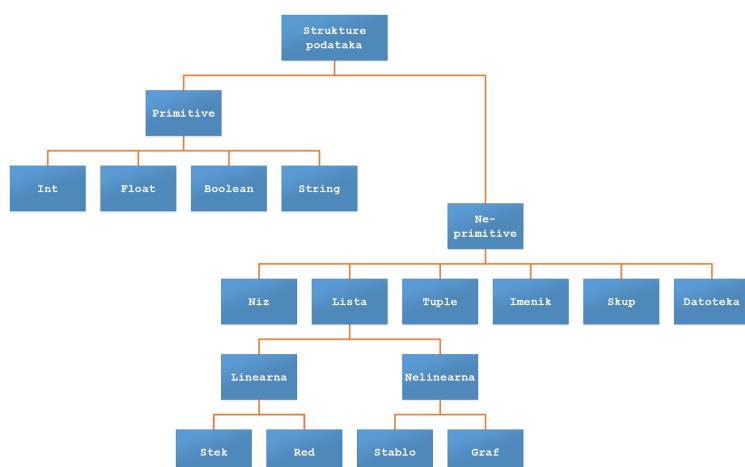
## UVOD U STRUKTURE PODATAKA

*Strukture podatka mogu u sebi sadržati iste ili različite tipove podataka.*

Strukture podataka (en. **data structures**) jesu upravo to: strukture koje mogu sadržati podataka u sebi. Koriste se za smeštanje skupa povezanih podataka.

Strukture podatka se u opštem slučaju, pa tako i u Python jeziku dele na ugrađene skupove podataka, ali i na skupove podataka koje korisnik definiše. U nastavku biće reči o obe vrste struktura podataka.

Neke od struktura podataka su već obrađene, samo nisu formalno tako nazvani - a to su tipovi podataka, prvenstveno oni koji sadrže samo jednu vrednost. U opštem slučaju, na sledećoj slici data je opšta podela struktura podataka.



Slika 1.1.1 Podela struktura podataka u Python 3.x jeziku. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ❖ 1.1 Primitive u Python jeziku

### PRIMIVITE U PYTHONU

*Primitive jesu tipovi podataka koji svoja imena mapiraju na jedan objekat u memoriji.*

#### Definicija

Primitivna struktura podataka (en. primitive data structure) jesu takve strukture podataka koje programski jezik ima ugrađenu podršku, i koje sadrže uglavnom jednu vrednost, u zavisnosti od programske jezike.

U Python 3.x jeziku, primitive jesu tipovi podataka koji svoja imena mapiraju na jedan objekat u memoriji. Primitive jesu takvi tipovi podataka koji programski jezik ne dozvoljava naknadno menjanje način funkcionisanja od strane programera.

#### **Pitanje:**

*Zbog čega programski jezici ne dozvoljavaju modifikaciju primitiva?*

*Da li postoji programski jezik koji dozvoljava modifikaciju primitiva?*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

#### Primitive u Python 3.x programskom jeziku su

- Logički tip podataka (en. `boolean`, `<bool>`)
- Celobrojni tip (en. `integer`, `<int>`)
- Razlomljeni ti podataka (en. `float`, `<float>`)
- Znakovni niz ili string (en. `string`, `<str>`)

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### IMPLICITNA KONVERZIJA PRIMITIVA

*U Python-u je moguće izvršiti konverziju tipa podataka nad primitivnim tipovima podataka.*

U Pythonu je moguće izvršiti konverziju tipa podataka nad primitivnim tipovima podataka.

Postoje dva načina konverzije tipa podataka: implicitno i eksplisitno.

#### Implicitna konverzija

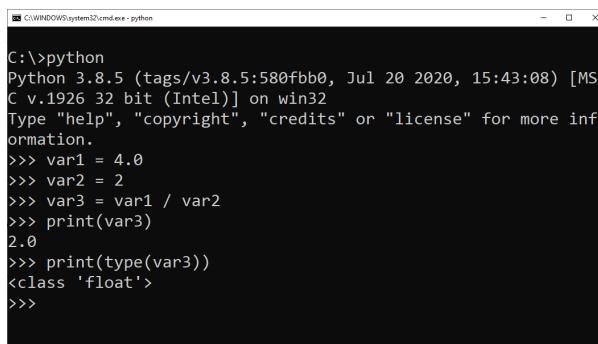
Implicitna konverzija zapravo automatski vrši konverziju prilikom interpretacije koda. Nije potrebno da se "ručno" unese u koji se tip podataka promenljiva želi konvertovati.

**Primer:**

Implicitna konverzija **int** u **float** deljenjem.

```
var1 = 4.0 #Float
var2 = 2    #int
var3 = var1 / var2
print(var3)
print(type(var3))
```

Rezultat deljenja razlomljenog tipa i podataka i celobrojnog tipa biće takođe razlomljeni (float) tip.



```
C:\>python
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MS
C v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more inf
ormation.
>>> var1 = 4.0
>>> var2 = 2
>>> var3 = var1 / var2
>>> print(var3)
2.0
>>> print(type(var3))
<class 'float'>
>>>
```

Slika 1.2.1 Implicitna konverzija deljenjem [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## EKSPLICITNA KONVERZIJA PODATAKA

*U Python-u je moguće izvršiti eksplisitnu konverziju tipa podataka, ali ne uvek.*

### Eksplisitna konverzija

Eksplisitna konverzija tipa podataka se vrši "ručno" tj. programer mora interpreteru da napiše naredbu konverzije tipa.

Interpreter ne može sam da shvati šta je programer htio da postigne, pogotovo kada promenljive uključuju i stringove. Zbog toga programer mora biti obazriv oko konverzije.

**Primer:**

Eksplisitna konkatenacija stringa sa celobrojnim podatkom.

Sledeći kod javiće grešku:

```
var1 = "Odgovor na pitanja o suštini života je "
var2 = 42
var3 = var1 + var2
print(var3)
```

Greška koja se javlja jeste greška tipa podataka, i javlja se jer ne možemo spojiti string i int u novi string.

Zbog toga je potrebno najpre eksplisitno izvršiti konverziju tipa:

```
var1 = "Odgovor na pitanja o suštini života je "
var2 = 42
var3 = var1 + str(var2)
print(var3)
```

Funkcijom **str()** čiji je argument var2, vrednost promenljive var2 (koja je bila int) konvertujemo u string.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ 1.2 Ne-primitive u Python jeziku

### NE-PRIMITIVE U PYTHON-U

*Nizovi u Python 3.x programskom jeziku predstavlja efikasan način da se smeštaju podaci istog tipa.*

#### Definicija

Ne-Primitivna struktura podataka (en. **non-primitive data structure**) jesu takve strukture podataka koje sadrže više vrednosti. Zapravo, u ovakvim strukturama zadržan je skup podataka.

U opštem slučaju, ovakve strukture podataka dele se na:

- Nizove (en. **arrays**)
- Liste (en. **lists**)
- Datoteke (en. **files**)

#### Nizovi

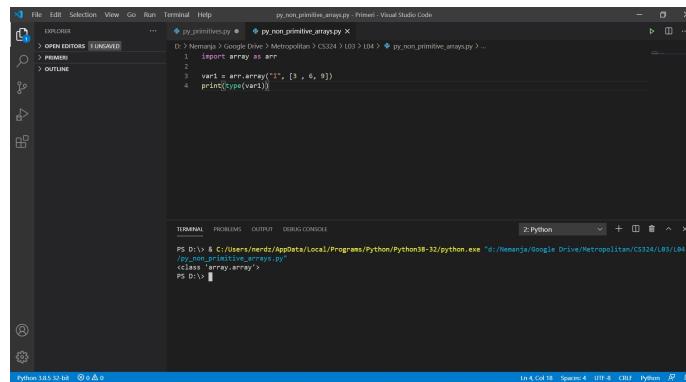
Nizovi u Python 3.x programskom jeziku predstavlja efikasan način da se smeštaju podaci istog tipa.

Svaki element niza je istog tipa podataka.

U Python 3.x programskom jeziku se nizovi kao tip podataka ne koristi često, već se umesto toga koristi ne-primitiva lista. Ukoliko želimo koristiti niz, mora se uvesti iz modula za rad sa nizovima.

```
import array as arr

var1 = arr.array("I", [3, 6, 9])
print(type(var1))
```



Slika 1.3.1 Uvoz modula array i imenovanje novog niza celobrojnih podataka. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## LISTE I MANIPULACIJA LISTI

*Dat je pregled najčešće korišćenih metoda za rad sa listama i elementima lista*

Liste u Python 3.x jeziku se koriste za smeštanje skupa heterogenih podataka (podatka koje ne moraju biti istog tipa). Liste su ugrađene u Python programski jezik.

Liste se mogu modifikovati, tj. može im se menjati sadržaj a da im se ne menja identitet tj. ime.

Liste se mogu najbolje prepoznati po srednjim zagradama, a elementi liste su razdvojeni zapetom.

Liste, naravno, mogu sadržati homogene podatke, i na taj način predstavljaju nizove.

Ono što nije do sada rečeno jeste da Python sadrži mnoštvo metoda za rad sa listama. U nastavku biće reči nekim najčešće korišćenim metodama za rad sa listama.

### Primer:

Date su sve liste, jedna sa brojevima, a druga karakterima

```
list_num = [1, 2, 3, 6, 9, 43, 90, 27, 333]
list_char = ['s', 'k', 'r', 'i', 'p', 't', 'i', 'n', 'g']
```

## Dodavanje elementa

Element se može dodati (po **default**-u) na kraj liste metodom **`.append()`**

```
list_num.append(324)
```

Element se može dodati na željenu poziciju metodom **`.insert()`**, gde se navodi pozicija, i sama vrednost.

```
list_num.insert(0, 749)
```

## Oduzimanje elementa

Element se može oduzeti metodom **`.remove()`**, gde unosimo vrednost element kojeg želimo da uklonimo. Ukoliko lista ima više istih elemenata, ukloniće prvi na koji nađe.

```
list_char.remove('i')
```

Element sa određenim indeksom se može ukloniti korišćenjem metode **`.pop()`**, gde se navodi indeks (pozitivni ili negativni) elementa kojeg uklanjamo.

```
list_char.pop(-2)
```

# METODE ZA RAD SA LISTAMA

*Dat je pregled najčešće korišćenih metod za rad sa listama*

## Sortiranje listi

Python podržava sortiranje listi metodama **`.sort()`** (za sortiranje po rastućem redosledu) i **`.reverse()`** (za sortiranje po opadajućem redosledu)

```
list_num.sort()  
list_num.reverse()
```

## Spajanje elemenata liste u string

Pojedinačni elementi liste mogu se spojiti u jedan string korišćenjem **`".join()`** metode pri pozivu stringa.

```
predmet_naziv = ''.join(list_char)  
print(predmet_naziv)
```

Dodavanje više elemenata u listu

Metodom **`.append()`** može se dodati element po element u listu, dok metodom **`.extend()`** mogu se dodati više elemenata odjednom.

```
list_num.extend([4, 5])
```

**Pitanje:**

Kako dodati više elemenata listi korišćenjem **.append()**?

Šta bi se desilo ako za dati primer ukucamo **list\_num.append([4, 5])**?

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## STEKOVI I REDOVI

### *Stekovi i redovi se implementiraju kroz liste*

Stek (en. **stack**) sadrži objekte koji se ubacuju i izbacuju po sistemu *Last-in-First-Out* ili LIFO.

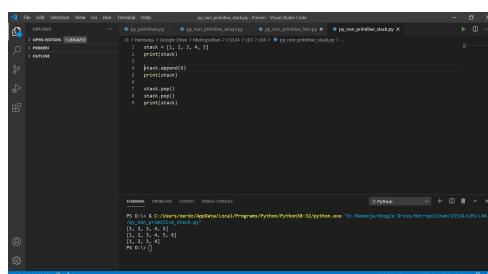
U računarstvu stekovi se koriste za računanje naredbi i sintaksno parsiranje, za algoritme i rutine raspoređivanja.

Stekovi se mogu implementirati u Python 3.x jeziku preko listi. Kada se element doda u stek, to se naziva operacija **push**, a kada se ukloni element, to je **pop** operacija.

#### **Primer: Dodavanje i uklanjanje elemenata u steku**

```
stack = [1,2,3,4,5]
stack.append(6)
print(stack)
```

```
stack.pop()
stack.pop()
print(stack)
```



Slika 1.3.2 Dodavanje i oduzimanje elemenata steka. [Izvor: Autor]

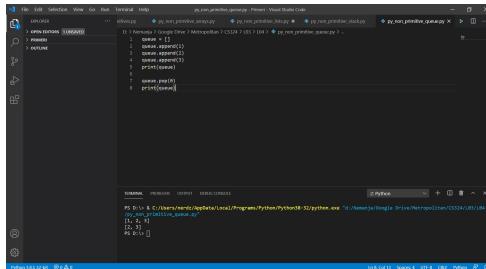
Red (en. **queue**) sadrži objekte koji se ubacuju i izbacuju po sistemu *First-In-First-Out* ili FIFO.

Red se ne može efikasno implementirati u Python 3.x jeziku kao stek, jer push/pop naredbe stavljuju elemente na kraj liste, a pri svakom ulazu/izlazu, potrebno je promeniti indekse elemenata.

Međutim, moguće je implementirati kroz liste tako što bi uz metodu **.pop()** uvek bio argument 0.

### Primer: Dodavanje i uklanjanje elemenata u redu

```
queue = []
queue.append(1)
queue.append(2)
queue.append(3)
print(queue)
queue.pop(0)
print(queue)
```



Slika 1.3.3 Dodavanje i oduzimanje elementa reda. [Izvor: Autor]

## PRIMER ZA STEKOVE I REDOVE

*Implementacije steka je nešto jednostavnije nego redova, ali i redovi se lako implementiraju pomoću listi u Python jeziku.*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## GRAFOVI

*Graf se sastoji od čvorova i veza između čvorova. Može biti neusmeren ili usmeren.*

Graf (en. **graph**) u matematici i računarskoj tehnici se sastoji od čvorova (en. **nodes**) koji mogu biti međusobno povezani.

Ukoliko postoji smer kako je koji čvor povezan sa drugim, onda je to usmeren graf.

Mnogi problemi u matematici i računarskoj tehnici, ali i šire, se mogu predstaviti kroz grafove. Nauka matematike koja se bavi proučavanjem grafova jeste teorija grafova (en. **graph theory**).

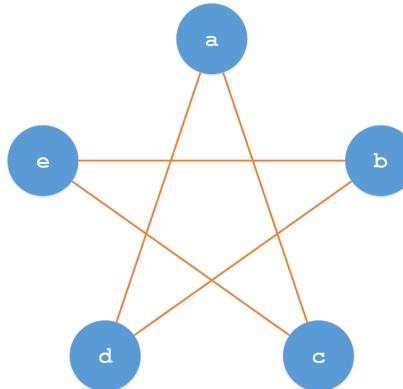
Čvor opisuje iz god se grafa dolazi u taj čvor, kao i ka kom (ili kojim) čvorovima se može doći iz tog čvora.

### Grafovi u Pythonu

Graf ne postoji kao ugrađen tip, već se može realizovati kao posebna struktura, kao klasa, ili preko lista ili imenika.

### Primer: Graf preko imenika

Realizovati graf sa slike preko imenika, tako da ključ bude ime čvora (string), a vrednost bude lista stringova sa povezanim čvorovima.



Slika 1.3.4 Graf sa 5 čvorova. [Izvor: Autor]

```
graph = { "a" : ["c", "d"],  
         "b" : ["d", "e"],  
         "c" : ["a", "e"],  
         "d" : ["a", "b"],  
         "e" : ["b", "c"]  
     }
```

## STABLA

*Stablo je najbolje realizovati kroz korisničke strukture, ili kroz klasu*

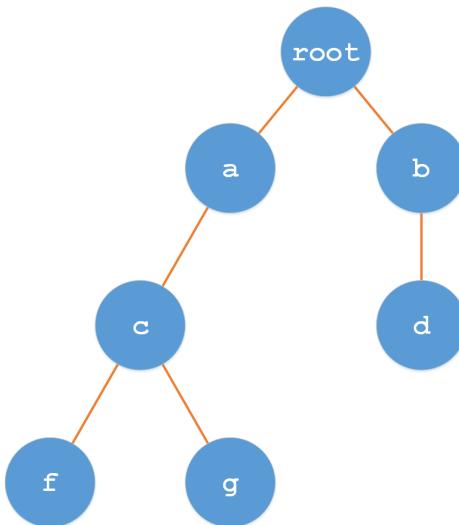
Stablo (en. **tree**) u matematici i računarskoj tehnici predstavlja strukturu podataka koja se takođe sastoji od čvorova, ali su hijerarhijski povezani.

Čvor korena (en. **root node**) je takav čvor da od njega kreću veze ka ostalim čvorovima.

Svi ostali čvorovi imaju roditeljski čvor (en. **parent node**) a mogu imati i decu čvorove (en. **child node**).

### Binarno stablo

Binarno stablo je poseban tip stabla, kod kojeg svaki čvor ima veze ka dva čvora u nižem stepenu hijerarhije.



Slika 1.3.5 Stablo. [Izvor: Autor]

Slika 1.3.6 Binarno stablo. [Izvor: Autor]

### Stabla u Python 3.x jeziku

Stablo se takođe može opisati kroz liste ili imenike, ali prava funkcionalnost stabla se može dobiti definisanjem nove strukture podataka, ili, još bolje, kroz klasu stabla.

O klasama biće reči u lekciji #5, kada se obrađuju paradigme objektno-orientisanog programiranja - OOP.

## PRIMER ZA GRAFOVE I STABLA

*Sledi video o grafovima i stablima*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 2

# Funkcije

## FUNKCIJE U PYTHON 3.X JEZIKU

*Funkcije se mogu podeliti na ugrađene i korisničke, u zavisnosti da li su "gotove za korišćenje" ili ih korisnik sam treba definisati.*

Funkcija (en. `function`) u opštem slučaju predstavlja deo koda koji se može ponovo iskoristiti.

Enkapsuliranjem dela koda, odnosno niz naredbi u funkciju, taj niz naredbi se može ponovo ponoviti samo navođenjem tj. pozivanjem funkcije, a ne ponovnim pisanjem koda.

### Funkcije u Python 3.x jeziku

Do sada je već bilo reči o funkcijama, a da prethodno nije eksplisitno rečeno da se o njima radi. To je prvenstveno zbog toga što su se koristile ugrađene (en. `built-in`) funkcije.

Svaka komanda koja sadrži zagrade na kraju, na primer `print()`, predstavlja funkciju.

To znači da je funkcija definisana svojim imenom, zagradačima, a promenljive koje se ubacuju u zagrade jesu argumenti te funkcije.

Gotovo svaki programski jezik dozvoljava kreiranje vlastitih, tj. korisničkih (en. `user-defined`) funkcija koje se mogu naknadno pozvati u programu.

### Definisanje funkcija u Python 3.x jeziku

Korisničke funkcije se mogu definisati korišćenjem ključne reči `def`, nakon čega se navodi ime funkcije, parametri, ukoliko postoje, i dvotačka da bi označila da deo koda koji sledi pripada funkciji. Funkcija takođe može imati jednu ili više povratnih vrednosti.

Sintaksa je sledeća:

```
def fun_name(par1, par2, ...):  
    function_code  
    return ...
```

O parametrima funkcije i povratnim vrednostima biće detaljno reči u nastavku lekcije.

## ŽIVOTNI CIKLUS PROMENLJIVIH UNUTAR FUNKCIJA

*Funkcija ne pamti vrednost promenljive iz ranijeg poziva.*

Opseg (en. `scope`) promenljive jeste deo koda gde se ta promenljiva može prepoznati,

Parametri i promenljive definisani unutar funkcije nisu vidljivi izvan funkcije. Zbog toga oni imaju lokalni opseg.

Životni ciklus promenljive (en. [the lifetime of a variable](#)) jeste vreme za koje ta promenljiva postoji u memoriji. Životni ciklus promenljive unutar funkcije je onoliko koliko se potrebno da se ta funkcija izvrši.

*Kada se nakon poziva funkcije izvrši povratak u glavni program, lokalne promenljive se brišu.*

Funkcija ne pamti vrednost promenljive iz ranijeg poziva.

### Primer

Napisati funkciju koja ima lokalnu promenljivu var1 i koja ispisuje vrednost promenljive.

Nakon toga, u glavnom programu takođe napraviti promenljivu var1.

Pozvati funkciju, pa zatim ispitati vrednost promenljive var1.

```
def local_var():
    var1 = "CS324"
    print("Vrednost promenljive var1 unutar funkcije:", var1)

var1 = "Skripting jezici"
local_var()
print("Vrednost promenljive var1 izvan funkcije:", var1)
```

## PRIMER KORISNIČKE FUNKCIJE BEZ PARAMETARA U PYTHON 3.X JEZIKU

*Primer definisanje i pozivanja jednostavne funkcije u Python 3.x jeziku*

### Primer: Ispisivanje šifre i imena predmeta

Definisati funkciju sifra\_predmeta() koja u sebi ima tri promenljive:

smer (string) - prvi deo šifre predmeta (CS/IT/AD i sl.)

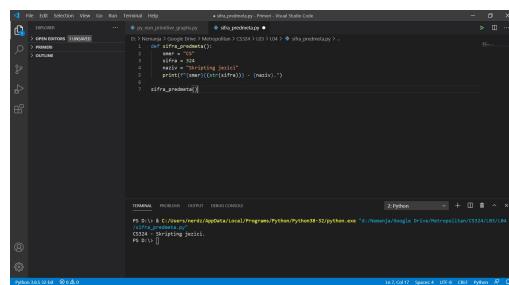
kod (int) - drugi deo šifre predmeta, trocifreni kod

naziv (string) - Naziv predmeta na srpskom jeziku.

Funkcija **sifra\_predmeta()** ispisuje pun naziv predmeta. Na primer:

CS324 - Skripting jezici.

```
def sifra_predmeta():
    smer = "CS"
    sifra = 324
    naziv = "Skripting jezici"
    print(f"{smer}{(str(sifra))} - {naziv}.")
```



```
sifra_predmeta.py
1 user = input('Unesite predmet: ')
2 predmet = sifra_predmeta(user)
3 print(predmet)
4
5 def sifra_predmeta():
6     pass

sifra_predmeta.py
Unesite predmet: SIFRA
SIFRA
```

Slika 2.1 Funkcija sifra\_predmeta(). [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 3

# Lambda funkcije

## DEFINICIJA LAMBDA FUNKCIJE

*Lambda funkcija može imati bilo koji broj parametra, ali se mora izvršiti u jednom redu koda.*

Korisničke definisane funkcije u opštem slučaju mogu sadržati proizvoljni broj linije koda.

Ukoliko se želi definisati funkcija koja je jednostavna i koja se može izvršiti u jednoj liniji, preporuka je koristiti tzv. lambda funkciju. (en. **lambda function**).

Lambda funkcije poznate su kao i anonimne funkcije. (en. **anonymous function**) jer nemaju ime, ali u Python 3.x jeziku im se može i dodeliti ime.

### Definisanje lambda funkcije

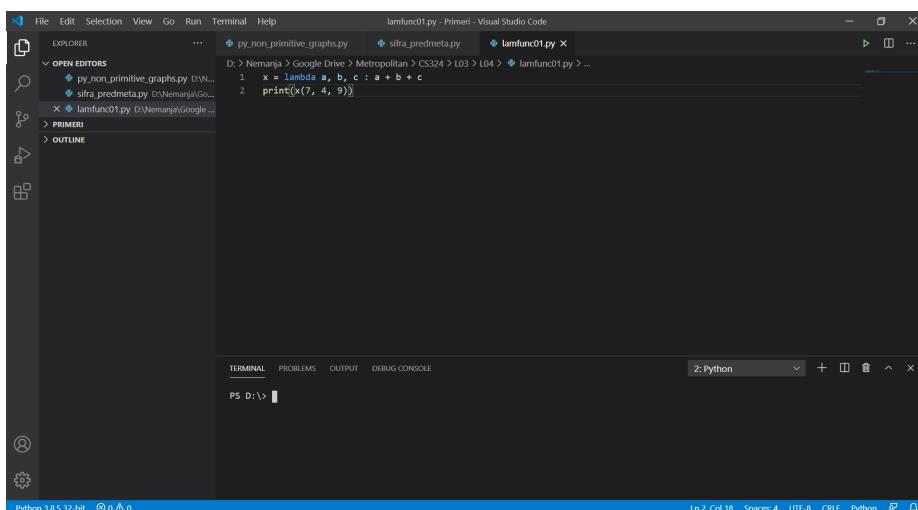
Standardne korisničke funkcije se definišu pomoću ključne reči **def**, a lambda funkcija se definiše rečju **lambda**.

*Lambda funkcija može imati bilo koji broj parametra, ali se mora izvršiti u jednom redu koda.*

### Primer: Sabiranje tri broja pomoću lambda funkcije

Napisati lambda funkciju za sabiranje tri broja koja se unose kao parametri.

```
x = lambda a, b, c : a + b + c
print(x(7, 4, 9))
```



Slika 3.1 Primer jednostavne lambda funkcije. [Izvor: Autor]

## GDE KORISTITI LAMBDA FUNKCIJE?

*Lambda funkcije jesu brze funkcije i najbolje ih je koristiti u telu drugih korisnički-definisanih funkcija.*

Lambda funkcije jesu brze funkcije i najbolje ih je koristiti u telu drugih korisnički-definisanih funkcija.

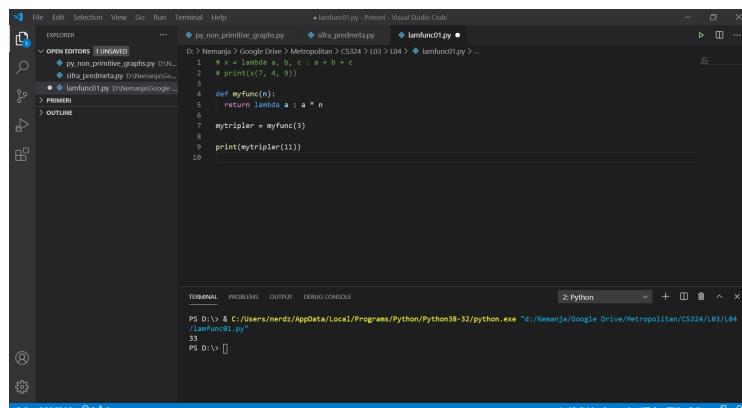
### Primer:

Napisati funkciju koja će vraćati trostruku vrednost unetog broja.

```
def myfunc(n):
    return lambda a : a * n

mytrippler = myfunc(3)

print(mytrippler(11))
```



Slika 3.2 Lambda funkcija unutar funkcije. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 4

# Parametri i povratne vrednosti

## PARAMETRI I PROSLEĐIVANJE ARGUMENATA

*Informacija koje se prosleđuje funkciji predstavlja argument funkcije.*

Do sada je bilo reči o pozivanju funkcije samim pozivom funkcije. Međutim, funkcija može imati ulazni **parametar**, odnosno **argument** koji se prosleđuju funkciji.

*Informacija koje se prosleđuje funkciji predstavlja argument funkcije.*

Pri definisanju funkcije, u opštem slučaju, ne postoji granica o broju parametara koji se prosleđuju funkciji. Svi ulazni parametri se razdvajaju zapetom.

```
def puno_ime(ime, prezime):  
    print(ime + " " + prezime)  
  
puno_ime("Petar", "Petrovic")
```

Postavlja se pitanje da li je pravilno reći argument ili parametar.

*Parametar predstavlja promenljivu koja se upisuje unutar zagrada prilikom definicije funkcije.*

*Argument predstavlja vrednost koja se šalje funkciji kada se ta funkcija poziva.*

Funkcija može imati i prozvoljan broj argumenta, tako što se navodi ključni znak **\*** pre imena parametra

```
def brojevi(*broj):  
    ...
```

Argumenti funkcije se mogu prosleđivati i u preko sledeće sintakse:

```
key = value
```

Na ovaj način redosled argumenata nije bitan.

```
def puno_ime(ime, prezime):  
    print(ime + " " + prezime)  
  
puno_ime(prezime = "Vukasinovic", ime = "Milic")
```

Funkcija se može definisati i sa **default**-nim argumentom, prilikom same definicije funkcije.

```
def ispit(sifra = "CS324"):  
    print("Slušate predmet " + sifra)
```

<autorski video, ime >

## VRAĆANJE JEDNOG ARGUMENTA

*Funkcija takođe može vratiti informaciju nazad u glavni program.*

Pored primanja informacija, funkcija može i vratiti informaciju nazad u glavni program.

Naredba `return` se koristi da se poziv funkcije završi i da se rezultat (vrednost naredbe koja sledi posle ključne reči `return`) vrati u program, odnosno onom ko je funkciju i pozvao

Kompletan sintaksu definicije funkcije je sledeća:

```
def func():  
    func_body  
    .  
    .  
    .  
    return expression
```

### Primer:

Napisati funkciju koja vraća `True` ukoliko je uneti broj paran, a `False` ukoliko je uneti broj neparan.

```
def provera_parnosti(broj):  
    if broj % 2 == 0:  
        return True  
    else:  
        return False
```

<autorski video, ime>>

## VRAĆANJE VIŠE ARGUMENATA

*U Python 3.x jeziku, moguće je da funkcija vrati više argumenata*

U Python 3.x programskom jeziku moguće je da `return` ključna reč vrati više od jednog argumenta.

Prilikom definisanje funkcije, bitno je kako odvojiti više argumenata koji se vraćaju, jer se mogu vratiti u više ne-primitivnih tipova podataka.

### Vraćanje više argumenata kao tuple

Najjednostavniji načina vraćanja više podataka jeste da se argumenti odvoje zapetom. Na taj način svi argumenti se smeštaju u **tuple**.

```
def funkcija01():
    smer = "CS"
    sifra = 324
    naziv = "Skripting Jezici"
    return smer, sifra, naziv

lst = funkcija01()
print(type(lst))
print(lst)

print(lst[0]+str(lst[1]), " - ", lst[2])
```

### Vraćanje više argumenata kao lista

Najlakši način vraćanja više podataka jeste da se ti podaci smeste u listu, pogotovo zbog osobine liste da može sadržati elemente različitog tipa, a i da se vrednosti mogu modifikovati.

```
def funkcija02():
    smer = "CS"
    sifra = 324
    naziv = "Skripting Jezici"
    return [smer, sifra, naziv]

lst2 = funkcija02()
print(type(lst2))
print(lst2)

print(lst2[0]+str(lst2[1]), " - ", lst2[2])
```

### Vraćanje više argumenata kao imenik

Ukoliko imamo samo dva izlazna argumenta, možemo ih smestiti u imenik.

### Vraćanje više argumenata kao objekat

Izlazni argumenti se mogu vratiti i kao objekat. O ovome više reči više u lekciji iz objektno-orientisanog programiranja u Python 3.x jeziku.

## ✓ Poglavlje 5

# Ugrađene funkcije

## UGRAĐENE FUNKCIJE U PYTHON 3.X JEZIKU

*Ugrađene funkcije jesu uvek dostupne jer su deo interpretera.*

Ugrađene funkcije (en. **built-in functions**) jesu funkcije koje su deo Python interpretera i one su uvek dostupne.

Mnoge funkcije su već bile korišćene, i neke su i intuitivne, ali neke nisu tako očigledne. Sledi tablela ugrađenih funkcija u Python 3.x jeziku.

Built-in Functions				
<code>abs()</code>	<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>
<code>all()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>any()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>ascii()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>bin()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bool()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>breakpoint()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	

Slika 5.1 Tabela ugrađenih funkcija. Izvor: [<https://docs.python.org/3/library/functions.html>]

Ugrađene funkcije se mogu podeliti na funkcije ulaza i izlaza (en. **input and output functions**), matematičke funkcije (en. **math functions**), funkcije za tipove podataka (en. **type functions**), i ostale funkcije koje nisu specifično kategorisane.

Spisak ugrađenih funkcija se može naći u dokumentaciji trenutne verzije Python programskega jezika:

<https://docs.python.org/3/library/functions.html>

Pored samog spiska, dati su i primer i korišćenja svake od funkcija.

## STANDARDNI ULAZ/IZLAZ

Za ulazno/izlazne operacije najpre se koriste funkcije `input()` i `print()`.

Za unos i štampu podataka (sa i na konzolu, respektivno) u Python 3.x programskom jeziku koriste se standardne funkcije ulaza i izlaza.

### Standardni ulaz - `input()`

Ukoliko programer želi da unese podatak putem tastature kroz konzolu, treba pri pisanju programa koristiti funkciju `input()`.

Sintaksa `input()` funkcije je sledeća:

```
input([prompt])
```

gde je `prompt` string koji se ubacuje.

#### Pitanje:

U Python 2.x i 3. drugačije je definisana standardna funkcija `input()`. U čemu je razlika?

U svakom primeru do sada gde je bilo potrebno uneti neku vrednost sa tastature, korišćena je funkcija `input()`.

### Standardni izlaz - `print()`

Ukoliko programer želi da ispiše podatak na ekran kroz konzolu, treba pri pisanju programa koristiti funkciju `print()`.

Sintaksa `print()` funkcije je sledeća:

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

gde su `objects` argumenti koji se štampaju, `sep` je separator između vrednosti argumenata. a `end` se štampa na kraju svih vrednosti, i po default-u to je novi red.

#### Pitanje:

Da li je moguće promeniti default-ne vrednosti `sep` i `end` argumenta?

O formatiranju standardnog izlaza bilo je reči u prethodnoj lekciji.

## MATEMATIČKE FUNKCIJE

Python interpreter ima ugrađene matematičke funkcije za neka osnovna izračunavanja bez potreba za pozivanjem dodatnog modula.

Matematičke ugrađene funkcije pomažu pri osnovnim matematičkim izračunavanjima.

### Funkcija absolutne vrednosti: **abs()**

Ova funkcija vraća absolutnu vrednost broja, koji može biti ceo broj, razlomljen broj, ili kompleksni broj.

### Konverzija u binarni broj - **bin()**

Ova funkcija konvertuje ceo broj u binarni string, sa prefiksom "0b" . Ovaj prefiks se može izostaviti dodatnim formatiranjem.

### Konverzija u heksadecimalni broj - **hex()**

Ova funkcija konvertuje ceo broj u heksadecimalni string sa prefiksom "0x". Karakter "x" može se formatirati da bude isписан velikim ili malim slovom, ili skroz da bude izostavljen.

### Najmanji broj - **min()**

Ova funkcija vraća najmanji element u strukturi ili manji od dva argumenta.

### Konverzija u oktalni broj - **oct()**

Ova funkcija konvertuje ceo broj u oktalni string sa prefiksom "0o". Kao i kod drugih konverzija, moguće je izostaviti ovaj prefiks formatiranjem stringa.

### Stepenovanje - **pow()**

Ova funkcija vraća stepen argumenta **base** na **exp**. Moguće je ubaciti i moduo kao opcioni argument.

### Zaokruživanje - **round()**

Ova funkcija vraća broj zaokružena na željeni broj decimala.

### Sumiranje - **sum()**

Ova funkcija vraća sumiranu vrednost elemenata ne-primitive.

#### **Napomena:**

Za veći izbor funkcija, treba uvesti modul **math**. O math modulu, kao i o drugim modulima, biće reči u posebnoj lekciji.

## ▼ Poglavlje 6

### Pokazna vežba #4

#### UVOD U POKAZNU VEŽBU #4

*Pokazna vežba #4 obrađuje primere iz različitih struktura podataka, kao i rada sa funkcijama.*

Pokazna vežba #4 obrađuje različite strukture podataka, kako primitive, tako i ne-primitive.

Najpre je dat zadatak spajanje dva imenika u jedan, a posle je dato ručno sortiranje liste, sa korišćenjem pomoćne liste, i bez.

Nastavak predstavlja zadatak koji od sortirane liste menja mesta najmanjem i najvećem elementu liste.

Nakon toga, biće obrađen primer koji opisuju operacije rad za funkcijama.

Dat je zadatak u kom treba napisati funkciju koja vrši transformaciju vrednosti otpornika vezanih u trougao, u otpornike vezanih u zvezdu.

Procenjeno trajanje pokazne vežbe iznosi 45 minuta.

#### SPAJANJE DVA IMENIKA U JEDAN, RUČNO SORTIRANJE LISTE

*Iako postoji `.sort()` metoda za sortiranje, potrebno je znati i ručno sortirati elemente liste.*

##### **Zadatak #1** (10 minuta)

Napisati program koji će spojiti dva imenika u jedan.

```
dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}
dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}

dict3 = {**dict1, **dict2}
print(dict3)
```

Komanda `**` u kontekstu poziva funkcija odnosi se za raspakovanje imenika u drugi imenik. Na taj način ostaće *ključ: vrednost*.

The screenshot shows a Visual Studio Code interface. The left sidebar shows a file tree with several Python files in the 'L03' and 'L04' folders. The main editor window contains the following Python code:

```

104 > primer_pv01.py >-
1  dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}
2  dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
3
4  print(dict1)
5  print(dict2)
6
7  # Spajanje dva imenika u jedan
8
9  dict3 = {**dict1, **dict2}
10 print(dict3)

```

The terminal at the bottom shows the execution of the script and its output:

```

PS D:\Nemanja\Google Drive\Metropolitan\CS324\L03> & D:\anaconda3\python.exe "d:\Nemanja\Google Drive\Metropolitan\CS324\L04\primer_pv01.py"
{'Ten': 10, 'Twenty': 20, 'Thirty': 30}
{'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
{'Ten': 10, 'Twenty': 20, 'Thirty': 30, 'Fourty': 40, 'Fifty': 50}
PS D:\Nemanja\Google Drive\Metropolitan\CS324\L03>

```

Slika 6.1 Spajanje dva imenika u jedan. [Izvor: Autor]

### Zadatak #2 (15 minuta)

Napisati program koji će sortirati niz (listu sa numeričkim vrednostima) bez korišćenja `.sort()` metode.

Rešenje #1 - pravljenje novog niza

```

data_list = [-5, -23, 5, 0, 23, -6, 28, 67]
new_list = []

while data_list:
    minimum = data_list[0]
    for x in data_list:
        if x < minimum:
            minimum = x
    new_list.append(minimum)
    data_list.remove(minimum)
print(new_list)

```

Rešenje #2 - direktna zamena mesta elemenata u nizu

```

data_list = [-5, -23, 5, 0, 23, -6, 28, 67]
n = len(data_list)
for i in range(n):
    for j in range(i + 1, n):
        if(data_list[i] > data_list[j]):
            temp = data_list[i]
            data_list[i] = data_list[j]
            data_list[j] = temp
print(data_list)

```

## ZAMENA VREDNOSTI NAJMANJEG I NAJVEĆEG ELEMENTA NIZA

*Često pitanje na intervjuima jeste zamena vrednosti bez korišćenja pomoćne promenljive.*

### Zadatak #3 (10 minuta)

Napisati program koji će sortirati niz od 10 elemenata. Zatim napisati funkciju koja kao ulazni parametar uzima niz, i menja mesta prvom i poslednjem članu, bez korišćenja dodatne pomoćne promenljive.

```
import random
lst_ran = []
for i in range(10):
    lst_ran.append(random.randint(0, 1000))
print(lst_ran)

lst_ran.sort()
print(lst_ran)

def swap_lst(lst):
    lst[0] = lst[0] + lst[-1]
    lst[-1] = lst[0] - lst[-1]
    lst[0] = lst[0] - lst[-1]
    return lst

print(swap_lst(lst_ran))
```

## FUNKCIJA SA VIŠE ARGUMENATA I VIŠE POVRATNIH VREDNOSTI

*Zadatak koji ima tri ulazna parametra, a na izlazu daje listu od tri vrednosti.*

### Zadatak #4 (10 minuta)

Zadatak #4 opisuje pisanje funkciju sa više argumenata i više povratnih vrednosti, kao elemente liste.

Napisati funkciju koja računa transformaciju ulaznih parametara, i kao izlaz daje listu tri elemenata. Transformacija se računa na sledeći način:

$$R_1 = \frac{R_b R_c}{R_a + R_b + R_c}, R_2 = \frac{R_a R_c}{R_a + R_b + R_c}, R_3 = \frac{R_a R_b}{R_a + R_b + R_c}$$

**Zanimljivost:**

Ova transformacija se često javlja u elektrotehnici/elektronici.

Više detalja na: [https://en.wikipedia.org/wiki/Y-%C0%94\\_transform](https://en.wikipedia.org/wiki/Y-%C0%94_transform)

```
def triange_to_star(Ra, Rb, Rc):  
    R1 = (Rb * Rc)/(Ra + Rb + Rc)  
    R2 = (Ra * Rc)/(Ra + Rb + Rc)  
    R3 = (Ra * Rb)/(Ra + Rb + Rc)  
    return [R1, R2, R3]  
  
star = triange_to_star(100, 14000, 10000)  
print(star)
```

## ▼ Poglavlje 7

### Individualna vežba #4

#### UVOD U INDIVIDUALNU VEŽBU #4

*Individualna vežba #4 odnosi se na rad sa funkcijama.*

Individualna vežba #4 daje dva primera u kojima se radi sa funkcijama.

Procenjeno trajanje individualnih vežbi iznosi 90 minuta.

#### INDIVIDUALNA VEŽBA #4

*Napisati jednostavne programe u Python programskoj jeziku koristeći instalirana okruženja.*

##### **Zadatak #1** (60 minuta)

Napisati program za unos imena, prezimena, godina rođenja, broja indeksa.

Zatim, napraviti imenik koji će kao ključ imati šifru predmeta, a vrednost ocena iz tog predmeta. Ubacite samo predmete koje ste položili.

Izračunati prosečnu ocenu tokom studiranja.

a) Napraviti novu promenljivu koja je string i koja je oblika:

"Ime Prezime, **broj\_indeksa**, rođen/a **godina\_rođenja** ima trenutno prosečnu osenu **prosečna\_ocena**" i štampati tu promenljivu.

b) Uraditi isto, ali bez posebne string promenljive.

##### **Zadatak #2** (30 minuta)

Napisati dve funkcije za transformaciju ulaznih vrednosti u izlazne parametre.

Prva funkcija na osnovu **Ra**, **Rb** i **Rc** računa **R1**, **R2**, i **R3**.

Druga funkcija na osnovu **R1**, **R2** i **R3** računa **Ra**, **Rb**, i **Rc**.

U glavnom programu uneti "1" za prvu transformaciju, a "2" za drugu transformaciju.

Ukoliko se nešto treće unese, program ispisuje grešku.

Transformacije se računaju na sledeći način:

$$R_1 = \frac{R_b R_c}{R_a + R_b + R_c}, R_2 = \frac{R_a R_c}{R_a + R_b + R_c}, R_3 = \frac{R_a R_b}{R_a + R_b + R_c}$$
$$R_a = \frac{R_1 R_2 + R_2 R_3 + R_3 R_1}{R_1}, R_b = \frac{R_1 R_2 + R_2 R_3 + R_3 R_1}{R_2}, R_c = \frac{R_1 R_2 + R_2 R_3 + R_3 R_1}{R_3}$$

## ✓ Poglavlje 8

### Domaći zadatak #4

#### DOMAĆI ZADATAK

*Domaći zadatak #4 se okvirno radi 2.5h*

##### **Zadatak #1**

Napisati funkciju sa dva ulazna parametra: **ime** i **br\_indeks**

Vrednost **br\_indeks** (četvorocifrena promenljiva) određuje broj elemenata u nizu koji se kreira unutar funkcije.

Elementi niza jesu sledeći:

- Ukoliko je broj slova u promenljivoj **ime** paran, generisati cele brojeve od 0 do **br\_indeks** sa uniformnom raspodelom
- Ukoliko je broj slova u promenljivoj **ime** neparan, generisati razlomljene brojeve u opsegu od negativne vrednosti prve dve cifre, to pozitivne vrednosti druge dve cifre u **br\_indeks**.

Koristiti **random** biblioteku za generisanje brojeva.

Sortirati niz korišćenjem for petlje i privremene promenljive.

Kao povratnu vrednost vratiti sortirani niz.

U glavnom programu uneti promenljive **ime** i **br\_indeks**, pozvati funkciju sa tim promenljivima i odštampati sortirani niz.

##### **Zadatak #2**

Napisati rekurzivnu funkciju za računanje Fibonačijevog niza do elementa **n**.

Napisati iterativnu funkciju za računanje Fibonačijevog niza do elementa **n**.

Opisati sličnosti i razlike u ova dva pristupa.

**Predaja domaćeg zadatka:**

##### **Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije nedelju dana nakon predavanja za 100% poena. Nakon toga poeni se umanjuju za 50%.

##### **Online studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganja ispita. **Domaći zadaci se  
brane!**

Svi studenti domaći zadatak poslati dr Nemanji Zdravkoviću:  
[nemanja.zdravkovic@metropolitan.ac.rs](mailto:nemanja.zdravkovic@metropolitan.ac.rs)

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Zaključak lekcije #4*

#### **Rezime:**

U ovoj lekciji bilo je reči o strukturama podataka, ugrađenim, ali i kako se mogu praviti i korisničke strukture, koje se uglavnom mogu kategorisati u nekoliko grupa.

Nakon toga, bilo je reči o funkcijama, opsegu promenljivih, ulaznim i izlaznim parametrima, kao i o anonimnim i brzim lambda funkcijama.

Sve oblasti propraćene su adekvatnim primerima.

#### **Literatura:**

1. David Beazley, Brian Jones, Python Cookbook: Recipes for Mastering Python 3, 3rd edition, O'Reilly Press, 2013.
2. Mark Lutz, Learning Python, 5th Edition, O'Reilly Press, 2013.
3. Andrew Bird, Lau Cher Han, et. al, The Python Workshop, Packt Publishing, 2019.
4. Al Sweigart, Automate the boring stuff with Python, 2nd Edition, No Starch Press, 2020.





## CS324 - SKRIPTING JEZICI

Python i OOP

Lekcija 05

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 05

### *PYTHON I OOP*

- ✓ Python i OOP
- ✓ Poglavlje 1: Razlika između OOP i proceduralnog programiranja
- ✓ Poglavlje 2: Osnove OOP-a u Python 3.x jeziku
- ✓ Poglavlje 3: Klasa i objekti
- ✓ Poglavlje 4: Rad sa klasama i instacama
- ✓ Poglavlje 5: Klase naspram modula
- ✓ Poglavlje 6: Pokazna vežba #5
- ✓ Poglavlje 7: Individualna Vežba #5
- ✓ Poglavlje 8: Domaći zadatak #5
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ✓ Uvod

# UVOD

### *Uvod u lekciju #5*

U petoj lekciji prelazi se na paradigmu objektno-orientisanog programiranja u Python 3.x jeziku.

OOP je paradigma koja se bazira na konceptu objekata. Objekti sadrže podatke, u formi polja, odnosno atributa, dok se sam kod, odnosno procedure, nazivaju metodi.

Za razliku od proceduralnog koje prenosi podatke od poziva procedura do poziva procedura dok se ne dobije željeni rezultat, objektno-orientisano programiranje enkapsulira podatke i ponašanje podataka u objekte.

Većina ovih osnovnih koncepata je verovatno već poznato iz ranijih predmeta, ali u ovoj lekciji biće reči o nekim specifičnostima koje izdvajaju Python.

U Python 3.x jeziku postoji opseg imenskih prostora, (en. [namespace](#)), koji se koristi prilikom definisanja klasa. U ovom kontekstu, imenski prostor predstavlja mapiranje imena na objekte.

Ono što je specifično za Python jeste da ukoliko ne postoji izjava koje je globalna ili ne-lokalna, imenovanje uvek ide u opsegu koji je najmanji (en. [innermost scope](#)).

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 1

# Razlika između OOP i proceduralnog programiranja

## RAZLIKE IZMEĐU OOP I PROC. PROGRAMIRANJA

*Programi napisani u OOP jezicima jesu projektovani tako da su sastavljeni od objekata i međusobne interakcije tih objekata.*

### **Pitanje:**

*U koju paradigmu spadaju proceduralno i objektno-orientisano programiranje?*

*Koje su glavne osobine te paradigmе?*

Objektno-orientisano programiranje (en. object-oriented programming, OOP) jeste paradaigma programiranja koja se zasniva na konceptu objekta.

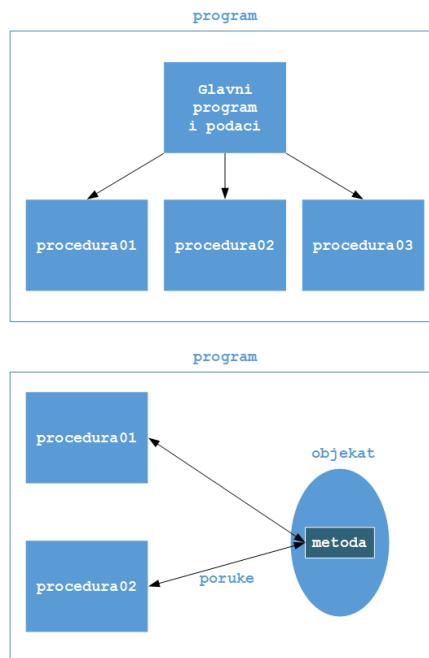
Objekat (en. object) predstavlja osnovni gradivni element ove paradaigme. Objekat može sadržati podatke (u obliku polja, tj. atributa ili osobina) i kod (u obliku procedura tj. metoda).

*Programi napisani u OOP jezicima jesu projektovani tako da su sastavljeni od objekata i međusobne interakcije tih objekata.*

Glavna razlika između paradaigmi proceduralnog programiranja i OOP-a jeste ta da se kod proceduralnog programiranja program sastoji od poziva procedura dok se ne dobije krajnji rezultat. Kod OOP-a objekti i njihova interakcija slanjem poruka čine sastav programa.

### **Podsetnik:**

*Na slici je vizuelno opisana razlika između programa napisanog na proceduralni način, i programa napisanog kroz OOP.*



Slika 1.1 Proceduralno programiranje naspram OOP. [Izvor: Autor.]

## PRIMER PISANJA PROGRAMA U PROC. I OOP PARADIGMI

*Proceduralno programiranje nema fleksibilnost koje pruža OOP pristup.*

### **Primer:**

Napisati proceduralni i OOP program koji će računati obim i površinu pravougaonika na osnovu unetih stranica.

### **Proceduralno programiranje**

```
# Pravougaonik
a = 30
b = 40
obim = 2 * (a + b)
povrsina = a * b

print(f"Pravougaonik stranica {a} i {b} ima obim {obim} i površinu {povrsina}.")
```

### **Objektno-orjentisano programiranje**

```
# Klasa pravougaonik
class pravougaonik:
    def __init__(self, a, b):
        self.a = a
        self.b = b
    def obim(self):
        return 2 * (self.a + self.b)
```

```
def povrsina(self):  
    return self.a * self.b  
  
p = pravougaonik(30, 40)  
print(f"Pravougaonik stranica {p.a} i {p.b} ima obim {p.obim()} i površinu  
{p.povrsina()}")
```

**Pitanje:**

Proširenje zadatka:

Uneti 10 pravougaonika sa različitim dužinama stranica i izračunati im površinu i obim.  
Kako bi se ovaj zadatak proširio u proceduralnom, a kako u OOP pristupu?

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## VIDEO O RAZLIKAMA IZMEĐU DVE PARADIGME IMPERATIVNOG PROGRAMIRANJA

*U nastavku je video lekcija koja objašnjava razlike između proceduralnog i OOP pristupa programiranju.*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 2

# Osnove OOP-a u Python 3.x jeziku

## OSNOVNI KONCEPTI OOP-A

*Python je i proceduralni i OOP jezik.*

Do sada viđeno da je Python 3.x jezik koji podržava proceduralno programiranje, a u prošlom primeru je pokazano da podržava i OOP. Python 3.x kao programski jezik stoga podržava više pristupa programiranju.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

Objektno-orientisano programiranje se najbolje može opisati kroz četiri glavna koncepta:

- Enkapsulacija (en. **encapsulation**)
- Apstrakcija (en. **abstraction**)
- Nasleđivanje (en. **inheritance**)
- Polimorfizam (en. **polymorphism**)

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ENKAPSULACIJA I APSTRACKIJA PODATAKA

*Enkapsulacija i apstrakcija podataka štite kod unutar klase.*

### **Enkapsulacija**

Različiti objekti unutar jednog programa u opštem slučaju moći će da komuniciraju jedan sa drugim. Ukoliko programer želi da zaustavi objekte da međusobno interaguju, onda treba da se ti objekti enkapsuliraju u izdvojene klase.

Kroz proces enkapsulacije, klase ne mogu da promene ili da interaguju sa specifičnim promenljivama i funkcijama objekta.

Analogno kapsuli koja zadržava sadržaj unutar ljuške, princip enkapsulacije pravi zaštitnu barijeru oko informacije koja je razdvaja od ostatka koda.

Programeri na taj način mogu da repliciraju objekat kroz različite delove programa ili ka drugim programima.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### Apstrakcija

Apstrakcija (podataka) se može smatrati kao proširenje enkapsulacije zbog toga što sakriva određene osobine i metode ostatku koda u programu, da bi na taj način interfejs ka objektu bio pojednostavljen.

Apstrakcija podataka je korisna prilikom pisanja programa iz više razloga ali prvenstveno pomaže izolaciji uticaja promene koda ukoliko se jave problemi unutar objekta, to će uticati samo na promenljive unutar, a ne na kod glavnog programa.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## NASLEĐIVANJE I POLIMORFIZAM

*Nasleđivanje i polimorfizam jesu koncepti OOP-a koji smanjuju pisanje istog koda više puta.*

### Nasleđivanje

Koncept nasleđivanja omogućava programerima da prilikom pisanja koda produže funkcionalnost postojećih klasa unutar koda da ne bi došlo do stalnog ponavljanja koda.

Može da se desi da elementi koda uključuju iste osobine, koje poseduju više pojedinačnih metoda. Umesto da se te osobine i metode svaki put ponovo definišu svaki put kada se nađe na element koda koji ih sadrži, mogu se definisati jednom kao generički objekat (en. generic object). Na taj način specifični objekti mogu naslediti osobine i metode, smanjujući ponavljanje koda.

Glavi objekat postaje nadkласа ili super klasа (en. superclass) a izveden objekat postaje подкласа ili изведенa klasа (en. subclass).

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### Polimorfizam

Koncept polimorfizma omogućava programerima da prilikom pisanja koda pruži mogućnost da jednom napisani kod daje različit rezultate u zavisnosti od ulaza.

Polimorfizam predstavlja obezbeđivanje jedinstvenog interfejsa prema entitetima različitih tipova.

Može biti *ad-hoc* polimorfizam, parametarski polimorfizam, hijerarhijski polimorfizam i implicitni polimorfizam.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 3

### Klase i objekti

#### POJAM KLASE

*Klase predstavlja šablon (en. blueprint) za pravljenje objekata*

Klase (en. `class`) pruža mogućnost združivanja podataka i funkcionalnosti.

Kreiranjem nove klase kreira se novi tip objekta, koji potom dozvoljava kreaciju novih instanci tog tipa podataka.

Svaka instanca klase ima atribute koje su joj dodeljeni. Instance klase takođe imaju metode, koje su definisane unutar klase.

U poređenju sa ostalim programskim jezicima, mehanizam klase u Python 3.x jeziku je sličan mehanizmima u C++ jeziku i Modula-3 jeziku.

Klase u Python jeziku pružaju sve standarde objektno-orientisane paradigme programiranja. Osobina nasleđivanja dozvoljava više baznih (super) klase, izvedena klasa može izvršiti redefinisanje metode (en. `method override`) svoje (ili svojih) super klase, a metoda može pozvati metodu bazne klase istog imena.

Objekat (en. `object`) može sadržati proizvoljan broj podataka bilo kog tipa.

Klase (i moduli) u Python 3.x jeziku putem osobine dinamičkog tipiziranja se prave prilikom izvršenja (en. `runtime`) i mogu se modifikovati nakon kreacije.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

#### IMENSKI PROSTOR I OPSEG U PYTHON 3.X JEZIKU

*Treba voditi računa o opsegu imenskog prostora prilikom definisanja klase - ne pretražuju se svi opsezi podjednako*

Imenski prostor (en. `namespace`) prestavlja mapiranje imena na objekte.

Najveći broj imenskih prostora u Python jeziku jesu implementirani kroz imenike. Primeri imenskih prostora uključuju skup ugrađenih imena (primer može biti ugrađena funkcija), globalna imena (primer unutar modula), i lokalna imena (primer unutar funkcije)

Imenski prostori se kreiraju u različitim trenucima i imaju različit vek trajanja.

Imenski prostor koji sadrži ugrađena imena kreira se prilikom inicijalizacije Python interpretera, i ne briše se.

Globalni imenski prostor za modul se pravi kada se učita definicija modula, i takođe ostaju dok se interpreter ne isključi.

Lokalni imenski prostor funkcije kreira se kada se poziva funkcija, i briše se kada se funkcija završi (vrati povratnu vrednost, ili vrati izuzetak koji je može obraditi).

<https://docs.python.org/3/tutorial/classes.html>

**Opseg** (en. *scope*) predstavlja deo koda u Python jeziku u kome je imenski prostor direktno dostupan.

Iako se opsezi statički određuju, koriste se dinamički. U bilo kom vremenu tokom izvršenja, postoje tri ili četiri ugnježđena opsega čiji su imenski prostori direktno dostupni:

- **Unutrašnji opseg** (en. *innermost scope*), koji se prvi pretražuje i sadrži lokalna imena,
- **Opseg okružujućih funkcija** (en. *enclosing functions scope*), koji se pretražuje počevši za najbližom okružujućom funkcijom, i sadrži ne-lokalna, ali i ne-globalna imena,
- **Opseg globalnih imena** modula koji se pretražuje preposlednji,
- **Spoljašnji opseg** (en. *outermost scope*) koji sadrži imenski prostori sa ugrađenim imenima i koje se pretražuje poslednji.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ 3.1 Metode u klasama

ŠTA PREDSTAVLJA METODA KLASE?

*Metode pružaju interfejs ka i od objekta.*

**Definicija:**

Metoda (en. *method*) u objektno-orientisanom programiranju prestavlja proceduru koja sadrži poruku i objekat.

Metode se definišu u klasi, i pružaju interfejs koji ostale klase koriste da pristupe i izmene osobine objekta.

**Zanimljivost:**

*U Python 3.x jeziku, metoda nije unikatna za instance klase. Drugi tipovi objekata takođe mogu imati metode.*

*Primer: Objekti tipa list imaju metode .append(), .insert(), .sort() i sl.*

U opštem slučaju, može se reći da metoda predstavlja funkciju koja pripada objektu.

Kao i u mnogim drugim OOP jezicima, postoje specijalni tipovi metoda.

### Konstruktor

**Konstruktor** (en. **constructor**) predstavlja specijalnu metodu koju Python poziva prilikominstanciranja objekta, korišćenjem definicija unutar klase.

Python se oslanja na konstruktor da izvrši inicijalizaciju, odnosno da dodeli vrednosti svakoj instanci objekta kada se napravi.

U Python 3.x jeziku ime konstruktora je **`__init__()`**

Konstruktor može prihvati argumente kada je neophodno da se napravi objekat. Kada se napravi klasa bez konstruktora, Python automatski napravi default-ni konstruktor koji zapravo ništa ne radi.

Svaka klasa mora imati konstruktor, makar i default konstruktor. Sintaksa konstruktora je sledeća:

```
def __init__(self):  
    # telo konstruktora
```

## VRSTE KONSTRUKTORA

*Konstruktor može biti default-ni ili parametrizovan*

### Default konstruktor

Ovakav konstruktor ne prihvata nikakve argumente. Njegova definicija ima samo jedan argument (**self**) koji je referenca na instancu koja se konstruiše.

#### Primer:

Napraviti klasu **CS324** koja sadrži atribut puno ime predmeta, i metodu koja štampa puno ime predmeta.

```
class CS324:  
  
    # default konstruktor  
    def __init__(self):  
        self.naziv = "Skripting jezici"  
  
    # metoda za štampanje  
    def printNaziv(self):  
        print(self.naziv)  
  
    # kreiranje objekta  
predmet = CS324()
```

```
# poziv metode
predmet.printNaziv()
```

### Parametrizovan konstruktor

Konstruktor koji pored self reference sadrži dodatne parametre naziva se parametrizovan konstruktor. Prvi parametar je i dalje referenca na sebe, dok ostale argumente prosleđuje programer.

#### Primer:

Napraviti klasu **fitPredmet** koja kao ulazne parametre ima šifru u naziv predmeta. Klasa sadrži artibut godina (početna vrednost nula), i metode za unos vrednosti godine (na kojoj se služa predmet), kao i metodu za štampanje šifre, naziva i godine na kojoj se sluša.

```
class fitPredmet:
    godina = 0
    def __init__(self, sifra, naziv):
        self.sifra = sifra
        self.naziv = naziv
    def unosGodine(self):
        self.godina = int(input("Uneti godinu na kojoj se sluša predmet: "))
    def predmetPrint(self):
        print(f"Predmet {self.sifra} - {self.naziv} se sluša na {self.godina}. godini studija.")

cs324 = fitPredmet("CS324", "Skripting Jezici")
cs324.unosGodine()
cs324.predmetPrint()
```

## VRSTE KONSTRUKTORA - VIDEO OBJAŠNJENJE

*U nastavku su dati video materijali koji objašnjavaju vrste konstruktora u Python 3.x jeziku*

### Default konstruktor

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### Parametrizovan konstruktor

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## 3.2 Nasleđivanje, kompozicija, delegacija

### NASLEĐIVANJE U PYTHON 3.X JEZIKU

*Izvedena klasa nasleđuje metode i atribute osnovne klase*

Nasleđivanje (en. *inheritance*) je jedan od osnovnih principa objektno-orientisanog programiranja.

Kod nasleđivanja, kao što je rečeno imamo super klasu (nadklasu, ili klasu roditelja) i izvedenu klasu (podklasu, klasu deteta) koja nasleđuje atribute i metode svoje super klase.

Može se reći da izvedena klasa proširuje super klasu.

Super klasu ne treba posebno definisati, već se definiše kao normalna klasa.

**Primer:**

Napraviti klasu **vozilo** koje ima atribute godina proizvodnje i broj točkova.

Izvesti klasu **automobil** koje nasleđuje sve od klase vozilo.

```
class vozilo:
    godinaProizvodnje = None
    brojTockova = None

class automobil(vozilo):
    pass

veh1 = vozilo()
veh1.godinaProizvodnje = 2000
veh1.brojTockova = 4

print(type(veh1))

auto1 = automobil()
auto1.godinaProizvodnje = 2020
auto1.brojTockova = 4

print(type(auto1))
```

Kod izvedene klase moguće je redefinisati postojeću metodu super klase..

**Primer:**

Napraviti klasu **playerCharacter** bez ulaznih argumenata, koja ima atribut **stats** koji je inicijalno niz od šest nula. Definisati metodu **setStats** kojom se ubacuju šest celobrojnih parametra, i metodu **showStats** kojom se štampa stats.

Izvesti klasu **human** iz klase **playerCharacter**, i redefinisati **setStats** tako da se na svaki ubačeni broj doda jedinica.

Napraviti instancu **playerCharacter** i **human**, i dati im iste **stats**, i nakon toga ih prikazati na izlazu.

```
class playerCharacter():
    stats = [0, 0, 0, 0, 0, 0]
    def setStats(self):
        for i in range(6):
            self.stats[i] = int(input(f"Uneti {i + 1}. stat: "))
    def showStats(self):
        print(f"Stats: {self.stats}")

pc1 = playerCharacter()
pc1.showStats()

pc1.setStats()
pc1.showStats()

class human(playerCharacter):
    def setStats(self):
        for i in range(6):
            self.stats[i] = int(input(f"Uneti {i + 1}. stat: ")) + 1

pc2 = human()
pc2.setStats()
pc2.showStats()
```

## NASLEĐIVANJE NASPRAM KOMPOZICIJE

*Veza između dva objekta u nasleđivanju je "je veza" dok je pri kompoziciji "ima veza"*

### Video iz prethodnog primera

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

Nasleđivanje jeste dobro ukoliko želimo da prenesemo sve atribute i metode super klase u izvedenu klasu.

Tada je izvedena klasa ujedno i super klasa, odnosno postoji tzv. *je veza* između super klase i izvedene klase.

Sa druge strane, ukoliko postoji tzv. *ima veza* između dva objekta, onda je u pitanju kompozicija.

### Primer za nasleđivanje:

Definiše se super klasa **povrće**, i izvedena klasa **krompir**. Postoji *je veza*, jer je krompir povrće, i naslediće sve elemente te klase.

**Primer za kompoziciju:**

Definišemo klasu **akumulator**. Zatim, definišemo klasu **automobil** čiji jedan atribut jeste instanca klase akumulator. U ovom slučaju postoji *ima veza*, jer će svaki automobil imati akumulator.

*Detaljnije razlike između kompozicije i nasleđivanja obrađuje se u drugim predmetima.*

## ✓ Poglavlje 4

### Rad sa klasama i instacama

#### INSTANCA KAO KONKRETNI OBJEKAT

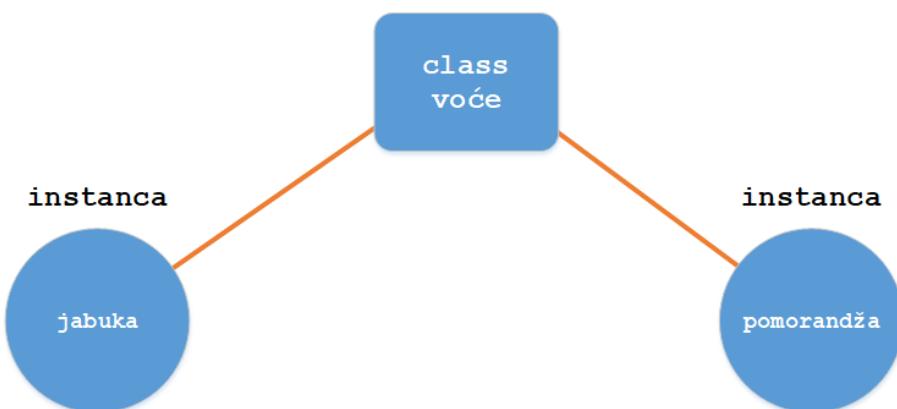
*Instanca ima životni ciklus obično tokom izvršenja programa.*

U OOP paradigmama, instanca (en. instance) predstavlja konkretan primerak bilo kog objekta.

*Instanca kao takva ima životni ciklus obično tokom izvršenja programa.*

Često je instanca sinonim za objekat jer oba termina označavaju konkretnu realizaciju neke klase. Međutim, kada se koristi termin instanca, naglašava se da je u pitanju poseban identitet objekta.

Kreacija instance jeste instaciranje (en. instantiation)



Slika 4.1.1 Klasa i instance klase [Izvor: Autor.]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

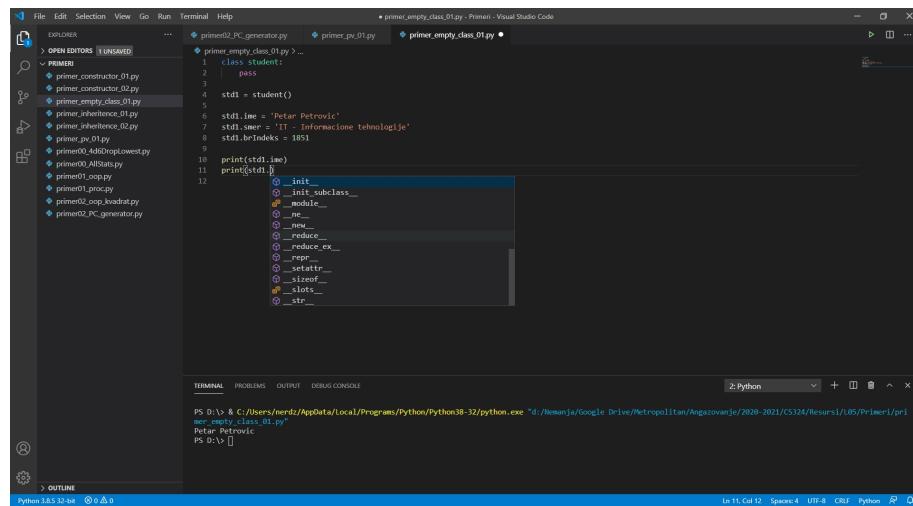
#### PRAZNA KLASA

*Moguće je napraviti praznu klasu koju kasnije mogu*

U Python 3.x programskom jeziku moguće je napraviti praznu klasu, odnosno klasu koja ne sadrži nikakve atribute i metode.

Nakon toga, mogu se naknadno dodati atributi.

Međutim, u klasičnom OOP pristupu ovo se ne preporučuje.



Slika 4.1.2 Definisanje prazne klase student, i dodavanje atributa nakon instaciranja. [Izvor: Autor.]

### Pitanje:

Kada i gde se koristi prazna klasa?

```
class student:  
    pass  
  
std1 = student()  
  
std1.ime = 'Petar Petrovic'  
std1.smer = 'IT - Informacione tehnologije'  
std1.brIndeks = 1851  
  
print(std1.ime)
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## 4.1 Uvoz klase

### UVOZ KLASE U PYTHON 3.X JEZIKU

*Prednost OOP-a u Python jeziku jeste ta što može da se iskoriste klase iz drugih datoteka ukoliko je poznata putanja to tih datoteka*

U proceduralnom programiranju čest je slučaj da je potrebno se sve promenljive i funkcije (koje nisu ugrađene) nalaze u istoj datoteci.

U Python 3.x programskom jeziku moguće je izvršiti uvoz klase (en. **class import**) i iz drugih datoteka.

Takođe, nije neophodno ni da se sve datoteke čije klase želimo koristiti nalaze u istom radnom direktorijumu, već je moguće koristiti datoteke iz drugih direktorijuma.

Potrebno je znati relativnu putanju od izvorne do odredišne datoteke, kao i da u direktorijumu iz koje uzimamo klase postoji prazna `__init__.py` datoteka, koja označava je moguće uvoziti datoteke iz tog direktorijuma.

**Primer:**

Napraviti klasu koja simulira bacanje kockica sa metodom koja pokazuje rezultat. U posebnoj datoteci uvesti klasu iz prve datoteke i napraviti novu instancu te klase. Isprobati funkcionalnost.

*Alternativno: Realizovati metodu vraćanja rezultata kao lambda funkcija.*

<https://stackoverflow.com/questions/4142151/how-to-import-the-class-within-the-same-directory-or-sub-directory>

```
# bacanjeKockica
import random

class d6roller():
    def rolld6(self):
        return random.randint(1,6)

# glavniProgram
import d6roller as d6
x = d6.d6roller()
print(x.rolld6())
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 5

### Klase naspram modula

## KLASA NASPRAM MODULA U PYTHON 3.X JEZIKU

*Klase i modul mogu naizgled da služe istoj svrsi, ali nije tako*

#### **Klasa**

Klase u Python 3.x jeziku, kao i kod drugih objektno-orientisanih programske jezike predstavlja osnovu za apstrakciju podataka.

Klase se koristi za apstrakciju zajedničkih karakteristika različitih objekata. Klase enkapsulira podatke i operacije za buduću upotrebu.

Klase se može definisati u glavnem programu ili unutar modula koji glavni program uveze.

Iz klase kreiraju se instance te klase. To su pojedinačni objekti sa jedinstvenim skupom podataka u svojim atributima.

Klase se može proširiti ili modifikovati korišćenjem osobine nasleđivanja.

#### **Modul**

Svaka .py datoteka može se smatrati da je **modul** (en. **module**). Nakon pisanja skript datoteka, definišu se funkcije i promenljive. Te funkcije i promenljive se mogu ponovo iskoristiti tako što će se uvesti (en. **import**) ta datoteka kao modul (bez ekstenzije **.py**)

Može se reći da je modul enkapsulira kod koji se može ponovo iskoristiti. Moduli sadrže funkcije ali i mogu sadržati i klase.

Može postojati samo jedan modul. Kada se uveze modul, to je isti objekat.

Modul se *ne može* proširivati ili modifikovati kao što klasa može (kroz nasleđivanje).

## KADA TREBA KORISTITI KLASE, A KADA UVESTI MODULE?

*Koristiti klase za šablove, a module za dodatnu funkcionalnost*

#### **Podsetnik:**

*Koristiti klase kao šablove za objekte koje modeluju doen problema.*

*Korstiti module za dodatnu funkcionalnost u kodu.*

### **Paket**

Za razliku od pojedinačnih modula, koji se čuvaju kao `.py` datoteke, paket predstavlja direktorijum u kome se nalaze pojedinačni moduli (obično slične problematike).

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 6

### Pokazna vežba #5

#### UVOD U POKAZNU VEŽBU #5

*U pokaznoj vežbi #5 prolaze se klase, atributi i metode, kao i izvođenje nove klase iz superklase.*

U pokaznoj vežbi proći će se kroz zadatak koji obuhvata sve tematske jedinice ove lekcije:

- Klase
- Metoda klase
- Superklasa i izvedena klasa
- Redefinisanje metode klase

Procenjeno trajanje pokazne vežbe iznosi 90 minuta.

#### PRIMER 1: KLASSA WARRIOR

*Primer klase sa parametrizovanim konstruktorom i različitim metodama*

**Zadatak #1** (20 minuta)

Napraviti klasu **warrior** koja sadrži atribute **name**, **health**, **armorClass**, **attackBonus**, i **damage**.

Klasa ima sledeće metode:

- **rollForAttack**: Vraća vrednost attackBonus i nasumičnog celog broja između 1 i 20 (simulira bacanje 20-strane kockice)
- **checkHit**: proverava da li je svoj AC manji od vrednosti napada.
- **takeDamage**: smanji svoj health za ulazni parametar damage.

Napraviti dva objekta klase warrior. Neka jedan napadne drugog. Štampati preostali **health**.

```
class warrior():
    def __init__(self, name, health, armorClass, attackBonus, damage):
        self.name = name
        self.health = health
        self.armorClass = armorClass
        self.attackBonus = attackBonus
        self.damage = damage
```

```
def rollForAttack(self):  
    import random  
    print(self.name, "attacks!")  
    return self.attackBonus + random.randint(1, 20)  
  
def checkHit(self, attackRoll):  
    if attackRoll >= self.armorClass:  
        return True  
    else:  
        return False  
  
def takeDamage(self, damage):  
    self.health -= damage  
  
warr1 = warrior("Rogar", 30, 14, 5, 10)  
print(f"{warr1.name} has", warr1.health, "HP left.")  
warr2 = warrior("Nimble", 20, 12, 6, 8)  
print(f"{warr2.name} has", warr2.health, "HP left.")  
  
# Rogar attacks Nimble  
atk1 = warr1.rollForAttack()  
if warr2.checkHit(atk1):  
    print(f"Rolled {atk1}, AC is {warr2.armorClass}, hit!")  
    warr2.takeDamage(warr1.damage)  
else:  
    print(f"Rolled {atk1}, AC is {warr2.armorClass}, miss!")  
  
print(f"{warr2.name} has", warr2.health, "HP left.")
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PRIMER 2: PLAYER CHARACTER I IZVEDENA HUMAN KLASA

*Primer u pokaznoj vežbi pokazuje inicijalizaciju klase, ali i redefinisanje metode u izvedenoj klasi.*

### **Zadatak #2 (25 minuta)**

Napraviti klasu **playerChar** koja ima sledeće atribute:

- Ime
- Pol
- Godine

I jedan niz za statistiku (*Strength, Dexterity, Constitution, Intelligence, Wisdom i Charisma*) koji je inicijalno popunjeno nulama.

Napraviti metodu za popunjavanje niza na sledeći način:

Simulirati bacanje kockice četiri puta, odbacivanje najmanjeg broja, i sabiranja preostala tri, kao posebnu metodu, i onda metodu koja poziva bacanje kockica, i popunjava statistiku lika.

Napraviti izvedenu klasu **human** iz klase **playerChar**, koja redefiniše popunjavanje statistike, tako što na svaki rezultat bacanja kockice doda 1.

U glavnom programu napraviti objekte pc1 koji je tipa **playerChar**, i pc2, koji je **human**, i popuniti im statistiku. Na kraju štampati statistiku.

```
# D&D 5e player character class

class playerChar:
    # stats are: Strength, Dexterity, Constitution, Intelligence, Wisdom & Charisma
    stats = [0, 0, 0, 0, 0, 0]
    # initialization
    def __init__(self, name, gender, age):
        self.name = name
        self.gender = gender
        self.age = age

    # stat roller method 4d6 drop lowest
    def statRoller(self):
        import random
        stat = []
        for _ in range(4):
            stat.append(random.randint(1, 6))
        stat.sort()
        stat.pop(0)
        return sum(stat)

    # roller for all stats
    def rollStats(self):
        allStats = []
        for _ in range(6):
            x = self.statRoller()
            allStats.append(x)
        return allStats

# human class from playerChar has all stats + 1
class human(playerChar):
    def rollStats(self):
        allStats = []
        for _ in range(6):
            x = self.statRoller()
            allStats.append(x + 1)
        return allStats

# main program
pc1 = playerChar("Random Name Placeholder the Third", "Male", 56)
pc1.stats = pc1.rollStats()
print(f"The character {pc1.name} has the followings stats: \n {pc1.stats}")
```

```
pc2 = human("Gilmithrie of Ashtramoor", "Male", 21)
pc2.stats = pc2.rollStats()
print(f"The character {pc2.name} has the followings stats: \n {pc2.stats}")
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 7

### Individualna Vežba #5

#### UVOD U INDIVIDUALNU VEŽBI #5

*U individualnoj vežbi 5 rade se tri zadatka koje bi studenti mogli samostalno da odrade u toku dva časa.*

Individualna vežba #5 odnosi se na rad sa klasama i objektima.

Procenjeno trajanje individualnih vežbi iznosi 90 minuta.

#### INDIVIDUALNA VEŽBA

*Studenti bi trebalo samostalno da reše sva tri zadatka u vreme trajanja dva časa individualnih vežbi.*

##### **Zadatak #1** (25 minuta)

Napraviti klasu **randomNiz** koji ima atribut broj elemenata u nizu i tip elemenata (u za pozitivne cele brojeve, i za cele brojeve, f za razlomljene brojeve).Napraviti (ručno) metode sortiranje niza u:

- rastućem redosledu
- opadajućem redosledu
- cik-cak redosledu (najmanji element ima prvi indeks, sledeći ima poslednji indeks, nakon njega ima drugi indeks, itd.)

##### **Zadatak #2** (35 minuta)

Napraviti klasu **geometrijskaFigura** koja kao atribute ima dužinu proizvoljnog broja stranica.

Na osnovu broja unetih stranica odrediti o kojoj figuri se radi (trougao, četvorougao, petougao i sl.), i izračunati obim i površinu. Za svaku figuru odrediti da li je moguće konstruisati takvu figuru sa unetim dužinama stranica.

Instancirati nekoliko različitih figura i štampati obim i površinu. Odrediti koja figura ima najmanju površinu i vratiti koja je figura u pitanju. Instancirati trougao, četvorougao, petougao i šestougao sa istim dužinama stranica. Poređati obime u niz u opadajućem redosledu.

##### **Zadatak #3** (30 minuta)

Napraviti klasu **videoSnimak** koja ima kao atribute **ime** i **duzinaTrajanja**.

Izvesti klasu **film** koji ima i atribute **godinalzdanja, zanr, reziser**.

Izvesti klasu **serija** koja ima i atribute **zanr, brojSezona, brojEpPoSezoni**.

Za obe izvedene klase napraviti **rejting** (1 do 5 zvezdica)

Instancirati nekoliko objekata od svake klase.

Optimizovati kod.

## ✓ Poglavlje 8

### Domaći zadatak #5

#### DOMAĆI ZADATAK

*Domaći zadatak #5 se okvirno radi 2h*

##### **Domaći zadatak #5**

###### **Zadatak #1**

Napraviti klasu dokument sa osobinama ime i broj reči. Izvesti klasu knjiga koja ima dodatne osobine autor, žanr, godina izdavanja.

Instancirati 10 knjiga. Napraviti imenik koji će kao ključ imati broj knjige (počevši od lib001) a kao vrednost knjigu. Štampati sve knjige u formatu:

*< broj knjige: žanr, autor, naziv. >*

###### **Zadatak #2**

Napraviti klasu osoba, koja će imati osobine ime i prezime.

Nakon toga, izvesti klasu student koja će imati dodatne osobine broj\_indeksa, smer, i položene ispite.

Položene ispite napraviti kao imenik gde je ključ šifra predmeta, a ocena vrednost. Napraviti dva objekta klase student i popuniti sve osobine.

Naći da li su studenti na istom smeru ili ne, koliko je koji student položio ispite, i da li imaju ispite koje su oba studenta položili.

###### **Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije nedelju dana nakon predavanja za 100% poena. Nakon toga poeni se umanjuju za 50%.

###### **Online studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganja ispita. **Domaći zadaci se brane!**

Svi studenti domaći zadatak poslati dr Nemanji Zdravkoviću:  
[nemanja.zdravkovic@metropolitan.ac.rs](mailto:nemanja.zdravkovic@metropolitan.ac.rs)

## ▼ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Zaključak lekcije #5*

#### **Rezime:**

U ovo lekciji bilo je reči o objektno-orientisanom programiranju u Python 3.x jeziku

Uvedeni su pojmovi klase, objekta i instance, i date su osnovne osobine OOP-a kao paradigme programiranja.

Zatim, bilo je reči o elementima klase, metodama i atributima, i njihovom međusobnom komunikacijom.

Bilo je reči o super klasama i izvedenim klasama, razlikama između klasa i modula, o kojima će biti mnogo više reči u sledećim lekcijama.

Primeri u okviru lekcije kao i zadaci za individualni rad treba da osposobe studente za rad u Python 3.x jeziku kroz paradigmu OOP-a.

#### **Literatura:**

1. David Beazley, Brian Jones, Python Cookbook: Recipes for Mastering Python 3, 3rd edition, O'Reilly Press, 2013.
2. Mark Lutz, Learning Python, 5th Edition, O'Reilly Press, 2013.
3. Andrew Bird, Lau Cher Han, et. al, The Python Workshop, Packt Publishing, 2019.
4. Al Sweigart, Automate the boring stuff with Python, 2nd Edition, No Starch Press, 2020.



## CS324 - SKRIPTING JEZICI

Datoteke i izuzeci

Lekcija 06

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 06

### *DATOTEKE I IZUZECI*

- ✓ Datoteke i izuzeci
- ✓ Poglavlje 1: Čitanje iz datoteka
- ✓ Poglavlje 2: Pisanje u datoteku
- ✓ Poglavlje 3: Čuvanje podataka
- ✓ Poglavlje 4: Izuzeci i obrada izuzetaka
- ✓ Poglavlje 5: Debagovanje u Python 3.x jeziku
- ✓ Poglavlje 6: Pokazna vežba #6
- ✓ Poglavlje 7: Individualna vežba #6
- ✓ Poglavlje 8: Domaći zadatak #6
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ▼ Uvod

# UVOD

### *Uvod u datoteke i obradu izuzetaka*

Kao i kod većine programskih jezika, postoje nekoliko načina se da izlaz iz programa pokaže korisniku. Podaci se mogu štampati u obliku koji korisnik može da pročita, ali, možda još bitnije, može se napisati u datoteku koja se može kasnije iščitati ili iskoristiti za druge svrhe. Bilo da radite u Python-u za razvoj web ili desktop aplikacija, sigurno ćete se susretati sa datotekama, i potrebno je znati imati pravilnu interakciju sa datotekama.

Python, naravno, ima ugrađene funkcije za ulaz i izlaz. Ugrađenu funkciju open najčešće ćete koristiti za otvaranje datoteka, naravno uz odgovarajuće parametre.

Nije potrebno samo da otvorite datoteku, već da u nju nešto i upišete. Na primer, za upisivanje tekstualnih sadržaja koristite funkciju write. Konačno, posle završetka rada sa datotekom, zatvorite je funkcijom close.

Pre ili kasnije, desiće se da imate greške u kodu. Kao budući programeri i inženjeri računarske tehnike, veoma je bitno da znate kako te greške obratiti. U nastavku lekcije govoriće se o obradi izuzetaka i grešaka.

Proces debagiranja ili debagovanja, kako već želite da izgovarate, je sličan kao i u ostalim programskim jezicima, ali ovde ćemo spomenuti specifičnosti Python jezika koji ima try, except i finally komande za obradu izuzetaka.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

# ✓ Poglavlje 1

## Čitanje iz datoteka

### UVOD U RAD SA DATOTEKAMA

*Pisanje programa koji koriste datoteke za čitanje i pisanje podataka rešava problem korišćenjem tastature kao jedinog ulaza, i konzole kao jedinog izlaza.*

Programi koji su bili pisani do sada čitali su sav ulaz sa tastature korisnika. Kao rezultat, ovaj pristup je zahtevao da se svi ulazni parametri ukucavaju svaki put kada se pokrene program. Ovaj pristup je neefikasan, naročito za programe koji zahtevaju unos velike količine podataka.

Na sličan način, programi koji su do sada pisani prikazivali su rezultate isključivu na ekranu (tačnije na konzoli ekrana). Ovaj pristup jeste adekvatan kada se štampa izlaz koji sadrži samo nekoliko linija teksta, ali je nepraktičan za veće rezultate, jer se prebrzo ispisuju na ekran. Još bitnije ovaj pristup je neefikasan kada je potrebno da izlaz bude bio naknadno analiziran od strane drugih programa.

Pisanje programa koji koriste datoteke za čitanje i pisanje podataka rešava problem korišćenje tastature kao jedinog ulaza, i konzole kao jedinog izlaza.

**Datoteka** (en. **file**) predstavlja skup podataka sadržanu u jednoj celini, identifikovana po imenu i tipu (ekstenziji, en. **extension**). Datoteka može biti tekstualni dokument, slika, audio ili video podatak, biblioteka podataka, izvršni program ili bilo koji skup podataka.

Datoteke se mogu otvoriti, sačuvati, obrisati i prenesti u različite direktorijume unutar sistema datoteka.

Direktorijum (en. **directory**, **folder**) predstavlja kataloški sistem koji sadrži reference na datoteke.

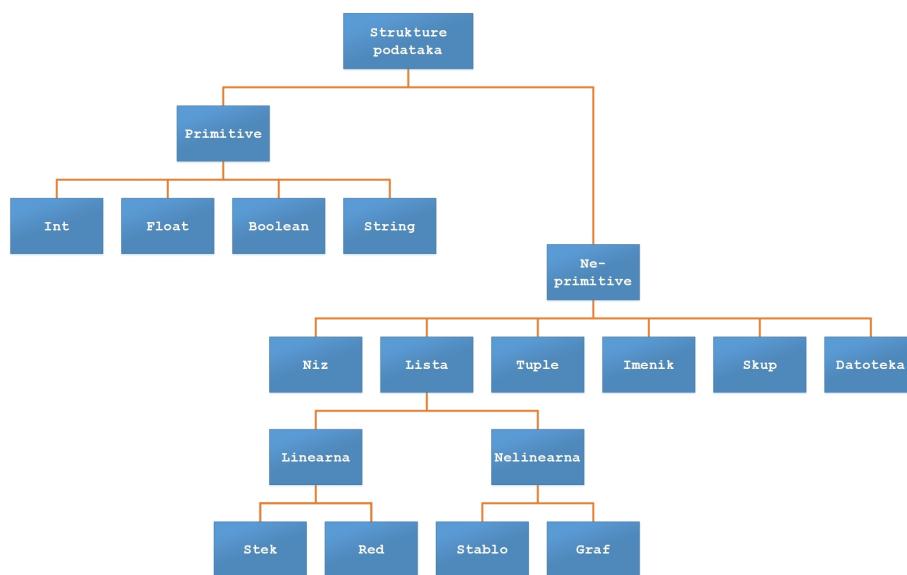
**Sistem datoteka** (en. **file system**) kontroliše kako se podaci smeštaju i pristupaju unutar računarskog sistema.

Bez sistema datoteka, podaci koji se nalaze u medijumu za smeštanje podataka bili bi jedan veliki skup podataka bez početka i kraja svake datoteke. Sistem datoteka služi da svaku datoteku zasebno smesti i imenuje, da bi se podaci mogli lako identifikovati i da bi im se moglo lako pristupiti.

# RAD SA DATOTEKAMA U PYTHON PROGRAMSKOM JEZIKU

*Datoteke se najčešće dele na tekstualne datoteke i binarne datoteke.  
Većina principa važi za oba tipa datoteka.*

Ako se pogleda struktura podataka koji Python 3.x jezik podržava, datoteka spada u ne-primitivu.



Slika 1.1.1 Struktura podataka koji Python 3.x jezik podržava. [Izvor: Autor]

Datoteke su relativno stalne po sadržaju. Vrednosti koje se čuvaju u datotekama se zadržavaju nakon što se program izvrši i nakon prestanka napajanja računara električnom energijom. Zbog ove osobine, datoteke jesu jako pogodne za smeštanje rezultata koji su potrebni za duži vremenski period, kao i za smeštanje ulaznih podataka za program koji treba izvršiti više puta.

Datoteke se najčešće dele na tekstualne datoteke (en. text files) i binarne datoteke (en. binary files).

Tekstualne datoteke sadrže samo sekvence bitova koji predstavljaju alfanumeričke karaktere, i koriste sistem kodovanja ASCII ili UTF-8. Ove datoteke se mogu otvoriti i modifikovati sa bilo kojim editorom teksta.

Dosadašnji Python programi su bile tekstualne datoteke.

Kao i tekstualne datoteke, binarne datoteke sadrže sekvence bitova. Međutim, za razliku od tekstualnih datoteka, ove sekvence mogu predstavljati bilo koji tip podataka. Tačnije, nisu ograničene na alfanumeričke karaktere. Većina principa koji važe za tekstualne datoteke važiće i za binarne datoteke.

## 1.1 Funkcija open()

### OSNOVE FUNKCIJE OPEN()

*Najjednostavniji načina otvaranje datoteke u Python 3.x jeziku jeste korišćenje funkcije `open()`, koja prima 3 argumenta (od kojih su dva opciona).*

Najjednostavniji načina otvaranje datoteke u Python 3.x jeziku jeste korišćenje funkcije [open\(\)](#).

Najpre treba imati datoteku sa podacima. U nastavku je dat sadržaj u deset linija koda koji treba prekopirati i smestiti u **test\_podaci.txt** datoteku i smestiti je u radni direktorijum Python programa.

Ukoliko se radi sa datotekama koje nisu u radnom direktorijumu, potrebno je naći relativnu ili absolutnu putanju do datoteke.

```
1] Ovo je datoteka sa test podacima
2] Takodje ima u sebi broj linija
3] Ovo je treca linija datoteke
4] Ovo je cetvrta linija datoteke
5] Ovo je, tako je, peta linija
6] Sesta linija
7] Sedma linija
8] Osma linija
9] Deveta i predzadnja linija
10] Zaustavicemo se kod deseta linije
```

#### **Pitanje:**

**Šta je relativna, a šta je absolutna putanja do datoteke?**

### Funkcija open()

Sintaksa funkcije open je sledeća:

```
file object = open(file_name [, access_mode][, buffering])
```

Prvi argument jeste string koji je ime datoteke (sa putanjom ukoliko je potrebno)

Funkcija open drugim argumentom specificira da li se datoteka koja se otvara koristi za pregled, čitanje, pisanje, dodavanje, ili čitanje i pisanje. Ukoliko se drugi argument ne specificira, podrazumevano se datoteka otvara za čitanje sadržaja.

Treći argument služi za opciju baferovanja, tj. da li se podaci čitaju preko operativnog sistema ili bafera.

[https://www.tutorialspoint.com/python/python\\_files\\_io.htm](https://www.tutorialspoint.com/python/python_files_io.htm)

```
r - Čitanje datoteke.
rb - Čitanje datoteke u binarnom formatu.
r+ - Čitanje i pisanje u datoteku.
rb+ - Čitanje i pisanje u datoteku u binarnom fomatu.
w - Samo pisanje u datoteku
wb - Samo pisanje u binarnom formatu.
w+ - Čitanje i pisanje u datoteku. Preimenuje staru datotku ili kreira novu sa tim imenom.
wb+ - Čitanje i pisanje u datoteku u binarnom formatu. Preimenuje staru datotku ili kreira novu sa tim imenom.
a - Otvara datoteku za dodavanje sadržaja.
ab - Otvara datoteku za dodavanje sadržaja u binarnom formatu.
a+ - Otvara datoteku za dodavanje sadržaja i za čitanje.
ab+ - Otvara datoteku za dodavanje sadržaja i za čitanje, u binarnom formatu.
```

## OTVARANJE DATOTEKE ZA ČITANJE FUNKCIJOM OPEN()

*Datoteka se može otvoriti za čitanje pozivom `ime_promenljive = open("ime_datoteke", "r")`*

### Otvaranje datoteke

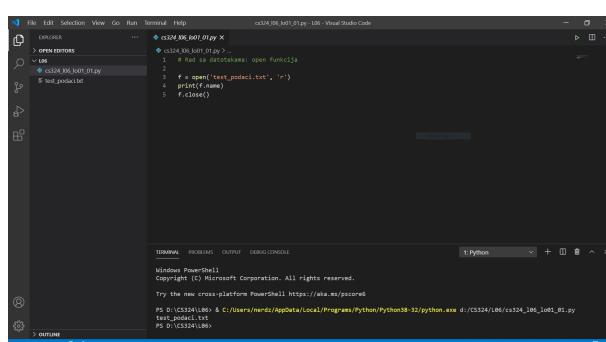
Otvaranje datoteke `test_podaci.txt` se može izvršiti sledećom komandom:

```
f = open('test_podaci.txt', 'r')
```

### Zatvaranje datoteke

Nakon završetka rada sa datotekom, potrebno je da eksplisitno zatvorimo datoteku da ne bi ostala u memoriji. Zatvaranje datoteke se može izvršiti pozivom metode `.close()`

```
f = open('test_podaci.txt', 'r')
.
.
.
f.close()
```



Slika 1.2.1 Rad sa datotekama - `open()` funkcije. [Izvor: Autor]

### Metode za rad za datotekama

**Ime datoteke** se može videti pozivom metode **.name()**

```
f = open('test_podaci.txt', 'r')
print(f.name)
f.close()
```

**Režim otvaranja** datoteke se može videti pozivom metode **.mode()**

Ukoliko nije sigurno kako je datoteka otvorena, uvek se može proveriti da li je otvorena za čitanje, pisanje ili dodavanje pozivom ove metode.

```
f = open('test_podaci.txt', 'r')
print(f.mode)
f.close()
```

Provera zatvaranja datoteke se može videti pozivom atributa **.closed**

```
f = open('test_podaci.txt', 'r')
print(f.mode)
print(f.name)
print(f.closed)

f.close()

print(f.closed)
```

## PRIMER RADA SA OTVARANJEM DATOTEKE KORIŠĆENJEM OPEN() FUNKCIJE

*Sledi video za otvaranje datoteke pomoću funkcije open()*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

- ✓ 1.2 Kontekstni menadžer prilikom otvaranja datoteka

## OTVARANJE DATOTEKE ZA ČITANJE KONTEKSTNIM MENADŽEROM

*Kontekstni menadžer dozvoljava alociranje i oslobođanje resursa u tačno preciziranim trenucima, i automatizuje kontrolu resursa.*

Do sada je bilo reči o eksplisitnom otvaranju i zatvaranju datoteka prilikom rada sa datotekama. Ukoliko se datoteka pravilno ne zatvori, doći će do tzv. curenja memorije (en. **memory leakage**), što može dovesti do usporenje rada ili do potpunog prestanka rada računara.

**Kontekstni menadžer** dozvoljava alociranje i oslobođanje resursa u tačno preciziranim trenucima, i automatizuje kontrolu resursa. Upotreba kontekstnog menadžera u Python jeziku postiže se ključnom rečju **with**, a sintaksa prilikom otvaranja datoteke na ovaj način je sledeća:

```
with open('test_podaci.txt', 'r') as f:  
    ...
```

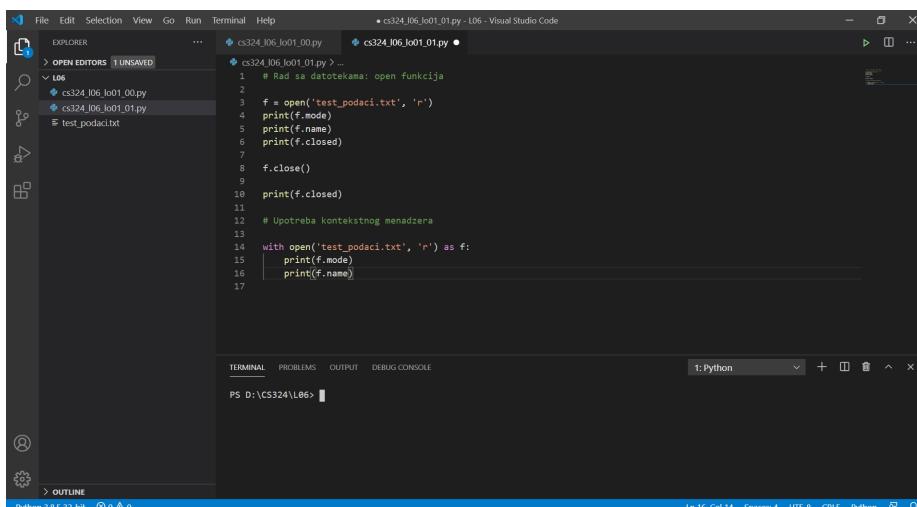
Treba obratiti pažnju da je ime promenljive dato na kraju izjave, a ne na početku kao kod korišćenja open() funkcije.

Prednost korišćenja kontekstnog menadžera jeste da omogućavaju rad sa datotekama unutar bloka koda, a kada se izade iz bloka koda, datoteka se automatski zatvara.

**Korišćenje kontekstnih menadžera za rad sa datotekama predstavlja dobru praksu pisanja koda u Python jeziku.**

Otvaranje prethodne datoteke korišćenjem kontekstnog menadžera ima sledeću sintaksu:

```
# Upotreba kontekstnog menadzera  
  
with open('test_podaci.txt', 'r') as f:  
    print(f.mode)  
    print(f.name)
```



Slika 1.3.1 Opotreba kontekstnog menadžera za otvaranje datoteka. [Izvor: Autor]

## ČITANJA DATOTEKE DOK JE OTVORENA, I NAKON ZATVARANJA

*Čitanje datoteke je moguće samo kada je datoteka otvorena.*

Ime promenljive postoji i nakon zatvaranja datoteke, i biće tipa `IO.TextWrapper`. `io.TextWrapper` je izvedena klasa iz `TextIOBase`, koja je osnovna klasa za rad sa tekstualnim tokovima (en. **streams**) podataka.

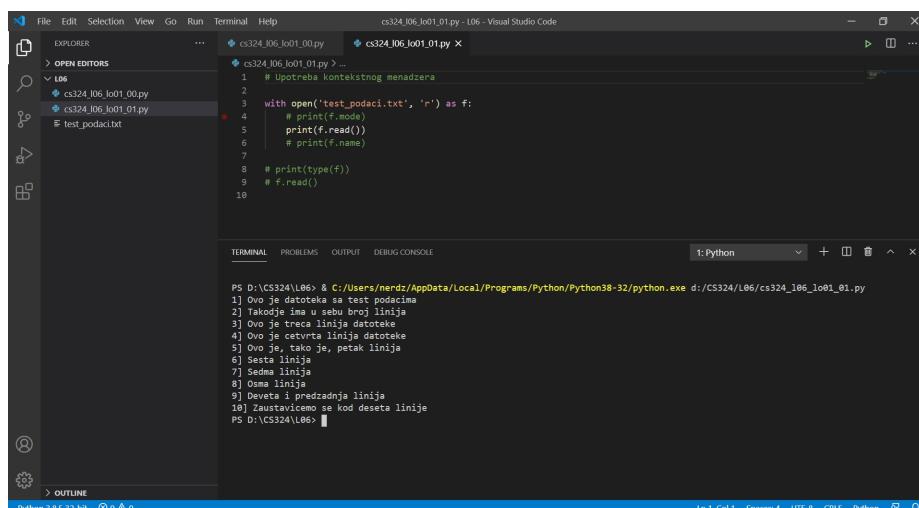
```
>>>print(type(f))

console output:
<class '_io.TextIOWrapper'>
```

Iako ime promenljive i dalje postoji, nije moguće pročitati sadržaj datoteke.

Čitanje datoteke je moguće pozivom metode `.read()`

```
with open('test_podaci.txt', 'r') as f:
    print(f.read())
```



Slika 1.3.2 Čitanje podataka iz datoteke. [Izvor: Autor]

Čitanje datoteke je moguće samo kada je datoteka otvorena, tj. dok je funkcija za čitanje unutar bloka koda kontekstnog menadžera.

Ukoliko se pokuša čitanje van bloka koda, interpreter će vratiti grešku.

```
# van kontekstnog menadzera
>>>f.read()

ValueError: I/O operation on closed file.
```

## PRIMER RADA SA OTVARANJEM DATOTEKE KORIŠĆENJEM KONTEKSTNOG MENADŽERA

*Sledi video za otvaranje datoteke pomoću kontekstnog menadžera*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### ✓ 1.3 Metoda za čitanje podataka

#### METODE .READ(), .READLINES() I .READLINE()

*Najjednostavniji način čitanja podataka iz datoteke jeste upravo `.read()` metoda, ali nije jedina koja se koristi, pogotovo ukoliko se radi o datoteci velikog kapaciteta.*

Najjednostavniji način čitanja podataka iz datoteke jeste upravo `.read()` metoda koja je objašnjena u prethodnoj sekciji.

Umesto samog štampanja sadržaja, sadržaj se može smestiti u novu promenljivu:

```
with open('test_podaci.txt', 'r') as f:  
    f_sadrzaj = f.read()  
    print(f_sadrzaj)  
    print(type(f_sadrzaj))
```

Poslednji print vratiće da je `f_sadrzaj` tipa string.

Ukoliko se želi napraviti promenljiva koja bi čitala sadržaj liniju po liniju, onda je bolje da se koristiti `.readlines()` metoda.

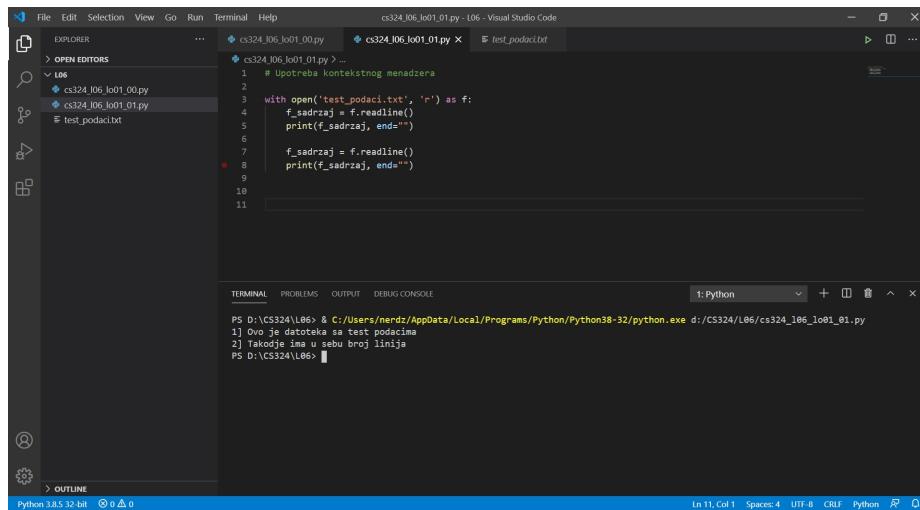
```
with open('test_podaci.txt', 'r') as f:  
    f_sadrzaj = f.readlines()  
    print(f_sadrzaj)  
    print(type(f_sadrzaj))
```

U ovom slučaju, `f_sadrzaj` biće lista, sa onoliko elemenata koliko ima linija u datoteci.

Ukoliko se želi pročitati linija po linija, koristi se `.readline()` metoda, koja pri svakom pozivu čita sledeću liniju unutar datoteke.

```
with open('test_podaci.txt', 'r') as f:  
    f_sadrzaj = f.readline()  
    print(f_sadrzaj, end="")
```

```
f_sadrzaj = f.readline()
print(f_sadrzaj, end="")
```



```
File Edit Selection View Go Run Terminal Help cs324_106_lo01_01.py - L06 - Visual Studio Code
EXPLORER ... cs324_106_lo01_01.py x test_podaci.txt
L06
cs324_106_lo01_00.py
cs324_106_lo01_01.py
test_podaci.txt
1 # Upotreba kontekstnog menadzera
2
3 with open('test_podaci.txt', 'r') as f:
4     f_sadrzaj = f.readline()
5     print(f_sadrzaj, end="")
6
7     f_sadrzaj = f.readline()
8     print(f_sadrzaj, end="")
9
10
11

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
1: Python
PS D:\CS324\L06> & C:/Users/nenadz/AppData/Local/Programs/Python/Python38-32/python.exe d:/CS324/L06/cs324_106_lo01_01.py
1] Ovo je datoteka sa test podacima
2] Takođe ima u sebi broj linija
PS D:\CS324\L06>

Python 3.8.5 32-bit 0 0 Δ 0
```

Slika 1.4.1 Čitanje sadržaja datoteke liniju po liniju. [Izvor: Autor]

## PRIMER .READLINE() NASPRAM .READLINES()

*Metode .readline() i .readlines() se mogu koristiti u zavisnosti da li je potrebno čitati sve ili samo neke linije iz datoteke.*

Ukoliko je datoteka velika, nije efikasno koristiti .readline() mnoštvo puta dok se svi redovi ne iščitaju.

Efikasnije je iterativno čitati red po red:

```
with open('test_podaci.txt', 'r') as f:
    for line in f:
        print(line, end='')
```

Ukoliko postoji prazna lista koju treba populisati liniju po liniju iz datoteke, može se koristiti sličan pristup:

```
sadrzaj = []

with open('test_podaci.txt', 'r') as f:
    for line in f:
        sadrzaj.append(line)

print(sadrzaj)
```

Pitanje:

**Kako se može isto (smeštanje sadržaja svih linija iz datoteke u listu) postići bez iteracije?**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## DODATNI ARGUMENT ZA .READ() METODU

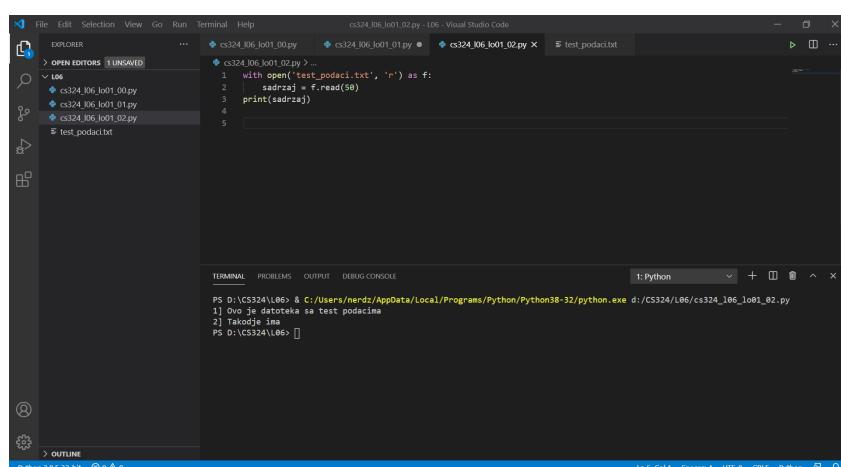
*Moguće je dodati argument u .read() metodi, i onda se čita samo toliko karaktera.*

Podrazumevano, metoda **.read()** pročitaće ceo sadržaj datoteke.

Dodavanjem argumenta u metodu, moguće je podesiti broj karaktera koji se čita pozivom ove metode:

```
with open('test_podaci.txt', 'r') as f:  
    sadrzaj = f.read(50)  
print(sadrzaj)
```

U primeru iznad broj karaktera jeste 50, a izlaz biće kao na slici ispod.

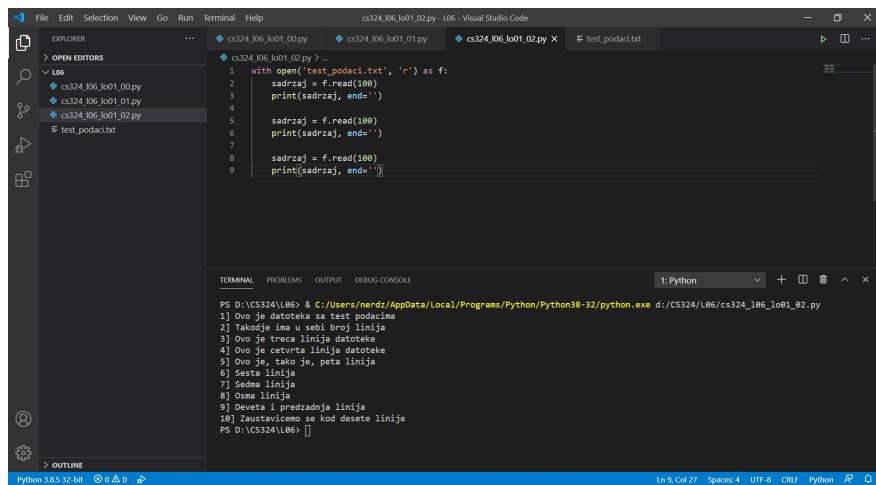


Slika 1.4.2 Izbor broja karaktera koji se čita. [Izvor: Autor]

Ukoliko se više puta pozove **.read()** metoda sa ovim argumentom, svaki sledeći put nastaviće sa čitanjem.

```
with open('test_podaci.txt', 'r') as f:  
    sadrzaj = f.read(100)  
    print(sadrzaj)  
  
    sadrzaj = f.read(100)  
    print(sadrzaj)  
  
    sadrzaj = f.read(100)  
    print(sadrzaj)
```

Kada **.read()** nađe na kraj datoteke, vratiće prazan string.



Slika 1.4.3 Ponavljanje metode `.read()` sa argumentom, dok se ne pročita cela datoteka. [Izvor: Autor]

## ČITANJE IZ VELIKIH TEKSTUALNIH DATOTEKA

*Za veću kontrolu nad čitanjem podataka iz datoteke, bolje je koristiti parametar koji kontroliše čitanje svake iteracije.*

U opštem slučaju, veličina datoteke ili broj redova datoteke neće biti poznati. Zbog toga je potrebno napraviti petlju koja će učitavati deo po deo datoteke, a koliko će se učitati u jednoj iteraciji kontroliše se parametrom, a ne direktnim ubacivanjem vrednosti.

```
with open('test_podaci.txt', 'r') as f:
    broj_karaktera = 20
    sadrzaj = f.read(broj_karaktera)

    while len(sadrzaj) > 0:
        print(sadrzaj, end=' ')
        sadrzaj = f.read(broj_karaktera)
```

**Pitanje:**

**Šta bi se desilo ukoliko nema poslednjeg reda unutar while petlje?**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 2

### Pisanje u datoteku

#### OSNOVE FUNKCIJE WRITE()

*Za razliku od čitanja iz datoteke, nije neophodno imati unapred napravljenu datoteku. Ukoliko datoteka ne postoji, ova metoda će napraviti.*

Metoda .write() predstavlja najjednostavniji način upis podataka u datoteku.

Za razliku od čitanja iz datoteke, nije neophodno imati unapred napravljenu datoteku. Ukoliko datoteka ne postoji, ova metoda će napraviti.

Sintaksa je sledeća:

```
fileObject.write( str )
```

Naravno, najpre treba otvoriti datoteku za čitanje, i to se može uraditi na više načina, kao i kod otvaranja datoteke za čitanje.

#### Otvaranje preko open() funkcije

Datoteka se može otvoriti za pisanje preko open() funkcije, ali da je drugi argument string 'w'. Nakon toga, može se pisati u datoteku pozivom metode .write()

```
f = open('test_podaci.txt', 'w')
f.write('Ovo su novi test podaci.')
f.close()
```

#### Primer - Otvaranje datoteke za čitanje i pokušaj pisanja

Ukoliko se pokuša sledeći kod:

```
with open('test_podaci.txt', 'r') as f:
    f.write('Ovo su novi test podaci.')
```

Konzola vraća grešku da se u datoteci ne mogu upisati podaci

```

File Edit Selection View Go Run Terminal Help
OPEN EDITORS primer_za_write.py novi_test_podaci.txt
primer_za_write.py ...
1 f = open('test_podaci.txt', 'r')
2 f.write('Ovo su novi test podaci. \n')
3 f.write('Ovo je drugi red test podataka. \n')
4 f.close()
5
6

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
PS D:\CS324\106> & D:\anaconda3\python.exe d:/CS324/106/primer_za_write.py
Traceback (most recent call last):
File "d:/CS324/106/primer_za_write.py", line 2, in module
f.write('Ovo su novi test podaci.')
io.UnsupportedOperation: not writable
PS D:\CS324\106>

```

Slika 2.1 Pokušaj pisanja u datoteku dok je otvorena za čitanje. [Izvor: Autor]

## PISANJE U DATOTEKU KORIŠĆENJEM KONTEKSTNOG MENADŽERA

*Efikasnije je otvoriti datoteku korišćenjem kontekstnog menadžera i u slučaju pisanja u datoteku.*

### Otvaranje preko kontekstnog menadžera

Kao i u ranijim primerima, efikasnije je otvoriti datoteku kroz kontekstni menadžer, jer onda ne treba voditi računa da se datoteka eksplisitno zatvori.

```

# Upotreba kontekstnog menadzera

with open('novi_test_podaci.txt', 'w') as f:
    f.write('Ovo su novi test podaci. \n')
    f.write('Ovo je drugi red test podataka. \n')

```

```

File Edit Selection View Go Run Terminal Help
OPEN EDITORS primer_za_write.py novi_test_podaci.txt
primer_za_write.py ...
1 # Upotreba kontekstnog menadzera
2
3 with open('novi_test_podaci.txt', 'w') as f:
4     f.write('Ovo su novi test podaci. \n')
5     f.write('Ovo je drugi red test podataka. \n')

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
PS D:\CS324\106> & D:\anaconda3\python.exe d:/CS324/106/primer_za_write.py
1 Ovo su novi test podaci.
2 Ovo je drugi red test podataka.
3

PS D:\CS324\106>

```

Slika 2.2 Korišćenje kontekstnog menadžera za pisanje u datoteku. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## METODE POZICIONIRANJA: .SEEK() I .TELL()

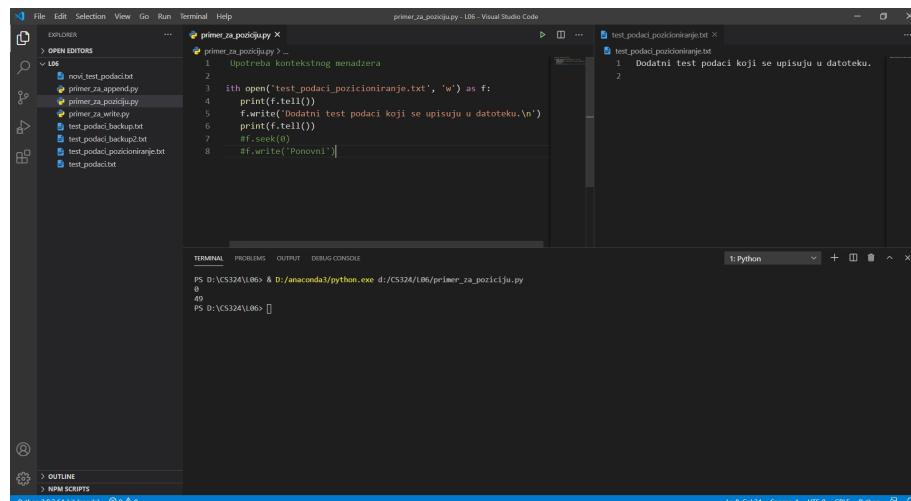
*Korišćenjem metoda .tell() i .seek() moguće je kontrolisati poziciju toka podataka za upis u datoteku.*

### Metoda .tell()

Metoda .tell() vraća poziciju gde je stao tok podataka koji se upisuje.

```
# Kontekstni menadzer

with open('test_podaci_pozicioniranje.txt', 'w') as f:
    print(f.tell())
    f.write('Dodatni test podaci koji se upisuju u datoteku.\n')
```



Slika 2.3 Poziv metode .tell(), koja vraća trenutnu poziciju toka podatka za pisanje. [Izvor: Autor]

### Metoda .seek()

Metoda .seek() podešava trenutnu poziciju toka podataka koji se upisuje.

```
# Kontekstni menadzer

with open('test_podaci_pozicioniranje.txt', 'w') as f:
    print(f.tell())
    f.write('Dodatni test podaci koji se upisuju u datoteku.\n')
    print(f.tell())
    f.seek(0)
    f.write('Ponovni')
```

```

File Edit Selection View Go Run Terminal Help
OPEN EDITORS
primer_za_posiziju.py
1 # Upotreba kontekstnog menadzera
2
3 with open('test_podaci_posicioniranje.txt', 'w') as f:
4     print(f.tell())
5     f.write('Dodatajni test podaci koji se upisuju u datoteku.\n')
6     print(f.tell())
7     f.seek(0)
8     f.write(['Ponovni'])

test_podaci_posicioniranje.txt
1 Ponovni test podaci koji se upisuju u datoteku.

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
PS D:\CS324\U06 & D:/anaconda3/python.exe d:/CS324/U06/primer_za_posiziju.py
0
49
PS D:\CS324\U06 & D:/anaconda3/python.exe d:/CS324/U06/primer_za_posiziju.py
0
49
PS D:\CS324\U06 > []

```

Slika 2.4 Poziv metode .seek() podešava se pozicija toka podataka za pisanje. [Izvor: Autor]

## DODAVANJE PODATAKA U DATOTEKU

*Upotrebom argumenta 'a' prilikom poziva funkcije open(), sadržaj koji upisujemo se dodaje, a ne presnimava u datoteku.*

Može se primetiti da svaki put kada se otvori datoteka za pisanje, datoteka presnimava sadržaj, i stari sadržaj datoteke se gubi. Da se to ne bi dešavalо, prilikom poziva open() funkcije, koristi se karakter 'a' kao drugi argument.

Upotreba ovog argumenta dozvoljava da sav upis ide nakon već postojećeg sadržaja.

```

# Upotreba kontekstnog menadzera

with open('test_podaci.txt', 'a') as f:
    f.write('11] Ipak se dodaje i jedanaesti red. \n')
    f.write('12] A moze i dvanaesti.')

```

Na ovaj način moguće je zadržati sadržaj datoteke, i dodati novi sadržaj.

```

File Edit Selection View Go Run Terminal Help
OPEN EDITORS
primer_za_append.py
1 # Upotreba kontekstnog menadzera
2
3 with open('test_podaci.txt', 'a') as f:
4     f.write('11] Ipak se dodaje i jedanaesti red. \n')
5     f.write('12] A moze i dvanaesti.')

test_podaci.txt
1 Ovo je datoteka sa test podacima
2 Takođe ima u sebi broj linija
3 Ovo je treća linija datoteke
4 Ovo je četvrt linija datoteke
5 Ovo je petnaesti red. Je, petnaesti
6 Šesta linija
7 Sedma linija
8 Osmi linija
9 Deveta i predzadnja linija
10 Zauštavimo se kod deseta linije

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
PS D:\CS324\U06 & D:/anaconda3/python.exe d:/CS324/U06/primer_za_append.py
PS D:\CS324\U06 > []

```

Slika 2.5 Sadržaj datoteke test\_podaci.txt preizvršenja programa. [Izvor: Autor]

The screenshot shows the Visual Studio Code interface. On the left, the 'OPEN EDITORS' sidebar shows 'primer\_na\_append.py' and 'test\_podaci.txt'. The 'primer\_na\_append.py' editor contains Python code for appending to a file. The 'test\_podaci.txt' editor contains a text file with 12 lines of text. The terminal at the bottom shows the execution of the script and its output, which is the content of the 'test\_podaci.txt' file.

```

primer_na_append.py
1] Ovo je datoteka sa test podacima
2] Takođe imu u sebi broj linija
3] Ivo je petnaesta linija datoteka
4] Ivo je petnaesta linija datoteka
5] Ovo je, tako je, petnaesta linija
6] Sesta linija
7] Sedma linija
8] Deveta linija
9] Deveta i predstavlja linija
10] Zastavljeno se kod deseta linije
11] Ipak se dodaje i jedanaesti red.
12] A može i dvanaesti.

TERMINAL: PROBLEMS: OUTPUT: DEBUG CONSOLE: i:Python
PS D:\VCS2A\1.0\6 & D:/anaconda3/python.exe d:/CS2A/1.06/primer_na_append.py
PS D:\VCS2A\1.0\6 & D:/anaconda3/python.exe d:/CS2A/1.06/primer_na_append.py
PS D:\VCS2A\1.0\6 & D:/anaconda3/python.exe d:/CS2A/1.06/primer_na_append.py
PS D:\VCS2A\1.0\6

```

Slika 2.6 Sadržaj datoteke test\_podaci.txt nakon izvršenja programa. [Izvor: Autor]

## PRIMER: DODAVANJE SADRŽAJA U DATOTEKU SA 'R+' ARGUMENTOM

*Primer koji sledi opisuje mogućnost dodavanja sadržaja u datoteku (bez brisanja prethodnog) iako se kao argument za otvaranje koristi 'r+'.*

Moguće je (ali je manje efikasno od append režima) koristiti read+write režim za dodavanje sadržaja na kraju datoteke.

Razmotriti sledeći kod:

```

# Upotreba kontekstnog menadzera

with open('test_podaci_primer_w_a.txt', 'r+') as f:
    print(f.tell())
    sadrzaj = f.read()
    broj_karaktera = len(sadrzaj)
    # print(sadrzaj)
    print(broj_karaktera)

    pisanje u datoteku
    f.seek(broj_karaktera)
    print(f.tell())
    f.write('\n11] Ipak imamo i jedanaestu liniju')

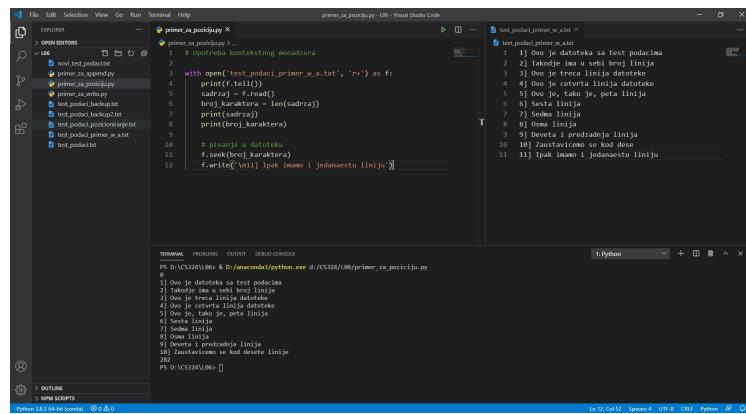
```

Nakon čitanja sadržaja u liniji 5, uzima se **broj\_karaktera**.

Onda se poziva **.seek()** metoda koja pozicionira tok podataka na **broj\_karaktera**. Konačno, piše se u datoteku od pozicioniranja.

**Zanimljivost:**

**Šta vraća napisani program? Proveriti sadržaj datoteke.**



```
primer_sa_posicijom.py
1 # Upotreba kontekstnog menadzera
2
3 with open('test_podaci_i_primer_w_n.txt', 'r+') as f:
4     print(f.tell())
5     sadrzaj = f.read()
6     print(sadrzaj)
7     print(len(sadrzaj))
8     print(broj_karaktera)
9
10    # pisanje u datoteku
11    f.seek(broj_karaktera)
12    f.write('\n') i pak inamo i jedanaestu liniju!
```

```
test_posledi.primer_w.txt
1 1] Ovo je datoteka sa test podacima
2 2] Takođe ima i neku drugu
3 3] liniju u datoteci
4 4] Ovo je četvrta linija datoteke
5 5] Ovo je, takođe, peta linija
6 6] u datoteci
7 7] Sedma linija
8 8] Osma linija
9 9] Deveta i predzadnja linija
10 10] Zastavljenu se kod doček
11 11] spisak imena i jedanaestu liniju!
```

TERMINAL: PROBLEMS: OUTPUT: DOKUMENCI

```
PS D:\CS320\106 & D:\me\code\python.exe d:\CS320\106\primer_sa_posicijom.py
```

```
1] Ovo je datoteka sa test podacima
2] Takođe ima i neku drugu
3] liniju u datoteci
4] Ovo je četvrta linija datoteke
5] Ovo je, takođe, peta linija
6] u datoteci
7] Sedma linija
8] Osma linija
9] Deveta i predzadnja linija
10] Zastavljenu se kod doček
11] spisak imena i jedanaestu liniju!
```

PS D:\CS320\106

Slika 2.7 Primer dodavanja sadržaja u datoteku sa `r+` argumentom. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 3

### Čuvanje podataka

#### ČUVANJE PODATAKA U CSV DATOTEKU

*CSV datoteka jeste tabelarno uređena datoteka koja se često koristi za razmenu podataka između aplikacija.*

Ukoliko je potrebno sačuvati veliku količinu podataka, obično su ti podaci nekako uređeni.

Iako sadrži tekstualne podatke, CSV datoteka (en. comma separated values file) jeste tabelarno uređena datoteka koja se koristi za razmenu podataka između aplikacija.

##### **Struktura CSV datoteke**

CSV datoteka ima relativno jednostavnu strukturu. Predstavlja listu podataka koje deli zapeta. Prednost CSV datoteke jeste u tome što je čitljiva i od strane korisnika (čoveka) ali i da može da se otvori u bilo kom editoru teksta.

##### **Primer: Telefonski imenik**

Name,Email,Phone Number,Address

Bob Smith,bob@example.com,123-456-7890,123 Fake Street

Mike Jones,mike@example.com,098-765-4321,321 Fake Avenue

Pojedine CSV datoteke ne moraju imati ni liniju za zaglavlje na vrhu.

##### **Rad sa CSV datotekama u Python jeziku**

Python ima ugrađene module za rad sa CSV datotekama. Potrebno je pre korišćenja ukucati:

```
import csv
```

Nakon toga, mogu se koristiti metode za pisanje i čitanje CSV datoteka.

U nastavku je tekst u CSV formatu koji se može koristiti za primere čitanja.

```
CS220,CS225,CS324,IT331,IT335,IT376,SE321,SE325
```

```
2,3,4,2,3,3,4,4
```

```
2020,2019,2018,2017,2020,2018,2017
```

```
Jun A,Jun A,Januar B,Jun B,Septembar,Januar A,Oktobar II,Novembar
```

9,8,10,10,8,7,6,9

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PISANJE U CSV DATOTEKU

*Otvaranje CSV datoteke isto je kao i otvaranje bilo koje datoteke, a za pisanje kreira se novi objekat iz klase csv.writer*

Otvaranje CSV datoteke isto je kao i otvaranje bilo koje datoteke. Moguće je otvoriti direktno (ali onda treba voditi računa da se zatvori), ili preko kontekstnog menadžera.

### Primer: Broj radnih sata u nedelji

Napisati CSV datoteku koja će sadržati u jednom redu radne dane (ponedeljak - petak), a u drugom broj radnih sata za svaki dan.

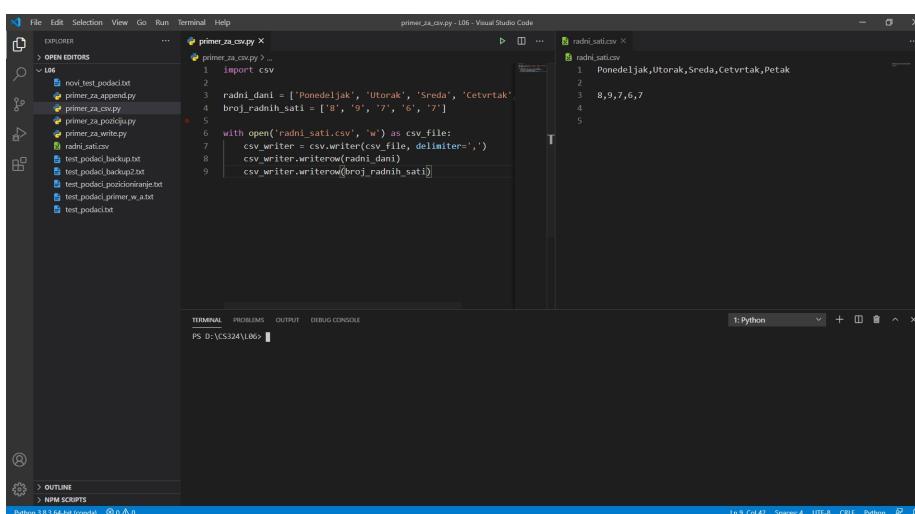
```
import csv

radni_dani = ['Ponedeljak', 'Utorak', 'Sreda', 'Cetvrtak', 'Petak']
broj_radnih_sati = ['8', '9', '7', '6', '7']

with open('radni_sati.csv', 'w') as csv_file:
    csv_writer = csv.writer(csv_file, delimiter=',')
    csv_writer.writerow(radni_dani)
    csv_writer.writerow(broj_radnih_sati)
```

Najpre treba uvesti csv biblioteku zbog dodatnih funkcija.

Kreiramo novi objekat iz klase **csv.writer** gde unosimo dva parametra, prvi je ime datoteke, a drugi jeste *separator* (en. **delimiter**), koji je u CSV datotekama najčešće *zapeta*. Zatim, koristimo metodu *.writerow()* za svaki red koji želimo da unesemo. Parametar ove metode jeste lista sa prethodno unetim elementima.



Slika 3.1.1 Sadržaj radni\_sati.csv datoteke nakon njene kreacije. Izvor: Autor.

## ČITANJE IZ CSV DATOTEKE

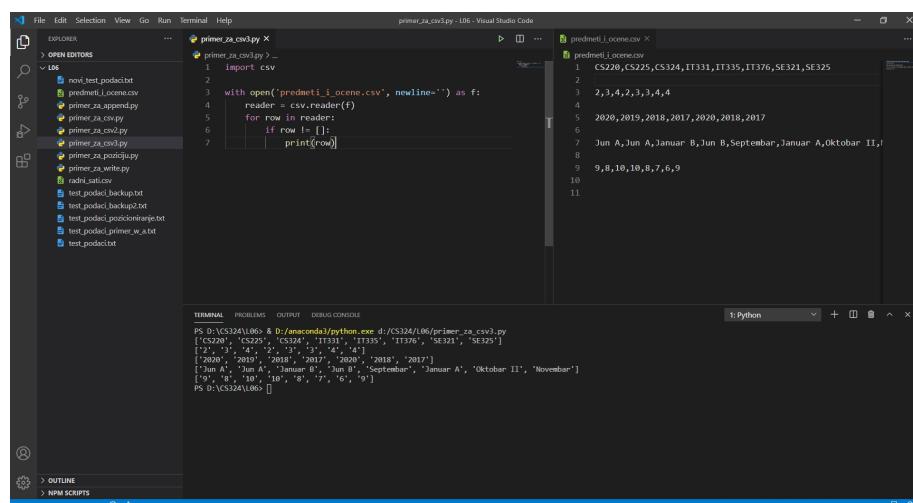
*Čitanje iz CSV datoteke je jednostavnije od pisanja u CSV datoteci, ali je svakako potrebno napraviti objekat klase csv.reader()*

### Čitanje iz CSV datoteke je takođe jednostavno.

Potrebno je kreirati objekat klase **csv.reader()** koji može pročitati sve redove CSV datoteke.

```
import csv

with open('predmeti_i_ocene.csv', newline='') as f:
    reader = csv.reader(f)
    for row in reader:
        if row != []:
            print(row)
```



Slika 3.1.2 Čitanje iz CSV datoteke. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ČUVANJE PODATAKA KAO JSON

*JSON podataka predstavlja otvoreni standard za smeštanje i razmenu podataka, koji koristi razumljivi tekst (en. human-readable text) za smeštanje i razmenu objekata podataka.*

**JSON podataka** (en. **JavaScript Object Notation**) predstavlja otvoreni standard za smeštanje i razmenu podataka, koji koristi razumljivi tekst (en. **human-readable text**) za smeštanje i razmenu objekata podataka.

Sastoji se od parova atribut-vrednost i tipova podataka koji imaju više elemenata.

Ovaj format se jako često koristi u raznim aplikacijama, i služi kao zamena za [XML](#) u [AJAX](#) sistemima.

JSON tip podataka je nezavistan od programske jezike iako je izведен iz JavaScript jezika. Zvanični tip podataka za JSON jeste application/json

<https://www.json.org/json-en.html>

### Rad sa JSON datotekama u Python jeziku

Python ima ugrađene module za rad sa JSON datotekama. Potrebno je pre korišćenja ukucati:

```
import json
```

Nakon toga, mogu se koristiti metode za pisanje i čitanje JSON datoteka.

U nastavku je tekst u JSON formatu koji se može koristiti za primere čitanja.

```
{"predmeti": [{"ime": "Skripting jezici", "sifra": "CS324", "godina studija": "IV", "smer": "RI"}, {"ime": "Operativni sistemi", "sifra": "CS220", "godina studija": "III", "smer": "IT"}, {"ime": "Arhitektura racunara", "sifra": "CS220", "godina studija": "II", "smer": "IT"}, {"ime": "Racunarske mreze i komunikacije", "sifra": "IT331", "godina studija": "II", "smer": "IT"}, {"ime": "Administracija racunarskih sistema i mreza", "sifra": "IT335", "godina studija": "III", "smer": "IT"}]}
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PISANJE U JSON DATOTEKU

*Prilikom pisanja podataka u JSON datoteku potrebno je kreirati objekat klase `json.dump()`, a kao parametre staviti promenljivu sa podacima koji se štampaju.*

Otvaranje JSON datoteke isto je kao i otvaranje bilo koje datoteke. Moguće je otvoriti direktno (ali onda treba voditi računa da se zatvori), ili preko kontekstnog menadžera.

### Primer: Imenik koji sadrži ime aplikacije, web adresu i državu odakle potiče

Napraviti prazan imenik podaci. Zatim dodati ključ "[apps](#)" i vrednost prazan string. Dodati kao elemente string nove imenike, sa parovima ključ: vrednost na sledeći način:

`'ime': 'ime aplikacije'`

`'website': 'web sajt aplikacije'`

`'drzava': 'drzava odakle potice aplikacija'`

Zatim, sačuvati imenik kao podaci.json

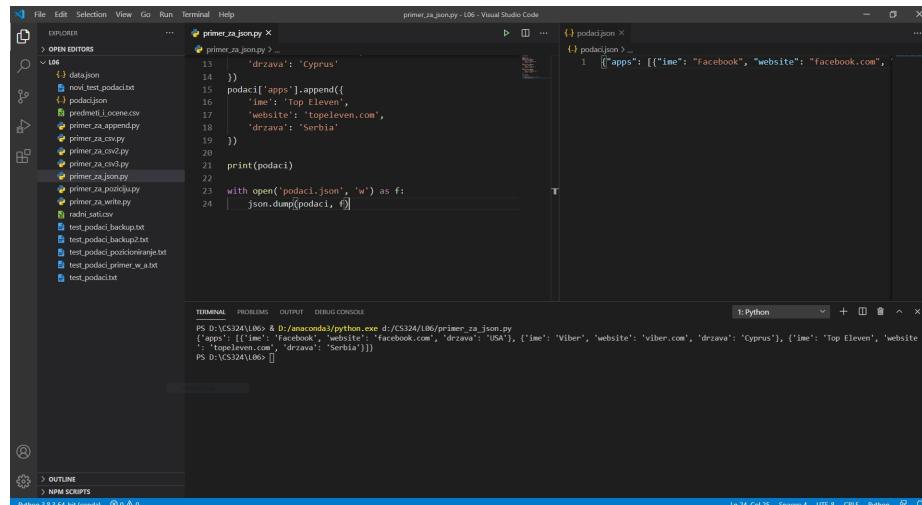
Prilikom otvaranja podataka potrebno je kreirati objekat klase `json.dump()` a kao parametre staviti koji se podaci štampaju i u koju otvorenu datoteku.

```
import json

podaci = []
podaci['apps'] = []
podaci['apps'].append({
    'ime': 'Facebook',
    'website': 'facebook.com',
    'drzava': 'USA'
})
podaci['apps'].append({
    'ime': 'Viber',
    'website': 'viber.com',
    'drzava': 'Cyprus'
})
podaci['apps'].append({
    'ime': 'Top Eleven',
    'website': 'topeleven.com',
    'drzava': 'Serbia'
})

print(podaci)

with open('podaci.json', 'w') as f:
    json.dump(podaci, f)
```



Slika 3.1.3 Primer pisanja podataka u JSON format. [Izvor: Autor]

## ČITANJE IZ JSON DATOTEKE

*Čitanje iz JSON datoteke je vrlo slično čitanju podataka iz bilo koje datoteke, i direktno će populisati imenik iz datoteke.*

Čitanje iz JSON datoteke je vrlo slično čitanju podataka iz bilo koje datoteke.

Ponovo je potrebno uvesti `json` biblioteku da bi se JSON string parsirao iz objekta datoteke.

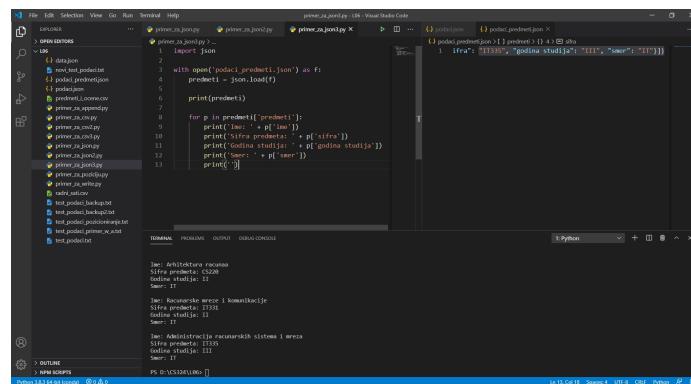
Za čitanje podataka treba kreirati objekat klase **`json.load()`** koji kao parametar uzima objekat JSON datoteke. Direktno će populisati imenik iz datoteke.

```
import json

with open('podaci_predmeti.json') as f:
    predmeti = json.load(f)

    print(predmeti)

    for p in predmeti['predmeti']:
        print('Ime: ' + p['ime'])
        print('Sifra predmeta: ' + p['sifra'])
        print('Godina studija: ' + p['godina studija'])
        print('Smer: ' + p['smer'])
        print('')
```



Slika 3.1.4 Čitanje iz CSV datoteke. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## 3.1 Rad sa binarnim datotekama

### PYTHON I BINARNE DATOTEKE

*Rad sa binarnim datotekama je sličan kao sa tekstualnim. Treba samo obratiti pažnju na drugi argument prilikom otvaranja datoteke.*

Rad sa binarnim datotekama je sličan kao sa tekstualnim.

Prilikom otvaranja datoteke, potrebno je u drugom argumentu dodati karakter 'b', što označava rad sa binarnim datotekama.

```
# otvaranje za citanje binarnih podataka
f_bin_read = open('Lenna.png', 'rb')

# otvaranje za pisanje binarnih podataka
f_bin_write = open('Lenna.png', 'wb')

#otvaranje kroz kontekstni menadzer za citanje
with open('Lenna.png', 'rb') as rfb:
    ...
    #otvaranje kroz kontekstni menadzer za pisanje
    with open('Lenna.png', 'wb') as wfb:
        ...
```

### Primer: Kopiranje binarne datoteke

Otvoriti sliku i kopirati je u drugu sliku.

Link do slike:

<https://en.wikipedia.org/wiki/Lenna>

```
# rad sa binarnim datotekama

with open('Lenna.png', 'rb') as rf:
    with open('Lenna_copy.png', 'wb') as wf:
        vel_bloka = 4096
        rf_blok = rf.read(vel_bloka)
        while len(rf_blok) > 0:
            wf.write(rf_blok)
            rf_blok = rf.read(vel_bloka)
```

## PRIMER RADA SA BINARNIM DATOTEKAMA

*Sledi video koji detaljno opisuje prethodni primer.*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 4

# Izuzeci i obrada izuzetaka

## ŠTA SU IZUZECI?

*Izuzetak predstavlja događaj koji se desi tokom izvršenja programa, a taj događaj poremeti normalni tok instrukcija.*

Izuzetak (en. `exception`) predstavlja događaj koji se desi tokom izvršenja programa, a taj događaj poremeti normalni tok instrukcija.

Postoje mnoge vrste grešaka koje mogu prouzrokovati izuzetke: od problema hardverske prirode (prestanak rada spoljašnje memorije ili diska), do jednostavnih grešaka u programiranju, kao što je pokušaj pristupa elementu liste koji ne postoji.

Kada se desi izuzetak, Python napravi objekat klase `Exception` i predala ga interpretéro. Ovaj objekat sadrži informaciju o grešci koja je napravljena.

Kada se izuzetak *"bací"* (en. `throwing an exception`), interpretér pokušava naći način za obradu izuzetka. Ukoliko postoji deo koda koji je odgovoran za obradu određenog tipa greške, onda se taj deo koda poziva. U tom slučaju se *"hvata"* izuzetak (en. `catching an exception`)

### Obrada izuzetaka u Python jeziku

U Python jeziku, obrada izuzetaka se sastoji od tri ključne reči:

- `try`
- `except`
- `finally`

Kombinacijom ovih ključnih reči moguće je detaljno obraditi greške koje se javljaju u Python programima, i omogućiti normalan tok rada programa.

U nastavku su dati konkretni primeri za upotrebu ovih ključnih reči prilikom obrade izuzetaka.

## OBRADA IZUZETAKA KROZ KLJUČNU REČ TRY

*U `try` bloku koda upisuje se deo koda koji može izazvati grešku, a u `except` bloku vršimo obradu izuzetka.*

### Primer: Pronalaženje greške u programu

Napisati program koji otvara nepostojeću datoteku za čitanje. Obraditi grešku `FileNotFoundException`.

U glavnom programu najpre treba napisati kod za otvaranje datoteke (u režimu za čitanje) koja ne postoji.

```
f = open('imedatoteke.txt', 'r')
```

Konzola vraća da datoteka ne postoji.

```
Traceback (most recent call last):
  File "d:/CS324/L06/Exceptions/test_exception.py", line 1, in <module>
    f = open('imedatoteke.txt', 'r')
FileNotFoundError: [Errno 2] No such file or directory: 'imedatoteke.txt'
```

Konzola vraća koja je greška u pitanju i u kojoj je liniji koda, što je prvi korak ka obradi izuzetaka. Ukoliko se mogu predvideti delovi koda koji mogu vratiti grešku ("baciti" izuzetak), onda je moguće koristiti blokove za obradu izuzetaka ("uhvatiti" izuzetak).

```
try:
    f = open('imedatoteke.txt', 'r')
except Exception:
    print('Datoteka ne postoji!')
```

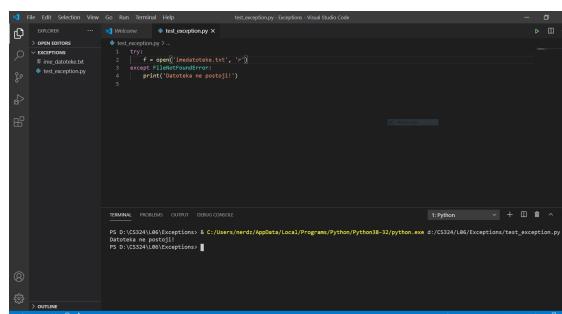
U **try** bloku koda upisuje se deo koda koji može izazvati grešku, a u **except** bloku vršimo obradu izuzetka.

#### Napomena:

Nakon ključne reči **except** piše se koji je izuzetak u pitanju. Ukoliko se želi opšti izuzetak, treba pisati **Exception**.

Ukoliko želimo specifičan izuzetak, posle **except** može se pisati *konkretni objekat klase exception*.

```
try:
    f = open('imedatoteke.txt', 'r')
except FileNotFoundError:
    print('Datoteka ne postoji!')
```



Slika 4.1 Hvatanje FileNotFoundError izuzetka. [Izvor: Autor]

# TIPOVI GREŠAKA I REDOSLED HVATANJA IZUZETAKA

*Najbolja praksa kod hvatanja izuzetaka jeste najpre napraviti kod za obradu sa konkretnim izuzecima, a nakon toga staviti opšti izuzetak.*

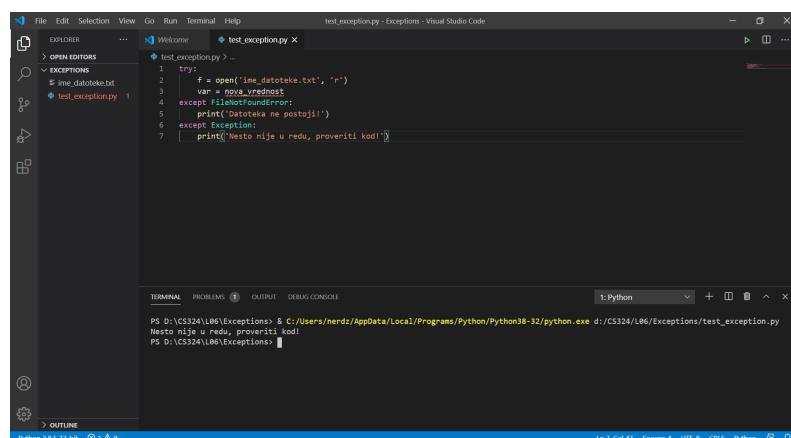
Budući da se greške mogu javiti zbog različitih uzroka, najbolja praksa jeste najpre napraviti kod za obradu sa konkretnim izuzecima, a nakon toga staviti opšti izuzetak.

```
# Cesti tipovi gresaka
EOFError - kada input() funkcija dodje do end-of-file uslova
FileNotFoundException - greska kada ne postoji datoteka kojoj se pristupa
FloatingPointError - greska u operaciji sa razlomljenim brojevima
ImportError - greska kod uvoza modula
IndexError - greska pri indeksiranju elementa iterabilnog objekta
KeyError - greska ako se ne pronadje kljuc u recniku ili skupu
KeyboardInterrupt - greska kada se pritisne Ctrl+C ili Delete na tastaturi
NameError - greska kada ime promenljive ne postoji u lokalnom ili globalnom oopsegu
SyntaxError - sintaksna greska
IndentationError - greska koja se javlja ukoliko nije kod dobro 'tabovan'
SystemError - interna greska interpretera
RuntimeError - ostale greske
```

## Primer: Hvatanje opšteg izuzetaka

Napisati program koji ima dve except izjave, jednu za FileNotFoundException, a drugu za opšti izuzetak. U programu otvoriti datoteku (ispravno ime) ali unutar try dela dodeliti vrednost promenljivi var nekoj promenljivoj koja prethodno nije definisana.

```
try:
    f = open('ime_datoteke.txt', 'r')
    var = nova_vrednost
except FileNotFoundException:
    print('Datoteka ne postoji!')
except Exception:
    print('Nesto nije u redu, proveriti kod!')
```



Slika 4.2 Hvatanje opšteg izuzetka. [Izvor: Autor]

## ISPISIVANJE PORAZUMEVANIH PORUKA ZA GREŠKE

*Moguće je videti podrazumevane poruke za greške ukoliko se naredba except proširi navođenjem imena objekta koji želimo da uhvatimo.*

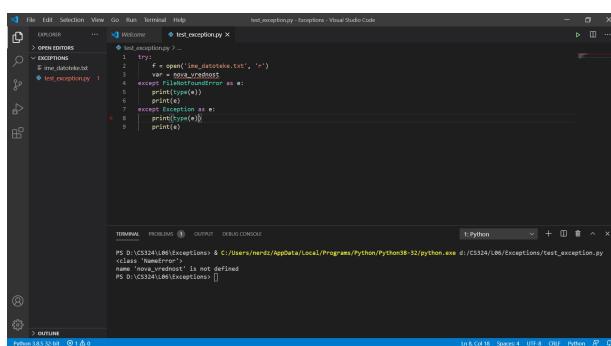
Moguće je videti podrazumevane poruke za greške ukoliko se naredba except proširi navođenjem imena objekta koji želimo da uhvatimo.

```
try:  
    f = open('ime_datoteke.txt', 'r')  
    var = nova_vrednost  
except FileNotFoundError as e:  
    print(type(e))  
    print(e)
```

U četvrtom redu koda dodajemo ključnu reč as, nakon čega pravimo objekat konkretnе klase izuzetka. Za deo koda u except bloku, konzola izbacuje sledeće:

```
<class 'FileNotFoundException'>  
[Errno 2] No such file or directory: 'imedatoteke.txt'
```

Na ovaj način može se videti koja je konkretna greška u pitanju, ali može se videti i šta je konkretno izazvalo grešku.



Slika 4.3 Hvatanje greške za podrazumevanom porukom. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## IZUZECI I ELSE I FINALLY BLOKOVI KODA

*Kod koji će se izvršiti kada nema izuzetaka biće u bloku koda nakon reči else, dok će se kod nakon reči finally izvršiti bilo da postoji greška ili ne.*

Ukoliko deo koda koji se nalazi u try bloku ne baci ni jedan izuzetak, onda je moguće odvojiti kod koji će se izvršiti (nakon provere izuzetka) komandom `else`.

```
try:  
    f = open('ime_datoteke.txt', 'r')  
except FileNotFoundError as e:  
    print(type(e))  
    print(e)  
except NameError as e:  
    print(type(e))  
    print(e)  
except Exception as e:  
    print(e)  
else:  
    sadrzaj = f.read()  
    print(sadrzaj)  
    f.close()
```

Blok `finally` izvršiće kod u svom bloku bilo da kod baci izuzetak ili ne.

```
try:  
    f = open('imedatoteke.txt', 'r')  
except FileNotFoundError as e:  
    print(type(e))  
    print(e)  
except NameError as e:  
    print(type(e))  
    print(e)  
except Exception as e:  
    print(e)  
else:  
    sadrzaj = f.read()  
    print(sadrzaj)  
finally:  
    print('Finally blok se izvršava...')  
    f.close()
```

## RUČNO DODAVANJE IZUZETAKA

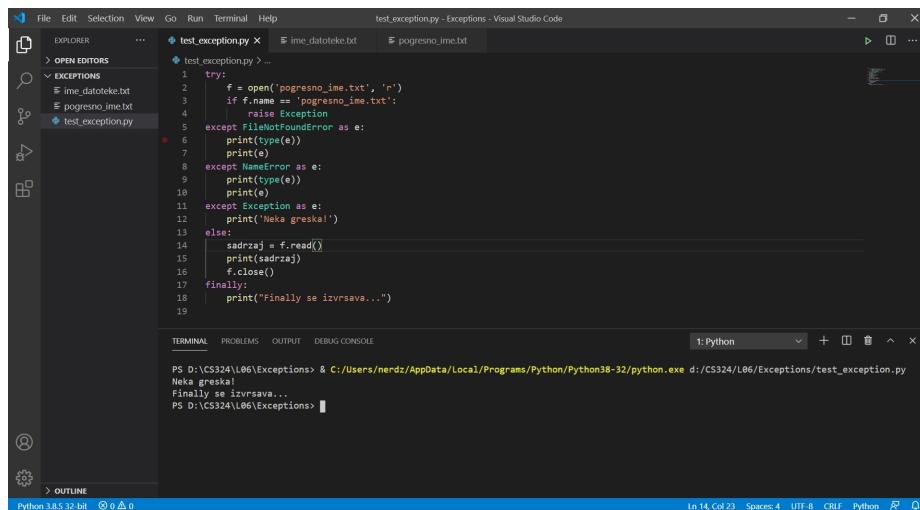
*Moguće je ručno dodati izuzetak korišćenjem komande `raise`*

Komanda `raise` služi za ručno dodavanja ("dizanje") izuzetaka.

Na sledećem primeru javlja se izuzetak ukoliko korisnik unese ime datoteke koje ima (u ovom primeru) ne treba da ima.

```
try:  
    f = open('pogresno_ime.txt', 'r')  
    if f.name == 'pogresno_ime.txt':  
        raise Exception  
except FileNotFoundError as e:
```

```
print(type(e))
print(e)
except NameError as e:
    print(type(e))
    print(e)
except Exception as e:
    print('Neka greska!')
else:
    sadrzaj = f.read()
    print(sadrzaj)
    f.close()
finally:
    print("Finally se izvrsava...")
```



Slika 4.4 Ručno dodavanje izuzetaka. [Izvor: Autor]

## ▼ Poglavlje 5

# Debagovanje u Python 3.x jeziku

## POJAM DEBAGOVANJA

*Debagovanje jestе proces koji podrazumeva identifikovanja i ispravljanja grešaka u kodu programa.*

Debagovanje (en. **debugging**) jestе proces koji podrazumeva identifikovanja i ispravljanja grešaka u kodu programa.

Postoje više metode debagovanja, i mogu uključivati interaktivno debagovanje, jedinično testiranje, integraciono testiranje, analiza log datoteka, kao i druge metode.

Debager (en. **debugger**) predstavlja softver ili softverski paket koji pruža pomoć u debagovanju.

### Proces debagovanja

Proces debagovanja se sastoji iz pet koraka:

1. Identifikacija problema
2. Opis problema
3. Hvatanje problema
4. Analiza uhvaćenog problema
5. Popravka problema

Nakon identifikacije problema, potrebno je opisati problem da bi se našao tačan razlog za pojavu problema. Nakon toga, treba reprodukovati pojavu problema, što obuhvata podešavanja svih promenljivih koje bi izazvale problem. Na osnovu reprodukcije, pokušati naći izvor problema, i na kraju rešiti problem, uz proveru da se taj problem ponovo ne javlja.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## DEBAGOVANJE U PYTHON JEZIKU KROZ MODUL PDB

*Python jezik ima ugrađen modul za debagovanje pdb (Python DeBugger), ali nije pregledan niti napredan. Bolje je koristiti debager unutar konkretnog IDE-a.*

Python jezik ima ugrađen modul za debagovanje pdb (Python DeBugger).

Dovoljno je uvesti ovaj modul, i na mestu gde se želi postaviti breakpoint, pozvati `pdb.set_trace()` funkciju. Kada program dođe do ove linije koda, u terminalu pokrenuće se Python Debugger.

```
import pdb

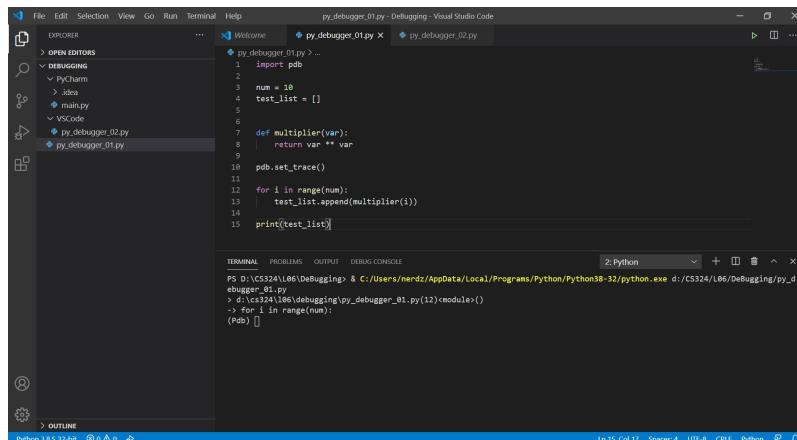
num = 10
test_list = []

def multiplier(var):
    return var ** var

pdb.set_trace()

for i in range(num):
    test_list.append(multiplier(i))

print(test_list)
```



Slika 5.1 Ugrađeni Python debugger pdb. Izvor: Autor.

Prilikom pokretanja pdb-a, na konzoli pojaviće se sledeći prompt:

```
(Pdb)
```

Kucanjem komandi može se izvršiti debagovanje. Komande su sledeće:

```
n - izvrsiti sledecu liniju koda
c - zavrsiti izvrsenje programa
l - listati prethodne i naredne tri linije koda u programu
s - uci u poziv funkcije
b - pokazati spisak svih breakpoint-ova
b[int] - postaviti breakpoint na liniji koda [int]
b[func] - postaviti breakpoint na ime funckije [func]
cl - izbrisati sve breakpoint-ove
cl[int] - izbrisati breakpoint na liniji koga [int]
p - stampa
```

Iako je modul pdb ugrađen u Python, nije najpregledniji, niti najnapredniji.

U nastavku obradiće se debuggeri u Visual Studio Code i PyCharm IDE.

## DEBAGOVANJE U VISUAL STUDIO CODE IDE

*Postavljanjem breakpoint-ova pored linije koda (u tzv. gutter-u ) moguće je pokrenuti debager u Visual Studio Code-u.*

Postavljanjem breakpoint-ova pored linije koda (u tzv. gutter-u ) moguće je pokrenuti debager u Visual Studio Code-u.

Program staje, zapravo pauzira sa izvršenjem prilikom nailaska na breakpoint i onda je moguće videti u levom panelu promenljive u tom trenutku izvršenja, moguće je postaviti sopstveni "watch" ukoliko je potrebno nešto specifično pratiti što program ne javlja automatski.

Program nastavlja sa radom klikom na *step over*, ulazi u funkciju koja se poziva sa *step into*, a izlazi sa *step out*.

### Primer:

Debugovati program koji ima klasu korisnik, sa atributima ime i email.

```
# Test VisualStudioCode

class Korisnik():
    # konstruktor
    def __init__(self, ime, email):
        self.ime = ime
        self.email = email

    # metoda koja vraca ime korisnika
    def vrati_ime(self):
        return self.ime

    # metoda koja vraca email korisnika
    def vrati_email(self):
        return self.email

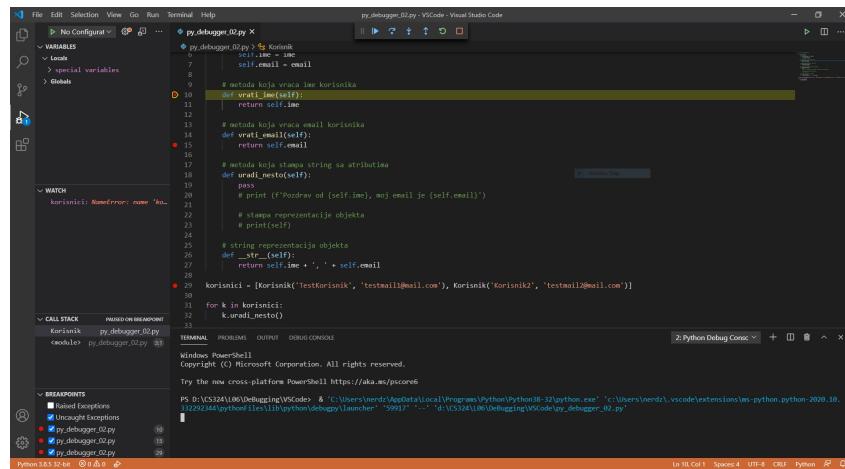
    # metoda koja stampa string sa atributima
    def uradi_nesto(self):
        pass
        # print (f'Pozdrav od {self.ime}, moj email je {self.email}')

        # stampa reprezentacije objekta
        # print(self)

    # string reprezentacija objekta
    def __str__(self):
        return self.ime + ', ' + self.email
```

```
korisnici = [Korisnik('TestKorisnik', 'testmail1@mail.com'), Korisnik('Korisnik2', 'testmail2@mail.com')]

for k in korisnici:
    k.uradi_nesto()
```



Slika 5.2 Debugovanje u Visual Studio Code-u. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## DEBAGOVANJE U PYCHARM IDE

*PyCharm debager automatski postavlja vrednosti promenljiva kao komentar nakon izjave u samom kodu!*

Kao i kod Visual Studio Code IDE, u PyCharm-u je moguće postaviti breakpoint pored same linije koda.

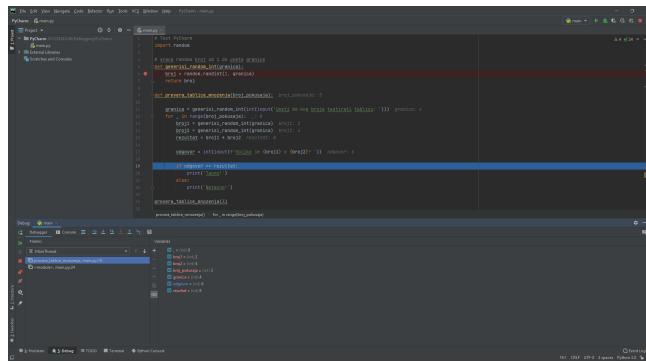
PyCharm debager otvara tab pored konzole za debagovanje, i prilikom nailaska na breakpoint moguće je videti sve promenljive u lokalnom opsegu, ali i pratiti ličnu unetu izjave.

Ostim toga, PyCharm debager automatski postavlja vrednosti promenljiva kao komentar nakon izjave u samom kodu! Na taj način moguće je nesmetano pratiti tok programa i uvek imati u vidu koje su vrednosti promenljiva, a da se pritom gleda u izvorni kod.

```
# Test PyCharm
import random

# vraca random broj od 1 do unete granice
def generisi_random_int(granica):
    broj = random.randint(1, granica)
    return broj
```

```
def provera_tablice_mnozenja(broj_pokusaja):  
  
    granica = generisi_random_int(int(input('Uneti do kog broja testirati tablicu: ')))  
    for _ in range(broj_pokusaja):  
        broj1 = generisi_random_int(granica)  
        broj2 = generisi_random_int(granica)  
        rezultat = broj1 * broj2  
  
        odgovor = int(input(f'Koliko je {broj1} x {broj2}? '))  
  
        if odgovor == rezultat:  
            print('Tacno!')  
        else:  
            print('Netacno!')  
  
provera_tablice_mnozenja(3)
```



Slika 5.3 Debugovanje u PyCharm-u. Izvor: [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 6

### Pokazna vežba #6

#### POKAZNA VEŽBA #1

*Prva zadatak pokazuje rad sa datotekama. Procenjeno vreme 20 minuta*

U pokaznoj vežbi proći će se kroz zadatak koji obuhvata sledećim tematskim jedinicama ove lekcije:

- Rad sa datotekama
- Obrada Izuzetaka
- Debagovanje

Procenjeno vreme izrade pokaznih vežbi je 45 minuta.

#### ZADATAK #1

*Prvi zadatak pokazuje rad sa datotekama. Procenjeno vreme 20 minuta*

**Zadatak #1** (20 minuta)

Napisati funkciju ascii\_sifre(broj\_sifri, duzina\_sifre) koja će generisati nasumičnu šifru od proizvoljnog broja ASCII karaktera (velika i mala slova). Funkcija vraća duple stringova čiji su elementi šifre.

U glavnom programu napraviti CSV datoteku **sifre.csv** koja će sadržati dva reda. Prvi red sadrži stringove oblika "Sifra #", gde je # indeks liste, a drugi red sadrži šifre.

koristiti module random, string i csv.

Debug-ovati program.

```
import random
import string
import csv

def ascii_sifre(broj_sifri,duzina_sifre):
    lista_sifre = []
    letters = string.ascii_letters
    for i in range(broj_sifri):
```

```
sifra = ''  
for j in range(duzina_sifre):  
    sifra += random.choice(letters)  
lista_sifre.append(sifra)  
return tuple(lista_sifre)  
  
lista = ascii_sifre(8, 8)  
  
redni_broj_sifre = []  
sama_sifra = []  
  
for i in range(0, len(lista)):  
    redni_broj_sifre.append('Sifra #' + str(i + 1))  
    sama_sifra.append(lista[i])  
  
with open('sifre.csv', 'w') as csv_file:  
    csv_writer = csv.writer(csv_file, delimiter=',')  
    csv_writer.writerow(redni_broj_sifre)  
    csv_writer.writerow(sama_sifra)
```

## ZADATAK #2

*U drugom zadatku treba ručno podići izuzetke kad se ispune neki uslovi. 25 minuta*

### Zadatak #2 (25 minuta)

Napraviti klasu **geo\_figura** koja može imati proizvoljan broj argumenata. Argument predstavljaju stranice geometrijske figure. Napraviti metodu **koja\_je\_figura** koja ispituje tip figure na sledeći način:

- Ukoliko je uneseno 3 stranice, štampati da je trougao, četiri za četvorougao, itd.
- Ukoliko je trougao pravougli (proveriti testiranjem Pitagorine teoreme) izračunati i štampati površinu.
- Ukoliko je četvorougao pravougaonik ili romb (testirati da li ima dva para identičnih stranica), izračunati i štampati površinu.
- Ukoliko je unet petougao ili veći, javiti da se površina ne može izračunati.
- Za bilo koju unetu figuru izračunati obim.

Podići izuzetke ako:

- Korisnik ne unese nikakvu vrednost za stranicu.
- Korisnik unese negativnu vrednost za stranicu.
- Korisnik unese nulu za stranicu.
- Korisnik unese samo jednu ili dve stranice.

Debug-ovati program.

```
try:
    class geo_figura():
        stranice = []
        def __init__(self, *duzina_stranice):
            self.stranice.extend(duzina_stranice)
            for i in range(0, len(self.stranice)):
                if self.stranice[i] == 0:
                    raise Exception('Stranica ne može biti jednaka nuli!')
                if self.stranice[i] < 0:
                    raise Exception('Stranica ne može biti negativna!')

                if len(self.stranice) == 0:
                    raise Exception("Ne postoji nultougao")
                elif (len(self.stranice) == 1) or (len(self.stranice) == 2):
                    raise Exception('Dvougao je emisija, a jednougao ne postoji.')

        def koja_je_figura(self):
            self.obim = sum(self.stranice)
            self.stranice.sort()
            if len(self.stranice) == 3:
                print("U pitanju je trougao!")
                print(f'Obim je {self.obim}')
                if (self.stranice[0]**2 + self.stranice[1]**2) ==
self.stranice[2]**2:
                    print(f'Povrsina je {((self.stranice[0] * self.stranice[1])/2)}')

            elif len(self.stranice) == 4:
                print("Cetvorougao")
                print(f'Obim je {self.obim}')
                if (self.stranice[0] == self.stranice[1]) and (self.stranice[2] ==
self.stranice[3]):
                    print(f'Povrsina je {self.stranice[0]*self.stranice[2]}')

            else:
                print(f"U pitanju je {len(self.stranice)}-ugao")
                print(f'Obim je {self.obim}')

    except Exception as e:
        print('Neka greska!')
```

## ▼ Poglavlje 7

### Individualna vežba #6

#### UVOD U INDIVIDUALNU VEŽBU #6

*U individualnoj vežbi 6 rade se tri zadatka koje bi studenti mogli samostalno da odrade u toku dva časa.*

Individualna vežba #6 odnosi se na rad sa datotekama, upis, čitanje i čuvanje, ali i sa procesom debagovanja i obradom izuzetaka.

Procenjeno trajanje individualnih vežbi iznosi 90 minuta.

#### **Zadatak #1 (30 minuta)**

Napisati program koji će učitati CSV datoteku (data u prilogu) koja sadrži spisak svih predmeta (šifre u jednom redu, i naziva u drugom).

Napraviti listu koja u sebi sadrži predmete koje slušate u ovom semestru po šifri.

Iz CSV datoteke učitati nazine samo onih predmeta koje slušate. Sačuvajte predmete koje slušate u ovom semestru u novu CSV datoteku.

CS101,IT101,MA104,NT111,CS102,IT210,CS115,NT112,IT331,SE201,IT350,NT213,CS230,IT370,CS220,CS323,IT255,CS225,IT335,CS324,SE325,IT355,IT381,IT376,CS322,SE401,OM350,CS232,NT213

Uvod u objektno-orientisano programiranje,Osnove informacionih tehnologija,Matematika,Engleski 1,Objekti i apstrakcija podataka,Sistemi informacionih tehnologija,Diskretne strukture,Engleski 2,Računarske mreže i komunikacij,Uvod u softversko inženjerstvo,Baze podataka,Engleski za informatičare,Distribuirani sistemi,Interakcija čovek-računar,Arhitektura računara,C/C++ programski jezik,Veb sistemi 1,Operativni sistemi,Administracija računarskih sistema i mreža,Skripting jezici,Upravljanje projektima razvoja softvera,Veb sistemi 2,Zaštita i bezbednost informacija,Robotika,C# Programska jezik,Timski razvoj softvera,Preduzetništvo,Programiranje 2D igara,Engleski za informatičare

#### **Zadatak #2 (30 minuta)**

Napisati program koji će sumirati sve brojeve koje korisnik unese sa komandne linije, ignorajući svaki ulaz koji nije validni broj. Program treba da pokazuje trenutnu sumu posle svakog unetog broja.

Javiti poruku ukoliko se ne unese ne-numeričku ulaz, ali nastaviti sa unosom. Izaći iz programa kada se unese prazna linija (Enter). Program treba da radi za celobrojne i razlomljene brojeve.

#### **Zadatak #3 (30 minuta)**

Programe individualnih vežbi 5. i 6. lekcije debug-ovati pomoću debugger unutar IDE-a.  
Diskutovati međusobno

## ✓ Poglavlje 8

### Domaći zadatak #6

#### DOMAĆI ZADATAK

*Domaći zadatak #6 se okvirno radi 3h*

##### **Zadatak #1**

Napisati program koji će praviti Vaš raspored časova i sačuvati kao CSV datoteku.

Otvoriti CSV datoteku u programu Excel (ili sličnom programu) kao dokaz da je uspešno napravljena datoteka.

##### **Zadatak #2**

Napisati funkciju **srednja\_vrednost(niz)** koja vraća srednju vrednost elemenata niza. Niz može biti lista celobrojnih ili razlomljenih brojeva. Koristiti pomoćnu promenljivu **tmp** koja akumulira vrednosti niza, element po element.

Uhvatiti izuzetak ukoliko element liste nije jedan od ovih tipova podataka (posebne izuzetke za tip string, list, dict, tuple), i vratiti posebne poruke.

U glavnom programu pozvati funkciju 4 puta sa različitim parametrima i pratiti vrednosti ulaznog parametra funkcije (**niz**) i **tmp** promenljive kroz debugger.

Screenshot-ovati nekoliko stanja debugger-a.

##### **Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije nedelju dana nakon predavanja za 100% poena. Nakon toga poeni se umanjuju za 50%.

##### **Online studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganja ispita. Domaći zadaci se brane!

Domaći zadatak poslati dr Nemanji Zdravkoviću: nemanja.zdravkovic@metropolitan.ac.rs

Obavezno koristiti uputstvo za izradu domaćeg zadatka.

Uz .doc dokument (koji treba sadržati i screenshot svakog urađenog zadatka kao i komentare za zadatak), poslati i izvorne i dodatne datoteke.

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Zaključak lekcije #6*

#### **Rezime:**

U ovo lekciji bilo je reči o radu sa datotekama i obradom izuzetaka u Python 3.x jeziku.

Posebna pažnja je data na radu sa CSV i JSON datotekama, i kako čitati i pisati u takve datoteke.

Zatim, bilo je reči o obradi izuzetaka u Python jeziku, kako napisati pravilan kod koji može javiti poruku o izuzecima kada se desi neka greška.

Konačno, bilo je reči u procesu debug-iranja unutar Python koda, i kako koristiti ugrađen debugger, ali i kako koristiti napredne debugger-e unutar IDE-a.

Primeri u okviru lekcije kao i zadaci za individualni rad treba da osposobe studente za rad u Python 3.x jeziku kroz rad sa datotekama, obradom izuzetaka, kao i proces debug-ovanja.

#### **Literatura:**

1. David Beazley, Brian Jones, *Python Cookbook: Recipes for Mastering Python 3*, 3rd edition, O'Reilly Press, 2013.
2. Mark Lutz, *Learning Python*, 5th Edition, O'Reilly Press, 2013.
3. Andrew Bird, Lau Cher Han, et. al, *The Python Workshop*, Packt Publishing, 2019.
4. Al Sweigart, *Automate the boring stuff with Python*, 2nd Edition, No Starch Press, 2020.



## CS324 - SKRIPTING JEZICI

### Python biblioteke i moduli

Lekcija 07

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 07

### *PYTHON BIBLIOTEKE I MODULI*

- ✓ Python biblioteke i moduli
- ✓ Poglavlje 1: Uvod u biblioteke, pakete i module
- ✓ Poglavlje 2: Menadžer Python paketa - PIP
- ✓ Poglavlje 3: Paketi math i datetime
- ✓ Poglavlje 4: Regularni izrazi u Python-u – Paket re
- ✓ Poglavlje 5: Python i JSON datoteke – Paket json
- ✓ Poglavlje 6: Python i relacione baze podataka - Paket sqlite
- ✓ Poglavlje 7: Pokazna vežba #7
- ✓ Poglavlje 8: Invividualna vežba #7
- ✓ Poglavlje 9: Domaći zadatak #7
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ❖ Uvod

### UVOD

*Python poseduje pregršt besplatnih biblioteka i modula za specifične svrhe koje se lako mogu instalirati i koristiti.*

U sedmoj lekciji biće reči o modulima i bibliotekama koje Python koristi.

Ono što je specifično za jedan scripting jezik kao što je Python, jeste da poseduje pregršt besplatnih biblioteka i modula za specifične svrhe. Upravo je raznovrsnost biblioteka i modula učinilo Python toliko popularnim za opštu primenu. Pre svega, potrebno je znati razliku između *paketa* i *modula*. Bilo koja python datoteka predstavlja jedan modul, i ime tog modula je isto kao i ime datoteke (bez .py ekstenzije). Paket predstavlja skup odnosno kolekciju modula. Paket kao takav je zapravo direktorijum Python modula koji takođe sadrži i dodatnu datoteku za inicijalizaciju, da bi se paket razlikovao od običnog direktorijuma koji sadrži Python datoteke.

Moduli i paketi se mogu jednostavno dodati koristeći python instalater paketa, ili skraćeno PIP. Korišćenjem PIP modula vrlo lako se mogu ubaciti dodatni paketi, i to direktno iz komandne linije Pythona. Kada je potrebno da se neki modul ili paket da uveze u trenutnu datoteku, koristi se komanda import, posle koje se navede ime paketa. Paket se može i skraćeno nazvati u trenutnoj datoteci, i kao takav koristiti u nastavku. U nastavku lekcije biće obrađeni paketi, odnosno moduli za matematiku i datum i vreme, za rad sa JASON datotekama, kao i za povezivanje sa relacionim bazama podataka.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 1

# Uvod u biblioteke, pakete i module

## PYTHON PROGRAMSKO JEZIK I OSOBINA MODULARNOSTI

*Python poseduje mogućnost da se radi sa konkretnim funkcijama, metodama, pa i celim radnim okvirima prilikom pisanja programa za specifičnu primenu.*

Prilikom pisanja programa u bilo kom programskom jeziku (pa i u Python programskom jeziku), veoma je neefikasno uvek pisati sve funkcionalnosti programa u jednoj izvornoj datoteci.

Podelom funkcionalnosti programa na više datoteka uvodi se strukture u pisanju i organizaciju programa. Na ovaj način jednostavnije i efikasnije je pisati glavni program, dok se pojedine funkcionalnosti mogu pozvati iz eksternih programa.

Može se reći da je srž Python jezika upravo osobina modularnosti, tj. mogućnost da se radi sa konkretnim funkcijama, metodama, pa i celim radnim okvirima prilikom pisanja programa za specifičnu primenu.

Ovu osobinu Python je nasledio od MODULA-3 programskog jezika, koji se upravo i bazira na modularnosti.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## MODUL U PYTHON JEZIKU

*U Python programskom jeziku, svaka Python datoteka sa ekstenzijom .py predstavlja jedan modul.*

Modul (en. **module**) predstavlja deo softvera koji poseduje specifičnu funkcionalnost.

U Python programskom jeziku, svaka Python datoteka (datoteka sa ekstenzijom .py) predstavlja jedan modul. Ime modula biće isto kao i ime datoteka.

Modul, kao i svaka druga Python datoteka, može imati promenljive, funkcije, klase sa atributima i metodama koje su već definisane i implementirane.

### **Uvoz modula**

Modul se može uvesti korišćenjem ključne reči import, nakon čega se navodi ime datoteke koje se uvozi. Postoje različiti načini uvoza modula

Modul se ceo može uvesti, i to bez preimenovanja. Tada je, prilikom poziva dela koda iz tog modula, u glavnom programu navedemo i ime modula, pa deo koda koji uvozimo. Ovaj pristup može biti složen ukoliko je ime modula dugačko ili ako se ne nalazi u istom direktorijumu. Zbog toga se može preimenovati modul zbog lakoće pisanja.

Modul ili deo modula se može uvesti u imenski prostor glavnog programa, i onda nije neophodno navesti ime modula prilikom svakog poziva dela koda.

### **Primer: Uvoz modula (15 minuta)**

Napisati program koji će u sebi sadržati dve funkcije:

- d6roller() - vraća nasumičnu celobrojnu vrednost od 1 do 6
- d20roller() - vraća nasumičnu celobrojnu vrednost od 1 do 20

Takođe, u ovom programu je definisana i klasa dice\_roller(broj\_strana) sa sledećim metodama:

konstruktor sa parametrom o broju strana

- roll() - vraća nasumičnu celobrojnu vrednost od 1 do unetog broja strana
- roll\_many(broj\_bacanja) - vraća zbir višestrukog bacanja
- roll\_adv() - vraća veći rezultat od dva bacanja
- roll\_dis() - vraća manji rezultat od dva bacanja

U odvojenoj datoteci uvesti:

- ceo modul bez preimenovanja
- ceo modul sa imenovanjem
- samo klasu dice\_roller (u postojeći imenski prostor)
- ceo modul u postojeći imenski prostor

## UVOZ MODULA - PRIMER

### *Primer za uvod modula u zasebnu datoteku*

```
# datoteka za import modula
import random

# funkcija za simulaciju bacanje kockica
def d6roller():
    return random.randint(1,6)

# funkcija za simulaciju bacanje 20-strane kockice
def d20roller():
    return random.randint(1,20)

# klasa za bacanja razlicitih kockica
```

```
class dice_roller():

    def __init__(self, number_of_sides):
        self.number_of_sides = number_of_sides

    def roll(self):
        return random.randint(1, self.number_of_sides)

    def roll_many(self, number_of_rolls):
        self.result = 0
        self.number_of_rolls = number_of_rolls
        for _ in range(self.number_of_rolls):
            self.result += random.randint(1, self.number_of_sides)
        return self.result

    def roll_adv(self):
        return max(random.randint(1, self.number_of_sides), random.randint(1,
self.number_of_sides))

    def roll_dis(self):
        return min(random.randint(1, self.number_of_sides), random.randint(1,
self.number_of_sides))
```

```
# uvoz bez preimenovanja
import cs324_L07_01_import_modula
print(cs324_L07_01_import_modula.d6roller())

# uvoz sa preimenovanjem
import cs324_L07_01_import_modula as dices
print(dices.d20roller())

# uvoz konkretnog dela koda u trenutni imenski prostor
from cs324_L07_01_import_modula import dice_roller
x = dice_roller(20)
print(x.roll())

# uvoz svega u trenutni imenski prostor
from cs324_L07_01_import_modula import *
print(d20roller())
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PAKET U PYTHON JEZIKU

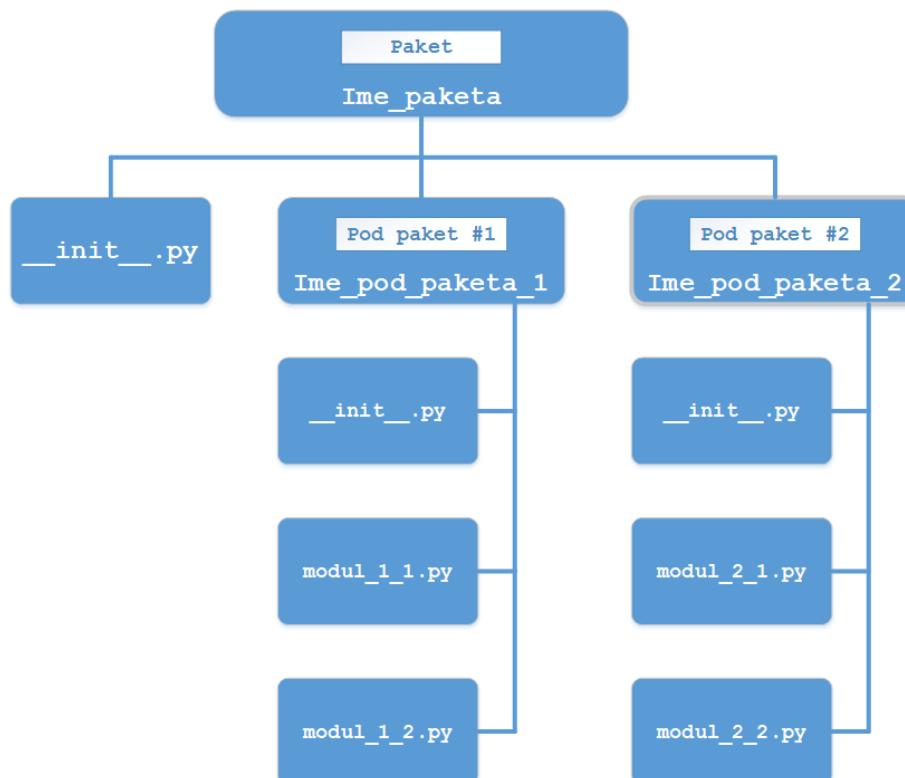
*Paket u Python jeziku predstavlja kolekciju modula unutar direktorijuma, a direktorijum mora sadržati konstruktor paketa.*

Paket (en. **package**) u Python jeziku predstavlja kolekciju modula unutar direktorijuma.

Treba istaći da bilo koji direktorijum sa Python modulima (direktorijum sa datotekama sa .py ekstenzijom) ne predstavlja modul sam po sebi, već mora ispuniti dodatan uslov da poseduje datoteku konstruktora modula, koja će biti jedna prazna `__init__.py` datoteka. Na taj način Python interpreter razumeće da je u pitanju paket, koji se može uvesti.

Kao i kod strukture direktorijuma, i paketi mogu sadržati pod-pakete (en. sub-packet), ali unutar svakog paketskog direktorijuma treba postojati paketni konstruktor.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**



Slika 1.1 Struktura Python paketa i modula. [Izvor: Autor]

## UVOZ UGRAĐENOG PAKETA

*Python interpreter uvek će prepoznati ugrađene pakete koji uvek može uvesti.*

Python 3 jezik dolazi sa mnoštvo ugrađenih paketa za specifične svrhe.

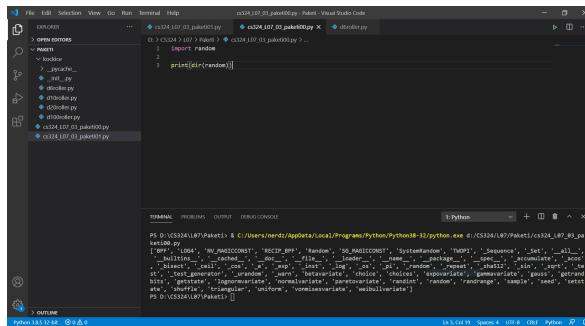
Kod ugrađenih paketa, iako se ne nalaze u direktorijumu u kome je glavni program, Python interpreter prepoznaće da se radi o ugrađenim paketima i moći će da ih uveze uvek.

Do sada su obrađeni primeri koji su uvozili paket `math` (za rad sa matematičkim funkcijama) i paket `random` (za rad sa nasumičnim brojevima).

Moguće je videti šta od funkcija i klasa sadrži korišćenjem funkcije `dir()`.

## Primer: izlistati moguće funkcije paketa radnom. (2 minuta)

```
import random  
  
print(dir(random))
```



Slika 1.2 Lista mogućih funkcija paketa random. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## UVOD KORISNIČKOG PAKETA

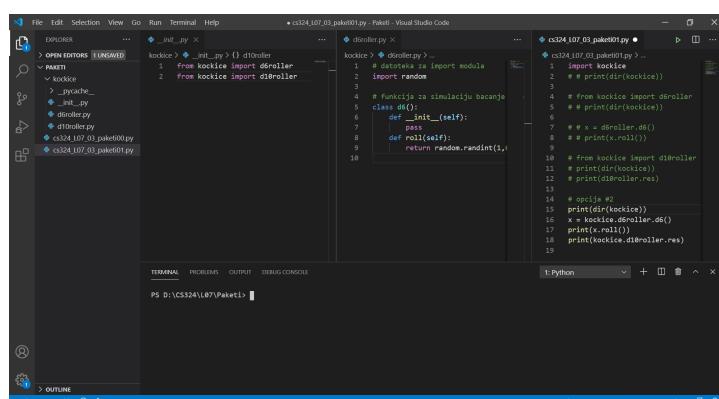
*Kod uvoza korisničkih paketa potrebno je uvesti pojedinačne module, a ne samo ime paketa.*

Korisnički paket se na vrlo sličan način može uvesti u trenutni radni direktorijum.

Postupak je sledeći:

- Napraviti prazni paketni konstruktor
  - U glavnom programu, korišćenjem naredbe `import`, uvesti korisnički paket po imenu
  - Nakon toga, uvesti pojedinačne module u imenski prostor korišćenjem naredbi `from [package] import [module]`

Alternativno, moguće je unutar paketnog konstruktora navesti pojedinačne naredbe uvoza da bi na taj način korisnički moduli uvek bili dostupni.



Slika 1.3 Uvoz korisničkih modula preko paketa. [Izvor: Autor]

### Primer: (10 minuta)

Napraviti paket kockice koje će imati pojedinačne module za vraćanje nasumičnog broja za dve različite kockice. U glavnom programu uvesti paket i isprobati funkcionalnost.

```
# --- moduli paketa kockice (pojedinačne datoteke) ---
# -----
# modul #1 d6roller.py
import random

# funkcija za simulaciju bacanje kockica
class d6():
    def __init__(self):
        pass
    def roll(self):
        return random.randint(1,6)
# -----
# modul #2 d10roller.py
import random
res = random.randint(1,10)
# -----
# glavni program main.py
import kockice
print(dir(kockice))

from kockice import d6roller
x = d6roller.d6()
print(x.roll())

from kockice import d10roller
print(d10roller.res)

print(dir(kockice))
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PYTHON BIBLIOTEKA

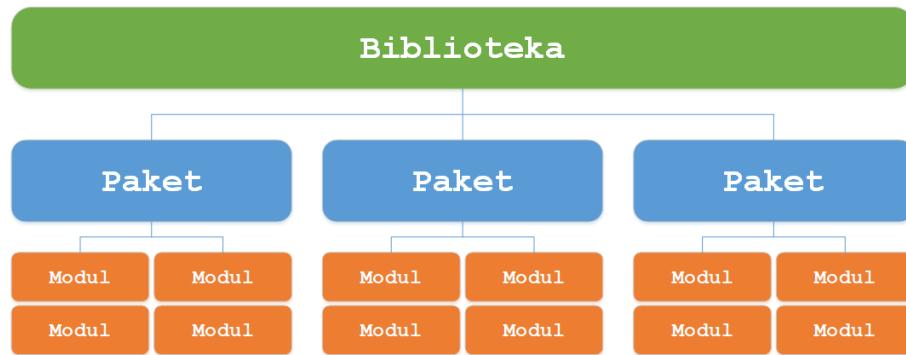
*Skup paketa (sa modulima) i dodatnim podešavanjima predstavlja Python biblioteku. Biblioteka koja se koristi prilikom pokretanja Python-a jeste standardna biblioteka.*

Kao što je paket predstavlja skup modula, tako i biblioteka (en. library) predstavlja skup paketa.

Moguće je napraviti sopstvenu biblioteku koja bi bila skup paketa i modula, ali i dodatnih podešavanja prilikom pokretanja IDE-a.

Zapravo, podešavao bi se virtuelni prostor (en. virtual environment).

Korisničke biblioteke su izvan skupa ovog predmeta.



Slika 1.4 Odnos modula, paketa i biblioteka u Python programskom jeziku. [Izvor: Autor]

### Standardna Python biblioteka

Standardna Python biblioteka se sastoji iz više različitih komponenti. Sadrži najpre tipove podataka koji se mogu smatrati da su "srž" programskog jezika, npr. brojevi i liste.

Biblioteka takođe sadrži ugrađene funkcije i izuzetke - objekte koji se mogu koristiti u svom kodu pisanom na Python jeziku bez potrebe za naredbom import.

Najveći deo biblioteke sadrži skup modula. Neki moduli su napisani na C jeziku i ugrađeni su u sam Python interpreter, dok su neki moduli napisani na Python jeziku i uvoze se u izvornom obliku.

Pojedini moduli služe kao interfejs operativnom sistemu i hardveru, ili specifičnim aplikacionim domenima, kao što je www.

Pojedini moduli dostupni su u svakoj verziji Python instalacije i za svaki operativni sistem, dok su neki dostupni samo kroz određene opcije konfigurisanja. Više o standardnoj biblioteci na stranici:

<https://docs.python.org/3/library/>

## ✓ Poglavlje 2

# Menadžer Python paketa - PIP

## UVOD U PIP

*PIP menadžer datoteka se poziva u komandnoj liniji i pruža detaljnu kontrolu nad instaliranim paketima u Python jeziku.*

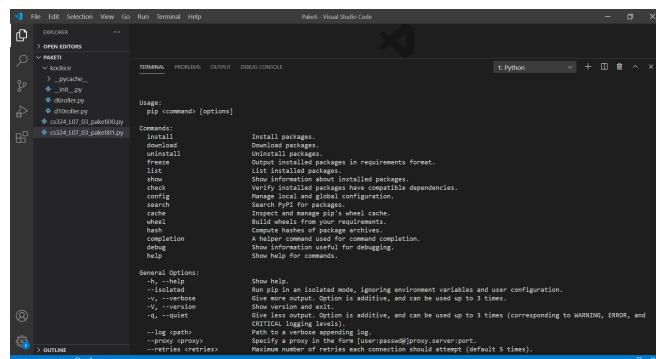
Do sada je bilo reči o ugrađenim i korisnički-definisanim paketima za Python.

Postavlja se pitanje: Kako ubaciti tuđe pakete i kako im pristupiti?

PIP menadžer paketa (en. **PIP - pip installs packets**) jeste softver koji dolazi uz instalaciju Python programskog jezika sa zadatkom da upravlja eksternim paketima.

Koristi se u komandnoj liniji i iako ima samo nekoliko osnovnih komandi, pruža detaljnu kontrolu nad instaliranim paketima. Prilikom instalacije Python jezika, instalira se i **pip**, i može se pristupiti komandom **pip**

```
C:\pip
```



Slika 2.1 Pokretanje pip menadžera u konzoli. [Izvor: Autor]

### Osnovne komande

PIP nudi sledeće osnovne komande:

- **install** - instalira pakete
- **download** - preuzima pakete
- **uninstall** - izbacuje pakete
- **freeze** - pokazuje instalirane pakete u formatu potražnje
- **list** - pokazuje instalirane pakete
- **show** - pokazuje informaciju o instaliranom paketu
- **check** - proverava zavisnosti instaliranih paketa

- **config** - upravlja lokalnom i globalnom konfiguracijom
- **search** - pretraga paketa
- **hash** - računa heš vrednosti arhiva paketa
- **completion** - opcija za auto-complete
- **debug** - pokazuje informaciju za debagovanje
- **help** - pomoć

## PRETRAGA I INSTALACIJA NOVOG PAKETA

*Pretraga, pregled, instalacija novih paketa preko PIP-a je jednostavno i izvršava se svega u nekoliko komandi.*

### Pretraga paketa

Pretraga paketa vrši se pomoću komande `search`, nakon koje treba navesti ime paketa.

Ukoliko postoji paket sa tim nazivom, pip vraća naziv, verziju, i opis paketa.

```
pip search ime_paketa
```

### Instalacija paketa

Instalacija paketa vrši se pomoću komande `install`, nakon koje treba navesti ime paketa.

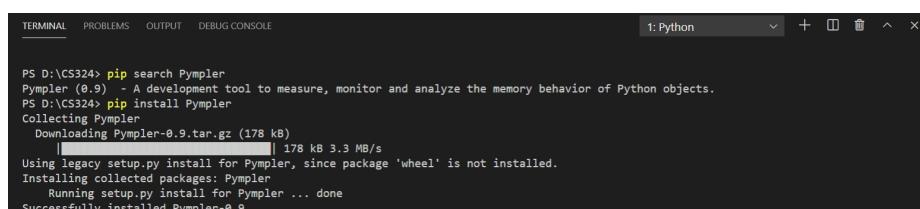
```
pip install ime_paketa
```

### Pregled instaliranih paketa

Pregled paketa vrši se pomoću komande `list`, nakon koje pip vraća listu instaliranih paketa u obliku tabele, gde su u jednoj koloni paketi, a u drugoj koloni verzije.

```
pip list
```

Primer: potražiti i instalirati paket Pympler



```
PS D:\CS324> pip search Pympler
Pympler (0.9) - A development tool to measure, monitor and analyze the memory behavior of Python objects.
PS D:\CS324> pip install Pympler
Collecting Pympler
  Downloading Pympler-0.9.tar.gz (178 kB)
    [██████████] 178 kB 3.3 MB/s
Using legacy setup.py install for Pympler, since package 'wheel' is not installed.
Installing collected packages: Pympler
  Running setup.py install for Pympler ... done
Successfully installed Pympler-0.9
```

Slika 2.2 Pretraga i instalacija paketa Pympler. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## AŽURIRANJE INSTALIRANIH PAKETA

*Provera paketa koji nisu ažurirani vrši se pomoću komande `list`, sa dodanim argumentom `--outdated`.*

Paketi se često ažuriraju, te je potrebno znati da li na računaru na kojem se piše program instalirana najnovija verzija paketa.

Provera paketa koji nisu ažurirani vrši se pomoću komande `list`, sa dodanim argumentom `--outdated`:

```
pip list --outdated
```

Sa dodatnim argumentom, `list` komanda vratiće tabelu instaliranih paketa u jednoj koloni, sa izdanjem novije verzije u drugoj koloni.

Package	Version	Latest	Type
colorama	0.4.3	0.4.4	wheel
isort	5.4.2	5.6.4	wheel
lazy-object-proxy	1.4.3	1.5.1	wheel
setuptools	47.1.0	50.3.2	wheel
toml	0.10.1	0.10.2	wheel

Slika 2.3 Lista paketa koji nisu ažurirani. [Izvor: Autor]

### Ažuriranje paketa

Ažuriranje paketa moguće je takođe kroz komandu `install`, ali sa dodatnim argumentom `-U`.

```
pip install -U ime_paketa
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PREGLED INSTALIRANIH PAKETA, IZVOZ U DATOTEKU I BRISANJE PAKETA

*Dobra praksa jeste imati spisak potrebnih paketa u posebnoj datoteci, ili u komentaru samog programa.*

Komanda `list` vraća sve instalirane pakete u obliku tabele, ali nekada nije poželjno na ovaj način predstaviti sve pakete.

Često se dešava da treba javiti koji su sve paketi potrebni da bi se program pokrenuo. Zbog toga je efikasnije koristiti komandu `freeze`, koja, kao i `list`, vraća spisak instaliranih paketa, ali u drugom obliku.

```
pip freeze
```

Dobra praksa jeste izvesti ovako formatiran spisak u posebnu datoteku, što je moguće dodavanjem komande:

```
pip list > izlazna_datoteka.ekstenzija
```

Na ovaj način, datoteka se može dodati u projektnu dokumentaciju, ili i dodati kao komentar u izvornom kodu koji koristi paket, kao komentar.

### **Brisanje paketa**

[Brisanje paketa](#) vrši se pomoću komande `uninstall`, nakon koje treba navesti ime paketa.

```
pip uninstall ime_paketa
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 3

### Paketi math i datetime

#### UVOD U MATH PAKET

*Math je ugrađeni paket i nije neophodno instalirati naknadno.*

**Paket math** je ugrađen paket, što znači da nije potrebno naknadno instalirati paket nakon instalacije Python programskog jezika.

Međutim, funkcije paketa math nisu direktno dostupne u imenskom prostoru, te je neophodno koristiti sledeću komandu:

```
# rad sa realnim brojevima
import math
```

#### **Pitanje:**

**U zadatku koji računa rešenja kvadratne jednačine, zog čega je bilo neophodno koristiti cmath, a ne math?**

Ukoliko se radi sa kompleksnim brojevima, potrebno je umesto *math* koristiti paket *cmath*.

```
# rad sa kompleksnim brojevima
import cmath
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

#### UVOD U DATETIME PAKET

*Datetime paket predstavlja glavni paket za rad sa datumima i vremenskim zapisima.*

**Paket datetime** jeste glavni paket za rad sa datumima i vremenom. Ovaj paket je veoma bitan prilikom razvoja aplikacija koje imaju u sebi neke vremenske zapise.

Ovaj paket je takođe ugrađen i ne treba se posebno instalirati, ali je potrebno ručno uvesti u imenski prostor.

```
import datetime
```

Rad sa datumima i vremenom je jako bitan u bilo kojoj **real-time** aplikaciji, ali i prilikom rada sa datotekama, kada je potrebno iz datoteke uneti string i parsirati kao datum, ili obrnuto, kada treba datum da se pretvori u string koji treba upisati u datoteku.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ 3.1 Rad sa math paketom

### FUNKCIJE PAKETA MATH

*Ugrađeni paket math sadrži matematičke funkcije koje su definisane C standardom*

Ugrađeni paket **math** sadrži matematičke funkcije koje su definisane C standardom - funkcije koje su identične (ili jako slične) onima uz C familije jezika.

U nastavku je dato objašnjenje pojedinih funkcija paketa math.

Funkcija ***floor()*** vraća najbliži ceo broj **manji** od unetog broja.

```
# funkcija floor vraca najblizi manji ceo broj
x = 3.4566
print(math.floor(x))
-----
output:
>>> 3
```

Funkcija ***ceil()*** vraća najbliži ceo broj **veći** od unetog broja.

```
# funkcija ceil vraca najblizi veci ceo broj
x = 3.4566
print(math.ceil(x))
-----
output:
>>> 4
```

Funkcija ***sqrt()*** vraća kvadratni koren unetog broja.

```
# funkcija kvadratnog korena
print(math.sqrt(2))
-----
output:
>>> 1.4142135623730951
```

Funkcija ***isqrt()*** vraća kvadratni koren broja, i to najbliži ceo broj manji od rezultata.

```
# funkcija celobrojnog kvadratnog korena
print(math.isqrt(2))
-----
output:
>>> 1
```

Funkcija `pow()` vraća stepen prvog argumenta na stepen drugog argumenta.

```
# funkcija stepenovanja
print(math.pow(2,6))
-----
output:
>>> 64
```

Eksponencijalna funkcija `exp()` i logaritamska funkcija sa proizvoljnom bazom `log()`:

```
# eksponencijalna funkcija
print(math.exp(5))
# logaritamska funkcija
print(math.log(100,10))
```

Broj `pi` i broj `e` se mogu uvesti iz math paketa

```
from math import pi
from math import e
```

## PRIMER ZA RAD SA PAKETOM MATH

*Sledi primer za rad sa paketom math i trigonometrijskim funkcijama.*

### Primer (5 minuta):

Napraviti funkciju `trig('f', 'stepeni')`, koja vraća sin, cos, tan, i cot.

f može biti **s**, **c**, **t**, i **ct**, a za ostale unose vraća grešku.

Koristiti math paket. **Napomena:** math trigonometrijske funkcije za ulaz uzimaju vrednost u radijanima.

```
import math

def trig(f, degrees):
    if f == 's':
        return math.sin(math.radians(degrees))
    elif f == 'c':
        return math.cos(math.radians(degrees))
    elif f == 't':
        return math.tan(math.radians(degrees))
    elif f == 'ct':
        return 1/math.tan(math.radians(degrees))
```

```
else:
    print('Treba uneti s, c, t, ili ct')
```

Trigonometry Ratio Table								
Angle (in Degrees)	0	30	45	60	90	180	270	360
sin	0	$\frac{1}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{2}$	1	0	-1	0
cos	1	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$	0	-1	0	1
tan	0	$-\frac{1}{\sqrt{3}}$	1	$\sqrt{3}$	Not Defined	0	Not Defined	1
cot	Not Defined	$\sqrt{3}$	1	$-\frac{1}{\sqrt{3}}$	0	Not Defined	0	Not Defined
sec	Not Defined	2	$-\sqrt{2}$	$-\frac{1}{\sqrt{3}}$	1	Not Defined	-1	Not Defined
csc	1	$-\frac{2}{\sqrt{3}}$	2	Not Defined	-1	Not Defined	1	

Slika 3.1.1 Primer zadatka sa trigonometrijskim funkcijama. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## 3.2 Rad sa datetime paketom

### FUNKCIJE PARSIRANJA DATUMA

*Često je potrebno da se string parsira kao datum ili vreme, ili da se datum ili vreme parsira kao datetime objekat.*

Pri radu sa datumima i vremenskim zapisima poželjno je koristiti paket datetime. Paket sadrži klase kreirane specifično za rad sa datumima, pravilnim predstavljanjem i konverzijom datuma i vremena, kao i uvid u vremenske zone.

Često je potrebno da se string parsira kao datum ili vreme, ili da se datum ili vreme parsira kao datetime objekat.

#### Funkcija parsiranja strftime()

Funkcija parsiranja *datetime.strptime(str, directives)* konvertuje string u kome je napisan datum u datetime objekat. Prvi argument jeste sam string, a drugi jeste string sa direktivama koje opisuju format datuma.

```
from datetime import datetime

datum1 = 'November 13, 2020'
datum1_dt_objekat = datetime.strptime(datum1, '%B %d, %Y')
print(datum1_dt_objekat)
```

#### Direktive formata datuma

Python datetime paket podržava veliki broj formata datuma. Ceo spisak direktiva je moguće naći na <http://strftime.org/>, dok su česti primeri dati u nastavku

Direktiva	Objašnjenje	Primer
<hr/>		
%a	Dan, kraci format.	Mon
%A	Dan, cela rec.	Monday
%w	Radni broj dana u nedelji (pon. je 0). 0	
%d	Dan u mesecu.	30
%b	Mesec, kraci format.	Sep
%B	Mesec, cela rec.	September
%y	Godina, poslednje dve cifre.	21
%Y	Godina, celo broj.	2021

## FUNKCIJE PARSIRANJA VREMENA

*Parsiranje vremena je, kao i za datume, moguće kroz funkciju datetime.strptime().*

### Parsiranje vremena

Funkcija `datetime.strptime()` može parsirati i vreme, a ne samo datume.

```
from datetime import datetime

datum1 = 'November 13, 2020'
datum1_dt_objekat = datetime.strptime(datum1, '%B %d, %Y')
print(datum1_dt_objekat)
print(type(datum1_dt_objekat))

vremel1 = '21:52:20'
vremel1_dt_objekat = datetime.strptime(vremel1, '%H:%M:%S')
print(vremel1_dt_objekat)
print(type(vremel1_dt_objekat))

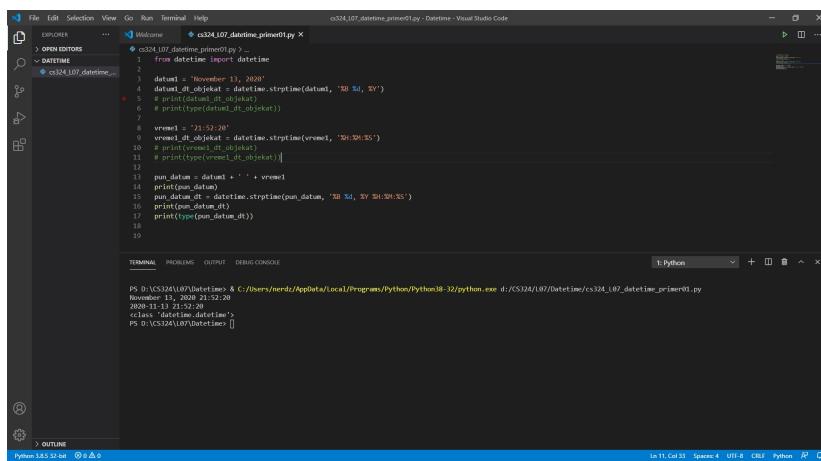
pun_datum = datum1 + ' ' + vremel1
print(pun_datum)
pun_datum_dt = datetime.strptime(pun_datum, '%B %d, %Y %H:%M:%S')
print(pun_datum_dt)
print(type(pun_datum_dt))
```

### Direktive za parsiranje vremena

Kao i za datume, postoje direktive za vreme.

Direktiva	Objašnjenje	Primer
<hr/>		
%H	Sat (24h) sa vodecom nulom.	07
%-H	Sat (24h) bez vodeće nule.	7
%I	Sat (12h) sa vodecom nulom.	07
%-I	Sat (12h) bez vodeće nule.	7

%p	Pre podne ili posle podne (AM/PM).	AM
%M	Minuti sa vodecom nulom.	06
%-M	Minuti bez vodece nule.	6
%S	Sekunde sa vodecom nulom.	09
%-S	Sekunde bez vodece nule.	9
%f	Mikrosekunde	000000



Slika 3.2.1 Parsiranje stringova u datum i vreme. [Izvor: Autor]

## PARSIRANJE IZ DATUMA U STRING

*Funkcija `datetime.strptime()` može parsirati datum i vreme u string.*

### Parsiranje datuma u string

Funkcija `datetime.strptime()` može parsirati datum i vreme u string.

Na ovaj način moguće je štampati string regularno, i u format koji je definisan direktivama, ali od strane korisnika.

#### Preporuka:

**Koristiti datetime formate unutar programa, a tek pri štampiparsovati u string.**

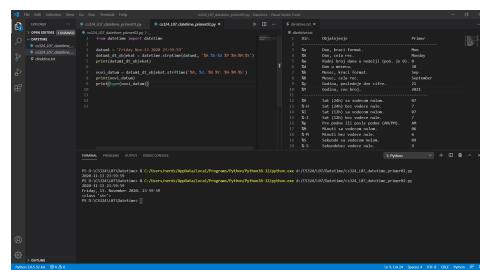
```

from datetime import datetime

datum1 = 'Friday Nov-13 2020 23:59:59'
datum1_dt_objekat = datetime.strptime(datum1, '%A %b-%d %Y %H:%M:%S')
print(datum1_dt_objekat)

novi_datum = datum1_dt_objekat.strftime('%A, %d. %B %Y. %H-%M-%S')
print(novi_datum)
print(type(novi_datum))

```



Slika 3.2.2 Parsiranje datum i vreme u string. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 4

# Regularni izrazi u Python-u – Paket re

## UVOD U REGULARNE IZRAZE

*Pri korišćenju regularnih izraza pretraga teksta ili provera korektnog unosa teksta predstavlja olakšicu pri programiranju.*

Pri radu sa tekstualnim podacima često postoji potreba da se izvuku pojedini podaci iz samog teksta.

Češće, potrebno je izvući određen deo teksta koji zadovoljava određen uslov, a koji nije direktno pretraživan.

Umesto direktnog traženja dela teksta u dokumentu, poželjno je koristiti regularne izraze (en. **Regular expressions, Regex**).

Regularni izraz predstavlja niz karaktera koji definišu šablon pretraživanja.

Pri korišćenju regularnih izraza pretraga teksta ili provera korektnog unosa teksta predstavlja olakšicu pri programiranju.

Validacija unosa korektne email adrese ili korektne šifre koja sadrži minimalan broj karaktera ostvarivo je pomoću **regex** izraza.

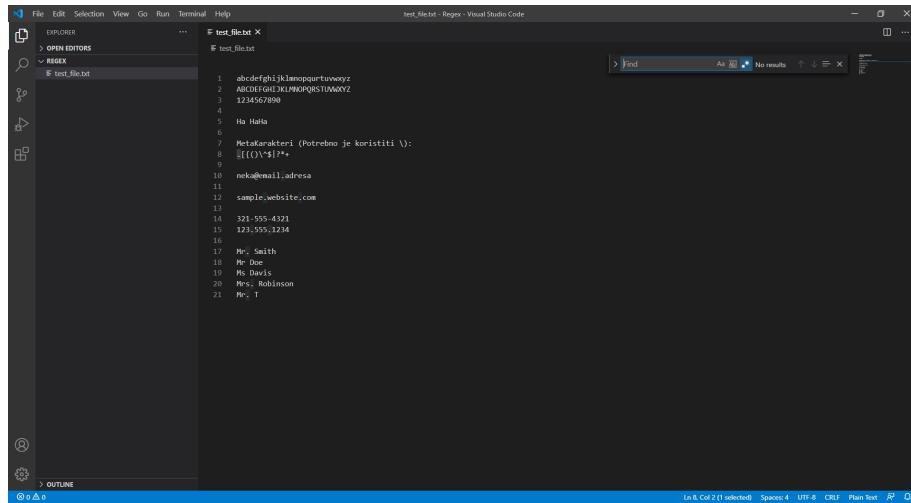
**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PRETRAGA KORIŠĆENJEM REGULARNIH IZRAZA

*Svaki bolji tekstualni editor ima mogućnost korišćenje regularnih izraza.*

### **Pretraga pomoću regularnih izraza u VSCode**

Prilikom pretrage (CTRL+F) potrebno je štiklirati **.\*** da bi se pretraga vršila sa regularnim izrazima. Moguće je pritisnuti i ALT+R.



Slika 4.1.1 Pretraga pomoću regularnih izraza. [Izvor: Autor]

Unutar polja za pretragu moguće je uneti regularni izraz koji se traži, korišćenjem ključnih karaktera u nastavku slajda.

```

.      - Any Character Except New Line
\d     - Digit (0-9)
\D     - Not a Digit (0-9)
\w     - Word Character (a-z, A-Z, 0-9, _)
\W     - Not a Word Character
\s     - Whitespace (space, tab, newline)
\S     - Not Whitespace (space, tab, newline)

\b     - Word Boundary
\B     - Not a Word Boundary
^     - Beginning of a String
$     - End of a String

[]    - Matches Characters in brackets
[^ ]  - Matches Characters NOT in brackets
|     - Either Or
( )  - Group

Quantifiers:
*     - 0 or More
+     - 1 or More
?     - 0 or One
{3}   - Exact Number
{3,4} - Range of Numbers (Minimum, Maximum)

```

## PRIMERI KORIŠĆENJA REGULARNIH IZRAZA

*Dati su primeri korišćenja regularnih izraza za pretragu unutar IDE-a*

Pretraga svih brojeva unutar datoteke.

```
\d
```

Pretraga svih karaktera koji nisu brojevi unutar datoteke.

```
\D
```

Pretraga brojeva sa tri cifre nakon koje sledi bilo koji karakter, pa još tri cifre, pa bilo koji karakter, pa proizvoljan broj cifara

```
\d{3}.\d{3}.\d+
```

Pretraga stringova koji počinju sa karakterima "Ha"

```
\bHa
```

Pretraga veb domena i pod-domena, i to web stranica koja se završavaju sa .com

```
[\w.]+\.com
```

```
abcdefghijklmnpqrstuvwxyz  
ABCDEFGHIJKLMNPQRSTUVWXYZ  
1234567890
```

```
Ha HaHa
```

MetaKarakteri (Potrebno je koristiti \):  
. [{()^\$|?\*+}

```
neka@email.adresa
```

```
sample.website.com
```

```
321-555-4321  
123.555.1234
```

```
Mr. Smith  
Mr Doe  
Ms Davis  
Mrs. Robinson  
Mr. T
```

## DETALJNIJE OBJAŠNJENJE KORIŠĆENJA REGULARNIZH IZRAZA U VISUAL STUDIO CODE

*Sledi autorski video o korišćenju regularnih izraza u Visual Studio Code razvojnom okruženju.*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ 4.1 Rad sa re paketom

### KLASE I FUNKCIJE PAKETA RE

*Korišćenjem klase `re.compile` definiše se pretraga, a metodom `.finditer()` se vrši pretraga nad tekstom.*

#### Sirovi stringovi

Nekada je potrebno štampanje stringova koji će uključiti i specijalne karaktere kao što su *tab*, *novi red*, i sl.

Funkcija za štampanje stringova koji pokazuju "nevidljive" karaktere razlikuje se samo u navođenju karaktera '**r**' ispred samog stringa.

Ovo su tzv.[sirovi stringovi](#) (en.[raw strings](#))

```
# Stampanje obicnog stringa
print('\t String koji ne pokazuje tab')

# Stampanje sirovog stringa
print(r'\t String koji pokazuje tab')
```

Pri radu sa regularnim izrazima upotreba sirovih stringova je česta, te je bitno razlikovati štampanje običnih i sirovih stringova.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

Python koristi [paket re](#) za rad sa regularnim izrazima.

#### Definisanje šablonu

Pri radu sa regularnim izrazima, nakon uvoza paketa [re](#), potrebno je napraviti objekat klase [re.compile\(\)](#), čiji je parametar (sirovi) string ili regularni izraz koji se traži.

#### Pretraga šablonu

Nakon toga, nad objektom se poziva metoda [finditer\(\)](#), čiji je parametar tekst koji se pretražuje. Ova metoda jeste iterabilni tip koji će sadržati sve "pogotke" unutar pretrage.

#### Štampanje šablonu

Nakon pronalaženja šablonu koji je definisan stringom ili regularnim izrazom, moguće je štampanje rezultata. Kako je rezultat prethodne metode iterabilni tip podataka, kroz petlju je moguće stampati sve rezultate.

### PRIMER: IZVLAČENJE EMAIL ADRESA IZ DATOTEKE.

*Regularni izrazi se mogu koristiti za pretragu dela teksta vrlo efikasno.*

### Primer: (15 minuta)

Tekst sa strane sačuvati u dokument `tekst_za_pretragu.txt`

Korišćenjem regularnih izraza (re paketa), napisati program koji će pronaći sve email adrese, štampati ih, i sačuvati u novu datoteku, `email.txt`

```
import re

with open('tekst_za_pretragu.txt', 'r') as f:
    text_uvoz = f.read()

sablon = re.compile(r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-\.]+')

pogodak = sablon.finditer(text_uvoz)

for i in pogodak:
    with open('email.txt', 'a') as fw:
        fw.write(i.group(0) + '\n')
```

### Objašnjenje šablonu:

```
[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-\.]+
```

String će pretražiti sve alfa-numeričke karaktere, donju i srednju crtlu, (email korisničko ime) nakon kojeg sledi karakter `@`, pa ponovo alfa-numeričke karaktere (domen), pa tačku `.`, pa, pa ponovo alfa-numeričke karaktere (domen najvišeg nivoa) .

```
-----
Tekst za pretragu
-----
Dave Martin
615-555-7164
173 Main St., Springfield RI 55924
davemartin@bogusemail.com

Charles Harris
800-555-5669
969 High St., Atlantis VA 34075
charlesharris@bogusemail.com

Eric Williams
560-555-5153
806 1st St., Faketown AK 86847
laurawilliams@bogusemail.com

Corey Jefferson
900-555-9340
826 Elm St., Epicburg NE 10671
coreyjefferson@bogusemail.com

Jennifer Martin-White
714-555-7405
```

212 Cedar St., Sunnydale CT 74983  
jenniferwhite@bogusemail.com

Erick Davis  
800-555-6771  
519 Washington St., Olympus TN 32425  
tomdavis@bogusemail.com

Neil Patterson  
783-555-4799  
625 Oak St., Dawnstar IL 61914  
neilpatterson@bogusemail.com

Laura Jefferson  
516-555-4615  
890 Main St., Pythonville LA 29947  
laurajefferson@bogusemail.com

Maria Johnson  
127-555-1867  
884 High St., Braavos ME 43597  
mariajohnson@bogusemail.com

Michael Arnold  
608-555-4938  
249 Elm St., Quahog OR 90938  
michaelarnold@bogusemail.com

Michael Smith  
568-555-6051  
619 Park St., Winterfell VA 99000  
michaelsmith@bogusemail.com

Erik Stuart  
292-555-1875  
220 Cedar St., Lakeview NY 87282  
robertstuart@bogusemail.com

Laura Martin  
900-555-3205  
391 High St., Smalltown WY 28362  
lauramartin@bogusemail.com

Barbara Martin  
614-555-1166  
121 Hill St., Braavos UT 92474  
barbaramartin@bogusemail.com

Linda Jackson  
530-555-2676  
433 Elm St., Westworld TX 61967  
lindajackson@bogusemail.com

Eric Miller  
470-555-2750  
838 Main St., Balmora MT 56526  
stevemiller@bogusemail.com

Dave Arnold  
800-555-6089  
732 High St., Valyria KY 97152  
davearnold@bogusemail.com

Jennifer Jacobs  
880-555-8319  
217 High St., Old-town IA 82767  
jenniferjacobs@bogusemail.com

Neil Wilson  
777-555-8378  
191 Main St., Mordor IL 72160  
neilwilson@bogusemail.com

Kurt Jackson  
998-555-7385  
607 Washington St., Blackwater NH 97183  
kurtjackson@bogusemail.com

Mary Jacobs  
800-555-7100  
478 Oak St., Bedrock IA 58176  
maryjacobs@bogusemail.com

Michael White  
903-555-8277  
906 Elm St., Mordor TX 89212  
michaelwhite@bogusemail.com

Jennifer Jenkins  
196-555-5674  
949 Main St., Smalltown SC 96962  
jenniferjenkins@bogusemail.com

Sam Wright  
900-555-5118  
835 Pearl St., Smalltown ND 77737  
samwright@bogusemail.com

### Objašnjenje koda:

Najpre se čita datoteka **tekst\_za\_pretragu.txt** korišćenjem kontekstnog menadžera, i smešta u novu promenljivu.

Pravi sa objekat šablon koji za parametar ima sirovi string koji pretražuje email adresu.

Nakon toga, otvoriće novu datoteku **email.txt**, za pisanje (dodavanja sadržaja) i upisivati samo pogotke.

## PRETRAGA EMAIL ADRESA KORIŠĆENJEM REGULARNIH IZRAZA U PYTHONU

*Sledi video za primer pretrage email adresa iz jedne datoteke putem regularnih izraza, i smeštanje u novu datoteku.*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 5

# Python i JSON datoteke – Paket json

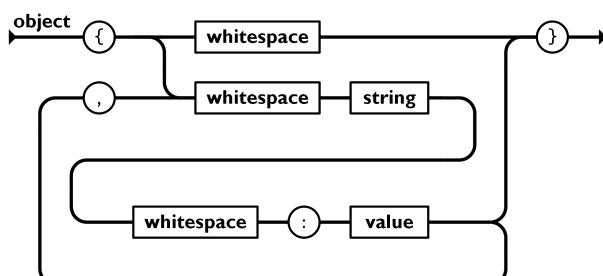
## STRUKTURA JSON DATOTEKE

*JSON datoteka je otvorena struktura podataka koje koriste razni programski jezici za razmenu podataka između aplikacija.*

JSON datoteka (en. **JavaScript Object Notation**) predstavlja tekstualnu datoteku koja služi za razmenu podataka. Laka je za čitanje i pisanje i od strane ljudi i od strane aplikacija, i može se lako generisati i parsirati.

### JSON Objekat

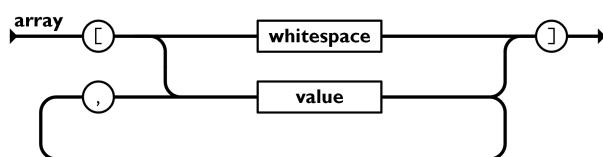
Objekat predstavlja neuređen skup parova ime-vrednost. Objekat počinje sa otvorenom velikom zagradom i završava se sa zatvorenom velikom zagradom. Posle svakog imena sledi dvotačka, a parovi ime-vrednost se razdvajaju zapetom.



Slika 5.1 JSON objekat. [Izvor: json.org]

### JSON niz

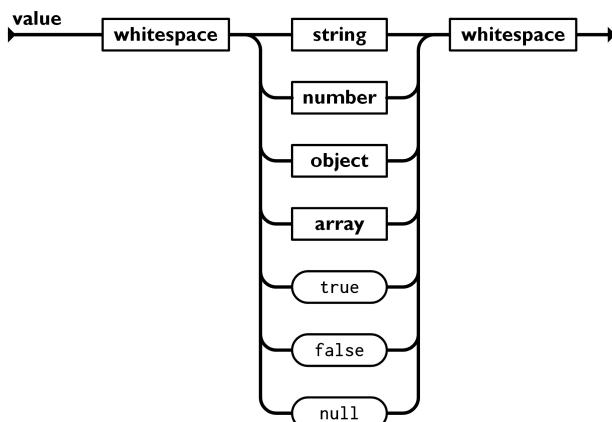
Niz se sastoji od skupa vrednosti. Niz počinje sa otvorenom srednjom zagradom, završava se sa zatvorenom srednjom zagradom, a vrednosti su razdvojene zapetom.



Slika 5.2 JSON niz. [Izvor: json.org]

### JSON vrednost

Vrednost može biti string pod navodnicima, broj, logička vrednost, prazna vrednost, objekat ili niz. Moguće je ugnezdati ovakve strukture.



Slika 5.3 JSON vrednost. [Izvor: json.org]

## PYTHON I PAKET JSON

*Struktura JSON datoteka vrlo je slična Python imenicima, i moguće je konvertovati različite tipove JSON podataka u ugrađene tipove u Python jeziku.*

U Python programskom jeziku, rad sa JSON datotekama zahteva uvoz paketa json.

### Učitavanje JSON datoteka

JSON datoteka se može uvesti u Python na sledeći način:

```
import json

# JSON datoteka se nalazi u promenljivoj json_primer

podaci = json.loads(json_primer)
```

Struktura JSON datoteka vrlo je slična Python imenicima, i moguće je konvertovati različite tipove JSON podataka u ugrađene tipove u Python jeziku.

### Štampanje JSON datoteka

Štampanje promenljive iz Python jezika u JSON datoteku može se postići na sledeći način:

```
import json

novi_podaci = json.dumps(ime_promenljive)
```

JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

Slika 5.4 JSON/Python konverzija. [Izvor: Autor.]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PREUZIMANJE JSON DATOTEKA ZA API

*Čest je slučaj da se JSON datoteke preuzimaju sa stranih veb lokacija da bi se dobilo na funkcionalnosti programa.*

Čest je slučaj da se JSON datoteke preuzimaju sa stranih veb lokacija da bi se dobilo na funkcionalnosti programa.

Nekada se JSON datoteke mogu preuzeti besplatno, a nekada je potrebno prijaviti se na servis ili platiti da bi se dobio API ključ (en. API key)

Primer:

Napisati program koji računa konverziju valuta. Trenutni kurs preuzeti sa <https://exchangeratesapi.io>

Za učitavanje stranice koristiti

```
from urllib.request import urlopen
```

Unutar glavnog programa napisati funkciju za računanje EUR u USD. Štampati rezultat.

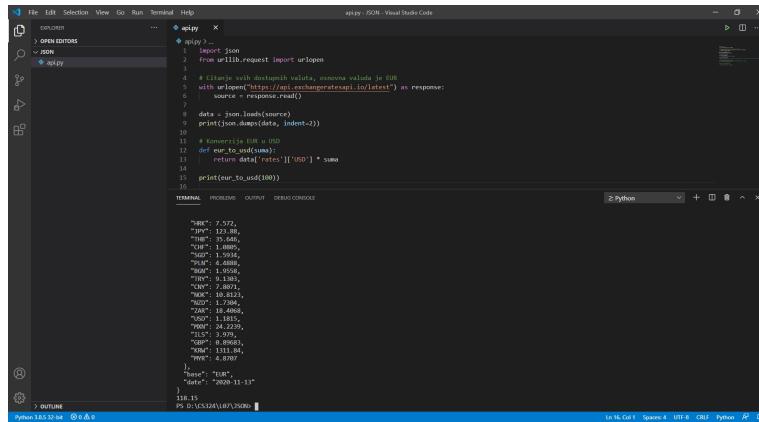
```
import json
from urllib.request import urlopen

# Citanje svih dostupnih valuta, osnovna valuta je EUR
with urlopen("https://api.exchangeratesapi.io/latest") as response:
    source = response.read()

data = json.loads(source)
print(json.dumps(data, indent=2))
```

```
# Konverzija EUR u USD
def eur_to_usd(suma):
    return data['rates']['USD'] * suma

print(eur_to_usd(100))
```



Slika 5.5 Preuzimanje JSON datoteke sa Interneta i korišćenje u programu. [Izvor: Autor]

## PRIMER KONVERZIJE VALUTA

*Sledi video sa konverzijom valuta, učitavanjem JSON datoteke sa Interneta.*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 6

# Python i relacione baze podataka - Paket sqlite

## KRATAK PREGLED SQLITE BAZA PODATAKA

*SQLite je jedna on najčešćih implementacija baza podataka na svetu.*

**SQLite** predstavlja biblioteku koja implementira funkcionalnost SQL baza podataka bez potrebe za serverom. SQLite je **open-access** i može se koristiti za privatne i komercijalne svrhe.

SQLite je jedna on najčešćih implementacija baza podataka na svetu.

Za razliku od ostalih SQL baza podataka, SQLite ne poseduje serverski proces. SQLite čita i upisuje direktno na datoteke na disku, unutar jedne datoteke.

SQLite se ne mora instalirati pre korišćenja. Ne postoji procedura podešavanja i instalacije, jer ne postoji potreba za konfiguracionim datotekama. Zbog toga ne postoji **troubleshooting**.

Glavne osobine SQLite baza podataka jesu sledeće:

- **Serverless** arhitektura (ne postoji potreba za serverskim procesom)
- Baza podataka se nalazi u jednoj datoteci
- Datoteka baze podataka može se koristiti na različitim platformama
- Kompaktna veličina baze podataka
- SQL izjave se kompajliraju u kod virtuelne mašine
- SQLite je otvorena za korišćenje i u privatne i u komercijalne svrhe

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## RAD SA PAKETOM SQLITE3 ZA RAD SA JEDNOSTAVNIM RELACIONIM BAZAMA PODATAKA

*Nije neophodno postavljati lokalne servere ili povezivati se na baze podataka na Internetu pri radu sa SQLite bazama i sqlite3 paketom.*

Paket sqlite3 je deo standardne biblioteke, I nije je potrebno naknadno instalirati. Paket je veoma lak za korišćenje iz razloga što baza podataka može biti i datoteka na disku, ili može postojati samo u RAM memoriji tokom testiranja funkcionalnosti programa. Drugim rečima, nije neophodno postavljati lokalne servere ili povezivati se na baze podataka na Internetu pri radu sa SQLite bazama i SQLite paketom.

### Povezivanje sa bazom podataka

Najpre je potrebno izvršiti povezivanje sa bazom podataka tako što se pravi objekat za povezivanje, kome se prosleđuje ime datoteke baze podataka

```
import sqlite3
conn = sqlite3.connect("ime_baze_podataka.db")
```

### Pravljenje kursora za izvršenje SQL naredbi

Nakon povezivanja sa bazom podataka, potrebno je napraviti kurzor (en. cursor) koji će moći da izvršava naredbe ka povezanoj bazi podataka.

Kada se napravi cursor, moguće je izvršiti SQL naredbe metodom [.execute\(\)](#).

```
c = conn.cursor()
```

### Unos podataka u bazu

Nakon izvršenja SQL naredbi, pravi "unos" podataka u bazu vrši se metodom [.commit\(\)](#), koja je metoda objekta povezivanja.

### Zatvaranje veze sa bazom podataka

Dobra praksa jeste zatvaranje veze sa bazom podataka nakon završenog rada tako što se pozove metoda [.close\(\)](#) nad objekat povezivanja.

```
import sqlite3
conn = sqlite3.connect("ime_baze_podataka.db")
c = conn.cursor()

c.execute("... SQL naredbe ... ")

conn.commit()

conn.close()
```

## PRAVLJENJE BAZE PODATAKA U RAM MEMORIJI RAČUNARA

*Pri testiranju Python programa koji rade sa SQL bazama podataka, moguće je koristiti RAM memoriju računara za smeštanje baze podataka.*

## Pravljenje baze podataka u RAM memoriji računara

Kada se jednom napravi baza podataka kao i tabele, nije moguće ponovo izvršiti te komande, jer se javlja greška

```
sqlite3.OperationalError: table ime_baze_podataka already exists
```

Pri testiranju Python programa koji rade sa SQL bazama podataka, moguće je koristiti RAM memoriju računara za smeštanje baze podataka, i na taj način prilikom svakog izvršenja programa, praviće se nova baza podataka.

```
conn = sqlite3.connect(':memory:')
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## UBACIVANJE I AŽURIRANJE VREDNOSTI BAZE PODATAKA U PYTHON-U

*Nakon povezivanja sa bazom podataka, kroz .execute() metodu moguće je izvršiti bilo koju SQL naredbu.*

### Primer: Baza Podataka student (20 minuta)

Koristiti sqlite3 paket i napraviti bazu podataka student u RAM memoriji računara.

Napraviti tabelu *student* sa kolonama **ime (tekst)**, **prezime (tekst)**, **brojIndeksa (broj)**, **email (text)** i **godinaStudiranja (broj)**.

Napraviti klasu *student(ime,prezime,broj\_indeksa)*, sa metodama *dodati\_email()* koja dodeljuje studentu email adresu kao *ime.prezime.broj\_indeksa@metropolitan.ac.rs*, *stampati\_sve()* koja stampa ime, prezime, broj indeksa i email adresu, kao i metode *dodati\_u\_bazu()*, koja dodaje studenta sa svim atributima u bazu podataka. Prilikom dodavanja svakom studentu je godina studiranja 1.

Napraviti funkciju za ažuriranje godine studiranja u bazu podataka *azurirati\_godinu(broj\_indeksa, azurirana\_godina)*.

Isprobati sve funkcionalnosti.

```
import sqlite3
conn = sqlite3.connect(':memory:')

c = conn.cursor()

c.execute("""CREATE TABLE studenti(
                ime text,
                prezime text,
                brojIndeksa integer,
                email text,
```

```
        godinaStudiranja integer
        )"""
class student:

    def __init__(self, ime, prezime, broj_indeksa):
        self.ime = ime
        self.prezime = prezime
        self.broj_indeksa = broj_indeksa

    def dodati_email(self):
        self.email = '{}.{}.{}@metropolitan.ac.rs'.format(self.ime.lower(),
        self.prezime.lower(), self.broj_indeksa)

    def stampati_sve(self):
        return "Student('{}', '{}', {}, {})".format(self.ime, self.prezime,
        self.broj_indeksa, self.email)

    def dodati_u_bazu(self):
        with conn:
            c.execute("INSERT INTO studenti VALUES (:ime, :prezime, :brojIndeksa,
        :email, :godinaStudiranja",
            {'ime': self.ime, 'prezime': self.prezime, 'brojIndeksa':
        self.broj_indeksa, 'email': self.email, 'godinaStudiranja': 1})

    def azurirati_godinu(broj, azurirana_godina):
        with conn:
            c.execute("UPDATE studenti SET godinaStudiranja = :godinaStudiranja WHERE
        brojIndeksa = :brojIndeksa", {'godinaStudiranja': azurirana_godina, 'brojIndeksa':
        broj})

stud1 = student('Petar', 'Petrovic', 1813)
stud1.dodati_email()
stud1.dodati_u_bazu()
conn.commit()

print('-----')
c.execute("""SELECT * FROM studenti WHERE prezime='Petrovic'""")
print('Prvobitna vrednost :')
print(c.fetchone())

print('Nova vrednost:')
azurirati_godinu(1813,4)
c.execute("""SELECT * FROM studenti WHERE prezime='Petrovic'""")
print(c.fetchone())

conn.close()
```

# PRIMER KREIRANJA BAZE, UBACIVANJA I AŽURIRANJA VREDNOSTI KROZ PYTHON SQLITE3 PAKET

*Sledi video sa primerom kreiranja i ažuriranja vrednosti baze  
poadataka "studenti"*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da  
otvorite LAMS lekciju.**

## ✓ Poglavlje 7

### Pokazna vežba #7

#### ZADATAK #1

*Prvi zadatak odnosi se na regulane izraze*

##### **Zadatak #1 (10 minuta)**

Napisati program koji će putem regularnih izraza proveriti da li je uneta email adresa validna ili ne.

Validna adresa može sadržati veći broj alfa-numeričkih karaktera, srednju i donju crtu za korisničko ime i domen, dok može sadržati samo alfa-numeričke karaktere za top-level domen.

Program stalno postavlja upit, dok se ne unese validna email adresa.

```
import re

pattern = r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-\.]+'

while True:
    email = input('Uneti email: ')
    if re.search(pattern,email):
        print('Validna email adresa!')
        break
    else:
        print('Nije validna email adresa!')
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

#### ZADATAK #2

*Prvi zadatak odnosi se rad sa SQLite bazama podataka*

##### **Zadatak #2 (30 minuta)**

Pomoću Python sqlite3 paketa napraviti bazu podataka *predmeti* koja sadrži kolone **sifraPredmeta (tekst)**, **punolmePredmeta (tekst)**, **godinaStudiranja (ceo broj)** i **FITsmer (text)**.

Napraviti klasu `fit_predmet` sa podrazumevanim konstruktorom. Napraviti metodu za dodavanja atributa **sifra\_predmeta**, **puno\_ime\_predmeta**, **godina\_studiranja** i **fit\_smer**.

Zatim, napraviti metodu koja vraća sve atribute klase u listu.

Izvršiti naredbe za povezivanje kreiranje baze podataka u RAM memoriju računara. Instancirati nekoliko objekata klase `fit_predmet`. Ubaciti vrednosti atributa instanci klase u bazu podataka. Izbrisati jedan red iz baze podataka.

Posle svake izvršene naredbe štampati sadržaj tabele `predmeti`.

```
import sqlite3
conn = sqlite3.connect(':memory:')

c = conn.cursor()

c.execute("""CREATE TABLE predmeti(
                sifraPredmeta text,
                punoImePredmeta text,
                godinaStudiranja integer,
                smer text
            )""")

class fit_predmet:
    def __init__(self):
        pass

    def popuniti_predmet(self, sifra_predmeta, puno_ime_predmeta,
godina_studiranja, fit_smer):
        self.sifra_predmeta = sifra_predmeta
        self.puno_ime_predmeta = puno_ime_predmeta
        self.godina_studiranja = godina_studiranja
        self.fit_smer = fit_smer

    def stampati_predmet(self):
        return [self.sifra_predmeta, self.puno_ime_predmeta,
self.godina_studiranja, self.fit_smer]

predmet01 = fit_predmet()
predmet01.popuniti_predmet('CS324', 'Skripting Jezici', 3, 'IT')
print(predmet01.stampati_predmet())
predmet02 = fit_predmet()
predmet02.popuniti_predmet('CS225', 'Operativni sistemi', 4, 'SI')
print(predmet02.stampati_predmet())

def ubaci_u_bazu(predmet):
    with conn:
        c.execute("INSERT INTO predmeti VALUES (?, ?, ?, ?)",
(predmet.sifra_predmeta, predmet.puno_ime_predmeta,
predmet.godina_studiranja, predmet.fit_smer))

ubaci_u_bazu(predmet01)
ubaci_u_bazu(predmet02)
```

```
conn.commit()  
# provera funkcionalnosti  
print('-----')  
c.execute("""SELECT * FROM predmeti""")  
print(c.fetchall())  
  
c.execute("DELETE FROM predmeti WHERE smer = 'IT'")  
print('-----')  
c.execute("""SELECT * FROM predmeti""")  
print(c.fetchall())  
  
c.close()
```

## ▼ Poglavlje 8

### Individualna vežba #7

#### ZADACI ZA INDIVIDUALNU VEŽBU #7

*Zadaci se rade ukupno 80 minuta*

Individualna vežba #7 odnosi se na rad sa različitim paketima unutar Python standardne biblioteke. Procenjeno trajanje individualnih vežbi iznosi 80 minuta.

##### **Zadatak #1 (35 minuta)**

Napisati program koji ispisuje meni picerije u datoteku.

Napraviti klasu *pizza* sa konstruktorom (*ime, prečnik\_u\_cm, cena*). Instancirati više objekata sa različitim vrednostima.

Napraviti funkciju koja će štampati sve vrednosti u datoteku *picerija.txt*, kao i trenutni datum (koristiti paket *datetime* za dobijanje datuma)

Zatim, unutar klase napraviti novu metodu *cena\_po\_cm2()*, koja će računati cenu po kvadratnom centimetru za konkretnu instancu.

Napraviti funkciju koja će učitati sve vrednosti iz datoteke i izračunati koja pica ima najbolji odnos cene po površini. Funkcija treba da štampa:

```
'Najbolji odnos cena po povrsini ima pizza {ime} od {prečnik} cm, i cene od {cena} dinara.'
```

##### **Zadatak #2 (45 minuta)**

Napisati koji proverava ispravnost unete email adrese i šifre na sledeći način:

Regularnim izrazom proveriti da li je uneta regularna email adresa validna. Validna adresa može sadržati veći broj alfa-numeričkih karaktera, srednju i donju crtu za korisničko ime i domen, dok može sadržati samo alfa-numeričke karaktere za top-level domen.

Program javlja kada se unese validna ili nevalidna adresa. Ako se unese nevalidna adresa, program se završava.

Kada se unese validna adresa, tražiti da se unese šifra korišćenjem:

```
from getpass import getpass
sifra = getpass()
```

Šifra treba biti minimalno dužine 8 karaktera, treba sadržati mala i velika slova, cifre, i specijalne karaktere - \_ # @ .

Program javlja ako šifra nije validna, ali nastavlja sa radom dok se ne unese validna šifra.

Email, kao i sve unete šifre (validne i nevalidne), sačuvati u datoteku **email\_sifre.txt**

## ✓ Poglavlje 9

### Domaći zadatak #7

#### DOMAĆI ZADATAK

*Domaći zadatak #7 se okvirno radi 2.5h*

##### **Zadatak #1**

Dati su podaci u tekstualnoj datoteci na lokaciji:

[https://github.com/CoreyMSchafer/code\\_snippets/blob/master/Regular-Expressions/data.txt](https://github.com/CoreyMSchafer/code_snippets/blob/master/Regular-Expressions/data.txt)

Koristeći regularne izraze i paket `re`, izvući sve adrese koje kreću cifrom koja je jednaka Vašem broju indeksa, i snimiti u tekstualnu datoteku **adrese.txt**

Ukoliko je broj indeksa **1234**, treba izvući adrese:

- **433** Elm St., Westworld TX 61967
- **478** Oak St., Bedrock IA 58176
- ...

##### **Zadatak #2**

Koristeći paket sqlite3, kreirati bazu podataka **predmeti** u memoriji računara, sa kolonama **sifra**, **punolme**, **profesor**, **godinaStudiranja**.

Popuniti tabelu svim predmetima koje ste slušali do sada i napravite funkciju koja pretražuje bazu po predmetnom profesoru i vraća rezultate.

##### **Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije nedelju dana nakon predavanja za 100% poena. Nakon toga poeni se umanjuju za 50%.

##### **Internet studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganja ispita. Domaći zadaci se brane!

Domaći zadatak poslati dr Nemanji Zdravkoviću: nemanja.zdravkovic@metropolitan.ac.rs

Obavezno koristiti uputstvo za izradu domaćeg zadatka.

Uz .doc dokument (koji treba sadržati i screenshot svakog urađenog zadatka kao i komentare za zadatak), poslati i izvorne i dodatne datoteke.

## ✓ Poglavlje 10

### Zaključak

## ZAKLJUČAK

### *Zaključak lekcije #7*

#### **Rezime:**

U ovoj lekciji bilo je reči o paketima za Python 3 programski jezik. Pojedini paketi jesu deo standardne biblioteke, dok se drugi moraju instalirati pomoću PIP paketa.

Bilo je reči o ugrađenim paketima math i datetime, re, json, i sqlite3.

Osim toga, kroz paket PIP moguće je upravljati instaliranim paketima, ažurirati pakete za koje postoji nova verzija, kao i instalirati nove pakete.

Primeri u okviru lekcije kao i zadaci za individualni rad treba da osposobe studente za rad u Python 3.x jeziku pri radu sa različitima paketima.

#### **Literatura:**

- David Beazley, Brian Jones, *Python Cookbook: Recipes for Mastering Python 3*, 3rd edition, O'Reilly Press, 2013.
- Mark Lutz, *Learning Python*, 5th Edition, O'Reilly Press, 2013.
- Andrew Bird, Lau Cher Han, et. al, *The Python Workshop*, Packt Publishing, 2019.
- Al Sweigart, *Automate the boring stuff with Python*, 2nd Edition, No Starch Press, 2020.



## CS324 - SKRIPTING JEZICI

### Python i numeričko programiranje

Lekcija 08

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 08

### *PYTHON I NUMERIČKO PROGRAMIRANJE*

- ✓ Python i numeričko programiranje
- ✓ Poglavlje 1: Uvod u numpy i pandas
- ✓ Poglavlje 2: Nizovi i objekti nizova
- ✓ Poglavlje 3: Rad sa nizovima, matricama
- ✓ Poglavlje 4: Uvod u pandas paket i instalacija
- ✓ Poglavlje 5: pandas: učitavanje i pisanje datoteka
- ✓ Poglavlje 6: Pokazne vežbe #8
- ✓ Poglavlje 7: Individualne vežbe #8
- ✓ Poglavlje 8: Domaći zadatak #8
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ❖ Uvod

# UVOD

### *Uvod u 8. lekciju - paket numpy i pandas*

**Numpy** je jedan od osnovnih paketa za naučno programiranje u Python programskom jeziku. Predstavlja biblioteku koja pruža podršku za višedimenzionalne nizove, matrice i grupu rutina za brzu operaciju nad nizovima. Ove rutine su matematičke, logičke, ali i za sortiranje, menjanje oblika, selekciju, kao i za ulaz i izlaz.

Takođe sadrži rutine za Furijeovu transformaciju koja se često koristi u obradi signala, osnovne rutine linearne algebre i statističkih operacija.

Numpy kao biblioteka predstavlja jednu od osnovnih, ako ne i najosnovniju biblioteku koja treba da se savlada ukoliko želite da se bavite razvojem bilo kakvih aplikacija koje imaju veze sa hardverom, embedded sistema ili sa Internet of Things uređajima.

Zbog toga je bitno da se savladaju osnove numpy paketa, a to je objekat *ndarray*. Ovaj objekat enkapsulira n-dimenzionalne nizove homogenih tipova podataka, uz operacije koje su brže od standardnih Python operacija za rad sa nizovima.

Pored numpy-a, u ovoj lekciji obradiće se i **pandas** biblioteka. Pandas predstavlja paket za obradu podataka i za analizu podataka. Pandas pruža strukture podataka i operacije za manipulaciju numeričkih tabela i vremenskih nizova. Ime pandas je skraćenica za **panel data**, što je izraz za skupove podataka koji se posmatraju u nekom vremenskom periodu.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 1

# Uvod u numpy i pandas

## PAKET NUMPY

*Paket Numpy predstavlja osnovni paket za naučno programiranje u Python programskom jeziku.*



Slika 1.1 NumPy logo. [Izvor: <https://numpy.org>]

Paket Numpy predstavlja osnovni paket za naučno programiranje u Python programskom jeziku. Najosnovniji objekat koji paket NumPy pruža jeste višedimenzionalni niz, različite izvedene objekte (maskirani nizovi i matrice), kao i assortiman rutina za brz rad sa nizovima.

Ove rutine uključuju matematičke i logičke operacije, operacije promene oblika niza, sortiranje, odabir, ulaz-izlaz (en. **input-output**, *I/O*), diskretne Furijeove transformacije, osnovne operacije linearne algebre, osnovne statističke operacije, alate za simulaciju i mnoge druge.

Više informacija o samom paketu na: <https://numpy.org>

Programeri koji su upoznati sa **MATLAB** (**Matrix Laboratory**) programskim jezikom, prepoznaće NumPy paket kao alternativu za rad sa višedimenzionalnim nizovima. Može se reći da je NumPy besplatna zamena za osnovne funkcionalnosti MATLAB programskog jezika, naravno unutar Python jezika.

NumPy predstavlja osnovu za bilo koju vrstu programiranja pri kojoj je potreban rad sa višedimenzionalnim nizovima. Paketi za naučno programiranje (**SciPy**), vizuelizaciju podataka (**matplotlib**), mašinsko učenje (**scikit-learn**), backend programiranje pri razvoju web aplikacija, i mnogi drugi paketi koriste upravo NumPy.

NumPy paket u potpunosti podržava objektno-orientisani pristup. Osnova NumPy paketa jeste klasa ndarray, koja podržava brojne atribute i metode. Mnoge metode jesu preslikane funkcijama u *spoljašnjem prostoru imena* (en. *outer-most namespace*), dozvoljavajući programerima da pišu programe u paradigmi koja im odgovara.

Fleksibilnost koja pruža paket NumPy niz dovela je do toga da je klasa ndarray *de-facto* način za rad sa višedimenzionalnim podacima u Python jeziku.

## INSTALACIJA NUMPY PAKETA

*Nepisano je pravilo da se numpy paket preimenuje u np prilikom uvoza u program.*

Instalaciju NumPy paketa je jednostavna kroz paketni menadžer *pip*. Dovoljno je u konzoli ukucati sledeću komandu:

```
pip install numpy
```



```
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
PS D:\CS324\L08> pip install numpy
Collecting numpy
  Downloading numpy-1.19.4-cp38-cp38-win32.whl (11.0 MB)
    11.0 MB 6.4 MB/s
Installing collected packages: numpy
  Successfully installed numpy-1.19.4
PS D:\CS324\L08> |
```

Slika 1.2 Instalacija NumPy paketa kroz pip. [Izvor: Autor]

Provera (sa verzijom koja je instalirana) se može izvršiti pokretanjem *pip list* komande.

Slika 1.3 Provera instalacije i verzije paketa. [Izvor: Autor]

Uvoz NumPy paketa u imenski prostor programa

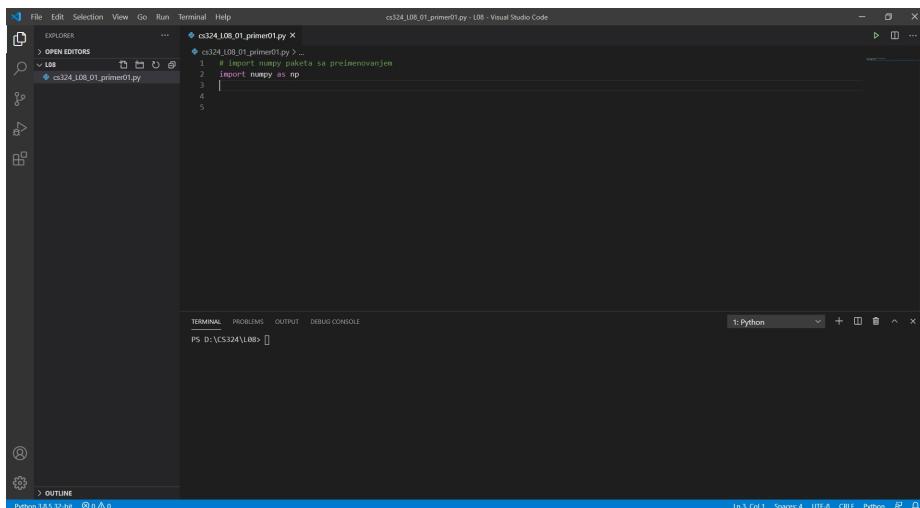
Nakon instalacije paketa NumPy, dovoljno je uvesti paket u glavni program.

```
import numpy
```

Preporučuje se uvoz sa preimenovanjem:

```
# import numpy paketa sa preimenovanjem
import numpy as np
```

Prilikom korišćenja NumPy paketa postalo je nepisano pravilo da se preimenuje u *np*.



Slika 1.4 Uvoz numpy paketa sa preimenovanjem u np paket. [Izvor: Autor]

## INSTALACIJA NUMPY PAKETA - VIDEO

*Sledi autorski video o instalaciji numpy paketa korišćenjem menadžera paketa PIP*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## VEKTORIZACIJA IZVRŠENJA OPERACIJA

*Vektorizacija izvršenja operacija predstavlja odsustvo eksplisitnih petlji i indeksiranja unutar koda.*

Vektorizacija izvršenja operacija predstavlja odsustvo eksplisitnih petlji i indeksiranja unutar koda. Petlje i indeksiranje i dalje postoje, ali se izvršavaju iza kulisa, i to u optimizovanom kodu koji je *pre-kompajlovan* (en. *pre-compiled code*) i napisanom u C jeziku.

Vektorizovan kod ima mnogo prednosti, a ističu se sledeće:

- koncizniji kod koji je lakši za čitanje,
- manji broj linija koda u opštem slučaju znači manji broj grešaka,
- kod više liči na standardnu matematičku notaciju, što omogućava da se pravilnije kodiraju matematički konstrukti,
- Vektorizacija kao rezultat ima kod koji sam po sebi deluje "Python-ovski". Ovo znači da kod ne poseduje brojne neefikasne *for petlje* koje se teško čitaju i tumače.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 2

### Nizovi i objekti nizova

#### KLASA I OBJEKAT NDARRAY

"Srž" NumPy paketa jeste *klasa ndarray* i njen *objekat ndarray*.

Najosnovniji deo NumPy paketa jeste *klas ndarray* i njen *objekat ndarray*. Ovaj objekat enkapsulira n-dimenzionalne nizove sa homogenim tipovima podataka.

##### **Pitanje:**

##### **Šta je homogeni, a šta heterogeni niz?**

Mnoge operacije koje se mogu izvršiti nad *ndarray* objektom izvršavaju se jako brzo jer su napisane u kompajlovanom kodu (en. *compiled code operations*)

Objekat *ndarray* jeste višedimenzionalni kontejner elemenata istog tipa podataka i veličine, i sam objekat je najčešće fiksne veličine.

Broj dimenzija i elemenata unutar niza jeste definisan pomoću oblika (en. *array shape*), koji predstavlja *tuple* ne-negativnih celih brojeva koji specificiraju veličine svake dimenzije.

Tip podataka elemenata niza je specificiran posebnim *data-type* objektom *dtype*.

Kao i kod ostalih iterabilnih objekata u Python jeziku, sadržaju *ndarray* objekta se može pristupiti indeksiranjem, ali i metodama i atributima samog objekta.

Različiti *ndarray* objekti mogu deliti iste podatke, tako da promene u jednom objektu budu vidljive u drugom.

Jedan objekat može služiti da se "pregleda" sadržaj drugog objekta, i podaci na koji taj objekat ukazuje jesu podaci u "osnovnom" (en. *base*) objektu.

#### KREIRANJE NIZA KLASE NDARRAY

*Kreiranje niza klase ndarray je jednostavno, ali treba obratiti pažnju da je parametar koji se prosleđuje lista.*

Kreiranje niza klase ndarray je jednostavno gotovo koliko i kreiranje liste.

```
arr = np.array(<list>)
```

Parametar `<list>` ukazuje da se treba proslediti lista. Ukoliko je u pitanju niz sa više dimenzija, unosi se lista listi, gde je svaka lista jedna dimenzija niza.

### Primer: Jednodimenzionalni i dvodimenzionalni niz (2 minuta)

Pomoću numpy napraviti tri niza. Prvi je niz od 5 elemenata, dok je drugi 2x3 matrica. Treći niz jeste niz stringova različitog broja karaktera.

```
import numpy as np

a = np.array([1, 2, 3, 4, 5])
b = np.array([[1, 2, 3], [4, 5, 6]])
c = np.array(['Ovo', 'je', 'test', 'string', 'za', 'rad', 'sa', 'numpy',
'paketom....'])
```

**Broj elemenata** se može dobiti pozivom metode `.size`

```
print(a.size)
>>> 5
print(b.size)
>>> 6
print(c.size)
>>> 9
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## DIMENZIJE, TIP PODATAKA I VELIČINE ELEMENATA U MEMORIJI

*Pozivanjem različitih atributa numpy niza moguće je dobiti dodatne informacije o samom nizu.*

**Dimenzije** niza se mogu vratiti pozivanjem atributa `.ndim`

```
print(a.ndim)
>>> 1
print(b.ndim)
>>> 2
print(c.ndim)
>>> 1
```

**Oblik** niza predstavlja tuple koji vraća broj elemenata po dimenziji niza, i dobija se pozivanjem atributa `.shape`

```
print(a.shape)
>>> (5, )
print(b.shape)
>>> (2, 3)
```

```
print(c.shape)
>>> (9, )
```

**Tip** niza se može vratiti pozivanjem atributa `.dtype`

```
print(a.dtype)
>>> int32
```

Prilikom kreacije niza moguće je kao dodatni argument napisati i **eksplicitno** koji je tip podataka.

```
a = np.array([1, 2, 3, 4, 5], dtype='int64')
```

```
# tipovi podataka
bool_ - Boolean tip podataka, 1B po elementu
int8 - ceo broj, 1B po elementu (vrednosti -128 do 127)
int32 - ceo broj, 4B po elementu
int64 - ceo broj, 8B po elementu
uint8/16/32/64 - nenegativan ceo broj, 1B/2B/4B/8B po elementu
float16/32/64 - razlomljeni broj, 2B/4B/8B po elementu
complex64/128 - kompleksni broj, 8B/16B po elementu
U_ - string, gde _ označava broj karaktera najvećeg stringa
```

**Veličina** elemenata niza se može dobiti pozivom metode `.itemsize`, koja vraća veličinu u bajtovima.

```
print(a.itemsize)
>>> 8
print(b.itemsize)
>>> 4
print(c.itemsize)
>>> 48
```

**Ukupna veličina niza** u memoriji može se dobiti množenjem broja elemenata i veličine elemenata

```
print(a.size * a.itemsize)
>>> 40
```

## RAZLIKE IZMEĐU NDARRAY, LISTA I UGRAĐENIH NIZOVA

*Većina softvera za matematičko i naučno izračunavanje koja je bazirana na Python programskom jeziku koristi NumPy nizove.*

Postoje nekoliko bitnih razlika između Numpy nizova (`ndarray` objekata) i standardnih Python iterativnih tipova, kao što su liste i "obični", ugrađeni `array` nizovi.

- Numpy nizovi imaju fiksnu veličinu pri pravljenju niza. Python liste je dinamički tip podataka (može proizvoljno da raste broj elemenata). Promena veličine ndarray niza napraviće novi niz, a stari će obrisati.
- Elementi ndarray niza moraju biti istog tipa, i zbog toga svaki element zauzima istu količinu memorije.
- Nad ndarray nizovima moguće je izvršiti veliki broj matematičkih operacija nad velikom količinom podataka istovremeno. Ovakav pristup izvršenja operacija je efikasniji i brži u odnosu na ugrađene operacije u Python interpretéruru.
- Sve veći broj naučnih i matematičkih paketa za Python programski jezik koriste ndarray nizove. Iako se unos podataka obavlja preko standardnih tipova podataka, oni se konvertuju u NumPy nizove pre procesiranja. Ovo znači da većina softvera za matematičko i naučno izračunavanje koja je bazirana na Python programskom jeziku koristi NumPy nizove.

**Napomena:**

***Moguće je imati niz objekata (uključujući i NumPy nizove), što omogućuje da niz sadrži elemente različitih veličina.***

## PRIMER: MNOŽENJE ELEMENATA DVA NIZA ISTE DUŽINE

*Korišćenjem Numpy paketa za rad sa nizovima pojednostavljuje pisanje koda.*

### Primer: Proizvod elemenata lista (5 minuta)

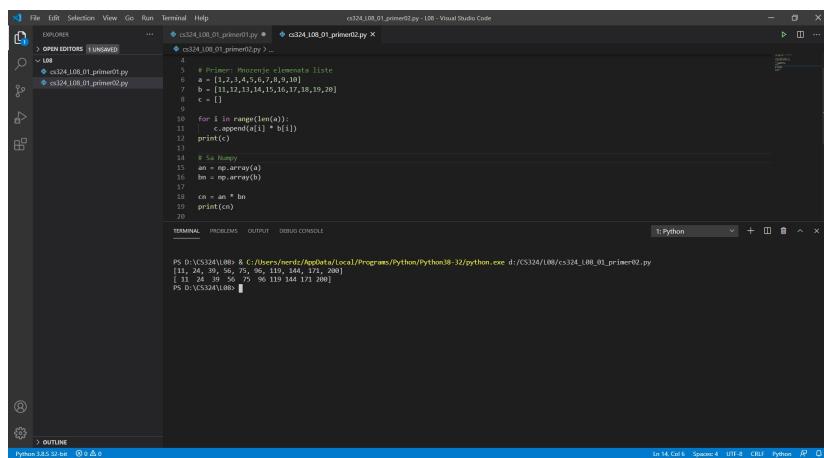
Data je lista **a** od 10 elemenata (brojevi od 1 do 10), i lista **b** od 10 elemenata (brojevi od 11 do 20). Napraviti listu **c** u kojoj je svaki element proizvod elemenata **a** i **b** sa istim indeksom.

```
# Primer: Mnozenje elemenata liste
a = [1,2,3,4,5,6,7,8,9,10]
b = [11,12,13,14,15,16,17,18,19,20]
c = []

for i in range(len(a)):
    c.append(a[i] * b[i])
print(c)
```

Korišćenjem paketa NumPy ista operacija se može mnogo lakše izvršiti:

```
# Preko Numpy
an = np.array(a)
bn = np.array(b)
cn = an * bn
print(cn)
```



The screenshot shows a Visual Studio Code interface with two tabs open: 'cs324\_108\_01\_prime01.py' and 'cs324\_108\_01\_prime02.py'. The 'prime02.py' tab contains the following Python code:

```
1 # Prime: mnozenje elemenata lista
2 a = [1,2,3,4,5,6,7,8,9,10]
3 b = [1,2,3,4,5,6,7,8,9,10]
4 c = []
5
6 for i in range(len(a)):
7     c.append(a[i] * b[i])
8
9 print(c)
10
11 # Sa nizom
12 a = np.array(a)
13 b = np.array(b)
14
15 c = a * b
16
17 cn = a * b
18 print(cn)
19
20
```

The terminal window shows the output of the script:

```
D:\CS324\108_01>python prime02.py
[ 1  2  3  4  5  6  7  8  9 10]
[ 1  2  3  4  5  6  7  8  9 10]
[ 1  2  3  4  5  6  7  8  9 10]
[ 1  2  3  4  5  6  7  8  9 10]
[ 1  2  3  4  5  6  7  8  9 10]
[ 1  2  3  4  5  6  7  8  9 10]
[ 1  2  3  4  5  6  7  8  9 10]
[ 1  2  3  4  5  6  7  8  9 10]
[ 1  2  3  4  5  6  7  8  9 10]
[ 1  2  3  4  5  6  7  8  9 10]
```

Slika 2.1 Množenje elemenata jednodimenzionog niza. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 3

# Rad sa nizovima, matricama

## INICIJALIZACIJA NIZOVA SA NULAMA I JEDINICAMA

*U okviru numpy paketa moguće je izvršiti inicijalizaciju nizova koji imaju sve nule ili sve jedinice.*

Specijalni nizovi i specijalne matrice predstavljaju nizove i matrice kojima su (početne) vrednosti inicijalizovane pozivom funkcije koja nije `nd.array`.

### **Nizovi i matrice sa nulama**

Pozivom funkcije `zeros()` vraća se numpy objekat koji ima sve elemente jednake nulama.

```
import numpy as np
n = zeros((shape), dtype='float')
```

Ukoliko se u parametar `shape` unese samo jedna vrednost, vratiće niz sa toliko elemenata. Ukoliko se naznači oblik (kao kod `shape` atributa - tuple sa brojem elemenata po dimenziji) napraviće se takav niz ili matrica.

Drugi argument je tip podataka i podrazumevano je `float`, dok se može eksplicitno namestiti da je u pitanju drugi tip podataka.

### **Primer: Nula matrica (2 minuta)**

Napraviti niz od 5 elemenata i matricu  $2 \times 10$  sa svim nulama.

```
import numpy as np
# nizovi i matrice sa sve nulama
za = np.zeros(5)
zb = np.zeros((2,10))
print(za)
print(zb)
```

Nizovi i matrice sa jedinicama

Pozivom funkcije `ones()` vraća se numpy objekat koji ima sve elemente jednake jedinicama.

```
import numpy as np
n = ones((shape), dtype='float')
```

Kao i kod nizova sa nulama, prvi parametar označava oblik niza, a drugo označava tip podataka.

### Primer: Matrice sa jedinicama (2 minuta)

Napraviti matricu 3x2 sa sve jedinicama. Pomnožiti celu matricu sa brojem 3. Štampati rezultat.

```
import numpy as np

o32 = np.ones((3,2))
o32 *= 3
print(o32)
```

Kada se nad svim elementima numpy nizova vrše aritmetičke operacije, moguće je direktno interagovati sa samim nizom, tj. nije neophodno praviti numpy niz sa jednim elementom koji bi bio drugi sabirak/množilac.

## FULL NIZOVI, NIZOVI SA OBLIKOM DRUGOG NIZA, JEDINIČNA MATRICA

*Takođe je moguće je izvršiti inicijalizaciju nizova koji imaju sve iste vrednosti, ili oblik drugog niza.*

### Nizovi sa podešenim početnim vrednostima - full nizovi

Pozivom funkcije `full()` pravi se numpy niz koji ima željeni oblik (prvi parametar), određenu vrednost (drugi parametar) i tip podataka.

```
import numpy as np
n = np.full((shape), value, dtype)
```

### Primer: Full niz stringova (2 minuta)

Napraviti numpy niz 2x2 kojem su sve vrednosti 'CS324 - FIT' a tip podataka je '<U5'

```
import numpy as np

fa = np.full((2,2), 'CS324 - FIT', dtype='<U5')
print(fa)
```

### Nizovi sa oblikom drugog niza

Nizovi sa nulama, jedinicama i puni nizovi se mogu napraviti da imaju oblik već postojećeg niza tako što se pozivaju funkcije sa dodatkom `_like` na kraju:

```
import numpy as np

#postojeci niz a = np.array(...)
```

```
a_zeros = np.zeros_like(a)
a_ones = np.ones_like(a)
a_full = np.full_like(a, value)
```

### Primer: Nizovi sa oblicima postojećih nizova (4 minuta)

Napraviti niz 2x5 (dva reda, 5 kolona) koji sadrži cele brojeve. Napraviti nizove sa jedinicama, nulama, i pun niz sa jednom od vrednosti prvobitnog niza.

```
import numpy as np

niz = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
print(niz)

niz_zeros = np.zeros_like(niz)
print(niz_zeros)

niz_ones = np.ones_like(niz)
print(niz_ones)

niz_full = np.full_like(niz, 10)
print(niz_full)
```

### Jedinična matrica

Moguće je napraviti jediničnu matricu (en.[identity matrix](#)) pozivom funkcije [identity\(\)](#), gde je parametar size veličina ove kvadratne matrice.

```
import numpy as np
n = np.identity(size)
```

## PRIMERI SA NIZOVIMA SA SPECIJALNIM VREDNOSTIMA

*Slede autorski video klipovi sa primerima sa specijalnim nizovima.*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## NASUMIČNI ELEMENTI NIZA

*Paket numpy poseduje svoj modul random koji je specijalizovan za rad sa nizovima i nasumičnim vrednostima.*

Paket numpy poseduje svoj modul random koji je specijalizovan za rad sa nizovima, i može se iskoristiti kada je potrebno da elementi niza budu nasumični brojevi.

### Nasumični razlomljeni brojevi

Nasumični razlomljeni brojevi se mogu pozvati funkcijom `random.rand()`

```
import numpy as np
n = np.random.rand(d0,d1,...)
```

Kao parametre ova funkcija uzima samo cele brojeve, gde je prvi broj elemenata u prvoj dimenziji, drugi broj za drugu dimenziju i sl. Interval koji se koristi jeste [0,1).

Ukoliko se kao parametar prosleđuje oblik postojećeg niza, onda treba koristiti funkciju `random.random_sample()`

```
import numpy as np
n = np.random.random_sample(size)
```

### Nasumični razlomljeni brojevi u datom intervalu

Niz sa nasumičnim brojevima u datom intervalu (sa uniformnom raspodelom) mogu se dobiti pozivom funkcije `random.uniform()`

```
import numpy as np
n = np.random.uniform(low, high, shape)
```

Prvi parametar je donja granica, drugi parametar je gornja granica.

### Nasumični celi brojevi u datom intervalu

Niz sa nasumičnim celim brojevima (sa diskretnom uniformnom raspodelom) mogu se dobiti pozivom funkcije `random.randint()`

```
import numpy as np
n = np.random.randint(low, high, shape)
```

Slede primeri poziva nizova sa različitim nasumičnim brojevima.

```
import numpy as np

# nasumicni brojevi u intervalu [0,1) sa uniformnom raspodelom
a = np.random.rand(2,4)
print(a)

# nasumicni brojevi u intervalu [0,1) sa uniformnom raspodelom,
# prima array.shape kao parametar
b = np.random.random_sample(a.shape)
print(b)

# nasumicni celi brojevi u datom intervalu sa diskretnom uniformnom raspodelom
c = np.random.randint(-10,10,b.shape)
print(c)
```

```
# nasumicni brojevi sa uniformnom raspodelom u odredjenom intervalu
d = np.random.uniform(-5,5,c.shape)
print(d)
```

## NIZOVI I LINEARNA ALGEBRA

*Numpy paket sadrži brojne funkcije za rad sa nizovima i matricama iz oblasti linearne algebre.*

Numpy paket sadrži brojne funkcije za rad sa nizovima i matricama iz oblasti linearne algebre, kroz modul [linalg](#)

### Inverzna matrica

Nalaženje inverzne matrice moguće je pozivom funkcije [linalg.inv\(\)](#)

```
import numpy as np
n_inv = np.linalg.inv(n)
```

### Determinanta matrice

Nalaženje determinante kvadratne matrice moguće je pozivom funkcije [linalg.det\(\)](#)

```
import numpy as np
n_det = np.linalg.det(n)
```

### Sopstveni vektori i sopstvene vrednosti

Nalaženje sopstvenih vektora kvadratne matrice i sopstvenih vrednosti matrice moguće je pozivom funkcija [linalg.eig\(\)](#) i [linalg.eigvals\(\)](#).

```
import numpy as np
n_eigvec = np.linalg.eig(n)
n_eigval = np.linalg.eigvals(n)
```

```
import numpy as np

# linearna algebra - mnozenje matrica
a = np.random.randint(-10, 10, (5, 2))
print(a)

b = np.random.randint(-10, 10, (2, 5))
print(b)

print(np.matmul(a,b))

# inverzna matrica
c = np.random.randint(-10, 10, (5, 5))
print(c)
```

```
c = np.linalg.inv(c)
print(c)

# determinanta
d = np.random.randint(-5, 5, (4, 4))
print(d)
d = np.linalg.det(d)
print(d)
```

## NUMPY NIZOVI I RAD SA DATOTEKAMA

*Kroz numpy moguće je raditi i sa datotekama.*

### Učitavanje iz datoteke

Učitavanje iz datoteke je moguće pozivom funkcije [genfromtxt\(\)](#)

```
import numpy as np
filedata = np.genfromtxt('path_to_file', delimiter, dtype)
```

Prvi argument jeste ime (i putanja) datoteke, drugi parametar jeste [delimiter](#), a opcionalno se može staviti koji je tip podataka u pitanju.

### Primer: (4 minuta)

Zapamtiti tekst u nastavku zadatka u datoteku "podaci.txt". Zatim, kroz numpy funkciju učitati tekst. Proveriti koji su elementi učitanog niza veći od 15.

Test podaci:

```
1,3,5,7,9,11,13,15,17,19
2,4,6,8,10,12,14,16,18,20
3,7,11,15,19,23,27,31,35,39
```

Rešenje:

```
import numpy as np

filedata = np.genfromtxt('podaci.txt', delimiter=',', dtype='int32')
print(filedata)

print(filedata > 15)
```

### Upis u datoteku

Upis u datoteku moguće je pozivom funkcije [savetxt\(\)](#)

```
import numpy as np
savetxt('path_to_file', source_array, fmt, delimiter, dtype)
```

Prvi argument jeste ime (i putanja) datoteke, drugi parametar jeste niz koji se upisuje u datoteku, parametar *fmt* jeste format svakog elementa, sledi delimiter, a opcionalno se može staviti koji je tip podataka u pitanju.

**Primer: (4 minuta)**

Uvesti tekst iz "podaci.txt". Zatim, napraviti novi niz istog oblika, a svaki element ima vrednost 100. Sabrati ta dva niza u sačuvati kao datoteku "novi\_podaci.txt" sa zapetom kao delimiter.

```
import numpy as np

filedata = np.genfromtxt('podaci.txt', delimiter=',', dtype='int32')

a = np.full_like(filedata, 100)
a += filedata

np.savetxt('novi_podaci.txt', a, fmt='%d', delimiter=',')
```

## ▼ Poglavlje 4

### Uvod u pandas paket i instalacija

#### PAKET PANDAS

*Pantad je osnovni paket za rad sa manipulacijom i analizom velike količine podataka*



Slika 4.1 Paket pandas. Izvor: [<https://pandas.pydata.org>]

Paket pandas predstavlja osnovni paket za rad sa manipulacijom podataka i analizom podataka. Pandas nudi rad sa strukturama podataka i operacijama nad numeričkim i alfanumeričkim tabelama, i vremenskim nizovima (en. **time series**).

Ime pandas je od "panel data", što predstavlja termin za skupove podataka koji se pregledaju u različitim vremenskim periodima.

Osnovni deo pandas paketa jeste objekat DataFrame koji služi za manipulaciju podataka. Pandas takođe sadrži:

- Alate za čitanje i pisanje podataka u različitima formatima,
- Alate za podešavanje podataka i obradu podataka koji nedostaju,
- Alate za promenu oblika predstavljanja podataka,
- Mogućnost naprednog indeksiranja i pravljenja podskupova podataka ukoliko se radi o velikom skupu podataka,
- Mogućnost dodavanje redova i kolona unutar strukture podataka,
- Grupisanje i sjednjavanje struktura podataka,
- Filtriranje podataka

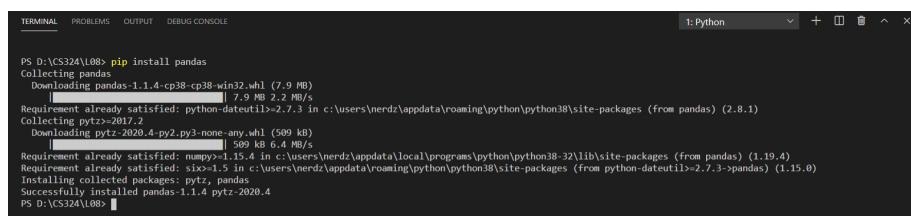
**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

# INSTALACIJA PANDAS PAKETA I JUPYTER SVEZAKA

*Za bolji pregled datoteka, preporučuje se korišćenje Jupyter svezaka.*

Instalaciju pandas paketa je jednostavna kroz paketni menadžer [pip](#). Dovoljno je u konzoli ukucati sledeću komandu:

```
pip install pandas
```



Slika 4.2 Instalacija pandas paketa kroz pip. [Izvor: Autor]

Nakon instalacije, potebno je uvesti paket u trenutni imenski prostor komandom:

```
# import pandas paketa sa preimenovanjem
import pandas as pd
```

Nepisano je pravilo da se pandas paket preimenuje u *pd*.

Nakon instalacije pandas paketa, poželjno je instalirati i [jupyter](#) ekstenziju, zbog boljeg pregleda tabelarnih podataka.

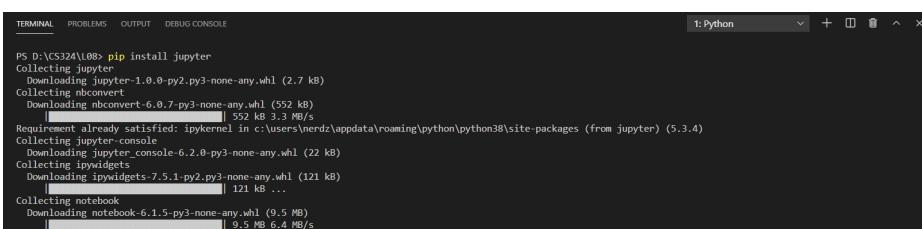
Pored konzolnog i klasičnog IDE prikaza, Python poseduje mogućnost da se kod izvršava kroz interaktivni IDE - [Jupyter](#).

Korišćenjem tzv. jupyter svezaka, moguće je izvršavati deo koda, postavljati tekst koji nije deo koda, izvršiti vizuelizaciju podataka, ali i izvršiti integraciju sa drugim programskim jezicima kap što je R programski jezik.

Jupyter se uglavnom koristi za rad sa velikim podacima, numeričku simulaciju, statističkom modelovanje, i mašinsko učenje.

Jupyter se može instalirati kao ekstenzija unutar Visual Studio Code IDE-a:

```
pip install jupyter
```



Slika 4.3 Instalacija jupyter kroz pip. [Izvor: Autor]

Nakon instalacije, potrebno je pokrenuti jupyter serverski proces pozivom sa Ctrl+Shift+P

Jupyter: Create New Jupyter Notebook

## DODATNI PAKETI PRI RADU SA PANDAS PAKETOM

*Nekada je potrebno koristiti dodatni paket pri radu sa nekim od formata koje pandas podržava.*

Pri radu sa dodatnim formatima nekada je neophodno instalirati dodatne pakete zbog zavisnosti.

Na primer, prilikom rada sa tabelama koje se mogu pokrenuti u Microsoft Excel-u ili nekom sličnom programu (datoteke sa ekstenzijom **.xls** ili **.xlsx**), potrebno je ubaciti dodatni paket ili se javlja sledeća greška:

```
Traceback (most recent call last):
  File "d:\CS324\108\Pandas\cs324_108_pandas01.py", line 6, in <module>
    df2 = pd.read_excel('FIT_IT_presmeti.xlsx')
          return func(*args, **kwargs)
    return func(*args, **kwargs)
File "C:\Users\verdz\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\io\excel\_base.py", line 304, in read_excel
    io = ExcelFile(io, engine=engine)
File "C:\Users\verdz\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\io\excel\_base.py", line 867, in __init__
    self._engine = self._get_engine()
File "C:\Users\verdz\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\io\excel\_xlrd.py", line 21, in __init__
    import_optional_dependency("xlrd", extra_err=msg)
File "C:\Users\verdz\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\compat\optional.py", line 110, in import_optional_dependency
    raise ImportError(msg) from None
ImportError: Missing optional dependency 'xlrd'. Install xlrd >= 1.0.0 for Excel support. Use pip or conda to install xlrd.
```

Slika 4.4 Greška pri radu sa .xls datotekama. [Izvor: Autor]

Neophodno je instalirati paket **xlrd**, koji daje dodatnu podršku za rad sa ovakvim tabelama.

pip install xlrd

Slika 4.5 Paket xlrd. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 5

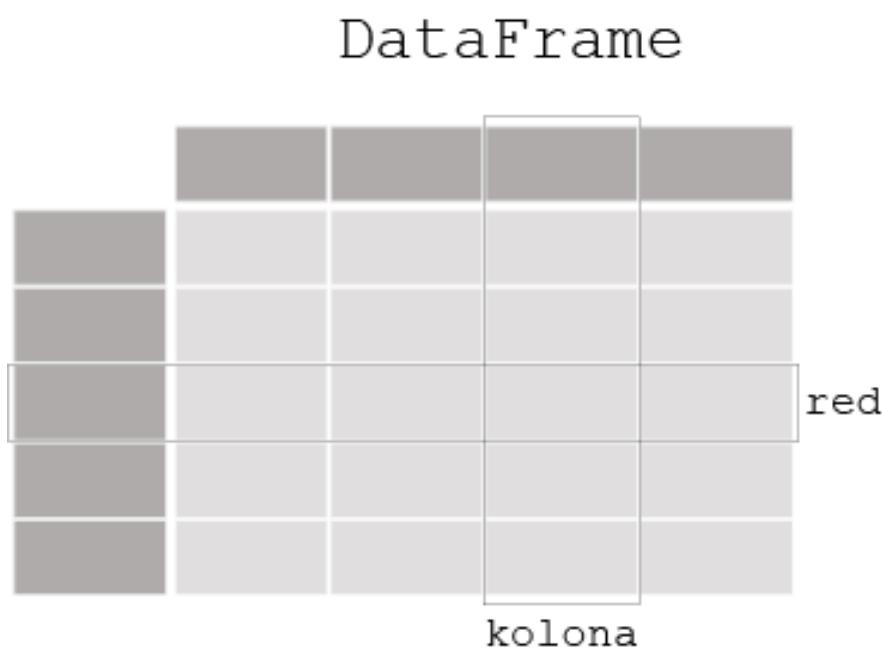
# pandas: učitavanje i pisanje datoteka

## PANDAS I DATAFRAME OBJEKAT

*U pandas paketu, tabela sa podacima se naziva DataFrame (okvir podataka).*

Kada se radi sa tabelarnim podacima, kao što su podaci koji su smešteni u tabelama ili bazama podataka, pandas paket olakšava rad sa ovakvim tipovima podataka.

U pandas paketu, tabela sa podacima se naziva [DataFrame](#) (okvir podataka).



Slika 5.1 DataFrame objekat u pandas paketu. [Izvor: Autor]

[DataFrame](#) predstavlja dvodimenzionalnu strukturu podataka koja može smestiti podatke različitih tipova (uključujući karaktere, cele brojeve, razlomljene brojeve, kategorisane podatke) u kolonama. Vrlo je slična običnoj tabeli.

Svaka kolona unutar [DataFrame](#) objekta jeste [Series](#) (red) objekat.

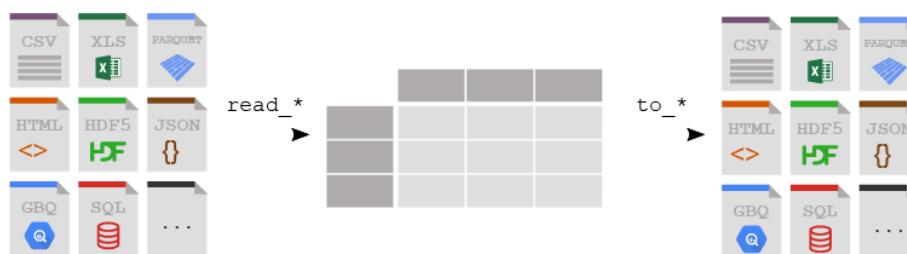
Mnoge operacije pri korišćenju pandas paketa vratiće kao tip podataka [DataFrame](#) ili [Series](#) objekte.

## PANDAS I PODRŽATI FORMATI

*Pandas podržava integraciju sa velikim brojem tipova datoteka ili sa izvorima podataka*

Pandas podržava integraciju sa velikim brojem tipova datoteka ili sa izvorima podataka ([excel](#) tabele, [csv](#) datoteke, [SQL](#), [JSON](#) i mnogi drugi formati).

Uvoz podataka iz svakog od ovih izvora podataka uvek kreće sa funkcijom `read_*` gde zvezdica opisuje koji je tip podataka u pitanju. Na sličan način, metode `to_*` upisuju podatke u datoteke.



Slika 5.2 Upis i čitanje preko pandas paketa. [Izvor: <https://pandas.pydata.org>]

Podaci za uvoz u excel:

```
Fakultet,Smer,Godina,Semestar,Sifra,PunoIme
FIT,IT,1,1,CS101,Uvod u objektno-orientisano programiranje
FIT,IT,1,1,IT101,Osnove informacionih tehnologija
FIT,IT,1,1,MA101,Matematika 1
FIT,IT,1,1,NT111,Engleski 1
FIT,IT,1,2,CS102,Objekti i apstrakcija podataka
FIT,IT,1,2,IT210,Sistemi informacionih tehnologija
FIT,IT,1,2,CS115,Diskretne strukture
FIT,IT,1,2,NT112,Engleski 2
FIT,IT,2,3,IT331,Racunarske mreze i komunikacije
FIT,IT,2,3,SE201,Uvod u softversko inzenjerstvo
FIT,IT,2,3,IT350,Baze podataka
FIT,IT,2,3,NT213,Engleski za informaticare
FIT,IT,2,4,CS230,Distribuirani sistemi
FIT,IT,2,4,IT370,Interakcija covek-racunar
FIT,IT,2,4,CS220,Arhitektura racunara
FIT,IT,2,4,CS323,C/C++ programski jezik
FIT,IT,3,5,IT225,Veb sistemi 1
FIT,IT,3,5,CS225,Operativni sistemi
FIT,IT,3,5,IT335,Administracija racunarskih sistema i mreza
FIT,IT,3,5,CS324,Skripting jezici
FIT,IT,3,6,SE325,Upravljenje projektima razvoja softvera
FIT,IT,3,6,IT355,Veb sistemi 2
FIT,IT,3,6,CS330,Razvoj mobilnih aplikacija
FIT,IT,3,6,MA273,Verovatnoca i statistika
FIT,IT,3,6,IT375,Racunarsko upravljanje sistemima
FIT,IT,3,6,CS450,Klaud kompjuting
```

```
FIT,IT,3,7,IT381,Zastita i bezbednost informacija
FIT,IT,3,7,CS322,C# programski jezik
FIT,IT,4,7,CS103,Algoritmi i strukture podataka
FIT,IT,4,7,IT333,Bezicne I mobilne komunikacije
FIT,IT,4,7,IT376,Robotika
FIT,IT,4,7,OM350,Preduzetnistvo
FIT,IT,4,7,SE311,Projektovanje i arhitektura softvera
FIT,IT,4,7,SE321,"Obezbedjivanje kvaliteta, testiranje i odrzavanje softvera"
FIT,IT,4,8,IT390,Prefesionalna praksa i etika
FIT,IT,4,8,NT310,Profesionalna komunikacija
FIT,IT,4,8,SE490,Strucna praksa (4 meseca)
FIT,IT,4,8,SE495,Zavrsni rad
```

Primer: Uvoz iz Excel tabela (6 minuta)

Tekst sa strane ubaciti u "FIT\_IT\_predmeti.xlsx" datoteku. Zatim, pročitati datoteku kroz [pandas](#) i prikazati u [jupyter](#) svesci.

Rešenje:

```
import pandas as pd
df2 = pd.read_excel('FIT_IT_predmeti.xlsx')
df2
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ATRIBUTI I METODE DATAFRAME OBJEKTA

*Pri radu sa DataFrame objektima potrebno je znati najčešće korišćene atribute i metode*

Konstrukcija [DataFrame](#) objekta iz drugih tipova podataka

### Imenik

Ključ imenika jeste ime kolone, a lista vrednosti jesu sadržaj te kolone.

```
import pandas as pd

d = {'col1': [1, 2], 'col2': [3, 4]}
df = pd.DataFrame(data=d)
```

### Lista

Pri unosu liste, potrebno je dodatno naznačiti ime kolone.

```
import pandas as pd

lst = [1994, 1996, 1998, 1999, 2002, 2003, 2005, 2007, 2010, 2011, 2013, 2016, 2019]
df = pd.DataFrame(lst, columns=['Album Issued'])
```

Ukoliko je unutar liste još listi, onda je svaka lista jedna kolona.

### Numpy Niz

Unos numpy niza je sličan kao kod unosa listi.

```
import numpy as np
import pandas as pd

n = np.array([1, 2, 3], [4, 5, 6], [7, 8, 9])
df = pd.DataFrame(n, columns=['a', 'b', 'c'])
```

### Tipovi podataka unutar *DataFrame* objekta

atribut *.dypes* vraća tiove podataka po koloni

```
import pandas as pd
df = pd.DataFrame(...)

print(df.dtypes)
```

### Broj redova i kolona

Atribut *.shape* vratiće tuple koji sadrži broj redova i kolona u *DataFrame* objektu

```
import pandas as pd
df = pd.DataFrame(...)

print(df.shape)
```

### Broj redova i kolona i tipovi podataka

Metoda *.info()* vratiće informaciju o broju redova i kolona, ali i o tipovima podataka po koloni

```
import pandas as pd

df = pd.DataFrame(...)
print(df.info())
```

## ATTRIBUTI ZA PRIKAZ DATAFRAME OBJEKATA

*Pri radu sa Jupyter sveskama, inicijalno se neće moći videti sve kolone i svi redovi, pa se moraju podesiti opcije prikaza.*

### Opcije prikaza

Pri radu sa Jupyter sveskama, inicijalno se neće moći videti sve kolone i svi redovi.

Ukoliko je potrebno da se vidi određeni broj redova i/ili kolona, potrebno je podesiti opcije prikaza

```
import pandas as pd

# DataFrame objekat sa puno redova i kolona
df = pd.DataFrame(...)

# podešavanje max kolona i redova
pd.set_option('display.max_columns', max_kolone)
pd.set_option('display.max_rows', max_redovi)
```

### Pregled samo prvih $n$ ili poslednjih $n$ redova

Metodama `.head()` i `.tail()` moguće je imati pregled samo prvih ili samo poslednjih redova objekta

```
import pandas as pd

df = pd.DataFrame(...)

print(df.head(no_rows_from_first))
print(df.tail(no_rows_from_last))
```

### Lociranje određenih elemenata

Pristup grupi redova i kolona po labelama moguće je korišćenjem metode `.loc[]`.

Ukoliko se koristi lociranje, a labele su celi brojevi, onda se koristi metoda `.iloc[]`

### Brisanje kolone

Moguće je brisanje kolone komandom `del`.

```
import pandas as pd

df = pd.DataFrame(...)
del df['column_name']
```

```
import pandas as pd

df = pd.DataFrame(...)

df.loc['column_label']
df.iloc[imteger]
```

## PRIMER PRIKAZA KROZ KONZOLU I KROZ JUPYTER SVESKU

*Kod pri normalnom radu i u Jupyter sveskama je (gotovo) identičan, ali je prikaz drugačiji.*

**Primer: Prikaz tabela kroz izlaz konzole i kroz izlaz Jupyter sveske (10 minuta)**

Tekst u nastavku sačuvati kao "**FIT\_SE\_predmeti.csv**" datoteku, i postaviti je u radni direktorijum. Otvoriti preko pandas paketa i prikazati sadržaj.

Ponoviti isto kroz Jupyter svesku i uočiti razliku.

```
Fakultet,Smer,Godina,Semestar,Sifra,PunoIme
FIT,SE,1,1,CS101,Uvod u objektno-orientisano programiranje
FIT,SE,1,1,IT101,Osnove informacionih tehnologija
FIT,SE,1,1,MA101,Matematika 1
FIT,SE,1,1,NT111,Engleski 1
FIT,SE,1,2,CS102,Objekti i apstrakcija podataka
FIT,SE,1,2,IT210,Sistemi informacionih tehnologija
FIT,SE,1,2,CS115,Diskretne strukture
FIT,SE,1,2,NT112,Engleski 2
FIT,SE,2,3,CS103,Algoritmi i strukture podataka
FIT,SE,2,3,SE201,Uvod u softversko inzenjerstvo
FIT,SE,2,3,IT350,Baze podataka
FIT,SE,2,3,NT213,Engleski za informaticare
FIT,SE,2,4,CS230,Distribuirani sistemi
FIT,SE,2,4,IT370,Interakcija čovek-racunar
FIT,SE,2,4,SE211,Konstruisanje softvera
FIT,SE,2,4,MA202,Matematika 2
FIT,SE,3,5,IT225,Veb sistemi 1
FIT,SE,3,5,SE311,Projektovanje i arhitektura softvera
FIT,SE,3,5,SE322,Inzenjerstvo zahteva
FIT,SE,3,5,SE321,"Obezbeđivanje kvaliteta, testiranje i održavanje softvera"
FIT,SE,3,6,SE325,Upravljenje projektima razvoja softvera
FIT,SE,3,6,IT355,Veb sistemi 2
FIT,SE,3,6,CS220,Arhitektura racunara
FIT,SE,3,6,CS323,C/C++ programski jezik
FIT,SE,3,6,MA273,Verovatnoca i statistika
FIT,SE,3,6,IT375,Racunarsko upravljanje sistemima
FIT,SE,3,6,CS450,Klaud kompjuting
FIT,SE,3,6,CS330,Razvoj mobilnih aplikacija
FIT,SE,4,7,IT381,Zastita i bezbednost informacija
FIT,SE,4,7,IT376,Robotika
FIT,SE,4,7,CS322,C# programski jezik
FIT,SE,4,7,SE401,Timski razvoj softvera
FIT,SE,4,7,CS225,Operativni sistemi
FIT,SE,4,7,CS324,Skripting jezici
FIT,SE,4,7,OM350,Preduzetništvo
FIT,SE,4,8,IT390,Prefesionalna praksa i etika
FIT,SE,4,8,NT310,Profesionalna komunikacija
FIT,SE,4,8,SE490,Strucna praksa (4 meseca)
FIT,SE,4,8,SE495,Završni rad
```

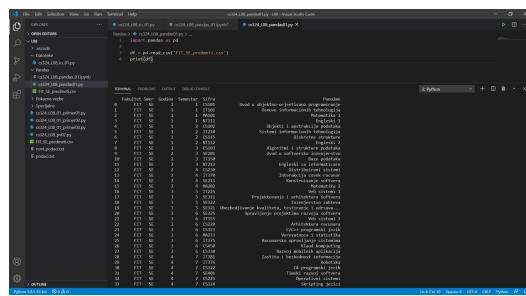
**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Rešenje:**

Kod je isti, ali prikaz biće drugačiji.

```
import pandas as pd

df = pd.read_csv('FIT_SE_predmeti.csv')
print(df)
```



Slika 5.3 Prikaz datoteke "FIT\_SE\_predmeti.csv" kroz konzolu. [Izvor: Autor]

Slika 5.4 Prikaz datoteke "FIT\_SE\_predmeti.csv" kroz Jupyter svesku. [Izvor: Autor]

## ✓ Poglavlje 6

### Pokazne vežbe #8

## UVOD U POKAZNE VEŽBE

*Pokazne vežbe rade se 45 minuta.*

Pokazna vežba #8 odnosi se na rad sa numpy i pandas paketima.

U pokaznoj vežbi prelaze se 4 zadatka, po dva za svaki paket.

Zadatak #1 odnosi se na rad sa transponovanim vektorima.

Zadatak #2 odnosi se na ubacivanje matrica u matricu.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ZADATAK #1 (15 MINUTA)

*Zadatak #1 odnosi se na rad sa transponovanim nizovima*

### **Zadatak #1 (15 minuta)**

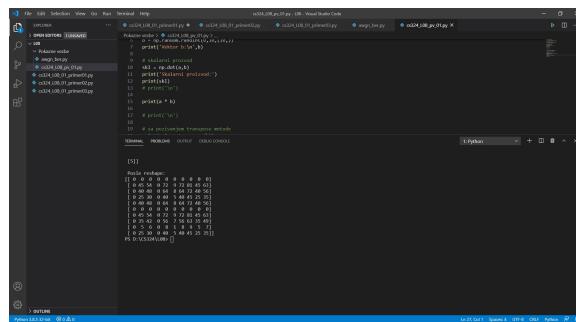
Napraviti dva niza od po sto elemenata. Svaki element je nasumično izabran broj od 0 do 10.

- 1. Pronaći skalarni proizvod vektora (en. **dot product**) koji se računa:

$$a \cdot b = \sum_{i=1}^n a_i b_i$$

- 2. Izračunati proizvod:

$$\begin{aligned} a \times b^T \\ b^T \times a \end{aligned}$$



Slika 6.1 Izlaz poslenjeg proizvoda kod pokazne vežbe #1. [Izvor: Autor]

```

import numpy as np

a = np.random.randint(0,10,(10,))
print('Vektor a:\n',a)

b = np.random.randint(0,10,(10,))
print('Vektor b:\n',b)

# skalarni prozvod
skl = np.dot(a,b)
print('Skalarni proizvod:')
print(skl)

# # poziv reshape metode
print(b)
print('posle pravog trnasponovanja')
b = b.reshape(-1,1)
print(b)

print('\n Posle reshape:')
print(b * a)

```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ZADATAK #2 (5 MINUTA)

*Zadatak #2 odnosi se na rad sa kreiranje željene matrice.*

### Zadatak #2 (5 minuta)

Napisati program koji će napraviti sledeću matricu:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 9 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

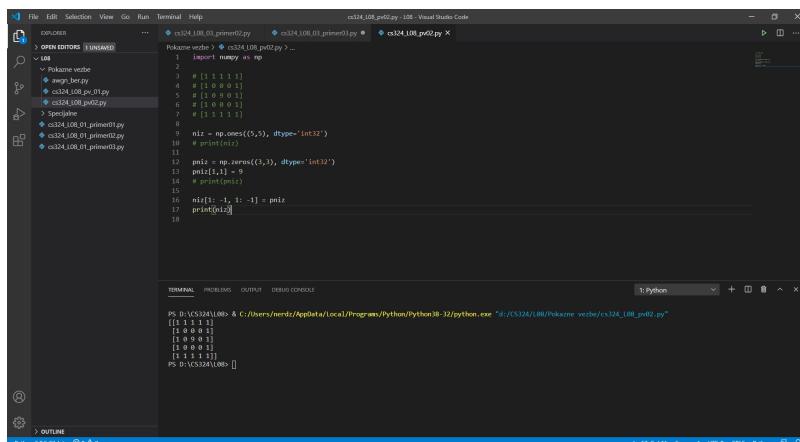
Moguće je direktno ubaciti samo cifru 9, a za ostale elemente koristiti numpy funkcije.

```
import numpy as np

# inicijalizovanje prvog niza
niz = np.ones((5,5), dtype='int32')
print(niz)

# pomocni, unutrasni niz
pomocni_niz = np.zeros((3,3), dtype='int32')
pomocni_niz[1,1] = 9
print(pomocni_niz)

# ubacivanje jednog niza u drugi
niz[1:-1, 1:-1] = pomocni_niz
print(niz)
```



Slika 6.2 Konstrukcija željene matrice. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ZADATAK #3 (10 MINUTA)

*Zadatak #3 odnosi se na preveru vrednosti DataFrame objekta i maskiranje*

### Zadatak #3 (10 minuta)

Napisati program koji će kroz pandas napraviti sledeću tabelu:

AAA	BBB	CCC
4	10	100
5	20	50
6	30	-30
7	40	-50

Zatim, uraditi sledeće:

- Ukoliko su vrednosti kolone AAA veće ili jednake broju 5, onda će u tim redovima, vrednosti kolone BBB biti jednake -1
- Ukoliko su vrednosti kolone AAA veće ili jednake broju 5, onda će u tim redovima, vrednosti kolona BBB i CCC biti jednake 555.
- Postaviti masku df\_mask. i primeniti na tabelu, sa vrednošću -1000. Maska je data:

```
df_mask = pd.DataFrame({'AAA': [True] * 4,
                        'BBB': [False] * 4,
                        'CCC': [True, False] * 2})
```

**Rešenje:**

```
import pandas as pd

df = pd.DataFrame({'AAA': [4, 5, 6, 7],
                   'BBB': [10, 20, 30, 40],
                   'CCC': [100, 50, -30, -50]})

print(df)

#1
df.loc[df.AAA >= 5, 'BBB'] = -1
print(df)

#2
df.loc[df.AAA >= 5, ['BBB', 'CCC']] = 555
print(df)

#3
df_mask = pd.DataFrame({'AAA': [True] * 4,
                        'BBB': [False] * 4,
                        'CCC': [True, False] * 2})
print(df.where(df_mask, -1000))
```

## ZADATAK #4 (15 MINUTA)

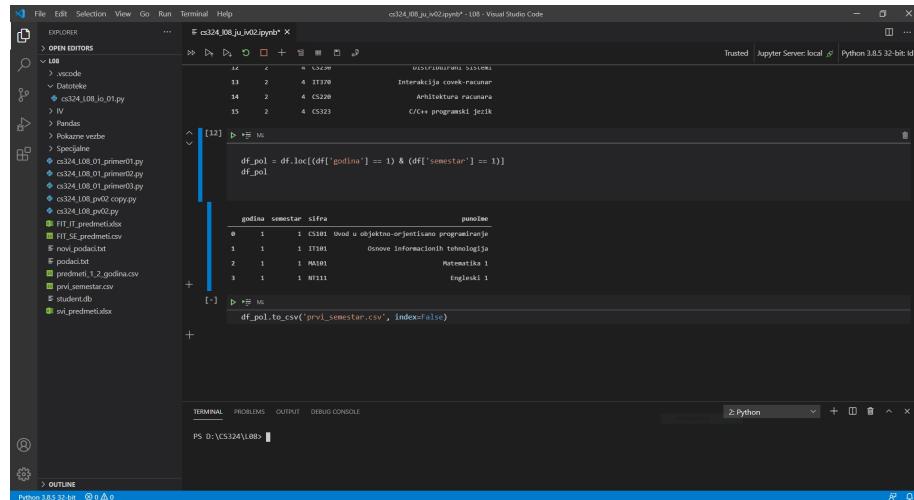
*Zadatak #4 se odnosi na odabir elemenata DataFrame objekta po više kriterijuma.*

### **Zadatak #4 (15 minuta)**

Dat je CSV dokument "predmeti\_1\_2\_godina.csv" sa podacima u nastavku.

Učitati i odštampati kao DataFrame objekat. Zatim, pronaći predmete iz prve godine i iz prvog semestra, i smestiti i novi DataFrame objekat.

Štampati novi DataFrame objekat u CSV datoteku bez indeksa.



Slika 6.3 Izlaz preko Jupyter svezaka. [Izvor: Autor]

```
godina,semestar,sifra,punoIme
1,1,CS101,Uvod u objektno-orientisano programiranje
1,1,IT101,Osnove informacionih tehnologija
1,1,MA101,Matematika 1
1,1,NT111,Engleski 1
1,2,CS102,Objekti i apstrakcija podataka
1,2,IT210,Sistemi informacionih tehnologija
1,2,CS115,Diskretne strukture
1,2,NT112,Engleski 2
2,3,IT331,Racunarske mreze i komunikacije
2,3,SE201,Uvod u softversko inzenjerstvo
2,3,IT350,Baze podataka
2,3,NT213,Engleski za informaticare
2,4,CS230,Distribuirani sistemi
2,4,IT370,Interakcija covek-racunar
2,4,CS220,Arhitektura racunara
2,4,CS323,C/C++ programska jezik
```

```
import pandas as pd

df = pd.read_csv('predmeti_1_2_godina.csv')
print(df)

df_pol = df.loc[(df['godina'] == 1) & (df['semestar'] == 1)]
df_pol
print(df_pol)

df_pol.to_csv('prvi_semestar.csv', index=False)
```

## ✓ Poglavlje 7

### Individualne vežbe #8

#### INDIVIDUALNE VEŽBE

*Zadaci individualnih vežbi se ukupno rade 90 minuta*

##### **Zadatak #1 (45 minuta)**

Korišćenjem numpy generisati niz od 10000 elemenata, koji sadrži elemente 0 i 1. Prebrojati koliko ima vrednosti 0 i koliko ima vrednosti 1. Zatim, sve vrednosti koje su 0 prebaciti u -1.

Napraviti novi niz koji ima nasumične razlomljene elemente u vrednosti od -1.5 do +1.5. Sabrati elemente niza u novi niz.

Elemente novog niza preuređiti tako da svi negativni elementi postanu nule, a svi pozitivni elementi postanu jedinice.

Prebrojati nule i jedinice, i uporediti sa brojem jedinice i nula prvobitnog niza. Štampati razlike u nulama i jedinicama podeljeno sa brojem elemenata (10000). Ovo je verovatnoća greške.

Ponoviti za 1000 elemenata i 100000 elemenata.

Uraditi isti zadatak bez numpy i uporediti brzinu izvršenja.

##### **Zadatak #2 (45 minuta)**

Pomoću sqlite3 paketa napraviti bazu podataka "moji\_predmeti.db" sa kolonama:

```
Godina: Int # Godina na kome ste slusali predmet
Semestar: Int # Semestar u kome ste slusali predmet
Sifra: Text # Sifra predmeta
PunoIme: Text # Puno ime predmeta
Polozen: Bool # Status predmeta (True = polozen, False = nije polozen)
Ocena: Int # Nema ocenu ako nije polozen
```

Zatim, napraviti funkciju koja učitava predmete (unosi se sa konzole) i ubacuje u bazu podataka.

Pomoću pandas paketa uzeti ceo sadržaj baze. Napraviti filter koji će ispisivati samo položene predmete. Sačuvati novu tabelu u datoteku "polozeni\_predmeti.csv". Tabela ima sledeće kolone:

```
Sifra: Text # Sifra predmeta
PunoIme: Text # Puno ime predmeta
Ocena: Int # Ocena
```

Godina: Int # Godina na kome ste slusali predmet  
Semestar: Int # Semestar u kome ste slusali predmet

## ✓ Poglavlje 8

### Domaći zadatak #8

#### ZADACI ZA DOMAĆI ZADATAK #8

*Osmi domaći zadatak se radi okvirno 3h.*

##### Domaći zadatak #1

Korišćenjem numpy paketa napraviti sledeće matrice. Nije dozvoljen ručni unos vrednosti po indeksu, već treba uneti matricu u matricu.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 \\ 0 & 10 & 10 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

##### Domaći zadatak #2

Napraviti bazu podataka "biblioteka.db", koja sadrži tabelu:

```
knjige:  
(katBroj int, naslov text, izdavac text, godinaIzdavanja int, izdata bool)
```

Napraviti funkciju za ubacivanja novih knjiga sa konzole. Ubacuje se **katBroj**, **naslov**, **izdavac**, **godinaIzdavanja**, a atribut **izdat** se podrazumevano stavlja na **False**. Uhvatiti izuzetak ukoliko je knjiga izdata u budućnosti (proveriti preko **datetime** paketa).

Napraviti funkciju **podesiIzdat(katBroj)** koja podešava vrednost **izdat** i upisuje u bazu novu vrednost za datu knjigu.

Preko pandas paketa učitati sadržaj tabele knjige. Napraviti novi DataFrame objekat **savremene\_knjige** koji će sadržati knjige kod kojih je godina izdavanja od 2000. godine.

Upisati u csv datoteku sve knjige koje su izdate u datoteku **izdate\_knjige.csv** sa kolonama **Naslov, izdavac, godina izdavanja**.

## PREDAJA DOMAĆEG ZADATKA

*Nakon 10 dana od lekcije, poeni za domaći zadatak za tradicionalne studente se umanjuju za 50%.*

### **Tradisionalni studenti:**

Domaći zadatak treba dostaviti najkasnije nedelju dana nakon predavanja za 100% poena. Nakon toga poeni se umanjuju za 50%.

### **Internet studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganja ispita. Domaći zadaci se brane!

Domaći zadatak poslati dr Nemanji Zdravkoviću: nemanja.zdravkovic@metropolitan.ac.rs

Obavezno koristiti uputstvo za izradu domaćeg zadatka.

Uz .doc dokument (koji treba sadržati i screenshot svakog urađenog zadatka kao i komentare za zadatak), poslati i izvorne i dodatne datoteke.

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Zaključak lekcije #8*

#### **Rezime:**

U ovoj lekciji bilo je reči o paketima numpy i pandas za Python 3 programski jezik. Oba paketa se instaliraju preko pip upravljača paketa.

Paket NumPy radi sa homogenim nizovima i više je optimizovan nego pri radu sa listama, zbog vektorizacije izvršenja operacija. NumPy se može smatrati alternativom za MATLAB programski jezik.

Paket pandas služi za manipulaciju podacima koja se mogu predstaviti tabelarno. Pandas pruža mnoge mogućnosti sortiranja, filtracije nad podacima, ali i podržava veliki broj formata koji koriste razne aplikacije, pa i baze podataka.

#### **Literatura:**

- David Beazley, Brian Jones, *Python Cookbook: Recipes for Mastering Python 3*, 3rd edition, O'Reilly Press, 2013.
- Mark Lutz, *Learning Python*, 5th Edition, O'Reilly Press, 2013.
- Andrew Bird, Lau Cher Han, et. al, *The Python Workshop*, Packt Publishing, 2019.
- Al Sweigart, *Automate the boring stuff with Python*, 2nd Edition, No Starch Press, 2020.





CS324 - SKRIPTING JEZICI

Vizuelizacija podataka i rad sa  
slikama

Lekcija 09

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 09

### *VIZUELIZACIJA PODATAKA I RAD SA SLIKAMA*

- ✓ Vizuelizacija podataka i rad sa slikama
- ✓ Poglavlje 1: Uvod u matplotlib
- ✓ Poglavlje 2: pyplot i vizuelizacija podataka
- ✓ Poglavlje 3: Grafikoni i vrste grafikona
- ✓ Poglavlje 4: Uvod u pillow
- ✓ Poglavlje 5: Rad sa slikama
- ✓ Poglavlje 6: Pokazna vežba #9
- ✓ Poglavlje 7: Individualna vežba #9
- ✓ Poglavlje 8: Domaći zadatak #9
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ❖ Uvod

# UVOD

### *Uvod u lekciju #9*

U devetoj lekciji biće reči o prikazivanju podataka u Python 3.x programskom jeziku, kao i o korišćenju paketa za iscrtavane podataka [matplotlib](#).

Pri manipulaciji podataka, pogotovo kada se radi o velikoj količini podataka, Python poseduje niz biblioteka koje te podatke mogu vizuelno predstaviti. Najpopularnija biblioteka jeste matplotlib.

U početku lekcije biće reči o matplotlib paketu. Matplotlib je počeo je kao projekat ranih 2000tih za vizuelizaciju elektronskih signala mozga za pacijente koji pate od epilepsije.

Autor matplotlib-a, dr. Džon Hunter, neuro biolog, htio je da u jeziku Python dobije mogućnost prikaza podataka kao i u MATLAB programskom jeziku.

Matplotlib je jako koristan paket koji služi za pravljenje 2D grafikona, najčešće onih koji se koriste u naučnim radovima u prezentacijama. Gotovo svaki grafikon koji se može napraviti u Excel-u ili sličnim programima, može se napraviti i pomoću Matplotlib paketa. Osim 2D, ovaj paket poseduje mogućnost pravljenja 3D grafika, kao i animacija.

Python imaging library, skraćeno [PIL](#), ili u novijim verzijama [Pillow](#), jeste paket koji dodaje podršku za otvaranje, manipulaciju i čuvanje različitih formata slika u Python programskom jeziku.

Pillow nudi manipulaciju po pikselu, obradu sa transparentnošću slika i maskiranje, kao i razne filtere. Najčešće korišćeni formati su naravno PNG, JPEG, GIF, TIFF i Bitmap slike.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 1

### Uvod u matplotlib

#### UVOD U MATPLOTLIB

*Paket `matplotlib` predstavlja paket za vizuelizaciju podataka kroz grafikone.*



Slika 1.1 matplotlib paket. [Izvor: [matplotlib.org](http://matplotlib.org)]

Paket `matplotlib` predstavlja ekstenziju numeričkog paketa `NumPy`, sa svrhom vizuelizacije podataka.

Matplotlib proširuje rad sa numeričkim podacima predstavljajući ih kroz različite 2D i 3D grafikone.

Matplotlib podržava objektno-orientisani API za ubacivanje grafikona (en. `plots`) unutar drugih alata za grafički korisnički interfejs (en. `Graphical User Interface, GUI`)

Analogno `ndarray` klasi koja je predstavljala osnovu za `NumPy` paket, tako je i `pyplot` "srž" paketa `matplotlib`, jer predstavlja skup funkcija za prikaz grafikona u Python programskom jeziku.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

#### INSTALACIJA MATPLOTLIB PAKETA

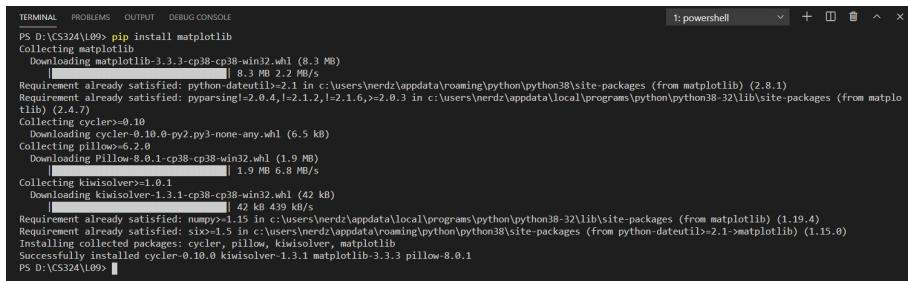
*Paket `matplotlib` nije deo standardne Python biblioteke, te je neophodno dodatno preuzeti paket korišćenjem konzole i pip komandi.*

Paket matplotlib nije deo standardne Python biblioteke, te je neophodno dodatno preuzeti paket.

Kao i prethodne pakete, paket matplotlib se može instalirati korišćenjem konzole preko pip komande:

```
pip install matplotlib
```

Nakon preuzimanja potrebnih datoteka, matplotlib se može koristiti prilikom razvijanja Python aplikacija.



```
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
PS D:\CS324\LO9> pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.3.3-cp38-cp38-win32.whl (8.3 MB)
    8.3 MB 2.3 MB/s
Requirement already satisfied: python-dateutil>=2.1 in c:\users\nerdz\appdata\roaming\python\python38\site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: pytz!=2020.1,>=2019.3 in c:\users\nerdz\appdata\local\programs\python\python38-32\lib\site-packages (from matplotlib) (2.4.7)
Collecting cycler>0.10
  Downloading cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Collecting pillow>6.2.0
  Downloading Pillow-8.0.1-cp38-cp38-win32.whl (1.9 MB)
    1.9 MB 0.8 MB/s
Collecting kiwisolver>1.0.1
  Downloading kiwisolver-1.3.1-cp38-cp38-win32.whl (42 kB)
    42 kB 439 kB/s
Requirement already satisfied: numpy>=1.15 in c:\users\nerdz\appdata\local\programs\python\python38\lib\site-packages (from matplotlib) (1.19.4)
Requirement already satisfied: six>1.5 in c:\users\nerdz\appdata\roaming\python\python38\site-packages (from python-dateutil>=2.1>matplotlib) (1.15.0)
Installing collected packages: cycler, pillow, kiwisolver, matplotlib
Successfully installed cycler-0.10.0 kiwisolver-1.3.1 matplotlib-3.3.3 pillow-8.0.1
PS D:\CS324\LO9>
```

Slika 1.2 Instalacija matplotlib paketa. [Izvor: Autor]

Provera (sa verzijom koja je instalirana) se može izvršiti pokretanjem pip list komande.

```
pip list
```

Slika 1.3 Provera instalacije matplotlib paketa. [Izvor: Autor]

## UVOZ MATPLOTLIB PAKETA U RADNI DIREKTORIJUM

*Sledi video o instalaciji matplotlib paketa*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

Nakon instalacije matplotlib paketa, moguće je uvesti paket u radni direktorijum.

Poželjno je uvesti paket sa preimenovanjem, i to:

```
import matplotlib.pyplot as plt
```

Napisano je pravila da se skup funkcija za kreaciju grafikona **pyplot** preimenuje u **plt**.

Moguće je izvršiti i uvoz u imenski prostor sa preimenovanjem

```
from matplotlib import pyplot as plt
```

## ▼ Poglavlje 2

# pyplot i vizuelizacija podataka

## SKUP FUNKCIJA PYPLOT

*Skup funkcija pyplot predstavlja glavni deo matplotlib paketa, jer sadrži svu funkcionalnost za kreaciju različitih vrsta grafikona.*

Skup funkcija [pyplot](#) predstavlja glavni deo matplotlib paketa, jer sadrži svu funkcionalnost za kreaciju različitih vrsta grafikona.

Svaki grafikon koji se napravi preko matplotlib paketa predstavlja [figuru](#) (en. [Figure](#)), od koje svaka može sadržati jednu ili više osa (en. [axis](#)).

Ose predstavljaju prostor gde se specificiraju tačke figure u koordinatama.

Ukoliko se radi od 2D figuri, onda su koordinate **x** i **y**, ili u polarnom koordinatnom sistemu **θ** i **r**, a ukoliko se radi u 3D figuri, koordinate su **x**, **y** i **z**.

U matplotlib paketu nije neophodno posebno isCRTavati podatke za svaku osu, već je moguće proslediti podatke kao skup listi, gde su elementi svaki liste tačke po jednoj osi.

Kao i kod Numpy paketa, matplotlib paket predstavlja alternativu grafikonima u [MATLAB](#) programskom jeziku, stim što je matplotlib besplatna alternativa u Python jeziku.

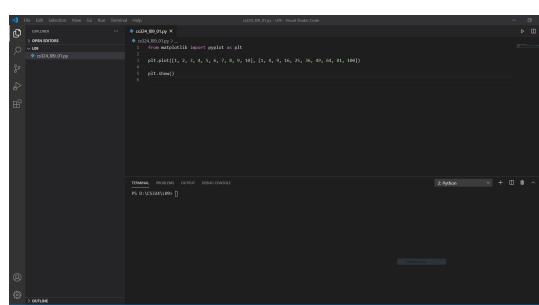
### Primer (3 minuta)

Napraviti običan grafikon koji na x osi pokazuje brojeve od 1 do 10, a na y osi pokazuje njihove kvadrate.

```
from matplotlib import pyplot as plt

plt.plot([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], [1, 4, 9, 16, 25, 36, 49, 64, 81, 100])

plt.show()
```



Slika 2.1 Jednostavno iscrtavanje u 2D grafikonu. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## NASLOV I LABLE OSА

*Vizuelizacija podataka sama po sebi nije dovoljna, već je neophodno uz samu figuru dodati i informacije na šta se ona i odnosi.*

Vizuelizacija podataka sama po sebi nije dovoljna, već je neophodno uz samu figuru dodati i informacije na šta se ona i odnosi.

### Naslov figure

Dodavanje naslova figuri se može ostvariti korišćenjem funkcije **title()**

```
#dodavanje naslova figuri
from matplotlib import pyplot as plt

plt.title("Naslov figure")
```

### Lable osа

Naslovi osa (labele) se mogu dodati pojedinačno za svaku osu korišćenjem funkcija **xlabel()** i **ylabel()**.

```
#dodavanje labela osa
from matplotlib import pyplot as plt

plt.xlabel("Ime x ose")
plt.ylabel("Ime y ose")
```

### Primer (7 minuta)

Napisati program koji iscrtava linearu funkciju ( $y = k * x + n$ ):

Definisati funkciju **linear\_func(x,k,n)** gde je **x** lista celih brojeva. Napraviti figuru sa naslovom i labelama osa.

```
from matplotlib import pyplot as plt

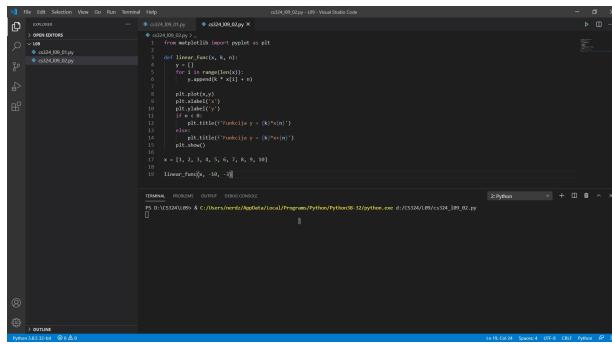
def linear_func(x, k, n):
    y = []
    for i in range(len(x)):
        y.append(k * x[i] + n)

    plt.plot(x,y)
    plt.xlabel('x')
    plt.ylabel('y')
    if n < 0:
```

```
plt.title(f'Funkcija y = {k}*x^{n}')
else:
    plt.title(f'Funkcija y = {k}*x+{n}')
plt.show()

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

linear_func(x, -10, -3)
```



Slika 2.2 Program za crtanje linearne funkcije. [Izvor: Autor]

Slika 2.3 Figura linearne funkcije. [Izvor: Autor]

## PRIMER: ISCRTAVANJE LINEARNE FUNKCIJE

*Sledi autorski video sa objašnjenjem kreiranja prve figure kroz matplotlib*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PRIKAZ VIŠESTRUKIH PODATAKA I LEGENDA FIGURE

*Na jednoj figuri moguće je prikazati više od jednog skupa podataka (više od jedne krive). U tom slučaju, poželjno je dodati legendu.*

### Prikaz višestrukih podataka na istoj figuri

Moguće je prikazati više podataka na istoj figuri, i to pozivom **plot()** funkcije za svaki skup podataka.

### Prikaz legende

Da bi se lakše razumelo koja kriva se odnosi na koje podatke, potrebno je ubaciti i legendu, pozivom funkcije **legend()**, kojoj su argumenti imena podataka u onom redosledu koji se pojavljuju

```
#dodavanje legende u grafikon
from matplotlib import pyplot as plt

...
plt.legend(["Prvi podaci", "Drugi podaci", ...])
```

Moguće je i direktno dodavanje labele samim podacima prilikom poziva **plot()** funkcije, i na taj način se mora se brinuti o redosledu pojave podataka u grafikonu.

### Primer (8 minuta):

Iscrati podatke koje su date u nastavku na istoj figuri. Koristiti legendu.

```
# Test podaci
# godina developera

godine = [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
          36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
          54, 55]

zarada_prosek = [17784, 16500, 18012, 20628, 25206, 30252, 34368, 38496, 42000,
                  46752, 49320, 53200, 56000, 62316, 64928, 67317, 68748, 73752, 77232,
                  78000, 78508, 79536, 82488, 88935, 90000, 90056, 95000, 90000, 91633,
                  91660, 98150, 98964, 100000, 98988, 100000, 108923, 105000, 103117]

zarada_python = [20046, 17100, 20000, 24744, 30500, 37732, 41247, 45372, 48876,
                  53850, 57287, 63016, 65998, 70003, 70000, 71496, 75370, 83640, 84666,
                  84392, 78254, 85000, 87038, 91991, 100000, 94796, 97962, 93302, 99240,
                  102736, 112285, 100771, 104708, 108423, 101407, 112542, 122870, 120000]

zarada_javascript = [16446, 16791, 18942, 21780, 25704, 29000, 34372, 37810, 43515,
                      46823, 49293, 53437, 56373, 62375, 66674, 68745, 68746, 74583, 79000,
                      78508, 79996, 80403, 83820, 88833, 91660, 87892, 96243, 90000, 99313,
                      91660, 102264, 100000, 100000, 91660, 99240, 108000, 105000, 104000]
```

```
from matplotlib import pyplot as plt

godine = [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
          36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
          54, 55]

python_zarada = [20046, 17100, 20000, 24744, 30500, 37732, 41247, 45372, 48876,
                  53850, 57287, 63016, 65998, 70003, 70000, 71496, 75370, 83640, 84666,
                  84392, 78254, 85000, 87038, 91991, 100000, 94796, 97962, 93302, 99240,
                  102736, 112285, 100771, 104708, 108423, 101407, 112542, 122870, 120000]

javascript_zarada = [16446, 16791, 18942, 21780, 25704, 29000, 34372, 37810, 43515,
                      46823, 49293, 53437, 56373, 62375, 66674, 68745, 68746, 74583, 79000,
                      78508, 79996, 80403, 83820, 88833, 91660, 87892, 96243, 90000, 99313,
                      91660, 102264, 100000, 100000, 91660, 99240, 108000, 105000, 104000]
```

```
prosecna_zarada = [17784, 16500, 18012, 20628, 25206, 30252, 34368, 38496, 42000, 46752, 49320, 53200, 56000, 62316, 64928, 67317, 68748, 73752, 77232, 78000, 78508, 79536, 82488, 88935, 90000, 90056, 95000, 90000, 91633, 91660, 98150, 98964, 100000, 98988, 100000, 108923, 105000, 103117]
```

```
plt.plot(godine, python_zarada, label='Python')
plt.plot(godine, javascript_zarada, label='JavaScript')
plt.plot(godine, proseckna_zarada, label='Prosek svih developera')

plt.xlabel('Godine')
plt.ylabel('Godisnja zarada u USD (USA)')
plt.title('Godisnja zarada developera u USD po godini (USA)')

plt.legend()
plt.show()
```

## PRIMER: GODIŠNJA ZARADA DEVELOPERA IZ 2019. GODINE.

*Sledi autorski video o iscrtavanju više krivih na istoj figuri.*

Slika 2.4 Prikaz programa za prosečne zadate u SAD. [Izvor: Autor]

Slika 2.5 Vizuelizacija godišnje zarade po godinama po programskom jeziku. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

# PROMENE PRIKAZA KRIVE U FIGURI

*Moguće je promeniti boju marker, boju, i izgled krive kroz string fmt*

Moguće je promeniti boju marker, boju, i izgled krive.

Prilikom poziva **plot()** funkcije, pored podataka po osi i labele, moguće je dodatno ubaciti string za formatiranje krive **fmt**

```
# formatiranje krive
from matplotlib import pyplot as plt

...
plt.plot(x_podaci, y_podaci, fmt, label)
```

String **fmt** je sledećeg oblika:

```
fmt = '[marker][line][color]'
```

**Marker** označava tip markera za svaku tačku. **line** označava kakva će linija biti (puna, isprekidana i sl.), dok **color** označava boju linije.

Potpun spisak opcionog formatiranja linije dat je na matplotlib sajtu.

Sledi spisak najčešće korišćenih markera, linija i boja:

```
# spisak markera, linija, i boja
# Markeri
'.' marker tacka
',' marker piksel
'o' marker krug
've' marker trougao na dole
'^' marker trougao na gore
'<' marker trougao na levo
'>' marker trougao na desno
# Linije
'--' puna linija
'-' isprekidana linija
'-.' crta-tacka linija
':.' tackasta linija
# Boje
'b' plava
'g' zelena
'r' crvena
'y' zuta
'k' crna
'w' bela
```

## PRIMER: TRIGONOMETRIJSKE FUNKCIJE

*Sledi primer sa formatiranjem krive u figuri, kroz primer prikaza trigonometrijskih funkcija.*

**Primer (10 minuta):**

Napisati program koji će računati **sinus, kosinus i zbir sinusa i kosinusa** vrednosti **x** (**x** je lista od 0 do 1000)

Nakon toga, iscrtati krive za sve tri trigonometrijske funkcije, i podesiti da je *zbir sinusa i kosinusa isprekidana crvena linija*.

```
import numpy as np
from matplotlib import pyplot as plt

x = np.arange(0,1001, 10)

y_sin = np.sin(np.deg2rad(x))
y_cos = np.cos(np.deg2rad(x))

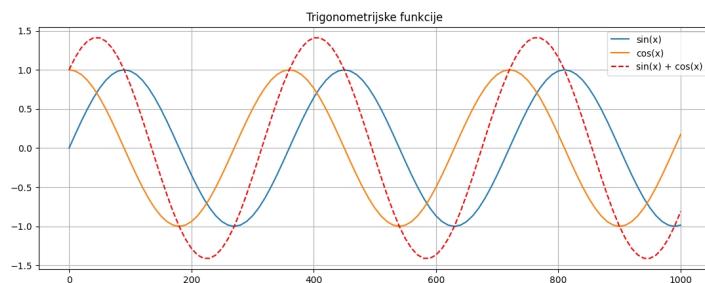
y_sum = y_sin + y_cos

plt.plot(x,y_sin, label='sin(x)')
plt.plot(x,y_cos, label='cos(x)')
plt.plot(x,y_sum, '--r', label='sin(x) + cos(x)')

plt.title('trigonometrijske funkcije')
plt.legend()

plt.grid()

plt.show()
```



Slika 2.6 Izgled izlaza primera sa trigonometrijskim funkcijama. [Izvor: Autor]

Slika 2.7 IsCRTavanje trigonometrijskih funkcija. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## DODATNO FORMATIRANJE

*Moguće je izvršiti napredno formatiranje, navođenjem pojedinih atributa za svaku krivu.*

Osim osnovnog formatiranja markera, tipa i boje linije kroz **fmt** string, moguće je izvršiti napredno formatiranje, navođenjem pojedinih atributa za svaku krivu.

Sledi tabela sa atributima i vrednosti koje mogu uzimati

# dodatno formatiranje	
Atribut	Vrednosti
linestyle ili ls	['solid'   'dashed', 'dashdot', 'dotted'   (offset, on-off-dash-seq)   '-'   '--'   '-.'   ':'   'None'   ' '   '')]
linewidth ili lw	float vrednost
marker	Stil za marker
markeredgecolor ili mec	Bilo koja matplotlib boja
markeredgewidth ili mew	float vrednost
markerfacecolor ili mfc	Bilo koja matplotlib boja
markersize ili ms	float vrednost

Pojedini atributi imaju skraćenice te nije neophodno koristiti puno ime atributa.

Boje se mogu uneti kao string "**ime\_boje**" ili kao string koja ima heksadekadnu vrednost vrednost boje.

## STIL FIGURE

*Moguće je promeniti opšti stil figure funkcijom **style.use()**, gde je parametar koji se prodleđuje ime stila.*

### Stil figure

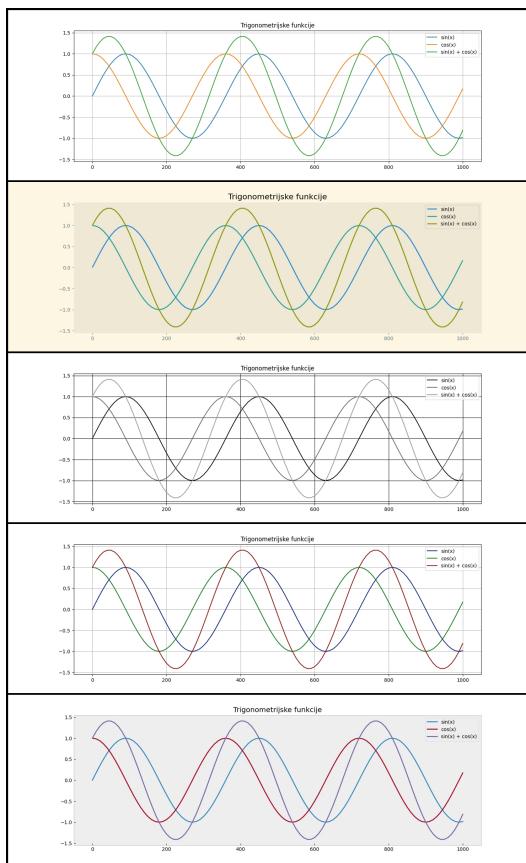
Moguće je promeniti opšti stil figure. Najpre treba pogledati moguće stilove, što se radi pozivom atributa **style.available**

```
from matplotlib import pyplot as plt
print(plt.style.available)
```

Nakon prikaza mogućih stilova, odabir željenog stila može se izvršiti funkcijom **style.use()**

```
from matplotlib import pyplot as plt
plt.style.use("ime stila")
```

U nastavku je primer sa trigonometrijskim funkcijama sa različitim stilovima.



Slika 2.8 Različiti stilovi iste figure. [Izvor: Autor]

Slika 2.9 Promena stila figure unutar programa. [Izvor: Autor]

```
# promena stila figure
from matplotlib import pyplot as plt

...
print(plt.style.available)
plt.style.use("dark_background")
```

## ČUVANJE FIGURE

*Figura se može direktno sačuvati pozivom savefig() funkcije, kojoj se prenosi ime i putanje figure.*

### Čuvanje figure

Moguće je sačuvati figuru direktno iz programa pozivom funkcije **savefig()**, čiji je parametar ime (i putanja) figure.

```
# cuvanje figure
from matplotlib import pyplot as plt

...
plt.savefig("ime_figure.png")
```

### Primer (8 minuta)

Napisati program koji će generisati više nizova od po 50 nasumičnih (float) vrednosti korišćenjem numpy paketa. Niz **x\_1** imaće vrednosti od -10 do -5. **x\_2** imaće vrednosti od -2 do 2, a **x\_3** vrednosti od 5 do 8.

Zatim, iscrtati sve vrednosti na grafikonu, tako da **x\_1** ima debljinu 3, crne boje, i isprekidanu liniju, **x\_2** ima debljinu 2, plave boje i punu liniju, a niz **x\_3** ima samo tačkaste markere crvene boje.

Namestiti da opseg osa bude od 0 do 100 po x-osi, i od -10 do 8 po y-osi.

Sačuvati figuru kao "**random\_seed.png**"

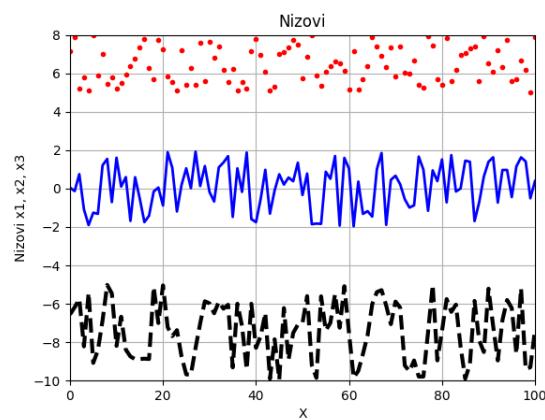
```
import numpy as np
from matplotlib import pyplot as plt

x = np.arange(0,101)
x_1 = np.random.uniform(-10, -5, x.shape)
x_2 = np.random.uniform(-2, 2, x.shape)
x_3 = np.random.uniform(5, 8, x.shape)

plt.plot(x,x_1, linewidth="3", linestyle="--", color="k", label="niz x1")
plt.plot(x,x_2, linewidth="2", linestyle="-", color="b", label="niz x2")
plt.plot(x,x_3, ".r", label="niz x3")

plt.xlabel("X")
plt.ylabel("Nizovi x1, x2, x3")
plt.grid()
plt.title("Nizovi")

plt.axis([0, 100, -10, 8])
plt.savefig("random_seed.png")
plt.show()
```



Slika 2.10 Random brojevi. [Izvor: Autor]

## ✓ Poglavlje 3

# Grafikoni i vrste grafikona

## ALTERNATIVNI NAČINI PRIKAZA PODATAKA

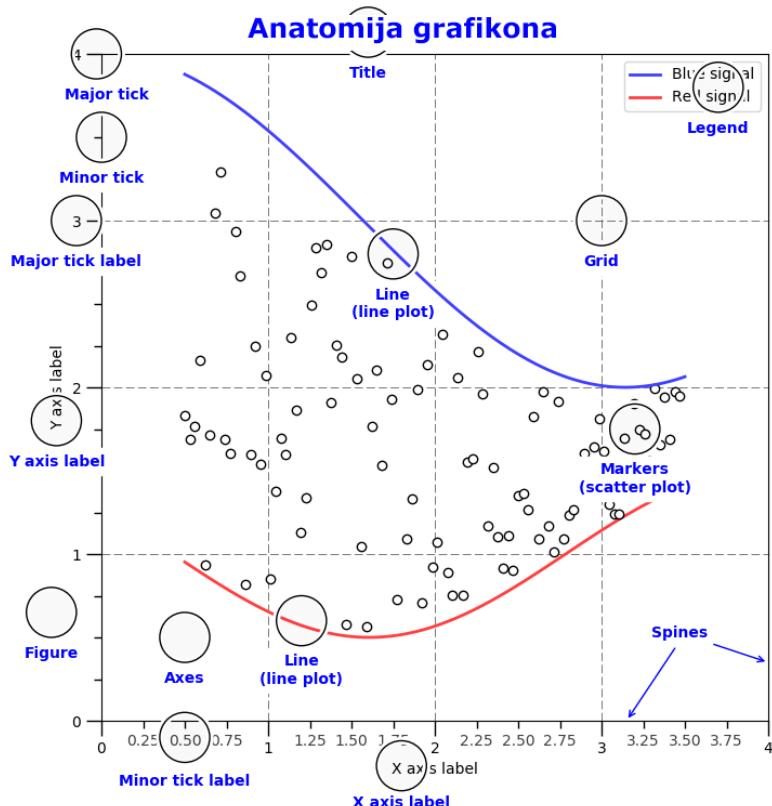
*Osim linijskog grafikona, moguće je napraviti i druge 2D grafikone. Najčešće korišćeni jesu bar plot, horizontalni bar plot, i kružni (pie) grafikon*

Funkcija plot podrazumevano pravi linijske grafikone, ali to nije jedini tip grafikona koji je dostupan.

### Anatomija grafikona

Sve dodatna polja prilikom poziva neke od `pyplot` funkcija data su na slici 1 sa imenima na engleskom, jer se tako i označavaju same funkcije.

U nastavku biće reči o bar plotovima kao i o kružnom (pie) grafikonu, budući da se pri analizi podataka ovi tipovi grafikona najčešće i koriste.



Slika 3.1 Anatomija grafikona. [Izvor: matplotlib.org]

## BAR PLOT

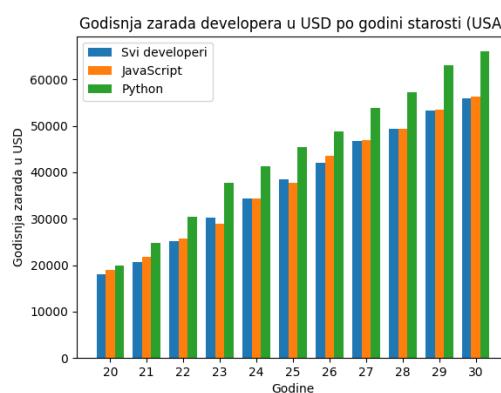
*Vertikalni bar grafikon može se napraviti pozivom plt.bar() funkcije.*

### Bar grafikon

Bar grafikon se postavlja pozivom funkcije **bar()**

#### Primer (8 minuta)

Korišćenjem podataka za prosečne zarade, napraviti program koji će iscrtavati **bar** grafikone, jedan pored drugog.



Slika 3.2 Bar grafikoni. [Izvor: Autor]

```
from matplotlib import pyplot as plt
import numpy as np

godine = [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

g_idx = np.arange(len(godine))
sirina_bar = 0.25

python_zarada = [20000, 24744, 30500, 37732, 41247, 45372, 48876, 53850, 57287,
63016, 65998]
javascript_zarada = [18942, 21780, 25704, 29000, 34372, 37810, 43515, 46823, 49293,
53437, 56373]
prosecna_zarada = [18012, 20628, 25206, 30252, 34368, 38496, 42000, 46752, 49320,
53200, 56000]

plt.bar(g_idx - sirina_bar, prosecna_zarada, width=sirina_bar, label='Svi
developeri')
plt.bar(g_idx, javascript_zarada, width=sirina_bar, label='JavaScript')
plt.bar(g_idx + sirina_bar, python_zarada, width=sirina_bar, label='Python')

plt.xticks(ticks=g_idx, labels=godine)
```

```
plt.xlabel('Godine')
plt.ylabel('Godisnja zarada u USD')
plt.title('Godisnja zarada developera u USD po godini starosti (USA)')

plt.legend()

plt.show()
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## HORIZONTALNI BAR PLOT

*Horizontalni bar grafikon može se napraviti pozivom plt.barh() funkcije.*

### Horizontalni bar grafikon

Horizontalni bar grafikon se postavlja pozivom funkcije **barh()**

```
# horizontal bar graph
from matplotlib import pyplot as plt

...
plt.barh(...)
```

### Primer (10 minuta)

Korišćenjem iz CSV datoteke na adresi:

[https://github.com/CoreyMSchafer/code\\_snippets/tree/master/Python/Matplotlib/02-BarCharts](https://github.com/CoreyMSchafer/code_snippets/tree/master/Python/Matplotlib/02-BarCharts)

napraviti program koji će iscrtavati **barh** grafikone. Deo podataka je dat i u nastavku.

```
Responder_id,LanguagesWorkedWith
1,HTML/CSS Java JavaScript Python
2,C++ HTML/CSS Python
3,HTML/CSS
4,C C++ C# Python SQL
5,C++ HTML/CSS Java JavaScript Python SQL VBA
6,Java R SQL
7,HTML/CSS JavaScript
8,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Python SQL
9,Bash/Shell/PowerShell C# HTML/CSS JavaScript Python Ruby Rust SQL TypeScript
WebAssembly Other(s):
10,C# Go JavaScript Python R SQL
11,Other(s):
```

12, Bash/Shell/PowerShell HTML/CSS Java Python R SQL  
13, Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL TypeScript  
14, C++  
15, Assembly Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript PHP SQL  
16, Bash/Shell/PowerShell C# HTML/CSS JavaScript TypeScript VBA  
17, Bash/Shell/PowerShell HTML/CSS JavaScript TypeScript  
18, Python R  
19, C# HTML/CSS Java JavaScript SQL TypeScript  
20, Bash/Shell/PowerShell C# HTML/CSS Java JavaScript PHP Python R SQL  
21, Assembly Bash/Shell/PowerShell C C++ Go Java JavaScript Kotlin Python Rust SQL  
Swift  
22, Bash/Shell/PowerShell C++ HTML/CSS JavaScript Python Ruby SQL TypeScript  
23, Bash/Shell/PowerShell HTML/CSS JavaScript Python Ruby SQL  
24, HTML/CSS JavaScript PHP TypeScript  
25, HTML/CSS JavaScript PHP SQL TypeScript  
26, Bash/Shell/PowerShell C++ C# HTML/CSS JavaScript PHP Python Ruby SQL Swift  
TypeScript VBA  
27, C++ JavaScript Python Ruby SQL TypeScript  
28, JavaScript TypeScript  
29, Bash/Shell/PowerShell JavaScript SQL  
31, Python  
32, Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python  
33, C++ Python R  
34, HTML/CSS JavaScript  
35, HTML/CSS JavaScript  
36, Java Kotlin Python  
37, Bash/Shell/PowerShell JavaScript Python Other(s):  
38, C# HTML/CSS JavaScript SQL  
39, C# JavaScript SQL TypeScript  
40, C# HTML/CSS  
41, Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript SQL  
42, HTML/CSS JavaScript PHP TypeScript  
43, C++ C# HTML/CSS Java JavaScript Objective-C SQL  
44, Bash/Shell/PowerShell C# HTML/CSS Java JavaScript SQL TypeScript WebAssembly  
45, Python  
46, Bash/Shell/PowerShell C C# HTML/CSS JavaScript PHP Python SQL Other(s):  
47, Java PHP Ruby  
48, HTML/CSS PHP SQL  
49, Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP Python SQL TypeScript  
50, Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript PHP Python SQL  
51, Bash/Shell/PowerShell C++ HTML/CSS Java JavaScript Python R TypeScript  
52, Bash/Shell/PowerShell C C++ Elixir Erlang Go HTML/CSS Java JavaScript Kotlin  
Python Ruby Rust SQL TypeScript  
53, Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP SQL TypeScript  
54, Bash/Shell/PowerShell HTML/CSS JavaScript Python  
55, Java Python SQL  
56, Bash/Shell/PowerShell HTML/CSS Java SQL  
57, JavaScript Python  
58, C# Java SQL  
59, HTML/CSS JavaScript PHP SQL  
60, Bash/Shell/PowerShell Go JavaScript PHP Python Ruby SQL  
61, C++ C# HTML/CSS JavaScript PHP Python SQL VBA  
62, Bash/Shell/PowerShell C++ Go HTML/CSS Java JavaScript Kotlin PHP Python Ruby SQL

TypeScript VBA

63,Bash/Shell/PowerShell Clojure Java Python Other(s):  
64,Bash/Shell/PowerShell C C++ C#  
65,Assembly C C++ C# HTML/CSS Java  
66,Clojure Go HTML/CSS Java JavaScript R SQL  
67,C C# HTML/CSS Java JavaScript PHP Python SQL VBA  
68,HTML/CSS Java JavaScript Python  
69,C# HTML/CSS Java JavaScript Objective-C SQL TypeScript  
70,C# HTML/CSS JavaScript SQL  
71,HTML/CSS JavaScript PHP Python SQL VBA  
72,C# HTML/CSS JavaScript PHP SQL TypeScript  
73,SQL  
74,HTML/CSS Java JavaScript Kotlin Python Ruby  
75,HTML/CSS JavaScript  
76,PHP SQL  
77,HTML/CSS JavaScript PHP SQL  
78,HTML/CSS Java JavaScript Kotlin Python  
79,C# HTML/CSS JavaScript TypeScript  
80,Bash/Shell/PowerShell C# F# Go HTML/CSS Java JavaScript Python SQL TypeScript  
81,Assembly Bash/Shell/PowerShell C C++ Python  
82,Bash/Shell/PowerShell C++ HTML/CSS Java JavaScript Python Rust  
83,HTML/CSS JavaScript  
84,C C++ C# Java Kotlin PHP SQL  
85,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL TypeScript  
86,Bash/Shell/PowerShell C# HTML/CSS JavaScript PHP Python SQL  
87,Bash/Shell/PowerShell C C++ C# Go HTML/CSS Java JavaScript Objective-C Python SQL  
88,C++ C# HTML/CSS Java JavaScript SQL TypeScript  
89,Bash/Shell/PowerShell Python  
90,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript Objective-C PHP Python Ruby  
SQL Swift TypeScript  
91,HTML/CSS JavaScript TypeScript  
92,HTML/CSS Java JavaScript Kotlin SQL VBA  
93,Python SQL  
94,C# HTML/CSS JavaScript SQL  
95,C# HTML/CSS JavaScript Python R SQL  
96,C C++ Java Python R Scala SQL  
97,Bash/Shell/PowerShell JavaScript Python R SQL  
98,HTML/CSS Java JavaScript SQL  
99,C C++ HTML/CSS Java JavaScript Kotlin PHP Python SQL  
100,HTML/CSS JavaScript Ruby SQL TypeScript  
101,Bash/Shell/PowerShell HTML/CSS JavaScript SQL  
102,C# HTML/CSS JavaScript SQL TypeScript  
103,Clojure Go Java Kotlin  
104,C# HTML/CSS TypeScript  
105,Bash/Shell/PowerShell HTML/CSS JavaScript SQL  
106,Bash/Shell/PowerShell HTML/CSS Java JavaScript Python SQL  
107,HTML/CSS Java Python  
108,C++ Python  
109,C# HTML/CSS JavaScript SQL  
110,Python SQL  
111,Bash/Shell/PowerShell C# HTML/CSS JavaScript Python SQL VBA  
112,Assembly C C++ C# HTML/CSS Java JavaScript Python VBA  
113,Bash/Shell/PowerShell C C++

114, Bash/Shell/PowerShell C++ Erlang JavaScript PHP Python  
115, Assembly C# HTML/CSS Java JavaScript SQL Swift  
116, Scala Other(s):  
117, C# HTML/CSS JavaScript SQL TypeScript  
118, Bash/Shell/PowerShell C C++  
119, C C++ C# HTML/CSS Java JavaScript SQL  
120, Elixir HTML/CSS JavaScript Python Ruby SQL  
121, Bash/Shell/PowerShell C C++ HTML/CSS JavaScript Python  
122, Assembly Bash/Shell/PowerShell C++ Python SQL  
123, HTML/CSS JavaScript TypeScript  
124, HTML/CSS Java SQL  
125, Bash/Shell/PowerShell Dart HTML/CSS Java JavaScript Scala  
126, C# SQL  
127, Bash/Shell/PowerShell C Python  
128, Bash/Shell/PowerShell Go Ruby  
129, C++ Python R  
130, Bash/Shell/PowerShell C++ Clojure HTML/CSS Java Python Ruby SQL  
131, Java  
132, Bash/Shell/PowerShell C++ C# HTML/CSS Java JavaScript Objective-C Python  
TypeScript  
133, JavaScript PHP SQL  
134, C C++ HTML/CSS JavaScript SQL  
135, Go HTML/CSS JavaScript TypeScript  
136, C# Java PHP Python  
137, HTML/CSS Java JavaScript PHP SQL  
139, Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL  
140, Java JavaScript Kotlin PHP SQL  
141, Assembly Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Python SQL  
TypeScript  
142, Bash/Shell/PowerShell C#  
143, C HTML/CSS Java JavaScript Python SQL TypeScript  
144, Java SQL  
145, Assembly C++ Python VBA  
146, Bash/Shell/PowerShell C Python Scala  
147, HTML/CSS Java JavaScript Kotlin  
148, HTML/CSS JavaScript TypeScript  
149, Bash/Shell/PowerShell HTML/CSS JavaScript Python SQL  
150, Bash/Shell/PowerShell C# HTML/CSS Java JavaScript Python SQL  
151, Bash/Shell/PowerShell C++ C# HTML/CSS Java JavaScript SQL TypeScript  
152, HTML/CSS Java JavaScript Kotlin Objective-C Python Swift Other(s):  
153, C C++ Python  
154, HTML/CSS Java JavaScript PHP SQL VBA  
155, Bash/Shell/PowerShell HTML/CSS JavaScript TypeScript  
156, Bash/Shell/PowerShell Python  
157, Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL  
158, Java SQL Other(s):  
159, Bash/Shell/PowerShell HTML/CSS JavaScript Python  
160, C#  
161, HTML/CSS JavaScript PHP  
162, Bash/Shell/PowerShell C++ HTML/CSS Java JavaScript PHP SQL  
163, C# HTML/CSS JavaScript SQL Other(s):  
164, C# HTML/CSS JavaScript SQL  
165, Assembly Bash/Shell/PowerShell C C++ C# JavaScript Python SQL

166,Bash/Shell/PowerShell Go HTML/CSS Java JavaScript Ruby SQL  
167,HTML/CSS Java JavaScript PHP Python SQL  
168,Bash/Shell/PowerShell Python  
169,C C++ HTML/CSS Java JavaScript PHP Python SQL  
170,Clojure Go HTML/CSS Java JavaScript Python Ruby TypeScript  
171,Assembly C C++ C# HTML/CSS Java Objective-C PHP Python R SQL Swift  
172,HTML/CSS JavaScript Objective-C SQL  
173,HTML/CSS JavaScript PHP SQL TypeScript  
174,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript PHP Python R  
175,Assembly Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript PHP Python SQL  
176,C C++ C# HTML/CSS Java JavaScript Python SQL  
177,C# Python SQL  
178,JavaScript PHP Python Ruby SQL TypeScript  
179,HTML/CSS Java JavaScript Objective-C SQL Swift  
181,HTML/CSS JavaScript  
182,Bash/Shell/PowerShell C C++ Go HTML/CSS Java JavaScript PHP Python Rust SQL  
TypeScript  
183,Dart Go Java Kotlin Swift  
184,Bash/Shell/PowerShell C C++ Java JavaScript Kotlin Objective-C Python Rust Swift  
185,Bash/Shell/PowerShell Java Kotlin  
186,Assembly C C++ HTML/CSS Java JavaScript Python  
187,HTML/CSS JavaScript Ruby Scala SQL  
188,C# Python R SQL VBA  
189,C++ HTML/CSS JavaScript Python SQL  
190,Java Scala SQL TypeScript  
191,C# HTML/CSS JavaScript SQL TypeScript  
192,Bash/Shell/PowerShell JavaScript Python  
193,Bash/Shell/PowerShell HTML/CSS JavaScript Ruby SQL  
194,Assembly Bash/Shell/PowerShell C HTML/CSS Java JavaScript PHP  
195,C# TypeScript Other(s):  
196,C# HTML/CSS JavaScript SQL TypeScript  
197,HTML/CSS R VBA  
198,HTML/CSS Java JavaScript Python SQL  
199,R SQL  
200,Java JavaScript SQL VBA  
201,Bash/Shell/PowerShell Go HTML/CSS Java JavaScript Python Scala  
202,Bash/Shell/PowerShell C C++ C# HTML/CSS Java JavaScript Objective-C Python SQL  
TypeScript  
203,C C++ HTML/CSS JavaScript PHP SQL  
204,Python Other(s):  
205,C# HTML/CSS JavaScript PHP  
206,Scala SQL  
207,Java  
208,Other(s):  
209,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL  
210,C++ Go Java JavaScript Python Rust WebAssembly  
211,C# HTML/CSS Java JavaScript Objective-C SQL  
212,C# HTML/CSS Java JavaScript PHP  
213,Go Python  
214,R Other(s):  
215,Python Rust  
216,Go Java Python  
217,Go HTML/CSS JavaScript PHP TypeScript

218,HTML/CSS JavaScript PHP SQL  
219,C# HTML/CSS JavaScript SQL  
220,C# HTML/CSS JavaScript SQL  
221,C C++ HTML/CSS Java JavaScript PHP SQL  
222,Assembly Bash/Shell/PowerShell C HTML/CSS JavaScript Other(s):  
223,Java Kotlin  
224,Go  
225,Bash/Shell/PowerShell C C++ C# HTML/CSS Java JavaScript PHP Python R SQL  
226,HTML/CSS JavaScript PHP Python SQL  
227,Bash/Shell/PowerShell HTML/CSS JavaScript Python SQL  
228,HTML/CSS Java JavaScript Python Scala SQL  
229,HTML/CSS Java JavaScript Python TypeScript  
230,Bash/Shell/PowerShell Go Python  
231,Elixir Go JavaScript Ruby SQL  
232,Bash/Shell/PowerShell C# Dart HTML/CSS JavaScript  
233,C++ Python  
234,Bash/Shell/PowerShell C Python  
235,HTML/CSS Java Python SQL  
236,HTML/CSS JavaScript  
237,C++ JavaScript Rust  
238,HTML/CSS JavaScript Objective-C PHP Python  
239,C# HTML/CSS JavaScript TypeScript  
240,Bash/Shell/PowerShell HTML/CSS Java JavaScript Python SQL  
241,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL  
242,HTML/CSS Java JavaScript SQL TypeScript  
243,C C++ C# HTML/CSS Java JavaScript Objective-C PHP Python Swift  
244,Assembly Bash/Shell/PowerShell C Python VBA  
245,C# JavaScript TypeScript  
246,C# HTML/CSS JavaScript SQL TypeScript  
247,C# C#  
248,HTML/CSS Java JavaScript Python SQL  
249,Bash/Shell/PowerShell C# HTML/CSS Java Python SQL  
250,Python Other(s):  
251,Assembly C C++ C# HTML/CSS JavaScript Python VBA  
252,HTML/CSS JavaScript Rust Swift Other(s):  
253,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Ruby SQL  
254,C# Python SQL TypeScript  
255,Bash/Shell/PowerShell C HTML/CSS Java JavaScript Python TypeScript Other(s):  
256,C C++ Java  
257,Bash/Shell/PowerShell Other(s):  
258,C HTML/CSS Java JavaScript PHP Python  
259,C# HTML/CSS Java JavaScript Python SQL TypeScript  
260,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP SQL  
261,HTML/CSS JavaScript  
262,C++ C# HTML/CSS Java JavaScript PHP Python SQL TypeScript  
263,HTML/CSS JavaScript Kotlin Python  
264,HTML/CSS Java JavaScript Python SQL Other(s):  
265,HTML/CSS JavaScript  
266,HTML/CSS JavaScript TypeScript Other(s):  
267,HTML/CSS JavaScript Python Ruby  
268,C Java PHP Ruby  
269,C# HTML/CSS Java JavaScript SQL Swift  
270,C++ HTML/CSS JavaScript

271,C# JavaScript SQL  
272,HTML/CSS JavaScript PHP  
273,Bash/Shell/PowerShell C++ Clojure Erlang HTML/CSS Java JavaScript Ruby Scala  
274,Java JavaScript  
275,C# HTML/CSS Java JavaScript Objective-C Python Ruby Rust SQL TypeScript  
276,HTML/CSS Java JavaScript Scala SQL  
277,Bash/Shell/PowerShell C C++ Python  
278,Bash/Shell/PowerShell C# HTML/CSS JavaScript PHP Python SQL  
279,HTML/CSS Java JavaScript PHP Python SQL  
280,C# HTML/CSS JavaScript SQL TypeScript  
281,C# HTML/CSS Java JavaScript TypeScript  
282,Python SQL Swift  
283,HTML/CSS JavaScript TypeScript  
284,Assembly Bash/Shell/PowerShell C C++ C# Java  
285,C# HTML/CSS JavaScript PHP SQL  
286,Dart HTML/CSS JavaScript Python TypeScript Other(s):  
287,Java Kotlin  
288,HTML/CSS JavaScript PHP SQL  
289,C++ C# PHP Rust TypeScript  
290,HTML/CSS PHP Python  
291,Go HTML/CSS Python SQL TypeScript  
292,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL TypeScript  
293,HTML/CSS Java JavaScript SQL  
295,C C++ C# HTML/CSS Java JavaScript Kotlin PHP Python Ruby SQL TypeScript  
WebAssembly  
296,HTML/CSS Java JavaScript SQL  
297,HTML/CSS Java JavaScript SQL TypeScript  
298,HTML/CSS JavaScript PHP SQL Other(s):  
299,Assembly C C++ Java Python SQL  
300,HTML/CSS Java JavaScript SQL  
301,C++ C# Clojure JavaScript PHP Python SQL VBA  
302,Bash/Shell/PowerShell C# Python Other(s):  
303,JavaScript  
304,Bash/Shell/PowerShell Go HTML/CSS JavaScript Python Rust SQL  
305,C# HTML/CSS JavaScript SQL  
306,C HTML/CSS JavaScript Python SQL TypeScript  
307,Assembly Bash/Shell/PowerShell C HTML/CSS Java PHP Python R SQL  
308,HTML/CSS JavaScript Ruby  
309,Java  
310,C# HTML/CSS JavaScript SQL Swift  
311,Assembly C C++ C# HTML/CSS Java Python Scala SQL  
312,Bash/Shell/PowerShell Python Ruby SQL  
313,C C# HTML/CSS PHP Python  
314,C Python  
315,C++ HTML/CSS Java JavaScript Python SQL Other(s):  
316,Bash/Shell/PowerShell C++ HTML/CSS Java JavaScript Python  
317,Java SQL  
318,Bash/Shell/PowerShell Java  
319,Bash/Shell/PowerShell HTML/CSS JavaScript PHP TypeScript  
320,C# HTML/CSS JavaScript SQL TypeScript  
321,HTML/CSS Java JavaScript Ruby SQL TypeScript  
322,C# HTML/CSS Java Python  
323,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL

324,C# HTML/CSS JavaScript SQL TypeScript  
325,Assembly Bash/Shell/PowerShell C C++ C# HTML/CSS Java JavaScript PHP Python  
326,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP Python SQL TypeScript  
327,HTML/CSS Java JavaScript PHP Python SQL  
328,HTML/CSS Java JavaScript TypeScript  
329,C C++ C# F# HTML/CSS JavaScript Rust SQL TypeScript Other(s):  
330,Java Kotlin  
331,Bash/Shell/PowerShell C++ C# Python  
332,Bash/Shell/PowerShell HTML/CSS JavaScript Objective-C Python SQL Swift  
TypeScript  
333,Python Rust Scala  
334,Bash/Shell/PowerShell C C++ Python  
335,C# HTML/CSS Java JavaScript Python SQL Swift  
336,Bash/Shell/PowerShell JavaScript Python Rust SQL TypeScript  
337,Python R  
338,Bash/Shell/PowerShell HTML/CSS Java Python SQL  
339,Bash/Shell/PowerShell Dart Go HTML/CSS Java JavaScript Python Rust SQL Swift  
340,C# HTML/CSS SQL  
341,C++ C# SQL  
342,HTML/CSS Java JavaScript Python SQL TypeScript  
343,C++  
344,Assembly C# HTML/CSS JavaScript SQL  
345,Java JavaScript  
346,Java  
347,Java  
348,HTML/CSS PHP  
349,C C++ HTML/CSS Python  
350,C C++ Elixir HTML/CSS Java Kotlin PHP Ruby SQL Other(s):  
351,Assembly C C++ Java JavaScript Python SQL  
352,HTML/CSS JavaScript Ruby TypeScript  
353,HTML/CSS JavaScript TypeScript  
354,HTML/CSS Java JavaScript Kotlin TypeScript  
355,Bash/Shell/PowerShell Go HTML/CSS Java JavaScript Kotlin TypeScript  
356,HTML/CSS JavaScript PHP Python SQL  
357,HTML/CSS TypeScript  
358,C C++ C# HTML/CSS JavaScript PHP SQL TypeScript  
359,Java JavaScript Python  
360,Bash/Shell/PowerShell HTML/CSS JavaScript Python TypeScript  
361,Elixir Erlang F# Go HTML/CSS JavaScript PHP SQL  
362,HTML/CSS Java JavaScript SQL  
363,Bash/Shell/PowerShell HTML/CSS Java JavaScript Objective-C Python SQL TypeScript  
364,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL TypeScript  
365,Java SQL  
366,HTML/CSS PHP SQL  
367,HTML/CSS JavaScript  
368,Go HTML/CSS JavaScript PHP Python SQL TypeScript  
369,Python  
370,R SQL  
371,Bash/Shell/PowerShell SQL  
372,Bash/Shell/PowerShell C# HTML/CSS Java Python SQL  
373,C# HTML/CSS Java JavaScript Python  
374,HTML/CSS Python SQL TypeScript  
375,Bash/Shell/PowerShell HTML/CSS Java JavaScript Python SQL

376,C++ Python  
377,HTML/CSS JavaScript PHP SQL  
378,C# HTML/CSS JavaScript SQL TypeScript  
379,Bash/Shell/PowerShell C C# Java Python  
380,Bash/Shell/PowerShell JavaScript Python  
381,C# Go HTML/CSS Java JavaScript Python R SQL  
382,C# HTML/CSS Java JavaScript PHP Python SQL  
383,Bash/Shell/PowerShell  
384,Bash/Shell/PowerShell Go SQL TypeScript  
385,HTML/CSS Java JavaScript Python R SQL VBA  
386,HTML/CSS JavaScript PHP  
387,Java Python  
388,C HTML/CSS Java JavaScript PHP SQL TypeScript  
389,Bash/Shell/PowerShell JavaScript Objective-C Ruby Swift  
390,Bash/Shell/PowerShell HTML/CSS Java Kotlin Python Scala  
391,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL  
392,Bash/Shell/PowerShell C# HTML/CSS JavaScript Python SQL TypeScript VBA  
393,Java Kotlin PHP SQL  
394,C# HTML/CSS TypeScript Other(s):  
395,HTML/CSS Java JavaScript SQL  
396,HTML/CSS JavaScript  
397,HTML/CSS JavaScript PHP Ruby Rust SQL  
398,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Python SQL VBA  
399,C++  
400,Bash/Shell/PowerShell C++ Java Kotlin PHP  
401,JavaScript Python Ruby SQL TypeScript  
402,C#  
403,Bash/Shell/PowerShell C C# F# HTML/CSS JavaScript PHP Python SQL  
404,C# SQL TypeScript  
405,Java JavaScript TypeScript  
406,HTML/CSS Java JavaScript PHP SQL TypeScript  
407,C C++ HTML/CSS Java JavaScript Python SQL  
408,Assembly Bash/Shell/PowerShell C C++ Go HTML/CSS Java JavaScript Python R Rust  
SQL TypeScript WebAssembly  
409,Bash/Shell/PowerShell C# Go HTML/CSS JavaScript PHP  
410,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL  
411,C C++ JavaScript Python R SQL  
412,HTML/CSS JavaScript PHP Python SQL  
413,Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL TypeScript  
414,Bash/Shell/PowerShell JavaScript Python SQL  
415,HTML/CSS JavaScript Ruby  
416,C C++ C# HTML/CSS JavaScript SQL TypeScript  
417,HTML/CSS JavaScript  
418,HTML/CSS JavaScript SQL VBA  
419,C C++ C# HTML/CSS Java JavaScript PHP Python R SQL  
420,HTML/CSS JavaScript Python  
421,Bash/Shell/PowerShell HTML/CSS JavaScript TypeScript  
422,Bash/Shell/PowerShell C C# HTML/CSS JavaScript Python Other(s):  
423,C++ C#  
424,C C++  
425,Bash/Shell/PowerShell C HTML/CSS JavaScript PHP SQL VBA  
426,Java  
427,C# HTML/CSS JavaScript SQL TypeScript

428,Bash/Shell/PowerShell Java Python  
429,C#  
430,Assembly C++ C# HTML/CSS Java JavaScript Objective-C PHP SQL Other(s):  
431,VBA Other(s):  
432,HTML/CSS Java JavaScript SQL TypeScript VBA Other(s):  
433,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Python  
434,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL  
435,Bash/Shell/PowerShell C# HTML/CSS JavaScript Python SQL Swift  
436,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP Python SQL  
438,Bash/Shell/PowerShell C C++ Go HTML/CSS JavaScript R SQL  
439,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript PHP Python SQL TypeScript  
440,Bash/Shell/PowerShell C++ HTML/CSS Java PHP SQL  
441,C# HTML/CSS Java JavaScript PHP  
442,HTML/CSS JavaScript  
443,HTML/CSS JavaScript TypeScript  
444,Bash/Shell/PowerShell C C++ HTML/CSS Python  
445,C# SQL  
446,Bash/Shell/PowerShell HTML/CSS Java JavaScript Python  
447,Bash/Shell/PowerShell Java Python  
448,Bash/Shell/PowerShell Java SQL Other(s):  
449,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript TypeScript Other(s):  
450,C# HTML/CSS JavaScript TypeScript  
451,Bash/Shell/PowerShell C C++ C# Java Python Rust  
452,SQL VBA  
453,C++ C# SQL  
454,Bash/Shell/PowerShell Elixir HTML/CSS JavaScript Ruby SQL Other(s):  
455,C++ Python Ruby  
456,Bash/Shell/PowerShell JavaScript Python SQL  
457,HTML/CSS JavaScript  
458,Assembly C Python Rust  
459,C# SQL  
460,Bash/Shell/PowerShell C# HTML/CSS JavaScript PHP Python  
461,HTML/CSS JavaScript PHP SQL  
462,Bash/Shell/PowerShell C# HTML/CSS JavaScript Python SQL  
463,HTML/CSS Java JavaScript PHP SQL Other(s):  
464,C Python  
465,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL TypeScript  
466,HTML/CSS JavaScript TypeScript  
467,Java Kotlin Python SQL  
468,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript Kotlin SQL Other(s):  
469,Bash/Shell/PowerShell JavaScript PHP SQL  
470,HTML/CSS Java JavaScript PHP SQL TypeScript Other(s):  
471,Java JavaScript  
472,HTML/CSS JavaScript PHP Python SQL  
473,C# JavaScript  
474,Java JavaScript Other(s):  
475,C# HTML/CSS JavaScript PHP TypeScript  
476,HTML/CSS JavaScript PHP Scala  
477,Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL TypeScript  
478,C++ HTML/CSS JavaScript PHP Python Ruby SQL  
479,C# HTML/CSS JavaScript SQL Other(s):  
480,C C++ Dart Java Python  
481,Bash/Shell/PowerShell Kotlin

482,Bash/Shell/PowerShell HTML/CSS PHP  
483,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript PHP Python SQL  
484,HTML/CSS Java PHP Python SQL  
485,Java JavaScript Scala  
486,Bash/Shell/PowerShell HTML/CSS JavaScript TypeScript  
487,Bash/Shell/PowerShell C++ C# HTML/CSS JavaScript PHP SQL TypeScript  
488,C C++ C# Java  
489,C++ Python  
490,HTML/CSS JavaScript Ruby  
491,Java Objective-C  
492,Bash/Shell/PowerShell Go HTML/CSS JavaScript  
493,C# Java SQL Other(s):  
494,Assembly C Erlang HTML/CSS Java JavaScript Python SQL  
495,R  
496,Bash/Shell/PowerShell Python SQL  
497,Dart JavaScript Kotlin PHP SQL Swift TypeScript  
498,C++ HTML/CSS JavaScript Python  
499,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP Ruby SQL VBA  
500,C++ C#  
501,HTML/CSS  
502,C++ C#  
503,Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL TypeScript  
504,Bash/Shell/PowerShell Python SQL  
505,C C++ HTML/CSS Java SQL  
506,Java  
507,Go HTML/CSS Java JavaScript PHP Python SQL TypeScript  
508,C C++ HTML/CSS JavaScript  
509,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL  
510,Python  
511,HTML/CSS JavaScript Ruby SQL TypeScript  
512,Bash/Shell/PowerShell Python Other(s):  
513,C++ Rust  
514,HTML/CSS JavaScript Python Ruby TypeScript  
515,C++ HTML/CSS Java JavaScript PHP Python SQL  
516,Bash/Shell/PowerShell C++ Python  
517,Assembly C++ F# HTML/CSS PHP Python R Rust TypeScript Other(s):  
518,C HTML/CSS Java JavaScript SQL  
519,HTML/CSS PHP  
520,HTML/CSS Java JavaScript Python SQL TypeScript  
521,HTML/CSS JavaScript  
522,HTML/CSS Python  
523,Bash/Shell/PowerShell HTML/CSS JavaScript Python R  
524,Python Other(s):  
525,C# SQL VBA  
526,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL  
527,HTML/CSS JavaScript Objective-C Swift  
528,Bash/Shell/PowerShell Java JavaScript Python SQL  
529,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP Python SQL TypeScript  
530,C C++ C#  
531,C#  
532,Bash/Shell/PowerShell C C++ Java Python  
533,C# Java JavaScript SQL  
535,Elixir VBA

536,C# HTML/CSS Java JavaScript PHP Python SQL  
537,C# HTML/CSS JavaScript SQL TypeScript Other(s):  
538,HTML/CSS Java JavaScript PHP R SQL TypeScript  
539,Bash/Shell/PowerShell JavaScript Python SQL  
540,JavaScript Python  
541,Bash/Shell/PowerShell C# HTML/CSS JavaScript Python SQL TypeScript Other(s):  
542,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript Python R SQL  
543,C C# HTML/CSS Java JavaScript Kotlin PHP TypeScript  
544,Assembly Bash/Shell/PowerShell C C++ Go HTML/CSS JavaScript Objective-C PHP  
Python SQL  
545,Bash/Shell/PowerShell C++ C# Go HTML/CSS JavaScript PHP Python SQL TypeScript  
Other(s):  
546,C Java SQL  
547,C# HTML/CSS JavaScript SQL  
548,Python SQL VBA  
550,C# HTML/CSS JavaScript SQL  
551,Assembly Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Python R Scala SQL  
552,HTML/CSS Java JavaScript PHP  
553,C# HTML/CSS JavaScript Python SQL TypeScript  
554,Go JavaScript PHP Ruby SQL TypeScript  
555,C# HTML/CSS JavaScript SQL  
556,Bash/Shell/PowerShell HTML/CSS Java JavaScript Python SQL  
557,C C++ PHP Python SQL VBA  
558,Bash/Shell/PowerShell C HTML/CSS Java JavaScript Kotlin PHP SQL TypeScript  
559,Bash/Shell/PowerShell C C++ C# HTML/CSS Java PHP Python SQL VBA  
560,Bash/Shell/PowerShell C# HTML/CSS JavaScript PHP SQL  
561,HTML/CSS Java JavaScript Python Ruby SQL  
562,C# JavaScript SQL  
563,Assembly Bash/Shell/PowerShell C C++ Python SQL  
564,C C++ Python R SQL Other(s):  
565,Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL  
566,Bash/Shell/PowerShell C# HTML/CSS JavaScript  
567,Bash/Shell/PowerShell C Java JavaScript Objective-C Rust Swift Other(s):  
568,C++ C# Clojure HTML/CSS JavaScript Python SQL  
569,Bash/Shell/PowerShell HTML/CSS JavaScript Python  
571,Assembly Bash/Shell/PowerShell C C++ Erlang R Rust Scala  
572,Assembly C C++ Java JavaScript Kotlin PHP Python Rust SQL TypeScript  
573,Assembly Bash/Shell/PowerShell C HTML/CSS PHP Python SQL  
574,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python Ruby  
575,Bash/Shell/PowerShell Go Python SQL  
576,C# HTML/CSS JavaScript Ruby SQL TypeScript  
577,Bash/Shell/PowerShell Python Other(s):  
578,Bash/Shell/PowerShell HTML/CSS Java JavaScript SQL Swift  
579,Bash/Shell/PowerShell C++ C# HTML/CSS Java JavaScript Python  
580,Bash/Shell/PowerShell HTML/CSS Java JavaScript Python Scala SQL  
581,Bash/Shell/PowerShell HTML/CSS JavaScript PHP TypeScript  
582,C C++ C# HTML/CSS Java JavaScript PHP Python SQL  
583,Java Ruby Scala SQL Other(s):  
584,Java Python  
585,C# Java Python R VBA  
586,Java Python Ruby  
587,Bash/Shell/PowerShell C# Java Other(s):  
588,C#

589,C# HTML/CSS JavaScript SQL VBA  
590,C# SQL  
591,C# JavaScript  
592,C C++ HTML/CSS PHP Python SQL  
594,HTML/CSS Java JavaScript Kotlin PHP  
595,C# HTML/CSS JavaScript SQL TypeScript  
596,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Python SQL TypeScript  
597,Java JavaScript PHP Python SQL  
598,Assembly Bash/Shell/PowerShell C C++ C# HTML/CSS JavaScript PHP Python SQL  
599,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Ruby SQL  
600,Bash/Shell/PowerShell HTML/CSS Java PHP Python SQL  
601,HTML/CSS PHP SQL  
602,C++ C# HTML/CSS Java JavaScript PHP Python SQL Swift TypeScript  
603,Bash/Shell/PowerShell HTML/CSS Java JavaScript  
604,Bash/Shell/PowerShell C# HTML/CSS JavaScript PHP SQL TypeScript  
605,Java Kotlin  
606,Bash/Shell/PowerShell Clojure HTML/CSS Java JavaScript PHP Python Ruby  
TypeScript  
607,C# JavaScript SQL  
608,Bash/Shell/PowerShell C C++ Erlang Go HTML/CSS Java Python SQL  
609,HTML/CSS JavaScript Python  
610,Bash/Shell/PowerShell C C++ Go HTML/CSS Java JavaScript PHP Python Rust SQL  
TypeScript  
611,Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL  
612,Python Other(s):  
613,Assembly Bash/Shell/PowerShell C C++ C# F# HTML/CSS JavaScript Python TypeScript  
614,Bash/Shell/PowerShell C C++ C# HTML/CSS Java Python SQL TypeScript  
615,C# HTML/CSS JavaScript Python SQL TypeScript VBA  
616,Ruby SQL  
617,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript SQL  
618,C C++ C# Java PHP SQL  
619,Assembly Bash/Shell/PowerShell C C++ C# HTML/CSS Java JavaScript PHP SQL  
620,HTML/CSS Java JavaScript SQL TypeScript WebAssembly  
621,Bash/Shell/PowerShell SQL Other(s):  
622,Java Objective-C SQL Swift  
623,Objective-C Swift  
624,C++ HTML/CSS JavaScript PHP Python  
625,Bash/Shell/PowerShell HTML/CSS JavaScript Ruby TypeScript  
626,HTML/CSS Java JavaScript SQL  
627,Assembly Bash/Shell/PowerShell C++ C# Go HTML/CSS JavaScript Objective-C PHP  
Python Ruby SQL Swift  
628,Bash/Shell/PowerShell JavaScript SQL TypeScript  
629,Assembly Bash/Shell/PowerShell C C++ C# HTML/CSS Java Objective-C SQL  
630,C# HTML/CSS JavaScript SQL TypeScript  
631,Bash/Shell/PowerShell Go HTML/CSS JavaScript Python SQL  
632,C# HTML/CSS JavaScript SQL  
633,C C++ Python Rust  
634,C C++ C# HTML/CSS Java JavaScript Objective-C Scala SQL Swift TypeScript  
635,Bash/Shell/PowerShell JavaScript Python  
636,Bash/Shell/PowerShell C C++ Go Java  
637,C++ C# HTML/CSS Java JavaScript SQL  
638,Bash/Shell/PowerShell C# HTML/CSS JavaScript Rust  
639,HTML/CSS JavaScript Kotlin SQL Swift

640,HTML/CSS JavaScript SQL  
641,C# HTML/CSS Java JavaScript Kotlin PHP SQL Swift  
642,Bash/Shell/PowerShell C++ HTML/CSS JavaScript TypeScript  
643,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP Ruby SQL  
645,C++ C# HTML/CSS SQL  
646,JavaScript PHP Python SQL VBA  
647,Assembly Bash/Shell/PowerShell C C++ C# HTML/CSS Rust SQL TypeScript  
648,C# HTML/CSS JavaScript SQL VBA  
649,Java JavaScript Python SQL  
650,Bash/Shell/PowerShell C++ JavaScript Python  
651,Python  
652,HTML/CSS Java  
653,C# HTML/CSS JavaScript Kotlin Objective-C PHP SQL Swift  
654,Clojure HTML/CSS Java JavaScript  
655,Assembly C++ C# Java Scala Other(s):  
656,C++ Python R Other(s):  
657,Bash/Shell/PowerShell HTML/CSS Java JavaScript SQL  
658,C++ HTML/CSS Java JavaScript TypeScript  
659,Python  
660,HTML/CSS Java JavaScript Python SQL TypeScript  
661,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL  
662,Python SQL VBA  
663,HTML/CSS JavaScript Python SQL  
664,HTML/CSS Java JavaScript PHP  
665,Ruby SQL  
666,Bash/Shell/PowerShell HTML/CSS JavaScript Python SQL  
667,HTML/CSS JavaScript  
668,Bash/Shell/PowerShell HTML/CSS JavaScript Python SQL  
669,Bash/Shell/PowerShell C Python Rust SQL  
670,C# HTML/CSS JavaScript SQL Swift  
671,Bash/Shell/PowerShell HTML/CSS Java JavaScript Kotlin PHP Python  
672,C C++ HTML/CSS Java JavaScript PHP SQL  
673,C C++ C# HTML/CSS JavaScript PHP SQL  
674,C# HTML/CSS JavaScript Python  
675,C# HTML/CSS Java JavaScript SQL VBA Other(s):  
676,Bash/Shell/PowerShell Objective-C Ruby Swift  
677,Other(s):  
678,Java JavaScript Scala SQL  
679,C# HTML/CSS Python SQL  
680,C# HTML/CSS JavaScript SQL  
681,Bash/Shell/PowerShell C C++ C# F# HTML/CSS Java JavaScript Python R Ruby Scala SQL  
682,HTML/CSS Objective-C Swift  
683,HTML/CSS JavaScript PHP Python SQL  
684,C# HTML/CSS JavaScript TypeScript  
685,C# HTML/CSS Java SQL  
686,C# SQL  
687,C C++ Java Python SQL Other(s):  
688,HTML/CSS JavaScript PHP Python SQL  
689,C++ C# HTML/CSS Java JavaScript Other(s):  
690,Assembly  
691,HTML/CSS JavaScript PHP SQL  
692,HTML/CSS Java JavaScript SQL

693,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL  
694,C# Go HTML/CSS Java JavaScript PHP Python SQL VBA  
695,C# HTML/CSS JavaScript TypeScript  
696,C# SQL  
697,Bash/Shell/PowerShell HTML/CSS JavaScript Python  
698,HTML/CSS Java JavaScript Python  
699,Bash/Shell/PowerShell C C++ C# HTML/CSS JavaScript SQL  
700,HTML/CSS JavaScript Python SQL TypeScript  
701,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Rust SQL  
702,C C++ HTML/CSS JavaScript PHP Python SQL  
703,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL  
704,Dart HTML/CSS Java PHP SQL  
705,Go JavaScript  
706,Java Python Scala SQL  
707,C# HTML/CSS JavaScript Objective-C PHP Ruby SQL  
708,Bash/Shell/PowerShell C++ Go JavaScript Python SQL  
709,JavaScript TypeScript  
710,Assembly C HTML/CSS Java JavaScript Python R  
711,Java  
712,HTML/CSS JavaScript  
713,HTML/CSS JavaScript PHP SQL  
714,Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL VBA  
715,HTML/CSS JavaScript PHP SQL  
716,HTML/CSS JavaScript PHP Other(s):  
717,C# HTML/CSS Java PHP SQL  
718,C# HTML/CSS JavaScript SQL TypeScript  
719,Bash/Shell/PowerShell C++ C# HTML/CSS Java JavaScript Objective-C PHP Python SQL Swift  
720,C# HTML/CSS JavaScript PHP SQL TypeScript  
721,Bash/Shell/PowerShell Go SQL TypeScript  
723,JavaScript Python R Other(s):  
724,Bash/Shell/PowerShell C C++ C# HTML/CSS Java JavaScript Python  
725,C# HTML/CSS JavaScript SQL  
726,HTML/CSS JavaScript PHP SQL TypeScript  
727,Bash/Shell/PowerShell C C++ HTML/CSS JavaScript Python SQL  
728,Python  
729,Bash/Shell/PowerShell Go HTML/CSS JavaScript Python SQL  
730,C HTML/CSS JavaScript PHP SQL  
731,C C++ HTML/CSS JavaScript PHP SQL  
732,Bash/Shell/PowerShell C# F# HTML/CSS JavaScript SQL TypeScript  
733,C# HTML/CSS JavaScript SQL  
734,C# SQL  
735,Bash/Shell/PowerShell Java JavaScript Kotlin Python  
736,C# HTML/CSS Java JavaScript Objective-C SQL TypeScript  
737,C C++ Python  
738,Bash/Shell/PowerShell C C++ HTML/CSS Java Python R SQL  
739,Bash/Shell/PowerShell Java JavaScript Swift  
740,Bash/Shell/PowerShell Java PHP Python  
741,C++ C# HTML/CSS Java JavaScript PHP Python SQL TypeScript  
742,Go JavaScript Python SQL TypeScript  
743,HTML/CSS JavaScript PHP Ruby  
744,C C++ C# HTML/CSS JavaScript Kotlin Swift  
745,Bash/Shell/PowerShell HTML/CSS Java JavaScript Kotlin PHP SQL TypeScript

746,Go HTML/CSS Python  
747,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript Python R Ruby SQL TypeScript  
748,Java JavaScript  
749,C# HTML/CSS JavaScript SQL  
750,Dart HTML/CSS Java Kotlin Python  
751,HTML/CSS JavaScript PHP Python SQL  
752,C#  
753,C# HTML/CSS JavaScript TypeScript  
754,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Other(s):  
755,Bash/Shell/PowerShell C++ HTML/CSS Java JavaScript Python SQL  
756,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Kotlin PHP Python SQL  
Other(s):  
757,C SQL  
758,Java Python  
759,Bash/Shell/PowerShell Java Python Scala  
760,C# JavaScript SQL TypeScript  
761,C# HTML/CSS Java JavaScript PHP Python SQL TypeScript VBA  
762,C# HTML/CSS JavaScript SQL  
763,Python Rust TypeScript  
764,C# HTML/CSS JavaScript SQL  
765,Java JavaScript Ruby  
766,HTML/CSS JavaScript PHP  
767,HTML/CSS JavaScript PHP SQL  
768,Bash/Shell/PowerShell C#

```
from matplotlib import pyplot as plt
import numpy as np
from collections import Counter
import csv

with open('data.csv') as csv_file:
    csv_reader = csv.DictReader(csv_file)

    language_counter = Counter()
    for row in csv_reader:
        language_counter.update(row['LanguagesWorkedWith'].split(';'))

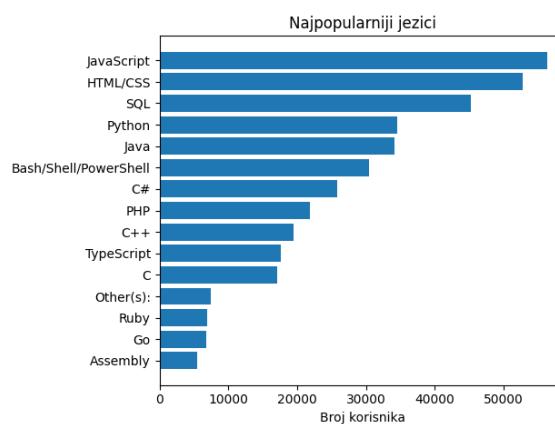
jezici = []
popularnost = []

for item in language_counter.most_common(15):
    jezici.append(item[0])
    popularnost.append(item[1])

jezici.reverse()
popularnost.reverse()

plt.barh(jezici,popularnost)
plt.title('Najpopularniji jezici')
plt.xlabel('Broj korisnika')
```

```
plt.show()
```



Slika 3.3 Horiyontalni bar grafovi. [Izvor: Autor]

## HORIZONTALNI BAR PLOT - VIDEO

*plt.barh() funkcija pravi horizontalni barplot*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PIE PLOT

*Kružni grafikon može se napraviti pozivom plt.pie() funkcije.*

### Kružni (en. pie ) grafikon

Kružni grafikon se postavlja pozivom funkcije **pie()**.

```
# pie graph
from matplotlib import lyplot as plt

...
plt.pie(...)
```

### Primer (8 minuta)

Korišćenjem podataka datih u nastavku, napraviti program koji će iscrtavati **pie** grafikon

```
slices = [59219, 55466, 47544, 36443, 35917]
labels = ['JavaScript', 'HTML/CSS', 'SQL', 'Python', 'Java']
```



Slika 3.4 Pie grafikon [Izvor: Autor]

```
from matplotlib import pyplot as plt
import numpy as np

slices = [59219, 55466, 47544, 36443, 35917]
labels = ['JavaScript', 'HTML/CSS', 'SQL', 'Python', 'Java']
explode = [0, 0, 0, 0.1, 0]

plt.pie(slices, labels=labels, explode=explode, startangle=120, autopct='%1.1f%%',
wedgeprops={'edgecolor': 'white'})

plt.style.use('fivethirtyeight')
plt.title('Pie grafikon')
plt.show()
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 4

### Uvod u pillow

#### UVOD U PILLOW PAKET

*Paket Pillow predstavlja paket za rad sa datotekama koje su slike.*



Slika 4.1 Python Pillow paket. Izvor: [<https://pillow.readthedocs.io>]

Paket **Pillow** predstavlja paket za rad sa datotekama koje su slike, i na taj način dodaje Python interpretoru mogućnost procesiranje slika (en. **image processing**).

Ovaj paket dodaje podršku za veliki broj formata, mogućnost konverzije formata, rotacije, promenu veličine i transformacije slika.

Takođe sadrži metodu za pokazivanje histograma, tj. statistike slika, koje se može iskoristiti za podešavanje konstrasta ili za globalnu statističku analizu slika

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

#### INSTALACIJA PILLOW PAKETA

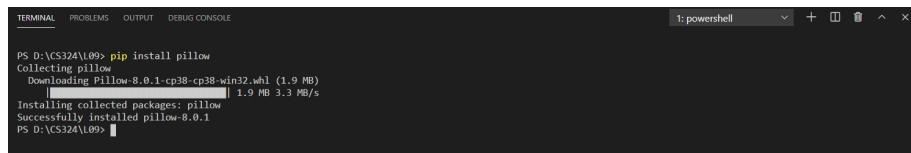
*Paket Pillow nije deo standardne Python biblioteke, te je neophodno dodatno preuzeti paket.*

Paket Pillow nije deo standardne Python biblioteke, te je neophodno dodatno preuzeti paket.

Kao i prethodne pakete, paket Pillow se može instalirati korišćenjem konzole preko pip komande:

```
pip install Pillow
```

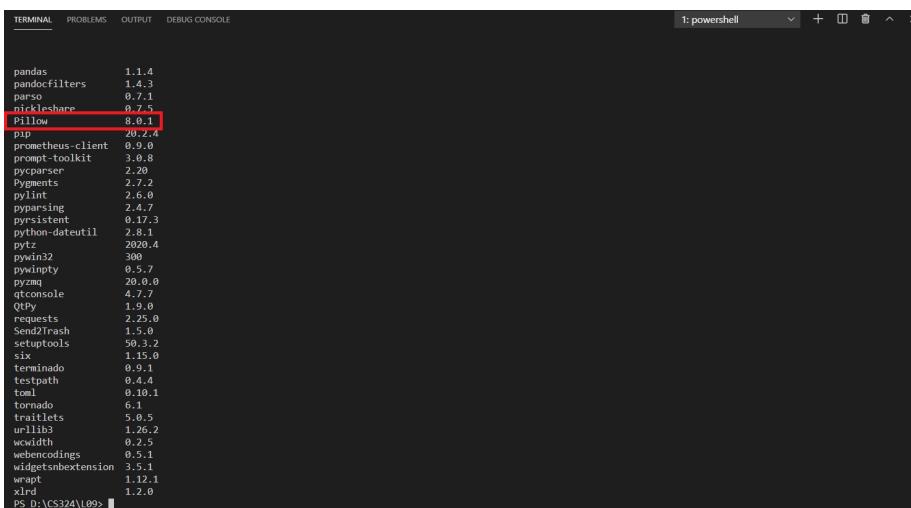
Nakon preuzimanja potrebnih datoteka, Pillow se može koristiti prilikom razvijanja Python aplikacija.



Slika 4.2 Instalacija Pillow paketa. [Izvor: Autor]

Provera (sa verzijom koja je instalirana) se može izvršiti pokretanjem pip **list** komande, ili pretragom preko komande **show**.

```
pip list  
#ili  
pip show Pillow
```



Slika 4.3 Provera instalacije Pillow paketa. [Izvor: Autor]

## UVOZ PILLOW PAKETA U RADNI DIREKTORIJUM

*Sledi video o instalaciji Pillow paketa*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

Nakon instalacije Pillow paketa, treba uvesti paket u radni direktorijum. Najviše se koristi klasa Image, te je dovoljno samo nju uvesti:

```
from PIL import Image
```

Treba obratiti pažnju da iako je paket **Pillow**, prilikom uvoza navodi se paket **PIL**.

## ✓ Poglavlje 5

### Rad sa slikama

#### KLASA IMAGE

*Osnovna klasa u Pillow paketu jeste Image klasa.*

Može se reći da je "srž" Pillow paketa upravo klasa Image. Pomoću nje možemo otvoriti, modifikovati, procesirati i sačuvati slike različitih formata.

Tokom rada sa slikama, koristiće se sledećih pet slika.



Slika 5.1 Pillow slika01. [Izvor: <http://sipi.usc.edu/database/database.php?volume=misc&image=13#top>]

Slika 5.2 Pillow slika02. [Izvor: <http://sipi.usc.edu/database/database.php?volume=misc&image=13#top>]

Slika 5.3 Pillow slika03. [Izvor: <http://sipi.usc.edu/database/database.php?volume=misc&image=13#top>]

Slika 5.4 Pillow slika04. [Izvor: <http://sipi.usc.edu/database/database.php?volume=misc&image=13#top>]

Slika 5.5 Pillow slika05. [Izvor: <http://sipi.usc.edu/database/database.php?volume=misc&image=13#top>]

# OTVARANJE I PRIKAZ SLIKE

*Ukoliko je uspešno otvorena slika, onda će se ovaj objekat otvoriti preko pregledača slika u operativnom sistemu.*

## Otvaranje slike

Otvaranje se vrši pozivom metode `.open()`.

```
from PIL import Image  
  
image01 = Image.open("slika01.png")
```

## Prikaz slike

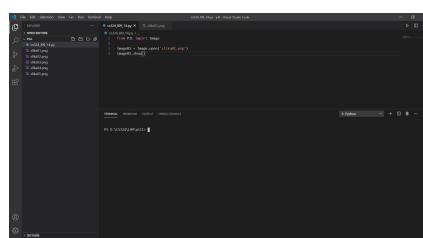
Prikaz slike vrši se metodom `.show()`

Ukoliko je uspešno otvorena slika, onda će se ovaj objekat otvoriti preko pregledača slika u operativnom sistemu.

## Čuvanje slike

Sliku je moguće vrlo jednostavno sačuvati, i u nekom drugom formatu, pozivom metode `.save()`.

```
from PIL import Image  
  
#otvaranje slike  
image01 = Image.open("slika01.png")  
  
#prikaz slike  
image01.show()  
  
#cuvanje slike  
image01.save('novo_ime.nova_ekstenzija')
```



Slika 5.6 Otvaranje i pregled slike pomoću Pillow paketa. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## KONVERTOVANJE SLIKA

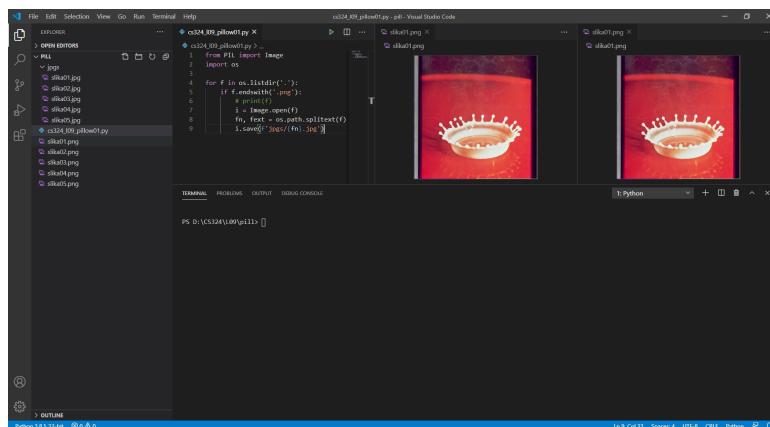
*Moguće je raditi sa više slika istovremeno ubacivanjem image objekata u petlju i vršiti modifikaciju unutar petlje.*

### Primer (6 minuta)

Konvertovati slike iz `.png` formata u `.jpg` format korišćenjem Pillow paketa.

```
from PIL import Image
import os

for f in os.listdir('.'):
    if f.endswith('.png'):
        # print(f)
        i = Image.open(f)
        fn, fext = os.path.splitext(f)
        i.save(f'jpgs/{fn}.jpg')
```



Slika 5.7 Konvertovanje slika iz PNG u JPG. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## MODIFIKACIJA SLIKA

*Modifikacije kao što su promena veličine, rotacija, i osnovni filteri mogući su u ugrađenim metodatama unutar Image i ImageFilter klase*

### Primer (6 minuta)

Promeniti veličinu slika po pikselu u 300x300 piksela, i smestiti u direktorijum sa imanom **300**.

```
from PIL import Image
import os

size_300 = (300, 300)

for f in os.listdir('.'):
    if f.endswith('.png'):
        i = Image.open(f)
        fn, fext = os.path.splitext(f)
        i.thumbnail(size_300)
        i.save(f'300/{fn}_300{fext}')
```

U prethodnom kodu nepoznata je samo metoda `.thumbnail()` kojoj se prosleđuje tuple sa veličinom u pikselima. Zbog toga je prvo bitno napravljena promenljiva **size\_300** u kojoj je stavljana veličina.

### Primer (10 minuta)

- Otvoriti sliku i rotirati je za 90 stepeni, i sačuvati je kao **slika\_mod**.
- Otvoriti sliku i prebaciti je u crno-beo format i sačuvati je kao **slika\_BW**
- Otvoriti sliku i ubaciti normalni blur sa vrednošću 5, i sačuvati je kao **slika\_blur**

```
from PIL import Image, ImageFilter

image02 = Image.open('slika02.png')
image02.rotate(90).save('slika02_mod.png')

image03 = Image.open('slika03.png')
image03.convert(mode='L').save('slika03_BW.png')

image04 = Image.open('slika04.png')
image04.filter(ImageFilter.GaussianBlur(5)).save('slika04_blur.png')
```

## ▼ Poglavlje 6

### Pokazna vežba #9

#### ZADATAK #1

*Zadatak #1 odnosi se na iscrtavanje trigonometrijske funkcije*

##### **Zadatak #1 (15 minuta)**

Napisati program koji računa funkciju:

$$y(x) = \exp(-x) \times \sin(4x), \quad x \in [0, 4]$$

Zatim, podesiti grafikon da bude puna linija crvene boje. Sačuvati grafikon kao "sin\_talas.png"

Interval vrednosti **x** treba da bude u 500 tačaka.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

def y(x):
    return np.exp(-x)*np.sin(4*x)

x = np.linspace(0, 4, 501)
plt.plot(x, y(x), 'r-', label="Sin talas")

plt.xlabel('$x$')
plt.ylabel('$y(x)$')
plt.xlim(0,4)
plt.ylim(-1,1)

plt.grid()
plt.title('Opadajući sin talas')
plt.savefig('sin_talas.png')
plt.legend()
plt.show()
```

#### ZADATAK #2, ZADATAK #3

*Zadatak #2 odnosi se na promenu RGB podešavanja slike, zadatak #3 se odnosi na doavanje pečata.*

### Zadatak #2 (10 minuta)

Napisati program koji će korišćenjem paketa Pillow promeniti vrednosti RGB svojstva slike u BGR. Sačuvati sliku kao "**slika\_invertovano.png**"

```
from PIL import Image

image = Image.open("slika01.png")
r, g, b = image.split()

image = Image.merge("RGB", (b, g, r))
image.save("slika01_invertovano.png")
```

### Zadatak #3 (20 minuta)

Napisati program koji će korišćenjem paketa Pillow dodati pečat (en. **watermark**) na sliku. Tekst pečata je "CS324"

```
from PIL import Image, ImageDraw, ImageFont

im = Image.open('slika01.png')
width, height = im.size

draw = ImageDraw.Draw(im)
text = "CS324"

font = ImageFont.truetype('arial.ttf', 36)
textwidth, textheight = draw.textsize(text, font)

margin = 10
x = width - textwidth - margin
y = height - textheight - margin

draw.text((x, y), text, font=font)
im.show()

im.save('slika01_wmark.png')
```

## ✓ Poglavlje 7

### Individualna vežba #9

#### ZADATAK #1, ZADATAK #2

*Zadatak #1 odnosi se na iscrtavanje trigonometrijske funkcije*

##### **Zadatak #1 (20 minuta)**

Napisati program koji računa funkciju:

$$g(x) = \exp\left(-\frac{1}{2}x\right) \times \cos(-2x), \quad x \in [0, 4]$$

Interval vrednosti promenljive **x** treba da bude 500 tačaka.

Zatim, podesiti grafikon da bude puna linija crvene boje.

Interval vrednosti promenljive **x** treba da bude 500 tačaka.

##### **Zadatak #2 (24 minuta)**

Napisati program koji računa funkciju:

$$h(x) = \exp\left(-\frac{3}{2}x\right) \times \sin(4x), \quad x \in [0, 4]$$

Interval vrednosti promenljive **x** treba da bude 500 tačaka.

Zatim, podesiti grafikon da bude isprekidana linija crne boje.

Nacrtati grafikon zajedno sa grafikonom **g(x)**.

Sačuvati vrednosti **x**, **g(x)** i **h(x)** u liste.

Korišćenjem pandas paketa izvesti sve vrednosti **x**, **g(x)** i **h(x)** u datoteku "**funkcije.csv**", tako da imaju i naslove kolona.

#### ZADATAK #3

*Zadatak 3 odnosi se na OOP i crtavanje grafikona.*

##### **Zadatak #3 (45 minuta)**

Napraviti klasu **warrior** koja sadrži atribute **name**, **health**, **armorClass**, **attackBonus**, i **damage**.

Klasa ima sledeće metode:

**rollForAttack**: Vraća zbir **attackBonus** i nasumičnog celog broja između 1 i 20 (simulira bacanje 20-strane kockice)

**checkHit**: proverava da li je svoj AC manji od vrednosti napada.

**takeDamage**: smanji svoj health za ulazni parametar damage.

Napraviti dva objekta klase warrior sličnih statistika (**health**, **armorClass**, **attackBonus** i damage treba da se razlikuju za do 20%)

**Primer**: warr1.health = 20, warr2.health = 24.

Pozivom metoda **rollForAttack** pratiti health oba objekta i beležiti u posebne liste. Metoda se poziva dok jedan objekat ne padne na **health=0**.

Nacrtati grafikon koji prati health oba objekta od prvog poziva do prestanka.

## ✓ Poglavlje 8

### Domaći zadatak #9

#### ZADATAK #9

*Domaći zadatak radi se okvirno 2h*

##### **Domaći zadatak #9**

Napisati program koji će za svaki položeni ispit računati prosečnu ocenu:

Sa tastature se unosi ocena (ne sme biti manja od 6) sve dok se ne pritisne ENTER. Prilikom svakog unosa računa se prosečna ocena za (do tada) unete ocene.

Prvi grafikon štampa prosečna ocenu: iscrtava promenu prolazne ocene sa brojem položenih ispita.

Drugi grafikon štampa broj pojedinačnih ocena u pie grafikonu.

```
#Primer kako bi izlaz proseccne ocene izgledao:  
Uneti Ocenu: 9  
Prosek: 9  
Uneti ocenu: 10  
Prosek: 9.5  
Uneti ocenu: 8  
Prosek: 9  
Uneti ocenu: 6  
Prosek: 8.25  
...
```

##### **Predaja domaćeg zadatka**

Nakon nedelju dana od lekcije, poeni za domaći zadatak za tradicionalne studente se umanjuju za 50%.

##### **Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije nedelju dana nakon predavanja za 100% poena. Nakon toga poeni se umanjuju za 50%.

##### **Internet studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganja ispita. Domaći zadaci se brane!

Domaći zadatak poslati dr Nemanji Zdravkoviću: nemanja.zdravkovic@metropolitan.ac.rs

Obavezno koristiti uputstvo za izradu domaćeg zadatka.

Uz .doc dokument (koji treba sadržati i screenshot svakog urađenog zadatka kao i komentare za zadatak), poslati i izvorne i dodatne datoteke.

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Rezime lekcije #9*

#### **Rezime:**

U ovoj lekciji bilo je reči o paketima matplotlib i Pillow za Python 3 programski jezik. Oba paketa se instaliraju preko pip upravljača paketa.

Paket matplotlib radi sa vizuelizacijom podataka i prikazom različitih grafikona. Matplotlib omogućuje detaljno podešavanje grafikona.

Paket Pillow jeste paket za rad sa slikama. Moguće je izvršiti manipulaciju i modifikaciju slika, kako pojedinačno, tako i nad više slika odjednom.

#### **Literatura:**

- David Beazley, Brian Jones, *Python Cookbook: Recipes for Mastering Python 3*, 3rd edition, O'Reilly Press, 2013.
- Mark Lutz, *Learning Python*, 5th Edition, O'Reilly Press, 2013.
- Andrew Bird, Lau Cher Han, et. al, *The Python Workshop*, Packt Publishing, 2019.
- Al Sweigart, *Automate the boring stuff with Python*, 2nd Edition, No Starch Press, 2020.





CS324 - SKRIPTING JEZICI

Python i naučno programiranje:  
SciPy

Lekcija 10

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 10

### *PYTHON I NAUČNO PROGRAMIRANJE: SCIPY*

- ✓ Python i naučno programiranje: SciPy
- ✓ Poglavlje 1: Uvod u SciPy
- ✓ Poglavlje 2: Napredni rad sa višedimenzionalnim nizovima
- ✓ Poglavlje 3: Napredna vizuelizacija podataka
- ✓ Poglavlje 4: Crtanje više grafikona na jednoj figuri
- ✓ Poglavlje 5: Rad sa .mat datotekama
- ✓ Poglavlje 6: Pokazne vežbe #10
- ✓ Poglavlje 7: Individualne vežbe #10
- ✓ Poglavlje 8: Domaći zadatak #10
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ❖ Uvod

### UVOD

*SciPy predstavlja skup paketa za naučno programiranje. Uz paketa kao što su NumPy, matplotlib, i pandas, SciPy predstavlja kompletno besplatno rešenje za naučno programiranje.*

U ovoj lekciji obradiće se [SciPy](#), paket za naučno programiranje, koji je nastavak [NumPy](#), [pandas](#), i [matplotlib](#) paketa.

SciPy kao celina predstavlja jedan ekosistem za matematičko, naučno i inženjersko programiranje u Python jeziku. Suštinski, [SciPy](#) predstavlja jednu biblioteku u kojoj su uključeni i prethodno obrađeni paketi ([NumPy](#) [pandas](#), i [matplotlib](#)), ali to nisu jedini paketi koji su dostupni.

Python jezik ima veliku primenu ne samo u komercijalnom programiranju, već i u naučnom i inženjerskom programiranju, a veliki doprinos popularnosti Python jezika je upravo donela paleta [SciPy](#).

U nastavku lekcije biće šire rečeno koje sve module sadrži [SciPy](#). Neka opšta podela modula bi bila za optimizaciju podataka, za linearu algebru, za obradu signala, slike i zvuka, kao i za rešavanje diferencijalnih jednačina.

Lekcija se bazira na znanju iz [NumPy](#) i [matplotlib](#) paketa, i biće obrađeni dodatni načini za rad sa višedimenzionalnim nizovima, kao i za različito predstavljanje podataka.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 1

### Uvod u SciPy

#### SCIPY KAO EKOSISTEM

*SciPy ne sadrži samo sam `scipy` paket, već se tretira kao ekosistem za naučno programiranje u Python jeziku.*



Slika 1.1 SciPy - skup alata za naučno programiranje u Python jeziku. [Izvor: <https://www.scipy.org/>]

Scipy predstavlja skup alata za naučno programiranje u Python jeziku.

Često se u literaturi Scipy navodi kao **ekosistem**, tj. skup open-source softvera za naučno programiranje, koje uključuje sledeće osnovne komponente:

- **Python** programski jezik,
- **NumPy** paket za numeričko programiranje,
- **matplotlib** paket za vizuelizaciju podataka,
- **SciPy** paket, koji uključuje kolekciju algoritama za specifičnu namenu (obrada signala, optimizacija, statistika i mnoge druge).

Osim ovih osnovnih komponenti, SciPy ekosistem uključuje opšte i specijalizovane alate za obradu podataka i izračunavanje:

- **pandas** paket za rad sa strukturama podataka,
- **Sympy** paket za simboličku matematiku i računarsku algebru,
- **NetworkX** skup alata za mrežnu analizu,
- **scikit-image** skup algoritma za obradu slika,
- **scikit-learn** skup algoritama za mašinsko učenje
- **Jupyter** sveske, Python interpreter kao veb aplikacija,
- **IPython** interaktivni interfejs,
- **Cython** ekstenzija za rad sa kodom napisan na C jeziku

SciPy predstavlja skup matematičkih algoritama i funkcija, koja se nadograđuje na NumPy paket za Python jezik. Dodaje značajno poboljšanje za interaktivno Python programiranje tako što omogućava korisniku da koristi komande visokog nivoa, kao i klase za manipulaciju i vizuelizaciju podataka. Sa SciPy paketom, Python postaje razvojno okruženje za obradu podataka koje može stati rame uz rame sa programskim jezicima kao što su MATLAB, IDL, Octave, R-Lab i i SciLab.

Dodatno poboljšanje koje SciPy ekosistem pruža jeste moguće razvijati sofisticirane i specijalizovane aplikacije, budući da je ekosistem baziran na Python programskom jeziku koji je opšteg tipa.

## INSTALACIJA POTREBNIH PAKETA ZA SCIPY EKOSISTEM

*Paket `scipy` nije deo standardne Python biblioteke, te je neophodno dodatno preuzeti paket korišćenjem konzole i `pip` komandi.*

Paket SciPy nije deo standardne Python biblioteke, te je neophodno dodatno preuzeti paket.

Kao i prethodne pakete, paket SciPy se može instalirati korišćenjem konzole preko pip komande:

```
pip install scipy
```

Nakon preuzimanja potrebnih datoteka, SciPy se može koristiti prilikom razvijanja Python aplikacija.



```
PS D:\CS324\110> pip install scipy
Collecting scipy
  Downloading scipy-1.5.4-cp38-cp38-win32.whl (28.4 kB)
Requirement already satisfied: numpy>=1.14.5 in c:\users\nerdz\appdata\local\programs\python\python38-32\lib\site-packages (from scipy) (1.19.4)
Installing collected packages: scipy
Successfully installed scipy-1.5.4
PS D:\CS324\110>
```

Slika 1.2 Instalacija `scipy` paketa. [Izvor: Autor]

Provera (sa verzijom koja je instalirana) se može izvršiti pokretanjem `pip list` komande.

```
pip list
```

Osim samog paketa SciPy, potrebno je instalirati (barem) pakete NumPy, matplotlib, i pandas. Dodatni paketi se mogu instalirati jednom komandom:

```
pip install numpy matplotlib pandas sympy
```

Slika 1.3 Provera instalacije `scipy` paketa. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## TESTIRANJE INSTALACIJE

*Posle instalacije potrebnih paketa, poželjno ih je testirati pisanjem jednostavnog programa.*

Najbolji način da se testira da li su svi paketi korektno instalirani jeste pisanjem sledećeg programa.

```
import numpy as np
print("I like ", np.pi)

from scipy import misc
import matplotlib.pyplot as plt

face = misc.face()
plt.imshow(face)
plt.show()
```

Kao izlaz, na konzoli treba da se ispiše sledeće:

```
>>> I like  3.141592653589793
```

Osim konzolnog izlaza, iz **SciPy** paketa uvozi se **misc** modul, a iz **matplotlib** paketa uvozi se **pyplot** modul za prikaz grafikona.

Prilikom korektno instaliranih paketa, osim što program neće javiti grešku, program treba da se iscrta grafikon na slici 4.



Slika 1.4 Testiranje SciPy i matplotlib paketa kroz crtanje grafikona. [Izvor: Autor]

# INSTALACIJA SCIPY PAKETA I TESTIRANJE KROZ TEST PROGRAM

*Sledi video o instalaciji Scipy paketa i provere osnovnih funkcionalnosti*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 2

# Napredni rad sa višedimenzionalnim nizovima

## INDEKSIRANJE VIŠEDIMENZIONALNIH NIZOVA

*Za razliku od jednodimenzionalnih nizova, višedimenzionalni nizovi imaju indeks po svakoj osi.*

U lekciji za numeričko programiranje u Python jeziku i radu sa NumPy paketom, obrađene su osnovne funkcije kreiranja više-dimenzionalih homogenih nizova.

U nastavku biće reči o naprednijim funkcijama unutar NumPy paketa za rad sa nizovima.

Kao i ranije, pri radu sa NumPy paketom potrebno je uvesti NumPy u radnu datoteku:

```
# uvoz NumPy paketa
import numpy as np
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### Indeksiranje višedimenzionalnih nizova

Za razliku od jednodimenzionalnih nizova, višedimenzionalni nizovi imaju indeks po svakoj osi (en. **axis**), koji se razdvaja zapetom. Moguće je indeksirati u posebnim zagradama. Kao i kod listi, indeksiranje kreće od nule.

### **Primer: Indeksiranje matrice**

Napraviti **Numpy** matricu 5x4 (pet redova i četiri kolone). Indeksirati:

- četvrti element u trećem redu.
- svaki red u drugoj koloni (sve elemente druge kolone),
- svaku kolonu u drugom i trećem redu (ceo drugi i treći red).

```
import numpy as np

b = np.array([[ 0,  1,  2,  3],
              [10, 11, 12, 13],
              [20, 21, 22, 23],
              [30, 31, 32, 33],
              [40, 41, 42, 43]])

# jedan element
```

```
print(b[2,3])
print(b[2][3])
# kolone
print(b[0:5, 1])
print(b[:,1])
# redovi
print(b[1:3,:])
```

## NIZOVI SPECIFIČNOG REDOSLEDA - ARANGE()

*Funkcijom arange() vraća se niz sa elementima u datom intervalu sa određenim korakom.*

### Funkcija arange()

Funkcijom **arange()** vraća se niz sa elementima u datom intervalu sa određenim korakom.

```
numpy.arange(start, stop, step, dtype=None)
```

Parametar **start** je opcioni (podrazumevani je 0) i označava početak intervala.

Parametar **stop** označava kraj intervala. Niz ne uključuje i samu vrednost koja se ubaci kao interval.

Parametar **step** jeste korak i označava razliku vrednosti između elemenata niza. Ovaj parametar je opcioni i podrazumevana vrednost je 1.

```
np.arange(3)
>>>array([0, 1, 2])
np.arange(3.0)
>>>array([ 0.,  1.,  2.])
np.arange(3,7)
>>>array([3, 4, 5, 6])
np.arange(3,7,2)
>>>array([3, 5])
```

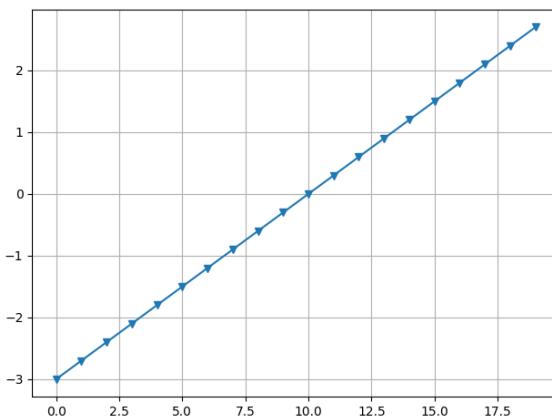
### Primer (5 minuta)

Napisati program koji će vratiti numpy niz od -3 do 3 sa korakom 0.3.

Nacrtati grafikon koji prati vrednosti niza kao linijski grafikon.

```
import numpy as np
from matplotlib import pyplot as plt

a = np.arange(-3, 3, 0.3)
plt.plot(a, marker='v', label='arange niz')
plt.tight_layout()
plt.grid()
plt.show()
```



Slika 2.1 Niz preko aranje funkcije. [Izvor: Autor]

## NIZOVI SPECIFIČNOG REDOSLEDA - Linspace()

*Funkcijom `linspace()` vraća se niz sa elementima u datom intervalu i određenim brojem elemenata.*

### Funkcija `linspace()`

Funkcijom `linspace()` vraća se niz sa elementima u datom intervalu sa određenim brojem elemenata.

```
numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None)
```

Parametar **start** označava početak intervala.

Parametar **stop** označava kraj intervala. Niz uključuje i samu vrednost koja se ubaci kao interval. Ukoliko je **endpoint** parametar **False**, onda se ne uključuje u sam niz.

Parametar **num** jeste broj vrednosti koji se generiše između start i stop (uključujući i start i stop). Podrazumevana vrednost je 50.

Parametar **retstep** (podrazumevano je `False`) vratiće i vrednost koraka. U ovom slučaju niz i korak se vraćaju kao **tuple**.

```
np.linspace(2.0, 3.0, num=5)
>>>array([2.0, 2.25, 2.5, 2.75, 3.0])
np.linspace(2.0, 3.0, num=5, endpoint=False)
>>>array([2.0, 2.2, 2.4, 2.6, 2.8])
np.linspace(2.0, 3.0, num=5, retstep=True)
>>>(array([2.0, 2.25, 2.5, 2.75, 3.0]), 0.25)
```

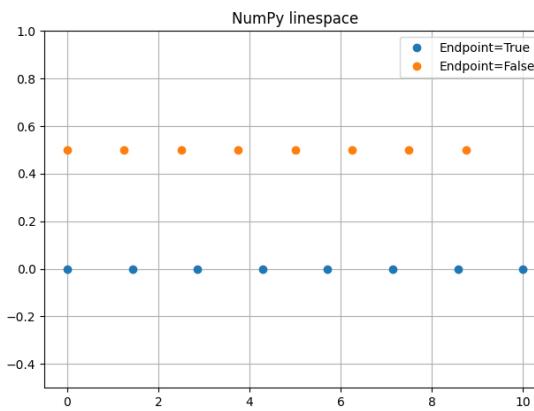
### Primer (5 minuta)

Napisati program koji će vratiti numpy nizove od 0 do 10 sa 8 elementa. Kod prvog niza **endpoint=False**, a kod drugog je `True`.

Nacrtati grafikon koji prati vrednosti niza (prvi niz se crta na  $y=0$ , drugi na  $y=0.5$ ) kao linijski grafikon samo sa markerima.

```
import numpy as np
import matplotlib.pyplot as plt

x1 = np.linspace(0, 10, 8, endpoint=True)
x2 = np.linspace(0, 10, 8, endpoint=False)
y = np.zeros(x1.shape)
plt.plot(x1, y, 'o')
plt.plot(x2, y + 0.5, 'o')
plt.ylim([-0.5, 1])
plt.grid()
plt.tight_layout()
plt.show()
```



Slika 2.2 Niz preko linspace funkcije. [Izvor: Autor]

## NIZOVI SPECIFIČNOG REDOSLEDA - GEOMSPACE() I LOGSPACE()

*Funkcije geomspace() i logspace() vraćaju nizove sa razmakom elemenata u geometrijskoj i logaritamskoj progresiji.*

### Funkcija geomspace()

Funkcijom **geomspace()** vraća se niz sa elementima u datom intervalu sa određenim brojem elemenata i veličine razmaka između elemenata u geometrijskoj progresiji.

```
numpy.geomspace(start, stop, num=50, endpoint=True, dtype=None)
```

Parametar **start** označava početak intervala.

Parametar **stop** označava kraj intervala. Niz uključuje i samu vrednost koja se ubaci kao interval. Ukoliko je **endpoint** parametar **False**, onda se ne uključuje u sam niz.

Parametar **num** jeste broj vrednosti koji se generiše između start i stop (uključujući i start i stop). Podrazumevana vrednost je 50.

### Funkcija `logspace()`

Funkcijom `logspace()` vraća se niz sa elementima u datom intervalu sa određenim brojem elemenata i veličine razmaka između elemenata u logaritamskoj progresiji.

```
numpy.logspace(start, stop, num=50, endpoint=True, base=10.0, dtype=None)
```

Parametar **start** označava prvi stepen.

Parametar **stop** poslednji stepen. Niz uključuje i samu vrednost koja se ubaci kao interval. Ukoliko je **endpoint** parametar **False**, onda se ne uključuje u sam niz.

Parametar **num** jeste broj vrednosti koji se generiše između start i stop (uključujući i start i stop). Podrazumevana vrednost je 50.

Parametar **base** (podrazumevano je 10) označava koja je osnova za logaritam.

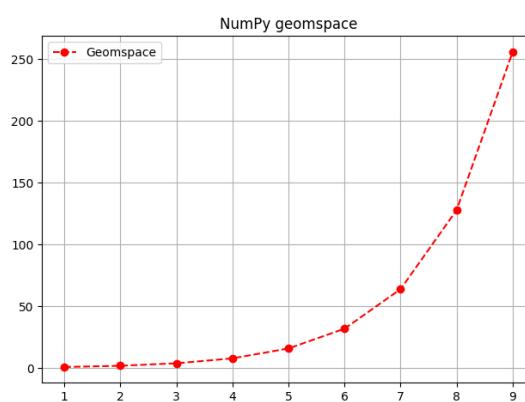
## PRIMERI ZA GEOMSPACE() I LOGSPACE()

*Slede primjeri za korišćenje `geomspace` i `logspace` funkcija.*

### Primer (5 minuta)

Napisati program za prikazivanje vrednosti stepena dvojke od 1 do 256, korišćenjem `geomspace()` NumPy funkcije.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(1, 10)
y1 = np.geomspace(1, 256, 9)
plt.plot(x, y1, 'or--', label='Geomspace')
plt.grid()
plt.legend()
plt.title('NumPy geomspace')
plt.tight_layout()
plt.show()
```

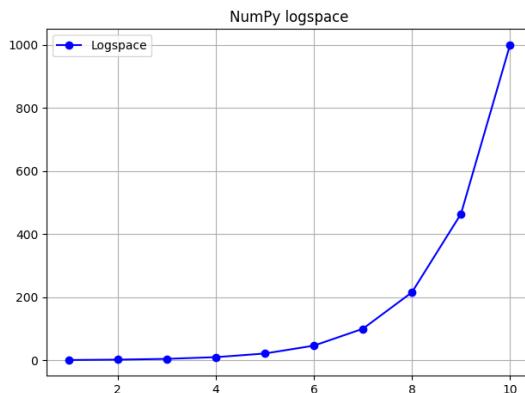


Slika 2.3 Niz preko geomspace funkcije. [Izvor: Autor]

### Primer (5 minuta)

Napisati program za prikazivanje vrednosti od  $10^0$  do  $10^3$  kroz 10 elemenata, korišćenjem **logspace()** NumPy funkcije.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(1, 11)
y1 = np.logspace(0, 3, num=10, base=10)
plt.plot(x, y1, 'ob-', label='Logspace')
plt.grid()
plt.legend()
plt.title('NumPy logspace')
plt.tight_layout()
plt.show()
```



Slika 2.4 Niz preko logspace funkcije. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PROMENA OBLIKA NIZA: FUNKCIJA RESHAPE

*Eksplisitna promena oblika niza bez promene sadržaja niza moguća je funkcijom **reshape()**.*

**Promena oblika niza: **reshape()** funkcija**

Eksplisitna promena oblika niza bez promene sadržaja niza moguća je funkcijom **reshape()**

```
import numpy as np
np.reshape(a, newshape, order)
```

Parametar **a** odnosi se na niz koji treba da promeni obliku

Parametar **newshape** je ceo broj ili tuple sa dimenzijama novog oblika

Parametar **order** je opcioni parametar i odnosi se na redosled čitanja elemenata niza. Order prima kao argument "C", "F" ili "A" i odnosi se na indeksiranje kao u C-u, Fortran-u, dok "A" čita elemente kao da su u elementi niza sukcesivni u memoriji kao u Fortran programskom jeziku.

### Primer (5 minuta)

Napraviti jednodimenzionalni niz od šest elemenata. Promeniti oblika niza sa tuple-om (2, 3) bez redosleda, i sa C i F redosledom.

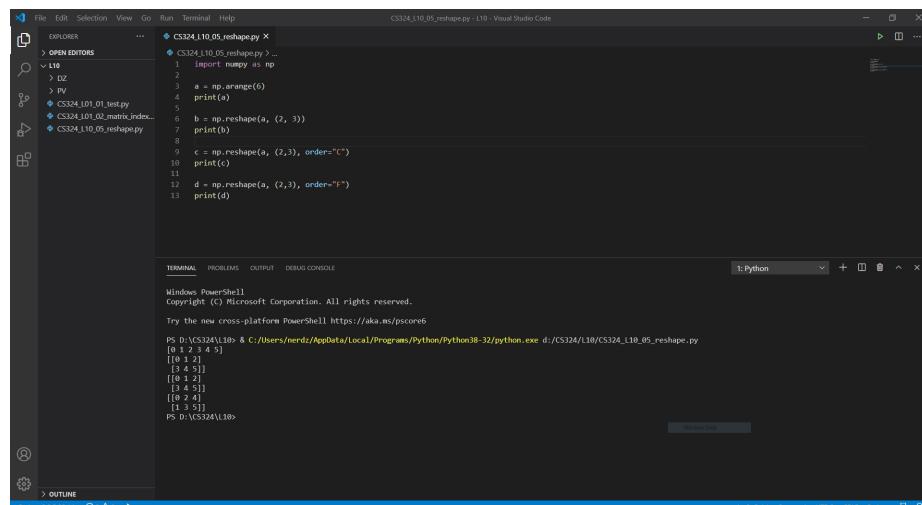
```
import numpy as np

a = np.arange(6)
print(a)

b = np.reshape(a, (2, 3))
print(b)

c = np.reshape(a, (2,3), order="C")
print(c)

d = np.reshape(a, (2,3), order="F")
print(d)
```



Slika 2.5 Različite promene oblika niza u odnosu na redosled indeksiranja. [Izvor: Autor]

## RAZDVAJANJE I SPAJANJE NIZA

*Niz se može podeliti u više manjih nizova korišćenjem funkcije `split()`.*

### Funkcija `split()`

Niz se može podeliti u više manjih nizova korišćenjem funkcije `split()`.

```
numpy.split(ary, indices_or_sections, axis=0)
```

Prvi parametar, **ary**, odnosi se na niz koji treba podeliti.

Drugi parametar, **indices\_or\_sections**, odnosi se na broj nizova na koliko treba prvobitni niz podeliti.

Treći parametar je **osa** (odnosno dimenzija) i odnosi se na dimenziju po kojoj je podela urađena. Podrazumevana vrednost je 0.

Funkcija vratiće listu NumPy nizova.

```
import numpy as np
a = np.arange(1,11,1)

print(a)
b = np.split(a,2)
print(b)
```

```
# izlaz konzole:
>>> [ 1  2  3  4  5  6  7  8  9 10]
>>> [array([1, 2, 3, 4, 5]), array([ 6,  7,  8,  9, 10])]
```

### Funkcija concatenate()

Nizovi se mogu spojiti u jedan veći niz korišćenjem funkcije **concatenate()**.

```
numpy.concatenate((a1, a2, ...), axis=0)
```

Prvi parametar je **tuple nizova** i odnosi se na nizove koji treba spojiti.

Drugi parametar je **osa** (odnosno dimenzija) i odnosi se na dimenziju po kojoj je spajanje urađeno. Podrazumevana vrednost je 0.

Funkcija vratiće jedan NumPy niz.

```
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6]])
np.concatenate((a, b), axis=0)
np.concatenate((a, b.T), axis=1)
np.concatenate((a, b), axis=None)
```

```
# izlaz konzole:
>>> array([[1, 2],
   [3, 4],
   [5, 6]])

>>> array([[1, 2, 5],
   [3, 4, 6]])

>>> array([1, 2, 3, 4, 5, 6])
```

## ▼ Poglavlje 3

# Napredna vizuelizacija podataka

## POPUNJAVANJE POVRŠINE NA LINE PLOT GRAFIKONU

*Funkcija `fill_between()` puni površinu unutar linijskog grafikona po nekom uslovu.*

Dodatno obeležavanje unutar linijskih grafikona može se postići crtanjem površina iznad ili ispod linija kada se ispuni neki uslov pomoću funkcije `fill_between()`.

Definicija funkcije je sledeća:

```
fill_between(x, y1, y2=0, where=None, alpha=0, interpolate=False)
```

Vrednosti **x** i **y1** jesu podaci **X** i **Y** ose, dok je **y2** (podrazumevano jednaka nuli) vrednost granice popunjavanja površine. Ukoliko postoji uslov, onda se prosleđuje u **where** parametru, dok je **alpha** providnost površine. Opcioni parametar **interpolate** popuniće deo linija koje nedostaju.

### Primer (6 minuta)

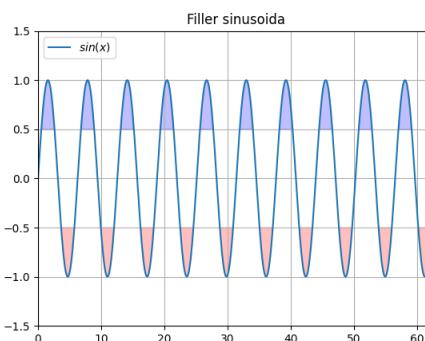
Napraviti numpy niz **x** koji kreće od nula do  $20 * \pi$  sa korakom 0.1. Izračunati  $\sin(x)$  i iscrtati na linijskom grafikonu. zatim, iscrtati površinu plave boje i providnosti 0.25 kada je ispunjen uslov da je  $y > 0.5$ , i površinu crvene boje iste providnosti, kada je ispunjen uslov da je  $y < -0.5$ .

```
import matplotlib.pyplot as plt
import numpy as np

# Sinusoidna funkcija
x = np.arange(0, 20 * np.pi, 0.1)
y = np.sin(x)

plt.plot(x,y, label='$\sin(x)$')

plt.fill_between(x,y, 0.5, where=(y > 0.5), color='blue', alpha=0.25)
plt.fill_between(x,y, -0.5, where=(y <= -0.5), color='red', alpha=0.25)
plt.grid()
plt.legend(loc='upper left')
plt.axis([0, 20 * np.pi, -1.5, 1.5])
plt.title('Filler sinusoida')
plt.show()
```



Slika 3.1 IsCRTavanje površine po datom uslovu na sinusnoj funkciji. [Izvor: Autor]

## SCATTER PLOT

*Scatter plot predstavlja grafikon po tačkama, gde osim x-y osa i sama boja tačke može nositi dodatnu informaciju*

Scatter plot jeste prikaz po tačkama.

Funkcija za ovakav prikaz jeste **scatter()**, gde su prvi parametri **X** i **Y** ose, dok su ostali parametri vezani za formatiranje tačaka.

**Primer (20 minuta):**

Napisati program koji pravi scatter grafikon odnosa trending klipova na YouTube-u koristeći podatke u nastavku (sačuvati podatke kao CSV datoteku). X osa predstavlja broj pregleda, Y osa predstavlja broj lajkova, dok boja tačke predstavlja odnos lajkova i dislajkova. Ose su logaritamske.

```
view_count,likes,ratio
8036001,324742,96.91
9378067,562589,98.19
2182066,273650,99.38
6525864,94698,96.25
9481284,582481,97.22
1853121,89903,97.46
2875684,183163,94.52
483827,4864,91.53
1677046,103227,97.52
289756,2387,92.95
2561907,237728,98.8
468390,25346,98.34
18977153,768968,98.73
365731,5997,93.29
680701,41543,97.99
5748289,225966,99.17
3575950,374937,97.69
865788,31806,98.3
5433739,389145,98.84
```

3643458,369667,97.88  
247602,1516,89.18  
300443,25429,99.49  
313500,56891,98.35  
3525217,92948,95.29  
195072,23832,98.97  
142697,20708,98.91  
456783,2625,94.53  
601565,38792,98.34  
6021472,342044,97.54  
940583,14292,97.7  
446569,7557,97.15  
767900,11091,97.14  
5895810,98088,95.87  
381910,45178,99.21  
2468645,188315,98.73  
407859,19407,98.77  
846399,29308,95.93  
872092,27298,94.85  
1279718,98471,99.06  
1068377,92634,98.89  
4691951,164807,98.93  
1091006,55346,98.53  
891230,30612,88.39  
720734,35647,98.11  
1025214,19926,94.86  
505146,3309,59.69  
265430,2124,91.99  
3651318,283911,98.64  
1290212,201881,99.3  
420393,5434,95.99  
655107,21485,96.16  
1010207,23720,95.85  
777547,9167,94.46  
686703,34001,98.54  
1625877,62101,98.35  
2107926,59334,97.3  
1564214,81581,97.96  
2277765,53425,89.82  
1558609,95695,98.23  
1689305,88050,95.43  
3382856,74078,93.32  
4835746,276098,94.3  
248754,2041,90.75  
687182,63309,97.61  
751948,24359,98.3  
737756,23093,82.35  
964229,18898,86.34  
973121,22810,97.6  
575508,16975,94.75  
1114419,35208,94.3  
722956,21843,97.6  
1560200,38185,96.52

281397,3706,91.53  
1122525,28232,97.23  
20650480,212862,91.88  
225207,1524,84.76  
598367,24260,94.51  
2117363,162960,99.12  
1233027,16400,88.81  
2566897,112005,54.67  
11907188,1234111,83.49  
1477059,36018,98.75  
292469,5656,92.71  
466862,47754,98.96  
1055798,46122,97.84  
1278142,26021,97.37  
1938747,16942,87.66  
338563,8416,96.46  
645274,17943,94.67  
730110,26868,92.31  
1521090,19761,86.6  
1719425,79646,98.33  
3028604,75484,97.22  
1236239,55409,96.0  
906642,14128,91.88  
1257902,20899,92.93  
1163635,30173,89.82  
1413936,90918,97.87  
709519,6013,95.14  
628111,41450,97.03  
2478832,143686,98.28  
2524598,32486,93.66  
821547,18708,97.31  
3016943,38294,95.76  
743575,20181,89.7  
919626,22114,95.84  
2536083,538376,99.6  
959442,13220,95.94  
2044159,41080,92.48  
1554417,67165,93.0  
2181022,180132,98.19  
1010899,13696,97.57  
2620663,72681,96.68  
5732609,189529,97.16  
1187273,73120,99.24  
1594532,85661,97.01  
8403016,294629,96.97  
5972754,133474,96.6  
6189511,267690,99.03  
1042734,23761,91.61  
9476773,417402,97.8  
8040754,789213,98.73  
2724624,88968,91.74  
1085592,27288,98.51  
3393417,219213,95.68

16396012,208578,79.21  
3226905,19814,91.77  
6276301,286642,98.15  
647094,19753,89.98  
8081040,477122,98.81  
886934,29360,98.46  
1228396,29893,98.2  
697471,6452,94.85  
1605670,78364,96.63  
2056991,121925,98.44  
397981,6185,58.36  
2760289,106828,97.14  
3655043,54069,89.65  
10662064,320959,97.89  
3105500,108620,96.6  
2238691,48825,96.77  
1153518,25832,96.44  
686228,24882,96.57  
7523411,614901,98.87  
2641916,49354,95.78  
11657853,233343,97.82  
5932061,172195,95.91  
6313988,323119,98.18  
2850316,218273,98.14  
2620142,36637,93.99  
854120,54821,98.05  
13799864,317613,96.07  
906841,35315,98.09  
689607,20658,98.58  
441729,14901,99.0  
797800,14327,95.41  
1682016,75706,98.17  
1426251,57965,98.73  
2268534,91796,97.75  
750032,39406,98.19  
4272799,26229,98.03  
2449662,80825,97.54  
5988592,512483,99.4  
3662227,75552,97.46  
725964,42700,98.98  
1647440,111190,98.85  
985104,12721,96.5  
1665692,23961,92.37  
2051794,81790,96.64  
4112883,116481,93.46  
33297045,1293427,99.07  
1517628,19931,96.25  
1675692,18803,72.76  
3626738,173591,98.44  
1169663,7766,92.99  
446959,4923,89.48  
6995153,195994,96.69  
519706,18975,98.94

```
4373224,169228,93.01
4024087,73080,97.71
731349,42205,98.52
94366013,4539630,97.66
2458132,34337,95.52
1812670,17476,94.43
2028445,158178,97.94
1335703,12622,94.14
938717,17120,97.26
2926955,42554,97.73
4018930,32919,82.1
6439402,81148,51.58
5665790,166892,96.95
899728,28115,96.49
2792057,206926,96.99
12839663,722491,97.84
5694139,146797,98.19
1069693,3970,90.66
590760,70454,99.18
319347,1208,92.5
27594927,1351963,96.4
26993425,437561,97.42
```

```
import pandas as pd
from matplotlib import pyplot as plt

data = pd.read_csv('scatter_data.csv')
view_count = data['view_count']
likes = data['likes']
ratio = data['ratio']

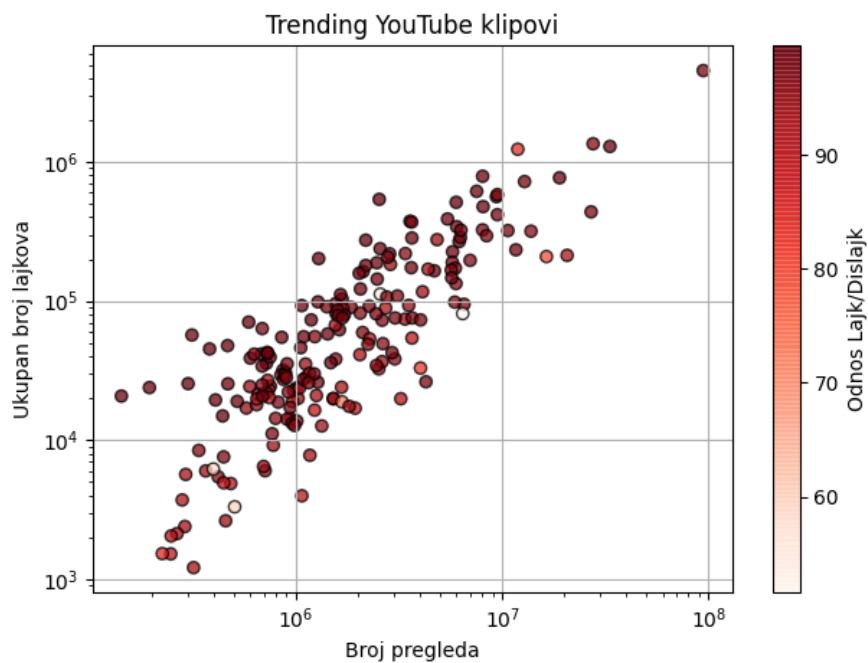
plt.scatter(view_count, likes, c=ratio, cmap='Reds', edgecolor='black',
linewidth=1, alpha=0.75)

cbar = plt.colorbar()
cbar.set_label('Odnos Lajk/Dislajk')

plt.xscale('log')
plt.yscale('log')

plt.title('Trending YouTube klipovi')
plt.xlabel('Broj pregleda')
plt.ylabel('Ukupan broj lajkova')

plt.tight_layout()
plt.grid()
plt.show()
```



Slika 3.2 Scatter plot za odnos lajk/dislajk za YouTube klipove. [Izvor: Autor]

## CRTANJE GRAFIKONA U REALNOM VREMENU

*Moguće je iscrtavati grafikon u realnom vremenu korišćenjem FuncAnimation modula iz matplotlib paketa.*

Moguće je iscrtavati grafikon u realnom vremenu korišćenjem [FuncAnimation](#) modula iz [matplotlib](#) paketa.

### Primer: 15 minuta

Napraviti program koji u CSV dokumentu u dve kolone upisuje vrednosti koji se kreću od 1000. Prvoj vrednosti se dodaje nasumični element od -6 do 8, a drugoj od -5 do 6. Program radi u beskonačnoj petlji dok se ručno ne zaustavi. Svaka iteracija kasni jednu sekundu.

U drugoj datoteci u realnom vremenu učitavati podatke i iscrtavati kao linijski plot.

```
import csv
import random
import time
import os

x_value = 0
total_1 = 1000
total_2 = 1000

fieldnames = ["x_value", "total_1", "total_2"]
```

```
with open('PLOT/data.csv', 'w') as csv_file:
    csv_writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
    csv_writer.writeheader()

while True:

    with open('PLOT/data.csv', 'a') as csv_file:
        csv_writer = csv.DictWriter(csv_file, fieldnames=fieldnames)

        info = {
            "x_value": x_value,
            "total_1": total_1,
            "total_2": total_2
        }

        csv_writer.writerow(info)
        print(x_value, total_1, total_2)

        x_value += 1
        total_1 = total_1 + random.randint(-6, 8)
        total_2 = total_2 + random.randint(-5, 6)

    time.sleep(1)
```

```
import random
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

def animate(i):
    data = pd.read_csv('data.csv')
    x = data['x_value']
    y1 = data['total_1']
    y2 = data['total_2']

    plt.cla()

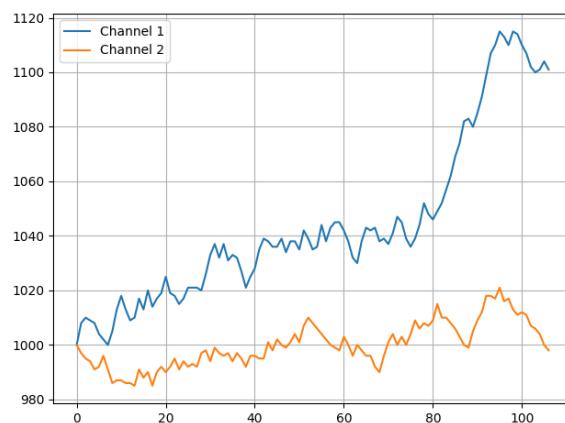
    plt.plot(x, y1, label='Channel 1')
    plt.plot(x, y2, label='Channel 2')

    plt.legend(loc='upper left')
    plt.tight_layout()
    plt.grid()

ani = FuncAnimation(plt.gcf(), animate, interval=1000)

plt.tight_layout()

plt.show()
```



Slika 3.3 Prikaz u realnom vremenu. Sačuvana figure je posle 100 iteracija. [Izvor: Autor]

## ▼ Poglavlje 4

# Crtanje više grafikona na jednoj figuri

## CRTANJE VIŠE GRAFIKONA NA JEDNOJ FIGURI - SUBPLOT

*Moguće je imati više grafikona na jednoj figuri korišćenjem subplot() funkcije.*

Moguće je imati više grafikona na jednoj figuri korišćenjem `subplot()` funkcije. Na ovaj način moguće je jednim pozivom za crtanje pokazati više različitih tipova grafikona.

Kroz primer sinusoidne funkcije opisaće se više načina postavljanja grafikona:

```
import matplotlib.pyplot as plt
import numpy as np

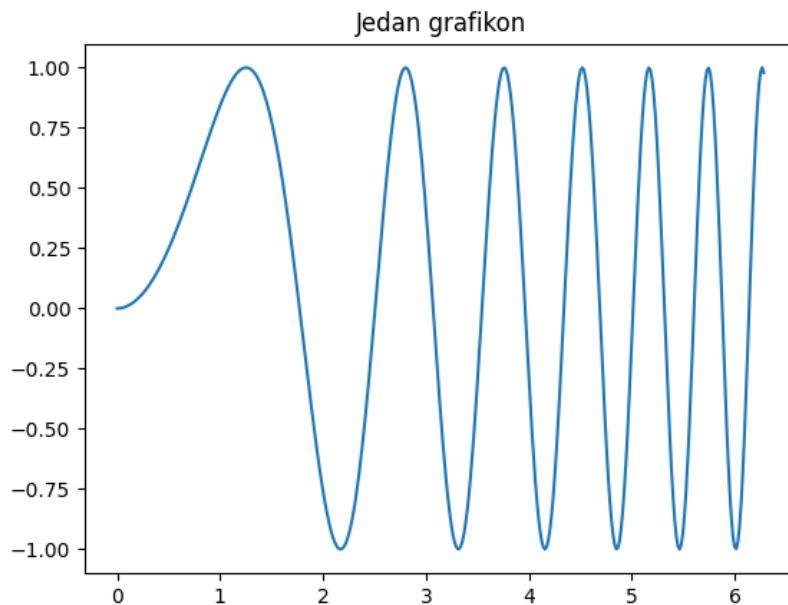
x = np.linspace(0, 2 * np.pi, 400)
y = np.sin(x ** 2)
```

### Jedan grafikon

Funkcija `subplots()` bez argumenata vratiće jednu figuru (en. `figure`) i jedan skup osa (en. `axes`)

```
# Jeden grafikon
fig, ax = plt.subplots()

ax.plot(x, y)
ax.set_title('Jeden grafikon')
plt.show()
```



Slika 4.1 Jedan grafikon. [Izvor: Autor]

## DVA VERTIKALNA GRAFIKONA

*Kada se grafikoni redaju samo po jednoj dimenziji, vraćena **axs** vrednost predstavlja jednodimenzionalni NumPy niz koji sadrži napravljene ose.*

Funkcija **subplots()** ima više opcionih argumenata. Prvi argument definiše broj redova i kolona subplot mreže (što se preslikava na broj horizontalno i vertikalno poredanih grafikona).

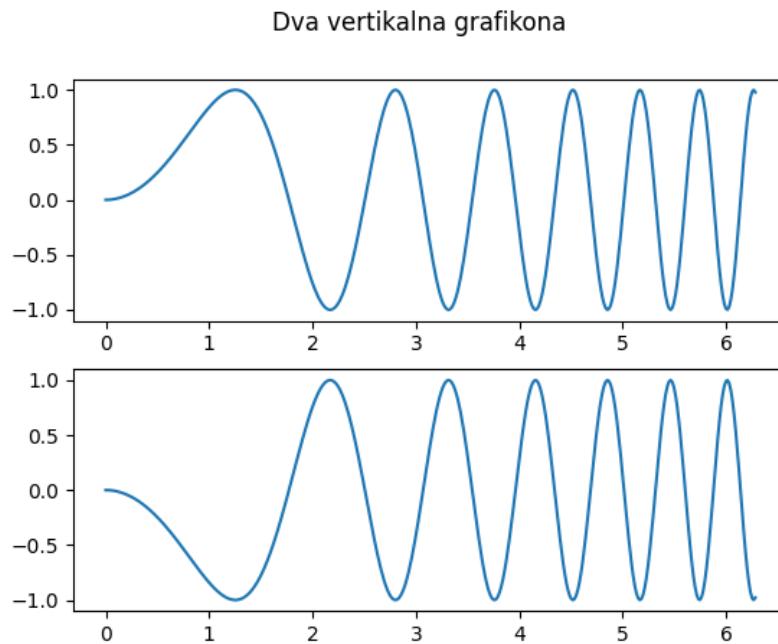
Kada se grafikoni redaju samo po jednoj dimenziji, vraćena **axs** vrednost predstavlja jednodimenzionalni **NumPy** niz koji sadrži napravljene ose.

### Dva vertikalno poređana grafikona

```
# Dva vertikalna grafikona
fig, axs = plt.subplots(2)
fig.suptitle('Dva vertikalna grafikona')
axs[0].plot(x, y)
axs[1].plot(x, -y)
plt.show()
```

Takođe je moguće **axs** niz eksplisitno raspakovati u pojedinačne promenljive:

```
# Dva vertikalna grafikona sa odvojenim osama po promenljivama.
fig, (ax1, ax2) = plt.subplots(2)
fig.suptitle('Vertically stacked subplots')
ax1.plot(x, y)
ax2.plot(x, -y)
```



Slika 4.2 Dva vertikalna grafikona. [Izvor: Autor]

## DVA HORIZONTALNA GRAFIKONA

*Za dva grafikona koja su "jedan do drugog, odnosno horizontalno raspoređena, potrebno je proslediti parametre za broj redova i kolona unutar iste figure.*

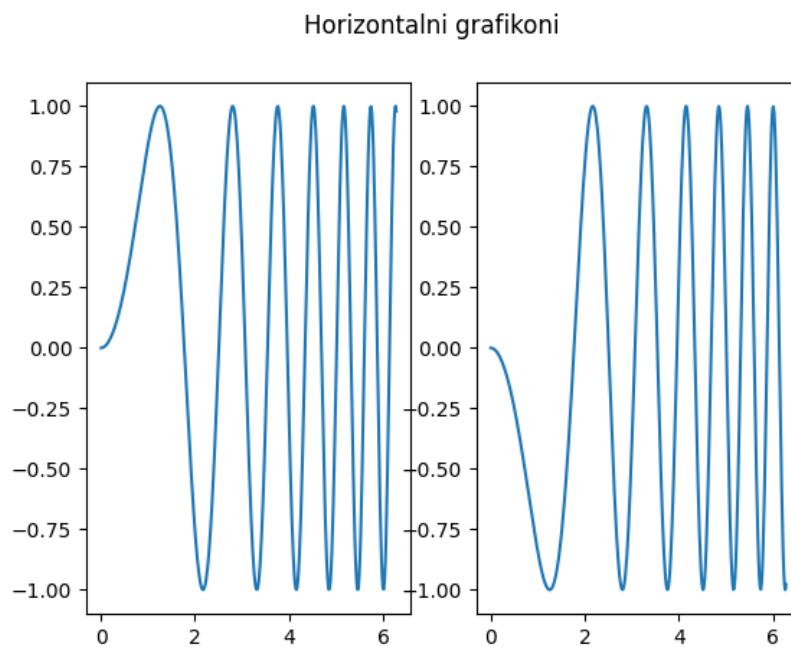
Za dva grafikona koja su "jedan do drugog, odnosno horizontalno raspoređena, potrebno je proslediti parametre (1, 2) za jedan red i dve kolone unutar iste figure.

```
# Dva horizontalna grafikona

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Horizontalni grafikoni')
ax1.plot(x, y)
ax2.plot(x, -y)
plt.show()
```

Moguće je navesti samo ose, i dobiti listu pojedinačnih grafikona.

```
# Dva horizontalna grafikona
fig, axs = plt.subplots(1, 2)
fig.suptitle('Dva horizontalna grafikona')
axs[0].plot(x, y)
axs[1].plot(x, -y)
plt.show()
```



Slika 4.3 Dva horizontalna grafikona. [Izvor: Autor]

## ČETIRI GRAFIKONA

*Kada postoji više grafikona u obe dimenzije (više horizontalnih i više vertikalnih), treba podesiti parametre za svaki subplot.*

Kada postoji više grafikona u obe dimenzije (više horizontalnih i više vertikalnih), treba podesiti parametre za svaki **subplot**.

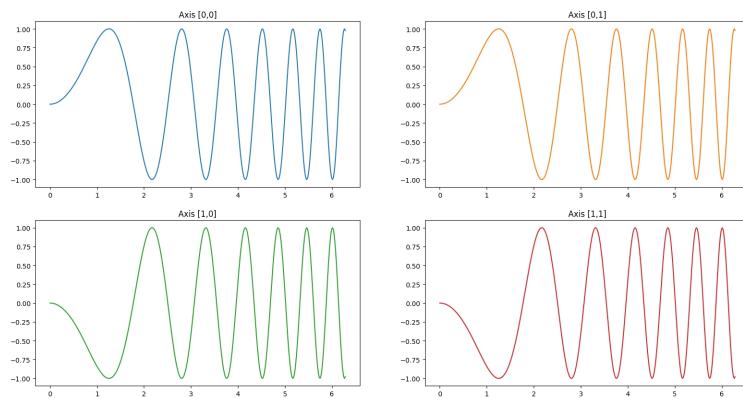
Za četiri grafikona, parametri za sublots koji se prosleđuju jesu (2, 2)

```
import matplotlib.pyplot as plt
import numpy as np

# Sinusoidna funkcija
x = np.linspace(0, 2 * np.pi, 400)
y = np.sin(x ** 2)

fig, axs = plt.subplots(2, 2)
axs[0, 0].plot(x, y)
axs[0, 0].set_title('Axis [0,0]')
axs[0, 1].plot(x, y, 'tab:orange')
axs[0, 1].set_title('Axis [0,1]')
axs[1, 0].plot(x, -y, 'tab:green')
axs[1, 0].set_title('Axis [1,0]')
axs[1, 1].plot(x, -y, 'tab:red')
axs[1, 1].set_title('Axis [1,1]')

plt.show()
```



Slika 4.4 Četiri grafikona. [Izvor: Autor]

Dodatni primeri za više grafikona na jednoj figuri mogu se naći na:

[https://matplotlib.org/3.1.0/gallery/subplots\\_axes\\_and\\_figures/subplots\\_demo.html](https://matplotlib.org/3.1.0/gallery/subplots_axes_and_figures/subplots_demo.html)

## ▼ Poglavlje 5

### Rad sa .mat datotekama

## UVOZ IZ MATLAB/OCTAVE PROGRAMSKIH JEZIKA

*Nizovi napravljeni u MATLAB ili Octave programskim jezicima mogu se uvesti `loadmat()` funkcijom.*

Pri radu sa višedimenzionalnim nizovima često se vuče paralela sa programskim jezicima MATLAB (i Octave).

MATLAB i Octave rade sa datotekama koje imaju ekstenziju .m i .mat, i one se pomoću modula **scipy.io** uvesti u Python.

#### Uvoz nizova

Komanda za uvoz samog modula je sledeća:

```
import scipy.io as sio
```

Unutar **scipy.io** modula postoje funkcije za uvoz i izvoz .mat datoteka:

```
sio.loadmat  
sio.savemat
```

#### **Primer:**

Napraviti dva niza u **MATLAB** programskom jeziku, sačuvati kao **myfile.mat**. Zatim, uvesti podatke kao NumPy niz.

```
% MATLAB datoteka  
a = 1 : 12  
b = [5, 6, 7, 8; 9, 10, 11, 12; 13, 14, 15, 16]  
save('myfile.mat', '-v7')
```

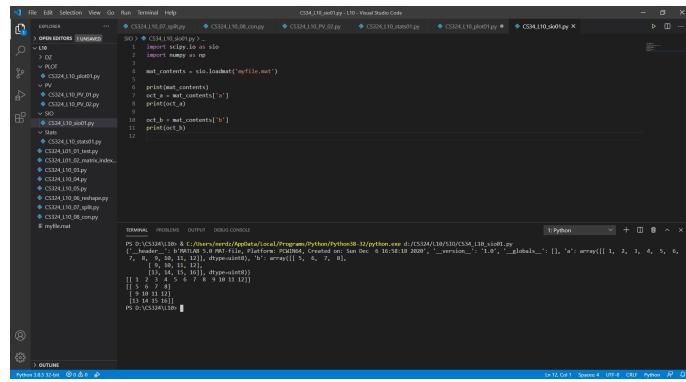
Prilikom uvoza koristi se funkcija **loadmat()** koja za parametar ima ime **.mat** datoteke. Povratna vrednost je nalik na imenik, i mora se dodatno izvući navođenjem ključa.

Uvezene vrednosti postaju NumPy nizovi.

```
import scipy.io as sio  
import numpy as np  
  
mat_contents = sio.loadmat('myfile.mat')
```

```
oct_a = mat_contents['a']
print(oct_a)

oct_b = mat_contents['b']
print(oct_b)
```



Slika 5.1 Uvoz iz .mat datoteke. [Izvor: Autor]

## IZVOZ U MATLAB/OCTAVE PROGRAMSKE JEZIKE

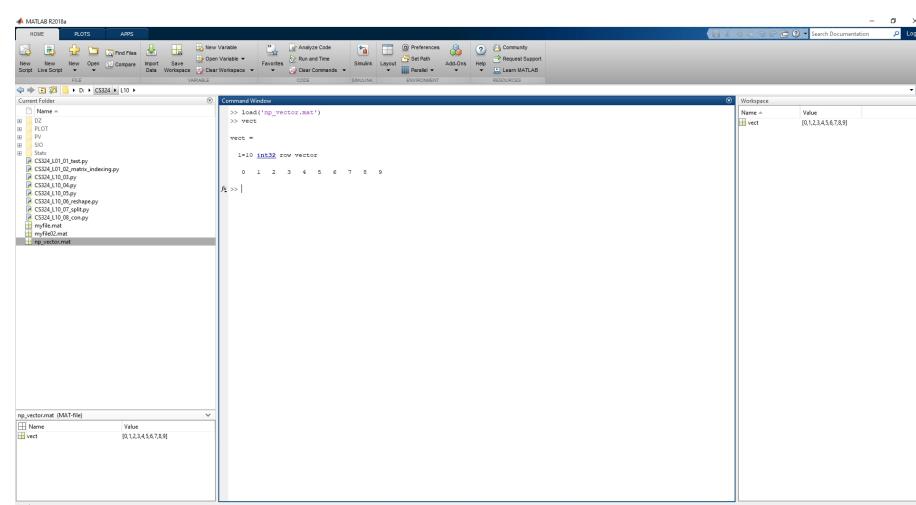
*Nizovi napravljeni u Python programskom jeziku mogu se izvesti savemat() funkcijom.*

### Izvoz nizova

Funkcija za izvoz niza napisanog u Python jeziku za MATLAB ili Octave format .mat može se postići funkcijom **savemat()**.

```
import numpy as np
import scipy.io as sio

vect = np.arange(10)
sio.savemat('np_vector.mat', {'vect':vect})
```



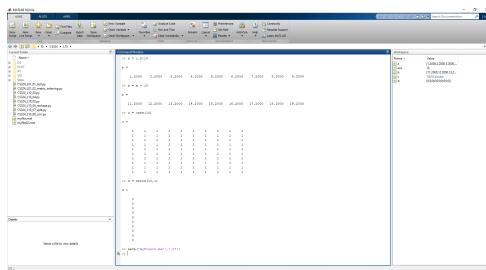
Slika 5.2 Otvaranje .mat datoteke nakon izvoza iz Python-a. [Izvor: Autor]

### Provera .mat niza

Moguće je proveriti oblik i tip sadržaja **.mat** datoteke bez eksplisitnog uvoza u Python promenljivu. Ovo se postiže funkcijom **whosmat()**.

```
import numpy as np
import scipy.io as sio

sio.whosmat('myfile02.mat')
```



Slika 5.3 Sadržaj MATLAB promenljive u MATLAB prozoru. [Izvor: Autor]

Slika 5.4 Sadržaj MATLAB promenljive u kroz whosmat funkciju. [Izvor: Autor]

## PRIMER UVOZA, IZVOZA I PREGLEDA .MAT DATOTEKA.

*Sledi autorski video o pregledu, uvozu i izvozu .mat datoteka u Python.*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 6

### Pokazne vežbe #10

#### ZADATAK #1 (12 MINUTA)

*Zadatak #1 odnosi se na napredno indeksiranje višedimenzionalnih numpy nizova*

##### Zadatak #1 (12 minuta)

Napraviti Numpy niz sa sledećim dimenzijama: (5,5,5). Elementi niza jesu nasumični celi brojevi od 0 do 100 sa seed-om 10.

Indeksirati:

- Središnji element
- Četvrta i peta kolona poslednje matrice,
- Sve redove druge matrice,
- Treći i četvrti element četvrtog reda pете matrice,

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

##### Objašnjenje i rešenje:

Funkcija **seed** dodeliće pseudo nasumične brojeve, tako da dobijene matrice imaju sledeće vrednosti:

$$\begin{bmatrix} 9 & 15 & 64 & 28 & 89 \\ 93 & 29 & 8 & 73 & 0 \\ 40 & 36 & 16 & 11 & 54 \\ 88 & 62 & 33 & 72 & 78 \\ 49 & 51 & 54 & 77 & 69 \end{bmatrix} \begin{bmatrix} 13 & 25 & 13 & 92 & 86 \\ 30 & 30 & 89 & 12 & 65 \\ 31 & 57 & 36 & 27 & 18 \\ 93 & 77 & 22 & 23 & 94 \\ 11 & 28 & 74 & 88 & 9 \end{bmatrix} \begin{bmatrix} 15 & 18 & 80 & 71 & 88 \\ 11 & 17 & 46 & 7 & 75 \\ 28 & 33 & 84 & 96 & 88 \\ 44 & 5 & 4 & 71 & 88 \\ 88 & 50 & 54 & 34 & 15 \end{bmatrix}$$
$$\begin{bmatrix} 77 & 88 & 15 & 6 & 85 \\ 22 & 11 & 12 & 92 & 96 \\ 62 & 57 & 79 & 42 & 57 \\ 97 & 50 & 45 & 40 & 89 \\ 73 & 37 & 0 & 18 & 23 \end{bmatrix} \begin{bmatrix} 3 & 29 & 16 & 84 & 82 \\ 14 & 51 & 79 & 17 & 50 \\ 53 & 25 & 48 & 17 & 32 \\ 81 & 80 & 41 & 90 & 12 \\ 30 & 81 & 17 & 16 & 0 \end{bmatrix}$$

```
import numpy as np

np.random.seed(10)
a = np.random.randint(0, 100, (5,5,5))
```

```
print(a)

# sredisnji element
print(a[2,2,2])

# Cetvrta i peta kolone poslednje matrice
print(a[4, :, 3:5])

#Svi redovi druge matrice
print(a[1,...,:])

# Treci i cetvrti element cetvrtog reda pete matrice
print(a[-1, 3, 2:4])
```

## ZADATAK #2 (13 MINUTA)

*Zadatak #2 odnosi se na formiranje nizova sa određenim vrednostima elemenata.*

### Zadatak #2 (13 minuta)

Napisati program koji će preračunati subnet masku napisanu u CIDR obliku (npr. /24) prebaciti u dekadni oblik. Koristiti NumPy **geomspace()** funkciju.

Primer: Za ulaz od **/24** treba vratiti **255.255.255.0**

#### Rešenje:

```
import numpy as np

cidr = int(input('CIDR maska je: '))
bin_mask = np.zeros(32, dtype='int32')
bin_mask[0:(cidr)] = np.ones(cidr, dtype='int32')

bin_list = np.split(bin_mask, 4)

dec_mask = np.around(np.geomspace(1, 128, num=8)).astype(int)
dec_mask = np.flip(dec_mask)

ABCD = []
for i in range(0,4):
    ABCD.append(np.dot(bin_list[i],dec_mask))

str_subnet = [str(x) for x in ABCD]
print('Subnet maska je: {}'.format('.'.join(str_subnet)))
```

```

File Edit Selection View Go Run Terminal Help CS324_L10_PV_02.py - L10 - Visual Studio Code

EXPLORER ... CS324_L10_07_sp14.py CS324_L10_08_copy.py CS324_L10_PV_01.py CS324_L10_PV_02.py
> L10
  > DZ
  > PV
    > CS324_L10_PV_01.py
    > CS324_L10_PV_02.py
    > CS324_L10_01_test.py
    > CS324_L10_02_matrix_index_
    > CS324_L10_03.py
    > CS324_L10_04.py
    > CS324_L10_05.py
    > CS324_L10_06_reshape.py
    > CS324_L10_07_split.py
    > CS324_L10_08_copy.py

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
PS D:\CS324\10 & C:/Users/nerdz/AppData/Local/Programs/Python/Python38-32/python.exe d:/CS324/L10/PV/CS324_L10_PV_02.py
CIDR maska je: 130
Subnet maska je: 255,255,255,252
PS D:\CS324\10 & C:/Users/nerdz/AppData/Local/Programs/Python/Python38-32/python.exe d:/CS324/L10/PV/CS324_L10_PV_02.py
CIDR maska je: 127
Subnet maska je: 255,255,255,224
PS D:\CS324\10

```

Slika 6.1 Zadatak #2. [Izvor: Autor]

## ZADATAK #3 (20 MINUTA)

*Zadatak #3 odnosi se na prikaz više grafikona na istoj figuri.*

### Zadatak #3 (20 minuta)

Napisati program za crtanje tri grafikona prema podacima datim u nastavku:

- Prvi grafikon (gornji levi ugao) je bar plot svih prebranjanih programskega jezika,
- Drugi grafikon (gornji levi ugao) je pie plot koji je raspodela prvih 5 programskega jezika.
- Treći grafikon (ceo donji grafikon) jeste plata JavaScript i Python programera prema uzrastu. Kada je plata Python programera ispod srednje vrednosti plate svih programera, popuniti crvenom bojom, a kada je iznad, popuniti plavom.

```

Responder_id,LanguagesWorkedWith
1,HTML/CSS Java JavaScript Python
2,C++ HTML/CSS Python
3,HTML/CSS
4,C C++ C# Python SQL
5,C++ HTML/CSS Java JavaScript Python SQL VBA
6,Java R SQL
7,HTML/CSS JavaScript
8,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Python SQL
9,Bash/Shell/PowerShell C# HTML/CSS JavaScript Python Ruby Rust SQL TypeScript
WebAssembly Other(s):
10,C# Go JavaScript Python R SQL
11,Other(s):
12,Bash/Shell/PowerShell HTML/CSS Java Python R SQL
13,Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL TypeScript
14,C++
15,Assembly Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript PHP SQL
16,Bash/Shell/PowerShell C# HTML/CSS JavaScript TypeScript VBA
17,Bash/Shell/PowerShell HTML/CSS JavaScript TypeScript
18,Python R

```

19,C# HTML/CSS Java JavaScript SQL TypeScript  
20,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript PHP Python R SQL  
21,Assembly Bash/Shell/PowerShell C C++ Go Java JavaScript Kotlin Python Rust SQL  
Swift  
22,Bash/Shell/PowerShell C++ HTML/CSS JavaScript Python Ruby SQL TypeScript  
23,Bash/Shell/PowerShell HTML/CSS JavaScript Python Ruby SQL  
24,HTML/CSS JavaScript PHP TypeScript  
25,HTML/CSS JavaScript PHP SQL TypeScript  
26,Bash/Shell/PowerShell C++ C# HTML/CSS JavaScript PHP Python Ruby SQL Swift  
TypeScript VBA  
27,C++ JavaScript Python Ruby SQL TypeScript  
28,JavaScript TypeScript  
29,Bash/Shell/PowerShell JavaScript SQL  
31,Python  
32,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python  
33,C++ Python R  
34,HTML/CSS JavaScript  
35,HTML/CSS JavaScript  
36,Java Kotlin Python  
37,Bash/Shell/PowerShell JavaScript Python Other(s):  
38,C# HTML/CSS JavaScript SQL  
39,C# JavaScript SQL TypeScript  
40,C# HTML/CSS  
41,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript SQL  
42,HTML/CSS JavaScript PHP TypeScript  
43,C++ C# HTML/CSS Java JavaScript Objective-C SQL  
44,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript SQL TypeScript WebAssembly  
45,Python  
46,Bash/Shell/PowerShell C C# HTML/CSS JavaScript PHP Python SQL Other(s):  
47,Java PHP Ruby  
48,HTML/CSS PHP SQL  
49,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP Python SQL TypeScript  
50,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript PHP Python SQL  
51,Bash/Shell/PowerShell C++ HTML/CSS Java JavaScript Python R TypeScript  
52,Bash/Shell/PowerShell C C++ Elixir Erlang Go HTML/CSS Java JavaScript Kotlin  
Python Ruby Rust SQL TypeScript  
53,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP SQL TypeScript  
54,Bash/Shell/PowerShell HTML/CSS JavaScript Python  
55,Java Python SQL  
56,Bash/Shell/PowerShell HTML/CSS Java SQL  
57,JavaScript Python  
58,C# Java SQL  
59,HTML/CSS JavaScript PHP SQL  
60,Bash/Shell/PowerShell Go JavaScript PHP Python Ruby SQL  
61,C++ C# HTML/CSS JavaScript PHP Python SQL VBA  
62,Bash/Shell/PowerShell C++ Go HTML/CSS Java JavaScript Kotlin PHP Python Ruby SQL  
TypeScript VBA  
63,Bash/Shell/PowerShell Clojure Java Python Other(s):  
64,Bash/Shell/PowerShell C C++ C#  
65,Assembly C C++ C# HTML/CSS Java  
66,Clojure Go HTML/CSS Java JavaScript R SQL  
67,C C# HTML/CSS Java JavaScript PHP Python SQL VBA  
68,HTML/CSS Java JavaScript Python

69,C# HTML/CSS Java JavaScript Objective-C SQL TypeScript  
70,C# HTML/CSS JavaScript SQL  
71,HTML/CSS JavaScript PHP Python SQL VBA  
72,C# HTML/CSS JavaScript PHP SQL TypeScript  
73,SQL  
74,HTML/CSS Java JavaScript Kotlin Python Ruby  
75,HTML/CSS JavaScript  
76,PHP SQL  
77,HTML/CSS JavaScript PHP SQL  
78,HTML/CSS Java JavaScript Kotlin Python  
79,C# HTML/CSS JavaScript TypeScript  
80,Bash/Shell/PowerShell C# F# Go HTML/CSS Java JavaScript Python SQL TypeScript  
81,Assembly Bash/Shell/PowerShell C C++ Python  
82,Bash/Shell/PowerShell C++ HTML/CSS Java JavaScript Python Rust  
83,HTML/CSS JavaScript  
84,C C++ C# Java Kotlin PHP SQL  
85,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL TypeScript  
86,Bash/Shell/PowerShell C# HTML/CSS JavaScript PHP Python SQL  
87,Bash/Shell/PowerShell C C++ C# Go HTML/CSS Java JavaScript Objective-C Python SQL  
88,C++ C# HTML/CSS Java JavaScript SQL TypeScript  
89,Bash/Shell/PowerShell Python  
90,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript Objective-C PHP Python Ruby  
SQL Swift TypeScript  
91,HTML/CSS JavaScript TypeScript  
92,HTML/CSS Java JavaScript Kotlin SQL VBA  
93,Python SQL  
94,C# HTML/CSS JavaScript SQL  
95,C# HTML/CSS JavaScript Python R SQL  
96,C C++ Java Python R Scala SQL  
97,Bash/Shell/PowerShell JavaScript Python R SQL  
98,HTML/CSS Java JavaScript SQL  
99,C C++ HTML/CSS Java JavaScript Kotlin PHP Python SQL  
100,HTML/CSS JavaScript Ruby SQL TypeScript  
101,Bash/Shell/PowerShell HTML/CSS JavaScript SQL  
102,C# HTML/CSS JavaScript SQL TypeScript  
103,Clojure Go Java Kotlin  
104,C# HTML/CSS TypeScript  
105,Bash/Shell/PowerShell HTML/CSS JavaScript SQL  
106,Bash/Shell/PowerShell HTML/CSS Java JavaScript Python SQL  
107,HTML/CSS Java Python  
108,C++ Python  
109,C# HTML/CSS JavaScript SQL  
110,Python SQL  
111,Bash/Shell/PowerShell C# HTML/CSS JavaScript Python SQL VBA  
112,Assembly C C++ C# HTML/CSS Java JavaScript Python VBA  
113,Bash/Shell/PowerShell C C++  
114,Bash/Shell/PowerShell C++ Erlang JavaScript PHP Python  
115,Assembly C# HTML/CSS Java JavaScript SQL Swift  
116,Scala Other(s):  
117,C# HTML/CSS JavaScript SQL TypeScript  
118,Bash/Shell/PowerShell C C++  
119,C C++ C# HTML/CSS Java JavaScript SQL  
120,Elixir HTML/CSS JavaScript Python Ruby SQL

121, Bash/Shell/PowerShell C C++ HTML/CSS JavaScript Python  
122, Assembly Bash/Shell/PowerShell C++ Python SQL  
123, HTML/CSS JavaScript TypeScript  
124, HTML/CSS Java SQL  
125, Bash/Shell/PowerShell Dart HTML/CSS Java JavaScript Scala  
126, C# SQL  
127, Bash/Shell/PowerShell C Python  
128, Bash/Shell/PowerShell Go Ruby  
129, C++ Python R  
130, Bash/Shell/PowerShell C++ Clojure HTML/CSS Java Python Ruby SQL  
131, Java  
132, Bash/Shell/PowerShell C++ C# HTML/CSS Java JavaScript Objective-C Python  
TypeScript  
133, JavaScript PHP SQL  
134, C C++ HTML/CSS JavaScript SQL  
135, Go HTML/CSS JavaScript TypeScript  
136, C# Java PHP Python  
137, HTML/CSS Java JavaScript PHP SQL  
139, Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL  
140, Java JavaScript Kotlin PHP SQL  
141, Assembly Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Python SQL  
TypeScript  
142, Bash/Shell/PowerShell C#  
143, C HTML/CSS Java JavaScript Python SQL TypeScript  
144, Java SQL  
145, Assembly C++ Python VBA  
146, Bash/Shell/PowerShell C Python Scala  
147, HTML/CSS Java JavaScript Kotlin  
148, HTML/CSS JavaScript TypeScript  
149, Bash/Shell/PowerShell HTML/CSS JavaScript Python SQL  
150, Bash/Shell/PowerShell C# HTML/CSS Java JavaScript Python SQL  
151, Bash/Shell/PowerShell C++ C# HTML/CSS Java JavaScript SQL TypeScript  
152, HTML/CSS Java JavaScript Kotlin Objective-C Python Swift Other(s):  
153, C C++ Python  
154, HTML/CSS Java JavaScript PHP SQL VBA  
155, Bash/Shell/PowerShell HTML/CSS JavaScript TypeScript  
156, Bash/Shell/PowerShell Python  
157, Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL  
158, Java SQL Other(s):  
159, Bash/Shell/PowerShell HTML/CSS JavaScript Python  
160, C#  
161, HTML/CSS JavaScript PHP  
162, Bash/Shell/PowerShell C++ HTML/CSS Java JavaScript PHP SQL  
163, C# HTML/CSS JavaScript SQL Other(s):  
164, C# HTML/CSS JavaScript SQL  
165, Assembly Bash/Shell/PowerShell C C++ C# JavaScript Python SQL  
166, Bash/Shell/PowerShell Go HTML/CSS Java JavaScript Ruby SQL  
167, HTML/CSS Java JavaScript PHP Python SQL  
168, Bash/Shell/PowerShell Python  
169, C C++ HTML/CSS Java JavaScript PHP Python SQL  
170, Clojure Go HTML/CSS Java JavaScript Python Ruby TypeScript  
171, Assembly C C++ C# HTML/CSS Java Objective-C PHP Python R SQL Swift  
172, HTML/CSS JavaScript Objective-C SQL

173,HTML/CSS JavaScript PHP SQL TypeScript  
174,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript PHP Python R  
175,Assembly Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript PHP Python SQL  
176,C C++ C# HTML/CSS Java JavaScript Python SQL  
177,C# Python SQL  
178,JavaScript PHP Python Ruby SQL TypeScript  
179,HTML/CSS Java JavaScript Objective-C SQL Swift  
181,HTML/CSS JavaScript  
182,Bash/Shell/PowerShell C C++ Go HTML/CSS Java JavaScript PHP Python Rust SQL  
TypeScript  
183,Dart Go Java Kotlin Swift  
184,Bash/Shell/PowerShell C C++ Java JavaScript Kotlin Objective-C Python Rust Swift  
185,Bash/Shell/PowerShell Java Kotlin  
186,Assembly C C++ HTML/CSS Java JavaScript Python  
187,HTML/CSS JavaScript Ruby Scala SQL  
188,C# Python R SQL VBA  
189,C++ HTML/CSS JavaScript Python SQL  
190,Java Scala SQL TypeScript  
191,C# HTML/CSS JavaScript SQL TypeScript  
192,Bash/Shell/PowerShell JavaScript Python  
193,Bash/Shell/PowerShell HTML/CSS JavaScript Ruby SQL  
194,Assembly Bash/Shell/PowerShell C HTML/CSS Java JavaScript PHP  
195,C# TypeScript Other(s):  
196,C# HTML/CSS JavaScript SQL TypeScript  
197,HTML/CSS R VBA  
198,HTML/CSS Java JavaScript Python SQL  
199,R SQL  
200,Java JavaScript SQL VBA  
201,Bash/Shell/PowerShell Go HTML/CSS Java JavaScript Python Scala  
202,Bash/Shell/PowerShell C C++ C# HTML/CSS Java JavaScript Objective-C Python SQL  
TypeScript  
203,C C++ HTML/CSS JavaScript PHP SQL  
204,Python Other(s):  
205,C# HTML/CSS JavaScript PHP  
206,Scala SQL  
207,Java  
208,Other(s):  
209,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL  
210,C++ Go Java JavaScript Python Rust WebAssembly  
211,C# HTML/CSS Java JavaScript Objective-C SQL  
212,C# HTML/CSS Java JavaScript PHP  
213,Go Python  
214,R Other(s):  
215,Python Rust  
216,Go Java Python  
217,Go HTML/CSS JavaScript PHP TypeScript  
218,HTML/CSS JavaScript PHP SQL  
219,C# HTML/CSS JavaScript SQL  
220,C# HTML/CSS JavaScript SQL  
221,C C++ HTML/CSS Java JavaScript PHP SQL  
222,Assembly Bash/Shell/PowerShell C HTML/CSS JavaScript Other(s):  
223,Java Kotlin  
224,Go

225,Bash/Shell/PowerShell C C++ C# HTML/CSS Java JavaScript PHP Python R SQL  
226,HTML/CSS JavaScript PHP Python SQL  
227,Bash/Shell/PowerShell HTML/CSS JavaScript Python SQL  
228,HTML/CSS Java JavaScript Python Scala SQL  
229,HTML/CSS Java JavaScript Python TypeScript  
230,Bash/Shell/PowerShell Go Python  
231,Elixir Go JavaScript Ruby SQL  
232,Bash/Shell/PowerShell C# Dart HTML/CSS JavaScript  
233,C++ Python  
234,Bash/Shell/PowerShell C Python  
235,HTML/CSS Java Python SQL  
236,HTML/CSS JavaScript  
237,C++ JavaScript Rust  
238,HTML/CSS JavaScript Objective-C PHP Python  
239,C# HTML/CSS JavaScript TypeScript  
240,Bash/Shell/PowerShell HTML/CSS Java JavaScript Python SQL  
241,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL  
242,HTML/CSS Java JavaScript SQL TypeScript  
243,C C++ C# HTML/CSS Java JavaScript Objective-C PHP Python Swift  
244,Assembly Bash/Shell/PowerShell C Python VBA  
245,C# JavaScript TypeScript  
246,C# HTML/CSS JavaScript SQL TypeScript  
247,C++ C#  
248,HTML/CSS Java JavaScript Python SQL  
249,Bash/Shell/PowerShell C# HTML/CSS Java Python SQL  
250,Python Other(s):  
251,Assembly C C++ C# HTML/CSS JavaScript Python VBA  
252,HTML/CSS JavaScript Rust Swift Other(s):  
253,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Ruby SQL  
254,C# Python SQL TypeScript  
255,Bash/Shell/PowerShell C HTML/CSS Java JavaScript Python TypeScript Other(s):  
256,C C++ Java  
257,Bash/Shell/PowerShell Other(s):  
258,C HTML/CSS Java JavaScript PHP Python  
259,C# HTML/CSS Java JavaScript Python SQL TypeScript  
260,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP SQL  
261,HTML/CSS JavaScript  
262,C++ C# HTML/CSS Java JavaScript PHP Python SQL TypeScript  
263,HTML/CSS JavaScript Kotlin Python  
264,HTML/CSS Java JavaScript Python SQL Other(s):  
265,HTML/CSS JavaScript  
266,HTML/CSS JavaScript TypeScript Other(s):  
267,HTML/CSS JavaScript Python Ruby  
268,C Java PHP Ruby  
269,C# HTML/CSS Java JavaScript SQL Swift  
270,C++ HTML/CSS JavaScript  
271,C# JavaScript SQL  
272,HTML/CSS JavaScript PHP  
273,Bash/Shell/PowerShell C++ Clojure Erlang HTML/CSS Java JavaScript Ruby Scala  
274,Java JavaScript  
275,C# HTML/CSS Java JavaScript Objective-C Python Ruby Rust SQL TypeScript  
276,HTML/CSS Java JavaScript Scala SQL  
277,Bash/Shell/PowerShell C C++ Python

278,Bash/Shell/PowerShell C# HTML/CSS JavaScript PHP Python SQL  
279,HTML/CSS Java JavaScript PHP Python SQL  
280,C# HTML/CSS JavaScript SQL TypeScript  
281,C# HTML/CSS Java JavaScript TypeScript  
282,Python SQL Swift  
283,HTML/CSS JavaScript TypeScript  
284,Assembly Bash/Shell/PowerShell C C++ C# Java  
285,C# HTML/CSS JavaScript PHP SQL  
286,Dart HTML/CSS JavaScript Python TypeScript Other(s):  
287,Java Kotlin  
288,HTML/CSS JavaScript PHP SQL  
289,C++ C# PHP Rust TypeScript  
290,HTML/CSS PHP Python  
291,Go HTML/CSS Python SQL TypeScript  
292,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL TypeScript  
293,HTML/CSS Java JavaScript SQL  
295,C C++ C# HTML/CSS Java JavaScript Kotlin PHP Python Ruby SQL TypeScript  
WebAssembly  
296,HTML/CSS Java JavaScript SQL  
297,HTML/CSS Java JavaScript SQL TypeScript  
298,HTML/CSS JavaScript PHP SQL Other(s):  
299,Assembly C C++ Java Python SQL  
300,HTML/CSS Java JavaScript SQL  
301,C++ C# Clojure JavaScript PHP Python SQL VBA  
302,Bash/Shell/PowerShell C# Python Other(s):  
303,JavaScript  
304,Bash/Shell/PowerShell Go HTML/CSS JavaScript Python Rust SQL  
305,C# HTML/CSS JavaScript SQL  
306,C HTML/CSS JavaScript Python SQL TypeScript  
307,Assembly Bash/Shell/PowerShell C HTML/CSS Java PHP Python R SQL  
308,HTML/CSS JavaScript Ruby  
309,Java  
310,C# HTML/CSS JavaScript SQL Swift  
311,Assembly C C++ C# HTML/CSS Java Python Scala SQL  
312,Bash/Shell/PowerShell Python Ruby SQL  
313,C C# HTML/CSS PHP Python  
314,C Python  
315,C++ HTML/CSS Java JavaScript Python SQL Other(s):  
316,Bash/Shell/PowerShell C++ HTML/CSS Java JavaScript Python  
317,Java SQL  
318,Bash/Shell/PowerShell Java  
319,Bash/Shell/PowerShell HTML/CSS JavaScript PHP TypeScript  
320,C# HTML/CSS JavaScript SQL TypeScript  
321,HTML/CSS Java JavaScript Ruby SQL TypeScript  
322,C# HTML/CSS Java Python  
323,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL  
324,C# HTML/CSS JavaScript SQL TypeScript  
325,Assembly Bash/Shell/PowerShell C C++ C# HTML/CSS Java JavaScript PHP Python  
326,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP Python SQL TypeScript  
327,HTML/CSS Java JavaScript PHP Python SQL  
328,HTML/CSS Java JavaScript TypeScript  
329,C C++ C# F# HTML/CSS JavaScript Rust SQL TypeScript Other(s):  
330,Java Kotlin

331,Bash/Shell/PowerShell C++ C# Python  
332,Bash/Shell/PowerShell HTML/CSS JavaScript Objective-C Python SQL Swift  
TypeScript  
333,Python Rust Scala  
334,Bash/Shell/PowerShell C C++ Python  
335,C# HTML/CSS Java JavaScript Python SQL Swift  
336,Bash/Shell/PowerShell JavaScript Python Rust SQL TypeScript  
337,Python R  
338,Bash/Shell/PowerShell HTML/CSS Java Python SQL  
339,Bash/Shell/PowerShell Dart Go HTML/CSS Java JavaScript Python Rust SQL Swift  
340,C# HTML/CSS SQL  
341,C++ C# SQL  
342,HTML/CSS Java JavaScript Python SQL TypeScript  
343,C++  
344,Assembly C# HTML/CSS JavaScript SQL  
345,Java JavaScript  
346,Java  
347,Java  
348,HTML/CSS PHP  
349,C C++ HTML/CSS Python  
350,C C++ Elixir HTML/CSS Java Kotlin PHP Ruby SQL Other(s):  
351,Assembly C C++ Java JavaScript Python SQL  
352,HTML/CSS JavaScript Ruby TypeScript  
353,HTML/CSS JavaScript TypeScript  
354,HTML/CSS Java JavaScript Kotlin TypeScript  
355,Bash/Shell/PowerShell Go HTML/CSS Java JavaScript Kotlin TypeScript  
356,HTML/CSS JavaScript PHP Python SQL  
357,HTML/CSS TypeScript  
358,C C++ C# HTML/CSS JavaScript PHP SQL TypeScript  
359,Java JavaScript Python  
360,Bash/Shell/PowerShell HTML/CSS JavaScript Python TypeScript  
361,Elixir Erlang F# Go HTML/CSS JavaScript PHP SQL  
362,HTML/CSS Java JavaScript SQL  
363,Bash/Shell/PowerShell HTML/CSS Java JavaScript Objective-C Python SQL TypeScript  
364,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL TypeScript  
365,Java SQL  
366,HTML/CSS PHP SQL  
367,HTML/CSS JavaScript  
368,Go HTML/CSS JavaScript PHP Python SQL TypeScript  
369,Python  
370,R SQL  
371,Bash/Shell/PowerShell SQL  
372,Bash/Shell/PowerShell C# HTML/CSS Java Python SQL  
373,C# HTML/CSS Java JavaScript Python  
374,HTML/CSS Python SQL TypeScript  
375,Bash/Shell/PowerShell HTML/CSS Java JavaScript Python SQL  
376,C++ Python  
377,HTML/CSS JavaScript PHP SQL  
378,C# HTML/CSS JavaScript SQL TypeScript  
379,Bash/Shell/PowerShell C C# Java Python  
380,Bash/Shell/PowerShell JavaScript Python  
381,C# Go HTML/CSS Java JavaScript Python R SQL  
382,C# HTML/CSS Java JavaScript PHP Python SQL

383,Bash/Shell/PowerShell  
384,Bash/Shell/PowerShell Go SQL TypeScript  
385,HTML/CSS Java JavaScript Python R SQL VBA  
386,HTML/CSS JavaScript PHP  
387,Java Python  
388,C HTML/CSS Java JavaScript PHP SQL TypeScript  
389,Bash/Shell/PowerShell JavaScript Objective-C Ruby Swift  
390,Bash/Shell/PowerShell HTML/CSS Java Kotlin Python Scala  
391,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL  
392,Bash/Shell/PowerShell C# HTML/CSS JavaScript Python SQL TypeScript VBA  
393,Java Kotlin PHP SQL  
394,C# HTML/CSS TypeScript Other(s):  
395,HTML/CSS Java JavaScript SQL  
396,HTML/CSS JavaScript  
397,HTML/CSS JavaScript PHP Ruby Rust SQL  
398,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Python SQL VBA  
399,C++  
400,Bash/Shell/PowerShell C++ Java Kotlin PHP  
401,JavaScript Python Ruby SQL TypeScript  
402,C#  
403,Bash/Shell/PowerShell C C# F# HTML/CSS JavaScript PHP Python SQL  
404,C# SQL TypeScript  
405,Java JavaScript TypeScript  
406,HTML/CSS Java JavaScript PHP SQL TypeScript  
407,C C++ HTML/CSS Java JavaScript Python SQL  
408,Assembly Bash/Shell/PowerShell C C++ Go HTML/CSS Java JavaScript Python R Rust  
SQL TypeScript WebAssembly  
409,Bash/Shell/PowerShell C# Go HTML/CSS JavaScript PHP  
410,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL  
411,C C++ JavaScript Python R SQL  
412,HTML/CSS JavaScript PHP Python SQL  
413,Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL TypeScript  
414,Bash/Shell/PowerShell JavaScript Python SQL  
415,HTML/CSS JavaScript Ruby  
416,C C++ C# HTML/CSS JavaScript SQL TypeScript  
417,HTML/CSS JavaScript  
418,HTML/CSS JavaScript SQL VBA  
419,C C++ C# HTML/CSS Java JavaScript PHP Python R SQL  
420,HTML/CSS JavaScript Python  
421,Bash/Shell/PowerShell HTML/CSS JavaScript TypeScript  
422,Bash/Shell/PowerShell C C# HTML/CSS JavaScript Python Other(s):  
423,C++ C#  
424,C C++  
425,Bash/Shell/PowerShell C HTML/CSS JavaScript PHP SQL VBA  
426,Java  
427,C# HTML/CSS JavaScript SQL TypeScript  
428,Bash/Shell/PowerShell Java Python  
429,C#  
430,Assembly C++ C# HTML/CSS Java JavaScript Objective-C PHP SQL Other(s):  
431,VBA Other(s):  
432,HTML/CSS Java JavaScript SQL TypeScript VBA Other(s):  
433,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Python  
434,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL

435,Bash/Shell/PowerShell C# HTML/CSS JavaScript Python SQL Swift  
436,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP Python SQL  
438,Bash/Shell/PowerShell C C++ Go HTML/CSS JavaScript R SQL  
439,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript PHP Python SQL TypeScript  
440,Bash/Shell/PowerShell C++ HTML/CSS Java PHP SQL  
441,C# HTML/CSS Java JavaScript PHP  
442,HTML/CSS JavaScript  
443,HTML/CSS JavaScript TypeScript  
444,Bash/Shell/PowerShell C C++ HTML/CSS Python  
445,C# SQL  
446,Bash/Shell/PowerShell HTML/CSS Java JavaScript Python  
447,Bash/Shell/PowerShell Java Python  
448,Bash/Shell/PowerShell Java SQL Other(s):  
449,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript TypeScript Other(s):  
450,C# HTML/CSS JavaScript TypeScript  
451,Bash/Shell/PowerShell C C++ C# Java Python Rust  
452,SQL VBA  
453,C++ C# SQL  
454,Bash/Shell/PowerShell Elixir HTML/CSS JavaScript Ruby SQL Other(s):  
455,C++ Python Ruby  
456,Bash/Shell/PowerShell JavaScript Python SQL  
457,HTML/CSS JavaScript  
458,Assembly C Python Rust  
459,C# SQL  
460,Bash/Shell/PowerShell C# HTML/CSS JavaScript PHP Python  
461,HTML/CSS JavaScript PHP SQL  
462,Bash/Shell/PowerShell C# HTML/CSS JavaScript Python SQL  
463,HTML/CSS Java JavaScript PHP SQL Other(s):  
464,C Python  
465,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL TypeScript  
466,HTML/CSS JavaScript TypeScript  
467,Java Kotlin Python SQL  
468,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript Kotlin SQL Other(s):  
469,Bash/Shell/PowerShell JavaScript PHP SQL  
470,HTML/CSS Java JavaScript PHP SQL TypeScript Other(s):  
471,Java JavaScript  
472,HTML/CSS JavaScript PHP Python SQL  
473,C# JavaScript  
474,Java JavaScript Other(s):  
475,C# HTML/CSS JavaScript PHP TypeScript  
476,HTML/CSS JavaScript PHP Scala  
477,Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL TypeScript  
478,C++ HTML/CSS JavaScript PHP Python Ruby SQL  
479,C# HTML/CSS JavaScript SQL Other(s):  
480,C C++ Dart Java Python  
481,Bash/Shell/PowerShell Kotlin  
482,Bash/Shell/PowerShell HTML/CSS PHP  
483,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript PHP Python SQL  
484,HTML/CSS Java PHP Python SQL  
485,Java JavaScript Scala  
486,Bash/Shell/PowerShell HTML/CSS JavaScript TypeScript  
487,Bash/Shell/PowerShell C++ C# HTML/CSS JavaScript PHP SQL TypeScript  
488,C C++ C# Java

489,C++ Python  
490,HTML/CSS JavaScript Ruby  
491,Java Objective-C  
492,Bash/Shell/PowerShell Go HTML/CSS JavaScript  
493,C# Java SQL Other(s):  
494,Assembly C Erlang HTML/CSS Java JavaScript Python SQL  
495,R  
496,Bash/Shell/PowerShell Python SQL  
497,Dart JavaScript Kotlin PHP SQL Swift TypeScript  
498,C++ HTML/CSS JavaScript Python  
499,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP Ruby SQL VBA  
500,C++ C#  
501,HTML/CSS  
502,C++ C#  
503,Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL TypeScript  
504,Bash/Shell/PowerShell Python SQL  
505,C C++ HTML/CSS Java SQL  
506,Java  
507,Go HTML/CSS Java JavaScript PHP Python SQL TypeScript  
508,C C++ HTML/CSS JavaScript  
509,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL  
510,Python  
511,HTML/CSS JavaScript Ruby SQL TypeScript  
512,Bash/Shell/PowerShell Python Other(s):  
513,C++ Rust  
514,HTML/CSS JavaScript Python Ruby TypeScript  
515,C++ HTML/CSS Java JavaScript PHP Python SQL  
516,Bash/Shell/PowerShell C++ Python  
517,Assembly C++ F# HTML/CSS PHP Python R Rust TypeScript Other(s):  
518,C HTML/CSS Java JavaScript SQL  
519,HTML/CSS PHP  
520,HTML/CSS Java JavaScript Python SQL TypeScript  
521,HTML/CSS JavaScript  
522,HTML/CSS Python  
523,Bash/Shell/PowerShell HTML/CSS JavaScript Python R  
524,Python Other(s):  
525,C# SQL VBA  
526,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL  
527,HTML/CSS JavaScript Objective-C Swift  
528,Bash/Shell/PowerShell Java JavaScript Python SQL  
529,Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP Python SQL TypeScript  
530,C C++ C#  
531,C#  
532,Bash/Shell/PowerShell C C++ Java Python  
533,C# Java JavaScript SQL  
535,Elixir VBA  
536,C# HTML/CSS Java JavaScript PHP Python SQL  
537,C# HTML/CSS JavaScript SQL TypeScript Other(s):  
538,HTML/CSS Java JavaScript PHP R SQL TypeScript  
539,Bash/Shell/PowerShell JavaScript Python SQL  
540,JavaScript Python  
541,Bash/Shell/PowerShell C# HTML/CSS JavaScript Python SQL TypeScript Other(s):  
542,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript Python R SQL

543,C C# HTML/CSS Java JavaScript Kotlin PHP TypeScript  
544,Assembly Bash/Shell/PowerShell C C++ Go HTML/CSS JavaScript Objective-C PHP  
Python SQL  
545,Bash/Shell/PowerShell C++ C# Go HTML/CSS JavaScript PHP Python SQL TypeScript  
Other(s):  
546,C Java SQL  
547,C# HTML/CSS JavaScript SQL  
548,Python SQL VBA  
550,C# HTML/CSS JavaScript SQL  
551,Assembly Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Python R Scala SQL  
552,HTML/CSS Java JavaScript PHP  
553,C# HTML/CSS JavaScript Python SQL TypeScript  
554,Go JavaScript PHP Ruby SQL TypeScript  
555,C# HTML/CSS JavaScript SQL  
556,Bash/Shell/PowerShell HTML/CSS Java JavaScript Python SQL  
557,C C++ PHP Python SQL VBA  
558,Bash/Shell/PowerShell C HTML/CSS Java JavaScript Kotlin PHP SQL TypeScript  
559,Bash/Shell/PowerShell C C++ C# HTML/CSS Java PHP Python SQL VBA  
560,Bash/Shell/PowerShell C# HTML/CSS JavaScript PHP SQL  
561,HTML/CSS Java JavaScript Python Ruby SQL  
562,C# JavaScript SQL  
563,Assembly Bash/Shell/PowerShell C C++ Python SQL  
564,C C++ Python R SQL Other(s):  
565,Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL  
566,Bash/Shell/PowerShell C# HTML/CSS JavaScript  
567,Bash/Shell/PowerShell C Java JavaScript Objective-C Rust Swift Other(s):  
568,C++ C# Clojure HTML/CSS JavaScript Python SQL  
569,Bash/Shell/PowerShell HTML/CSS JavaScript Python  
571,Assembly Bash/Shell/PowerShell C C++ Erlang R Rust Scala  
572,Assembly C C++ Java JavaScript Kotlin PHP Python Rust SQL TypeScript  
573,Assembly Bash/Shell/PowerShell C HTML/CSS PHP Python SQL  
574,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python Ruby  
575,Bash/Shell/PowerShell Go Python SQL  
576,C# HTML/CSS JavaScript Ruby SQL TypeScript  
577,Bash/Shell/PowerShell Python Other(s):  
578,Bash/Shell/PowerShell HTML/CSS Java JavaScript SQL Swift  
579,Bash/Shell/PowerShell C++ C# HTML/CSS Java JavaScript Python  
580,Bash/Shell/PowerShell HTML/CSS Java JavaScript Python Scala SQL  
581,Bash/Shell/PowerShell HTML/CSS JavaScript PHP TypeScript  
582,C C++ C# HTML/CSS Java JavaScript PHP Python SQL  
583,Java Ruby Scala SQL Other(s):  
584,Java Python  
585,C# Java Python R VBA  
586,Java Python Ruby  
587,Bash/Shell/PowerShell C# Java Other(s):  
588,C#  
589,C# HTML/CSS JavaScript SQL VBA  
590,C# SQL  
591,C# JavaScript  
592,C C++ HTML/CSS PHP Python SQL  
594,HTML/CSS Java JavaScript Kotlin PHP  
595,C# HTML/CSS JavaScript SQL TypeScript  
596,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Python SQL TypeScript

597, Java JavaScript PHP Python SQL  
598, Assembly Bash/Shell/PowerShell C C++ C# HTML/CSS JavaScript PHP Python SQL  
599, Bash/Shell/PowerShell HTML/CSS JavaScript PHP Ruby SQL  
600, Bash/Shell/PowerShell HTML/CSS Java PHP Python SQL  
601, HTML/CSS PHP SQL  
602, C++ C# HTML/CSS Java JavaScript PHP Python SQL Swift TypeScript  
603, Bash/Shell/PowerShell HTML/CSS Java JavaScript  
604, Bash/Shell/PowerShell C# HTML/CSS JavaScript PHP SQL TypeScript  
605, Java Kotlin  
606, Bash/Shell/PowerShell Clojure HTML/CSS Java JavaScript PHP Python Ruby  
TypeScript  
607, C# JavaScript SQL  
608, Bash/Shell/PowerShell C C++ Erlang Go HTML/CSS Java Python SQL  
609, HTML/CSS JavaScript Python  
610, Bash/Shell/PowerShell C C++ Go HTML/CSS Java JavaScript PHP Python Rust SQL  
TypeScript  
Other(s):  
611, Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL  
612, Python Other(s):  
613, Assembly Bash/Shell/PowerShell C C++ C# F# HTML/CSS JavaScript Python TypeScript  
614, Bash/Shell/PowerShell C C++ C# HTML/CSS Java Python SQL TypeScript  
615, C# HTML/CSS JavaScript Python SQL TypeScript VBA  
616, Ruby SQL  
617, Bash/Shell/PowerShell C# HTML/CSS Java JavaScript SQL  
618, C C++ C# Java PHP SQL  
619, Assembly Bash/Shell/PowerShell C C++ C# HTML/CSS Java JavaScript PHP SQL  
620, HTML/CSS Java JavaScript SQL TypeScript WebAssembly  
621, Bash/Shell/PowerShell SQL Other(s):  
622, Java Objective-C SQL Swift  
623, Objective-C Swift  
624, C++ HTML/CSS JavaScript PHP Python  
625, Bash/Shell/PowerShell HTML/CSS JavaScript Ruby TypeScript  
626, HTML/CSS Java JavaScript SQL  
627, Assembly Bash/Shell/PowerShell C++ C# Go HTML/CSS JavaScript Objective-C PHP  
Python Ruby SQL Swift  
628, Bash/Shell/PowerShell JavaScript SQL TypeScript  
629, Assembly Bash/Shell/PowerShell C C++ C# HTML/CSS Java Objective-C SQL  
630, C# HTML/CSS JavaScript SQL TypeScript  
631, Bash/Shell/PowerShell Go HTML/CSS JavaScript Python SQL  
632, C# HTML/CSS JavaScript SQL  
633, C C++ Python Rust  
634, C C++ C# HTML/CSS Java JavaScript Objective-C Scala SQL Swift TypeScript  
635, Bash/Shell/PowerShell JavaScript Python  
636, Bash/Shell/PowerShell C C++ Go Java  
637, C++ C# HTML/CSS Java JavaScript SQL  
638, Bash/Shell/PowerShell C# HTML/CSS JavaScript Rust  
639, HTML/CSS JavaScript Kotlin SQL Swift  
640, HTML/CSS JavaScript SQL  
641, C# HTML/CSS Java JavaScript Kotlin PHP SQL Swift  
642, Bash/Shell/PowerShell C++ HTML/CSS JavaScript TypeScript  
643, Bash/Shell/PowerShell HTML/CSS Java JavaScript PHP Ruby SQL  
645, C++ C# HTML/CSS SQL  
646, JavaScript PHP Python SQL VBA  
647, Assembly Bash/Shell/PowerShell C C++ C# HTML/CSS Rust SQL TypeScript

648,C# HTML/CSS JavaScript SQL VBA  
649,Java JavaScript Python SQL  
650,Bash/Shell/PowerShell C++ JavaScript Python  
651,Python  
652,HTML/CSS Java  
653,C# HTML/CSS JavaScript Kotlin Objective-C PHP SQL Swift  
654,Clojure HTML/CSS Java JavaScript  
655,Assembly C++ C# Java Scala Other(s):  
656,C++ Python R Other(s):  
657,Bash/Shell/PowerShell HTML/CSS Java JavaScript SQL  
658,C++ HTML/CSS Java JavaScript TypeScript  
659,Python  
660,HTML/CSS Java JavaScript Python SQL TypeScript  
661,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL  
662,Python SQL VBA  
663,HTML/CSS JavaScript Python SQL  
664,HTML/CSS Java JavaScript PHP  
665,Ruby SQL  
666,Bash/Shell/PowerShell HTML/CSS JavaScript Python SQL  
667,HTML/CSS JavaScript  
668,Bash/Shell/PowerShell HTML/CSS JavaScript Python SQL  
669,Bash/Shell/PowerShell C Python Rust SQL  
670,C# HTML/CSS JavaScript SQL Swift  
671,Bash/Shell/PowerShell HTML/CSS Java JavaScript Kotlin PHP Python  
672,C C++ HTML/CSS Java JavaScript PHP SQL  
673,C C++ C# HTML/CSS JavaScript PHP SQL  
674,C# HTML/CSS JavaScript Python  
675,C# HTML/CSS Java JavaScript SQL VBA Other(s):  
676,Bash/Shell/PowerShell Objective-C Ruby Swift  
677,Other(s):  
678,Java JavaScript Scala SQL  
679,C# HTML/CSS Python SQL  
680,C# HTML/CSS JavaScript SQL  
681,Bash/Shell/PowerShell C C++ C# F# HTML/CSS Java JavaScript Python R Ruby Scala SQL  
682,HTML/CSS Objective-C Swift  
683,HTML/CSS JavaScript PHP Python SQL  
684,C# HTML/CSS JavaScript TypeScript  
685,C# HTML/CSS Java SQL  
686,C# SQL  
687,C C++ Java Python SQL Other(s):  
688,HTML/CSS JavaScript PHP Python SQL  
689,C++ C# HTML/CSS Java JavaScript Other(s):  
690,Assembly  
691,HTML/CSS JavaScript PHP SQL  
692,HTML/CSS Java JavaScript SQL  
693,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Python SQL  
694,C# Go HTML/CSS Java JavaScript PHP Python SQL VBA  
695,C# HTML/CSS JavaScript TypeScript  
696,C# SQL  
697,Bash/Shell/PowerShell HTML/CSS JavaScript Python  
698,HTML/CSS Java JavaScript Python  
699,Bash/Shell/PowerShell C C++ C# HTML/CSS JavaScript SQL

700,HTML/CSS JavaScript Python SQL TypeScript  
701,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Rust SQL  
702,C C++ HTML/CSS JavaScript PHP Python SQL  
703,Bash/Shell/PowerShell C# HTML/CSS JavaScript SQL  
704,Dart HTML/CSS Java PHP SQL  
705,Go JavaScript  
706,Java Python Scala SQL  
707,C# HTML/CSS JavaScript Objective-C PHP Ruby SQL  
708,Bash/Shell/PowerShell C++ Go JavaScript Python SQL  
709,JavaScript TypeScript  
710,Assembly C HTML/CSS Java JavaScript Python R  
711,Java  
712,HTML/CSS JavaScript  
713,HTML/CSS JavaScript PHP SQL  
714,Bash/Shell/PowerShell HTML/CSS JavaScript PHP SQL VBA  
715,HTML/CSS JavaScript PHP SQL  
716,HTML/CSS JavaScript PHP Other(s):  
717,C# HTML/CSS Java PHP SQL  
718,C# HTML/CSS JavaScript SQL TypeScript  
719,Bash/Shell/PowerShell C++ C# HTML/CSS Java JavaScript Objective-C PHP Python  
SQL Swift  
720,C# HTML/CSS JavaScript PHP SQL TypeScript  
721,Bash/Shell/PowerShell Go SQL TypeScript  
723,JavaScript Python R Other(s):  
724,Bash/Shell/PowerShell C C++ C# HTML/CSS Java JavaScript Python  
725,C# HTML/CSS JavaScript SQL  
726,HTML/CSS JavaScript PHP SQL TypeScript  
727,Bash/Shell/PowerShell C C++ HTML/CSS JavaScript Python SQL  
728,Python  
729,Bash/Shell/PowerShell Go HTML/CSS JavaScript Python SQL  
730,C HTML/CSS JavaScript PHP SQL  
731,C C++ HTML/CSS JavaScript PHP SQL  
732,Bash/Shell/PowerShell C# F# HTML/CSS JavaScript SQL TypeScript  
733,C# HTML/CSS JavaScript SQL  
734,C# SQL  
735,Bash/Shell/PowerShell Java JavaScript Kotlin Python  
736,C# HTML/CSS Java JavaScript Objective-C SQL TypeScript  
737,C C++ Python  
738,Bash/Shell/PowerShell C C++ HTML/CSS Java Python R SQL  
739,Bash/Shell/PowerShell Java JavaScript Swift  
740,Bash/Shell/PowerShell Java PHP Python  
741,C++ C# HTML/CSS Java JavaScript PHP Python SQL TypeScript  
742,Go JavaScript Python SQL TypeScript  
743,HTML/CSS JavaScript PHP Ruby  
744,C C++ C# HTML/CSS JavaScript Kotlin Swift  
745,Bash/Shell/PowerShell HTML/CSS Java JavaScript Kotlin PHP SQL TypeScript  
746,Go HTML/CSS Python  
747,Bash/Shell/PowerShell C# HTML/CSS Java JavaScript Python R Ruby SQL TypeScript  
748,Java JavaScript  
749,C# HTML/CSS JavaScript SQL  
750,Dart HTML/CSS Java Kotlin Python  
751,HTML/CSS JavaScript PHP Python SQL  
752,C#

```
753,C# HTML/CSS JavaScript TypeScript
754,Bash/Shell/PowerShell HTML/CSS JavaScript PHP Other(s):
755,Bash/Shell/PowerShell C++ HTML/CSS Java JavaScript Python SQL
756,Bash/Shell/PowerShell C C++ HTML/CSS Java JavaScript Kotlin PHP Python SQL
Other(s):
757,C SQL
758,Java Python
759,Bash/Shell/PowerShell Java Python Scala
760,C# JavaScript SQL TypeScript
761,C# HTML/CSS Java JavaScript PHP Python SQL TypeScript VBA
762,C# HTML/CSS JavaScript SQL
763,Python Rust TypeScript
764,C# HTML/CSS JavaScript SQL
765,Java JavaScript Ruby
766,HTML/CSS JavaScript PHP
767,HTML/CSS JavaScript PHP SQL
768,Bash/Shell/PowerShell C#
```

```
godine = np.array([20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30])

python_zarada = np.array([20000, 24744, 30500, 37732, 41247, 45372, 48876, 53850,
57287, 63016, 65998])
javascript_zarada = np.array([18942, 21780, 25704, 29000, 34372, 37810, 43515,
46823, 49293, 53437, 56373])
prosecna_zarada = np.array([18012, 20628, 25206, 30252, 34368, 38496, 42000, 46752,
49320, 53200, 56000])
```

```
import numpy as np
from matplotlib import pyplot as plt
import csv
from collections import Counter

godine = np.array([20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30])

python_zarada = np.array([20000, 24744, 30500, 37732, 41247, 45372, 48876, 53850,
57287, 63016, 65998])
javascript_zarada = np.array([18942, 21780, 25704, 29000, 34372, 37810, 43515,
46823, 49293, 53437, 56373])
prosecna_zarada = np.array([18012, 20628, 25206, 30252, 34368, 38496, 42000, 46752,
49320, 53200, 56000])
prosek = np.mean(prosecna_zarada)

with open('data.csv') as csv_file:
    csv_reader = csv.DictReader(csv_file)

    language_counter = Counter()
    for row in csv_reader:
        language_counter.update(row['LanguagesWorkedWith'].split(';'))

jezici = []
popularnost = []
```

```

for item in language_counter.most_common(5):
    jezici.append(item[0])
    popularnost.append(item[1])

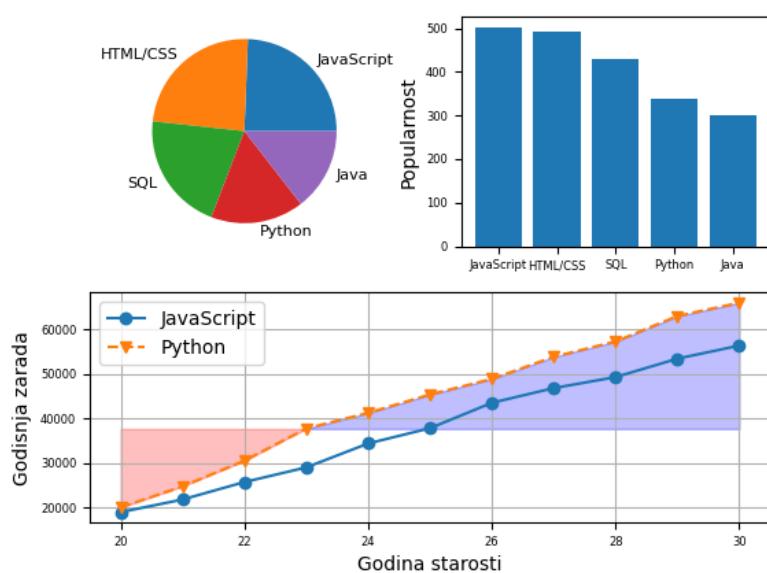
fig = plt.figure()

# dva reda, dve kolone, prva celija
sub1 = fig.add_subplot(2,2,1)
sub1.pie(popularnost, labels=jezici, textprops={'fontsize': 8})

# dva reda, dve kolone, druga celija
sub2 = fig.add_subplot(2,2,2)
sub2.bar(jezici, popularnost)
plt.ylabel('Popularnost')
plt.tick_params(axis='both', which='major', labelsize=6)

# dva reda, dve kolone, kombinovana prva i druga celija
sub3 = fig.add_subplot(2,2,(3,4))
sub3.plot(godine, javascript_zarada, 'o-', label='JavaScript')
sub3.plot(godine, python_zarada, 'v--', label='Python')
plt.fill_between(godine, python_zarada, prosek, where=(python_zarada >= prosek),
interpolate=True, color='blue', alpha=0.25)
plt.fill_between(godine, python_zarada, prosek, where=(python_zarada < prosek),
interpolate=True, color='red', alpha=0.25)
plt.xlabel('Godina starosti')
plt.ylabel('Godisnja zarada')
plt.tick_params(axis='both', which='major', labelsize=6)
plt.legend()
plt.grid()
plt.show()

```



Slika 6.2 Popularnost programskih jezika. [Izvor: Autor]

## ✓ Poglavlje 7

### Individualne vežbe #10

#### ZADATAK #1 | ZADATAK #2

*Zadaci #1 i #2 rade se po 25 minuta*

##### **Zadatak #1 (25 minuta)**

Napisati u jednoj datoteci program koji računa funkcije:

$$y(x) = \sin(x) - 3$$
$$g(x) = \cos(x) + 3$$

Program radi u 1000 iteracija. Svaka iteracija kasni jednu sekundu i čuva podatke u datoteci "trig.csv".

U drugoj datoteci u realnom vremenu učitavati podatke iz "trig.csv" i iscrtavati kao linijski plot.

##### **Zadatak #2 (25 minuta)**

Napraviti Numpy niz sa sledećim dimenzijama: (10,10,10). Elementi niza jesu nasumični celi brojevi od -50 do 50 sa seed-om 7.

Indeksirati:

- Nultu, drugu, četvrtu, šestu i osmu matricu,
- Sve prve redove neparnih matrica,
- Od prvog do osmog elementa prvog reda sedme, osme i devete matrice,
- Sve redove poslednje matrice,
- Peti, šesti i sedmi element četvrtog reda pretposlednje matrice

#### ZADATAK #3

*Zadatak #3 radi se 40minuta*

##### **Zadatak #3 (40 minuta)**

Nacrtati tri vertikalna grafikona. Prvi grafikon računa funkciju:

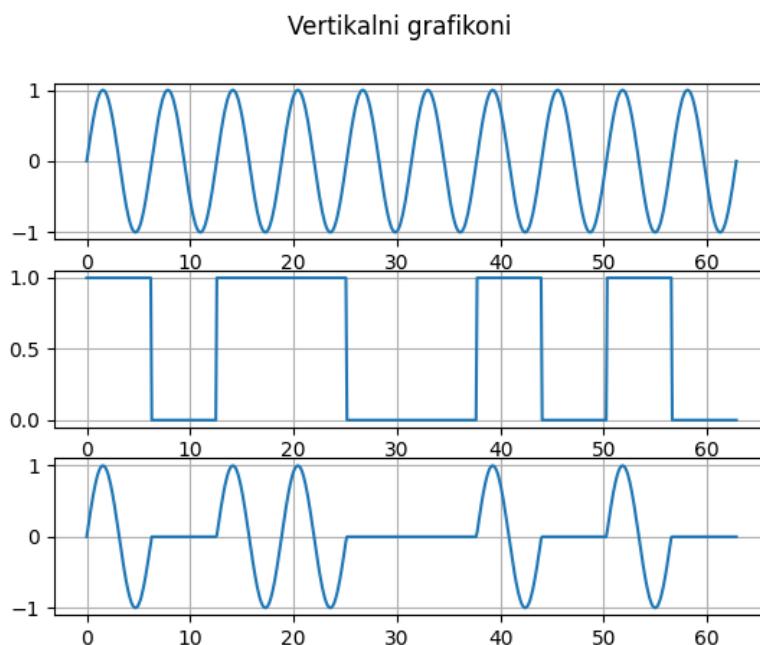
$$y = \sin(x)$$

Dužina X-ose jeste tolika da stanu tačno 10 puna ciklusa funkcije, a broj elemenata niza treba da bude deljiv sa deset.

Zatim, formirati nasumični NumPy niz sa nulama i jedinicama po uniformnoj raspodeli od ukupno deset elemenata. Korišćenjem funkcije **np.repeat()** produžiti svaki element niza nula i jedinica tako da se izjednači sa brojem elemenata niza trigonometrijske funkcije. Prikazati ovaj niz kao drugi grafikon.

Konačno, u trećem nizu pomnožiti trigonometrijsku funkciju sa nulama i jedinicama i prikazati kao treći grafikon.

Konačni grafikon (sa **np.random.seed(20)**) treba da ima sledeći oblik:



Slika 7.1 Izgled vertikalnih grafikona ako se koristi seed=20. [Izvor: Autor]

## ✓ Poglavlje 8

### Domaći zadatak #10

#### TEKST DOMAĆEG ZADATKA #10

*Domaći zadatak radi se okvirno 3h.*

##### **Domaći zadatak #10 (3h)**

Napisati python program koji beleži informacije o položenim ispitima:

Student upisuje sa konzole položeni ispit, najpre šifru predmeta, pa ocenu. Ova informacija se čuva **u imeniku**.

Iz imenika, preko **pandas** paketa napraviti **CSV** datoteku u kojoj je jedna kolona **Predmet** (vrednosti su šifre predmeta) a druga kolona **Ocena** (vrednosti su ocene).

Iz ovih podataka, računati:

- Broj ocena (koliko ima šestica, sedmica, itd).
- Broj položenih ispita po šifri predmeta (koliko ima CS predmeta, IT predmeta, SE predmeta, itd.)
- Promenu prosečne ocene (kako se menja prosečna ocena po položenom predmetu)

Na jednoj figuri iscrtati raspored ocena kao **pie plot** (gornji levi subplot), broj položenih predmeta kao **horizontalni bar plot** (gornji deni subplot) i promenu prosečne ocene kao linijski plot (donji subplot cele dužine).

Snimiti Figuru kao **Student\_izvestaj.png**, i CSV datoteku kao **Student\_izvestaj.csv**.

##### **Predaja domaćeg zadatka**

##### **Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije 7 dana nakon predavanja, za 100% poena. Nakon toga poeni se umanjuju za 50%.

##### **Internet studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganje ispita. Domaći zadaci se brane!

Domaći zadatak poslati dr Nemanji Zdravkoviću: nemanja.zdravkovic@metropolitan.ac.rs

Obavezno koristiti uputstvo za izradu domaćeg zadatka.

Uz .doc dokument (koji treba sadržati i screenshot svakog urađenog zadatka kao i komentare za zadatak), poslati i izvorne i dodatne datoteke.

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Rezime lekcije #10*

#### **Rezime:**

U ovoj lekciji bilo je reči o paketu SciPy i dodatnim funkcionalnostima paketa NumPy i matplotlib.

Paket matplotlib radi sa vizuelizacijom podataka i prikazom različitih grafikona. Matplotlib omogućuje detaljno podešavanje grafikona.

NumPy omogućava rad sa višedimenzionalnim nizovima i kreaciju "gotovih" nizova sa određenim brojem elemenata ili sa određenim korakom između elemenata.

SciPy paket omogućava i konverziju nizova sa drugim programskim jezicima bez poteškoća.

#### **Literatura:**

- David Beazley, Brian Jones, *Python Cookbook: Recipes for Mastering Python 3*, 3rd edition, O'Reilly Press, 2013.
- Mark Lutz, *Learning Python*, 5th Edition, O'Reilly Press, 2013.
- Andrew Bird, Lau Cher Han, et. al, *The Python Workshop*, Packt Publishing, 2019.
- Al Sweigart, *Automate the boring stuff with Python*, 2nd Edition, No Starch Press, 2020.





## CS324 - SKRIPTING JEZICI

### Python i Data Science

Lekcija 11

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 11

### *PYTHON I DATA SCIENCE*

- ✓ Python i Data Science
- ✓ Poglavlje 1: Uvod u Data Science
- ✓ Poglavlje 2: Data Science i Biznis inteligencija (BI)
- ✓ Poglavlje 3: Alati za Data Science
- ✓ Poglavlje 4: Python i Data Science
- ✓ Poglavlje 5: Python i osnove statistika
- ✓ Poglavlje 6: Pokazne vežbe #11
- ✓ Poglavlje 7: Individualne vežbe #11
- ✓ Poglavlje 8: Domaći zadatak #11
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ❖ Uvod

### UVOD

*Data Science predstavlja četvrtu paradigmu nauke.*

U ovoj lekciji biće reči o Python jeziku kao jednim od najpopularnijih jezika za sve popularniju oblast kako na naučnom planu tako i u privredi, a to je *Data Science*.

Prethodne lekcije obrađivale su pakete koje su obrađene uglavnom su služile za obradu i prikaz podataka. Upravo su to veštine koje su potrebne za budućeg inženjere računarske tehnike koji želi da se bavi ovom oblašću, kao *Data Analyst*, *Data Scientist*, ili *Data Engineer*.

U lekciji biće dat pregled Data Science i njenih grana, biznis inteligenciji, kao i alata za Data Science gde spada i Python.

Prema Wikipediji, Data Science predstavlja jedno interdisciplinarno naučno polje koje koristi naučne metode, procese, algoritme i sisteme da izvuče znanje iz različitih tipova uređenih i neuređenih podataka. Data Science, kao "nauka" o podacima je usko povezana sa pretragom podataka (en. **data mining**), velikim količinama podataka (en. **big data**), ali i sa oblastima veštačke inteligencije kao što je mašinsko učenje.

Može se reći da Data Science, ili nauka o podacima, predstavlja jedan koncept da se objedini analiza podataka, statistika, mašinsko učenje, i domensko znanje da bi se podaci analizirali i konačno razumeli.

Budući da koristi tehnike i teorije koje su povezane sa matematikom, statistikom, računarskom tehnikom i informacionim tehnologijama, rečeno je da Data Science predstavlja četvrtu paradigmu nauke, pored empirijske, teoretske i računarske.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 1

# Uvod u Data Science

## ŠTA JE DATA SCIENCE?

*Data science objedinjuje statistiku, matematiku, računarsku tehniku i informacione tehnologije da bi se bolje razumeli nestruktuirani podaci.*

**Data Science** predstavlja multidisciplinarno polje nauke koje se zasniva na naučnim metodama, procesima, algoritmima i sistemima da izvuče znanje iz strukturiranih i (češće) nestruktuiranih podataka.

Ovo naučno polje zapravo predstavlja koncept koji objedinjuje statistiku i obradu podataka da bi se razumeo i analizirao "*fenomen podataka*". Koristi tehnike i teorije iz nauka kao što su matematika, statistika, računarska tehniku i informacione tehnologije.

Može se reći da je **DataScience** studijski domen koji obrađuje velike količine podataka koristeći savremene tehnologije i tehnike, sa ciljem da pronađe šablon u podacima koji nisu strukturirani, iz tih podataka izvuče korisne informacije i konačno da napravi određene odluke.

Pojam **data science** ili **data-driven science** još uvek nema zvanični prevod na srpski jezik, te će se u ostatku lekcije koristiti izraz na engleskom jeziku.

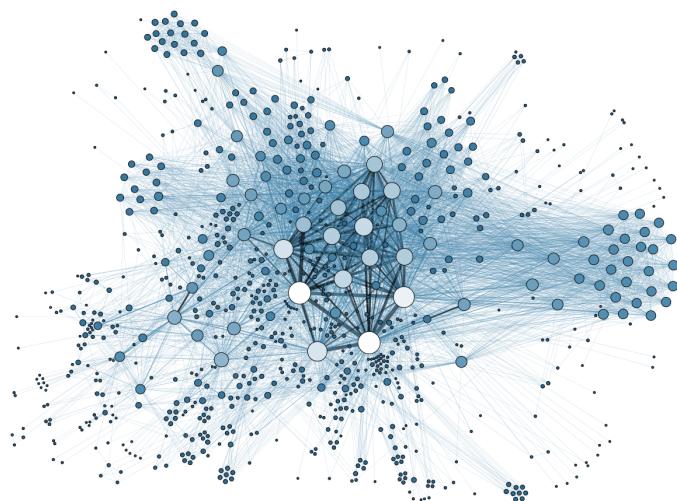
**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ČEMU SLUŽI DATA SCIENCE?

*Osnovni cilj Data Science jeste da pomogne organizacijama da prave brže i bolje odluke u svom poslovanju.*

Razne organizacije (državne, akademske, komercijalne, i dr.) primetile su da se iz podataka mogu izvući korisne informacije koje nisu odmah vidljive i na osnovu njih doneti određene zaključke i odluke u poslovanju.

Osnovni cilj **Data Science** jeste da pomogne organizacijama da prave brže i bolje odluke u svom poslovanju. Organizacije upravo zbog ove mogućnosti prelaze na tzv. **data-driven approach**, tj. pristup rešavanju (novih) problema iz (postojećih) podataka. Na ovaj način Data Science predstavlja još jedan spoj informaciono-komunikacionih tehnologija u biznis domen.



Slika 1.1 Klasifikacija podataka. [Izvor: <https://hackr.io>]

Najčešći primeri korišćenja Data Science:

- Pretraga na Internetu. Mnogi portalni za Internet pretraživanje (en. **search engines**) kao što su Google, Yahoo!, Bing, Ask, Duckduckgo, i dr. koriste algoritme **Data Science**-a da bi dobili najbolje rezultate pretraživanja za manje od sekunde. Google kakav danas poznajemo bez Data Science algoritma bio bi samo jedan u moru pretraživača.
- Plasiranje reklama na Internetu. Ceo savremeni digitalni marketing oslanja se na algoritme **Data Science**-a da bi korisnicima plasirali reklame za proizvode i usluge koje bi im sa velikom verovatnoćom bili primamljivi.
- Sistemi preporuke (en. **recommender systems**). Ovi sistemi koriste algoritme **Data Science**-a da korisnicima pronađu i ponude relevantne proizvode i usluge prema korisnikovoj istoriji. Ovi sistemi uključuju preporuke sličnih filmova, serija, muzike, knjiga, proizvoda i sl.
- Prepoznavanje slika i glasa (en. **image recognition, speech recognition**). Potreba za prepoznavanjem slike (tačnije, onog što se nalazi na slici) i izgovorenih reči postoji od ranih rana računarstva, a tek sa pojavom **Data Science** ovi algoritmi su postali dovoljno napredni da je moguće izvršiti korektno prepoznavanje.

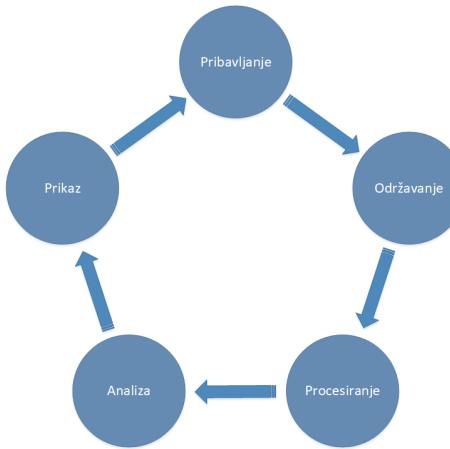
## ŽIVOTNI CIKLUS DATA SCIENCE PROJEKTA

*Svaki Data Science projekat ima pet faza životnog ciklusa: pribavljanje podataka, održavanje, procesiranje, analiza, i prikaz.*

Postoje pet faza životnog ciklusa svakog Data Science projekta:

- Pribavljanje podataka (en. **data capture**)
- Održavanje podataka (en. **data maintenance**)
- Procesiranje podataka (en. **data processing**)
- Analiza podataka (en. **data analysis**)
- Prikaz podataka (en. **data communication**)

Sva od ovih faza treba da odgovori na jedno ključno pitanje, i sastoji se od dodatnih koraka.



Slika 1.2 Ciklus Data Science projekta. [Izvor: Autor]

### Pribavljanje podataka

Ova faza treba da odgovori na pitanje "Kako se podaci pribavljaju?" i sastoji se od sledećih koraka:

- **Skupljanje podataka** (en. [data acquisition](#), [data collection](#)). Ovo je prvi korak u bilo kojem Data Science projektu. Zahtevani skup podataka (en. [dataset](#)) uglavnom se ne nalazi na jednoj lokaciji, već je distribuirana kroz različite aplikacije i sisteme organizacije (en. [line-of-business applications and systems](#)).
- **Unos podataka** (en. [data entry](#)). Podaci se takođe mogu napraviti kao novi podaci (en. [new data](#)) i mogu se uneti ručno ili korišćenjem uređaja i aplikacija.
- **Prijem signala** (en. [signal reception](#)). Još jedan izvor podataka (najčešće u projektima koji zahtevaju određene podatke iz prirode) jeste putem pametnih uređaja kao što su senzorske mreže (en. [sensor networks](#)) ili pametni uređaji kao što su [Internet of Things](#).
- **Izvlačenje podataka** (en. [data extraction](#)). Izvlačenje podataka jeste proces koji uključuje nabavku podatka iz različitih izvora kao što su serveri, baze podataka, dnevnični (en. [logs](#)) ili repozitorijumi na Internetu.

## ODRŽAVANJE PODATAKA

*Održavanje podataka odnosi se na upravo prikupljene podatke i pripremom za procesiranje tih podataka.*

### Održavanje podataka

Ova faza treba da odgovori na pitanje "Šta se dešava sa pribavljenim podacima?" i sastoji se od sledećih koraka:

- **Skladištenje podataka** (en. [data warehousing](#)). Skladištenje u ovoj fazi podaci iz različitih izvora se skladište na jednom repozitorijumu kome se može pristupiti za dalju analizu.

- **Čišćenje podataka** (en. **data cleansing**, **data cleaning**). Ovo je proces identifikovanja i odbacivanja (ili ispravljanja) netačnih unosa u skupu podataka, tabelu ili bazu podataka. Ova faza uključuje prepoznavanje i (po potrebi uklanjanje) nedovršenih, nepouzdanih, netačnih, dupliranih podataka, kao i podataka koji nedostaju ili koji sadrže nerelevantne unose. Takođe uključuje sortiranje i ponovno sortiranje podataka, ponovno modelovanje podataka.
- **Priprema podataka** (en. **data staging**). Ova faza uključuje privremeno skladištenje podataka pre predaje na pre-procesiranje.
- **Pre-procesiranje** (en. **data preprocessing**). Ova faza uključuje pre-procesiranje podataka za interpretaciju. Najčešće se ovde koriste algoritmi mašinskog učenja; međutim, primjenjeni algoritam pre-procesiranja zavisiće od izvora podataka i svrhu za koju će se podaci koristiti.
- **Arhitektura podataka** (en. **data architecture**). Ovaj korak zapravo predstavlja radni okvir (en. **framework**) za efikasan prenos podataka sa jedne lokacije na drugu. Sadrži modele i pravila o tome koji se podaci sakupljaju. Takođe kontroliše kako se prikupljeni podaci skladište, uređuju i integrišu, i smeštaju u sistem podataka jedne organizacije. Može se reći da arhitektura podataka postavlja standarde za sisteme podataka kao vizija ili model interakcije sistema podataka.

## PROCESIRANJE I ANALIZA PODATAKA

*Procesiranje podataka daje načine pretrage, modelovanja i opisa podataka. Podaci se mogu analizirati na više načina, u zavisnosti od cilja organizacije.*

### Procesiranje podataka

Ova faza treba da odgovori na pitanje "Šta treba uraditi sa očišćenim podacima?" i sastoji se od sledećih koraka:

- **Pretraga podataka** (en. **data mining**). U ovoj fazi pronalaze se smerovi (en. **trends**) skupova podataka. Ovi smerovi se koriste da se identifikuju budući šabloni. Ova faza često uključuje analizu velike količine istorijskih podataka (en. **historical data**) koji su prethodno odbačeni.
- **Klasifikacija podataka** (en. **data clustering**, **data classification**). Klasifikacija jeste zadatak da se odvoje ili klasifikuju podaci u veći određeni broj grupa na takav način da su podaci unutar iste grupe međusobno sličniji u odnosu na podatke drugih grupa. Cilj ove faze jeste podjela podataka u grupe sa sličnim osobinama.
- **Modelovanje podataka** (en. **data modeling**). Ova faza predstavlja proces realizacije opisnog dijagrama odnosa između različitih tipova podataka koji se obično smeštaju u bazu podataka
- **Sumiranje podataka** (en. **data summarization**). Ova faza uključuje tehnike pronalaska kompaktnog opisa skupa podataka i predstavlja ključni deo pretrage podataka. Zapravo, ova faza predstavlja kratak zaključak posle analize velikog skupa podataka.

### Analiza podataka

Ova faza treba da odgovori na pitanje "Kako analizirati podatke?" i sastoji se od sledećih koraka:

- **Istraživanje/potvrđna analiza** (en. *exploratory/confirmatory analysis*). Ova vrsta analize vrši se uporedno - istraživanje predstavlja proces prikupljanja "dokaza" dok potvrđna analiza procenjuje i potvrđuje ili odbacuje dokaze.
- **Prediktivna analiza** (en. *predictive analysis*). Ova vrsta analize predstavlja korišćenje analitike podataka (en. *data analytics*) za pravljenje predikcija na osnovu podataka. Ovaj proces koristi statistiku podataka i mašinsko učenje da bi se dobili odgovori na buduće događaje.
- **Regresija** (en. *regression*). Ova vrsta analize jeste oblik prediktivne analize koja istražuje odnos između nezavisne promenljive (prediktorske promenljive) i zavisne promenljive (ciljane promenljive). Ova tehnika se koristi za predviđanje događaja u vremenu, kao i nalaženje odnosa između promenljivih u skupu.
- **Pretraga teksta** (en. *text mining*). Ova vrsta analize izvlači šablonе iz nestruktuiranog teksta. Informacija i odnosi podataka jesu skriveni u tekstu i nisu uvek vidljivi kao kod pretrage podataka.

## ANALIZA PODATAKA I PRIKAZ PODATAKA

*Prikaz podataka jeste poslednji, ali najčešće najbitniji deo ciklusa Data Science projekta, jer se na osnovu prikaza donose odluke.*

### Analiza podataka

Kada podaci nisu u numeričkom obliku teže ih je razumeti. Zbog toga se koristi termin kvalitativni podaci (en. *qualitative data*) koji definiše podatke koji su nekako okarakterisani i aproksimovani. Kvalitativni podaci se mogu posmatrati i sačuvati, i ovi podaci su uglavnom po prirodi ne-numerički. Prikupljaju se posmatranjem, intervjuima, kroz fokus grupe i slične metode.

- **Kvalitativna analiza** (en. *qualitative analysis*). Kvalitativna analiza predstavlja proces pregleda kvalitativnih podataka da bi se izveo zaključak o nekoj pojavi. Kvalitativna analiza pruža razumevanje kroz otkrivanje šablonе i tema (en. *data patterns and themes*) samih podataka.

### Prikaz podataka

Ova faza treba da odgovori na pitanje "Kako prikazati rezultate?" i sastoji se od sledećih koraka:

- **Izveštavanje** (en. *data reporting*). Izveštaji pružaju informaciju o rezultatima izvršenog istraživanja i analize podataka, kao i o mogućim poteškoćama. Tipovi izveštaja jesu mnogobrojni, i mogu pokriti mnoge teme, ali obično se ovakvi izveštaji fokusiraju na pružanje informacija sa tačnim ciljem, konkretnoj publici. Dobri izveštaji su dokumenti koji su tačni, objektivni, i kompletni.
- **Vizuelizacija podataka** (en. *data visualization*). Sa vizuelizacijom podataka moguće je izvršiti grafički prikaz podataka i informacija. Korišćenjem grafikona, mapa i drugih alata,

moguće je predstaviti podatke, smerove tih podataka, šablone i izuzetke podataka na razumljiv način.

- **Biznis inteligencija** (en. **business intelligence**). Neotuđivi deo Data Science jeste biznis inteligencija, i može se tretirati kao podskup Data Science-a. Više u biznis inteligenciji u posebnom poglavlju.
- **Odlučivanje** (en. **decision making**). Značaj podataka jeste taj da se na osnovu njih doneše odluka koja će poboljšati neki aspekt poslovanja unutar organizacije.

## ▼ Poglavlje 2

# Data Science i Biznis inteligencija (BI)

## ŠTA JE BIZNIS INTELGINCIJA?

*Proces analiziranja i izveštavanja o istorijskim biznis podacima sa ciljem donošenja boljih strateških odluka predstavlja biznis inteligenciju.*

Biznis inteligencija (en. **business intelligence**) predstavlja proces analiziranja i izveštavanja o istorijskim biznis podacima sa ciljem donošenja strateških biznis odluka.

Biznis podaci (en. **business data**) predstavlja informaciju koja se koristi za planiranje i rad organizacije. Uključuje podatke u izvornom obliku (en. **source data**) koje organizacija sakuplja, kao i podatke koji su rezultat obrade tih podataka.

***Cilj biznis inteligencije jeste da objasni događaje u prošlosti koristeći biznis podatke.***

Biznis inteligencija je podskup Data Science, jer predstavlja korak koji prethodi prediktivnoj analizi. Ipak, treba razlikovati ova dva pojma:

- Data Science je okrenuta najviše ka budućim podacima i budućim događajima.
- Biznis inteligencija je okrenuta ka istorijskim podacima i prošlim događajima.

Istorijski gledano, biznis inteligencija se pojavila 60-tih godina 20. veka kao sistem deljenja informacija između organizacija. Zajedno sa razvojem računarske tehnike, 80-tih godina 20. veka biznis inteligencija je evoluirala u modele za donošenje odluka na osnovu istorijskih podataka.

Danas, savremena rešenja baziraju se na istorijskim biznis podacima, softverom za prikupljanje, obradu u prikaz podataka, ponovo sa ciljem donošenja boljih odluka unutar organizacije za poboljšanje performansi same organizacije.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PROCESI BIZNIS INTELIGENCIJE

*Biznis inteligencija obuhvata procese i metode za prikupljanje, skladištenje i analizu biznis podataka.*

Biznis inteligencija ne predstavlja eksplizitno jedan pojam, već obuhvata procese i metode za prikupljanje, skladištenje i analizu podataka unutar biznis okruženja ili aktivnosti, sa ciljem poboljšanje performansi. Ovi procesi omogućavaju da se napravi bolji pogled na poslovanje same organizacije da bi se donele bolje odluke.

Procesi biznis inteligencije obuhvataju neke od procesa iz životnog ciklusa svakog Data Science projekta, sa naznakom da se radi o biznis podacima:

- Pretraga podataka,
- Izveštavanje,
- Metrike performansi (en. **benchmarking**),
- Deskriptivna analiza,
- Upitnici,
- Statistička analiza,
- Vizuelizacija podataka,
- Vizuelna analiza,
- Priprema podataka.

Metrike performansi predstavlja proces poređenja trenutnih performansi organizacije sa istorijskim podacima, sa zadatim određenim ciljem.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ZNAČAJ BIZNIS INTELIGENCIJE

*Biznis inteligencija pomaže organizacijama da prave bolje odluke pokazujući istorijske (nekad i trenutne) podatke u biznis kontekstu.*

Biznis inteligencija pomaže organizacijama da prave bolje odluke pokazujući istorijske (nekad i trenutne) podatke u biznis kontekstu. Analitičari mogu iskoristiti biznis inteligenciju da pružaju performanse organizacije, najčešće u poređenju sa sličnim organizacijama, da bi matična organizacija radila brže i efikasnije. Analitičari takođe mogu lakše uočiti smerove tržišta da bi povećali prodaje ili profit.

Ukoliko se pravilno koristi, pravi podaci mogu pomoći kod svih aspekata poslovanja organizacije.

Biznis inteligencija može pomoći organizacijama na sledeće načine:

- Identifikovati načine za povećanje profita,
- Analiza ponašanje kupaca,

- Poređenje sa konkurencijom,
- Praćenje performansi,
- Optimizacija operacija,
- Predikcija uspešnosti,
- Hvatanje trendova na tržištu,
- Pronalazak problema.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## KAKO FUNKCIONIŠE BIZNIS INTELIGENCIJA I ANALITIKA?

*Da bi se odgovorilo na pitanja i pratila uspešnost zadatih ciljeva, prikuplja neophodne podatke, analiziraju te podatke, i na osnovu te analize donose odluke o poslovanju.*

Sve organizacije, od malih i srednjih preduzeća, do međunarodnih korporacija, postavljaju svoje ciljeve, kao i pitanja u poslovanju. Da bi se odgovorila pitanja i pratila uspešnost zadatih ciljeva, prikupljaju neophodne podatke, analiziraju te podatke, i na osnovu te analize određuju da li treba i koje treba odluke doneti da bi se ti ciljevi u poslovanju postigli.

Sa tehničke strane, aktivnost organizacije se prikuplja kao sirovi podaci (en. **raw data**). Jednom skladišteni, korisnici mogu pristupiti ovim podacima, i na taj način pokrenuti proces analize u cilju odgovora na pitanja poslovanja.

**Analitika podataka** (en. **data analytics**) i **biznis analitika** (en. **business analytics**) jesu deo biznis inteligencije, ali biznis inteligencija koristi samo za deo svog procesa. Biznis inteligencija pomaže korisnicima da donesu zaključke iz analize podataka.

Za razliku od Data Science analitičara koji se brinu o tome zašto se neki događaj desio i šta se sledeće može dogoditi, u biznis analitici bitno je pretvoriti modele i algoritme na konkretnе aktivnosti.

Biznis analitika nije linearan proces iz razloga što odgovorom na jedno pitanje javljaju se druga pitanja, što dovodi do iterativnog procesa. Ovaj proces je iz tog razloga kružni proces, koji uključuje pristup, pronalazak, istraživanje i deljenje informacija. Ovo predstavlja tzv. ciklus analitike (en. **cycle of analytics**), termin koji objašnjava na koji način organizacije koriste analitiku da reakciju na pitanja i očekivanja koja se kontinualno menjaju.

## ✓ Poglavlje 3

### Alati za Data Science

#### POTREBNE VEŠTINE ZA DATA SCIENCE

*Data Science zahteva skup veština iz matematike, programiranja, statistike, i veštačke inteligencije.*

Data Science je multidisciplinarna oblast i zahteva znanje iz oblasti programiranja, matematike, statistike, ali i sposobnost predstavljanja kompleksnih pojmove na jednostavna način.

U zavisnosti od željene karijere, veštine i alati koji su potrebni zavisiće od konkretnе pozicije, ali se mogu svrstati u sledeće kategorije:

- **Programiranje.** Bilo koja od pozicija iz Data Science zahteva znanje nekog programskog jezika koji ima mogućnost statističke obrade podataka. Najpopularniji su Python i R, ali i upitni jezici kao što je SQL.
- **Statistika.** U zavisnosti od pozicije, potrebno je osnovno ili napredno znanje iz statistike.
- **Mašinsko učenje.** Svaka organizacija koja radi sa velikom količinom podataka zahtevaće znanje iz algoritama nadgledanog i nenadgledanog mašinskog učenja.
- **Linearna algebra.** Napredne pozicije zahtevaju pravljenje modela sa podacima, najčešće kroz višedimenzionalne nizove, te je potrebno znanje iz linearne algebre.
- **Vizuelizacija i predstavljanje podataka.** Možda najbitnija stavka sa ove liste jeste mogućnost prikaza podataka kroz izveštaje i grafikone, na što jednostavniji način.
- **Softversko inženjerstvo.** Za pozicije koje zahtevaju rad sa podacima (organizacija skladišta i pristupa), potrebno je znanje iz softverskog inženjerstva.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

#### ANALITIČAR PODATAKA (DATA ANALYST)

*Analitičar podataka smatra se početnom pozicijom iz oblasti Data Science.*

Analitičar podataka (en.**Data Analyst**) smatra se početnim pozicijom iz oblasti Data Science. Glavni zadatak analitičara jeste da pregleda podatke organizacije i da iz tih podataka izvuče

odgovore na pitanja iz poslovanja. Zatim, treba svoje odgovore da predstavi ostatku tima unutar organizacije.

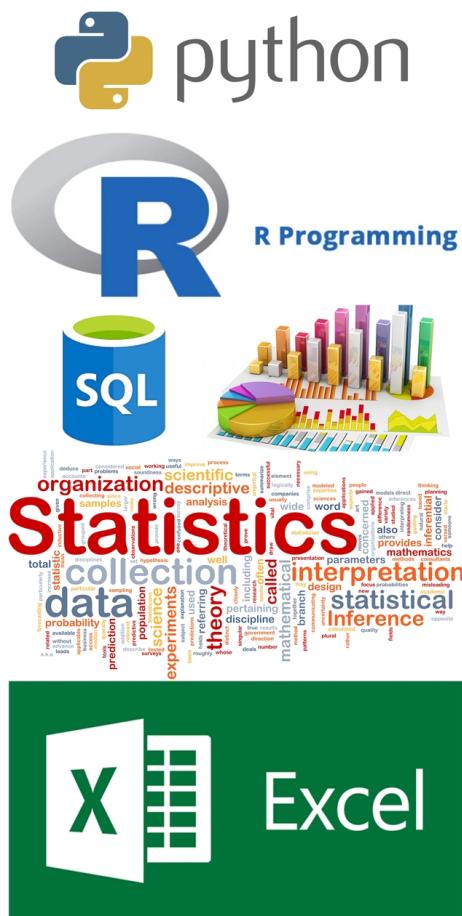
Ovaj posao obuhvata pristup podacima, čišćenje podataka, statističku analizu nad podacima, i vizuelizaciju podataka.

**Primer:** Od analitičara se traži da pogleda podatke o prodaji u toku jednog vremenskog perioda i da pronađe jake i slabe strane u toku tog vremenskog perioda.

Analitičar može da bude član različitih timova unutar organizacije, ali pitanja na koja treba dati odgovore u najvećem broju slučajeva odnose se na istorijske podatke.

## Potrebne veštine za analitičara:

- Znanje programskog jezika Python ili R, sa odgovarajućim paketima,
  - Znanje SQL jezika,
  - Rad sa podacima (pravilan pristup, čišćenje, vizuelizacija),
  - Statistika i verovatnoća,
  - Prikaz analize podataka publici koja nema iskustvu u programiranju i statistici,
  - Tableau/Power BI,
  - SAS/SPSS,
  - Excel



Slika 3.1 Potrebne veštine za analitičara podataka. [Izvor: stock slike]

## NAUČNIK PODATAKA (DATA SCIENTIST)

*Naučnik podataka treba da zna sve veštine analitičara, ali i modele mašinskog učenja.*

**Naučnik podataka** (en. **Data Scientist**) predstavlja jaču poziciju od analitičara. Mnoge od aktivnosti koje radi analitičar treba i naučnik da zna; međutim, fokus naučnika jeste pravljenje modela i smernica za budući tok poslovanja, ne samo pregled istorijskih podataka.

**Primer:** Od naučnika se traži da istrenira modele za mašinsko učenje koji će na osnovu istorijskih podataka, dati bolji uvid i procene budućih podataka.

Ovaj posao, pored svih aktivnosti analitičara, obuhvata i veštačku inteligenciju, najčešće mašinsko učenje. Naučnik podataka ima više slobode u odnosu na analitičara da izrazi svoje ideje i eksperimente nad podacima da bi našao šablove i smerove (trendove) u podacima.

Potrebne veštine za naučnika uključuju sve veštine analitičara, ali i:

- Razumevanje algoritama nadgledanog i nenadgledanog mašinskog učenja,
- Znanje statistike i mogućnost procene statističkih modela,
- Napredno znanje programskih jezika R i Python, kao i alata kao što je Apache Spark
- Napredno znanje u SAS/SPSS
- Napredno znanje NPL-a
- Napredno znanje Tableau/Power BI



Slika 3.2 Potrebne veštine za naučnika podataka. [Izvor: stock slike]

## INŽENJER PODATAKA (DATA ENGINEER)

*Inženjer podataka je zadužen za održavanje infrastrukture podataka jedne organizacije.*

Inženjer podataka (en. **Data Engineer**) je zadužen za održavanje infrastrukture podataka jedne organizacije. Ova pozicija zahteva manje znanja iz statistike, ali više znanje iz softverskog inženjerstva i programiranja. Inženjer podataka unutar organizacije može biti zadužen za pravljenje kanala podataka (en. **data pipeline**) da bi se dobili podaci u prodaji, marketingu i prihodima da analitičari i naučnici mogu lako pristupiti tim podacima i u odgovarajućem formatu.

Posao inženjera podataka takođe obuhvata kreaciju i održavanje infrastrukture za skladištenje i brz pristup podacima.

Veštine koje se zahtevaju od inženjera podataka jesu više orijentisane ka softverskom inženjerstvu i razvoju softvera:

- Napredno programersko znanje u Python jeziku,
- Napredno programersko znanje u jezicima za rad sa velikim skupovima podataka i za kreaciju kanala podataka (Apache Hadoop, Apache Spark),

- Napredno znanje SQL jezika
- Postgres
- AWS/Azure



Slika 3.3 Potrebne veštine za inženjera podataka. [Izvor: stock slike]

## ▼ Poglavlje 4

# Python i Data Science

## ZAŠTO PYTHON ZA DATA SCIENCE?

*Python je postao programski jezik za naučno programiranje zahvaljujući eksternim paketima.*

Python kao programski jezik jednostavne sintakse poslednje decenije postao je prvi izbor kao programski jezik za naučno programiranje, koji svojim modulima proširuje svoju funkcionalnost.

Danas su najpopularniji paketi upravo oni koji su neophodni za Data Science pozicije:

- Višedimenzionalni nizovi (**numpy**),
- Vizuelizacija podataka (**matplotlib**)
- Analiza tabelarnih podataka (**pandas**),

Python je prvo bitno zamišljen kao programski jezik koji nije orijentisan ka ovakvim zadacima, ali je vremenom postao vodeći programski jezik (pored jezika R) za Data Science.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## OSOBINE PYTHON PROGRAMSKOG JEZIKA POGODNE ZA DATA SCIENCE.

*Programski jezik Python je otvorenog koda i svi paketi za Data Science su besplatni.*

Programski jezik Python poseduje nekoliko značajnih osobina za Data Science projekte:

- **Python je fleksibilan jezik**

Python je programski jezik opšteg tipa i omogućuje razvijanja različitih aplikacija, od veb sajtova, do naučnog programiranja, rad sa bazama podataka, kratkih skripti, ali i ima mnogo drugih primena.

- **Lak je za učenje**

Python jezik osmišljen je da se fokusira na jednostavnost i čitljivost napisanog koda. Python na taj način je pogodan programski jezik za početnike, ali i za napredne korisnike.

- **Otvorenog je koda**

Python jezik se razvija od strane svoje zajednice programera (en. **community**). Razvijen je da radi na različitim platformama (Windows/MacOS/Linux). Takođe, svi paketi koji se mogu dodatno uvesti u Python jesu besplatni. Mnogi od tih paketa (koji se posebno naplaćuju u jezicima kao što je MATLAB) jesu jako pogodni za alate Data Science-a, kao što je vizuelizacija podataka, statistika, mašinsko učenje, i mnogi drugi.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PYTHON NASPRAM JEZIKA R ZA DATA SCIENCE

*Dva vodeća programska jezika za Data Science imaju svoje prednosti i mane.*

Programski jezik R je razvijen od strane statističara sa glavnom namenom statističke analize podataka. Jezik R poseduje podršku za vizuelizaciju podataka i dodatnu analizu velike količine podataka.

Poređenje programskih jezika R i Python za Data Science:

### Brzina

Do 1000 iteracija, Python brže izvršava kod. Nakon 1000 iteracija, jezik R je brži jer koristi funkciju koja je više optimizovana za rad sa velikim brojem iteracija.

### Sintaksa

Sintaksa R programskega jezika je detaljnija od Python jezika. Primer: operacija dodele vrednosti koristi znak manje ili veće (<- ili ->), da bi označila smer dodele vrednosti, dok Python koristi operator jednako ( = )

### Podrška za veštačku inteligenciju

Python ima bolju podršku za veštačku inteligenciju kroz pakete i kroz radne okvire, dok R ima manje funkcionalnosti.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 5

# Python i osnove statistika

## UVOD U STATISTIKU U PYTHON PROGRAMSKOM JEZIKU

*Pri radu sa velikom količinom podataka, podaci se mogu opisati statističkim veličinama i na taj način izvući zaključke o samim podacima.*

U ovom delu opisaće se odnosni statistički pojmovi i način izračunavanja u Python programskom jeziku.

Pri radu sa velikom količinom podataka, podaci se mogu opisati statističkim veličinama i na taj način izvući zaključke o samim podacima.

Za potrebe ovog predmeta, potrebno je znati sledeće:

- Srednja vrednost (aritmetička srednja vrednost),
- Medijana,
- Modus,
- Varijansa,
- Standardna devijacija.

Pri radu sa velikom količinom podataka, podaci se mogu opisati statističkim veličinama i na taj način izvući zaključke o samim podacima.

Sve statističke veličine se u Python programskom jeziku mogu naći u **numpy** ili **scipy** paketu.

```
# uvoz potrebnih paketa i modula
import numpy as np
from scipy import stats
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## SREDNJA VREDNOST

*Srednja vrednost niza predstavlja aritmetičku sredinu tog niza.*

Aritmetička srednja vrednost, ili samo srednja vrednost, nazvana i očekivana vrednost (en. **mean**), predstavlja centralnu vrednost diskretnog skupa brojeva.

<https://en.wikipedia.org/wiki/Mean>

Osim aritmetičke sredine, postoji geometrijska i harmonijska sredina.

Aritmetička sredina predstavlja sumu svih vrednosti podeljenu sa ukupnim brojem vrednosti:

$$\bar{x} = \frac{1}{n} \left( \sum_{i=1}^n x_i \right) = \frac{x_1 + x_2 + \cdots + x_{n-1} + x_n}{n}$$

U Python programskom jeziku, ukoliko imamo numpy niz, možemo naći aritmetičku sredinu pozivom funkcije **mean()**

```
mean(a, axis=None, dtype=None)
```

Parametar **a** označava numpy niz koji se ubacuje.

Parametar **axis** označava po kojoj osi (dimenziji) se traži srednja vrednost.

Parametar **dtype** odnosi se na tip podataka.

### Primer (3 minuta):

Napisati program koji će računati aritmetičku srednju vrednost niza a, koji je sačinjen od parnih brojeva od 0 do 1000.

```
import numpy as np

x = np.arange(0,1000,2)
print(np.mean(x))
```

```
File Edit Selection View Go Run Terminal Help CS324_L11_stats.py - D2 - Visual Studio Code
OPEN EDITORS 1 UNSAVED
d2
CS324_L11_stats.py
data_get.txt
data.csv
d2_populate.py
prog_data_2.csv

import numpy as np
x = np.arange(0,1000,2)
print(x)
print(np.mean(x))

500.0
```

Slika 5.1 Aritmetička sredina. [Izvor: Autor]

## MEDIJANA I MODUS

*Medijana razdvaja gornju i donju polovinu uzoraka niza. Modus je vrednost koja se najčešće pojavljuje.*

Medijana (en. **median**) se u teoriji verovatnoće i statistici opisuje kao broj koji razdvaja gornju polovinu uzorka, populacije ili raspodele verovatnoće od donje polovine. Medijana konačnog niza brojeva se može naći tako što se brojevi poređaju po veličini, i uzme se srednji član niza. Ukoliko postoji paran broj članova niza, medijana nije jedinstvena, pa se često uzima aritmetička sredina dve vrednosti koje su kandidati za medijanu.

Za neparan broj elemenata:

$$\text{median} = x \left( \frac{n+1}{2} \right)$$

Za paran broj elemenata:

$$\text{median} = \frac{x \left( \frac{n}{2} \right) + x \left( \frac{n}{2} + 1 \right)}{2}$$

Modus (en. **mode**) je vrednost koja se u uzorku ili grupi podataka pojavljuje najčešće.

Medijana se dobija pozivom funkcija **median()**:

```
median(a, axis=None)
```

Modus se dobija pozivom funkcije **mode()**, iz **stats** modula paketa **scipy**

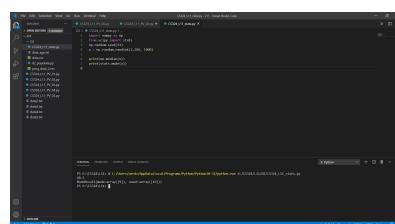
```
mode(a, axis=None)
```

**Primer (3 minuta):**

Napisati program koji će računati medijanu i modus niza a, koji je sačinjen nasumičnih celih brojeva od 0 do 100, dužine 1000, i seed-om 11

```
import numpy as np
from scipy import stats
np.random.seed(11)
x = np.random.randint(1,100, 1000)

print(np.median(x))
print(stats.mode(x))
```



Slika 5.2 Medijana i modus. [Izvor: Autor]

# VARIJANSA I STANDARDNA DEVIJACIJA

*Varijansa predstavlja odstupanje od srednje vrednosti. Standardna devijacija je kvadratni koren varijanse.*

Varijansa (en. **variance**) je pojam iz teorije verovatnoće i statistike i predstavlja matematičko očekivanje odstupanja slučajne promenljive od njene srednje vrednosti.

$$Var(x) = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$
$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

Standardna devijacija (en. **standard deviation**) jeste vrednost koja je jednaka kvadratnom koren varijanse.

U Python programskom jeziku, varijansa se poziva funkcijom **var()**, a standardna devijacija se poziva funkcijom **std()**. Obe funkcije se nalaze u **numpy** paketu.

```
var(a, axis=None)
```

```
std(a, axis=None)
```

Kao i kod prethodnih funkcija, parametar **a** označava **numpy** niz koji se prosleđuje, a **axis** označava po kojoj osi se računa funkcija.

## Primer (5 minuta):

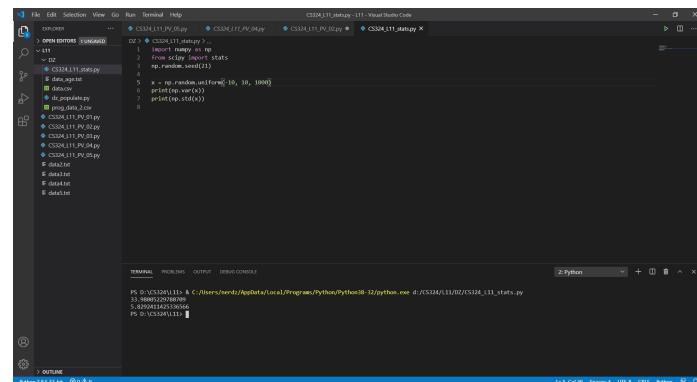
Napisati program koji će naći varijansu i standardnu devijaciju numpy niza od 1000 elemenata (seed 21), sa vrednostima od -10 do 10. Elementi su razlomljeni brojevi uniformne raspodele.

```
import numpy as np

np.random.seed(21)

x = np.random.uniform(-10, 10, 1000)

print(np.var(x))
print(np.std(x))
```



Slika 5.3 Varijansa i standardna devijacija. [Izvor: Autor]

## ✓ Poglavlje 6

### Pokazne vežbe #11

## UVOD U POKAZNE VEŽBE #11

*Pokazne vežbe #11 dopunjuju znanje o statističkim metodama pri radu sa podacima*

Pokazne vežbe #11 dopunjuju znanje o statističkim metodama pri radu sa podacima koji su najčešće predstavljeni nizovima.

Statističke veličine koje se predstavljaju jesu:

- Srednja vrednost,
- Medijana,
- Modus,
- Varijansa,
- Standardna devijacija

Pored ovih veličina, urađen je i zadatak koji predstavlja vizuelizaciju podataka kroz histogram.

## SREDNJA VREDNOST

*Srednja vrednost predstavlja sumu svih elemenata podeljena sa brojem elemenata.*

### Zadatak #1 (7 minuta)

Napisati funkciju koja vraća srednju vrednost za proizvoljan broj unetih brojeva. Isprobati kada se kao argument ubaci lista.

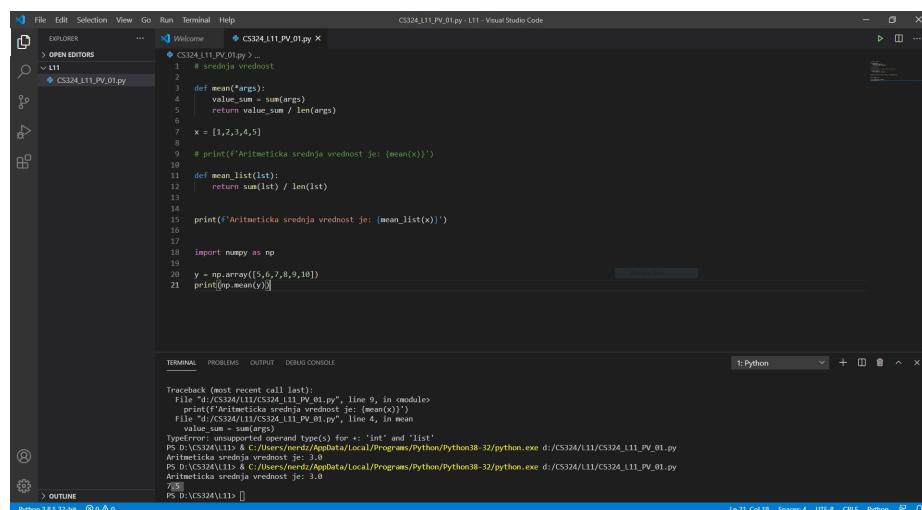
Napisati program koji računa srednju vrednost elemenata liste i numpy nizova.

```
# srednja vrednost

# ako se rucno unose elementi
def mean(*args):
    value_sum = sum(args)
    return value_sum / len(args)

print(mean(1,2,3,4,5))
```

```
# print(mean([1,2,3,4,5]))\n\ndef mean_list(lst):\n    return sum(lst) / len(lst)\n\nprint(mean_list([1,2,3,4,5]))\n\n\nimport numpy as np\n\nx = np.array([1,2,3,4,5])\nprint(np.mean(x))
```



Slika 6.1 Aritmetička srednja vrednost. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## MEDIJANA

*Medijana se u teoriji verovatnoće i statistici opisuje kao broj koji razdvaja gornju polovinu uzorka, populacije ili raspodele verovatnoće od donje polovine.*

### Zadatak #2 (8 minuta)

Napisati program koji računa medijanu za proizvoljni broj elemenata koji se sortirano unose ručno. Napisati program koji računa medijanu za listu brojeva koja ne mora biti sortirana, kao i za NumPy nizove.

```
# medijana\n\n# kada ubacujemo sortirane elemente\ndef median(*args):
```

```

if len(args) % 2 == 0:
    i = round((len(args)+1) / 2)
    j = i - 1
    return (args[i] + args[j]) / 2
else:
    k = round(len(args) / 2)
    return args[k]

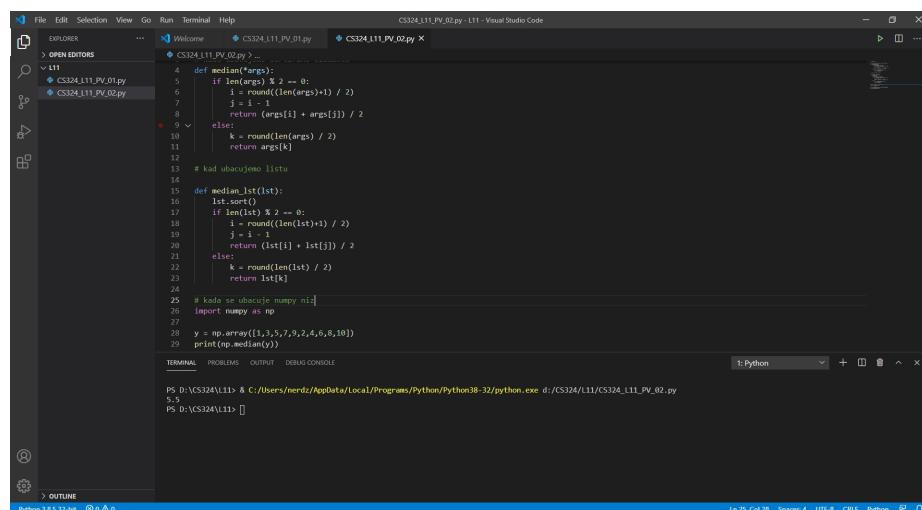
# kad ubacujemo listu

def median_lst(lst):
    lst.sort()
    if len(lst) % 2 == 0:
        i = round((len(lst)+1) / 2)
        j = i - 1
        return (lst[i] + lst[j]) / 2
    else:
        k = round(len(lst) / 2)
        return lst[k]

# kada se ubacuje numpy niz
import numpy as np

y = np.array([1,3,5,7,9,2,4,6,8,10])
print(np.median(y))

```



Slika 6.2 Medijana. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## MODUS

*Modus je vrednost koja se u uzorku ili grupi podataka pojavljuje najčešće.*

### Zadatak #3 (10 minuta)

Napisati program koji računa varijansu za proizvoljni broj elemenata koji se ručno unose.  
Napisati program koji računa varijansu za listu brojeva, kao i za NumPy nizove.

```
# modus

def mode(*args):
    dict_values = {}
    for i in args:
        dict_values[i] = args.count(i)
    max_list = []
    for k, v in dict_values.items():
        if v == max(dict_values.values()):
            max_list.append(k)
    return max_list

print(f'Modus je: {mode(1,2,2,1,3,3,3,3)}')

# sa listama

def mode_list(lst):
    dict_values = {}
    for i in lst:
        dict_values[i] = lst.count(i)
    max_list = []
    for k, v in dict_values.items():
        if v == max(dict_values.values()):
            max_list.append(k)
    return max_list

print(f'Modus je: {mode_list([1,2,2,1,3,3,3,3])}')

# sa numpy/scipy
import numpy as np
from scipy import stats

y = np.array([1,2,2,3,3,3,3])
print(stats.mode(y))
```

```

File Edit Selection View Go Run Terminal Help CS324_L11_PV_03.py - L11 - Visual Studio Code

OPEN EDITORS
> L11
  CS324_L11_PV_01.py
  CS324_L11_PV_02.py
  CS324_L11_PV_03.py X
  CS324_L11_PV_04.py

14
15 # sa listama
16
17 def mode_list(lst):
18     dict_values = {}
19     for i in lst:
20         dict_values[i] = lst.count(i)
21     max_list = []
22     for k, v in dict_values.items():
23         if v == max(dict_values.values()):
24             max_list.append(k)
25     return max_list
26
27 print("Modus je: mode_list([1,2,1,3,3,3,3])")
28
29 # sa numpy/scipy
30 import numpy as np
31 from scipy import stats
32
33 y = np.array([1,2,2,1,3,3,3])
34 print(stats.mode(y))
35

```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

PS D:\CS324\l11> & C:/Users/nerdz/appData/Local/Programs/Python/Python38-32/python.exe d:/CS324/l11/CS324\_L11\_PV\_03.py

Modus je: [2]  
Modus je: [3]  
Modus je: [3]

PS D:\CS324\l11>

Slika 6.3 Modus niza. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## VARIJANSA I STANDARDNA DEVIJACIJA

*Varijansa predstavlja matematičko očekivanje odstupanja slučajne promenljive od njene srednje vrednosti.*

### Zadatak #4 (12 minuta)

Napisati program koji računa varijansu i standardnu devijaciju za proizvoljni broj elemenata koji se ručno unose. Napisati program koji računa varijansu za listu brojeva, kao i za NumPy nizove.

```

# varijansa i standardna devijacija
import math

def mean(*args):
    value_sum = sum(args)
    return value_sum / len(args)

def mean_list(lst):
    return sum(lst) / len(lst)

# kada ubacujemo vrednosti direktno

def variance(*args):
    mean_value = mean(*args)
    delilac = 0
    for i in args:
        delilac += (i - mean_value) ** 2
    imenilac = len(args)

```

```

return delilac / imenilac

def standard_deviation(*args):
    return math.sqrt(variance(*args))

# kada ubacujemo liste

def variance_list(lst):
    mean_value = mean_list(lst)
    delilac = 0
    for i in lst:
        delilac += (i - mean_value) ** 2
    imenilac = len(lst)
    return delilac / imenilac

def standard_deviation_list(lst):
    return math.sqrt(variance_list(lst))

# numpy
import numpy as np
y = np.array([1,2,3,4,5])

```

```

File Edit Selection View Go Run Terminal Help CS324_L11_PV_04.py - L11 - Visual Studio Code
OPEN EDITORS CS324_L11_PV_01.py CS324_L11_PV_02.py CS324_L11_PV_03.py CS324_L11_PV_04.py
L11 CS324_L11_PV_01.py CS324_L11_PV_02.py CS324_L11_PV_03.py CS324_L11_PV_04.py
CS324_L11_PV_01.py CS324_L11_PV_02.py CS324_L11_PV_03.py CS324_L11_PV_04.py
# kada ubacujemo liste
def variance_list(lst):
    mean_value = mean_list(lst)
    delilac = 0
    for i in lst:
        delilac += (i - mean_value) ** 2
    imenilac = len(lst)
    return delilac / imenilac
def standard_deviation_list(lst):
    return math.sqrt(variance_list(lst))
# numpy
import numpy as np
y = np.array([1,2,3,4,5])
print(np.var(y))
print(np.std(y))

```

TERMINAL: Python

```

PS D:\CS324\l11> & C:/Users/nerdz/AppData/Local/Programs/Python/Python38-32/python.exe d:/CS324/l11/CS324_L11_PV_04.py
Varijansa je 2.0
Standardna devijacija je 1.4142135623738951
Z.A.
1.4142135623738951
PS D:\CS324\l11>

```

Slika 6.4 Varijansa i standardna devijacija. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## HISTOGRAM

*Histogram je vrsta grafikona koji pomaže da se brzo uoči tip raspodele za uzorke koji sadrže veliki broj podataka.*

Histogram je vrsta grafikona koji pomaže da se brzo uoči tip raspodele za uzorce koji sadrže veliki broj podataka. Na osnovu izgleda histograma donose se zaključci o statističkoj prirodi podataka.

### Zadatak #5 (8 minuta)

Nacrtati histogram za sledeće podatke:

```
ages = [18, 19, 21, 25, 26, 26, 30, 32, 38, 45, 55]
bins = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

Zatim, naći srednju vrednost za godine i nacrtati na histogramu.

#### Rešenje:

```
# histogram
import numpy as np
from matplotlib import pyplot as plt

ages = [18, 19, 21, 25, 26, 26, 30, 32, 38, 45, 55]

bins = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

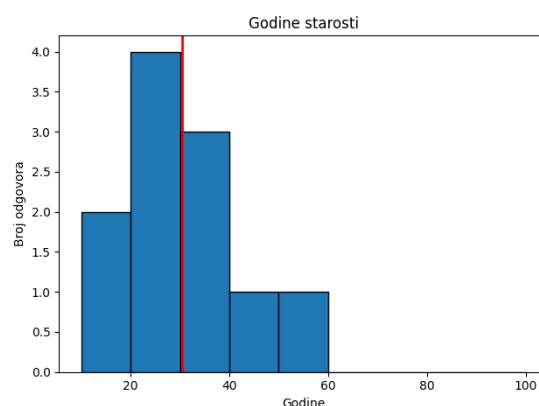
mean_age = np.mean(np.array(ages))

plt.axvline(mean_age, color='red', label='Srednja vrednost godina starosti',
linewidth=2)

plt.hist(ages, bins=bins, edgecolor='black')

plt.title('Godine starosti')
plt.xlabel('Godine')
plt.ylabel('Broj odgovora')

plt.tight_layout()
plt.show()
```



Slika 6.5 Histogram. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 7

### Individualne vežbe #11

#### ZADATAK #1

*Zadatak #1 se okvirno radi 45 minuta*

##### **Zadatak #1 (45 minuta)**

Date podatke ručno kopirati u tekstualnu datoteku i sačuvati kao CSV datoteku. Zatim, napisati program koji učitava CSV datoteku sa podacima, i iz datih podatka računa sledeće:

- Srednju vrednost godina starosti,
- Najčešću vrednost za godine starosti,
- Standardnu varijaciju i varijansu godina starosti.
- Broj ljudi čija je godina starosti u opsegu
  - 1. do 20 godina,
  - 2. od 21 do 30 godina,
  - 3. od 31 do 40 godina,
  - 4. od 41 do 50 godina,
  - 5. 50 i više godina.

Nacrtati pie grafikon za broj ljudi po godini starosti i napisati procente.

Nacrtati histogram za sve podatke.

Napisati izveštaj o izvučenim podacima.

Responder\_id,Age

1,14  
2,19  
3,28  
4,22  
5,30  
6,28  
7,42  
8,24  
9,23  
11,22  
12,21  
13,28  
14,31  
15,20  
16,26  
17,29

19,31  
20,38  
22,47  
23,22  
24,23  
25,22  
26,34  
27,22  
29,32  
30,28  
32,21  
33,25  
34,17  
35,35  
36,21  
37,27  
38,44  
39,42  
40,23  
42,30  
43,27  
44,43  
45,62  
46,37  
47,45  
48,20  
49,34  
50,23  
51,18  
52,22  
53,18  
54,37  
55,25  
56,33  
57,31  
58,47  
59,38  
60,27  
61,34  
62,19  
63,42  
64,25  
65,21  
67,24  
68,29  
69,18  
70,42  
71,21  
72,29  
73,36  
74,24  
75,27  
76,32

77,43  
78,25  
79,20  
80,24  
81,28  
83,22  
84,26  
85,31  
86,26  
87,36  
88,16  
90,25  
91,25  
92,35  
93,32  
94,27  
95,30  
96,43  
97,25  
98,30  
99,21  
100,39  
101,35  
102,37  
104,29  
106,33  
107,24  
108,64  
109,41  
110,26  
111,54  
112,19  
113,49  
114,26  
115,21  
116,31  
117,38  
118,26  
119,28  
120,26  
121,20  
123,24  
124,27  
125,35  
126,32  
127,36  
128,30  
129,34  
130,23  
132,30  
133,39  
134,54  
135,32

137,25  
139,24  
140,25  
141,21  
142,24  
143,36  
144,47  
146,47  
148,28  
149,40  
150,31  
151,34  
152,39  
154,40  
155,34  
156,36  
157,25  
158,29  
159,35  
160,43  
161,35  
163,32  
164,47  
166,30  
167,28  
169,32  
170,25  
172,31  
173,31  
174,24  
175,20  
176,31  
177,22  
178,28  
179,35  
180,32  
181,29  
182,18  
183,30  
184,29  
185,28  
186,26  
187,23  
188,28  
189,28  
190,27  
191,27  
192,28  
193,56  
194,29  
195,22  
196,40  
197,40

198,30  
200,43  
201,35  
202,45  
203,29  
204,12  
205,24  
206,33  
207,34  
208,26  
209,33  
210,47  
211,30  
212,30  
213,41  
214,58  
215,46  
216,35  
217,36  
218,36  
219,39  
221,47  
222,59  
223,23  
224,28  
225,38  
227,33  
228,25  
229,25  
231,31  
232,32  
233,36  
234,45  
235,21  
236,29  
237,21  
238,31  
240,21  
241,32  
242,30  
243,17  
244,44  
245,35  
246,33  
247,33  
248,30  
249,24  
250,27  
251,20  
252,27  
253,28  
254,41  
255,20

256,20  
257,43  
258,23  
259,31  
261,27  
262,24  
263,40  
265,40  
266,33  
267,23  
268,25  
269,32  
270,18  
271,24  
272,39  
273,29  
274,35  
275,34  
276,29  
277,29  
278,31  
279,27  
280,47  
281,40  
283,26  
284,19  
285,32  
286,23  
289,37  
290,30  
292,27  
293,31  
295,36  
296,22  
297,34  
298,23  
299,25  
301,39  
302,33  
303,28  
304,30  
305,28  
306,51  
307,21  
308,36  
309,20  
310,32  
311,24  
312,31  
313,23  
314,47  
315,23  
316,22

317,24  
318,21  
319,30  
320,46  
321,32  
322,23  
323,48  
324,34  
325,49  
326,34  
328,31  
329,47  
330,35  
331,29  
332,24  
333,33  
334,30  
335,30  
336,38  
337,31  
338,28  
339,16  
340,35  
341,23  
342,35  
343,24  
344,29  
345,33  
346,19  
347,41  
348,39  
349,26  
350,19  
351,25  
352,26  
353,30  
354,25  
355,35  
356,29  
357,33  
358,24  
359,37  
360,28  
361,25  
362,32  
363,37  
364,40  
366,22  
367,42  
368,25  
369,18  
370,32  
371,30

372,25  
373,37  
374,26  
375,57  
376,25  
377,37  
378,32  
379,19  
380,34  
381,22  
382,26  
383,38  
384,28  
386,27  
388,27  
389,38  
390,43  
391,16  
392,52  
394,30  
395,41  
396,46  
397,39  
398,22  
399,31  
400,48  
401,31  
402,49  
403,19  
404,27  
405,30  
406,24  
407,21  
408,32  
409,33  
410,43  
411,33  
412,28  
413,35  
414,44  
415,50  
416,21  
417,23  
418,45  
419,23  
421,27  
422,19  
423,34  
424,28  
425,44  
426,29  
427,32  
428,28

429,51  
430,20  
431,64  
432,24  
433,20  
434,41  
435,24  
436,38  
437,25  
438,24  
439,23  
440,34  
441,20  
442,33  
443,28  
444,33  
445,31  
446,25  
447,29  
449,44  
450,24  
451,20  
452,35  
453,39  
455,29  
456,24  
457,35  
458,25  
459,33  
460,32  
461,55  
462,37  
463,29  
464,26  
465,39  
466,23  
467,23  
468,40  
469,28  
470,23  
471,26  
472,46  
473,28  
474,23  
475,30  
476,29  
477,31  
478,17  
479,46  
480,19  
481,38  
482,41  
483,22

484,15  
485,22  
486,31  
487,24  
488,25  
489,32  
490,30  
491,22  
492,48  
493,22  
494,20  
495,23  
496,27  
497,41  
498,25  
499,27  
500,27  
501,18  
502,21  
503,37  
504,51  
505,21  
506,24  
507,24  
510,23  
511,33  
513,16  
514,31  
515,20  
516,29  
517,25  
519,34  
520,24  
521,35  
522,41  
523,21  
524,31  
525,31  
526,32  
527,31  
528,29  
529,20  
530,67  
531,22  
532,21  
533,30  
536,22  
537,22  
538,19  
539,40  
540,23  
541,31  
542,30

543,22  
544,38  
545,24  
546,19  
547,25  
548,46  
550,37  
551,30  
552,29  
553,42  
554,25  
555,24  
556,30  
557,44  
558,36  
560,33  
561,24  
562,23  
563,35  
564,35  
565,20  
566,40  
568,22  
569,30  
571,30  
572,21  
573,55  
574,30  
575,28  
576,21  
577,16  
578,28  
579,32  
580,42  
581,27  
582,24  
583,23  
585,23  
587,30  
589,24  
590,25  
591,29  
592,29  
593,23  
594,23  
595,33  
597,34  
598,41  
599,22  
600,15  
602,31  
603,43  
604,29

605,30  
606,35  
607,29  
608,28  
611,22  
613,34  
614,34  
615,29  
616,30  
617,27  
618,21  
619,22  
621,23  
622,26  
623,29  
624,18  
625,32  
626,30  
627,28  
628,33  
629,14  
630,25  
631,32  
632,41  
633,52  
635,36  
636,42  
638,34  
639,24  
640,36  
642,17  
643,39  
644,19  
645,30  
646,35  
647,23  
648,58  
650,24  
652,44  
654,37  
655,21  
656,43  
657,37  
658,28  
659,28  
660,35  
661,27  
662,31  
663,24  
664,30  
665,52  
666,33  
667,36

668,27  
669,23  
670,28  
671,29  
672,24  
673,13  
674,18  
675,28  
677,31  
678,31  
679,27  
680,36  
681,24  
682,37  
683,23  
684,27  
685,29  
686,37  
687,23  
688,19  
689,24  
691,28  
692,37  
693,33  
694,37  
695,25  
696,28  
697,35  
698,21  
699,28  
700,20  
701,29  
703,34  
704,31  
705,32  
706,40  
707,24  
708,36  
709,32  
710,15  
711,34  
712,28  
714,22  
715,41  
716,38  
717,23  
718,35  
720,34  
721,36  
722,1  
723,24  
724,39  
725,49

726,30  
727,22  
728,22  
729,36  
731,24  
732,25  
733,50  
734,28  
735,33  
736,33  
737,52  
738,18  
739,24  
740,29  
741,22  
742,19  
743,26  
744,38  
746,32  
747,40  
748,24  
749,34  
750,21  
751,32  
752,34  
753,27  
754,55  
755,37  
756,27  
757,28  
758,31  
759,50  
760,42  
761,28  
762,45  
763,30  
764,33  
765,27  
766,37  
767,30  
768,34  
769,24  
770,29  
771,27  
772,25  
773,21  
774,26  
776,31  
777,45  
778,32  
779,24  
780,24  
781,29

782,20  
783,22  
784,24  
785,37  
786,26  
787,29  
788,41  
789,31  
790,25  
791,24  
792,24  
793,28  
794,64  
795,22  
796,41  
797,36  
798,20  
799,21  
800,21  
801,30  
802,42  
803,39  
805,27  
806,31  
807,39  
808,44  
810,33  
811,24  
812,41  
813,43  
814,24  
815,50  
816,37  
817,23  
818,23  
819,33  
820,18  
821,29  
822,34  
823,35  
824,22  
825,26  
826,19  
828,28  
829,32  
830,36  
831,30  
832,21  
833,30  
834,28  
835,36  
837,21  
838,31

839,32  
841,35  
842,44  
843,27  
844,27  
845,44  
846,22  
848,26  
849,18  
850,36  
851,20  
852,28  
853,26  
854,46  
855,40  
856,45  
857,23  
858,17  
859,30  
860,24  
861,27  
862,29  
863,18  
864,25  
865,27  
866,35  
867,42  
868,21  
869,30  
870,25  
871,23  
872,32  
873,23  
874,25  
875,18  
876,29  
877,49  
878,32  
879,17  
880,30  
881,20  
882,20  
883,46  
884,28  
885,23  
886,26  
887,33  
888,34  
890,28  
891,15  
892,33  
893,22  
894,27

896,21  
898,25  
899,28  
900,26  
901,25  
902,23  
903,22  
904,40  
907,25  
909,28  
910,25  
911,22  
912,27  
913,20  
914,51  
915,38  
916,41  
917,33  
918,20  
919,32  
920,28  
921,30  
922,28  
923,31  
925,28  
926,20  
927,22  
928,27  
929,32  
930,33  
931,35  
932,34  
933,24  
934,27  
935,38  
936,16  
937,15  
938,31  
939,22  
941,20  
942,28  
943,36  
944,32  
945,27  
946,37  
947,18  
948,32  
949,24  
951,42  
952,38  
953,23  
954,19  
955,40

957,26  
958,28  
959,35  
960,20  
962,29  
964,30  
965,22  
967,36  
968,31  
969,27  
970,53  
971,33  
972,27  
973,22  
975,28  
976,16  
977,45  
978,27  
979,40  
980,35  
981,35  
982,19  
983,29  
984,25  
985,26  
986,24  
987,20  
989,19  
990,25  
991,29  
992,20  
993,36  
994,33  
995,23  
996,18  
997,25  
999,28  
1000,25  
1001,30  
1002,20  
1003,19  
1004,38  
1005,25  
1006,25  
1007,27  
1008,38  
1009,26  
1010,27  
1013,26  
1015,32  
1016,25  
1017,27  
1018,26

1019,32  
1021,41  
1022,24  
1023,29  
1024,26  
1025,29  
1026,25  
1027,34  
1028,22  
1029,59  
1030,29  
1031,33  
1032,25  
1033,50  
1034,26  
1035,43  
1036,40  
1037,31  
1038,21  
1039,37  
1041,35  
1042,44  
1043,35  
1045,32  
1046,23  
1047,25  
1048,27  
1050,21  
1051,40  
1052,37  
1053,18  
1054,28  
1055,25  
1056,34  
1057,38  
1061,38  
1062,25  
1063,31  
1064,29  
1065,28  
1066,40  
1067,29  
1068,27  
1069,40  
1070,35  
1071,39  
1072,20  
1073,37  
1074,28  
1075,50  
1076,27  
1077,22  
1079,24

1080,18  
1081,38  
1082,37  
1083,22  
1084,36  
1085,32  
1086,21  
1087,19  
1088,28  
1089,28  
1090,43  
1091,21  
1092,19  
1093,21  
1094,69  
1095,31  
1096,19  
1097,26  
1098,27  
1099,33  
1101,30  
1102,53  
1103,29  
1105,65  
1106,17  
1108,21  
1109,20  
1110,30  
1111,28  
1113,38  
1114,42  
1115,18  
1116,26  
1117,25  
1118,48  
1119,17  
1120,26  
1121,22  
1122,27  
1123,37  
1124,35  
1125,19  
1127,28  
1128,27  
1129,30  
1130,33  
1131,24  
1132,29  
1133,24  
1134,19  
1135,19  
1136,28  
1137,34

1138,21  
1141,15  
1142,25  
1143,24  
1144,36  
1145,25  
1146,23  
1147,45  
1148,30  
1149,29  
1150,42  
1151,44  
1152,33  
1153,27  
1154,28  
1155,25  
1157,23  
1158,37  
1159,28  
1161,33  
1162,27  
1164,40  
1165,28  
1166,26  
1167,49  
1169,36  
1170,55  
1173,21  
1174,22  
1175,19  
1176,35  
1177,23  
1178,27  
1179,26  
1181,23  
1182,34  
1183,29  
1184,18  
1185,27  
1186,31  
1187,29  
1188,26  
1189,25  
1190,20  
1192,40  
1193,57  
1194,38  
1195,15  
1197,27  
1198,18  
1200,28  
1201,28  
1202,32

1203,36  
1205,28  
1206,29  
1207,26  
1208,24  
1209,31  
1210,35  
1211,34  
1212,22  
1214,28  
1215,24  
1216,25  
1217,63  
1218,15  
1219,38  
1221,23  
1222,27  
1223,37  
1224,32  
1225,34  
1226,48  
1228,27  
1229,34  
1230,39  
1232,25  
1233,30  
1234,42  
1235,23  
1236,33  
1237,32  
1238,38  
1239,29  
1240,22  
1241,34  
1243,24  
1244,37  
1245,24  
1246,37  
1247,26  
1249,18  
1250,26  
1251,18  
1253,40  
1254,37  
1255,29  
1256,19  
1257,52  
1258,42  
1259,26  
1260,28  
1262,29  
1263,23  
1264,21

1266,33  
1267,26  
1268,48  
1269,27  
1270,50  
1271,37  
1272,25  
1273,21  
1274,46  
1275,21  
1276,23  
1277,24  
1278,55  
1279,33  
1280,28  
1281,29  
1283,59  
1284,37  
1285,25  
1286,36  
1287,23  
1288,35  
1289,35  
1290,31  
1291,22  
1292,36  
1294,23  
1295,23  
1296,32  
1297,34  
1298,32  
1299,24  
1300,28  
1301,33  
1302,29  
1303,22  
1304,30  
1305,27  
1307,39  
1308,49  
1309,43  
1310,35  
1311,16  
1312,61  
1313,25  
1314,39  
1315,28  
1316,40  
1318,36  
1319,18  
1320,28  
1321,36  
1322,43

1323,22  
1324,30  
1325,43  
1326,42  
1328,36  
1329,33  
1330,26  
1331,33  
1332,52  
1333,22  
1334,29  
1335,24  
1336,30  
1337,42  
1339,42  
1340,22  
1341,32  
1342,23  
1343,33  
1344,28  
1345,23  
1346,28  
1347,26  
1348,32  
1349,22  
1351,23  
1354,30  
1355,22  
1356,25  
1357,36  
1358,26  
1359,29  
1360,18  
1361,24  
1362,33  
1363,35  
1364,30  
1365,27  
1367,29  
1369,22  
1370,43  
1371,42  
1372,26  
1373,38  
1374,18  
1375,31  
1376,36  
1378,40  
1379,26  
1380,20  
1381,33  
1382,32  
1383,29

1384,19  
1385,44  
1386,31  
1387,29  
1388,38  
1390,50  
1391,27  
1392,40  
1393,27  
1395,30  
1396,40  
1397,38  
1398,67  
1399,21  
1400,62  
1401,20  
1402,32  
1403,46  
1404,31  
1405,28  
1406,20  
1407,23  
1408,34  
1409,35  
1410,28  
1411,34  
1412,33  
1413,29  
1414,26  
1417,29  
1418,19  
1419,28  
1420,38  
1421,32  
1422,42  
1423,31  
1424,20  
1425,23  
1426,26  
1427,43  
1428,37  
1429,35  
1430,28  
1431,31  
1432,54  
1433,22  
1434,22  
1435,43  
1437,29  
1438,26  
1439,22  
1440,15  
1441,33

1443,23  
1444,30  
1445,38  
1446,46  
1449,68  
1450,30  
1451,30  
1452,28  
1453,40  
1454,32  
1455,35  
1456,51  
1457,48  
1458,32  
1459,25  
1460,26  
1461,29  
1462,24  
1463,47  
1464,23  
1466,33  
1469,24  
1470,23  
1471,30  
1472,33  
1473,34  
1474,30  
1475,34  
1477,40  
1478,25  
1480,33  
1481,36  
1482,34  
1483,20  
1484,23  
1485,25  
1486,29  
1487,35  
1488,39  
1489,25  
1490,29  
1491,33  
1492,20  
1493,28  
1496,26  
1499,38  
1500,22  
1501,24  
1502,50  
1503,27  
1504,18  
1505,34  
1506,16

1507,22  
1508,19  
1509,25  
1510,24  
1511,34  
1512,28  
1513,38  
1514,32  
1515,24  
1516,25  
1517,30  
1518,30  
1519,27  
1520,31  
1521,28  
1522,24  
1523,30  
1524,38  
1525,41  
1527,18  
1528,17  
1529,24  
1530,28  
1531,33  
1532,28  
1533,22  
1534,23  
1535,29  
1536,33  
1537,23  
1538,45  
1539,20  
1540,27  
1541,32  
1542,42  
1543,30  
1544,26  
1545,42  
1546,49  
1548,32  
1549,26  
1550,24  
1551,25  
1552,38  
1553,30  
1554,22  
1555,17  
1556,53  
1557,30  
1558,36  
1559,27  
1560,31  
1561,24

1563,26  
1564,39  
1565,27  
1566,24  
1567,30  
1568,21  
1569,25  
1570,17  
1571,22  
1572,32  
1573,25  
1574,15  
1575,21  
1577,30  
1578,23  
1579,26  
1580,27  
1581,17  
1583,31  
1584,22  
1585,58  
1586,24  
1588,57  
1589,30  
1590,23  
1591,52  
1592,38  
1593,30  
1594,45  
1595,19  
1597,27  
1598,32  
1599,63  
1600,34  
1601,42  
1602,28  
1603,31  
1604,28  
1605,27  
1606,40  
1607,31  
1608,44  
1609,29  
1610,34  
1611,19  
1612,26  
1613,27  
1614,30  
1616,65  
1617,54  
1619,33  
1620,29  
1621,26

1622,23  
1623,34  
1624,22  
1625,23  
1626,44  
1627,44  
1628,35  
1629,34  
1630,22  
1632,26  
1633,54  
1634,22  
1635,28  
1636,39  
1637,44  
1639,31  
1641,28  
1642,24  
1643,35  
1644,30  
1645,34  
1647,21  
1648,28  
1649,19  
1650,32  
1651,26  
1652,28  
1653,30  
1654,24  
1655,24  
1656,25  
1657,32  
1658,30  
1659,37  
1660,23  
1661,28  
1663,16  
1664,35  
1665,31  
1666,25  
1667,49  
1668,29  
1669,28  
1670,48  
1671,33  
1673,28  
1674,34  
1675,34  
1676,24  
1677,25  
1678,42  
1679,23  
1680,26

1681,22  
1682,35  
1683,29  
1684,34  
1685,29  
1686,21  
1687,18  
1689,42  
1690,30  
1691,73  
1692,29  
1693,33  
1694,49  
1695,21  
1696,34  
1697,24  
1698,63  
1699,31  
1702,18  
1703,32  
1704,17  
1705,25  
1706,28  
1708,27  
1709,34  
1711,53  
1712,44  
1713,33  
1714,20  
1715,38  
1716,25  
1717,35  
1719,29  
1720,20  
1722,27  
1723,30  
1724,37  
1725,48  
1726,22  
1727,23  
1728,35  
1729,16  
1730,31  
1731,36  
1732,28  
1733,24  
1734,21  
1735,35  
1736,32  
1737,27  
1738,20  
1739,26  
1740,22

1742,25  
1743,29  
1744,50  
1745,27  
1748,41  
1749,34  
1750,42  
1751,21  
1752,32  
1753,26  
1754,30  
1755,55  
1756,32  
1758,26  
1759,23  
1761,26  
1762,39  
1763,27  
1764,23  
1766,36  
1767,25  
1768,27  
1769,25  
1770,28  
1771,42  
1772,47  
1773,24  
1774,25  
1776,33  
1777,29  
1778,23  
1779,25  
1780,25  
1781,29  
1783,22  
1784,28  
1785,24  
1786,24  
1787,30  
1788,23  
1789,34  
1790,29  
1791,24  
1793,31  
1795,23  
1796,24  
1797,33  
1798,30  
1800,23  
1801,36  
1802,29  
1803,25  
1804,28

1805,38  
1807,24  
1808,26  
1809,27  
1811,34  
1812,52  
1813,38  
1814,33  
1815,22  
1816,26  
1817,57  
1818,14  
1819,27  
1820,34  
1821,34  
1822,26  
1823,36  
1824,27  
1825,38  
1826,33  
1827,24  
1828,61  
1829,44  
1830,28  
1831,27  
1832,23  
1833,40  
1834,29  
1835,38  
1836,31  
1837,25  
1838,28  
1839,58  
1840,30  
1841,32  
1842,38  
1843,37  
1844,44  
1845,26  
1846,25  
1847,28  
1848,32  
1849,34  
1850,40  
1851,38  
1852,35  
1853,47  
1854,38  
1855,26  
1856,27  
1857,14  
1858,22  
1859,25

1860,50  
1861,36  
1862,39  
1863,26  
1864,48  
1865,25  
1867,39  
1868,30  
1869,28  
1870,25  
1871,32  
1873,36  
1874,37  
1875,23  
1876,15  
1877,27  
1878,33  
1879,43  
1880,39  
1881,41  
1883,34  
1884,35  
1885,25  
1886,27  
1887,35  
1888,31  
1891,31  
1892,30  
1893,26  
1894,35  
1895,27  
1896,23  
1897,29  
1898,44  
1899,21  
1900,20  
1901,18  
1902,35  
1903,36  
1904,27  
1906,33  
1907,30  
1908,35  
1909,21  
1910,41  
1911,32  
1913,27  
1914,39  
1915,30  
1916,27  
1917,28  
1918,41  
1919,39

1920,55  
1921,30  
1922,35  
1923,29  
1924,23  
1925,27  
1928,30  
1929,23  
1931,38  
1932,21  
1933,35  
1934,29  
1935,35  
1936,30  
1937,34  
1938,49  
1939,35  
1940,32  
1941,30  
1942,30  
1943,36  
1944,34  
1946,46  
1947,22  
1948,34  
1949,26  
1950,28  
1951,26  
1952,29  
1953,41  
1954,37  
1955,25  
1956,40  
1957,27  
1958,33  
1959,34  
1960,28  
1961,37  
1962,24  
1963,25  
1964,25  
1965,19  
1966,22  
1967,50  
1968,35  
1969,28  
1970,29  
1971,24  
1973,15  
1974,28  
1975,24  
1976,24  
1977,23

1978,36  
1979,24  
1981,22  
1982,27  
1983,31  
1984,30  
1985,47  
1986,45  
1987,17  
1988,23  
1989,23  
1990,31  
1992,25  
1993,35  
1997,50  
1999,32  
2000,29

## ZADATAK #2

*Zadatak #2 se radi okvirno 45 minuta*

### **Zadatak #2 (45 minuta)**

Date podatke ručno kopirati u tekstualnu datoteku i sačuvati kao CSV datoteku. Zatim, napisati program koji učitava CSV datoteku sa podacima, i iz datih podatka računa sledeće:

- Prosečnu godinu starosti svih koji su odgovorili da znaju Python jezik,
- Najčešću godinu starosti svih koji znaju C++ jezik,
- Najčešću godinu starosti svih koji znaju JavaScript jezik,
- Histogram svih učesnika u anketi po godinama,
- Histogram svih učesnika u anketi po broju programske jezika koje znaju (1, 2, 3, 4, 5+)
- Svi koji su ispod 30 godina i znaju R, i svi koji su iznad 30 godina i znaju Go,
- Svi koji su iznad 40 godina i znaju PHP i HTML/CSS.

Napisati izveštaj o izvučenim podacima.

```
Responder_id,Responder_age,LanguagesWorkedWith
1,28,HTML/CSS;Java;JavaScript;Python
2,31,C++;HTML/CSS;Python
3,48,HTML/CSS
4,29,C;C++;C#;Python;SQL
5,20,C++;HTML/CSS;Java;JavaScript;Python;SQL;VBA
6,29,Java;R;SQL
7,31,HTML/CSS;JavaScript
8,36,Bash/Shell/PowerShell;C;C++;HTML/CSS;Java;JavaScript;Python;SQL
9,49,Bash/Shell/PowerShell;C#;HTML/
CSS;JavaScript;Python;Ruby;Rust;SQL;TypeScript;WebAssembly;Other(s):
10,41,C#;Go;JavaScript;Python;R;SQL
11,19,Other(s):
```

12,37,Bash/Shell/PowerShell;HTML/CSS;Java;Python;R;SQL  
13,41,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;SQL;TypeScript  
14,22,C++  
15,18,Assembly;Bash/Shell/PowerShell;C;C++;HTML/CSS;Java;JavaScript;PHP;SQL  
16,41,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;TypeScript;VBA  
17,32,Bash/Shell/PowerShell;HTML/CSS;JavaScript;TypeScript  
18,43,Python;R  
19,56,C#;HTML/CSS;Java;JavaScript;SQL;TypeScript  
20,52,Bash/Shell/PowerShell;C#;HTML/CSS;Java;JavaScript;PHP;Python;R;SQL  
21,47,Assembly;Bash/Shell/  
PowerShell;C;C++;Go;Java;JavaScript;Kotlin;Python;Rust;SQL;Swift  
22,38,Bash/Shell/PowerShell;C++;HTML/CSS;JavaScript;Python;Ruby;SQL;TypeScript  
23,42,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Python;Ruby;SQL  
24,24,HTML/CSS;JavaScript;PHP;TypeScript  
25,41,HTML/CSS;JavaScript;PHP;SQL;TypeScript  
26,43,Bash/Shell/PowerShell;C++;C#;HTML/  
CSS;JavaScript;PHP;Python;Ruby;SQL;Swift;TypeScript;VBA  
27,22,C++;JavaScript;Python;Ruby;SQL;TypeScript  
28,42,JavaScript;TypeScript  
29,26,Bash/Shell/PowerShell;JavaScript;SQL  
31,45,Python  
32,54,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;Python  
33,41,C++;Python;R  
34,23,HTML/CSS;JavaScript  
35,27,HTML/CSS;JavaScript  
36,25,Java;Kotlin;Python  
37,41,Bash/Shell/PowerShell;JavaScript;Python;Other(s):  
38,35,C#;HTML/CSS;JavaScript;SQL  
39,57,C#;JavaScript;SQL;TypeScript  
40,47,C#;HTML/CSS  
41,51,Bash/Shell/PowerShell;C;C++;HTML/CSS;Java;JavaScript;SQL  
42,20,HTML/CSS;JavaScript;PHP;TypeScript  
43,42,C++;C#;HTML/CSS;Java;JavaScript;Objective-C;SQL  
44,20,Bash/Shell/PowerShell;C#;HTML/CSS;Java;JavaScript;SQL;TypeScript;WebAssembly  
45,25,Python  
46,20,Bash/Shell/PowerShell;C;C#;HTML/CSS;JavaScript;PHP;Python;SQL;Other(s):  
47,50,Java;PHP;Ruby  
48,20,HTML/CSS;PHP;SQL  
49,38,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;PHP;Python;SQL;TypeScript  
50,20,Bash/Shell/PowerShell;C;C++;HTML/CSS;Java;JavaScript;PHP;Python;SQL  
51,25,Bash/Shell/PowerShell;C++;HTML/CSS;Java;JavaScript;Python;R;TypeScript  
52,52,Bash/Shell/PowerShell;C;C++;Elixir;Erlang;Go;HTML/  
CSS;Java;JavaScript;Kotlin;Python;Ruby;Rust;SQL;TypeScript  
53,43,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;PHP;SQL;TypeScript  
54,23,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Python  
55,39,Java;Python;SQL  
56,27,Bash/Shell/PowerShell;HTML/CSS;Java;SQL  
57,26,JavaScript;Python  
58,54,C#;Java;SQL  
59,40,HTML/CSS;JavaScript;PHP;SQL  
60,32,Bash/Shell/PowerShell;Go;JavaScript;PHP;Python;Ruby;SQL  
61,47,C++;C#;HTML/CSS;JavaScript;PHP;Python;SQL;VBA  
62,42,Bash/Shell/PowerShell;C++;Go;HTML/

CSS;Java;JavaScript;Kotlin;PHP;Python;Ruby;SQL;TypeScript;VBA  
63,34,Bash/Shell/PowerShell;Clojure;Java;Python;Other(s):  
64,22,Bash/Shell/PowerShell;C;C++;C#  
65,32,Assembly;C;C++;C#;HTML/CSS;Java  
66,27,Clojure;Go;HTML/CSS;Java;JavaScript;R;SQL  
67,19,C;C#;HTML/CSS;Java;JavaScript;PHP;Python;SQL;VBA  
68,44,HTML/CSS;Java;JavaScript;Python  
69,47,C#;HTML/CSS;Java;JavaScript;Objective-C;SQL;TypeScript  
70,50,C#;HTML/CSS;JavaScript;SQL  
71,25,HTML/CSS;JavaScript;PHP;Python;SQL;VBA  
72,24,C#;HTML/CSS;JavaScript;PHP;SQL;TypeScript  
73,54,SQL  
74,27,HTML/CSS;Java;JavaScript;Kotlin;Python;Ruby  
75,40,HTML/CSS;JavaScript  
76,26,PHP;SQL  
77,32,HTML/CSS;JavaScript;PHP;SQL  
78,25,HTML/CSS;Java;JavaScript;Kotlin;Python  
79,28,C#;HTML/CSS;JavaScript;TypeScript  
80,27,Bash/Shell/PowerShell;C#;F#;Go;HTML/CSS;Java;JavaScript;Python;SQL;TypeScript  
81,50,Assembly;Bash/Shell/PowerShell;C;C++;Python  
82,34,Bash/Shell/PowerShell;C++;HTML/CSS;Java;JavaScript;Python;Rust  
83,28,HTML/CSS;JavaScript  
84,44,C;C++;C#;Java;Kotlin;PHP;SQL  
85,22,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;Python;SQL;TypeScript  
86,21,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;PHP;Python;SQL  
87,28,Bash/Shell/PowerShell;C;C++;C#;Go;HTML/  
CSS;Java;JavaScript;Objective-C;Python;SQL  
88,50,C++;C#;HTML/CSS;Java;JavaScript;SQL;TypeScript  
89,51,Bash/Shell/PowerShell;Python  
90,30,Bash/Shell/PowerShell;C#;HTML/  
CSS;Java;JavaScript;Objective-C;PHP;Python;Ruby;SQL;Swift;TypeScript  
91,42,HTML/CSS;JavaScript;TypeScript  
92,50,HTML/CSS;Java;JavaScript;Kotlin;SQL;VBA  
93,45,Python;SQL  
94,27,C#;HTML/CSS;JavaScript;SQL  
95,42,C#;HTML/CSS;JavaScript;Python;R;SQL  
96,22,C;C++;Java;Python;R;Scala;SQL  
97,39,Bash/Shell/PowerShell;JavaScript;Python;R;SQL  
98,33,HTML/CSS;Java;JavaScript;SQL  
99,40,C;C++;HTML/CSS;Java;JavaScript;Kotlin;PHP;Python;SQL  
100,42,HTML/CSS;JavaScript;Ruby;SQL;TypeScript  
101,31,Bash/Shell/PowerShell;HTML/CSS;JavaScript;SQL  
102,27,C#;HTML/CSS;JavaScript;SQL;TypeScript  
103,57,Clojure;Go;Java;Kotlin  
104,57,C#;HTML/CSS;TypeScript  
105,38,Bash/Shell/PowerShell;HTML/CSS;JavaScript;SQL  
106,53,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;Python;SQL  
107,53,HTML/CSS;Java;Python  
108,35,C++;Python  
109,26,C#;HTML/CSS;JavaScript;SQL  
110,37,Python;SQL  
111,37,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;Python;SQL;VBA  
112,40,Assembly;C;C++;C#;HTML/CSS;Java;JavaScript;Python;VBA

113,34,Bash/Shell/PowerShell;C;C++  
114,34,Bash/Shell/PowerShell;C++;Erlang;JavaScript;PHP;Python  
115,55,Assembly;C#;HTML/CSS;Java;JavaScript;SQL;Swift  
116,45,Scala;Other(s):  
117,34,C#;HTML/CSS;JavaScript;SQL;TypeScript  
118,36,Bash/Shell/PowerShell;C;C++  
119,37,C;C++;C#;HTML/CSS;Java;JavaScript;SQL  
120,54,Elixir;HTML/CSS;JavaScript;Python;Ruby;SQL  
121,50,Bash/Shell/PowerShell;C;C++;HTML/CSS;JavaScript;Python  
122,22,Assembly;Bash/Shell/PowerShell;C++;Python;SQL  
123,50,HTML/CSS;JavaScript;TypeScript  
124,31,HTML/CSS;Java;SQL  
125,37,Bash/Shell/PowerShell;Dart;HTML/CSS;Java;JavaScript;Scala  
126,29,C#;SQL  
127,49,Bash/Shell/PowerShell;C;Python  
128,31,Bash/Shell/PowerShell;Go;Ruby  
129,53,C++;Python;R  
130,50,Bash/Shell/PowerShell;C++;Clojure;HTML/CSS;Java;Python;Ruby;SQL  
131,19,Java  
132,55,Bash/Shell/PowerShell;C++;C#;HTML/  
CSS;Java;JavaScript;Objective-C;Python;TypeScript  
133,33,JavaScript;PHP;SQL  
134,21,C;C++;HTML/CSS;JavaScript;SQL  
135,23,Go;HTML/CSS;JavaScript;TypeScript  
136,46,C#;Java;PHP;Python  
137,36,HTML/CSS;Java;JavaScript;PHP;SQL  
139,37,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;Python;SQL  
140,22,Java;JavaScript;Kotlin;PHP;SQL  
141,53,Assembly;Bash/Shell/PowerShell;C;C++;HTML/  
CSS;Java;JavaScript;Python;SQL;TypeScript  
142,53,Bash/Shell/PowerShell;C#  
143,32,C;HTML/CSS;Java;JavaScript;Python;SQL;TypeScript  
144,48,Java;SQL  
145,55,Assembly;C++;Python;VBA  
146,36,Bash/Shell/PowerShell;C;Python;Scala  
147,39,HTML/CSS;Java;JavaScript;Kotlin  
148,55,HTML/CSS;JavaScript;TypeScript  
149,45,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Python;SQL  
150,45,Bash/Shell/PowerShell;C#;HTML/CSS;Java;JavaScript;Python;SQL  
151,31,Bash/Shell/PowerShell;C++;C#;HTML/CSS;Java;JavaScript;SQL;TypeScript  
152,25,HTML/CSS;Java;JavaScript;Kotlin;Objective-C;Python;Swift;Other(s):  
153,49,C;C++;Python  
154,55,HTML/CSS;Java;JavaScript;PHP;SQL;VBA  
155,48,Bash/Shell/PowerShell;HTML/CSS;JavaScript;TypeScript  
156,48,Bash/Shell/PowerShell;Python  
157,32,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;SQL  
158,25,Java;SQL;Other(s):  
159,31,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Python  
160,51,C#  
161,57,HTML/CSS;JavaScript;PHP  
162,23,Bash/Shell/PowerShell;C++;HTML/CSS;Java;JavaScript;PHP;SQL  
163,38,C#;HTML/CSS;JavaScript;SQL;Other(s):  
164,26,C#;HTML/CSS;JavaScript;SQL

165,49,Assembly;Bash/Shell/PowerShell;C;C++;C#;JavaScript;Python;SQL  
166,46,Bash/Shell/PowerShell;Go;HTML/CSS;Java;JavaScript;Ruby;SQL  
167,28,HTML/CSS;Java;JavaScript;PHP;Python;SQL  
168,44,Bash/Shell/PowerShell;Python  
169,36,C;C++;HTML/CSS;Java;JavaScript;PHP;Python;SQL  
170,20,Clojure;Go;HTML/CSS;Java;JavaScript;Python;Ruby;TypeScript  
171,53,Assembly;C;C++;C#;HTML/CSS;Java;Objective-C;PHP;Python;R;SQL;Swift  
172,51,HTML/CSS;JavaScript;Objective-C;SQL  
173,37,HTML/CSS;JavaScript;PHP;SQL;TypeScript  
174,46,Bash/Shell/PowerShell;C;C++;HTML/CSS;Java;JavaScript;PHP;Python;R  
175,54,Assembly;Bash/Shell/PowerShell;C;C++;HTML/CSS;Java;JavaScript;PHP;Python;SQL  
176,25,C;C++;C#;HTML/CSS;Java;JavaScript;Python;SQL  
177,53,C#;Python;SQL  
178,28,JavaScript;PHP;Python;Ruby;SQL;TypeScript  
179,22,HTML/CSS;Java;JavaScript;Objective-C;SQL;Swift  
181,31,HTML/CSS;JavaScript  
182,31,Bash/Shell/PowerShell;C;C++;Go;HTML/  
CSS;Java;JavaScript;PHP;Python;Rust;SQL;TypeScript  
183,25,Dart;Go;Java;Kotlin;Swift  
184,29,Bash/Shell/  
PowerShell;C;C++;Java;JavaScript;Kotlin;Objective-C;Python;Rust;Swift  
185,24,Bash/Shell/PowerShell;Java;Kotlin  
186,35,Assembly;C;C++;HTML/CSS;Java;JavaScript;Python  
187,51,HTML/CSS;JavaScript;Ruby;Scala;SQL  
188,38,C#;Python;R;SQL;VBA  
189,39,C++;HTML/CSS;JavaScript;Python;SQL  
190,55,Java;Scala;SQL;TypeScript  
191,22,C#;HTML/CSS;JavaScript;SQL;TypeScript  
192,38,Bash/Shell/PowerShell;JavaScript;Python  
193,48,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Ruby;SQL  
194,40,Assembly;Bash/Shell/PowerShell;C;HTML/CSS;Java;JavaScript;PHP  
195,47,C#;TypeScript;Other(s):  
196,26,C#;HTML/CSS;JavaScript;SQL;TypeScript  
197,25,HTML/CSS;R;VBA  
198,45,HTML/CSS;Java;JavaScript;Python;SQL  
199,21,R;SQL  
200,56,Java;JavaScript;SQL;VBA  
201,18,Bash/Shell/PowerShell;Go;HTML/CSS;Java;JavaScript;Python;Scala  
202,53,Bash/Shell/PowerShell;C;C++;C#;HTML/  
CSS;Java;JavaScript;Objective-C;Python;SQL;TypeScript  
203,24,C;C++;HTML/CSS;JavaScript;PHP;SQL  
204,52,Python;Other(s):  
205,35,C#;HTML/CSS;JavaScript;PHP  
206,56,Scala;SQL  
207,34,Java  
208,52,Other(s):  
209,29,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;Python;SQL  
210,31,C++;Go;Java;JavaScript;Python;Rust;WebAssembly  
211,57,C#;HTML/CSS;Java;JavaScript;Objective-C;SQL  
212,46,C#;HTML/CSS;Java;JavaScript;PHP  
213,29,Go;Python  
214,45,R;Other(s):  
215,19,Python;Rust

216,36,Go;Java;Python  
217,22,Go;HTML/CSS;JavaScript;PHP;TypeScript  
218,33,HTML/CSS;JavaScript;PHP;SQL  
219,28,C#;HTML/CSS;JavaScript;SQL  
220,21,C#;HTML/CSS;JavaScript;SQL  
221,29,C;C++;HTML/CSS;Java;JavaScript;PHP;SQL  
222,32,Assembly;Bash/Shell/PowerShell;C;HTML/CSS;JavaScript;Other(s):  
223,24,Java;Kotlin  
224,45,Go  
225,48,Bash/Shell/PowerShell;C;C++;C#;HTML/CSS;Java;JavaScript;PHP;Python;R;SQL  
226,20,HTML/CSS;JavaScript;PHP;Python;SQL  
227,21,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Python;SQL  
228,54,HTML/CSS;Java;JavaScript;Python;Scala;SQL  
229,51,HTML/CSS;Java;JavaScript;Python;TypeScript  
230,49,Bash/Shell/PowerShell;Go;Python  
231,20,Elixir;Go;JavaScript;Ruby;SQL  
232,36,Bash/Shell/PowerShell;C#;Dart;HTML/CSS;JavaScript  
233,49,C++;Python  
234,31,Bash/Shell/PowerShell;C;Python  
235,22,HTML/CSS;Java;Python;SQL  
236,41,HTML/CSS;JavaScript  
237,18,C++;JavaScript;Rust  
238,27,HTML/CSS;JavaScript;Objective-C;PHP;Python  
239,42,C#;HTML/CSS;JavaScript;TypeScript  
240,51,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;Python;SQL  
241,24,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;SQL  
242,29,HTML/CSS;Java;JavaScript;SQL;TypeScript  
243,18,C;C++;C#;HTML/CSS;Java;JavaScript;Objective-C;PHP;Python;Swift  
244,42,Assembly;Bash/Shell/PowerShell;C;Python;VBA  
245,28,C#;JavaScript;TypeScript  
246,32,C#;HTML/CSS;JavaScript;SQL;TypeScript  
247,36,C++;C#  
248,20,HTML/CSS;Java;JavaScript;Python;SQL  
249,41,Bash/Shell/PowerShell;C#;HTML/CSS;Java;Python;SQL  
250,47,Python;Other(s):  
251,45,Assembly;C;C++;C#;HTML/CSS;JavaScript;Python;VBA  
252,29,HTML/CSS;JavaScript;Rust;Swift;Other(s):  
253,25,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;Ruby;SQL  
254,46,C#;Python;SQL;TypeScript  
255,50,Bash/Shell/PowerShell;C;HTML/CSS;Java;JavaScript;Python;TypeScript;Other(s):  
256,20,C;C++;Java  
257,21,Bash/Shell/PowerShell;Other(s):  
258,29,C;HTML/CSS;Java;JavaScript;PHP;Python  
259,24,C#;HTML/CSS;Java;JavaScript;Python;SQL;TypeScript  
260,55,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;PHP;SQL  
261,54,HTML/CSS;JavaScript  
262,37,C++;C#;HTML/CSS;Java;JavaScript;PHP;Python;SQL;TypeScript  
263,34,HTML/CSS;JavaScript;Kotlin;Python  
264,56,HTML/CSS;Java;JavaScript;Python;SQL;Other(s):  
265,34,HTML/CSS;JavaScript  
266,25,HTML/CSS;JavaScript;TypeScript;Other(s):  
267,40,HTML/CSS;JavaScript;Python;Ruby  
268,52,C;Java;PHP;Ruby

269,48,C#;HTML/CSS;Java;JavaScript;SQL;Swift  
270,43,C++;HTML/CSS;JavaScript  
271,38,C#;JavaScript;SQL  
272,39,HTML/CSS;JavaScript;PHP  
273,53,Bash/Shell/PowerShell;C++;Clojure;Erlang;HTML/CSS;Java;JavaScript;Ruby;Scala  
274,38,Java;JavaScript  
275,56,C#;HTML/CSS;Java;JavaScript;Objective-C;Python;Ruby;Rust;SQL;TypeScript  
276,50,HTML/CSS;Java;JavaScript;Scala;SQL  
277,23,Bash/Shell/PowerShell;C;C++;Python  
278,41,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;PHP;Python;SQL  
279,47,HTML/CSS;Java;JavaScript;PHP;Python;SQL  
280,52,C#;HTML/CSS;JavaScript;SQL;TypeScript  
281,44,C#;HTML/CSS;Java;JavaScript;TypeScript  
282,49,Python;SQL;Swift  
283,56,HTML/CSS;JavaScript;TypeScript  
284,54,Assembly;Bash/Shell/PowerShell;C;C++;C#;Java  
285,39,C#;HTML/CSS;JavaScript;PHP;SQL  
286,32,Dart;HTML/CSS;JavaScript;Python;TypeScript;Other(s):  
287,38,Java;Kotlin  
288,24,HTML/CSS;JavaScript;PHP;SQL  
289,18,C++;C#;PHP;Rust;TypeScript  
290,20,HTML/CSS;PHP;Python  
291,35,Go;HTML/CSS;Python;SQL;TypeScript  
292,36,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;SQL;TypeScript  
293,48,HTML/CSS;Java;JavaScript;SQL  
295,36,C;C++;C#;HTML/  
CSS;Java;JavaScript;Kotlin;PHP;Python;Ruby;SQL;TypeScript;WebAssembly  
296,54,HTML/CSS;Java;JavaScript;SQL  
297,25,HTML/CSS;Java;JavaScript;SQL;TypeScript  
298,53,HTML/CSS;JavaScript;PHP;SQL;Other(s):  
299,41,Assembly;C;C++;Java;Python;SQL  
300,36,HTML/CSS;Java;JavaScript;SQL  
301,47,C++;C#;Clojure;JavaScript;PHP;Python;SQL;VBA  
302,52,Bash/Shell/PowerShell;C#;Python;Other(s):  
303,39,JavaScript  
304,53,Bash/Shell/PowerShell;Go;HTML/CSS;JavaScript;Python;Rust;SQL  
305,52,C#;HTML/CSS;JavaScript;SQL  
306,57,C;HTML/CSS;JavaScript;Python;SQL;TypeScript  
307,42,Assembly;Bash/Shell/PowerShell;C;HTML/CSS;Java;PHP;Python;R;SQL  
308,23,HTML/CSS;JavaScript;Ruby  
309,26,Java  
310,42,C#;HTML/CSS;JavaScript;SQL;Swift  
311,55,Assembly;C;C++;C#;HTML/CSS;Java;Python;Scala;SQL  
312,19,Bash/Shell/PowerShell;Python;Ruby;SQL  
313,28,C;C#;HTML/CSS;PHP;Python  
314,52,C;Python  
315,46,C++;HTML/CSS;Java;JavaScript;Python;SQL;Other(s):  
316,25,Bash/Shell/PowerShell;C++;HTML/CSS;Java;JavaScript;Python  
317,47,Java;SQL  
318,37,Bash/Shell/PowerShell;Java  
319,33,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;TypeScript  
320,35,C#;HTML/CSS;JavaScript;SQL;TypeScript  
321,53,HTML/CSS;Java;JavaScript;Ruby;SQL;TypeScript

322,45,C#;HTML/CSS;Java;Python  
323,41,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;SQL  
324,23,C#;HTML/CSS;JavaScript;SQL;TypeScript  
325,31,Assembly;Bash/Shell/PowerShell;C;C++;C#;HTML/CSS;Java;JavaScript;PHP;Python  
326,35,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;PHP;Python;SQL;TypeScript  
327,43,HTML/CSS;Java;JavaScript;PHP;Python;SQL  
328,21,HTML/CSS;Java;JavaScript;TypeScript  
329,21,C;C++;C#;F#;HTML/CSS;JavaScript;Rust;SQL;TypeScript;Other(s):  
330,54,Java;Kotlin  
331,35,Bash/Shell/PowerShell;C++;C#;Python  
332,27,Bash/Shell/PowerShell;HTML/  
CSS;JavaScript;Objective-C;Python;SQL;Swift;TypeScript  
333,30,Python;Rust;Scala  
334,37,Bash/Shell/PowerShell;C;C++;Python  
335,42,C#;HTML/CSS;Java;JavaScript;Python;SQL;Swift  
336,28,Bash/Shell/PowerShell;JavaScript;Python;Rust;SQL;TypeScript  
337,19,Python;R  
338,28,Bash/Shell/PowerShell;HTML/CSS;Java;Python;SQL  
339,56,Bash/Shell/PowerShell;Dart;Go;HTML/CSS;Java;JavaScript;Python;Rust;SQL;Swift  
340,27,C#;HTML/CSS;SQL  
341,39,C++;C#;SQL  
342,38,HTML/CSS;Java;JavaScript;Python;SQL;TypeScript  
343,39,C++  
344,38,Assembly;C#;HTML/CSS;JavaScript;SQL  
345,25,Java;JavaScript  
346,57,Java  
347,28,Java  
348,23,HTML/CSS;PHP  
349,54,C;C++;HTML/CSS;Python  
350,34,C;C++;Elixir;HTML/CSS;Java;Kotlin;PHP;Ruby;SQL;Other(s):  
351,54,Assembly;C;C++;Java;JavaScript;Python;SQL  
352,51,HTML/CSS;JavaScript;Ruby;TypeScript  
353,38,HTML/CSS;JavaScript;TypeScript  
354,52,HTML/CSS;Java;JavaScript;Kotlin;TypeScript  
355,42,Bash/Shell/PowerShell;Go;HTML/CSS;Java;JavaScript;Kotlin;TypeScript  
356,50,HTML/CSS;JavaScript;PHP;Python;SQL  
357,22,HTML/CSS;TypeScript  
358,47,C;C++;C#;HTML/CSS;JavaScript;PHP;SQL;TypeScript  
359,30,Java;JavaScript;Python  
360,18,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Python;TypeScript  
361,40,Elixir;Erlang;F#;Go;HTML/CSS;JavaScript;PHP;SQL  
362,44,HTML/CSS;Java;JavaScript;SQL  
363,24,Bash/Shell/PowerShell;HTML/  
CSS;Java;JavaScript;Objective-C;Python;SQL;TypeScript  
364,21,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;Python;SQL;TypeScript  
365,37,Java;SQL  
366,35,HTML/CSS;PHP;SQL  
367,35,HTML/CSS;JavaScript  
368,19,Go;HTML/CSS;JavaScript;PHP;Python;SQL;TypeScript  
369,29,Python  
370,33,R;SQL  
371,32,Bash/Shell/PowerShell;SQL  
372,55,Bash/Shell/PowerShell;C#;HTML/CSS;Java;Python;SQL

373,57,C#;HTML/CSS;Java;JavaScript;Python  
374,31,HTML/CSS;Python;SQL;TypeScript  
375,23,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;Python;SQL  
376,21,C++;Python  
377,26,HTML/CSS;JavaScript;PHP;SQL  
378,36,C#;HTML/CSS;JavaScript;SQL;TypeScript  
379,27,Bash/Shell/PowerShell;C;C#;Java;Python  
380,54,Bash/Shell/PowerShell;JavaScript;Python  
381,54,C#;Go;HTML/CSS;Java;JavaScript;Python;R;SQL  
382,46,C#;HTML/CSS;Java;JavaScript;PHP;Python;SQL  
383,38,Bash/Shell/PowerShell  
384,29,Bash/Shell/PowerShell;Go;SQL;TypeScript  
385,48,HTML/CSS;Java;JavaScript;Python;R;SQL;VBA  
386,46,HTML/CSS;JavaScript;PHP  
387,44,Java;Python  
388,39,C;HTML/CSS;Java;JavaScript;PHP;SQL;TypeScript  
389,54,Bash/Shell/PowerShell;JavaScript;Objective-C;Ruby;Swift  
390,47,Bash/Shell/PowerShell;HTML/CSS;Java;Kotlin;Python;Scala  
391,45,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;Python;SQL  
392,56,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;Python;SQL;TypeScript;VBA  
393,32,Java;Kotlin;PHP;SQL  
394,26,C#;HTML/CSS;TypeScript;Other(s):  
395,51,HTML/CSS;Java;JavaScript;SQL  
396,53,HTML/CSS;JavaScript  
397,43,HTML/CSS;JavaScript;PHP;Ruby;Rust;SQL  
398,44,Bash/Shell/PowerShell;C;C++;HTML/CSS;Java;JavaScript;Python;SQL;VBA  
399,20,C++  
400,54,Bash/Shell/PowerShell;C++;Java;Kotlin;PHP  
401,42,JavaScript;Python;Ruby;SQL;TypeScript  
402,31,C#  
403,22,Bash/Shell/PowerShell;C;C#;F#;HTML/CSS;JavaScript;PHP;Python;SQL  
404,48,C#;SQL;TypeScript  
405,26,Java;JavaScript;TypeScript  
406,51,HTML/CSS;Java;JavaScript;PHP;SQL;TypeScript  
407,18,C;C++;HTML/CSS;Java;JavaScript;Python;SQL  
408,49,Assembly;Bash/Shell/PowerShell;C;C++;Go;HTML/  
CSS;Java;JavaScript;Python;R;Rust;SQL;TypeScript;WebAssembly  
409,22,Bash/Shell/PowerShell;C#;Go;HTML/CSS;JavaScript;PHP  
410,36,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;SQL  
411,34,C;C++;JavaScript;Python;R;SQL  
412,55,HTML/CSS;JavaScript;PHP;Python;SQL  
413,57,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;SQL;TypeScript  
414,42,Bash/Shell/PowerShell;JavaScript;Python;SQL  
415,33,HTML/CSS;JavaScript;Ruby  
416,24,C;C++;C#;HTML/CSS;JavaScript;SQL;TypeScript  
417,30,HTML/CSS;JavaScript  
418,55,HTML/CSS;JavaScript;SQL;VBA  
419,55,C;C++;C#;HTML/CSS;Java;JavaScript;PHP;Python;R;SQL  
420,45,HTML/CSS;JavaScript;Python  
421,25,Bash/Shell/PowerShell;HTML/CSS;JavaScript;TypeScript  
422,29,Bash/Shell/PowerShell;C;C#;HTML/CSS;JavaScript;Python;Other(s):  
423,55,C++;C#  
424,36,C;C++

425,26,Bash/Shell/PowerShell;C;HTML/CSS;JavaScript;PHP;SQL;VBA  
426,49,Java  
427,40,C#;HTML/CSS;JavaScript;SQL;TypeScript  
428,38,Bash/Shell/PowerShell;Java;Python  
429,18,C#  
430,56,Assembly;C++;C#;HTML/CSS;Java;JavaScript;Objective-C;PHP;SQL;Other(s):  
431,40,VBA;Other(s):  
432,47,HTML/CSS;Java;JavaScript;SQL;TypeScript;VBA;Other(s):  
433,35,Bash/Shell/PowerShell;C;C++;HTML/CSS;Java;JavaScript;Python  
434,39,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;SQL  
435,51,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;Python;SQL;Swift  
436,43,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;PHP;Python;SQL  
438,42,Bash/Shell/PowerShell;C;C++;Go;HTML/CSS;JavaScript;R;SQL  
439,56,Bash/Shell/PowerShell;C#;HTML/CSS;Java;JavaScript;PHP;Python;SQL;TypeScript  
440,53,Bash/Shell/PowerShell;C++;HTML/CSS;Java;PHP;SQL  
441,34,C#;HTML/CSS;Java;JavaScript;PHP  
442,29,HTML/CSS;JavaScript  
443,37,HTML/CSS;JavaScript;TypeScript  
444,37,Bash/Shell/PowerShell;C;C++;HTML/CSS;Python  
445,55,C#;SQL  
446,56,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;Python  
447,54,Bash/Shell/PowerShell;Java;Python  
448,21,Bash/Shell/PowerShell;Java;SQL;Other(s):  
449,26,Bash/Shell/PowerShell;C#;HTML/CSS;Java;JavaScript;TypeScript;Other(s):  
450,32,C#;HTML/CSS;JavaScript;TypeScript  
451,56,Bash/Shell/PowerShell;C;C++;C#;Java;Python;Rust  
452,33,SQL;VBA  
453,56,C++;C#;SQL  
454,34,Bash/Shell/PowerShell;Elixir;HTML/CSS;JavaScript;Ruby;SQL;Other(s):  
455,36,C++;Python;Ruby  
456,39,Bash/Shell/PowerShell;JavaScript;Python;SQL  
457,36,HTML/CSS;JavaScript  
458,54,Assembly;C;Python;Rust  
459,24,C#;SQL  
460,54,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;PHP;Python  
461,21,HTML/CSS;JavaScript;PHP;SQL  
462,42,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;Python;SQL  
463,41,HTML/CSS;Java;JavaScript;PHP;SQL;Other(s):  
464,37,C;Python  
465,36,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;SQL;TypeScript  
466,43,HTML/CSS;JavaScript;TypeScript  
467,19,Java;Kotlin;Python;SQL  
468,23,Bash/Shell/PowerShell;C#;HTML/CSS;Java;JavaScript;Kotlin;SQL;Other(s):  
469,25,Bash/Shell/PowerShell;JavaScript;PHP;SQL  
470,19,HTML/CSS;Java;JavaScript;PHP;SQL;TypeScript;Other(s):  
471,33,Java;JavaScript  
472,18,HTML/CSS;JavaScript;PHP;Python;SQL  
473,22,C#;JavaScript  
474,21,Java;JavaScript;Other(s):  
475,55,C#;HTML/CSS;JavaScript;PHP;TypeScript  
476,32,HTML/CSS;JavaScript;PHP;Scala  
477,20,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;SQL;TypeScript  
478,46,C++;HTML/CSS;JavaScript;PHP;Python;Ruby;SQL

479,18,C#;HTML/CSS;JavaScript;SQL;Other(s):  
480,22,C;C++;Dart;Java;Python  
481,28,Bash/Shell/PowerShell;Kotlin  
482,25,Bash/Shell/PowerShell;HTML/CSS;PHP  
483,19,Bash/Shell/PowerShell;C#;HTML/CSS;Java;JavaScript;PHP;Python;SQL  
484,37,HTML/CSS;Java;PHP;Python;SQL  
485,40,Java;JavaScript;Scala  
486,22,Bash/Shell/PowerShell;HTML/CSS;JavaScript;TypeScript  
487,54,Bash/Shell/PowerShell;C++;C#;HTML/CSS;JavaScript;PHP;SQL;TypeScript  
488,31,C;C++;C#;Java  
489,20,C++;Python  
490,48,HTML/CSS;JavaScript;Ruby  
491,49,Java;Objective-C  
492,48,Bash/Shell/PowerShell;Go;HTML/CSS;JavaScript  
493,57,C#;Java;SQL;Other(s):  
494,55,Assembly;C;Erlang;HTML/CSS;Java;JavaScript;Python;SQL  
495,52,R  
496,33,Bash/Shell/PowerShell;Python;SQL  
497,43,Dart;JavaScript;Kotlin;PHP;SQL;Swift;TypeScript  
498,32,C++;HTML/CSS;JavaScript;Python  
499,35,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;PHP;Ruby;SQL;VBA  
500,55,C++;C#  
501,57,HTML/CSS  
502,57,C++;C#  
503,49,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;SQL;TypeScript  
504,38,Bash/Shell/PowerShell;Python;SQL  
505,31,C;C++;HTML/CSS;Java;SQL  
506,40,Java  
507,28,Go;HTML/CSS;Java;JavaScript;PHP;Python;SQL;TypeScript  
508,21,C;C++;HTML/CSS;JavaScript  
509,36,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;Python;SQL  
510,44,Python  
511,41,HTML/CSS;JavaScript;Ruby;SQL;TypeScript  
512,54,Bash/Shell/PowerShell;Python;Other(s):  
513,48,C++;Rust  
514,50,HTML/CSS;JavaScript;Python;Ruby;TypeScript  
515,33,C++;HTML/CSS;Java;JavaScript;PHP;Python;SQL  
516,39,Bash/Shell/PowerShell;C++;Python  
517,52,Assembly;C++;F#;HTML/CSS;PHP;Python;R;Rust;TypeScript;Other(s):  
518,32,C;HTML/CSS;Java;JavaScript;SQL  
519,57,HTML/CSS;PHP  
520,46,HTML/CSS;Java;JavaScript;Python;SQL;TypeScript  
521,25,HTML/CSS;JavaScript  
522,29,HTML/CSS;Python  
523,23,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Python;R  
524,30,Python;Other(s):  
525,51,C#;SQL;VBA  
526,50,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;SQL  
527,18,HTML/CSS;JavaScript;Objective-C;Swift  
528,29,Bash/Shell/PowerShell;Java;JavaScript;Python;SQL  
529,23,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;PHP;Python;SQL;TypeScript  
530,28,C;C++;C#  
531,21,C#

532,34,Bash/Shell/PowerShell;C;C++;Java;Python  
533,37,C#;Java;JavaScript;SQL  
535,44,Elixir;VBA  
536,29,C#;HTML/CSS;Java;JavaScript;PHP;Python;SQL  
537,48,C#;HTML/CSS;JavaScript;SQL;TypeScript;Other(s):  
538,51,HTML/CSS;Java;JavaScript;PHP;R;SQL;TypeScript  
539,18,Bash/Shell/PowerShell;JavaScript;Python;SQL  
540,21,JavaScript;Python  
541,20,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;Python;SQL;TypeScript;Other(s):  
542,28,Bash/Shell/PowerShell;C#;HTML/CSS;Java;JavaScript;Python;R;SQL  
543,50,C;C#;HTML/CSS;Java;JavaScript;Kotlin;PHP;TypeScript  
544,44,Assembly;Bash/Shell/PowerShell;C;C++;Go;HTML/  
CSS;JavaScript;Objective-C;PHP;Python;SQL  
545,53,Bash/Shell/PowerShell;C++;C#;Go;HTML/  
CSS;JavaScript;PHP;Python;SQL;TypeScript;Other(s):  
546,27,C;Java;SQL  
547,29,C#;HTML/CSS;JavaScript;SQL  
548,36,Python;SQL;VBA  
550,53,C#;HTML/CSS;JavaScript;SQL  
551,54,Assembly;Bash/Shell/PowerShell;C;C++;HTML/  
CSS;Java;JavaScript;Python;R;Scala;SQL  
552,19,HTML/CSS;Java;JavaScript;PHP  
553,39,C#;HTML/CSS;JavaScript;Python;SQL;TypeScript  
554,40,Go;JavaScript;PHP;Ruby;SQL;TypeScript  
555,36,C#;HTML/CSS;JavaScript;SQL  
556,21,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;Python;SQL  
557,34,C;C++;PHP;Python;SQL;VBA  
558,49,Bash/Shell/PowerShell;C;HTML/CSS;Java;JavaScript;Kotlin;PHP;SQL;TypeScript  
559,54,Bash/Shell/PowerShell;C;C++;C#;HTML/CSS;Java;PHP;Python;SQL;VBA  
560,52,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;PHP;SQL  
561,43,HTML/CSS;Java;JavaScript;Python;Ruby;SQL  
562,35,C#;JavaScript;SQL  
563,26,Assembly;Bash/Shell/PowerShell;C;C++;Python;SQL  
564,27,C;C++;Python;R;SQL;Other(s):  
565,31,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;SQL  
566,55,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript  
567,48,Bash/Shell/PowerShell;C;Java;JavaScript;Objective-C;Rust;Swift;Other(s):  
568,35,C++;C#;Clojure;HTML/CSS;JavaScript;Python;SQL  
569,26,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Python  
571,57,Assembly;Bash/Shell/PowerShell;C;C++;Erlang;R;Rust;Scala  
572,24,Assembly;C;C++;Java;JavaScript;Kotlin;PHP;Python;Rust;SQL;TypeScript  
573,40,Assembly;Bash/Shell/PowerShell;C;HTML/CSS;PHP;Python;SQL  
574,56,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;Python;Ruby  
575,37,Bash/Shell/PowerShell;Go;Python;SQL  
576,54,C#;HTML/CSS;JavaScript;Ruby;SQL;TypeScript  
577,34,Bash/Shell/PowerShell;Python;Other(s):  
578,37,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;SQL;Swift  
579,47,Bash/Shell/PowerShell;C++;C#;HTML/CSS;Java;JavaScript;Python  
580,47,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;Python;Scala;SQL  
581,50,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;TypeScript  
582,35,C;C++;C#;HTML/CSS;Java;JavaScript;PHP;Python;SQL  
583,20,Java;Ruby;Scala;SQL;Other(s):  
584,38,Java;Python

585,23,C#;Java;Python;R;VBA  
586,29,Java;Python;Ruby  
587,19,Bash/Shell/PowerShell;C#;Java;Other(s):  
588,27,C#  
589,23,C#;HTML/CSS;JavaScript;SQL;VBA  
590,26,C#;SQL  
591,40,C#;JavaScript  
592,57,C;C++;HTML/CSS;PHP;Python;SQL  
594,43,HTML/CSS;Java;JavaScript;Kotlin;PHP  
595,31,C#;HTML/CSS;JavaScript;SQL;TypeScript  
596,39,Bash/Shell/PowerShell;C;C++;HTML/CSS;Java;JavaScript;Python;SQL;TypeScript  
597,39,Java;JavaScript;PHP;Python;SQL  
598,55,Assembly;Bash/Shell/PowerShell;C;C++;C#;HTML/CSS;JavaScript;PHP;Python;SQL  
599,41,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;Ruby;SQL  
600,41,Bash/Shell/PowerShell;HTML/CSS;Java;PHP;Python;SQL  
601,18,HTML/CSS;PHP;SQL  
602,55,C++;C#;HTML/CSS;Java;JavaScript;PHP;Python;SQL;Swift;TypeScript  
603,48,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript  
604,23,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;PHP;SQL;TypeScript  
605,31,Java;Kotlin  
606,54,Bash/Shell/PowerShell;Clojure;HTML/  
CSS;Java;JavaScript;PHP;Python;Ruby;TypeScript  
607,20,C#;JavaScript;SQL  
608,23,Bash/Shell/PowerShell;C;C++;Erlang;Go;HTML/CSS;Java;Python;SQL  
609,25,HTML/CSS;JavaScript;Python  
610,54,Bash/Shell/PowerShell;C;C++;Go;HTML/  
CSS;Java;JavaScript;PHP;Python;Rust;SQL;TypeScript;Other(s):  
611,23,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;SQL  
612,56,Python;Other(s):  
613,52,Assembly;Bash/Shell/PowerShell;C;C++;C#;F#;HTML/  
CSS;JavaScript;Python;TypeScript  
614,51,Bash/Shell/PowerShell;C;C++;C#;HTML/CSS;Java;Python;SQL;TypeScript  
615,30,C#;HTML/CSS;JavaScript;Python;SQL;TypeScript;VBA  
616,27,Ruby;SQL  
617,44,Bash/Shell/PowerShell;C#;HTML/CSS;Java;JavaScript;SQL  
618,24,C;C++;C#;Java;PHP;SQL  
619,46,Assembly;Bash/Shell/PowerShell;C;C++;C#;HTML/CSS;Java;JavaScript;PHP;SQL  
620,54,HTML/CSS;Java;JavaScript;SQL;TypeScript;WebAssembly  
621,40,Bash/Shell/PowerShell;SQL;Other(s):  
622,27,Java;Objective-C;SQL;Swift  
623,49,Objective-C;Swift  
624,19,C++;HTML/CSS;JavaScript;PHP;Python  
625,50,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Ruby;TypeScript  
626,44,HTML/CSS;Java;JavaScript;SQL  
627,45,Assembly;Bash/Shell/PowerShell;C++;C#;Go;HTML/  
CSS;JavaScript;Objective-C;PHP;Python;Ruby;SQL;Swift  
628,53,Bash/Shell/PowerShell;JavaScript;SQL;TypeScript  
629,30,Assembly;Bash/Shell/PowerShell;C;C++;C#;HTML/CSS;Java;Objective-C;SQL  
630,41,C#;HTML/CSS;JavaScript;SQL;TypeScript  
631,36,Bash/Shell/PowerShell;Go;HTML/CSS;JavaScript;Python;SQL  
632,22,C#;HTML/CSS;JavaScript;SQL  
633,38,C;C++;Python;Rust  
634,40,C;C++;C#;HTML/CSS;Java;JavaScript;Objective-C;Scala;SQL;Swift;TypeScript

635,26,Bash/Shell/PowerShell;JavaScript;Python  
636,23,Bash/Shell/PowerShell;C;C++;Go;Java  
637,39,C++;C#;HTML/CSS;Java;JavaScript;SQL  
638,31,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;Rust  
639,43,HTML/CSS;JavaScript;Kotlin;SQL;Swift  
640,40,HTML/CSS;JavaScript;SQL  
641,48,C#;HTML/CSS;Java;JavaScript;Kotlin;PHP;SQL;Swift  
642,18,Bash/Shell/PowerShell;C++;HTML/CSS;JavaScript;TypeScript  
643,38,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;PHP;Ruby;SQL  
645,43,C++;C#;HTML/CSS;SQL  
646,25,JavaScript;PHP;Python;SQL;VBA  
647,48,Assembly;Bash/Shell/PowerShell;C;C++;C#;HTML/CSS;Rust;SQL;TypeScript  
648,25,C#;HTML/CSS;JavaScript;SQL;VBA  
649,19,Java;JavaScript;Python;SQL  
650,25,Bash/Shell/PowerShell;C++;JavaScript;Python  
651,38,Python  
652,22,HTML/CSS;Java  
653,37,C#;HTML/CSS;JavaScript;Kotlin;Objective-C;PHP;SQL;Swift  
654,55,Clojure;HTML/CSS;Java;JavaScript  
655,34,Assembly;C++;C#;Java;Scala;Other(s):  
656,23,C++;Python;R;Other(s):  
657,51,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;SQL  
658,23,C++;HTML/CSS;Java;JavaScript;TypeScript  
659,53,Python  
660,19,HTML/CSS;Java;JavaScript;Python;SQL;TypeScript  
661,25,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;Python;SQL  
662,48,Python;SQL;VBA  
663,25,HTML/CSS;JavaScript;Python;SQL  
664,23,HTML/CSS;Java;JavaScript;PHP  
665,28,Ruby;SQL  
666,57,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Python;SQL  
667,38,HTML/CSS;JavaScript  
668,25,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Python;SQL  
669,52,Bash/Shell/PowerShell;C;Python;Rust;SQL  
670,40,C#;HTML/CSS;JavaScript;SQL;Swift  
671,18,Bash/Shell/PowerShell;HTML/CSS;Java;JavaScript;Kotlin;PHP;Python  
672,56,C;C++;HTML/CSS;Java;JavaScript;PHP;SQL  
673,34,C;C++;C#;HTML/CSS;JavaScript;PHP;SQL  
674,24,C#;HTML/CSS;JavaScript;Python  
675,54,C#;HTML/CSS;Java;JavaScript;SQL;VBA;Other(s):  
676,57,Bash/Shell/PowerShell;Objective-C;Ruby;Swift  
677,56,Other(s):  
678,19,Java;JavaScript;Scala;SQL  
679,39,C#;HTML/CSS;Python;SQL  
680,55,C#;HTML/CSS;JavaScript;SQL  
681,38,Bash/Shell/PowerShell;C;C++;C#;F#;HTML/  
CSS;Java;JavaScript;Python;R;Ruby;Scala;SQL  
682,26,HTML/CSS;Objective-C;Swift  
683,50,HTML/CSS;JavaScript;PHP;Python;SQL  
684,56,C#;HTML/CSS;JavaScript;TypeScript  
685,53,C#;HTML/CSS;Java;SQL  
686,25,C#;SQL  
687,44,C;C++;Java;Python;SQL;Other(s):

688,38,HTML/CSS;JavaScript;PHP;Python;SQL  
689,44,C++;C#;HTML/CSS;Java;JavaScript;Other(s):  
690,53,Assembly  
691,28,HTML/CSS;JavaScript;PHP;SQL  
692,35,HTML/CSS;Java;JavaScript;SQL  
693,38,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;Python;SQL  
694,27,C#;Go;HTML/CSS;Java;JavaScript;PHP;Python;SQL;VBA  
695,56,C#;HTML/CSS;JavaScript;TypeScript  
696,21,C#;SQL  
697,33,Bash/Shell/PowerShell;HTML/CSS;JavaScript;Python  
698,29,HTML/CSS;Java;JavaScript;Python  
699,56,Bash/Shell/PowerShell;C;C++;C#;HTML/CSS;JavaScript;SQL  
700,31,HTML/CSS;JavaScript;Python;SQL;TypeScript  
701,41,Bash/Shell/PowerShell;C;C++;HTML/CSS;Java;JavaScript;Rust;SQL  
702,20,C;C++;HTML/CSS;JavaScript;PHP;Python;SQL  
703,28,Bash/Shell/PowerShell;C#;HTML/CSS;JavaScript;SQL  
704,33,Dart;HTML/CSS;Java;PHP;SQL  
705,57,Go;JavaScript  
706,33,Java;Python;Scala;SQL  
707,42,C#;HTML/CSS;JavaScript;Objective-C;PHP;Ruby;SQL  
708,52,Bash/Shell/PowerShell;C++;Go;JavaScript;Python;SQL  
709,55,JavaScript;TypeScript  
710,39,Assembly;C;HTML/CSS;Java;JavaScript;Python;R  
711,49,Java  
712,32,HTML/CSS;JavaScript  
713,44,HTML/CSS;JavaScript;PHP;SQL  
714,42,Bash/Shell/PowerShell;HTML/CSS;JavaScript;PHP;SQL;VBA  
715,20,HTML/CSS;JavaScript;PHP;SQL  
716,46,HTML/CSS;JavaScript;PHP;Other(s):  
717,54,C#;HTML/CSS;Java;PHP;SQL  
718,26,C#;HTML/CSS;JavaScr

## ✓ Poglavlje 8

### Domaći zadatak #11

#### TEKST DOMAĆEG ZADATKA #11

*Domaći zadatak #11 okvirno se radi 3h*

Data je datoteka sa podacima o studentima, studijskom programu i trajanju studija.

Ručno prebaciti podatke u CSV datoteku. Razmak može ostati TAB, ili promeniti razmak da bude zapeta.

Napisati program u Python jeziku koji će naći sledeće:

- Srednju vrednost trajanja studija na svim studijskim programima,
- Srednju vrednost trajanja studija na pojedinačnim studijskim programima (BAS; MAS; DAS)
- Najčešću vrednost trajanja studija po pojedinačnim studijskim programima,
- Najčešću vrednost trajanja studija po univerzitetima,
- Histogram broja studenata po trajanju studija, za naznačenom srednjom vrednošću,
- Histogram broja studenata po univerzitetu
- Pie grafikone raspodele studenata po studijskom programu, po univerzitetu,

Štampati grafikone odvojeno. Dokument koji se šalje uz domaći treba da izgleda kao izveštaj.

StudentID	Univerzitet	NivoStudija	StudijskiProgram	Trajanje
1	Univerzitet EDUCONS	BAS	Menadzment	5
2	Univerzitet Singidunum	DAS	Menadzment	3
3	Univerzitet Singidunum	DAS	Informacione Tehnologije	3
4	Univerzitet Metropolitan	DAS	Softversko inzenjerstvo	4
5	Univerzitet EDUCONS	BAS	Softversko inzenjerstvo	5
6	Univerzitet u Nisu	BAS	Menadzment	4
7	Univerzitet Singidunum	BAS	Informacione Tehnologije	6
8	Univerzitet Singidunum	BAS	Ostalo	4
9	Univerzitet Singidunum	DAS	Ostalo	3
10	Univerzitet Singidunum	MAS	Informacione Tehnologije	2
11	Univerzitet Singidunum	MAS	Ostalo	3
12	Univerzitet u Kragujevcu	BAS	Softversko inzenjerstvo	5
13	Univerzitet Singidunum	BAS	Ostalo	4
14	Univerzitet EDUCONS	BAS	Menadzment	5
15	Univerzitet u Nisu	MAS	Softversko inzenjerstvo	2
16	Univerzitet Singidunum	BAS	Menadzment	5
17	Univerzitet Singidunum	MAS	Informacione Tehnologije	1
18	Univerzitet u Kragujevcu	BAS	Softversko inzenjerstvo	5
19	Univerzitet u Beogradu	BAS	Informacione Tehnologije	6

- 20 Univerzitet Singidunum DAS Menadzment 4
- 21 Univerzitet u Beogradu MAS Menadzment 5
- 22 Univerzitet Singidunum BAS Softversko inzenjerstvo 4
- 23 Univerzitet u Beogradu MAS Ostalo 1
- 24 Univerzitet EDUCONS BAS Ostalo 6
- 25 Univerzitet EDUCONS DAS Softversko inzenjerstvo 4
- 26 Univerzitet Singidunum DAS Ostalo 4
- 27 Univerzitet u Beogradu BAS Menadzment 6
- 28 Univerzitet u Kragujevcu BAS Ostalo 6
- 29 Univerzitet Metropolitan DAS Softversko inzenjerstvo 5
- 30 Univerzitet u Nisu BAS Softversko inzenjerstvo 5
- 31 Univerzitet u Nisu MAS Softversko inzenjerstvo 1
- 32 Univerzitet Singidunum MAS Informacione Tehnologije 2
- 33 Univerzitet u Kragujevcu DAS Softversko inzenjerstvo 6
- 34 Univerzitet Singidunum BAS Informacione Tehnologije 4
- 35 Univerzitet EDUCONS MAS Softversko inzenjerstvo 3
- 36 Univerzitet u Kragujevcu BAS Menadzment 4
- 37 Univerzitet u Kragujevcu DAS Softversko inzenjerstvo 5
- 38 Univerzitet Metropolitan BAS Informacione Tehnologije 5
- 39 Univerzitet Singidunum BAS Informacione Tehnologije 6
- 40 Univerzitet Metropolitan BAS Menadzment 5
- 41 Univerzitet Metropolitan BAS Informacione Tehnologije 6
- 42 Univerzitet Singidunum BAS Informacione Tehnologije 5
- 43 Univerzitet u Nisu MAS Informacione Tehnologije 2
- 44 Univerzitet Singidunum MAS Softversko inzenjerstvo 1
- 45 Univerzitet u Beogradu MAS Softversko inzenjerstvo 1
- 46 Univerzitet u Beogradu BAS Softversko inzenjerstvo 4
- 47 Univerzitet u Nisu DAS Softversko inzenjerstvo 4
- 48 Univerzitet u Beogradu BAS Softversko inzenjerstvo 4
- 49 Univerzitet u Nisu BAS Menadzment 6
- 50 Univerzitet Metropolitan BAS Softversko inzenjerstvo 4
- 51 Univerzitet Singidunum MAS Softversko inzenjerstvo 1
- 52 Univerzitet u Nisu BAS Ostalo 5
- 53 Univerzitet EDUCONS BAS Informacione Tehnologije 6
- 54 Univerzitet EDUCONS BAS Menadzment 5
- 55 Univerzitet Metropolitan BAS Informacione Tehnologije 4
- 56 Univerzitet EDUCONS MAS Informacione Tehnologije 1
- 57 Univerzitet EDUCONS BAS Ostalo 5
- 58 Univerzitet u Beogradu BAS Ostalo 4
- 59 Univerzitet Metropolitan MAS Softversko inzenjerstvo 1
- 60 Univerzitet Singidunum BAS Ostalo 6
- 61 Univerzitet u Kragujevcu BAS Softversko inzenjerstvo 5
- 62 Univerzitet EDUCONS BAS Menadzment 4
- 63 Univerzitet u Kragujevcu BAS Menadzment 4
- 64 Univerzitet Singidunum BAS Informacione Tehnologije 4
- 65 Univerzitet Metropolitan BAS Ostalo 5
- 66 Univerzitet Metropolitan MAS Softversko inzenjerstvo 1
- 67 Univerzitet Singidunum BAS Informacione Tehnologije 5
- 68 Univerzitet Singidunum BAS Menadzment 5
- 69 Univerzitet u Nisu BAS Menadzment 5
- 70 Univerzitet Singidunum BAS Informacione Tehnologije 5
- 71 Univerzitet Singidunum DAS Informacione Tehnologije 5
- 72 Univerzitet EDUCONS DAS Ostalo 6

73	Univerzitet Singidunum	BAS Informacione Tehnologije	4
74	Univerzitet EDUCONS	DAS Ostalo	4
75	Univerzitet u Kragujevcu	BAS Informacione Tehnologije	4
76	Univerzitet u Beogradu	BAS Informacione Tehnologije	5
77	Univerzitet Metropolitan	DAS Informacione Tehnologije	6
78	Univerzitet u Beogradu	MAS Softversko inzenjerstvo	1
79	Univerzitet Singidunum	DAS Softversko inzenjerstvo	5
80	Univerzitet Metropolitan	BAS Softversko inzenjerstvo	5
81	Univerzitet EDUCONS	BAS Informacione Tehnologije	4
82	Univerzitet u Nisu	BAS Ostalo	6
83	Univerzitet u Kragujevcu	BAS Menadzment	5
84	Univerzitet Singidunum	BAS Ostalo	4
85	Univerzitet u Kragujevcu	BAS Softversko inzenjerstvo	5
86	Univerzitet Singidunum	MAS Ostalo	2
87	Univerzitet Metropolitan	BAS Menadzment	5
88	Univerzitet Metropolitan	BAS Ostalo	5
89	Univerzitet Singidunum	BAS Ostalo	6
90	Univerzitet EDUCONS	BAS Ostalo	4
91	Univerzitet Singidunum	BAS Softversko inzenjerstvo	5
92	Univerzitet u Kragujevcu	DAS Menadzment	5
93	Univerzitet Singidunum	BAS Informacione Tehnologije	5
94	Univerzitet u Kragujevcu	MAS Ostalo	3
95	Univerzitet Singidunum	BAS Ostalo	4
96	Univerzitet u Kragujevcu	BAS Ostalo	4
97	Univerzitet Metropolitan	DAS Softversko inzenjerstvo	5
98	Univerzitet Singidunum	BAS Softversko inzenjerstvo	6
99	Univerzitet u Nisu	BAS Informacione Tehnologije	6
100	Univerzitet u Beogradu	BAS Ostalo	6
101	Univerzitet u Kragujevcu	BAS Ostalo	5
102	Univerzitet u Beogradu	BAS Informacione Tehnologije	5
103	Univerzitet Singidunum	BAS Softversko inzenjerstvo	5
104	Univerzitet Singidunum	BAS Menadzment	4
105	Univerzitet EDUCONS	DAS Menadzment	4
106	Univerzitet Singidunum	BAS Ostalo	4
107	Univerzitet Metropolitan	BAS Informacione Tehnologije	4
108	Univerzitet Singidunum	BAS Menadzment	5
109	Univerzitet u Kragujevcu	BAS Menadzment	6
110	Univerzitet Metropolitan	BAS Ostalo	5
111	Univerzitet Singidunum	MAS Menadzment	3
112	Univerzitet u Kragujevcu	BAS Menadzment	4
113	Univerzitet u Beogradu	BAS Ostalo	4
114	Univerzitet Metropolitan	MAS Informacione Tehnologije	3
115	Univerzitet Singidunum	DAS Ostalo	5
116	Univerzitet u Nisu	BAS Informacione Tehnologije	4
117	Univerzitet Singidunum	BAS Menadzment	5
118	Univerzitet EDUCONS	MAS Softversko inzenjerstvo	2
119	Univerzitet Singidunum	BAS Softversko inzenjerstvo	6
120	Univerzitet Singidunum	BAS Ostalo	4
121	Univerzitet u Kragujevcu	BAS Softversko inzenjerstvo	4
122	Univerzitet Singidunum	BAS Informacione Tehnologije	6
123	Univerzitet u Beogradu	BAS Informacione Tehnologije	5
124	Univerzitet u Kragujevcu	BAS Softversko inzenjerstvo	6
125	Univerzitet Metropolitan	BAS Softversko inzenjerstvo	4

126	Univerzitet EDUCONS BAS	Informacione Tehnologije	5
127	Univerzitet Singidunum	MAS Informacione Tehnologije	2
128	Univerzitet u Kragujevcu	BAS Menadzment	4
129	Univerzitet Singidunum	MAS Softversko inzenjerstvo	3
130	Univerzitet Singidunum	MAS Informacione Tehnologije	1
131	Univerzitet Metropolitan	DAS Ostalo	6
132	Univerzitet u Kragujevcu	DAS Menadzment	5
133	Univerzitet Singidunum	BAS Ostalo	5
134	Univerzitet Singidunum	BAS Ostalo	6
135	Univerzitet Singidunum	BAS Informacione Tehnologije	6
136	Univerzitet Singidunum	BAS Softversko inzenjerstvo	5
137	Univerzitet u Nisu	BAS Informacione Tehnologije	4
138	Univerzitet Singidunum	DAS Informacione Tehnologije	6
139	Univerzitet u Kragujevcu	BAS Informacione Tehnologije	4
140	Univerzitet Singidunum	BAS Menadzment	5
141	Univerzitet Metropolitan	MAS Softversko inzenjerstvo	2
142	Univerzitet u Kragujevcu	MAS Informacione Tehnologije	3
143	Univerzitet Singidunum	BAS Ostalo	6
144	Univerzitet u Beogradu	DAS Menadzment	3
145	Univerzitet u Nisu	BAS Softversko inzenjerstvo	5
146	Univerzitet EDUCONS BAS	Softversko inzenjerstvo	5
147	Univerzitet Singidunum	BAS Softversko inzenjerstvo	4
148	Univerzitet EDUCONS BAS	Softversko inzenjerstvo	5
149	Univerzitet u Beogradu	MAS Softversko inzenjerstvo	2
150	Univerzitet Singidunum	DAS Menadzment	3

## PREDAJA DOMAĆEG ZADATKA

*Domaći zadaci se predaju 7 dana nakon predavanja za 100% poena (kod tradicionalnih studenata).*

### **Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije 7 dana nakon predavanja, za 100% poena. Nakon toga poeni se umanjuju za 50%.

### **Internet studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganje ispita. Domaći zadaci se brane!

Domaći zadatak poslati dr Nemanji Zdravkoviću: nemanja.zdravkovic@metropolitan.ac.rs

Obavezno koristiti uputstvo za izradu domaćeg zadatka.

Uz .doc dokument (koji treba sadržati i screenshot svakog urađenog zadatka kao i komentare za zadatak), poslati i izvorne i dodatne datoteke.

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Rezime lekcije #11*

#### **Rezime:**

U ovoj lekciji bilo je reči o primeni Python programskog jezika u oblasti Data Science. Bilo je reči o prednostima Python jezika u odnosu na druge jezike (neki se i plaćaju), kao i poređenje sa jezikom R.

Nakon toga, obređene su najčešće pozicije, analitičar, naučnik i inženjer podataka, koje su jako tražene.

Obrađene su osnovne statističke veličine sa primerima pomoću kojih se lako mogu izvući zaključci o velikoj količini podataka.

#### **Literatura:**

- David Beazley, Brian Jones, *Python Cookbook: Recipes for Mastering Python 3*, 3rd edition, O'Reilly Press, 2013.
- Mark Lutz, *Learning Python*, 5th Edition, O'Reilly Press, 2013.
- Andrew Bird, Lau Cher Han, et. al, *The Python Workshop*, Packt Publishing, 2019.
- Al Sweigart, *Automate the boring stuff with Python*, 2nd Edition, No Starch Press, 2020.





CS324 - SKRIPTING JEZICI

Python i mašinsko učenje

Lekcija 12

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 12

### *PYTHON I MAŠINSKO UČENJE*

- ✓ Python i mašinsko učenje
- ✓ Poglavlje 1: Uvod u veštačku inteligenciju
- ✓ Poglavlje 2: Algoritmi mašinskog učenja
- ✓ Poglavlje 3: Linearna regresija
- ✓ Poglavlje 4: Logistička regresija
- ✓ Poglavlje 5: Paket scikit-learn
- ✓ Poglavlje 6: Pokazne vežbe #12
- ✓ Poglavlje 7: Individualne vežbe #12
- ✓ Poglavlje 8: Domaći zadatak
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ❖ Uvod

# UVOD

### *Uvod u lekciju #12*

U dvanaestoj lekciji biće reči o korišćenju Python programskog jezika u algoritmima veštačke inteligencije, konkretno za mašinsko učenje.

Python kao programski jezik se dobro pokazao kada je u pitanju manipulacija nad podacima, i upravo je jedna od primena rada sa velikom količinom podataka veštačka inteligencija, sa naglaskom na mašinsko učenje, kao oblast veštačke inteligencije koja je u velikom usponu.

Veštačka inteligencija i mašinsko učenje se uče na posebnim predmetima, ali u ovoj lekciji uvode se osnovni pojmovi mašinskog učenja kao i najjednostavniji algoritmi.

U delu lekcije koji se odnosi na programski jezik Python, biće reči o **scikit-learn** paketu za veštačko učenje i modul **linear\_model**, za dva algoritma koja će se obrađivati.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 1

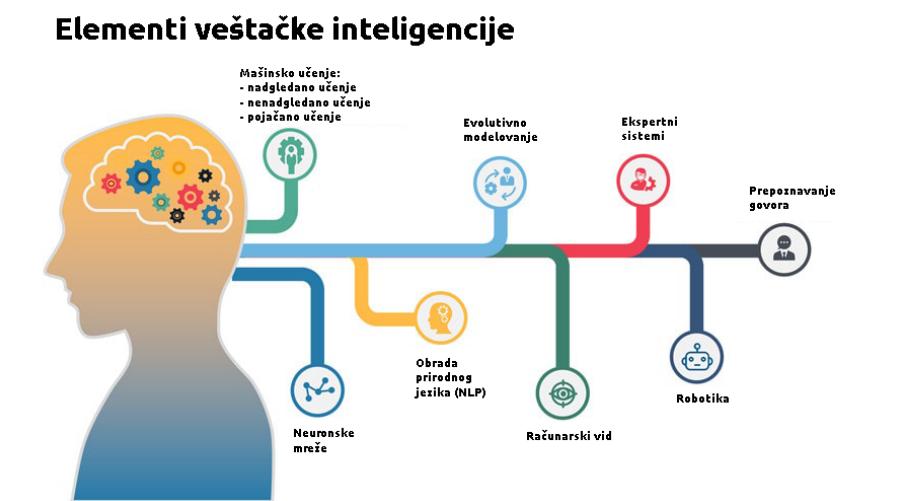
# Uvod u veštačku inteligenciju

## POJAM VEŠTAČKE INTELIGENCIJE

*Veštačka inteligencija se koristi za konstruisanje inteligencije upotrebom hardverskih i softverskih rešenja.*

Veštačka inteligencija (en. **Artificial Intelligence**, AI) je nauka koja se koristi za konstruisanje inteligencije upotrebom hardverskih i softverskih rešenja.

Veštačka inteligencija automatizuje ljudsku inteligenciju na osnovu načina na koji ljudski mozak obrađuje informacije



Slika 1.1 Elementi veštačke inteligencije. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## DISCIPLINE KOJE UKLJUČUJE AI

*Eksperti u domenu veštačke inteligencije trebaju znati koje su discipline pridružene veštačkoj inteligenciji.*

Za kreiranje dobrog ali i upotrebljivo AI rešenja, potrebno je uključiti različite discipline.

- **robotika** - za pomeranje objekata u prostoru,
- **teorija algoritama** - za konstruisanje efikasnih algoritama,

- **statistika** - za izvođenje korisnih rezultata, predviđanje budućnosti i analizu prošlosti,
- **psihologija** - za modelovanje načina funkcionisanja ljudskog mozga
- **softversko inženjerstvo** - za kreiranje rešenja koja se mogu održavati i koja mogu izdržati test vremena
- **računarska nauka** - za implementaciju softverskih rešenja u praksi
- **matematika** - za izvršavanje složenih matematičkih operacija
- **upravljanje sistema** - za kreiranje feed-forward i feedback sistema
- **teorija informacija** - za predstavljanje, kodiranje, dekodiranje i kompresovanje informacija
- **teorija grafova** - za modelovanje i optimizovanje različitih tačaka u prostoru i za predstavljanje hijerarhija
- **fizika** - za modelovanje realnog sveta
- **računarska grafika i obrada slika** - za prikaz i obradu slika i filmova

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PRIMER: KAKO AI REŠAVA PROBLEME IZ REALNOG SVETA?

*Veštačka inteligencija automatizuje ljudsku inteligenciju na osnovu načina na koji ljudski mozak obrađuje informacije.*

**Veštačka inteligencija automatizuje ljudsku inteligenciju na osnovu načina na koji ljudski mozak obrađuje informacije.**

Kad god rešimo problem ili komuniciramo sa ljudima, prolazimo kroz proces. Kad god ograničimo opseg problema ili interakcije, ovaj proces često može da bude modelovan i automatizovan.

**AI izvršavaju računari koji izvršavaju instrukcije nižeg nivoa.**

Većina AI aplikacija ima jedan primarni cilj. Kada komuniciramo sa AI aplikacijom, čini se kao da je ljudska, jer može da ograniči domen problema na primarni cilj. Prema tome, možemo da razdvojimo složene procese i simuliramo inteligenciju pomoću računarskih instrukcija nižeg nivoa.

**AI može da stimuliše ljudska čula i procese razmišljanja za specijalizovane oblasti.** Treba da simuliramo ljudska čula i razmišljanje i ponekad da prevarimo AI da veruje da komuniciramo sa drugim ljudskim bićem. U posebnim slučajevima možemo čak da poboljšamo i naša čula.

Slično tome, kada komuniciramo sa chatbotom, očekujemo da nas bot razume. Očekujemo da chatbot ili čak sistem za prepoznavanje govora obezbede interfejs između računara i čoveka. Da bi ispunili naša očekivanja, računari treba da simuliraju procese ljudskog razmišljanja.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PRIMENA AI: SINTEZA GOVORA

*Prepoznavanje govora omogućuje razumevanje ljudskog govora od strane mašine. Sinteza govora omogućuje mašini da "priča".*

### Simulacija ljudskog ponašanja

Ljudi imaju pet osnovnih čula: vida, sluha, dodira, mirisa i ukusa. Međutim, da bismo bolje razumeli kako da kreiramo inteligentne mašine, možemo da razdvojimo discipline na sledeći način:

- slušanje i govor,
- razumevanje jezika,
- pamćenje,
- razmišljanje,
- vid,
- pokret

Na primer, kada želimo da pomerimo robotsku ruku, treba da poznajemo složenu matematiku visokog nivoa da bismo razumeli šta se dešava.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### Slušanje i govor

Upotrebom sistema za prepoznavanje govora AI može da sakuplja informacije. Upotrebom sinteze govora može da pretvori interne podatke u razumljive zvukove. Tehnike prepoznavanja govora i sinteze govora prepoznaju i konstruišu zvukove koje ljudi emituju ili koje ljudi mogu da razumeju.

Zamislite da ste na putovanju u državi u kojoj se govori jezikom koji vi ne razumete. Možete da govorite u mikrofon svog telefona, očekujući da telefon „razume“ ono što vi kažete, a zatim da to prevede na drugi jezik. Isto se dešava i obratno - kada lokalni stanovnik govorи, a AI prevodi zvukove na jezik koji razumete. Prepoznavanje govora i sinteza govora to omogućavaju.

**Primer sinteze govora je Google Translate. Dugme sa zvučnikom ispod prevedene reči da bi prevodilac glasno izgovorio reči na jeziku koji nije engliski.**

## PRIMENA AI: NLP I NN

*Duboko učenje predstavlja "crnu kutiju" jer prevazilazi obrazac kategorizacije.*

### Razumevanje jezika

Možemo da razumemo prirodni jezik ako ga obradimo. Ova oblast se naziva obrada prirodnog jezika (en. **natural language processing**, NLP). Kada je reč o obradi prirodnog jezika, želimo da naučimo jezike na osnovu statističkog učenja.

### Pamćenje

Treba svetu da predstavimo nešto što poznajemo. Za to kreiramo baze znanja i hijerarhijske reprezentacije pod nazivom ontologije. Ontologije kategoriju elemente i ideje u našem svetu i sadrže veze između ovih kategorija.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### Razmišljanje

Al sistem treba da bude ekspert u određenom domenu upotrebom ekspertskega sistema, koji može da bude zasnovan na matematičkoj logici na deterministički način, kao i na nejasan, nedeterministički način.

Baza znanja ekspertskega sistema je predstavljena upotrebom različitih tehnika. Kako problem domena raste, kreiramo hijerarhijske ontologije.

Možemo da repliciramo ovu strukturu modelovanjem mreže na gradivnim blokovima mozga.

Ovi gradivni blokovi se nazivaju *neuroni*, a sama mreža se naziva **neuronska mreža** (en. **neural network**, NN).

Postoji još jedan ključni termin koji treba da povežemo sa neuronskim mrežama - **duboko učenje** (en. **deep learning**).

Ovakvo učenje se naziva duboko jer prevazilazi obrazac prepoznavanja i kategorizacije. Ono je utisnuto u neuronsku strukturu mreže. Na primer, specijalni zadatak dubokog učenja je prepoznavanje objekta pomoću računarskog vida.

## PRIMENA AI: RAČUNARSKI VID I ROBOTIKA

*Robotika se bavi složenim funkcijama omogućavanje pokreta mašina. Računarski vid omogućava mašini da prepozna šta se nalazi na slici ili na video sadržaju.*

### Vid

Treba da komuniciramo sa realnim svetom pomoću naših čula. Do sada smo govorili samo o čulu sluha, o prepoznavanju i sintezi govora. Šta se dešava ako treba da vidimo nešto? U tom

slučaju treba da kreiramo tehnike računarskog vida za učenje o okruženju. Prepoznavanje lica je korisno i većinom su ljudi u tome eksperti.

**Računarski vid** (en. **computer vision**) zavisi od obrade slike. Iako obrada slike nije direktno AI disciplina, ona je potrebna disciplina za AI.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

### **Pokret**

Pokret i dodir su prirodni za nas ljudi, ali su veoma složeni zadaci za računare. Pokret se vrši pomoću robotike. Robotika je zasnovana na upravljanju sistemima, gde kreiramo povratnu petlju i kontrolišemo pokret objekta na osnovu sakupljenih povratnih informacija.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## **MAŠINSKO UČENJE**

*Mašinsko učenje predstavlja polje znanja koje daje računarima mogućnost da uče bez eksplicitnog programiranja.*

Mašinsko učenje (en. **Machine Learning**, ML), pored obrade prirodnog jezika (en. **Natural Language Processing**, NLP), računarske vizije (en. **Computer Vision**, CV) i ekspertskega sistema (en. **Expert Systems**, ES), je pod oblast veštačke inteligencije koja mašinama (računarima) pruža mogućnost učenja bez eksplicitnog programiranja.

### **Definicija mašinskog učenja**

*Arthur Samuel (1959):*

Mašinsko učenje predstavlja polje znanja koje daje računarima mogućnost da uče bez eksplicitnog programiranja.

*Tom Mitchell (1998):*

Za računarski program kaže se da uči da rešava određeni zadatak T, iz iskustva E, i merom performansi P, ukoliko se njegove performanse rešavanja zadatka T, mereno prema P, poboljšavaju kroz iskustvo E.

Uopšteno posmatrano, proces mašinskog učenja sadrži šest koraka:

1. Prikupljanje podataka,
2. Pre-procesiranje, „čišćenje“ i formatiranje podataka,
3. Podela podataka na skup podataka za treniranje i za testiranje,
4. Kreiranje i treniranje modela mašinskog učenja nad skupom podataka za treniranje,
5. Testiranje istreniranog modela korišćenjem skupa podataka za testiranje, prilikom čega se dobijaju različite metrike koje pokazuju koliko je model „dobar“,
6. Unapređivanje i isporuka modela.

U profesionalnom okruženju, u trećem koraku se najčešće inicijalni skup podataka deli na tri, a ne na dva podskupa: treniranje, validacija i testiranje. Naime, skup podataka za treniranje se koristi za *fitovanje*, odnosno treniranje parametara modela. Potom, skup podataka za validaciju se koristi za dobijanje metrika koje možemo iskoristiti za unapređivanje modela podešavanjem njegovih parametara. Konačno, skup podataka za testiranje se koristi za dobijanje finalnih metrika performansi modela.

## ▼ Poglavlje 2

# Algoritmi mašinskog učenja

## RAZLIČITI ALGORITMI MAŠINSKOG UČENJA

*Mašinsko učenje deli se na nadgledano, nenadgledano i pojačano učenje*

Algoritmi mašinskog učenja se mogu podeliti u tri osnovne grupe:

- Nadgledano učenje (en. *Supervised learning*),
- Nenadgledano učenje (en. *Unsupervised learning*),
- Pojačano učenje (en. *Reinforcement learning*)

U nastavku ovog poglavlja biće ukratko objašnjene ove tri osnovne grupe algoritama.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## NADGLEDANO UČENJE

*Kod nadgledanog učenja, algoritmu se pružaju ulazni i izlazni istorijski podaci iz kojih algoritam uči kako da od ulaznih podataka „proizvede“ izlazne podatke.*

Kod nadgledanog učenja, algoritmu se pružaju ulazni i izlazni istorijski podaci iz kojih algoritam uči kako da od ulaznih podataka „proizvede“ izlazne podatke.

Skup podataka za treniranje modela se sastoji od niza primera; svaki primer sadrži jednu osobinu ili više njih, kao i željeni izlaz. Samim tim, najčešći format skupa podataka jeste matrica sastavljena od niza vektora koji reprezentuju primere.

Kroz iterativnu optimizaciju, algoritmi nadgledanog učenja formiraju funkciju koja se može koristiti za predviđanje rezultata primera koji nisu bili korišćeni za treniranje modela.

Kako izlazi primera mogu biti kontinualne i diskrete vrednosti, tako delimo i algoritme nadgledanog mašinskog učenja u algoritme.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ALGORITMI REGRESIJE

*Ovi algoritmi pokušavaju da mapiraju ulazne podatke u kontinualne funkcije.*

### Algoritmi regresije

Ovi algoritmi koji pokušavaju da mapiraju ulazne podatke u kontinualne funkcije koje, samim tim, proizvode vrednosti kontinualnog tipa (npr. predviđanje cena nekretnine, predviđanje broja godina, i slično).

Najpoznatiji algoritmi regresije su:

- **Linearna regresija** (en. [Linear Regression](#)),
- **Višestruka linearna regresija** (en. [Multiple Linear Regression](#)),
- **Polinomna regresija** (en. [Polynomial Regression](#)).

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ALGORITMI KLASIFIKACIJE

*Ovi algoritmi pokušavaju da mapiraju ulazne podatke u diskretne kategorije.*

### Algoritmi klasifikacije

Ovi algoritmi pokušavaju da mapiraju ulazne podatke u diskretne kategorije (npr. predviđanje da li je tumor maligni ili benigni, kojoj kategoriji pripada novinski članak, koja je vrsta životinje prikazana na slici i slično). U zavisnosti od broja mogućih kategorija izlaza, klasifikacija može biti binarna i višeklasna.

Najpoznatiji algoritmi klasifikacije su:

- **Logistička regresija** (en. [Logistic Regression](#)),
- **Stablo odlučivanja** (en. [Decision Tree](#)),
- **Slučajne šume** (en. [Random Forest](#)),
- **Naivni Bajes** (en. [Naive Bayes](#)),
- **K-najbližih suseda** (en. [K-Nearest Neighbors](#)),
- **Metoda potpornih vektora** (en. [Support Vector Machine](#)).

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## NENADGLEDANO UČENJE

*Kod nenadgledanog učenja, algoritmu se pružaju samo ulazni podaci primera koje treba obraditi.*

Kod nenadgledanog učenja, algoritmu se pružaju samo ulazni podaci primera koje treba obraditi, ne i odgovarajući izlazni podaci, te najčešće ne postoji bilo kakva povratna informacija o tome kako treba da izgledaju izlazni podaci.

Algoritam ima za cilj otkrivanje šablonu i grupisanje tih neoznačenih podataka, kako bi otkrio neku vrstu strukture istih.

### Grupisanje

Grupisanje (en. **Clustering**), metoda za grupisanje u skupu podataka na osnovu nezavisnih varijabli (npr. analiza osoba na društvenim mrežama, analiza potrošača, grupisanje gena koji su slični ili povezani različitim varijablama). Najpoznatiji algoritam iz ove klase je algoritam **K-srednje vrednosti** (en. **K-Means**).

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## POJAČANO UČENJE

*Pojačano učenje je model mašinskog učenja koji se bavi načinom na koji softverski agenti treba da preduzmu akcije u okruženju u kome se nalaze kako bi maksimizovali nagradu.*

Pojačano učenje je model mašinskog učenja koji se bavi načinom na koji softverski agenti treba da preduzmu akcije u okruženju u kome se nalaze kako bi maksimizovali nagradu.

Pojačano učenje „vuče korene“ iz psihologije ponašanja, te se često kao primer u toku objašnjavanja pojačanog učenja uzima dresiranje pasa. Naime, kad psu kažemo šta treba da uradi, verovatno nas neće odmah razumeti, ali metodom nagrađivanja i kažnjavanja pas polako uči šta je uradio dobro odnosno loše. Primenom iste metode možemo istrenirati programe da izvršavaju različite zadatke, igraju igrice i slično.

Najpoznatiji algoritmi pojačanog učenja su **Markovljev proces donošenja odluke** (en. **Markov Decision Process**) i **Kju-učenje** (en. **Q-Learning**).

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## OPTIMIZACIJA ALGORITAMA

*Jednom istrenirani algoritam mašinskog učenja može se optimizovati korišćenjem različitih tehnika.*

Jednom istrenirani algoritam mašinskog učenja može se optimizovati korišćenjem različitih tehnika.

**Smanjivanje dimenzionalnosti** (en. **Dimensionality Reduction**), metoda za smanjivanje broja nezavisnih varijabli prilikom treniranja u tzv. glavne komponente koje sažeto prenose slične informacije. Svrha smanjivanja dimenzionalnosti se ogleda u ubrzavanju ostalih algoritama mašinskog učenja, jer je tako potreban manji broj računanja; kao i pronalaženje grupe najpouzdanih karakteristika odnosno nezavisnih varijabli u podacima.

Najpoznatiji algoritam za smanjivanje dimenzionalnosti jeste tzv. **analiza osnovnih komponenti** (en. **Principal Component Analysis**, PCA), koja koristi operacije linearne algebre nad matricama podataka da izračuna projekciju originalnih podataka na isti (ili poželjno manji) broj dimenzija.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 3

# Linearna regresija

## POJAM LINEARNE REGRESIJE

*Linearna regresija hipotezu  $h$  mapira  $x$  na linearu funkciju.*

Linearna regresija predstavlja osnovni algoritam nadgledanog učenja.

Kod linearne regresije, na osnovu već postojećih ulaznih i izlaznih podataka, algoritam treba da predvidi izlaz za nove podatke koje su u opštem slučaju realne vrednosti.

Kao kod svih algoritama nadgledanog učenja, postoji skup podataka (en. **dataset**) koji se naziva **trening skup** (en. **training set**)

Unutar trening skupa postoji ukupno **m** trening primera, a svaki od primera sadrži **x** ulaznih promenljiva, i **y** izlaznih promenljiva.

Jedan trening primer predstavlja skup:

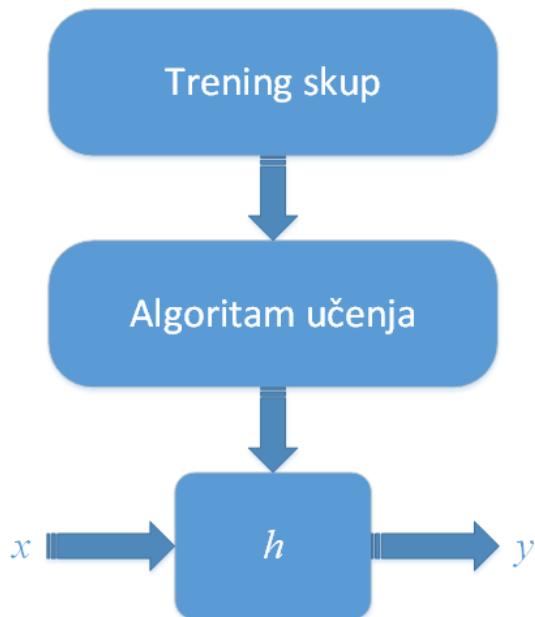
$$(x, y)$$

Trening primer na **i**-toj poziciji (**i**-ti indeks):

$$\left( x^{(i)}, y^{(i)} \right)$$

Trening skup ubacujemo u algoritam mašinskog učenja. Zadatak algoritma učenja jeste da vrati funkciju  $h$  (hipotezu), koja predstavlja funkciju koja mapira ulazne promenljive **x** na **estimiranu (predviđenu) vrednost y**.

**Funkcija  $h$  mapira  $x$  na  $y$ .**



Slika 3.1 Proces nadgledanog učenja. [Izvor: Autor]

Za **linearu regresiju**, funkcija  $h$  mapira  $x$  na linearu funkciju:

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

Veličine  $\theta_0$  i  $\theta_1$  predstavljaju parametre hipoteze.

Kada  $x$  ima samo jednu dimenziju, onda je u pitanju *linearna regresija sa jednom promenljivom* ili univarijabilna linearna regresija (en. *univariate linear regression*)

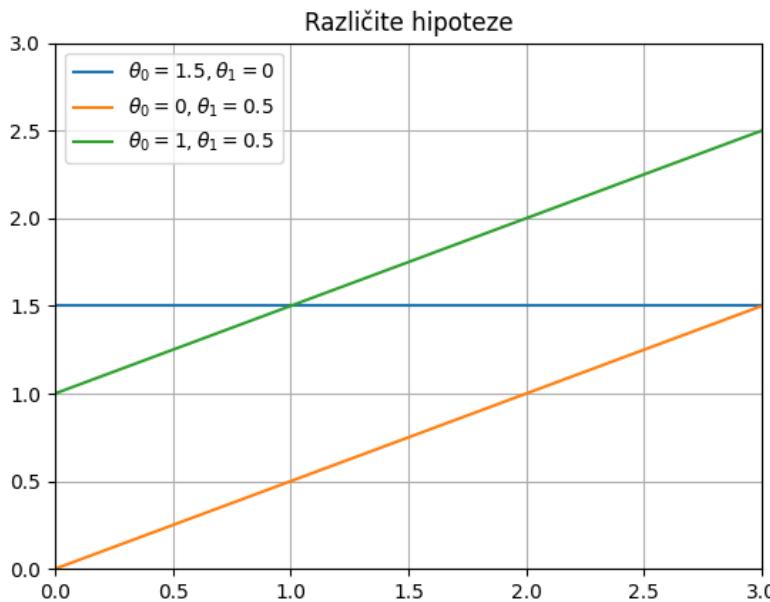
## FUNKCIJA GREŠKE

*Funkcija greške predstavlja kvadriranu razliku estimiranih i pravih vrednosti izlaznih promenljivih.*

Data je hipoteza  $h(x)$ :

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

Sa različitim vrednostima parametara  $\theta_0$  i  $\theta_1$  hipoteza će različito izgledati.



Slika 3.2 Različite vrednosti hipoteze za različite parametre  $\theta$ . [Izvor: Autor]

U linearnoj regresiji cilj jeste pronaći vrednosti za parametre  $\theta_0$  i  $\theta_1$  koji će najbolje opisati trening set kroz linearu funkciju. Ovo znači da estimirane vrednosti  $h_{\theta}(x)$  treba da budu što bliže pravim vrednostima  $y$ , za sve vrednosti trening skupa  $(x, y)$ .

Matematički gledano, treba minimizovati razliku između estimirane vrednosti i prave vrednosti, u odnosu na parametre  $\theta$ .

$$\begin{aligned} & \underset{\theta_0, \theta_1}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \\ & \underset{\theta_0, \theta_1}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2 \end{aligned}$$

Definiše se funkcija greške (en. cost function)  $J(\theta_0, \theta_1)$  koja predstavlja ovu razliku, i nju treba minimizovati. Ova funkcija zove se i funkcija srednjekvadratne greške (en. mean-squared error function).

$$\begin{aligned} J(\theta_0, \theta_1) &= \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \\ & \underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1) \end{aligned}$$

## ALGORITAM OPADAJUĆEG GRADIJENTA

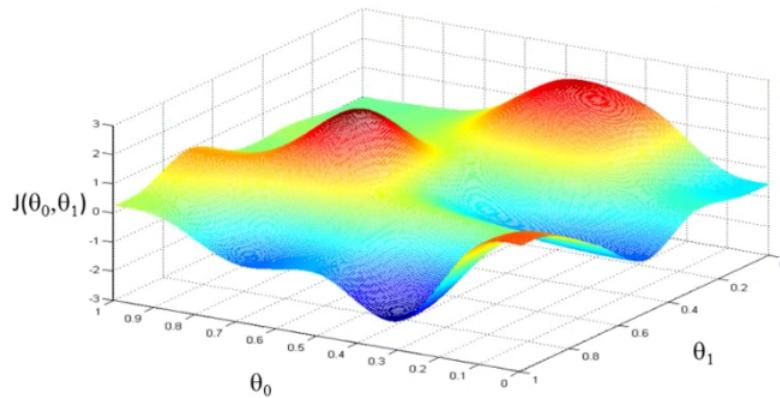
*Najčešće korišćen algoritam za linearne modele mašinskog učenja jeste opadajući gradijent.*

Jedan od najčešće korišćenih algoritama za minimiziranje funkcije greške jeste algoritam opadajućeg gradijenta (en. *gradient descent algorithm*).

Koraci algoritma jesu sledeći

- Poči sa nekim početnim vrednostima parametra  $\theta_0$  i  $\theta_1$ ,
- Menjati parametre da se smanji funkcija greške  $J$  dok se ne dođe do minimuma.

Na slici se mogu videti vrednosti funkcije  $J$  za različite vrednosti parametara. Algoritam iterativno radi dok ne dođe minimuma funkcije.



Slika 3.3 Prikaz odnosa funkcije greške i parametara. [Izvor: coursera.org]

Matematički, algoritam opadajućeg gradijenta radi sledeće:

ponoviti do konvergencije :

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1), \quad \text{za } j = 0 \text{ i } j = 1$$

Parametar  $\alpha$  predstavlja brzinu učenja, dok je drugi činilac izvod funkcije greške po  $\theta_0$  i  $\theta_1$ .

Potrebno je izračunati nove vrednosti za oba parametra, pa onda istovremeno izvršiti dodelu novih vrednosti.

$$\begin{aligned} temp0 &:= \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\ temp1 &:= \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\ \theta_0 &:= temp0 \\ \theta_1 &:= temp1 \end{aligned}$$

## PRIMENA ALGORITMA NA LINEARNU REGRESIJU

*Računa se posebno izvod za prvi, a posebno za drugi parametar.*

Algoritam opadajućeg gradijenta se primenjuje na linearnu regresiju sa jednom promenljivom:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1), \quad \text{za } j = 0 \text{ i } j = 1$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

Računa se posebno izvod za prvi, a posebno za drugi parametar.

$$\begin{aligned} \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_0} \times \frac{1}{2m} \sum_{i=1}^m (h_\theta(x) - y)^2 \\ &= \frac{\partial}{\partial \theta_0} \times \frac{1}{2m} \sum_{i=1}^m (h_\theta(x)^2 - 2h_\theta(x)y + y^2) \\ &= \frac{\partial}{\partial \theta_0} \times \frac{1}{2m} \sum_{i=1}^m ((\theta_0 + \theta_1 x)^2 - 2(\theta_0 + \theta_1 x)y + y^2) \\ &= \frac{\partial}{\partial \theta_0} \times \frac{1}{2m} \sum_{i=1}^m (\theta_0^2 + 2\theta_0\theta_1 x + \theta_1^2 x^2 - 2\theta_0 y - 2\theta_1 x y + y^2) \\ &= \frac{1}{2m} \sum_{i=1}^m (2\theta_0 + 2\theta_1 x - 2y) = \frac{1}{2m} \sum_{i=1}^m (2(\theta_0 + \theta_1 x - y)) \\ &= \frac{1}{m} \sum_{i=1}^m (h_\theta(x) - y) \\ \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_1} \times \frac{1}{2m} \sum_{i=1}^m (h_\theta(x) - y)^2 \\ &= \frac{\partial}{\partial \theta_1} \times \frac{1}{2m} \sum_{i=1}^m (h_\theta(x)^2 - 2h_\theta(x)y + y^2) \\ &= \frac{\partial}{\partial \theta_1} \times \frac{1}{2m} \sum_{i=1}^m ((\theta_0 + \theta_1 x)^2 - 2(\theta_0 + \theta_1 x)y + y^2) \\ &= \frac{\partial}{\partial \theta_1} \times \frac{1}{2m} \sum_{i=1}^m (\theta_0^2 + 2\theta_0\theta_1 x + \theta_1^2 x^2 - 2\theta_0 y - 2\theta_1 x y + y^2) \\ &= \frac{1}{2m} \sum_{i=1}^m (2\theta_0 x + 2\theta_1 x^2 - 2x y) = \frac{1}{2m} \sum_{i=1}^m (2x(\theta_0 + \theta_1 x - y)) \\ &= \frac{1}{m} \sum_{i=1}^m (h_\theta(x) - y)x \end{aligned}$$

## IZVOĐENJE IZRAZA ZA LINEARNA REGRESIJA

*U nastavku je video za izvođenje izraza za linearu regresiju.*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 4

# Logistička regresija

## POJAM LOGISTIČKE REGRESIJE

*Logistička regresija predstavlja osnovni algoritam nadgledanog učenja za diskretne podatke.*

Logistička regresija predstavlja osnovni algoritam nadgledanog učenja za diskretne podatke.

Kod logističke regresije, na osnovu postojećih ulaznih i izlaznih podataka, algoritam treba da predvidi izlaz za nove podatke koje su u opštem slučaju **diskretne vrednosti**, najčešće 0 i 1.

$$y \in \{0, 1\}$$

Kao kod svih algoritama nadgledanog učenja skup podataka se deli na trening i test skupove.

Za razliku od linearne regresije, gde estimirane vrednosti za  $y$  mogu bili realne vrednosti različite veličine, kod logističke regresije estimirane vrednosti treba da budu od 0 do 1

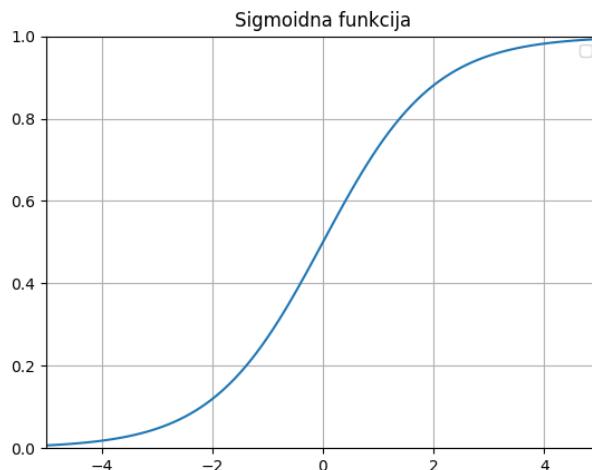
$$0 \leq h_{\theta}(x) \leq 1$$

Hipoteza kod logističke regresije je sledećeg oblika:

$$h_{\theta}(x) = g(\theta^T x)$$

U prethodnoj jednačini funkcija **g()** je sigmoidna funkcija:

$$g(z) = \frac{1}{1 + e^{-z}}$$



Slika 4.1 Sigmoidna funkcija. [Izvor: Autor]

## HIPOTEZA LOGISTIČKE REGRESIJE

*Hipoteza koja je predstavljena sigmoidnom funkcijom na ovaj način predstavlja estimiranu verovatnoću da je  $y=1$  za dati ulaz  $x$ .*

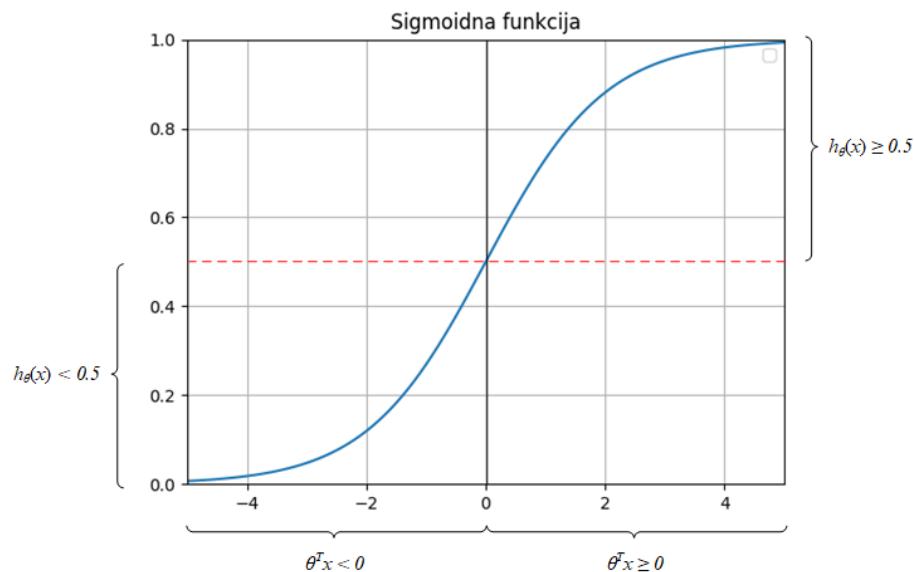
Hipoteza koja je predstavljena sigmoidnom funkcijom na ovaj način predstavlja **estimiranu verovatnoću** da je  $y=1$  za dati ulaz  $x$ , koji je parametrizovan sa  $\theta$ .

$$h_{\theta}(x) = P(y=1|x; \theta)$$

Kako je zbir svih verovatnoća 1, onda se lako može izračunati verovatnoća da je  $y=0$ .

$$\begin{aligned} P(y=0|x; \theta) + P(y=1|x; \theta) &= 1 \\ P(y=0|x; \theta) &= 1 - P(y=1|x; \theta) \end{aligned}$$

Po izgledu sigmoidne funkcije može se zaključiti da kad je vrednost parametra (u ovom slučaju  $\theta^T x$ ) veće od nule, onda je vrednost funkcije veća od 0.5.



Slika 4.2 Vrednosti granica za sigmoidnu funkciju. [Izvor: Autor]

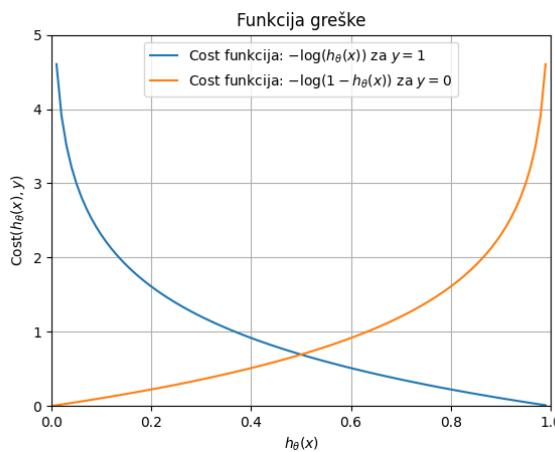
## FUNKCIJA GREŠKE KOD LOGISTIČKE REGRESIJE

*Kod logističke regresije, zbog nelinearne prirode sigmoidne funkcije, nije poželjno koristiti srednjekvadratnu grešku kao funkciju greške.*

Kod logističke regresije, zbog nelinearne prirode sigmoidne funkcije, nije poželjno koristiti srednjekvadratnu grešku kao funkciju greške.

Umesto toga, funkcija greške koja se koristi u logističkoj regresiji jeste sledeća:

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{za } y = 1 \\ -\log(1 - h_\theta(x)) & \text{za } y = 0 \end{cases}$$



Slika 4.3 Funkcija greške. [Izvor: Autor]

Potrebno je spojiti izraz za funkciju greške za  $y=1$  i  $y=0$  u jedinstven izraz.. Nakon toga, treba sabrati za sve vrednosti trening skupa, i podeliti sa brojem elemenata  $m$  da bi se izraz usrednjio.

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \log \left( h_{\theta} \left( x^{(i)} \right) \right) + \left( 1 - y^{(i)} \right) \log \left( 1 - h_{\theta} \left( x^{(i)} \right) \right)$$

## ALGORITAM OPADAJUĆEG GRADIJENTA KOD LOGISTIČKE REGRESIJE

*Kod logističke regresije koristi se sigmoidna funkcija.*

Kao i kod linearne regresije, potrebno je minimizovati funkciju greške. Korišćenjem algoritma opadajućeg građivjenta dobija se sledeći izraz:

ponoviti do konvergencije :

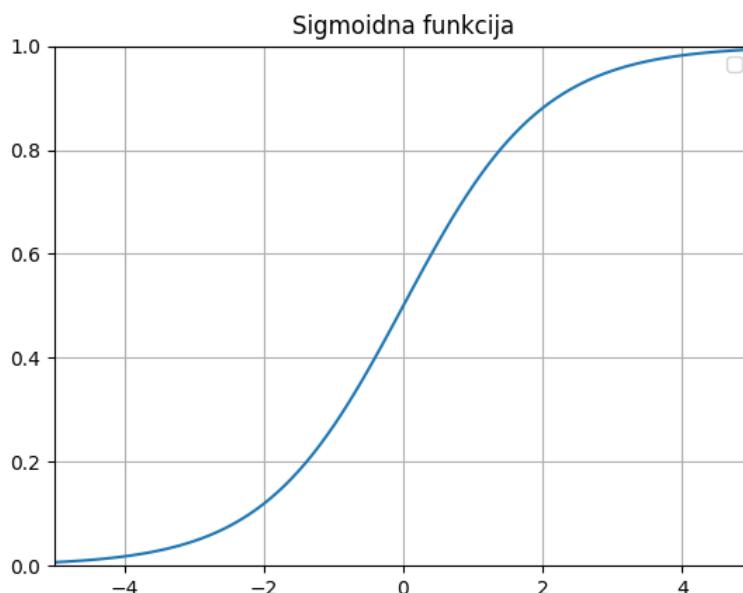
$$\theta_j := \theta_j - \alpha \sum_{i=1}^m \left( h_{\theta} \left( x^{(i)} \right) - y^{(i)} \right) x_j^{(i)}$$

Hipoteza kod logističke regresije je sledećeg oblika:

$$h_{\theta} (x) = g(\theta^T x)$$

U prethodnoj jednačini funkcija  $g()$  je sigmoidna funkcija:

$$g(z) = \frac{1}{1 + e^{-z}}$$



Slika 4.4 Sigmoidna funkcija. [Izvor: Autor]

## ▼ Poglavlje 5

### Paket scikit-learn

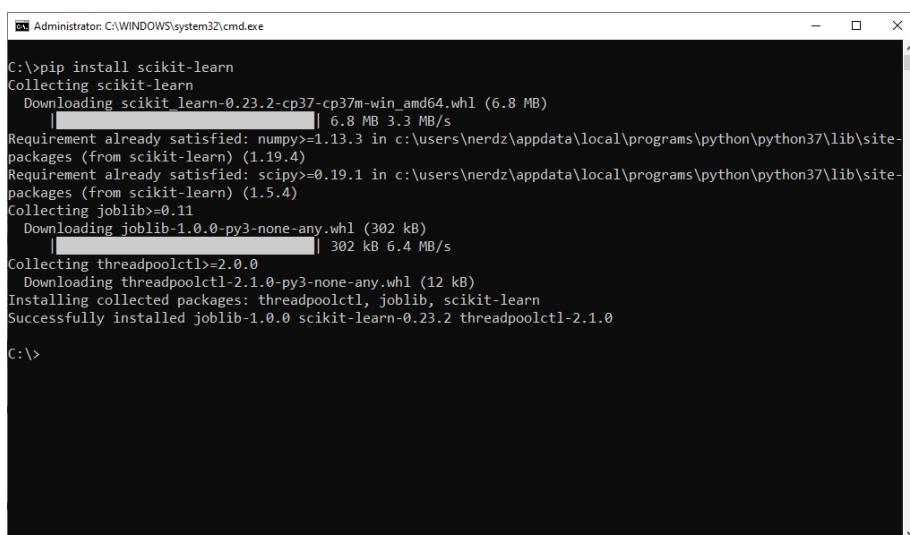
## INSTALACIJA SCIKIT-LEARN PAKETA

*Paket za Python jezik koji sadrži estimatore mašinskog učenja jeste scikit-learn.*

**Scikit-learn** paket jeste paket koji sadrži različite algoritme za mašinsku učenje unutar Python programskog jezika.

Instalira se preko pip instalera:

```
pip install scikit-learn
```



```
C:\>pip install scikit-learn
Collecting scikit-learn
  Downloading scikit_learn-0.23.2-cp37-cp37m-win_amd64.whl (6.8 MB)
    |████████████████████████████████| 6.8 MB 3.3 MB/s
Requirement already satisfied: numpy>=1.13.3 in c:\users\nerdz\appdata\local\programs\python\python37\lib\site-packages (from scikit-learn) (1.19.4)
Requirement already satisfied: scipy>=0.19.1 in c:\users\nerdz\appdata\local\programs\python\python37\lib\site-packages (from scikit-learn) (1.5.4)
Collecting joblib>0.11
  Downloading joblib-1.0.0-py3-none-any.whl (302 kB)
    |████████████████████████████████| 302 kB 6.4 MB/s
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\nerdz\appdata\local\programs\python\python37\lib\site-packages (from scikit-learn) (2.1.0)
  Downloading threadpoolctl-2.1.0-py3-none-any.whl (12 kB)
Installing collected packages: threadpoolctl, joblib, scikit-learn
Successfully installed joblib-1.0.0 scikit-learn-0.23.2 threadpoolctl-2.1.0
C:\>
```

Slika 5.1 Instalacija scikit-learn paketa. [Izvor: Autor]

Scikit-learn u sebi sadrži više modula, od kojih je svaki modul jedan algoritam mašinskog učenja, ili estimator.

Scikit-learn paket se uvozi u Python radno okruženje naredbom:

```
# uvoz scikit-learn paketa
import sklearn
```

Scikit-learn radi sa numpy nizovima, te je neophodno uvesti i numpy paket pri pravljenju programa za mašinsko učenje.

# LINEARNA REGRESIJA U SCIKIT-LEARN PAKETU

*Linerana regresija je unutar modula `linear_models`*

Algoritmi linearne i logističke regresije spadaju u linearne modele, te je neophodno uvesti odgovarajuću klasu iz modula `linear_model`.

```
from sklearn import linear_model
```

## Linearna regresija

Kada se učita skup podataka (sa ulazima i izlazima), najbolje je da se deo skupa podeli na trening skup, a ostatak na test skup, koji će testirati model.

Po konvenciji, ovi skupovi se nazivaju `X_train`, `X_test`, `y_train` i `y_test`.

Nakon podele skupova, pravi se objekat klase `LinearRegression()`

```
regr = linear_model.LinearRegression()
```

Pozivom metode `.fit()` treba ubaciti test parametre (`X_train` i `y_train`)

```
regr.fit(X_train,y_train)
```

Sada je model istreniran, i njegova tačnost se može izračunati pozivom metode `.score()`, a kao ulazni i izlazni podaci mogu se staviti `X_test` i `y_test`.

```
accuracy = regr.score(X_test, y_test)
```

Estimirane vrednosti se mogu dobiti pozivom metode `.predict()` koje za ulazni parametar uzima `X_test` skup.

```
y_pred = regr.predict(X_test)
```

Model se može ponovo istrenirati, ponovnim pozivom metode `.fit()` ukoliko se smatra da je nedovoljna tačnost. Rad na istom skupu podataka obično će dati istu ili sličnu tačnost.

Parametri  $\theta_0$  i  $\theta_1$  se respektivno mogu dobiti pozivanjem atributa `intercept_` i `coef_`

```
# sticanje parametara
print(regr.coef_)
print(regr.intercept_)
```

Kod univariatne linearne regresije, moguće je vizuelno i predstaviti odstupanje estimiranih (`y_pred`) vrednosti sa pravim (`y_test`) pozivom `scatter` i `plot` funkcija iz paketa `matplotlib`.

```
# graficki prikaz
plt.scatter(X_test, y_test)
plt.plot(X_test, y_pred, color='red', label='Linearna regresija')
```

```
plt.grid()  
plt.show()
```

# LOGISTIČKA REGRESIJA U SCIKIT-LEARN PAKETU

*Logistička regresija je unutar modula linear\_models*

Algoritmi linearne i logističke regresije spadaju u linearne modele, te je neophodno uvesti odgovarajuću klasu iz modula **linear\_model**.

```
from sklearn import linear_model
```

## Linearna regresija

Kada se učita skup podataka (sa ulazima i izlazima), najbolje je da se deo skupa podeli na trening skup, a ostatak na test skup, koji će testirati model.

Po konvenciji, ovi skupovi se nazivaju **X\_train**, **X\_test**, **y\_train** i **y\_test**.

Nakon podele skupova, pravi se objekat klase **LogisticRegression()**

```
regr = linear_model.LogisticRegression()
```

Pozivom metode **.fit()** treba ubaciti test parametre (**X\_train** i **y\_train**)

```
regr.fit(X_train,y_train)
```

Sada je model istreniran, i njegova tačnost se može izračunati pozivom metode **.score()**, a kao ulazni i izlazni podaci mogu se staviti **X\_test** i **y\_test**.

```
accuracy = regr.score(X_test, y_test)
```

Estimirane vrednosti se mogu dobiti pozivom metode **.predict()** koje za ulazni parametar uzima **X\_test** skup.

```
y_pred = regr.predict(X_test)
```

Model se može ponovo istrenirati, ponovnim pozivom metode **.fit()** ukoliko se smatra da je nedovoljna tačnost. Rad na istom skupu podataka obično će dati istu ili sličnu tačnost.

Parametri **θ₀** i **θ₁** se respektivno mogu dobiti pozivanjem atributa **intercept\_** i **coef\_**

```
# stampanje parametara  
print(regr.coef_)  
print(regr.intercept_)
```

Kod logističke regresije moguće je izbrojati greške preko **numpy** nizova.

```
# brojanje gresaka  
errors = np.sum(np.where(y_test != y_pred, 1, 0))
```

## ✓ Poglavlje 6

### Pokazne vežbe #12

#### ZADATAK #1

*Zadatak #1 se odnosi na linearu regresiju i radi se okvirno 25 minuta*

##### **Zadatak #1 (25 minuta)**

Iz datoteke **student-mat.csv** učitati kolone **G1** i **G3**. Koristiti kolonu **G1** kao skup ulaznih vrednosti, a kolonu **G3** kao skup izlaznih vrednosti.

Istrenirati model linearne regresije koja uzima 90% skupa za trening skup, a ostalo za test skup.

Štampati koeficijente, tačnost, i vizuelno pokazati linearni model estimacije.

Pokušati sa više ulaznih parametara.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import sklearn
from sklearn import linear_model

df = pd.read_csv('student-mat.csv', sep=' ')

# stampanje samo imena kolona
# print(df.columns)

# stampanje imena kolona i prvih 5 redova
# print(df.head())

# biramo ono sto zelimo da pratimo
data_list = ['G1', 'G3']

data = df[data_list]

# izvlecimo promenljivu koju zelimo da predvidjamo
predict = 'G3'

# dftmp = df.drop([predict], 1)
# print(dftmp)

# novi niz bez onoga sto predvidjamo
x = np.array(data.drop([predict], 1))
```

```
# novi niz samo sa onim sto predvidjamo
y = np.array(data[predict])

# delimo skup u train i test skupove
x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(x, y,
test_size=0.1)

print(len(x_train), len(x_test))

# pravimo objekat klase LinearRegression
linear = linear_model.LinearRegression()

# fit-ujemo prema trening podacima
linear.fit(x_train, y_train)

# predikcija na test podacima

predictions = linear.predict(x_test)

new_df = pd.DataFrame(x_test, columns=['G1'])
new_df['Y'] = y_test
new_df['Predikcija'] = predictions

print(new_df)

# racunamo tacnost, poredimo sa test podacima
accuracy = linear.score(x_test, y_test)
print(accuracy)

# sticanje koeficijenata
print(linear.coef_)
print(linear.intercept_)

# Plot outputs
plt.scatter(new_df['G1'], y_test, color='blue')
plt.plot(new_df['G1'], predictions, color='red')
plt.show()
```

```
school sex age address famsize Pstatus Medu Fedu Mjob Fjob reason guardian
traveltime studytime failures schoolsup famsup paid activities nursery higher
internet romantic famrel freetime goout Dalc Walc health absences G1 G2 G3
"GP" "F" 18 "U" "GT3" "A" 4 4 "at_home" "teacher" "course" "mother" 2 2 0 "yes"
"no" "no" "no" "yes" "yes" "no" "no" 4 3 4 1 1 3 6 "5" "6" 6
"GP" "F" 17 "U" "GT3" "T" 1 1 "at_home" "other" "course" "father" 1 2 0 "no" "yes"
"no" "no" "no" "yes" "yes" "no" 5 3 3 1 1 3 4 "5" "5" 6
"GP" "F" 15 "U" "LE3" "T" 1 1 "at_home" "other" "other" "mother" 1 2 3 "yes" "no"
"yes" "no" "yes" "yes" "yes" "no" 4 3 2 2 3 3 10 "7" "8" 10
"GP" "F" 15 "U" "GT3" "T" 4 2 "health" "services" "home" "mother" 1 3 0 "no" "yes"
"yes" "yes" "yes" "yes" "yes" 3 2 2 1 1 5 2 "15" "14" 15
"GP" "F" 16 "U" "GT3" "T" 3 3 "other" "other" "home" "father" 1 2 0 "no" "yes"
"yes" "no" "yes" "yes" "no" "no" 4 3 2 1 2 5 4 "6" "10" 10
"GP" "M" 16 "U" "LE3" "T" 4 3 "services" "other" "reputation" "mother" 1 2 0 "no"
```

```

"yes" "yes" "yes" "yes" "yes" "yes" "no" 5 4 2 1 2 5 10 "15" "15" 15
"GP" "M" 16 "U" "LE3" "T" 2 2 "other" "other" "home" "mother" 1 2 0 "no" "no" "no"
"no" "yes" "yes" "yes" "no" 4 4 4 1 1 3 0 "12" "12" 11
"GP" "F" 17 "U" "GT3" "A" 4 4 "other" "teacher" "home" "mother" 2 2 0 "yes" "yes"
"no" "no" "yes" "yes" "no" "no" 4 1 4 1 1 1 6 "6" "5" 6
"GP" "M" 15 "U" "LE3" "A" 3 2 "services" "other" "home" "mother" 1 2 0 "no" "yes"
"yes" "no" "yes" "yes" "no" 4 2 2 1 1 1 0 "16" "18" 19
"GP" "M" 15 "U" "GT3" "T" 3 4 "other" "other" "home" "mother" 1 2 0 "no" "yes"
"yes" "yes" "yes" "yes" "no" 5 5 1 1 1 5 0 "14" "15" 15
"GP" "F" 15 "U" "GT3" "T" 4 4 "teacher" "health" "reputation" "mother" 1 2 0 "no"
"yes" "yes" "no" "yes" "yes" "no" 3 3 3 1 2 2 0 "10" "8" 9
"GP" "F" 15 "U" "GT3" "T" 2 1 "services" "other" "reputation" "father" 3 3 0 "no"
"yes" "no" "yes" "yes" "yes" "no" 5 2 2 1 1 4 4 "10" "12" 12
"GP" "M" 15 "U" "LE3" "T" 4 4 "health" "services" "course" "father" 1 1 0 "no"
"yes" "yes" "yes" "yes" "yes" "no" 4 3 3 1 3 5 2 "14" "14" 14
"GP" "M" 15 "U" "GT3" "T" 4 3 "teacher" "other" "course" "mother" 2 2 0 "no" "yes"
"yes" "no" "yes" "yes" "no" 5 4 3 1 2 3 2 "10" "10" 11
"GP" "M" 15 "U" "GT3" "A" 2 2 "other" "other" "home" "other" 1 3 0 "no" "yes" "no"
"no" "yes" "yes" "yes" "yes" 4 5 2 1 1 3 0 "14" "16" 16
"GP" "F" 16 "U" "GT3" "T" 4 4 "health" "other" "home" "mother" 1 1 0 "no" "yes"
"no" "no" "yes" "yes" "yes" "no" 4 4 4 1 2 2 4 "14" "14" 14
"GP" "F" 16 "U" "GT3" "T" 4 4 "services" "services" "reputation" "mother" 1 3 0
"no" "yes" "yes" "yes" "yes" "yes" "no" 3 2 3 1 2 2 6 "13" "14" 14
"GP" "F" 16 "U" "GT3" "T" 3 3 "other" "other" "reputation" "mother" 3 2 0 "yes"
"yes" "no" "yes" "yes" "no" "no" 5 3 2 1 1 4 4 "8" "10" 10
"GP" "M" 17 "U" "GT3" "T" 3 2 "services" "services" "course" "mother" 1 1 3 "no"
"yes" "no" "yes" "yes" "yes" "no" 5 5 5 2 4 5 16 "6" "5" 5
"GP" "M" 16 "U" "LE3" "T" 4 3 "health" "other" "home" "father" 1 1 0 "no" "no"
"yes" "yes" "yes" "yes" "no" 3 1 3 1 3 5 4 "8" "10" 10
"GP" "M" 15 "U" "GT3" "T" 4 3 "teacher" "other" "reputation" "mother" 1 2 0 "no"
"no" "no" "yes" "yes" "yes" "no" 4 4 1 1 1 1 0 "13" "14" 15
"GP" "M" 15 "U" "GT3" "T" 4 4 "health" "health" "other" "father" 1 1 0 "no" "yes"
"yes" "no" "yes" "yes" "yes" "no" 5 4 2 1 1 5 0 "12" "15" 15
"GP" "M" 16 "U" "LE3" "T" 4 2 "teacher" "other" "course" "mother" 1 2 0 "no" "no"
"no" "yes" "yes" "yes" "yes" "no" 4 5 1 1 3 5 2 "15" "15" 16
"GP" "M" 16 "U" "LE3" "T" 2 2 "other" "other" "reputation" "mother" 2 2 0 "no"
"yes" "no" "yes" "yes" "yes" "no" 5 4 4 2 4 5 0 "13" "13" 12
"GP" "F" 15 "R" "GT3" "T" 2 4 "services" "health" "course" "mother" 1 3 0 "yes"
"yes" "yes" "yes" "yes" "yes" "no" 4 3 2 1 1 5 2 "10" "9" 8
"GP" "F" 16 "U" "GT3" "T" 2 2 "services" "services" "home" "mother" 1 1 2 "no"
"yes" "yes" "no" "yes" "yes" "no" 1 2 2 1 3 5 14 "6" "9" 8
"GP" "M" 15 "U" "GT3" "T" 2 2 "other" "other" "home" "mother" 1 1 0 "no" "yes"
"yes" "no" "yes" "yes" "yes" "no" 4 2 2 1 2 5 2 "12" "12" 11
"GP" "M" 15 "U" "GT3" "T" 4 2 "health" "services" "other" "mother" 1 1 0 "no" "no"
"yes" "no" "yes" "yes" "no" 2 2 4 2 4 1 4 "15" "16" 15
"GP" "M" 16 "U" "LE3" "A" 3 4 "services" "other" "home" "mother" 1 2 0 "yes" "yes"
"no" "yes" "yes" "yes" "yes" "no" 5 3 3 1 1 5 4 "11" "11" 11
"GP" "M" 16 "U" "GT3" "T" 4 4 "teacher" "teacher" "home" "mother" 1 2 0 "no" "yes"
"yes" "yes" "yes" "yes" "yes" "yes" 4 4 5 5 5 5 16 "10" "12" 11
"GP" "M" 15 "U" "GT3" "T" 4 4 "health" "services" "home" "mother" 1 2 0 "no" "yes"
"yes" "no" "yes" "yes" "no" 5 4 2 3 4 5 0 "9" "11" 12
"GP" "M" 15 "U" "GT3" "T" 4 4 "services" "services" "reputation" "mother" 2 2 0
"no" "yes" "no" "yes" "yes" "yes" "yes" "no" 4 3 1 1 1 5 0 "17" "16" 17

```

```

"GP" "M" 15 "R" "GT3" "T" 4 3 "teacher" "at_home" "course" "mother" 1 2 0 "no"
"yes" "no" "yes" "yes" "yes" "yes" "yes" 4 5 2 1 1 5 0 "17" "16" 16
"GP" "M" 15 "U" "LE3" "T" 3 3 "other" "other" "course" "mother" 1 2 0 "no" "no"
"no" "yes" "no" "yes" "yes" "no" 5 3 2 1 1 2 0 "8" "10" 12
"GP" "M" 16 "U" "GT3" "T" 3 2 "other" "other" "home" "mother" 1 1 0 "no" "yes"
"yes" "no" "no" "yes" "yes" "no" 5 4 3 1 1 5 0 "12" "14" 15
"GP" "F" 15 "U" "GT3" "T" 2 3 "other" "other" "father" 2 1 0 "no" "yes"
"no" "yes" "yes" "yes" "no" "no" 3 5 1 1 1 5 0 "8" "7" 6
"GP" "M" 15 "U" "LE3" "T" 4 3 "teacher" "services" "home" "mother" 1 3 0 "no" "yes"
"no" "yes" "yes" "yes" "no" 5 4 3 1 1 4 2 "15" "16" 18
"GP" "M" 16 "R" "GT3" "A" 4 4 "other" "teacher" "reputation" "mother" 2 3 0 "no"
"yes" "no" "yes" "yes" "yes" "yes" 2 4 3 1 1 5 7 "15" "16" 15
"GP" "F" 15 "R" "GT3" "T" 3 4 "services" "health" "course" "mother" 1 3 0 "yes"
"yes" "yes" "yes" "yes" "yes" "no" 4 3 2 1 1 5 2 "12" "12" 11
"GP" "F" 15 "R" "GT3" "T" 2 2 "at_home" "other" "reputation" "mother" 1 1 0 "yes"
"yes" "yes" "yes" "yes" "no" "no" 4 3 1 1 1 2 8 "14" "13" 13
"GP" "F" 16 "U" "LE3" "T" 2 2 "other" "other" "home" "mother" 2 2 1 "no" "yes" "no"
"yes" "no" "yes" "yes" 3 3 3 1 2 3 25 "7" "10" 11
"GP" "M" 15 "U" "LE3" "T" 4 4 "teacher" "other" "home" "other" 1 1 0 "no" "yes"
"no" "no" "yes" "yes" "yes" 5 4 3 2 4 5 8 "12" "12" 12
"GP" "M" 15 "U" "GT3" "T" 4 4 "services" "teacher" "course" "father" 1 2 0 "no"
"yes" "no" "yes" "yes" "yes" "no" 4 3 3 1 1 5 2 "19" "18" 18
"GP" "M" 15 "U" "GT3" "T" 2 2 "services" "services" "course" "father" 1 1 0 "yes"
"yes" "no" "yes" "yes" "no" 5 4 1 1 1 1 0 "8" "8" 11
"GP" "F" 16 "U" "LE3" "T" 2 2 "other" "at_home" "course" "father" 2 2 1 "yes" "no"
"no" "yes" "yes" "yes" "no" 4 3 3 2 2 5 14 "10" "10" 9
"GP" "F" 15 "U" "LE3" "A" 4 3 "other" "other" "course" "mother" 1 2 0 "yes" "yes"
"yes" "yes" "yes" "yes" "yes" 5 2 2 1 1 5 8 "8" "8" 6
"GP" "F" 16 "U" "LE3" "A" 3 3 "other" "services" "home" "mother" 1 2 0 "no" "yes"
"no" "no" "yes" "yes" "no" 2 3 5 1 4 3 12 "11" "12" 11
"GP" "M" 16 "U" "GT3" "T" 4 3 "health" "services" "reputation" "mother" 1 4 0 "no"
"no" "no" "yes" "yes" "yes" "no" 4 2 2 1 1 2 4 "19" "19" 20
"GP" "M" 15 "U" "GT3" "T" 4 2 "teacher" "other" "home" "mother" 1 2 0 "no" "yes"
"yes" "no" "yes" "yes" "no" 4 3 3 2 2 5 2 "15" "15" 14
"GP" "F" 15 "U" "GT3" "T" 4 4 "services" "teacher" "other" "father" 1 2 1 "yes"
"yes" "no" "yes" "no" "yes" "no" 4 4 4 1 1 3 2 "7" "7" 7
"GP" "F" 16 "U" "LE3" "T" 2 2 "services" "services" "course" "mother" 3 2 0 "no"
"yes" "yes" "no" "yes" "yes" "no" 4 3 3 2 3 4 2 "12" "13" 13
"GP" "F" 15 "U" "LE3" "T" 4 2 "health" "other" "other" "mother" 1 2 0 "no" "yes"
"yes" "no" "yes" "yes" "no" 4 3 3 1 1 5 2 "11" "13" 13
"GP" "M" 15 "U" "LE3" "A" 4 2 "health" "health" "other" "father" 2 1 1 "no" "no"
"no" "no" "yes" "yes" "no" 5 5 5 3 4 5 6 "11" "11" 10
"GP" "F" 15 "U" "GT3" "T" 4 4 "services" "services" "course" "mother" 1 1 0 "yes"
"yes" "yes" "no" "yes" "yes" "no" 3 3 4 2 3 5 0 "8" "10" 11
"GP" "F" 15 "U" "LE3" "A" 3 3 "other" "other" "other" "mother" 1 1 0 "no" "no"
"yes" "no" "yes" "yes" "yes" "no" 5 3 4 4 4 1 6 "10" "13" 13
"GP" "F" 16 "U" "GT3" "A" 2 1 "other" "other" "other" "mother" 1 2 0 "no" "no"
"yes" "yes" "yes" "yes" "yes" "yes" 5 3 4 1 1 2 8 "8" "9" 10
"GP" "F" 15 "U" "GT3" "A" 4 3 "services" "services" "reputation" "mother" 1 2 0
"no" "yes" "yes" "yes" "yes" "yes" "no" 4 3 2 1 1 1 0 "14" "15" 15
"GP" "M" 15 "U" "GT3" "T" 4 4 "teacher" "health" "reputation" "mother" 1 2 0 "no"
"yes" "no" "yes" "yes" "yes" "no" "no" 3 2 2 1 1 5 4 "14" "15" 15
"GP" "M" 15 "U" "LE3" "T" 1 2 "other" "at_home" "home" "father" 1 2 0 "yes" "yes"

```

```

"no" "yes" "yes" "yes" "yes" "no" 4 3 2 1 1 5 2 "9" "10" 9
"GP" "F" 16 "U" "GT3" "T" 4 2 "services" "other" "course" "mother" 1 2 0 "no" "yes"
"no" "no" "yes" "yes" "yes" "no" 4 2 3 1 1 5 2 "15" "16" 16
"GP" "F" 16 "R" "GT3" "T" 4 4 "health" "teacher" "other" "mother" 1 2 0 "no" "yes"
"no" "yes" "yes" "yes" "no" "no" 2 4 4 2 3 4 6 "10" "11" 11
"GP" "F" 16 "U" "GT3" "T" 1 1 "services" "services" "course" "father" 4 1 0 "yes"
"yes" "no" "yes" "no" "yes" "yes" "yes" 5 5 5 5 5 5 6 "10" "8" 11
"GP" "F" 16 "U" "LE3" "T" 1 2 "other" "services" "reputation" "father" 1 2 0 "yes"
"no" "no" "yes" "yes" "yes" "no" 4 4 3 1 1 1 4 "8" "10" 9
"GP" "F" 16 "U" "GT3" "T" 4 3 "teacher" "health" "home" "mother" 1 3 0 "yes" "yes"
"yes" "yes" "yes" "yes" "no" 3 4 4 2 4 4 2 "10" "9" 9
"GP" "F" 15 "U" "LE3" "T" 4 3 "services" "services" "reputation" "father" 1 2 0
"yes" "no" "no" "yes" "yes" "yes" "yes" "yes" 4 4 4 2 4 2 0 "10" "10" 10
"GP" "F" 16 "U" "LE3" "T" 4 3 "teacher" "services" "course" "mother" 3 2 0 "no"
"yes" "no" "yes" "yes" "yes" "no" 5 4 3 1 2 1 2 "16" "15" 15
"GP" "M" 15 "U" "GT3" "A" 4 4 "other" "services" "reputation" "mother" 1 4 0 "no"
"yes" "no" "yes" "no" "yes" "yes" 1 3 3 5 5 3 4 "13" "13" 12
"GP" "F" 16 "U" "GT3" "T" 3 1 "services" "other" "course" "mother" 1 4 0 "yes"
"yes" "yes" "no" "yes" "yes" "no" 4 3 3 1 2 5 4 "7" "7" 6
"GP" "F" 15 "R" "LE3" "T" 2 2 "health" "services" "reputation" "mother" 2 2 0 "yes"
"yes" "yes" "no" "yes" "yes" "yes" "no" 4 1 3 1 3 4 2 "8" "9" 8
"GP" "F" 15 "R" "LE3" "T" 3 1 "other" "other" "reputation" "father" 2 4 0 "no"
"yes" "no" "no" "yes" "yes" "no" 4 4 2 2 3 3 12 "16" "16" 16
"GP" "M" 16 "U" "GT3" "T" 3 1 "other" "other" "reputation" "father" 2 4 0 "no"
"yes" "yes" "no" "yes" "yes" "no" 4 3 2 1 1 5 0 "13" "15" 15
"GP" "M" 15 "U" "GT3" "T" 4 2 "other" "other" "course" "mother" 1 4 0 "no" "no"
"no" "no" "yes" "yes" "no" 3 3 3 1 1 3 0 "10" "10" 10
"GP" "F" 15 "R" "GT3" "T" 1 1 "other" "other" "reputation" "mother" 1 2 2 "yes"
"yes" "no" "no" "yes" "yes" "yes" 3 3 4 2 4 5 2 "8" "6" 5
"GP" "M" 16 "U" "GT3" "T" 3 1 "other" "other" "reputation" "mother" 1 1 0 "no" "no"
"no" "yes" "yes" "no" "no" 5 3 2 2 2 5 2 "12" "12" 14
"GP" "F" 16 "U" "GT3" "T" 3 3 "other" "services" "home" "mother" 1 2 0 "yes" "yes"
"yes" "yes" "yes" "yes" "no" 4 3 3 2 4 5 54 "11" "12" 11
"GP" "M" 15 "U" "GT3" "T" 4 3 "teacher" "other" "home" "mother" 1 2 0 "no" "yes"
"yes" "yes" "yes" "yes" "no" 4 3 3 2 3 5 6 "9" "9" 10
"GP" "M" 15 "U" "GT3" "T" 4 0 "teacher" "other" "course" "mother" 2 4 0 "no" "no"
"no" "yes" "yes" "yes" "yes" "no" 3 4 3 1 1 1 8 "11" "11" 10
"GP" "F" 16 "U" "GT3" "T" 2 2 "other" "other" "reputation" "mother" 1 4 0 "no" "no"
"yes" "no" "yes" "yes" "yes" "yes" 5 2 3 1 3 3 0 "11" "11" 11
"GP" "M" 17 "U" "GT3" "T" 2 1 "other" "other" "home" "mother" 2 1 3 "yes" "yes"
"no" "yes" "yes" "no" "yes" "no" 4 5 1 1 1 3 2 "8" "8" 10
"GP" "F" 16 "U" "GT3" "T" 3 4 "at_home" "other" "course" "mother" 1 2 0 "no" "yes"
"no" "no" "yes" "yes" "no" 2 4 3 1 2 3 12 "5" "5" 5
"GP" "M" 15 "U" "GT3" "T" 2 3 "other" "services" "course" "father" 1 1 0 "yes"
"yes" "yes" "yes" "no" "yes" "yes" "yes" "yes" 3 2 2 1 3 3 2 "10" "12" 12
"GP" "M" 15 "U" "GT3" "T" 2 3 "other" "other" "home" "mother" 1 3 0 "yes" "no"
"yes" "no" "no" "yes" "yes" "no" 5 3 2 1 2 5 4 "11" "10" 11
"GP" "F" 15 "U" "LE3" "T" 3 2 "services" "other" "reputation" "mother" 1 2 0 "no"
"yes" "yes" "no" "yes" "yes" "yes" "no" 4 4 4 1 1 5 10 "7" "6" 6
"GP" "M" 15 "U" "LE3" "T" 2 2 "services" "services" "home" "mother" 2 2 0 "no" "no"
"yes" "yes" "yes" "yes" "yes" "no" 5 3 3 1 3 4 4 "15" "15" 15
"GP" "F" 15 "U" "GT3" "T" 1 1 "other" "other" "home" "father" 1 2 0 "no" "yes" "no"
"yes" "no" "yes" "yes" "no" 4 3 2 2 3 4 2 "9" "10" 10

```

```

"GP" "F" 15 "U" "GT3" "T" 4 4 "services" "services" "reputation" "father" 2 2 2
"no" "no" "yes" "no" "yes" "yes" "yes" "yes" "yes" 4 4 4 2 3 5 6 "7" "9" 8
"GP" "F" 16 "U" "LE3" "T" 2 2 "at_home" "other" "course" "mother" 1 2 0 "no" "yes"
"no" "no" "yes" "yes" "no" 4 3 4 1 2 2 4 "8" "7" 6
"GP" "F" 15 "U" "GT3" "T" 4 2 "other" "other" "reputation" "mother" 1 3 0 "no"
"yes" "no" "yes" "yes" "yes" "no" 5 3 3 1 3 1 4 "13" "14" 14
"GP" "M" 16 "U" "GT3" "T" 2 2 "services" "other" "reputation" "father" 2 2 1 "no"
"no" "yes" "yes" "no" "yes" "yes" "no" 4 4 2 1 1 3 12 "11" "10" 10
"GP" "M" 16 "U" "LE3" "A" 4 4 "teacher" "health" "reputation" "mother" 1 2 0 "no"
"yes" "no" "no" "yes" "yes" "no" "no" 4 1 3 3 5 5 18 "8" "6" 7
"GP" "F" 16 "U" "GT3" "T" 3 3 "other" "other" "home" "mother" 1 3 0 "no" "yes"
"yes" "no" "yes" "yes" "yes" "yes" 4 3 3 1 3 4 0 "7" "7" 8
"GP" "F" 15 "U" "GT3" "T" 4 3 "services" "other" "reputation" "mother" 1 1 0 "no"
"no" "yes" "yes" "yes" "yes" "no" 4 5 5 1 3 1 4 "16" "17" 18
"GP" "F" 16 "U" "LE3" "T" 3 1 "other" "other" "home" "father" 1 2 0 "yes" "yes"
"no" "no" "yes" "no" "no" 3 3 3 2 3 2 4 "7" "6" 6
"GP" "F" 16 "U" "GT3" "T" 4 2 "teacher" "services" "home" "mother" 2 2 0 "no" "yes"
"yes" "yes" "yes" "yes" "no" 5 3 3 1 1 1 0 "11" "10" 10
"GP" "M" 15 "U" "LE3" "T" 2 2 "services" "health" "reputation" "mother" 1 4 0 "no"
"yes" "no" "yes" "yes" "yes" "no" 4 3 4 1 1 4 6 "11" "13" 14
"GP" "F" 15 "R" "GT3" "T" 1 1 "at_home" "other" "home" "mother" 2 4 1 "yes" "yes"
"yes" "yes" "yes" "yes" "no" 3 1 2 1 1 1 2 "7" "10" 10
"GP" "M" 16 "R" "GT3" "T" 4 3 "services" "other" "reputation" "mother" 2 1 0 "yes"
"yes" "no" "yes" "no" "yes" "yes" "no" 3 3 3 1 1 4 2 "11" "15" 15
"GP" "F" 16 "U" "GT3" "T" 2 1 "other" "other" "course" "mother" 1 2 0 "no" "yes"
"yes" "no" "yes" "no" "yes" 4 3 5 1 1 5 2 "8" "9" 10
"GP" "F" 16 "U" "GT3" "T" 4 4 "other" "other" "reputation" "mother" 1 1 0 "no" "no"
"no" "yes" "no" "yes" "yes" "no" 5 3 4 1 2 1 6 "11" "14" 14
"GP" "F" 16 "U" "GT3" "T" 4 3 "other" "at_home" "course" "mother" 1 3 0 "yes" "yes"
"yes" "no" "yes" "yes" "no" 5 3 5 1 1 3 0 "7" "9" 8
"GP" "M" 16 "U" "GT3" "T" 4 4 "services" "services" "other" "mother" 1 1 0 "yes"
"yes" "yes" "yes" "yes" "yes" "no" 4 5 5 5 5 4 14 "7" "7" 5
"GP" "M" 16 "U" "GT3" "T" 4 4 "services" "teacher" "other" "father" 1 3 0 "no"
"yes" "no" "yes" "yes" "yes" "yes" "yes" 4 4 3 1 1 4 0 "16" "17" 17
"GP" "M" 15 "U" "GT3" "T" 4 4 "services" "other" "course" "mother" 1 1 0 "no" "yes"
"no" "yes" "no" "yes" "no" 5 3 3 1 1 5 4 "10" "13" 14
"GP" "F" 15 "U" "GT3" "T" 3 2 "services" "other" "home" "mother" 2 2 0 "yes" "yes"
"yes" "no" "yes" "yes" "no" 4 3 5 1 1 2 26 "7" "6" 6
"GP" "M" 15 "U" "GT3" "A" 3 4 "services" "other" "course" "mother" 1 2 0 "no" "yes"
"yes" "yes" "yes" "yes" "no" 5 4 4 1 1 1 0 "16" "18" 18
"GP" "F" 15 "U" "GT3" "A" 3 3 "other" "health" "reputation" "father" 1 4 0 "yes"
"no" "no" "yes" "yes" "no" "no" 4 3 3 1 1 4 10 "10" "11" 11
"GP" "F" 15 "U" "GT3" "T" 2 2 "other" "other" "course" "mother" 1 4 0 "yes" "yes"
"yes" "no" "yes" "yes" "yes" "no" 5 1 2 1 1 3 8 "7" "8" 8
"GP" "M" 16 "U" "GT3" "T" 3 3 "services" "other" "home" "father" 1 3 0 "no" "yes"
"no" "yes" "yes" "yes" "no" 5 3 3 1 1 5 2 "16" "18" 18
"GP" "M" 15 "R" "GT3" "T" 4 4 "other" "other" "home" "father" 4 4 0 "no" "yes"
"yes" "yes" "yes" "yes" "yes" 1 3 5 3 5 1 6 "10" "13" 13
"GP" "F" 16 "U" "LE3" "T" 4 4 "health" "health" "other" "mother" 1 3 0 "no" "yes"
"yes" "yes" "yes" "yes" "yes" "yes" 5 4 5 1 1 4 4 "14" "15" 16
"GP" "M" 15 "U" "LE3" "A" 4 4 "teacher" "teacher" "course" "mother" 1 1 0 "no" "no"
"no" "yes" "yes" "yes" "yes" "no" 5 5 3 1 1 4 6 "18" "19" 19
"GP" "F" 16 "R" "GT3" "T" 3 3 "services" "other" "reputation" "father" 1 3 1 "yes"

```

```

"yes" "no" "yes" "yes" "yes" "yes" "no" 4 1 2 1 1 2 0 "7" "10" 10
"GP" "F" 16 "U" "GT3" "T" 2 2 "at_home" "other" "home" "mother" 1 2 1 "yes" "no"
"no" "yes" "yes" "yes" "yes" "no" 3 1 2 1 1 5 6 "10" "13" 13
"GP" "M" 15 "U" "LE3" "T" 4 2 "teacher" "other" "course" "mother" 1 1 0 "no" "no"
"no" "no" "yes" "yes" "yes" "no" 3 5 2 1 1 3 10 "18" "19" 19
"GP" "M" 15 "R" "GT3" "T" 2 1 "health" "services" "reputation" "mother" 1 2 0 "no"
"no" "no" "yes" "yes" "yes" "yes" "yes" 5 4 2 1 1 5 8 "9" "9" 9
"GP" "M" 16 "U" "GT3" "T" 4 4 "teacher" "teacher" "course" "father" 1 2 0 "no"
"yes" "no" "yes" "yes" "yes" "yes" "no" 5 4 4 1 2 5 2 "15" "15" 16
"GP" "M" 15 "U" "GT3" "T" 4 4 "other" "teacher" "reputation" "father" 2 2 0 "no"
"yes" "no" "yes" "yes" "yes" "no" "no" 4 4 3 1 1 2 2 "11" "13" 14
"GP" "M" 16 "U" "GT3" "T" 3 3 "other" "services" "home" "father" 2 1 0 "no" "no"
"no" "yes" "yes" "yes" "yes" "no" 5 4 2 1 1 5 0 "13" "14" 13
"GP" "M" 17 "R" "GT3" "T" 1 3 "other" "other" "course" "father" 3 2 1 "no" "yes"
"no" "yes" "yes" "yes" "no" 5 2 4 1 4 5 20 "9" "7" 8
"GP" "M" 15 "U" "GT3" "T" 3 4 "other" "other" "reputation" "father" 1 1 0 "no" "no"
"no" "no" "yes" "yes" "no" 3 4 3 1 2 4 6 "14" "13" 13
"GP" "F" 15 "U" "GT3" "T" 1 2 "at_home" "services" "course" "mother" 1 2 0 "no"
"no" "no" "no" "yes" "yes" "no" 3 2 3 1 2 1 2 "16" "15" 15
"GP" "M" 15 "U" "GT3" "T" 2 2 "services" "services" "home" "father" 1 4 0 "no"
"yes" "yes" "yes" "yes" "yes" "yes" "no" 5 5 4 1 2 5 6 "16" "14" 15
"GP" "F" 16 "U" "LE3" "T" 2 4 "other" "health" "course" "father" 2 2 0 "no" "yes"
"yes" "yes" "yes" "yes" "yes" "yes" 4 2 2 1 2 5 2 "13" "13" 13
"GP" "M" 16 "U" "GT3" "T" 4 4 "health" "other" "course" "mother" 1 1 0 "no" "yes"
"no" "yes" "yes" "yes" "no" 3 4 4 1 4 5 18 "14" "11" 13
"GP" "F" 16 "U" "GT3" "T" 2 2 "other" "other" "home" "mother" 1 2 0 "no" "no" "yes"
"no" "yes" "yes" "yes" "yes" 5 4 4 1 1 5 0 "8" "7" 8
"GP" "M" 15 "U" "GT3" "T" 3 4 "services" "services" "home" "father" 1 1 0 "yes"
"no" "no" "yes" "yes" "yes" "no" 5 5 5 3 2 5 0 "13" "13" 12
"GP" "F" 15 "U" "LE3" "A" 3 4 "other" "other" "home" "mother" 1 2 0 "yes" "no" "no"
"yes" "yes" "yes" "yes" "yes" 5 3 2 1 1 1 0 "7" "10" 11
"GP" "F" 19 "U" "GT3" "T" 0 1 "at_home" "other" "course" "other" 1 2 3 "no" "yes"
"no" "no" "no" "no" "no" 3 4 2 1 1 5 2 "7" "8" 9
"GP" "M" 18 "R" "GT3" "T" 2 2 "services" "other" "reputation" "mother" 1 1 2 "no"
"yes" "no" "yes" "yes" "yes" "no" 3 3 3 1 2 4 0 "7" "4" 0
"GP" "M" 16 "R" "GT3" "T" 4 4 "teacher" "teacher" "course" "mother" 1 1 0 "no" "no"
"yes" "yes" "yes" "yes" "yes" "no" 3 5 5 2 5 4 8 "18" "18" 18
"GP" "F" 15 "R" "GT3" "T" 3 4 "services" "teacher" "course" "father" 2 3 2 "no"
"yes" "no" "no" "yes" "yes" "yes" "yes" 4 2 2 2 2 5 0 "12" "0" 0
"GP" "F" 15 "U" "GT3" "T" 1 1 "at_home" "other" "course" "mother" 3 1 0 "no" "yes"
"no" "yes" "no" "yes" "yes" "yes" 4 3 3 1 2 4 0 "8" "0" 0
"GP" "F" 17 "U" "LE3" "T" 2 2 "other" "other" "course" "father" 1 1 0 "no" "yes"
"no" "no" "yes" "yes" "yes" "yes" 3 4 4 1 3 5 12 "10" "13" 12
"GP" "F" 16 "U" "GT3" "A" 3 4 "services" "other" "course" "father" 1 1 0 "no" "no"
"no" "no" "yes" "yes" "yes" "no" 3 2 1 1 4 5 16 "12" "11" 11
"GP" "M" 15 "R" "GT3" "T" 3 4 "at_home" "teacher" "course" "mother" 4 2 0 "no"
"yes" "no" "no" "yes" "yes" "no" "yes" 5 3 3 1 1 5 0 "9" "0" 0
"GP" "F" 15 "U" "GT3" "T" 4 4 "services" "at_home" "course" "mother" 1 3 0 "no"
"yes" "no" "yes" "yes" "yes" "yes" "yes" 4 3 3 1 1 5 0 "11" "0" 0
"GP" "M" 17 "R" "GT3" "T" 3 4 "at_home" "other" "course" "mother" 3 2 0 "no" "no"
"no" "no" "yes" "yes" "no" "no" 5 4 5 2 4 5 0 "10" "0" 0
"GP" "F" 16 "U" "GT3" "A" 3 3 "other" "other" "course" "other" 2 1 2 "no" "yes"
"no" "yes" "no" "yes" "yes" "yes" 4 3 2 1 1 5 0 "4" "0" 0

```

"GP" "M" 16 "U" "LE3" "T" 1 1 "services" "other" "course" "mother" 1 2 1 "no" "no"  
 "no" "no" "yes" "yes" "no" "yes" 4 4 4 1 3 5 0 "14" "12" 12  
 "GP" "F" 15 "U" "GT3" "T" 4 4 "teacher" "teacher" "course" "mother" 2 1 0 "no" "no"  
 "no" "yes" "yes" "yes" "no" 4 3 2 1 1 5 0 "16" "16" 15  
 "GP" "M" 15 "U" "GT3" "T" 4 3 "teacher" "services" "course" "father" 2 4 0 "yes"  
 "yes" "no" "no" "yes" "yes" "no" 2 2 2 1 1 3 0 "7" "9" 0  
 "GP" "M" 16 "U" "LE3" "T" 2 2 "services" "services" "reputation" "father" 2 1 2  
 "no" "yes" "no" "yes" "yes" "yes" "no" 2 3 3 2 2 2 8 "9" "9" 9  
 "GP" "F" 15 "U" "GT3" "T" 4 4 "teacher" "services" "course" "mother" 1 3 0 "no"  
 "yes" "yes" "yes" "yes" "yes" "no" 4 2 2 1 1 5 2 "9" "11" 11  
 "GP" "F" 16 "U" "LE3" "T" 1 1 "at\_home" "at\_home" "course" "mother" 1 1 0 "no" "no"  
 "no" "no" "yes" "yes" "no" 3 4 4 3 3 1 2 "14" "14" 13  
 "GP" "M" 17 "U" "GT3" "T" 2 1 "other" "other" "home" "mother" 1 1 3 "no" "yes" "no"  
 "no" "yes" "yes" "no" 5 4 5 1 2 5 0 "5" "0" 0  
 "GP" "F" 15 "U" "GT3" "T" 1 1 "other" "services" "course" "father" 1 2 0 "no" "yes"  
 "yes" "no" "yes" "yes" "no" 4 4 2 1 2 5 0 "8" "11" 11  
 "GP" "F" 15 "U" "GT3" "T" 3 2 "health" "services" "home" "father" 1 2 3 "no" "yes"  
 "no" "no" "yes" "yes" "no" 3 3 2 1 1 3 0 "6" "7" 0  
 "GP" "F" 15 "U" "GT3" "T" 1 2 "at\_home" "other" "course" "mother" 1 2 0 "no" "yes"  
 "yes" "no" "yes" "yes" "no" 4 3 2 1 1 5 2 "10" "11" 11  
 "GP" "M" 16 "U" "GT3" "T" 4 4 "teacher" "teacher" "course" "mother" 1 1 0 "no"  
 "yes" "no" "no" "yes" "yes" "no" 3 3 2 2 1 5 0 "7" "6" 0  
 "GP" "M" 15 "U" "LE3" "A" 2 1 "services" "other" "course" "mother" 4 1 3 "no" "no"  
 "no" "no" "yes" "yes" "no" 4 5 5 2 5 5 0 "8" "9" 10  
 "GP" "M" 18 "U" "LE3" "T" 1 1 "other" "other" "course" "mother" 1 1 3 "no" "no"  
 "no" "no" "yes" "no" "yes" 2 3 5 2 5 4 0 "6" "5" 0  
 "GP" "M" 16 "U" "LE3" "T" 2 1 "at\_home" "other" "course" "mother" 1 1 1 "no" "no"  
 "no" "yes" "yes" "no" "yes" 4 4 4 3 5 5 6 "12" "13" 14  
 "GP" "F" 15 "R" "GT3" "T" 3 3 "services" "services" "reputation" "other" 2 3 2 "no"  
 "yes" "yes" "yes" "yes" "yes" "yes" 4 2 1 2 3 3 8 "10" "10" 10  
 "GP" "M" 19 "U" "GT3" "T" 3 2 "services" "at\_home" "home" "mother" 1 1 3 "no" "yes"  
 "no" "no" "yes" "no" "yes" 4 5 4 1 1 4 0 "5" "0" 0  
 "GP" "F" 17 "U" "GT3" "T" 4 4 "other" "teacher" "course" "mother" 1 1 0 "yes" "yes"  
 "no" "no" "yes" "yes" "no" "yes" 4 2 1 1 1 4 0 "11" "11" 12  
 "GP" "M" 15 "R" "GT3" "T" 2 3 "at\_home" "services" "course" "mother" 1 2 0 "yes"  
 "no" "yes" "yes" "yes" "no" "no" 4 4 4 1 1 2 "11" "8" 8  
 "GP" "M" 17 "R" "LE3" "T" 1 2 "other" "other" "reputation" "mother" 1 1 0 "no" "no"  
 "no" "no" "yes" "yes" "no" "no" 2 2 2 3 3 5 8 "16" "12" 13  
 "GP" "F" 18 "R" "GT3" "T" 1 1 "at\_home" "other" "course" "mother" 3 1 3 "no" "yes"  
 "no" "yes" "no" "no" "no" 5 2 5 1 5 4 6 "9" "8" 10  
 "GP" "M" 16 "R" "GT3" "T" 2 2 "at\_home" "other" "course" "mother" 3 1 0 "no" "no"  
 "no" "no" "no" "yes" "no" "no" 4 2 2 1 2 3 2 "17" "15" 15  
 "GP" "M" 16 "U" "GT3" "T" 3 3 "other" "services" "course" "father" 1 2 1 "no" "yes"  
 "yes" "no" "yes" "yes" "yes" "yes" 4 5 5 4 4 5 4 "10" "12" 12  
 "GP" "M" 17 "R" "LE3" "T" 2 1 "at\_home" "other" "course" "mother" 2 1 2 "no" "no"  
 "no" "yes" "yes" "no" "yes" "yes" 3 3 2 2 2 5 0 "7" "6" 0  
 "GP" "M" 15 "R" "GT3" "T" 3 2 "other" "other" "course" "mother" 2 2 2 "yes" "yes"  
 "no" "no" "yes" "yes" "yes" "yes" 4 4 4 1 4 3 6 "5" "9" 7  
 "GP" "M" 16 "U" "LE3" "T" 1 2 "other" "other" "course" "mother" 2 1 1 "no" "no"  
 "no" "yes" "yes" "yes" "no" "no" 4 4 4 2 4 5 0 "7" "0" 0  
 "GP" "M" 17 "U" "GT3" "T" 1 3 "at\_home" "services" "course" "father" 1 1 0 "no"  
 "no" "no" "no" "yes" "no" "yes" "no" 5 3 3 1 4 2 2 "10" "10" 10  
 "GP" "M" 17 "R" "LE3" "T" 1 1 "other" "services" "course" "mother" 4 2 3 "no" "no"

```

"no" "yes" "yes" "no" "no" "yes" 5 3 5 1 5 5 0 "5" "8" 7
"GP" "M" 16 "U" "GT3" "T" 3 2 "services" "services" "course" "mother" 2 1 1 "no"
"yes" "no" "yes" "no" "no" "no" 4 5 2 1 1 2 16 "12" "11" 12
"GP" "M" 16 "U" "GT3" "T" 2 2 "other" "other" "course" "father" 1 2 0 "no" "no"
"no" "no" "yes" "no" "yes" "no" 4 3 5 2 4 4 4 "10" "10" 10
"GP" "F" 16 "U" "GT3" "T" 4 2 "health" "services" "home" "father" 1 2 0 "no" "no"
"yes" "no" "yes" "yes" "yes" 4 2 3 1 1 3 0 "14" "15" 16
"GP" "F" 16 "U" "GT3" "T" 2 2 "other" "other" "home" "mother" 1 2 0 "no" "yes"
"yes" "no" "yes" "yes" "no" 5 1 5 1 1 4 0 "6" "7" 0
"GP" "F" 16 "U" "GT3" "T" 4 4 "health" "health" "reputation" "mother" 1 2 0 "no"
"yes" "no" "yes" "yes" "yes" "yes" 4 4 2 1 1 3 0 "14" "14" 14
"GP" "M" 16 "U" "GT3" "T" 3 4 "other" "other" "course" "father" 3 1 2 "no" "yes"
"no" "yes" "no" "yes" "no" 3 4 5 2 4 2 0 "6" "5" 0
"GP" "M" 16 "U" "GT3" "T" 1 0 "other" "other" "reputation" "mother" 2 2 0 "no"
"yes" "yes" "yes" "yes" "yes" "yes" 4 3 2 1 1 3 2 "13" "15" 16
"GP" "M" 17 "U" "LE3" "T" 4 4 "teacher" "other" "reputation" "mother" 1 2 0 "no"
"yes" "yes" "yes" "yes" "yes" "no" 4 4 4 1 3 5 0 "13" "11" 10
"GP" "F" 16 "U" "GT3" "T" 1 3 "at_home" "services" "home" "mother" 1 2 3 "no" "no"
"no" "yes" "no" "yes" "yes" 4 3 5 1 1 3 0 "8" "7" 0
"GP" "F" 16 "U" "LE3" "T" 3 3 "other" "other" "reputation" "mother" 2 2 0 "no"
"yes" "yes" "yes" "yes" "yes" "yes" "no" 4 4 5 1 1 4 4 "10" "11" 9
"GP" "M" 17 "U" "LE3" "T" 4 3 "teacher" "other" "course" "mother" 2 2 0 "no" "no"
"yes" "yes" "yes" "yes" "no" 4 4 4 4 4 4 4 "10" "9" 9
"GP" "F" 16 "U" "GT3" "T" 2 2 "services" "other" "reputation" "mother" 2 2 0 "no"
"no" "yes" "yes" "no" "yes" "yes" "no" 3 4 4 1 4 5 2 "13" "13" 11
"GP" "M" 17 "U" "GT3" "T" 3 3 "other" "other" "reputation" "father" 1 2 0 "no" "no"
"no" "yes" "no" "yes" "no" 4 3 4 1 4 4 4 "6" "5" 6
"GP" "M" 16 "R" "GT3" "T" 4 2 "teacher" "services" "other" "mother" 1 1 0 "no"
"yes" "no" "yes" "yes" "yes" "yes" "yes" 4 3 3 3 4 3 10 "10" "8" 9
"GP" "M" 17 "U" "GT3" "T" 4 3 "other" "other" "course" "mother" 1 2 0 "no" "yes"
"no" "yes" "yes" "yes" "yes" "yes" 5 2 3 1 1 2 4 "10" "10" 11
"GP" "M" 16 "U" "GT3" "T" 4 3 "teacher" "other" "home" "mother" 1 2 0 "no" "yes"
"yes" "yes" "yes" "yes" "no" 3 4 3 2 3 3 10 "9" "8" 8
"GP" "M" 16 "U" "GT3" "T" 3 3 "services" "other" "home" "mother" 1 2 0 "no" "no"
"yes" "yes" "yes" "yes" "yes" "yes" 4 2 3 1 2 3 2 "12" "13" 12
"GP" "F" 17 "U" "GT3" "T" 2 4 "services" "services" "reputation" "father" 1 2 0
"no" "yes" "no" "yes" "yes" "yes" "no" "no" 5 4 2 2 3 5 0 "16" "17" 17
"GP" "F" 17 "U" "LE3" "T" 3 3 "other" "other" "reputation" "mother" 1 2 0 "no"
"yes" "no" "yes" "yes" "yes" "yes" 5 3 3 2 3 1 56 "9" "9" 8
"GP" "F" 16 "U" "GT3" "T" 3 2 "other" "other" "reputation" "mother" 1 2 0 "no"
"yes" "yes" "no" "yes" "yes" "yes" "no" 1 2 2 1 2 1 14 "12" "13" 12
"GP" "M" 17 "U" "GT3" "T" 3 3 "services" "services" "other" "mother" 1 2 0 "no"
"yes" "no" "yes" "yes" "yes" "yes" "yes" 4 3 4 2 3 4 12 "12" "12" 11
"GP" "M" 16 "U" "GT3" "T" 1 2 "services" "services" "other" "mother" 1 1 0 "no"
"yes" "yes" "yes" "yes" "yes" "yes" "yes" 3 3 3 1 2 3 2 "11" "12" 11
"GP" "M" 16 "U" "LE3" "T" 2 1 "other" "other" "course" "mother" 1 2 0 "no" "no"
"yes" "yes" "yes" "yes" "yes" "yes" 4 2 3 1 2 5 0 "15" "15" 15
"GP" "F" 17 "U" "GT3" "A" 3 3 "health" "other" "reputation" "mother" 1 2 0 "no"
"yes" "no" "no" "yes" "yes" "yes" "yes" 3 3 3 1 3 3 6 "8" "7" 9
"GP" "M" 17 "R" "GT3" "T" 1 2 "at_home" "other" "home" "mother" 1 2 0 "no" "no"
"no" "no" "yes" "yes" "no" "no" 3 1 3 1 5 3 4 "8" "9" 10
"GP" "F" 16 "U" "GT3" "T" 2 3 "services" "services" "course" "mother" 1 2 0 "no"
"no" "no" "no" "yes" "yes" "yes" "no" 4 3 3 1 1 2 10 "11" "12" 13

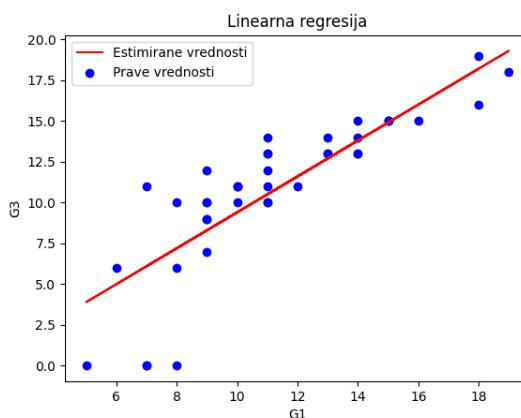
```

```

"GP" "F" 17 "U" "GT3" "T" 1 1 "at_home" "services" "course" "mother" 1 2 0 "no"
"no" "no" "yes" "yes" "yes" "yes" "no" 5 3 3 1 1 3 0 "8" "8" 9
"GP" "M" 17 "U" "GT3" "T" 1 2 "at_home" "services" "other" "other" 2 2 0 "no" "no"
"yes" "yes" "no" "yes" "yes" "no" 4 4 4 4 5 5 12 "7" "8" 8
"GP" "M" 16 "R" "GT3" "T" 3 3 "services" "services" "reputation" "mother" 1 1 0
"no" "yes" "no" "yes" "yes" "yes" "no" 4 3 2 3 4 5 8 "8" "9" 10
"GP" "M" 16 "U" "GT3" "T" 2 3 "other" "other" "home" "father" 2 1 0 "no" "no" "no"
"no" "yes" "yes" "yes" "no" 5 3 3 1 1 3 0 "13" "14" 14
"GP" "F" 17 "U" "LE3" "T" 2 4 "services" "services" "course" "father" 1 2 0 "no"
"no" "no" "yes" "yes" "yes" "yes" "yes" 4 3 2 1 1 5 0 "14" "15" 15
"GP" "M" 17 "U" "GT3" "T" 4 4 "services" "teacher" "home" "mother" 1 1 0 "no" "no"
"no" "no" "yes" "yes" "yes" "no" 5 2 3 1 2 5 4 "17" "15" 16
"GP" "M" 16 "R" "LE3" "T" 3 3 "teacher" "other" "home" "father" 3 1 0 "no" "yes"
"yes" "yes" "yes" "yes" "no" 3 3 4 3 5 3 8 "9" "9" 10
"GP" "F" 17 "U" "GT3" "T" 4 4 "services" "teacher" "home" "mother" 2 1 1 "no" "yes"
"no" "no" "yes" "yes" "no" 4 2 4 2 3 2 24 "18" "18" 18
"GP" "F" 16 "U" "LE3" "T" 4 4 "teacher" "teacher" "reputation" "mother" 1 2 0 "no"
"yes" "yes" "no" "yes" "yes" "no" 4 5 2 1 2 3 0 "9" "9" 10
"GP" "F" 16 "U" "GT3" "T" 4 3 "health" "other" "home" "mother" 1 2 0 "no" "yes"
"no" "yes" "yes" "yes" "no" 4 3 5 1 5 2 2 "16" "16" 16
"GP" "F" 16 "U" "GT3" "T" 2 3 "other" "other" "reputation" "mother" 1 2 0 "yes"
"yes" "yes" "yes" "yes" "no" "no" 4 4 3 1 3 4 6 "8" "10" 10
"GP" "F" 17 "U" "GT3" "T" 1 1 "other" "other" "course" "mother" 1 2 0 "no" "yes"
"yes" "no" "no" "yes" "no" 4 4 4 1 3 1 4 "9" "9" 10
"GP" "F" 17 "R" "GT3" "T" 2 2 "other" "other" "reputation" "mother" 1 1 0 "no"
"yes" "no" "no" "yes" "yes" "no" 5 3 2 1 2 3 18 "7" "6" 6
"GP" "F" 16 "R" "GT3" "T" 2 2 "services" "services" "reputation" "mother" 2 4 0
"no" "yes" "yes" "no" "yes" "yes" "no" 5 3 5 1 1 5 6 "10" "10" 11
"GP" "F" 17 "U" "GT3" "T" 3 4 "at_home" "services" "home" "mother" 1 3 1 "no" "yes"
"yes" "no" "yes" "yes" "yes" "yes" 4 4 3 3 4 5 28 "10" "9" 9
"GP" "F" 16 "U" "GT3" "A" 3 1 "services" "other" "course" "mother" 1 2 3 "no" "yes"
"yes" "no" "yes" "yes" "no" 2 3 3 2 2 4 5 "7" "7" 7
"GP" "F" 16 "U" "GT3" "T" 4 3 "teacher" "other" "other" "mother" 1 2 0 "no" "no"
"yes" "yes" "yes" "yes" "yes" "yes" 1 3 2 1 1 1 10 "11" "12" 13
"GP" "F" 16 "U" "GT3" "T" 1 1 "at_home" "other" "home" "mother" 2 1 0 "no" "yes"
"yes" "no" "yes" "yes" "no" 4 3 2 1 4 5 6 "9" "9" 10
"GP" "F" 17 "R" "GT3" "T" 4 3 "teacher" "other" "reputation" "mother" 2 3 0 "no"
"yes" "yes" "yes" "yes" "yes" "yes" 4 4 2 1 1 4 6 "7" "7" 7
"GP" "F" 19 "U" "GT3" "T" 3 3 "other" "other" "reputation" "other" 1 4 0 "no" "yes"
"yes" "yes" "yes" "yes" "no" 4 3 3 1 2 3 10 "8" "8" 8
"GP" "M" 17 "U" "LE3" "T" 4 4 "services" "other" "home" "mother" 1 2 0 "no" "yes"
"yes" "no" "yes" "yes" "yes" "yes" 5 3 5 4 5 3 13 "12" "12" 13
"GP" "F" 16 "U" "GT3" "A" 2 2 "other" "other" "reputation" "mother" 1 2 0 "yes"
"yes" "yes" "no" "yes" "yes" "yes" "no" 3 3 4 1 1 4 0 "12" "13" 14
"GP" "M" 18 "U" "GT3" "T" 2 2 "services" "other" "home" "mother" 1 2 1 "no" "yes"
"yes" "yes" "yes" "yes" "no" 4 4 4 2 4 5 15 "6" "7" 8
"GP" "F" 17 "R" "LE3" "T" 4 4 "services" "other" "other" "mother" 1 1 0 "no" "yes"
"yes" "no" "yes" "yes" "no" "no" 5 2 1 1 2 3 12 "8" "10" 10
"GP" "F" 17 "U" "LE3" "T" 3 2 "other" "other" "reputation" "mother" 2 2 0 "no" "no"
"yes" "no" "yes" "yes" "yes" "no" 4 4 4 1 3 1 2 "14" "15" 15
"GP" "F" 17 "U" "GT3" "T" 4 3 "other" "other" "reputation" "mother" 1 2 2 "no" "no"
"yes" "no" "yes" "yes" "yes" "yes" "yes" 3 4 5 2 4 1 22 "6" "6" 4
"GP" "M" 18 "U" "LE3" "T" 3 3 "services" "health" "home" "father" 1 2 1 "no" "yes"

```

```
"yes" "no" "yes" "yes" "no" 3 2 4 2 4 4 13 "6" "6" 8
"GP" "F" 17 "U" "GT3" "T" 2 3 "at_home" "other" "home" "father" 2 1 0 "no" "yes"
"yes" "no" "yes" "yes" "no" "no" 3 3 3 1 4 3 3 "7" "7" 8
"GP" "F" 17 "U" "GT3" "T" 2 2 "at_home" "at_home" "course" "mother" 1 3 0 "no"
"yes" "yes" "yes" "yes" "yes" "no" 4 3 3 1 1 4 4 "9" "10" 10
"GP" "F" 17 "R" "GT3" "T" 2 1 "at_home" "services" "reputation" "mother" 2 2 0 "no"
"yes" "no" "yes" "yes" "yes" "no" 4 2 5 1 2 5 2 "6" "6" 6
"GP" "F" 17 "U" "GT3" "T" 1 1 "at_home" "other" "reputation" "mother" 1 3 1 "no"
"yes" "no" "yes" "yes" "no" "yes" 4 3 4 1 1 5 0 "6" "5" 0
"GP" "F" 16 "U" "GT3" "T" 2 3 "services" "teacher" "other" "mother" 1 2 0 "yes"
"no" "no" "no" "yes" "yes" "yes" "no" 2 3 1 1 1 3 2 "16" "16" 17
"GP" "M" 18 "U" "GT3" "T" 2 2 "other" "other" "home" "mother" 2 2 0 "no" "yes"
"yes" "no" "yes" "yes" "no" 3 3 3 5 5 4 0 "12" "13" 13
"GP" "F" 16 "U" "GT3" "T" 4 4 "teacher" "services" "home" "mother" 1 3 0 "no" "yes"
"no" "yes"
```



Slika 6.1 Grafik linearne regresije. [Izvor: Autor]

## ZADATAK #2

*Zadatak #2 se odnosi na logističku regresiju i radi se okvirno 20 minuta*

### Zadatak #2 (20 minuta)

Učitati datoteku job.csv. Koristiti kolone GMAT, GPA, Experience kao skup ulaznih vrednosti, a kolonu Admitted kao skup izlaznih vrednosti.

Istrenirati model logističke regresije koja uzima 75% skupa za trening skup, a ostalo za test skup.

Štampati u koloni prave vrednosti, estimirane vrednosti. Izračunati i štampati broj grešaka.

Zatim, učitati datoteku new\_candidates.csv i na osnovu ulaznih parametara i istreniranog modela, napraviti estimaciju za **y** i štampati kao tabelu.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

df = pd.read_csv('jobs.csv', sep='\t')

X = df[['GMAT', 'GPA', 'Experience']]
y = df['Admitted']

#train is based on 75% of the dataset, test is based on 25% of dataset
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.25,random_state=0)

logistic_regression= LogisticRegression()

logistic_regression.fit(X_train,y_train)

y_pred=logistic_regression.predict(X_test)

# print
df2 = pd.DataFrame(X_test)
df2['Actual'] = y_test
df2['Predicted'] = y_pred
print(df2)
errors = np.sum(np.where(y_test != y_pred, 1, 0))
print("Broj gresaka: ", errors)

df2 = pd.read_csv('new_candidates.csv', sep='\t')

y_pred=logistic_regression.predict(df2)

df2['Predictions'] = y_pred
print(df2)
```

GMAT	GPA	Experience	Admitted
780	4.0	3	1
750	3.9	4	1
690	3.3	3	0
710	3.7	5	1
680	3.9	4	0
730	3.7	6	1
690	2.3	1	0
720	3.3	4	1
740	3.3	5	1
690	1.7	1	0
610	2.7	3	0
690	3.7	5	1
710	3.7	6	1
680	3.3	4	0
770	3.3	3	1

610	3.0	1	0
580	2.7	4	0
650	3.7	6	1
540	2.7	2	0
590	2.3	3	0
620	3.3	2	1
600	2.0	1	0
550	2.3	4	0
550	2.7	1	0
570	3.0	2	0
670	3.3	6	1
660	3.7	4	1
580	2.3	2	0
650	3.7	6	1
660	3.3	5	1
640	3.0	1	0
620	2.7	2	0
660	4.0	4	1
660	3.3	6	1
680	3.3	5	1
650	2.3	1	0
670	2.7	2	0
580	3.3	1	0
590	1.7	4	0
690	3.7	5	1

```
# new values
new_candidates = {'GMAT': [590,740,680,610,710],
                  'GPA': [2,3.7,3.3,2.3,3],
                  'Experience': [3,4,6,1,5]
                 }
```

	GMAT	GPA	Experience	Actual	Predicted
22	550	2.3		4	0
20	620	3.3		2	1
25	670	3.3		6	1
4	680	3.9		4	0
10	610	2.7		3	0
15	610	3.0		1	0
28	650	3.7		6	1
11	690	3.7		5	1
18	540	2.7		2	0
29	660	3.3		5	1
Broj gresaka: 2					
	GMAT	GPA	Experience	Predictions	
0	590	2.0		0	
1	740	3.7		1	
2	680	3.3		1	
3	610	2.3		0	
4	710	3.0		1	
PS D:\CS324\L12>					

Slika 6.2 Izlaz logističke regresije. [Izvor: Autor]

## VIDEO OBJAŠNJENJA ZA POKAZNE VEŽBE

*Slede video objašnjenja za zadatke pokaznih vežbi*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 7

### Individualne vežbe #12

#### ZADACI INDIVIDUALNIH VEŽBI

*Zadaci individualnih vežbi odnosi se na linearu i logističku regresiju i rade se okvirno po 45 min*

##### **Zadatak #1 (45 minuta)**

Učitati datoteku **LinReg\_ocene.csv** i istrenirati model linearne regresije tako da ulazni podaci budu kolone:

- samo kolona '**cs101\_ocena**'
- samo kolona '**cs102\_izostanci**'
- kolone '**cs101\_ocena**' i '**cs102\_izostanci**'

Izlazni podaci u svakom od slučaja jeste kolona '**cs102\_ocena**' i njene vrednosti treba da se estimiraju.

Istrenirati modele linearne regresije (koristiti 75%, a zatim 90% skupa za trening skup) za svaku od traženih kolona, naći broj grešaka i tačnost za svaki, i zaključiti koji je model najbolje koristiti.

Kada se koristi samo jedna kolona, nacrtati i grafike estimiranih vrednosti naspram pravih vrednosti.

##### **Zadatak #2 (45 minuta)**

Učitati datoteku **LogReg\_ocene.csv** i istrenirati model logističke regresije tako da ulazni podaci budu kolone:

- samo kolona '**broj\_polozenih\_ispita**',
- samo 'kolona '**cs101\_izostanci**',
- samo 'kolona '**cs101\_polozen**',
- kolone '**broj\_polozenih\_ispita**' i '**cs101\_izostanci**'
- kolone '**broj\_polozenih\_ispita**', '**cs101\_izostanci**', '**cs101\_polozen**', '**cs102\_izostanci**'

Izlazni podaci u svakom od slučaja jeste kolona '**cs102\_polozen**' i njene vrednosti treba da se estimiraju.

Istrenirati modele logističke regresije (koristiti 75%, a zatim 90% skupa za trening skup) za svaku od traženih kolona, naći broj grešaka i tačnost za svaki, i zaključiti koji je model najbolje koristiti.

# VREDNOSTI CSV DATOTEKA ZA ZADATKE INDIVIDUALNIH VEŽBI.

*U nastavku su polovične vrednosti CSV datoteka (cele datoteke su u Dodatnim aktivnostima)*

## **Vrednosti za Zadatak #1**

```
cs101_ocena,cs102_izostanci,cs102_ocena
9,0,9
10,2,10
10,0,10
10,0,10
10,0,10
9,1,9
10,0,10
10,0,10
10,0,10
9,1,9
10,0,10
10,0,10
10,0,10
9,0,9
10,0,10
9,0,9
10,0,10
10,1,10
9,0,9
10,0,10
9,0,9
10,2,9
10,0,10
9,2,9
10,0,10
10,0,10
10,2,9
9,0,9
10,0,10
9,0,9
10,0,10
9,0,9
10,2,9
9,0,9
10,0,10
9,0,9
10,0,10
10,0,10
10,0,10
9,1,9
10,0,10
10,0,10
10,0,10
9,0,9
```

10,0,10  
9,0,9  
10,0,10  
9,0,9  
10,0,10  
10,0,10  
10,0,10  
9,0,9  
10,0,10  
10,0,10  
9,0,9  
9,0,9  
10,0,10  
9,0,9  
10,0,10  
10,0,9  
10,0,10  
9,0,9  
10,0,10  
10,0,10  
9,0,9  
10,0,10  
10,0,10  
9,0,9  
9,0,9  
10,1,10  
10,0,10  
10,0,10  
10,0,10  
10,0,10  
10,0,10  
10,0,10  
9,0,9  
10,1,10  
10,0,10  
10,1,9  
10,0,10  
10,0,10  
10,0,9  
10,0,10  
10,0,10  
10,0,9  
10,0,10  
10,1,10  
10,0,10  
10,0,9  
10,0,10  
10,0,10  
10,0,10  
10,0,10  
10,0,10

10,0,10  
10,0,10  
10,1,10  
10,1,10  
8,2,8  
8,2,8  
8,2,8  
8,1,8  
8,2,8  
7,1,7  
9,1,9  
8,1,8  
8,1,8  
7,2,7  
8,2,8  
7,1,7  
8,1,8  
8,2,8  
8,2,8  
6,1,6  
8,2,8  
6,1,6  
7,1,7  
9,1,9  
8,2,8  
8,2,8  
8,2,7  
8,2,8  
8,2,7  
7,3,7  
8,1,8  
8,1,7  
8,1,8  
8,1,8  
8,3,8  
6,2,7  
7,2,7  
7,1,7  
8,2,8  
8,3,8  
8,2,8  
9,2,9  
8,3,8  
8,1,8  
8,1,8  
7,2,7  
7,2,7  
9,2,9  
8,2,8  
8,2,8  
7,3,7  
7,2,7  
8,1,7

8,2,8  
7,3,7  
8,2,8  
8,2,8  
8,2,8  
8,2,8  
8,2,8  
7,2,7  
8,2,8  
7,2,7  
8,2,8  
8,2,8  
8,2,8  
7,2,7  
7,2,7  
8,2,8  
8,2,8  
7,2,7  
8,3,8  
8,2,8  
8,2,8  
7,2,7  
7,2,7  
8,3,7  
9,2,9  
8,2,8  
7,2,7  
8,2,8  
7,2,7  
8,2,8  
8,2,8  
9,3,9  
8,3,8  
7,2,7  
8,2,8  
8,2,8  
8,2,8  
8,2,8  
7,2,7  
8,3,8  
7,2,7  
7,3,7  
8,2,8  
8,2,8  
7,2,8  
8,2,8  
8,2,8  
8,2,8  
8,2,8  
7,2,7  
7,4,7  
6,3,6

5,4,5  
7,4,7  
7,3,7  
7,3,7  
7,3,7  
6,2,6  
7,4,7  
7,0,7  
7,2,7  
7,4,7  
7,2,7  
7,4,7  
7,2,7  
7,1,7  
7,2,7  
7,2,7  
6,4,6  
7,4,6  
7,4,7  
8,2,8  
6,1,6  
7,2,7  
7,1,7  
7,3,7  
7,4,7  
5,4,6  
5,4,5  
7,2,6  
6,1,6  
7,2,7  
5,2,6  
7,4,6  
6,2,6  
6,3,6  
7,1,7  
5,4,5  
5,4,6  
5,3,5  
5,4,5  
5,4,5  
6,3,6  
7,4,7  
7,2,7  
5,4,5  
5,4,5  
5,3,6  
7,3,7  
7,3,7  
5,1,5  
6,4,6  
7,3,7  
6,4,6  
6,1,6

5,1,6  
6,2,6  
5,4,5  
5,1,5  
6,2,6  
5,4,5  
7,2,7  
7,4,7  
5,4,5  
7,2,7  
7,4,7  
7,2,7  
7,4,7  
7,4,6  
7,2,7  
7,4,6  
7,3,7  
7,4,6  
7,2,7  
7,4,7  
7,3,7  
6,2,6  
7,4,7  
7,4,7  
7,3,6  
7,3,7  
7,4,7  
7,2,7  
6,4,6  
7,3,7  
7,4,7  
6,2,6  
7,4,7  
7,4,7  
7,4,7  
7,4,7  
6,2,6  
7,4,7  
7,4,7  
7,1,7  
6,4,6  
7,2,7  
7,4,7  
6,4,6  
6,3,6  
7,4,7  
7,4,7  
7,4,7  
7,4,7  
7,3,7  
7,4,7  
7,4,7

7,4,7  
7,4,7  
7,4,7  
7,4,6  
7,3,6  
7,4,7  
7,3,6  
7,4,7  
7,2,7  
7,4,6  
7,3,6  
7,4,7  
7,4,7  
7,2,7  
7,3,7  
7,3,7  
7,4,7  
6,4,6  
7,4,7  
7,4,7  
7,4,7  
6,3,6  
7,1,7  
7,4,7  
7,4,7  
7,1,7  
6,4,6  
6,4,6  
7,4,7  
7,4,7  
7,1,6  
7,0,7  
7,1,7  
6,3,6  
5,4,5  
7,4,7  
7,1,7  
7,1,7  
7,3,7  
7,4,7  
6,4,6  
7,4,7  
7,3,7  
6,2,6  
7,4,7  
6,4,6  
7,4,7  
7,1,7  
7,4,6  
7,3,7  
7,4,7  
6,4,6  
6,4,6

7,4,7  
7,4,7  
7,3,7  
7,4,7  
7,4,7  
6,1,6  
6,4,6  
7,4,7  
6,4,6  
6,4,6  
7,4,7  
5,2,5  
6,4,6  
7,2,7  
6,2,6  
6,4,6  
7,4,7  
5,4,5  
7,1,7  
6,4,6  
6,1,6  
6,7,6  
7,7,7  
6,4,6  
6,3,6  
7,7,7  
7,7,6  
7,2,7  
7,4,7  
6,4,6  
7,7,7  
7,2,7  
7,4,7  
7,1,7  
7,7,7  
7,7,7  
6,4,6  
6,4,6  
7,7,7  
5,5,5  
5,2,6  
5,5,5  
5,5,5  
5,2,5  
5,7,5  
5,3,6  
5,5,5  
5,5,5  
5,3,5  
5,2,5  
5,2,5  
5,2,5  
5,5,5

5,5,5  
5,5,5  
5,3,5  
5,2,5  
5,2,6  
5,2,5  
5,5,5  
5,2,5  
5,3,6  
5,3,6  
5,5,5  
5,2,5  
5,5,5  
5,2,6  
5,2,5  
5,2,5  
5,3,5  
5,5,5  
5,5,5  
5,2,5  
5,5,5  
5,5,5  
5,5,5  
5,2,5  
5,5,5  
5,5,5  
5,2,5  
5,5,5  
5,5,5  
5,2,5  
5,4,5  
5,5,5  
5,5,5  
5,4,5  
5,5,6  
5,5,5  
5,2,6  
5,2,5  
5,5,5  
5,1,5  
5,5,6  
5,5,6  
5,1,6  
5,7,5  
5,7,6  
5,2,5  
5,5,5  
5,4,6

6,5,6  
6,2,6  
5,4,6  
6,4,6  
6,5,6  
5,7,5  
5,5,5  
6,2,6  
6,5,6  
6,7,6  
5,4,5  
6,7,6  
6,7,6  
5,5,5  
5,5,5  
6,5,6  
5,7,5  
6,2,6  
5,3,5  
6,5,6  
5,5,6  
5,5,5  
5,3,5  
6,5,6  
6,7,6  
5,3,5  
5,7,5  
6,5,6  
5,5,5  
5,3,5  
6,3,6  
5,7,5  
5,5,5

### Vrednosti za Zadatak #2

broj\_polozenih\_ispita,cs101\_izostanci,cs101\_polozen,cs101\_ocena,cs102\_izostanci,cs102\_polozen,cs102\_ocena  
11,7,0,5,4,1,10  
11,11,0,5,5,1,8  
3,12,1,6,10,0,5  
8,11,1,9,12,1,7  
7,10,0,5,9,0,5  
7,2,1,9,0,1,6  
4,8,1,9,12,1,10  
1,13,1,8,3,1,7  
11,13,1,6,12,1,7  
3,9,0,5,6,0,5  
13,8,0,5,15,1,9  
1,15,1,9,11,1,6  
9,4,1,8,9,1,8  
8,13,0,5,5,1,9  
7,6,0,5,14,1,6

2,8,0,5,5,1,6  
6,10,1,7,3,1,8  
10,0,1,10,4,1,9  
9,8,1,8,2,1,10  
8,4,1,7,2,1,10  
0,2,1,7,11,1,7  
10,5,0,5,6,0,5  
6,7,0,5,13,0,5  
8,12,0,5,12,0,5  
2,1,0,5,7,1,9  
3,1,1,7,8,1,8  
9,9,1,7,0,1,6  
2,4,1,7,14,0,5  
11,8,1,7,14,1,8  
1,3,1,10,0,1,8  
9,15,1,7,12,1,7  
9,0,1,7,0,1,10  
9,2,1,10,6,1,10  
11,13,0,5,9,1,9  
0,9,1,9,10,1,7  
0,10,0,5,4,1,7  
0,14,1,9,9,0,5  
8,6,1,9,9,1,8  
1,3,0,5,4,1,8  
6,3,1,9,11,1,6  
10,12,0,5,1,1,6  
3,14,0,5,15,1,6  
13,11,1,9,8,1,9  
9,15,0,5,10,0,5  
12,7,1,9,3,1,10  
10,6,0,5,3,1,9  
2,8,0,5,0,1,7  
9,4,1,6,15,1,8  
0,0,1,9,5,1,7  
2,8,1,6,15,1,8  
6,8,1,7,0,0,5  
9,15,1,8,11,0,5  
11,4,1,7,11,1,9  
12,13,1,9,2,1,10  
2,11,1,7,5,1,6  
10,11,1,9,0,1,6  
3,5,1,9,1,1,9  
6,2,1,8,3,1,9  
12,2,1,9,3,1,6  
12,1,1,6,8,1,7  
7,7,0,5,3,1,9  
11,1,1,6,11,1,8  
8,0,1,6,6,1,8  
6,5,1,9,13,1,8  
7,15,1,7,14,1,9  
0,1,1,8,11,1,8  
4,4,1,6,9,1,9  
8,9,1,6,13,1,7

7,4,1,6,10,0,5  
3,0,1,8,11,1,7  
5,2,1,10,11,1,7  
13,4,1,10,15,1,9  
9,14,1,8,8,0,5  
10,10,1,6,3,1,8  
5,7,1,9,7,1,7  
11,14,1,7,8,0,5  
5,10,1,8,9,0,5  
2,0,1,10,14,1,6  
1,15,1,8,0,1,9  
3,0,0,5,3,1,10  
12,2,1,10,6,1,7  
8,5,0,5,3,1,7  
7,7,1,6,4,0,5  
4,5,1,9,13,1,10  
1,5,0,5,3,0,5  
5,5,1,7,9,1,8  
13,1,1,7,5,1,9  
7,5,1,7,10,1,6  
0,15,1,9,7,1,10  
2,8,0,5,6,1,8  
7,3,1,9,0,1,8  
5,12,0,5,7,0,5  
10,15,0,5,1,1,8  
7,7,1,6,7,1,7  
10,4,1,10,6,1,7  
5,6,1,9,11,1,10  
10,11,0,5,4,1,9  
13,10,1,7,13,0,5  
9,13,0,5,8,1,6  
3,15,0,5,10,1,6  
7,10,1,9,1,1,9  
11,8,1,7,0,0,5  
9,10,0,5,10,0,5  
4,3,1,8,4,1,10  
0,12,1,6,11,0,5  
4,4,1,8,11,1,9  
4,1,1,7,8,0,5  
13,10,1,8,14,1,10  
13,14,1,7,4,1,6  
9,2,1,10,6,1,10  
13,6,1,7,10,0,5  
4,12,1,7,9,1,8  
6,8,0,5,3,1,8  
13,2,0,5,7,1,9  
5,6,1,9,0,1,10  
7,10,1,9,3,1,9  
10,9,0,5,5,0,5  
9,13,0,5,9,1,6  
5,4,1,6,15,1,7  
8,8,1,6,7,1,8  
10,14,0,5,1,1,9

6,9,1,6,10,0,5  
6,4,1,8,2,1,6  
5,0,1,10,14,1,10  
1,10,1,8,10,1,6  
0,0,1,8,8,1,8  
13,2,1,6,0,1,9  
13,6,1,9,9,1,7  
11,14,0,5,11,1,9  
0,10,0,5,7,0,5  
0,5,0,5,8,0,5  
12,5,1,6,7,0,5  
12,11,0,5,8,1,7  
3,5,0,5,5,0,5  
1,13,1,8,4,1,9  
13,9,1,6,5,1,7  
2,4,1,10,13,1,10  
0,15,0,5,5,1,8  
5,8,1,8,2,1,10  
8,2,0,5,7,1,8  
0,11,1,8,14,1,10  
9,2,1,8,5,1,9  
6,12,1,7,4,1,6  
3,9,0,5,0,1,10  
4,4,1,8,9,1,7  
9,8,1,7,4,1,8  
11,8,1,6,9,1,6  
2,4,1,8,8,1,10  
5,10,0,5,12,0,5  
12,7,0,5,8,0,5  
9,0,1,7,11,1,8  
11,10,1,6,3,1,7  
9,13,1,7,1,1,9  
0,8,1,7,12,1,8  
9,8,1,9,13,1,10  
8,5,1,6,15,1,8  
10,3,1,6,8,1,6  
8,4,1,10,13,1,10  
4,2,1,8,4,1,8  
10,9,1,7,0,1,6  
2,14,1,6,4,0,5  
12,10,1,7,3,0,5  
0,6,1,8,6,1,7  
3,10,1,7,15,1,8  
9,6,1,6,1,1,8  
3,5,1,8,0,1,10  
11,3,1,9,11,1,6  
5,8,1,6,15,0,5  
2,14,0,5,11,1,9  
4,13,1,7,10,1,7  
1,12,1,8,2,1,10  
3,15,1,8,3,1,10  
1,7,1,7,9,0,5  
13,9,1,9,7,1,6

12,3,1,8,10,1,9  
2,2,1,7,14,1,8  
1,1,1,6,10,1,7  
7,4,1,10,13,1,9  
11,12,1,7,7,1,9  
6,5,1,9,4,1,7  
4,7,1,6,9,0,5  
2,6,0,5,12,0,5  
9,13,1,6,13,1,9  
9,0,1,8,5,1,10  
0,10,0,5,10,1,6  
0,3,1,9,3,1,7  
6,14,1,9,10,1,9  
4,8,0,5,6,1,9  
9,5,0,5,11,1,6  
8,3,1,8,1,1,10  
2,7,0,5,11,1,7  
0,0,1,9,13,1,6  
10,11,0,5,13,0,5  
5,7,1,8,7,1,7  
5,2,1,10,9,1,6  
12,9,0,5,13,1,6  
0,10,1,7,8,1,8  
9,8,1,6,9,1,8  
5,6,1,6,7,0,5  
10,0,0,5,13,1,6  
8,7,1,6,4,1,6  
1,1,1,7,13,1,6  
9,13,0,5,5,1,7  
10,1,1,6,15,1,9  
12,12,1,8,5,0,5  
3,9,0,5,3,1,8  
9,14,0,5,1,1,7  
4,2,1,6,5,1,8  
6,6,0,5,3,1,7  
6,6,1,8,2,1,7  
4,15,1,9,4,1,8  
7,14,1,7,7,1,8  
5,9,0,5,0,0,5  
2,9,0,5,5,0,5  
4,1,1,7,15,1,6  
12,3,1,9,0,1,7  
10,3,0,5,7,1,8  
12,3,1,6,15,1,6  
2,7,0,5,13,1,9  
9,0,1,10,4,1,10  
12,13,1,8,6,1,9  
4,4,1,10,14,1,10  
3,0,1,8,0,1,6  
6,5,1,9,0,1,9  
8,13,0,5,11,1,6  
5,0,1,6,12,1,9  
6,2,1,6,6,1,6

7,13,1,7,10,0,5  
13,2,1,6,8,0,5  
2,14,0,5,2,1,6  
13,11,1,9,5,1,8  
1,0,1,6,1,1,9  
3,10,1,8,14,0,5  
8,14,0,5,14,1,9  
5,6,0,5,9,1,9  
12,9,1,9,1,1,6  
3,3,0,5,10,0,5  
1,15,1,7,7,1,7  
13,7,1,8,14,1,8  
1,4,1,6,5,0,5  
1,8,1,9,8,1,7  
0,3,1,7,3,1,7  
11,13,1,9,6,1,9  
5,10,1,7,4,1,6  
6,2,0,5,13,1,6  
1,0,1,6,1,0,5  
12,2,1,7,10,1,7  
13,15,1,7,4,1,10  
8,5,1,6,1,1,10  
4,9,0,5,15,1,6  
9,9,1,6,11,0,5  
12,3,1,7,10,1,9  
12,12,0,5,0,0,5  
5,4,0,5,13,0,5  
9,14,1,6,7,1,9  
13,2,0,5,14,1,6  
13,9,1,8,11,1,7  
10,15,1,6,15,1,8  
9,12,0,5,1,1,6  
0,11,1,8,9,1,6  
2,1,1,7,1,1,8  
13,0,1,10,8,1,6  
10,3,1,8,14,1,8  
5,5,0,5,10,1,7  
6,5,0,5,11,0,5  
10,3,1,7,0,1,8  
12,9,1,8,14,1,8  
10,0,1,6,0,1,8  
1,6,1,8,13,1,8  
13,2,1,6,3,0,5  
6,8,0,5,10,1,6  
9,7,1,7,4,1,8  
10,2,0,5,0,1,10  
12,4,1,6,4,1,7  
8,15,1,7,14,0,5  
5,8,1,8,3,1,6  
10,6,1,8,7,1,8  
6,6,1,8,14,1,8  
1,11,1,8,1,1,10  
10,10,1,6,14,0,5

9,7,0,5,6,1,7  
13,12,1,8,15,1,9  
0,4,1,8,6,1,10  
7,5,1,7,8,1,8  
6,4,0,5,10,1,8  
11,10,1,8,15,1,9  
2,7,1,8,14,1,6  
10,8,1,9,14,1,9  
10,0,1,8,8,1,6  
1,4,1,9,11,1,7  
1,11,1,6,10,0,5  
12,9,0,5,12,0,5  
7,10,0,5,6,1,6  
8,10,1,9,6,1,9  
11,5,1,8,0,1,10  
3,7,1,8,0,1,7  
10,8,1,7,2,1,8  
6,14,1,6,10,1,7  
10,15,0,5,4,0,5  
4,10,0,5,11,0,5  
8,4,1,8,7,1,6  
0,11,1,7,6,1,8  
4,7,1,9,5,1,7  
3,13,0,5,2,1,8  
3,12,1,9,6,1,8  
3,1,1,9,14,1,10  
2,10,0,5,12,0,5  
7,15,1,8,4,1,10  
13,9,0,5,0,0,5  
12,9,1,9,14,1,8  
7,3,1,6,13,0,5  
6,1,1,10,11,1,10  
2,9,1,8,15,1,9  
5,8,1,6,11,1,6  
10,8,0,5,7,1,7  
13,1,1,6,0,1,9  
8,6,1,8,12,1,8  
2,6,1,6,4,1,8  
7,5,0,5,15,1,7  
2,0,1,7,10,0,5  
6,6,1,6,5,1,9  
7,12,1,8,5,1,10  
9,13,1,6,0,1,9  
12,13,0,5,5,0,5  
0,5,1,9,5,1,9  
11,12,0,5,4,1,8  
8,12,1,9,15,1,10  
3,1,1,10,3,1,10  
8,3,1,8,10,1,6  
4,11,0,5,0,1,10  
0,11,0,5,2,1,7  
0,13,1,8,11,0,5  
3,12,1,8,5,1,7

11,5,1,8,1,1,9  
4,8,1,7,4,0,5  
10,11,1,9,5,1,7  
8,9,1,9,7,1,9  
9,3,1,10,3,1,9  
7,1,1,7,6,0,5  
10,13,1,8,14,1,9  
5,11,1,7,2,1,10  
7,14,1,9,8,1,6  
12,0,1,9,7,1,9  
11,0,1,10,9,1,9  
8,11,0,5,5,1,7  
6,2,1,6,9,1,6  
5,10,1,8,5,1,10  
12,15,1,7,6,0,5  
2,1,0,5,8,1,9  
4,3,1,6,10,1,9  
1,12,1,8,0,1,7  
8,3,1,6,7,1,6  
0,5,0,5,9,0,5  
5,15,1,8,11,1,6  
7,0,1,10,7,1,7  
6,15,0,5,15,1,7  
3,4,1,9,6,1,10  
1,12,0,5,14,0,5  
3,13,1,7,5,0,5  
1,15,1,6,3,0,5  
2,1,1,9,2,1,9  
11,4,1,8,1,1,10  
3,7,0,5,8,0,5  
1,9,0,5,1,0,5  
9,12,1,8,7,1,10  
8,7,1,8,10,0,5  
10,8,0,5,7,1,7  
12,10,0,5,10,1,6  
11,7,0,5,14,0,5  
13,11,0,5,3,0,5  
7,4,1,10,2,1,10  
4,1,1,7,9,0,5  
2,0,1,7,12,1,6  
0,11,1,6,3,1,8  
1,3,1,8,7,1,10  
4,11,0,5,5,0,5  
7,13,0,5,1,1,6  
7,5,0,5,3,1,9  
8,9,1,7,3,1,7  
9,10,0,5,2,0,5  
13,5,1,9,11,1,7  
9,1,1,10,12,1,9  
8,2,1,9,5,1,9  
4,4,0,5,15,1,7  
11,14,0,5,11,1,9  
6,14,1,8,3,1,9

2,3,1,7,1,1,8  
11,14,1,6,8,0,5  
6,2,0,5,2,1,6  
2,12,1,9,9,1,6  
7,11,1,6,2,1,8  
6,7,0,5,15,0,5  
6,10,1,8,3,1,9  
4,6,1,8,10,1,6  
10,7,1,7,10,1,7  
8,5,1,6,9,1,9  
4,2,1,9,10,1,9  
10,13,1,8,6,0,5  
8,8,1,7,4,1,6  
13,4,1,9,13,1,9  
9,12,1,8,8,1,7  
4,1,1,6,15,1,7  
10,13,0,5,5,1,8  
6,9,1,6,12,1,7  
8,3,1,7,1,1,9  
13,5,1,9,11,1,6  
12,13,1,9,13,1,7  
11,8,1,7,9,1,9  
3,7,0,5,0,1,8  
11,5,1,7,1,0,5  
7,0,1,6,14,1,9  
3,3,1,10,9,1,7  
4,6,1,8,0,1,10  
4,0,0,5,4,0,5  
3,11,1,9,14,1,9  
13,14,1,7,8,0,5  
2,8,1,6,7,1,6  
2,6,0,5,5,0,5  
13,9,1,6,3,1,7  
2,0,1,8,5,0,5  
4,2,1,6,11,0,5  
6,6,0,5,3,0,5  
6,12,1,6,8,1,6  
0,3,1,8,11,1,10  
0,9,1,8,14,1,7  
6,15,1,9,9,1,9  
0,3,1,10,11,1,6  
0,5,1,8,8,1,8  
9,0,1,9,4,1,9  
10,1,1,6,4,1,6  
8,6,0,5,0,1,6  
10,5,1,9,15,1,10  
11,0,1,7,2,1,10  
12,6,1,6,3,1,8  
5,5,0,5,15,0,5  
10,5,1,7,2,1,8  
13,10,1,9,14,1,8  
2,8,1,6,7,0,5  
4,15,1,6,8,1,6

12,4,1,6,10,1,8  
0,5,1,6,7,1,8  
13,15,0,5,9,0,5  
6,6,0,5,3,0,5  
7,9,0,5,0,1,8  
1,6,1,6,5,1,7  
13,3,1,10,5,1,9  
0,15,1,9,7,1,6  
5,10,1,9,1,1,9  
3,4,1,7,0,1,8  
3,9,0,5,4,1,9  
8,10,1,9,7,0,5  
6,1,1,6,10,1,9  
2,14,1,9,9,1,6  
12,15,1,8,3,1,9  
8,12,1,7,5,0,5  
2,14,1,9,3,1,10  
10,11,1,6,9,0,5  
2,0,1,7,15,1,8  
5,0,1,7,6,1,8  
1,3,1,6,0,1,7  
0,7,0,5,0,0,5  
4,1,1,8,5,1,9  
0,1,1,10,2,1,10  
8,2,0,5,0,0,5  
9,12,1,9,5,1,9  
0,10,1,7,8,1,9  
9,5,1,6,4,1,6  
6,7,1,7,9,1,7  
1,12,1,9,4,1,7  
12,5,1,6,9,1,7  
5,3,1,8,11,0,5  
10,5,0,5,5,1,6  
12,11,1,6,14,1,6  
4,4,1,7,9,0,5  
4,12,0,5,12,1,7  
8,5,1,7,12,1,6  
5,8,0,5,7,1,9  
7,15,1,9,2,1,8  
9,4,1,8,15,1,8  
10,5,1,8,2,1,10  
7,2,1,7,8,1,9  
3,10,1,8,6,1,7  
3,10,0,5,11,0,5  
7,8,0,5,13,1,8  
7,4,1,7,2,1,6  
9,11,1,9,6,1,6  
2,11,1,6,11,1,9  
5,13,0,5,13,1,7  
9,8,0,5,9,1,9  
10,11,1,9,8,1,10  
5,9,1,9,13,1,7  
11,12,1,6,0,1,9

2,6,0,5,1,1,9  
3,4,1,6,3,1,9  
5,0,1,10,10,1,9  
6,10,0,5,4,1,10  
0,4,0,5,0,0,5  
7,11,1,9,2,1,6  
12,9,1,6,9,1,6  
7,13,1,6,1,1,7

## ✓ Poglavlje 8

### Domaći zadatak

#### TEKST DOMAĆEG ZADATKA #12

*Domaći zadatak #12 se radi okvirno 4.5 h.*

##### Zadatak #1

Napisati program koji će generisati skup podataka koji će se koristiti za trening i test skupove. Podaci jesu sledeći:

- Kolona #1: Ime: **cs\_101\_ocena** (vrednost od 5 do 10)
- Kolona #2: Ime: **it\_101\_ocena** (vrednost od 5 do 10)
- Kolona #3: Ime: **ma\_101\_ocena** (vrednost od 5 do 10)
- Kolona #4: Ime: **cs\_115\_izostanci** (vrednosti od 0 do 15)
- Kolona #5: Ime **cs\_115\_položen** (vrednosti 0 ili 1)
- Kolona #6: Ime: **cs\_115\_ocena** (vrednosti od 5 do 10)

Kolone popuniti nasumično, i onda promeniti sledeće (ne ručno, već kroz kod):

Ukoliko je vrednost kolone #1 između 8 i 10, povećati vrednost kolone #6 za jedan. Ukoliko je vrednost kolone #4 između 5 i 12, smanjiti vrednost kolone #6 za jedan, a između 13 i 15, smanjiti vrednost kolone #6 za 2. Ukoliko je vrednost kolone #3 između 9 i 10, povećati vrednost kolone #6 za 1. Svuda gde je vrednost kolone #5 nula, vrednost kolone #6 je 5. Obratiti pažnju da pri promeni vrednosti kolona da minimalne i maksimalne vrednosti ostanu nepromenjene.

Svaka kolona ima 500 redova. Sačuvati tabelu kao **dataset.csv**.

##### Zadatak #2

Učitati skup **dataset.csv** zadatka #1.

Istrenirati model linearna regresije da estimira vrednosti kolone **cs\_115\_ocena**, ako su ulazni podaci:

- Samo kolona **cs\_101\_ocena**,
- Samo kolona **cs\_115\_izostanci**,
- Sve kolone (osim **cs\_115\_ocena** i **cs\_115\_položen**, naravno)

Istrenirati modele **linearne regresije** (koristiti 75%, a zatim 90% skupa za trening skup) za svaku od traženih kolona, naći broj grešaka i tačnost za svaki, i zaključiti koji je model najbolje koristiti. Kada se koristi samo jedna kolona, nacrtati i grafike estimiranih vrednosti naspram pravih vrednosti.

### Zadatak #3

Uraditi isto kao i kod **zadataka #2**, samo je model **logistička regresija**, a estimiraju se vrednosti kolone **cs\_115\_polozen**.

## PREDAJA DOMAĆEG ZADATKA

*Domaći zadaci se predaju 7 dana nakon predavanja za 100% poena (kod tradicionalnih studenata).*

#### **Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije 7 dana nakon predavanja, za 100% poena. Nakon toga poeni se umanjuju za 50%.

#### **Internet studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganje ispita. Domaći zadaci se brane!

Domaći zadatak poslati dr Nemanji Zdravkoviću: nemanja.zdravkovic@metropolitan.ac.rs

Obavezno koristiti uputstvo za izradu domaćeg zadatka.

Uz .doc dokument (koji treba sadržati i screenshot svakog urađenog zadatka kao i komentare za zadatak), poslati i izvorne i dodatne datoteke.

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Zaključak lekcije #12*

#### **Zaključak**

U ovoj lekciji bilo je reči o osnovama veštačke inteligencije i mašinskog učenja. U uvodnom delu definisan je pojam veštačke inteligencije, kao i primene iste. Zatim je data definicija mašinskog učenja kao jednog od polja veštačke inteligencije.

Opisana je razlika između nadgledanog, nenadgledanog i pojačanog učenja kao osnovne podele mašinskog učenja.

Obrađeni su osnovni algoritmi nadgledanog učenja, i to linearna regresija i logistička regresija. Opisan je algoritam opadajućeg gradijenta kao algoritma koji umanjuje grešku pri linearnoj i logističkoj regresiji.

#### **Literatura**

- Zsolt Nagy, veštačke inteligencije i mašinskog učenja, Kompjuter biblioteka, 2019. (prevod iste knjige Packt Publishing-a)
- David Beazley, Brian Jones, Python Cookbook: Recipes for Mastering Python 3, 3rd edition, O'Reilly Press, 2013.
- Mark Lutz, Learning Python, 5th Edition, O'Reilly Press, 2013.
- Andrew Bird, Lau Cher Han, et. al, The Python Workshop, Packt Publishing, 2019.
- Al Sweigart, Automate the boring stuff with Python, 2nd Edition, No Starch Press, 2020.





## CS324 - SKRIPTING JEZICI

### Python radna okruženja

Lekcija 13

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 13

### **PYTHON RADNA OKRUŽENJA**

- ✓ Python radna okruženja
- ✓ Poglavlje 1: Uvod u virtuelna okruženja
- ✓ Poglavlje 2: Mikroservisna arhitektura aplikacije
- ✓ Poglavlje 3: Prednosti mikroservisne arhitekture
- ✓ Poglavlje 4: Nedostaci mikroservisne arhitekture
- ✓ Poglavlje 5: Flask mikroservis sa razvoj veb aplikacija
- ✓ Poglavlje 6: Pokazne vežbe #13
- ✓ Poglavlje 7: Individualne vežbe #13
- ✓ Poglavlje 8: Domaći zadatak
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ✓ Uvod

# UVOD

### *Uvod u lekciju #13*

Do sada je bilo reči o Python kao jeziku za razne inženjerske i naučne svrhe, ali naravno, python se može koristiti i u komercijalnom programiranju.

Zbog toga uvodimo Flask.

Flask predstavlja jedan lagani WSGI, odnosno gateway interfejs ka veb serveru. WSGI jeste specifikacija koja opcije kako veb server komunicira sa veb aplikacijama, i kako se veb aplikacije mogu povezati da isprociraju jednan zahtev.

Flask je jedan framework za web aplikacije. Projektovanje je tako da se može brzo i lako postaviti, ali da je istovremeno skaliranje na kompleksne aplikacije takođe jednostavno. Trenutno je jedan od najpopularnijih framework-a za veb aplikacija.

Flask je specifičan da daje preporuke, ali ne zahteva nikakve zavisnosti niti specifičnu strukturu odnosno layout projekta. Ostavljeno je na programeru da izabere alate i biblioteke koje želi da koristi.

Flask podržava mnoge dodatke jer je aktivan razvoj samog *framework*-a, tako da je dodavanje novih funkcionalnosti lako.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 1

# Uvod u virtuelna okruženja

## POTREBA ZA VIRTUELnim OKRUŽENJEM

*Odvajanje virtuelnog okruženja od sistemskog omogućuje se migracija Python interpretera*

Nekada je potrebno izdvojiti radno okruženje od sistemskog okruženja, i to se postiže pomoću paketa `venv`.

Paket `venv` - `Virtual Environment` pravi virtuelno okruženje za Python u kojem nema naknadno instaliranih paketa, već su dostupni samo sistemski paketi.

Odvajanje virtuelnog okruženja od sistemskog omogućuje se migracija Python interpretera, a da se pri tom ne javljaju greške zbog nepostojecih paketa na drugom računaru.

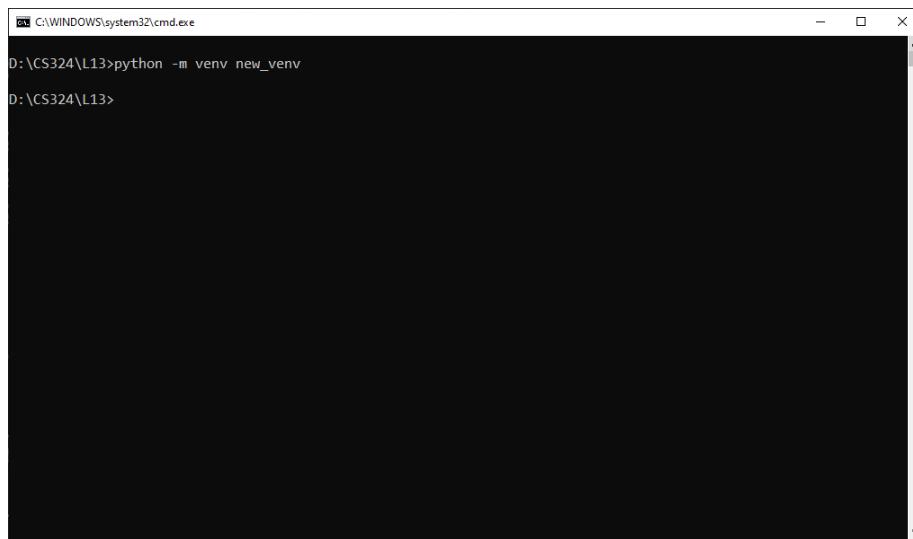
Prilikom kreacije virtuelnog okruženja, svaki put se pravi novi Python interpreter sa podrazumevanim paketima (`pip` i `setuptools`), i treba unutar tog okruženja instalirati ostale pakete koji su neophodni na Python projektu.

### Kreacija virtuelnog okruženja

Da bi se napravilo novo virtuelno okruženje, potrebno je u terminalu doći do direktorijuma u kojem želite napraviti okruženje, i ukucati sledeću komandu:

```
python -m venv new_venv
```

U ovom primeru opcija-`m` znači da se pokreće modul `venv`, a `new_venv` biće novi direktorijum unutar radnog direktorijuma gde će biti smešteno virtuelno okruženje.

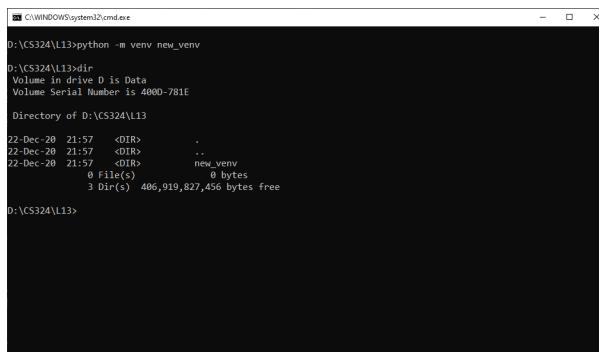


Slika 1.1 Pravljenje direktorijuma za virtuelno okruženje. [Izvor: Autor]

## AKTIVIRANJE VIRTUELNOG OKRUŽENJA

*Jednom kada se virtuelno okruženje napravi, unutar novog direktorijuma pojaviće se dodatne datoteke*

Jednom kada se virtuelno okruženje napravi, unutar novog direktorijuma pojaviće se dodatne datoteke.



Slika 1.2 Novi direktorijum sa virtuelnim okruženjem. [Izvor: Autor]

Ove datoteke zapravo predstavljaju zaseban Python interpreter, i biće iste verzije kao i sistemski interpreter, tačnije biće iste verzije kao i interpreter koji je napravio novo virtuelno okruženje.

Slika 1.3 Sadržaj direktorijuma Scripts unutar virtuelnog okruženja. [Izvor: Autor]

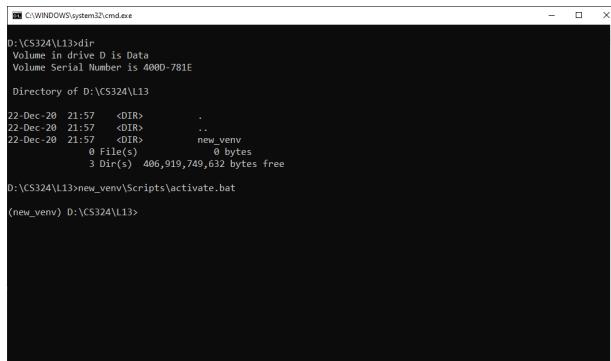
### Aktiviranje virtuelnog okruženja

Novo virtuelno okruženje aktivira se iz terminala pozivanjem datoteke **activate.bat** u direktorijumu **Scripts**.

new\_venv\Scripts\activate.bat

Virtuelno okruženje je aktivirano ukoliko pre *prompt-a* u zagradama piše ime samog virtuelnog okruženja, u primeru biće:

(new\_venv)



```
C:\WINDOWS\system32\cmd.exe
D:\CS324\L13>dir
Volume in drive D is Data
Volume Serial Number is 4000-781E

Directory of D:\CS324\L13

22-Dec-20 21:57 <DIR> .
22-Dec-20 21:57 <DIR> ..
22-Dec-20 21:57 <DIR> new_venv
0 File(s) 0 bytes
3 Dir(s) 406,919,749,632 bytes free

D:\CS324\L13>new_venv\Scripts\activate.bat

(new_venv) D:\CS324\L13>
```

Slika 1.4 Aktiviranje virtuelnog okruženja. [Izvor: Autor]

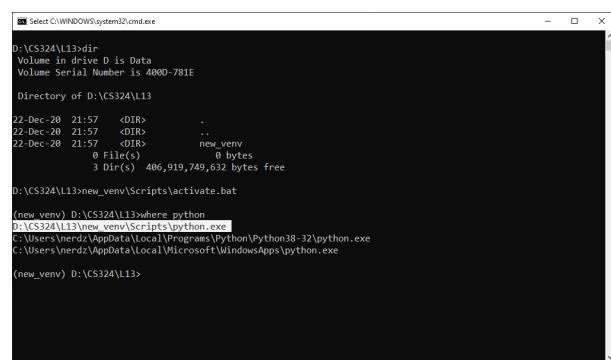
## RAD I DEAKTIVIRANJE VIRTUELNOG OKRUŽENJA

*Jednom kada se aktivira virtuelno okruženje, Python interpreter se ponaša nezavisno od sistemskog interpretera.*

Jednom kada se aktivira virtuelno okruženje, Python interpreter se ponaša nezavisno od sistemskog interpretera.

Provera interpretera se može uraditi komandom u terminalu:

where python



```
Select C:\WINDOWS\system32\cmd.exe
D:\CS324\L13>dir
Volume in drive D is Data
Volume Serial Number is 4000-781E

Directory of D:\CS324\L13

22-Dec-20 21:57 <DIR> .
22-Dec-20 21:57 <DIR> ..
22-Dec-20 21:57 <DIR> new_venv
0 File(s) 0 bytes
3 Dir(s) 406,919,749,632 bytes free

D:\CS324\L13>new_venv\Scripts\activate.bat

(new_venv) D:\CS324\L13>where python
D:\CS324\L13\new_venv\Scripts\python.exe
C:\Users\verndz\AppData\Local\Programs\Python\Python38-32\python.exe
C:\Users\verndz\AppData\Local\Microsoft\WindowsApps\python.exe

(new_venv) D:\CS324\L13>
```

Slika 1.5 Putanja do aktivnog Python interpretera. [Izvor: Autor]

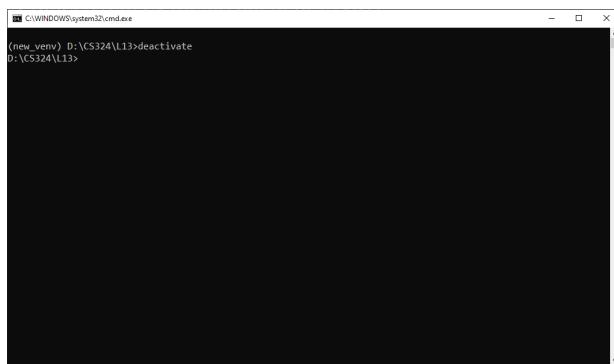
Svi paketi i moduli koji su prethodno bili instalirani neće biti dostupni, već se moraju naknadno instalirati.

Slika 1.6 Pip list komanda u novom virtuelnom okruženju. [Izvor: Autor]

## Deaktiviranje virtuelnog okruženja

Vraćanje u sistemski Python interpreter se vrši komandom

```
deactivate
```



Slika 1.7 Deaktiviranje virtuelnog okruženja. [Izvor: Autor]

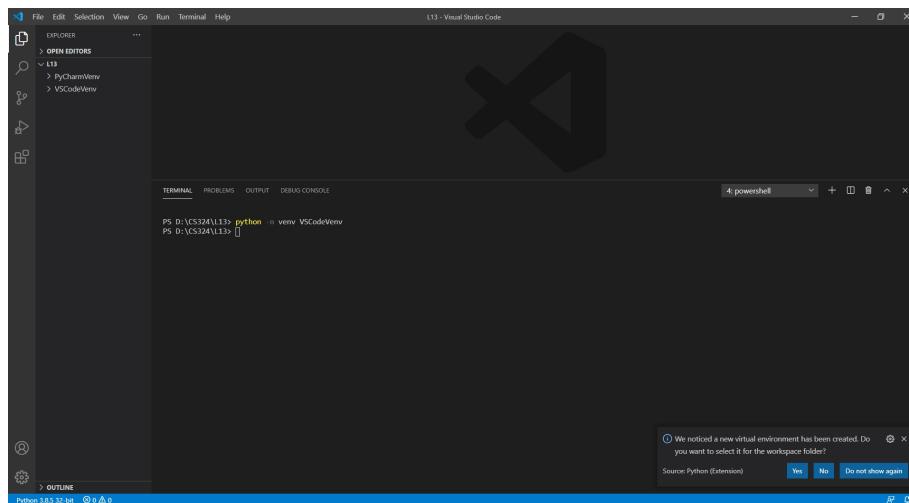
Prilikom ponovnog aktiviranja virtuelnog okruženja, ostaće instalacije paketa koji su instalirani unutar tog okruženja.

## AKTIVIRANJE VIRTUELNOG OKRUŽENJA U VISUAL STUDIO CODE

*Aktiviranje virtuelnog okruženja u Visual Studio Code-u je jednostavno kao i preko terminala.*

U **Visual Studio Code** IDE-u, kreacija virtuelnog okruženja je identična kao pri radu sa terminalom. **Visual Studio Code** odmah će prepoznati da je u pitanju novi interpreter i pitaće korisnika da li želi da se prebaci na novokreirani interpreter.

Nakon toga, potrebno je aktivirati i putem terminala, ali treba aktivirati datotekom `activate.ps1`, pošto je podrazumevani terminal **PowerShell**, a ne običan **Command Prompt**.



Slika 1.8 Kreiranje virtuelnog okruženja u VSCode. [Izvor: Autor]

#### Napomena:

Ukoliko se ne može aktivirati novo okruženje, potrebno je omogućiti pokretanje .ps1 skripti u Power Shell-u. Treba pokrenuti Power Shell sa administratorskim privilegijama i ukucati sledeće:

```
set-executionpolicy remotesigned
```

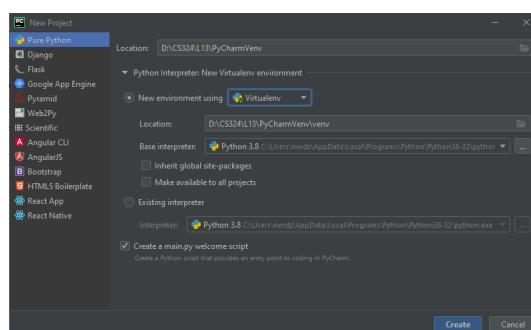
**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## AKTIVIRANJE VIRTUELNOG OKRUŽENJA U PYCHARM

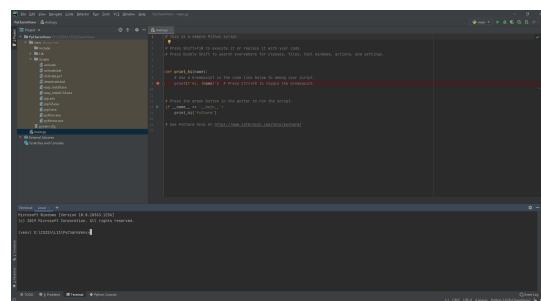
*PyCharm podrazumevano nudi opciju kreacije virtuelnog okruženja prilikom pravljenja novog projekta.*

PyCharm IDE prilikom pokretanja nudi da se napravi novi projekat sa postojećim Python interpretrom, ili sa kreacijom novog interpretara u novom virtuelnom okruženju.

Ukoliko se napravi virtuelno okruženje, ono se odmah aktivira.



Slika 1.9 Odabir sistemskog interpretara ili kreiranje virtuelnog okruženja. [Izvor: Autor]



Slika 1.10 Novo virtuelno okruženje je automatski aktivirano. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 2

# Mikroservisna arhitektura aplikacije

## UVOD U MIKROSERVISE - SOA

*Osnovni princip SOA jeste organizacija aplikacije u diskrete jedinice funkcionalnosti, kojima se može daljinski pristupiti, i koje se nezavisno ažuriraju.*

### Mikroservisna arhitektura

Pošto ne postoji zvanični standard za mikroservise, postoji mnoštvo definicija za njih.

Pojedinci često pominju servisno-orientisani arhitekturu (en. [Service-Oriented Architecture, SOA](#)) kada pokušavaju da objasne šta su mikroservisi

#### **Wikipedia:**

***SOA prethodi mikroservisima - njen osnovni princip je ideja da se organizuju aplikacije u diskrete jedinice funkcionalnosti kojima se može daljinski pristupiti da se na njih deluje i koje se nezavisno ažuriraju.***

Iako SOA jasno ističe da servisi treba da budu samostalni procesi, nije rečeno koji protokoli treba da se upotrebe za te procese za međusobnu interakciju i ostaje prilično nejasna u pogledu načina raspoređivanja i organizovanja aplikacije.

U SOA Manifestu (<http://www.soa-manifesto.org>), koji je publikovan na Vebu 2009. godine, stručnjaci koji su ga potpisali čak ne spominju da li servisi međusobno vrše interakciju pomoću mreže.

SOA servisi mogu da komuniciraju pomoću međuprocesne komunikacije (en. [Inter-Process Communication, IPC](#)), koristeći priključke (sokete, en. [sockets](#)) na istoj mašini, korišćenjem *deljene memorije* i *indirektnih redova*

poruka ili, čak, pomoću *udaljenih poziva procedura* (en. [Remote Procedure Calls, RPC](#)). Opcije su razne i na kraju SOA može da bude *sve i svašta*, sve dok ne pokrenete ceo kod aplikacije u jednom procesu.

Međutim, uobičajeno je reći da su mikroservisi specijalizacija SOA ciljeva, koji su počeli da se pojavljuju poslednjih godina, zato što ispunjavaju neke SOA ciljeve, kao što je izgradnja aplikacija pomoću samostalnih komponenata koje međusobno komuniciraju.

Ako želimo da formulšemo kompletну definiciju šta su mikroservisi, najbolji način je da prvo pogledamo kako je izgrađena većina softvera.

## MONOLITNA ARHITEKTURA

*Najpre su veb aplikacije pratile monolitnu arhitekturu.*

Pre objašnjenja mikroservisa, potrebno je objasniti pojam monolitne arhitekture (en. **monolithic application architecture**)

**Primer:**

Upotrebimo jedan veoma jednostavan primer tradicionalne monolitne aplikacije: veb sajt za rezervaciju hotela. Osim statičnog HTML sadržaja, veb sajt ima funkcije za rezervacije, koje će omogućiti korisnicima da rezervišu hotele u bilo kom gradu na svetu. Korisnici mogu da pretražuju hotele, pa da rezervišu smeštaj u njima, koristeći kreditnu karticu.

Kada korisnik izvrši pretragu na veb sajtu hotela, aplikacija prolazi kroz sledeće korake:

1. Pokreće dva SQL upita za baze podataka hotela.
2. HTTP zahtev za servis partnera je kreiran za dodavanje više hotela u listu.
3. Generisana je stranica sa rezultatima upotrebom mehanizma HTML šablona.

Od te tačke, kada korisnik pronađe odgovarajući hotel i klikne na njega da bi u njemu rezervisao smeštaj, aplikacija izvršava sledeće korake:

1. Korisnik je kreiran u bazi podataka ako je potrebno i treba da mu se proveri identitet.
2. Plaćanje se izvršava interakcijom sa veb servisom banke.
3. Aplikacija snima detalje o plaćanju u bazi podataka zbog pravnih razloga.
4. Potvrda prijema je generisana pomoću PDF generatora.
5. E-mail rekapitulacije je poslat korisniku pomoću e-mail servisa.
6. E-mail u vezi rezervacije je posleđen nezavisnom hotelu pomoću e-mail servisa.
7. Unos baze podataka je dodat za praćenje rezervacije.

Ovaj proces je, naravno, pojednostavljeni model, ali je prilično realističan.

Aplikacija vrši interakciju sa bazom podataka koja sadrži informacije o hotelu, detalje o rezervaciji i naplati, informacije o korisniku i tako dalje. Takođe vrši interakciju sa eksternim servisima za slanje e-mailova, izvršavanje plaćanja i dodavanje naziva za više hotela.

U **LAMP** (Linux-Apache-MySQL-PHP) arhitekturi svaki ulazni zahtev generiše niz SQL upita u bazi podataka i nekoliko mrežnih poziva ka eksternim servisima, a zatim server generiše HTML odgovor, koristeći mehanizam šablona.

## OPIS MONOLITNE ARHITEKTURE

*Najveća prednost je što se celu aplikaciju nalazi u jednoj osnovi koda.*

Ovo je tipična monolitna aplikacija, koja ima mnogo očiglednih prednosti.

Najveća prednost je što se celu aplikaciju nalazi u **jednoj osnovi koda**, pa, kada započne proces kodiranja projekta, sve postaje jednostavnije. Izgradnja dobrog testa je jednostavna i

kod može da se organizuje na jasan i strukturirani način unutar njegove osnove. Skladištenje svih podataka u jednu bazu podataka takođe pojednostavljuje razvoj aplikacije. Mogu da se podešavaju model podataka i način na koji će ih kod zatražiti.

Raspoređivanje je, takođe, jednostavno: možemo da označimo osnovu koda, da izgradimo paket i da ga negde pokrenemo. Da bismo skalirali aplikaciju, možemo da pokrenemo nekoliko instanci aplikacije za rezervacije i nekoliko baza podataka sa postavljenim mehanizmima

za repliciranje. Ako aplikacija ostane mala, ovaj model funkcioniše dobro i jedan tim ga može jednostavno održavati.

Međutim, projekti obično rastu i postaju veći nego što je prvočitno planirano. Ako je cela aplikacija u jednoj osnovi koda, to nam izaziva neke ozbiljne probleme u radu.

Na primer, ako treba da izvršimo promene čišćenja koje su velikog obima, kao što je menjanje bankovnog servisa ili sloja baze podataka, cela aplikacija postaje veoma nestabilna. Ove promene su veoma značajne u „životu“ projekta i zahtevaju mnogo dodatnih testiranja za raspoređivanje nove verzije.

Male promene mogu, takođe, generisati kolateralnu štetu, zato što različiti delovi sistema imaju različite zahteve za vreme ispravnog rada i za stabilnost. Postavljanje procesa za naplatu i rezervaciju u rizičnu situaciju, zato što je funkcija koja kreira PDF srušila server, malo je problematično.

Nekontrolisan rast je još jedan problem. Aplikacija dobija nove funkcije i ako programeri napuštaju projekat i priključuju se projektu, organizacija koda može da postane neuredna, a testovi sporiji. Ovaj rast se, obično, završava sa špageti osnovom koda, koja je veoma teška za održavanje, a neuredne baze podataka zahtevaju komplikovane planove migracije uvek kada neki od programera preradi model podataka.

Veliki projekti softvera obično zastarevaju za dve godine, a zatim polako počinju da se pretvaraju u nerazumljivu zbrku, koja je veoma teška za održavanje. A to se ne dešava zato što su programeri loši, već zato što, dok raste kompleksnost, sve manje ljudi razume implikacije veoma malih promena, pa pokušavaju da rade u izolaciji u jednom uglu osnove koda,

tako da, kada pogledate prikaz od 10.000 stopa projekta, vidite haos.

## PREDNOSTI I MANE MONOLITNE ARHITEKTURE

*Najveća mana monolitne arhitekture jeste loše skaliranje pri rastu aplikacije.*

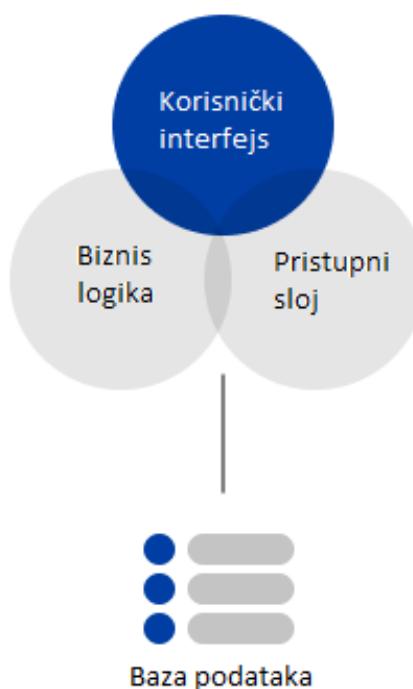
Sledeće tačke rezimiraju prednosti i mane monolitnog pristupa:

- Započinjanje projekta kao monolitnog je jednostavno i verovatno predstavlja najbolji pristup.
- Centralizovana baza podataka pojednostavljuje dizajn i organizaciju podataka.
- Raspoređivanje jedne aplikacije je jednostavno.
- Svaka promena u kodu može da utiče na nevezane funkcije. Kada se nešto ošteći, cela aplikacija može biti oštećena.
- Kako osnova koda raste, teže je da se održava čista i da bude pod kontrolom.

Očigledno rešenje je da razdvojimo aplikaciju u posebne delove, čak i ako će se rezultirajući kod i dalje pokretati u jednom procesu. Programeri to rešavaju izgradnjom svojih aplikacija, koristeći eksterne biblioteke i radne okvire.

Izgradnja veb aplikacije u Pythonu, ako koristimo radni okvir kao što je Flask, omogućava da se fokusiramo na poslovnu logiku i pojednostavljuje eksternalizaciju koda u Flask ekstenzije i male Python pakete. A razdvajanje koda u male pakete je često dobra ideja za kontrolisanje rasta aplikacije.

## Monolitna arhitektura

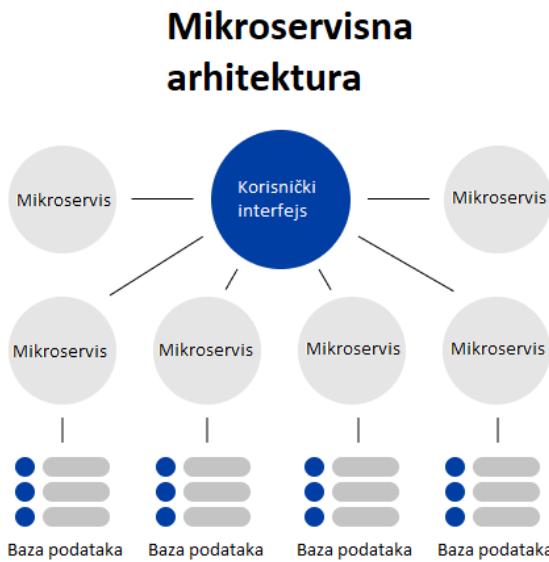


Slika 2.1 Monolitna arhitektura. [Izvor: Autor]

## MIKROSERVISNA ARHITEKTURA

*Umesto da imamo jednu aplikaciju koja je za sve odgovorna, razdvojićemo je na više različitih mikroservisa.*

Ako želimo da izgradimo istu aplikaciju upotrebom mikroservisa, organizovaćemo kod u nekoliko posebnih komponenata koje se pokreću u posebnim procesima. Umesto da imamo jednu aplikaciju koja je za sve odgovorna, razdvojićemo je na više različitih mikroservisa, kao što je prikazano na sledećem dijagramu.



Slika 2.2 Mikroservisna arhitektura. [Izvor: Autor]

#### Primer:

U primeru sa hotelom, prebacili smo deo složenosti na kraju smo dobili sledeće samostalne komponente:

1. Booking UI - servis čeonog prikaza koji generiše veb korisnički interfejs i vrši interakciju sa svim drugim mikroservisima,
2. PDF reporting service - veoma jednostavan servis koji kreira PDF-ove za potvrde ili bilo koji drugi dokument za određeni šablon ili neke podatke,
3. Search - servis koji može da bude ispitana za dobijanje liste hotela za određeni naziv grada. Ovaj servis ima sopstvenu bazu podataka.
4. Payments - servis koji vrši interakciju sa nezavisnim bankovnim servisom i upravlja bazom podataka naplate. Takođe šalje e-mail o uspešnom plaćanju.
5. Reservations - Skladišti rezervacije i generiše PDF-ove.
6. Users - Skladišti korisničke informacije i vrši interakciju sa korisnicima pomoću e-maila.
7. Authentication - servis koji vraća tokene provere identiteta koje svaki mikroservis može da upotrebi za proveru identiteta kada poziva druge.

## DEFINICIJA MIKROSERVISA

*Mikroservis je jednostavna aplikacija koja obezbeđuje suženu i jasno definisanu listu funkcija*

Ovi mikroservisi, zajedno sa nekoliko eksternih servisa, kao što je e-mail servis, obezbeđuju skup funkcija sličan monolitnoj aplikaciji. U ovom projektu svaka komponenta komunicira upotrebom **HTTP** protokola, a funkcije su dostupne pomoću **RESTful veb servisa**. Ne postoji

centralizovana baza podataka, jer svaki mikroservis interno koristi sopstvene strukture podataka, a podaci koji ulaze ili izlaze koriste jezički nezavisan format, kao što je [JSON](#). Mikroservis može da upotrebi i formate [XML](#) i [YAML](#), sve dok oni mogu da se proizvede i upotrebe u svakom jeziku i da „putuju“ kroz [HTTP](#) zahteve i odgovore.

Servis Booking UI je unekoliko poseban, zato što generiše korisnički interfejs (en. [User Interface](#), [UI](#)). U zavisnosti od radnog okvira čeonog prikaza koji je upotrebljen za izgradnju korisničkog interfejsa, izlaz servisa Booking UI može da bude mešavina [HTML](#)-a i [JSON](#)-a, ili, čak, čist [JSON](#) ako interfejs koristi statičnu alatku zasnovanu na JavaScript jeziku na strani klijenta za generisanje interfejsa direktno u pretraživaču.

Za razliku od ovog konkretnog slučaja korisničkog interfejsa, veb aplikacija koja je projektovana kao mikroservis je kompozicija od nekoliko mikroservisa koji mogu da vrše interakciju međusobno kroz [HTTP](#) da bi obezbedili ceo sistem.

U tom kontekstu mikroservisi su logičke jedinice koje se fokusiraju na veoma određene zadatke. Evo i cele definicije:

***Mikroservis je jednostavna aplikacija koja obezbeđuje suženu i jasno definisani listu funkcija. To je komponenta sa jednom odgovornošću, koja može da bude razvijena i raspoređena samostalno.***

Ova definicija ne spominje [HTTP](#) ili [JSON](#), zato što možete da razmotrite mali servis zasnovan na UDP-u koji, na primer, razmenjuje binarne podatke kao mikroservis.

Međutim, u našem slučaju u ovoj knjizi svi mikroservisi su samo jednostavne veb aplikacije koje koriste [HTTP](#) protokol i koriste i proizvode [JSON](#) kada to nije [UI](#).

## ✓ Poglavlje 3

# Prednosti mikroservisne arhitekture

## KARAKTERISTIKE MIKROSERVISA

*Mikroservisi su najčešće procesi koji komuniciraju putem mreže da izvrše zadatke koristeći protokole koji nisu vezani za određeni tip tehnologije.*

Karakteristika mikroservisa uključuju:

- Mikroservisi su najčešće procesi koji komuniciraju putem mreže da izvrše zadatke koristeći protokole koji nisu vezani za određeni tip tehnologije poput HTTP-a [1 - 3]. Ipak, servisi mogu da koriste i druge mehanizme inter-procesne komunikacije poput deljene memorije [4]. Servisi, takođe, mogu biti pokrenuti u okviru istog procesa, na primer, OSGi paketi.
- Servisi u mikroservis arhitekturi se mogu posebno isporučivati [5].
- Servisi su lako zamenjivi.
- Servisi su organizovani oko funkcionalnosti, npr. front-end korisnički interfejs, preporuke, logistika, naplata, itd.
- Servisi se mogu implementirati korišćenjem različitih programskih jezika, baza, hardverskih i softverskih okruženja, u zavisnosti šta od toga najviše odgovara za implementaciju.
- Servisi su mali veličinom, omogućena je razmena poruka, povezani su sadržajem, nezavisno razvijani, decentralizovani, kao i bildovani i izdavani pomoću automatskih procesa.

Osobine aplikacije čija je arhitektura zasnovana na mikroservisima:

- Prirodno se nameće modularna struktura.
- Je proces kontinualne isporuke softvera. Izmena malog dela aplikacije zahteva ponovno bildovanje i isporuku samo jednog ili malog broja servisa.
- Pridržava se principa poput sitno granuliranog interfejsa (za servise koji se mogu nezavisno izdavati), poslovno orijentisan razvoj, arhitekture IDEAL računarstva u oblaku, poliglot programiranje i postojanost, isporuka jednostavnih kontejnera, decentralizovana kontinualna isporuka, kao i DevOps sa holističkim nadzorom servisa [6].
- Obezbeđuje karakteristike koje doprinose skalabilnosti [7-9].

## PREDNOSTI MIKROSERVISNE ARHITEKTURE

*Mikroservisi pružaju razdvajanje dužnosti i bolje skaliranje u odnosu na monolitne arhitekture.*



Slika 3.1 Apstrakcija mikroservisne arhitekture. [Izvor: Wikipedia]

Iako arhitektura mikroservisa izgleda komplikovanije nego njen monolitni suparnik, prednosti su višestruke. Ona pruža sledeće:

- razdvajanje dužnosti,
- manje projekte za rad,
- više opcija za skaliranje i raspoređivanje.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## RAZDVAJANJE DUŽNOSTI

*Posebni timovi mogu da nezavisno razviju svaki mikroservis.*

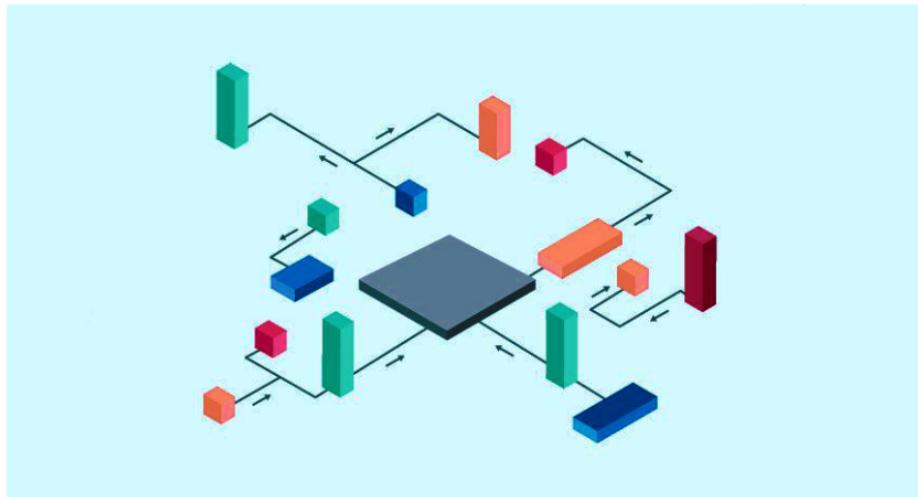
Pre svega, posebni timovi mogu da nezavisno razviju svaki mikroservis. Na primer, izgradnja servisa za rezervacije može da bude potpuno samostalan projekat. Tim koji je zadužen za ovaj servis može da ga kreira u bilo kom programskom jeziku, koristeći bilo koju bazu podataka, sve dok ima dobro dokumentovan HTTP API.

To takođe znači da je evolucija aplikacije više pod kontrolom, nego u slučaju monolitne aplikacije. Na primer, ako sistem za plaćanje promeni pozadinske akcije sa bankom, uticaj je lokalizovan unutar konkretnog servisa, a ostatak aplikacije ostaje stabilan i, verovatno, netaknut.

Ovo labavo povezivanje poboljšava brzinu projekta, jer se, na nivou servisa, primenjuje filozofija slična principu jedne odgovornosti.

Princip jedne odgovornosti je definisao Robert Martin da bi objasnio da treba da postoji samo jedan razlog da se klasa promeni; drugim rečima, svaka klasa treba da obezbedi jednu, dobro definisalu funkciju.

Primjeno na mikroservis, to znači da želimo da se uverimo da se svaki mikroservis fokusira na jednu ulogu.

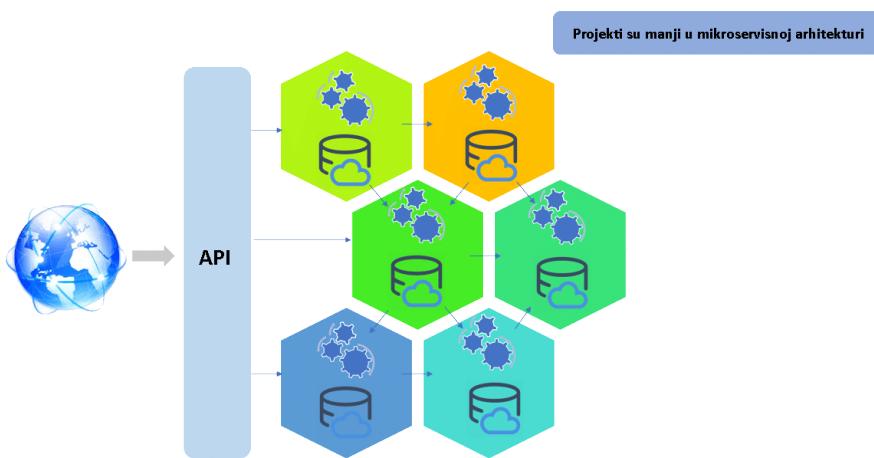


Slika 3.2 Apstrakcija razdvajanje dužnosti kod mikroservisa. [Izvor: wikipedia]

## MANJI PROJEKTI

*Velika prednost mikroservisne arhitekture jeste razdvajanje složenosti projekta.*

Druga prednost je razdvajanje složenosti projekta. Kada dodamo funkciju u aplikaciju, kao što je pisanje PDF izveštaja, čak i ako je kreirate jasno, osnovni kod će biti veći, komplikovaniji i ponekad sporiji. Izgradnja te funkcije u posebnoj aplikaciji sprečava pojavu ovog problema i olakšava pisanje bilo kojom alatkom.



Slika 3.3 Obim pojedinačnih projekata je manji. [Izvor: Autor]

Možemo da je prerađujemo često, da skratimo ciklus izdanja i da ostanemo u toku. Rast aplikacije ostaje pod kontrolom. Upotreba manjeg projekta takođe smanjuje rizik kada poboljšavamo aplikaciju: ako tim želi da isproba najnoviji programski jezik ili radni okvir, njegovi članovi mogu brzo da kreiraju

prototip koji implementira isti API mikroservisa, da ga isprobaju i da odluče da li će da ga zadrže ili ne.

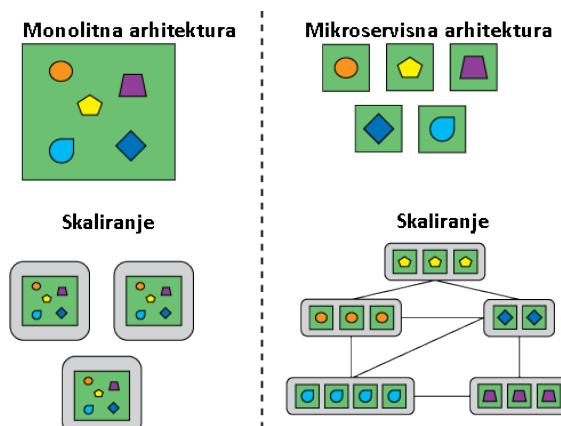
**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## SKALIRANJE I RAZVOJ

*Arhitektura mikroservisa je korisna u rešavanju mnoštva problema koji se mogu javiti kada aplikacija počne da raste.*

Na kraju, razdvajanje aplikacije na komponente olakšava skaliranje, u zavisnosti od ograničenja. Recimo da počinje da se svakodnevno javlja sve više korisnika koji rezervišu hotele, pa generisanje PDF-a počinje da „zagreva“ CPU-ove. Možete da rasporedite taj konkretni mikroservis na neke servere koji imaju veće CPU-ove.

Još jedan tipičan primer su mikroservisi koji koriste mnogo RAM-a – na primer, oni koji vrše interakciju sa bazama podataka u memoriji, kao što su *Redis* ili *Memcache*. Može da se podesi njihovo raspoređivanje na servere sa manje CPU-a i mnogo više RAM-a.



Slika 3.4 Skaliranje u monolitnoj i mikroservisnoj arhitekturi. [Izvor: Autor]

Prema tome, možemo da rezimiramo prednosti mikroservisa na sledeći način:

- Tim može da razvije svaki mikroservis nezavisno i da upotrebi bilo koji tehnološki stek. On može da definiše prilagođeni ciklus izdanja. Sve što treba da definiše je jezički nezavisan HTTP API.
- Programeri rastavljaju složenost aplikacije na logičke komponente. Svaki mikroservis se fokusira da izvršava dobro jedan zadatak.
- Pošto su mikroservisi samostalne aplikacije, postoji bolja kontrola za raspoređivanje koja olakšava skaliranje.

Arhitektura mikroservisa je korisna u rešavanju mnoštva problema koji se mogu javiti kada aplikacija počne da raste. Međutim, potrebno je da se čuvamo nekih novih problema koje oni uvode u aplikaciju.

## ▼ Poglavlje 4

# Nedostaci mikroservisne arhitekture

## MANE MIKROSERVISNE ARHITEKTURE

*Razvoj veb aplikacija pomoću mikroservisa ima mnogo prednosti, ali ima i mana.*

Kao što je ranije napomenuto, izgradnja aplikacije pomoću mikroservisa ima mnogo prednosti, ali nije savršena.

Potrebno je da obratimo pažnju na sledeće probleme sa kojima se možemo suočiti kada kodiramo mikroservise:

- neilogično razdvajanje,
- više mrežne interakcije,
- skladištenje i deljenje podataka,
- problemi kompatibilnosti,
- testiranje.

Ovi problemi će detaljno biti opisani u sledećim odeljcima.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## NELOGIČNO RAZDVAJANJE

*Razvoj veb aplikacije bazirana na mikroservisnoj arhitekturi može imati neilogično razdvajanje.*

Prvi problem arhitekture mikroservisa je kako su oni dizajnirani. Ne postoji način da tim može da kreira savršenu arhitekturu mikroservisa u prvom pokušaju.

Neki mikroservisi, kao što je PDF generator, predstavljaju očigledan primer. Međutim, čim se suočimo sa poslovnom logikom, postoji opasnost da će se kod pomeriti pre nego što jasno vidite kako

da ga razdvojite u odgovarajući skup mikroservisa.

Projekat treba da „sazri“ u ciklusima isprobavanja-i-neuspeha. A dodavanje i uklanjanje mikroservisa može da bude teže od prerade monolitne aplikacije.

Ovaj problem možemo da ublažimo izbegavanjem razdvajanja aplikacije u mikroservise ako ono nije evidentno.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

Ako postoji bilo kakva sumnja da razdvajanje ima smisla, zadržavanje koda u istoj aplikaciji je bezbedno. Uvek je lakše razdvojiti kod u novi mikroservis kasnije od spajanja nazad u dva mikroservisa u istoj osnovi koda kada se ispostavi da je odluka bila loša.

Na primer, ako uvek treba da rasporedimo dva mikroservisa zajedno ili ako jedna promena u mikroservisu utiče na model podataka drugog mikroservisa, možda nismo dobro razdvojili aplikaciju i možda bi ta dva servisa trebalo da ponovo budu spojena.

## VIŠE MREŽNIH INTERAKCIJA

*Broj mrežnih interakcija za izgradnju aplikacije jeste veći u odnosu na monolitnu arhitekturu.*

Drugi problem je količina mrežnih interakcija koje su dodate za izgradnju iste aplikacije. U monolitnoj verziji, čak i ako postane neuredan, sve se dešava u istom procesu i možemo da pošaljemo nazad rezultat, bez potrebe da pozivamo previše pozadinskih servisa za izgradnju aktuelnog odgovora.

To zahteva da se obrati pažnja na način kako je pozvan pozadinski servis i nameće pitanja:

- Šta se dešava kada Booking UI ne može da komunicira sa PDF servisom za izveštaje, zbog razdvojene mreže ili laganog servisa?
- Da li Booking UI poziva druge servise sinhrono ili asinhrono?
- Kako će to uticati na vreme odgovora?

Potrebno je da imamo jaku strategiju da bismo mogli da odgovorimo na sva ova pitanja.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## SKLADIŠTENJE I DELJENJE PODATAKA

*Efikasan mikroservis treba da bude nezavisан od drugih mikroservisa i ne bi trebalo da deli bazu podataka.*

Problematični su i skladištenje i deljenje podataka. Efikasan mikroservis treba da bude nezavisан od drugih mikroservisa i ne bi trebalo da deli bazu podataka.

Šta to znači za našu aplikaciju za rezervacije hotela?

To nameće pitanja, kao, na primer:

- Da li koristimo iste korisničke ID-ove u svim bazama podataka ili imamo nezavisne ID-ove u svakom servisu i čuvamo ih kao skrivene detalje implementacije?
- Kada je korisnik dodat u sistem, da li repliciramo neke informacije u drugim bazama podataka servisa pomoću strategija, kao što je „pumpanje“ podataka, ili je to preterivanje?
- Kako da rukujemo uklanjanjem podataka?

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## TESTIRANJE

*Aplikacija razvijena preko mikroservisa ima mnogo gradivnih blokova, i samim tim mnogo više mogućnosti da nešto pođe naopako.*

Na kraju, kada budemo želeli da izvršimo testove s kraja na kraj i da rasporedimo celu aplikaciju, suočićemo se sa mnogo gradivnih blokova. Potrebno je da imamo robustan i agilan proces raspoređivanja da bismo bili efikasni.

Potrebno je da se malo poigramo celom aplikacijom kada je razvijamo. Ne možemo u potpunosti da testiramo kod ako imamo samo jedan deo slagalice.

Srećom, postoje mnogi alati koje olakšavaju raspoređivanje aplikacija koje su građene pomoću nekoliko komponenata. A svi ti alati pomogli su u uspešnosti i prihvatanju mikroservisa, ali i obrnuto - mikroservisi su pomogli prihvatanje alata.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 5

# Flask mikroservis sa razvoj veb aplikacija

## UVOD U FLASK

*Flask predstavlja mikroservisno okruženje za razvoj veb aplikacija u Python programskom jeziku.*



Slika 5.1 Flask logo. [Izvor: <https://flask.palletsprojects.com/en/2.0.x>]

Python programski jezik poseduje mnogo paketa za razvoj veb aplikacija, koji se nazivaju i radnim okvirima (en. **frameworks**). Najpoznatiji su Django, Pyramid, Tornado, i Flask. U nastavku lekcije i u sledećim lekcijama obradiće se Flask paket.

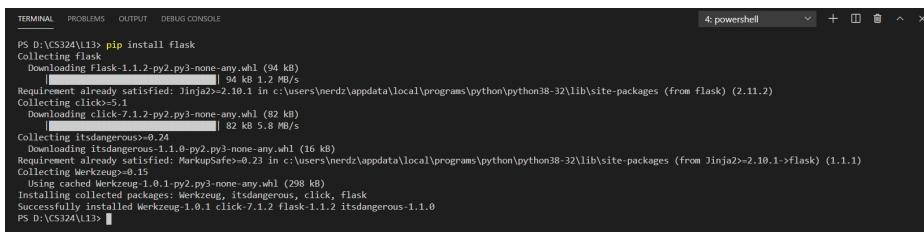
Flask predstavlja mikroservisno okruženje za razvoj veb aplikacija u Python programskom jeziku. Flask, budući da je mikroservis, ne zahteva dodatne biblioteke ili alate.

### Instalacija

Kao i ostali paket, Flask se instalira preko pip-a:

```
pip install flask
```

Kada se instalira paket, instaliraće se i Jinja2 paket koji dolazi uz Flask i predstavlja **endžin** (en. **engine**) za šablone veb aplikacija u Python-u. Jinja2 notacija koristiće se u sklopu Flask aplikacija.



```
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
PS D:\CS324\13> pip install flask
Collecting flask
  Downloading Flask-1.1.2-py2.py3-none-any.whl (94 kB)
    94 kB 1.2 MB/s
Requirement already satisfied: Jinja2>=2.10.1 in c:\users\nerdz\appdata\local\programs\python\python38-32\lib\site-packages (from flask) (2.11.2)
Collecting click>=7.0.0
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
    82 kB 5.8 MB/s
Collecting itsdangerous>=0.24
  Downloading itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\nerdz\appdata\local\programs\python\python38-32\lib\site-packages (from flask) (1.1.1)
Collecting Werkzeug>=0.15.2
  Using cached Werkzeug-1.0.1-py2.py3-none-any.whl (298 kB)
Installing collected packages: Werkzeug, itsdangerous, click, flask
Successfully installed Werkzeug-1.0.1 click-7.1.2 flask-2.1.2 itsdangerous-1.1.1
PS D:\CS324\13>
```

Slika 5.2 Instalacija Flask paketa. [Izvor: Autor]

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## MINIMALNA FLASK APLIKACIJA

*Potrebno je postaviti flask promenljivu, i pokrenuti flask iz terminala.*

Minimalna flask aplikacija se može naći na veb stranici Flask projekta, na **Quickstart** stranici.

Ovaj kod, kao i njegova proširenja, koristiće se za opisivanje paketa.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

### Objašnjenje koda

Najpre je izvršen uvoz klase **Flask** iz paketa **flask**. Onda je napravljena instanca klase sa imenom **name**. Ova promenljiva je specijalna Python promenljiva i odnosi se na ime skripte. Podrazumevano ime prilikom pokretanja aplikacije jeste **main**.

U četvrtom redu koda, napravljen je dekorator **.route()** sa parametrom **('')** koja govori o početnoj ruti veb aplikacije.

Definiše se funkcija **hello\_world()** koja će samo vratiti string **"Hello, World!"**

Kada se pokrene ovaj program, ne javljaju se greške, ali se ništa i ne dešava.

Potrebno je najpre podesiti Flask promenljivu u terminalu, u putanji gde se nalazi Python datoteka (ovde nazvana **hello.py**) ukucati:

```
# windows
set FLASK_APP=hello.py
# power shell
$env:FLASK_APP = "hello"

# mac/linux
export FLASK_APP=hello.py
```

Nakon toga, treba pokrenuti Flask komandom

```
flask run
```

Sada se pokrenula flask aplikacija, što se može videti na samom terminalu.

```
C:\WINDOWS\system32\cmd.exe - flask run
D:\CS324\113>set FLASK_APP=hello.py
D:\CS324\113>flask run
* Serving Flask app "hello.py"
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
* Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Slika 5.3 Pokretanje flask aplikacije. [Izvor: Autor]

## VIDEO OBJAŠNJENJE ZA MINIMALNU FLASK APLIKACIJU

*Sledi video objašnjenje minimalne Hello World flask aplikacije*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 6

### Pokazne vežbe #13

#### ZADATAK #1

*Pokazne vežbe rade se ovkorno 45 min*

##### **Zadatak #1 (45 min)**

Napraviti novo virtuelno okruženje u Python-u pod nazivom venv.

Instalirati paket flask.

Napraviti Flask aplikaciju sa rutama za stranicu "static", koja se nalazi u funkciji **prva\_strana** u kojoj treba ukucati sledeći HTML kod:

```
<!DOCTYPE html>
  <html lang="en">
    <head>

      <!-- Declared Vars To Go Here -->

      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1">

      <!-- Metadata -->
      <meta name="description" content="">
      <meta name="author" content="">

      <link rel="icon" href="mysource_files/favicon.ico">

      <!-- Page Name and Site Name -->
      <title>Page Name - Squiz Matrix HTML Example</title>

      <!-- CSS -->
      <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet">
      <link href="mysource_files/style.css" rel="stylesheet">

    </head>

    <body>

      <div class="container">
```

```
<header class="header clearfix" style="background-color: #ffffff">

    <!-- Main Menu -->
    <nav>
        <ul class="nav nav-pills pull-right">
            <li class="active"><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>

    <!-- Site Name -->
    <h1 class="h3 text-muted">Site Name</h1>

    <!-- Breadcrumbs -->
    <ol class="breadcrumb">
        <li><a href="#">Home</a></li>
        <li><a href="#">Level 1</a></li>
        <li class="active">Level 2</li>
    </ol>

</header>

<div class="page-heading">

    <!-- Page Heading -->
    <h1>Page Heading</h1>

</div>

<div class="row">

    <div class="col-sm-3">

        <!-- Sub Navigation -->
        <ul class="nav nav-pills nav-stacked">
            <li><a href="#">Level 2</a></li>
            <li class="active"><a href="#">Level 2</a>
                <ul>
                    <li><a href="#">Level 3</a></li>
                    <li><a href="#">Level 3</a></li>
                    <li><a href="#">Level 3</a></li>
                </ul>
            </li>
            <li><a href="#">Level 2</a></li>
        </ul>

    </div>

    <div class="col-sm-6">

        <div class="page-contents">
```

```
<!-- Design Body -->
<h2>Sub Heading</h2>
<p>Donec id elit non mi porta gravida at eget metus. Maecenas
faucibus mollis interdum.</p>
<h4>Sub Heading</h4>
<p>Morbi leo risus, porta ac consectetur ac, vestibulum at
eros. Cras mattis consectetur purus sit amet fermentum.</p>
<h4>Sub Heading</h4>
<p>Maecenas sed diam eget risus varius blandit sit amet non
magna.</p>

</div>

</div>

<div class="col-sm-3">

<!-- Login Section -->
<h2>Login</h2>

<!-- Search Section -->
<h2>Search</h2>

<!-- Nested Right Column Content -->

</div>

</div>

<footer class="footer">
<p class="pull-right">
<!-- Last Updated Design Area-->
Last Updated: Wednesday, January 6, 2016
</p>
<p>&copy; 2016 Company, Inc.</p>
</footer>

</div> <!-- /container -->

</body>
</html>
```

### Napomena:

HTML kod za stranicu static se nalazi u posebnoj datoteci. Treba uvesti funkciju **prva\_strana** u glavnu datoteku.

### Rešenje:

U terminalu treba uraditi sledeće:

```
C:\python -m venv venv
(venv) C:\pip install flask
```

Datoteka **PV01.py**:

```
from flask import Flask
from PV01_strana01 import prva_strana
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

@app.route('/static')
def page01():
    content = prva_strana()
    return content
```

Datoteka **PV01\_strana01.py**:

```
def prva_strana():
    content = """
        <!DOCTYPE html>
        <html lang="en">
        <head>

            <!-- Declared Vars To Go Here -->

            <meta charset="utf-8">
            <meta http-equiv="X-UA-Compatible" content="IE=edge">
            <meta name="viewport" content="width=device-width, initial-scale=1">

            <!-- Metadata -->
            <meta name="description" content="">
            <meta name="author" content="">

            <link rel="icon" href="mysource_files/favicon.ico">

            <!-- Page Name and Site Name -->
            <title>Page Name - Squiz Matrix HTML Example</title>

            <!-- CSS -->
            <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" rel="stylesheet">
            <link href="mysource_files/style.css" rel="stylesheet">

        </head>

        <body>

            <div class="container">

                <header class="header clearfix" style="background-color: #ffffff">

                    <!-- Main Menu -->
                    <nav>
```

```
<ul class="nav nav-pills pull-right">
    <li class="active"><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
</ul>
</nav>

<!-- Site Name -->
<h1 class="h3 text-muted">Site Name</h1>

<!-- Breadcrumbs -->
<ol class="breadcrumb">
    <li><a href="#">Home</a></li>
    <li><a href="#">Level 1</a></li>
    <li class="active">Level 2</li>
</ol>

</header>

<div class="page-heading">

    <!-- Page Heading -->
    <h1>Page Heading</h1>

</div>

<div class="row">

    <div class="col-sm-3">

        <!-- Sub Navigation -->
        <ul class="nav nav-pills nav-stacked">
            <li><a href="#">Level 2</a></li>
            <li class="active"><a href="#">Level 2</a>
                <ul>
                    <li><a href="#">Level 3</a></li>
                    <li><a href="#">Level 3</a></li>
                    <li><a href="#">Level 3</a></li>
                </ul>
            </li>
            <li><a href="#">Level 2</a></li>
        </ul>

    </div>

    <div class="col-sm-6">

        <div class="page-contents">

            <!-- Design Body -->
            <h2>Sub Heading</h2>
            <p>Donec id elit non mi porta gravida at eget metus. Maecenas
            faucibus mollis interdum.</p>
        </div>
    </div>
</div>
```

```
<h4>Sub Heading</h4>
<p>Morbi leo risus, porta ac consectetur ac, vestibulum at
eros. Cras mattis consectetur purus sit amet fermentum.</p>
<h4>Sub Heading</h4>
<p>Maecenas sed diam eget risus varius blandit sit amet non
magna.</p>

</div>

</div>

<div class="col-sm-3">

    <!-- Login Section -->
    <h2>Login</h2>

    <!-- Search Section -->
    <h2>Search</h2>

    <!-- Nested Right Column Content -->

</div>

</div>

<footer class="footer">
    <p class="pull-right">
        <!-- Last Updated Design Area-->
        Last Updated: Wednesday, January 6, 2016
    </p>
    <p>&copy; 2016 Company, Inc.</p>
</footer>

</div> <!-- /container -->

</body>
</html>
"""

return content
```

## VIDEO OBJAŠNJENJE ZA ZADATAK #1

*Sledi video objašnjenje za zadatak #1*

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 7

### Individualne vežbe #13

#### INDIVIDUALNE VEŽBE

*Individualne vežbe #13 rade se okvirno 45 minuta.*

Individualne vežbe #13 sastoje se iz dva do tri zadatka.

Vreme za izradu svih zadataka okvirno je 90 minuta.

Studenti u dogovoru sa predmetnim nastavnikom i asistentom dobijaju zadatke individualnih vežbi.

## ✓ Poglavlje 8

### Domaći zadatak

#### DOMAĆI ZADATAK #13

*Domaći zadatak #13 okvirno se radi 3h*

Domaći zadatak #13 daje se u dogovoru sa predmetnim nastavnikom i/ili asistentom.

**Predaja domaćeg zadatka:**

**Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije 7 dana nakon predavanja, za 100% poena. Nakon toga poeni se umanjuju za 50%.

**Internet studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganje ispita. Domaći zadaci se brane!

Domaći zadatak poslati dr Nemanji Zdravkoviću: nemanja.zdravkovic@metropolitan.ac.rs

Obavezno koristiti uputstvo za izradu domaćeg zadatka.

Uz .doc dokument (koji treba sadržati i screenshot svakog urađenog zadatka kao i komentare za zadatak), poslati i izvorne i dodatne datoteke.

## ▼ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Zaključak lekcije #13*

#### **Rezime:**

U ovoj lekciji najpre je objašnjeno kako napraviti virtuelno okruženje u Python-u i na taj način postaviti nezavistan Python interpreter u odnosu na sistemski.

Nakon toga, objašnjene su monolitne i mikroservisne arhitekture, sa naglaskom na prednosti mikroservisa.

Obrađen je Flask paket, i kako napraviti jednostavnu Flask aplikaciju sa statičkim HTML stranicama.

#### **Literatura:**

- Tarek Ziad, Python - Razboj mikroservisa, Kompjuter biblioteka, 2017. (prevod iste knjige Packt Publishing-a)
- David Beazley, Brian Jones, Python Cookbook: Recipes for Mastering Python 3, 3rd edition, O'Reilly Press, 2013.
- Mark Lutz, Learning Python, 5th Edition, O'Reilly Press, 2013.
- Andrew Bird, Lau Cher Han, et. al, The Python Workshop, Packt Publishing, 2019.
- Al Sweigart, Automate the boring stuff with Python, 2nd Edition, No Starch Press, 2020.

## DODATNE REFERENCE

### *Dodatne reference za mikroservise*

[1] Martin Fowler. „Microservices”

[2] Newman, Sam. Building Microservices, O'Reilly Media. ISBN 978-1491950357.

[3] Wolff, Eberhard. Microservices: Flexible Software Architectures

[4] „Micro-services for performance”. Vanilla Java

[5] Nadareishvili, I., Mitra, R., McLarty, M., Amundsen, M., Microservice Architecture: Aligning Principles, Practices, and Culture, O'Reilly 2016.

[6] „IFS: Microservices Resources and Positions”. hsr.ch

[7] Nicola Dragoni, Schahram Dustdar, Stephan T. Larsen, Manuel Mazzara. „Microservices: Migration of a Mission Critical System”

[8] Nicola Dragoni, Ivan Lanese, Stephan Thordal Larsen, Manuel Mazzara, Ruslan Mustafin, Larisa Safina. „Microservices: How To Make Your Application Scale” (PDF)

[9] Manuel Mazzara, Kevin Khanda, Ruslan Mustafin, Victor Rivera, Larisa Safina, Alberto Sillitti. „Microservices Science and Engineering”



## CS324 - SKRIPTING JEZICI

### Napredni rad u Flask paketu

Lekcija 14

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 14

### *NAPREDNI RAD U FLASK PAKETU*

- ✓ Napredni rad u Flask paketu
- ✓ Poglavlje 1: Funkcijsko zatvorene
- ✓ Poglavlje 2: Dekoratori u Python jeziku
- ✓ Poglavlje 3: Jinja notacija za Python jezik
- ✓ Poglavlje 4: Šabloni za Flask mikroservis
- ✓ Poglavlje 5: Nastavak rada u izradi Flask veb aplikacije
- ✓ Poglavlje 6: Pokazne vežbe
- ✓ Poglavlje 7: Individualne vežbe
- ✓ Poglavlje 8: Domaći zadatak
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ✓ Uvod

# UVOD

### *Uvod u lekciju #14*

U ovoj lekciji nastavlja se sa implementacijom Flask mikroservisa za razvoj veb aplikacija u Python programskom jeziku.

Da bi se razumeli napredni koncepti koji su korišćeni, najpre se uvodi pojam *funkcije prve klase* (en. *first class functions*), zatim *funkcijskog zatvorenja* (en. *function closures*), kao i upotreba *dekoratora* (en. *decorators*).

Da bi Python veb aplikacije bile dinamičke, treba znati koristiti Jinja notaciju za šablone koje koriste Flask aplikacije.

Nakon toga, nastavlja se rad u postojećoj Flask aplikaciji sa lekcije #13, prvenstveno kroz uvođenje šablonu.

Predstavljene su samo osnovne funkcionalnosti paketa Flask, a kompletno uputstvo ([playlist](#)) se može pogledati na:

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 1

# Funkcijsko zatvorenje

## FUNKCIJA PRVE KLASE

*Ako programski jezik podržava funkcije prve klase, onda se funkcija može tretirati kao promenljiva.*

Funkcija prve klase (en. **first class function**) jeste sposobnost programske jezike da tretira funkciju kao objekat prve klase (en. **first class citizen**). Objekat prve klase predstavlja entitet koji podržava sve operacije koje su dozvoljene ostalim entitetima, i uključuju prosleđivanje kao argument, povratna vrednost funkcije, kao i dodelu vrednosti promenljivoj.

To znači da jezik podržava slanje funkcija kao argumente drugim funkcijama, pri čemu ih vraća kao vrednost drugih funkcija i dodeljuje ih promenljivama ili ih čuva u strukturama podataka.

### Primer #1:

Napisati funkciju **square**, koja uzima parametar **x** i vraća kvadriranu vrednost ulaznog parametra. Zatim štampati funkciju sa nekim prirodnim brojem, kao i štampati funkciju bez promenljivih i bez zagrada (tj. bez poziva funkcije).

```
# funkcija prve klase

def square(x):
    return x * x
print( square(5) )
print( square )
```

### Izlaz:

```
>>> 25
>>> <function square at 0x01027808>
```

Može se videti da prilikom drugog poziva štampanja, kada se pozvala funkcija kao promenljiva, izlaz vraća da je objekat funkcija **square** na nekoj memorijskoj lokaciji.

### Primer #2

Koristeći funkciju **square** iz primera #1, napraviti dve promenljive, **f\_1** i **f\_2**. Prva promenljiva uzima vrednost funkcije za parametar 5, dok druga promenljiva uzima vrednost same funkcije.

```
# primer #2
f_1 = square(5)
f_2 = square

print(f_1)
print(f_2)
print(f_2(5))
```

Izlaz:

```
>>> 25
>>> <function square at 0x01027808>
>>> 25
```

Promenljiva **f\_1** je pozvala funkciju **square** za parametar 5, pa se njoj dodelila vrednost izlaza funkcije. Međutim, promenljivoj **f\_2** se dodelila sama funkcija, tako da je sada moguće pozvati funkciju **f\_2** koja je identična kao funkcija **square**.

## PROSLEĐIVANJE FUNKCIJE KAO PARAMETAR

*Pri definisanju funkcije, kao parametar moguće je proslediti i neku drugu funkciju.*

Ukoliko programski jezik podržava funkcije prve klase, onda se pri definisanju funkcije kao parametar proslediti i neka druga funkcija.

**Primer #3:**

Napisati funkciju **my\_map**, koja ima dva parametara: parametar **func** je funkcija, dok je drugi parametar lista **list\_arg**.

Unutar funkcije, za sve elemente parametra **list\_arg** pozivati funkciju **func**, i smestiti u listu **result**, koju treba vratiti kao povratnu vrednost.

Isprobati funkcionalnost za funkcije **square(x)**, koja vraća  $x^2$ , i funkciju **cube(x)**, koja vraća  $x^3$

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

```
# funkcija kao parametar druge funkcije

def my_map(func, arg_lst):
    result = []
    for item in arg_lst:
        result.append(func(item))
    return result

def square(x):
    return x*x
```

```
def cube(x):  
    return x*x*x  
  
lst = [1,2,3,4,5]  
  
print(my_map(square, lst))  
print(my_map(cube, lst))
```

Izlaz:

```
>>> [1, 4, 9, 16, 25]  
>>> [1, 8, 27, 64, 125]
```

## POJAM FUNKCIJSKOG ZATVORENJA

*Zatvorenje omogućava funkciji da pristupi uhvaćenim promenljivim kroz kopije vrednosti ili referenci koje se odnose na zatvorenje.*

Prema wikipediji, pojam zatvorenje, leksičko zatvorenje ili [funkcijsko zatvorenje](#) (en. [closure](#)), je tehnika za implementaciju leksički obuhvaćenog vezivanja imena u jeziku sa funkcijama prve klase.

Praktično, zatvorenje je zapis koji čuva funkciju zajedno sa okruženjem. Okruženje je preslikavanje koje povezuje svaku slobodnu promenljivu neke funkcije (promenljive koje se koriste kao lokalne, ali su definisane u obuhvatajućem opsegu), sa vrednošću ili referencom na koju je ime vezano kada je zatvorenje kreirano. Zatvorenje omogućava funkciji da pristupi uhvaćenim promenljivim kroz kopije vrednosti ili referenci koje se odnose na zatvorenje, čak i kada je funkcija pozvana izvan njihovih opsega.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Primer #4:**

Napisati funkciju **outer\_func** koja nema ulazne parametre. Unutar funkcije, napraviti lokalnu promenljivu **message** koja ima vrednost "Hello, World!". Takođe, unutar funkcije napraviti novu funkciju **inner\_func**, koja nema parametre. U telu unutrašnje funkcije štampati promenljivu **message**.

Kao povratnu vrednost spoljašnje funkcije vratiti a) poziv funkcije **inner\_func**, b) samo funkciju **inner\_func**. Diskutovati rezultate kada se pozove **outer\_func()**.

```
# zatvorenje funkcije  
def outer_func():  
    message = 'Hello, World!'  
  
    def inner_func():  
        print(message)  
  
    return inner_func()
```

```
# return inner_func  
  
outer_func()
```

## ▼ Poglavlje 2

# Dekoratori u Python jeziku

## POJAM DEKORATORA U PZTHON JEZIKU

*Dekoratorima je moguće proširiti funkcionalnost funkcije, bez modifikacije osnovne strukture funkcije.*

Dekorator (en. **decorator**) jeste šablon projektovanja u Python programskom jeziku koji omogućava korisniku da doda novu funkcionalnost postojećem objektu a da ne menja njegovu strukturu.

Dekoratori se obično javljaju pre definisanja funkcije za koju treba uraditi dekorator.

Dekoratori se najčešće koriste pri razvoju veb aplikacije korišćenjem Flask paketa ili Django radnog okvira.

Dekoratori se konstruišu slično kao funkcije zatvorena, s tim što se spoljašnjoj funkciji prosleđuje kao parametar funkcija.

### Primer #1:

Napraviti dekorator koji štampa "Dodatna funkcionalnost" pre izvršenja originalne funkcije.

```
# primer za dekoratore

def decorator_function(original_function):
    def wrapper_function():
        print('Dodatna funkcionalnost')
        return original_function()
    return wrapper_function

def display():
    print("Display funkcija pokrenuta")

decorated_display = decorator_function(display)
decorated_display()
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

# KONSTRUKCIJA DEKORATORA ZA BILO KOJI TIP FUNKCIJA

*Dekorator se poziva tako što se pre definicije originalne funkcije napiše ime spoljašnje funkcije sa znakom @.*

Pri konstrukciji dekoratora, treba obratiti pažnju na oblik originalne funkcije.

Neke funkcije mogu se pozivati bez argumenata, dok se druge moraju pozvati sa argumentima i/ili ključnim rečima.

Konvencija jeste pisati argumente i ključne reči kao **args** i **kwargs**.

Zbog toga, prilikom definisanja unutrašnje funkcije, potrebno je navesti argumente i ključne reči koje bi primila originalna funkcija:

```
def outer_func(orig_func):  
  
    def inner_func(*args, **kwargs):  
  
        # dodatna funkcionalnost  
        # ...  
        # ...  
  
        orig_func(*args, **kwargs)  
  
        # dodatna funkcionalnost  
        # ...  
        # ...  
  
    return inner_func
```

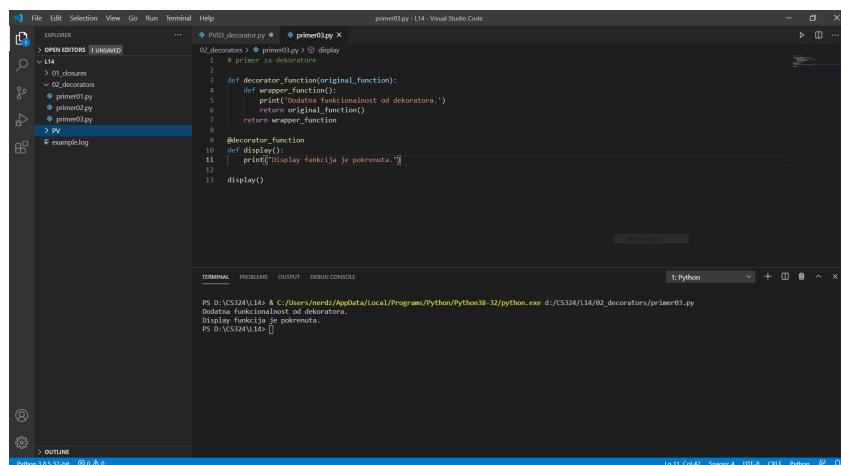
Na ovaj način moguće je pozvati bilo koju funkciju u dekorator.

## Pozivanje dekoratora

Dekorator se poziva tako što se pre definicije originalne funkcije napiše ime spoljašnje funkcije sa znakom @:

```
@outer_func  
def orig_func(*args, **kwargs):  
    # ...
```

Nakon toga, potrebno je samo pozvati originalnu funkciju.



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project structure with files: 01\_closures, 02\_decorators, prime01.py, prime02.py, prime03.py, and example01.log. The prime01.py file is open in the editor, displaying the following code:

```
02_decorators > prime01.py > display
1 # printer za dekoratore
2
3 def decorator_function(original_function):
4     def wrapper_function():
5         print("Dodatna funkcionalnost od dekoratora.")
6         return original_function()
7
8     @decorator_function
9     def display():
10        print("display funkcija je pokrenuta.")
11
12 display()
```

The terminal at the bottom shows the output of running the script:

```
PS D:\CS124\134 & C:/Users/merdi/AppData/Local/Programs/Python/Python38-32/python.exe d:/CS124/134/02_decorators/prime01.py
Dodatna funkcionalnost od dekoratora.
Display funkcija je pokrenuta.
PS D:\CS124\134 [1]
```

Slika 2.1 Pokretanje dekoratora. [Izvor: Autor]

## ▼ Poglavlje 3

# Jinja notacija za Python jezik

## INSTALACIJA JINJA2 PAKETA

*Flaks koristi Jinja2 jezik za šablone pri izrazi veb aplikacija.*



Slika 3.1 Jinja logo. [Izvor: <https://jinja.palletsprojects.com/en/2.11.x/>]

Da bi se razvile napredne veb aplikacije u Python jeziku korišćenjem Flask-a, potrebno je poznavati jezik za šablone [Jinja](#).

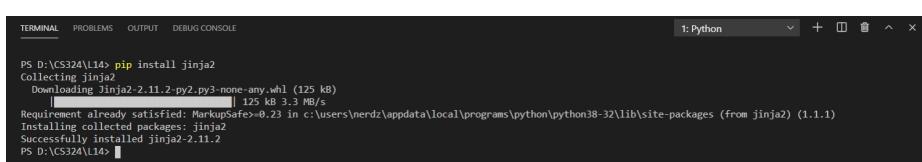
Jezik za šablone (en. [templating language](#)) sadrži promenljive i logiku koja, kada se izvršava (ili renderuje u HTML), zamenjuje sa pravim vrednostima.

Jinja šabloni su jednostavne .html datoteke. Po konvenciji, u Flask projektima nalaze se u direktorijumu /templates.

Instalacija [Jinja](#) paketa

Instalacija [Jinja2](#) paketa vrši se preko pip komande:

```
>>> pip install jinja2
```



```
PS D:\CS324\l14> pip install jinja2
Collecting jinja2
  Downloading Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
    125 kB 3.3 MB/s
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\nerdz\appdata\local\programs\python\python38-32\lib\site-packages (from jinja2) (1.1.1)
Installing collected packages: jinja2
Successfully installed jinja2-2.11.2
PS D:\CS324\l14>
```

Slika 3.2 Instalacija jinja2 paketa. [Izvor: Autor]

Prilikom instalacije Flask, [Jinja2](#) se automatski instalira, ali poželjno je proveriti korišćenjem [pip list](#) komande.

## JINJA 2 NOTACIJA

*Jinja notacija poseduje tagove koji kontrolišu logiku šablonu, i koji su inspirisani Django radnim okvirom za Python jezik*

Paket Flask koristi Jinja2 notaciju za šablove prilikom generisanja HTML stranica.

Jinja2 šablon predstavlja tekstualnu datoteku. Jinja može generisati bilo koji tekstualni format (HTML, XML, CSV, LaTeX, i dr.). Jinja šablon ne poseduje sopstvenu ekstenziju već koristi neke od postojećih.

Takođe, Jinja notacija poseduje tagove koji kontrolišu logiku šablonu, i koji su inspirisani Django radnim okvirom za Python jezik.

U nastavku je minimalni Jinja šalon koji ilustruje osnovnu Jinja konfiguraciju:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>My Webpage</title>
</head>
<body>
    <ul id="navigation">
        {% for item in navigation %}
            <li><a href="{{ item.href }}">{{ item.caption }}</a></li>
        {% endfor %}
    </ul>

    <h1>My Webpage</h1>
    {{ a_variable }}

    {% a comment %}
</body>
</html>
```

### Podrazumevani Jinja tagovi

**Naredbe** (en. **statements**) se pišu između sledećih tagova:

```
{% ... %}
```

**Izrazi** (en. **expressions**) se pišu između sledećih tagova:

```
{{ ... }}
```

**Komentari** (en. **comments**) se pišu između sledećih tagova:

```
{# ... #}
```

**Linijske naredbe** (en. **line statements**) se pišu između sledećih tagova:

```
# ... ##
```

## JINJA2 I PYTHON

*Kada se izvršava (render-uje) Ninja šablon, potrebno je pozvati metodu `.render()`, sa argumentom koji daje vrednost nekom od korišćenih tagova.*

Kada se Jinja2 uvozi u Python, treba uraditi sledeće

```
from jinja2 import Template
```

Ne uvozi se ceo jinja paket, već samo modul `Template` koji je potreban.

Najpre se pravi objekat klase `Template` koji kao argument može imati string sa nekim od Ninja tagova.

Kada se izvršava (render-uje) Ninja šablon, potrebno je pozvati metodu `.render()`, sa argumentom koji daje vrednost nekom od korišćenih tagova.

### Primer:

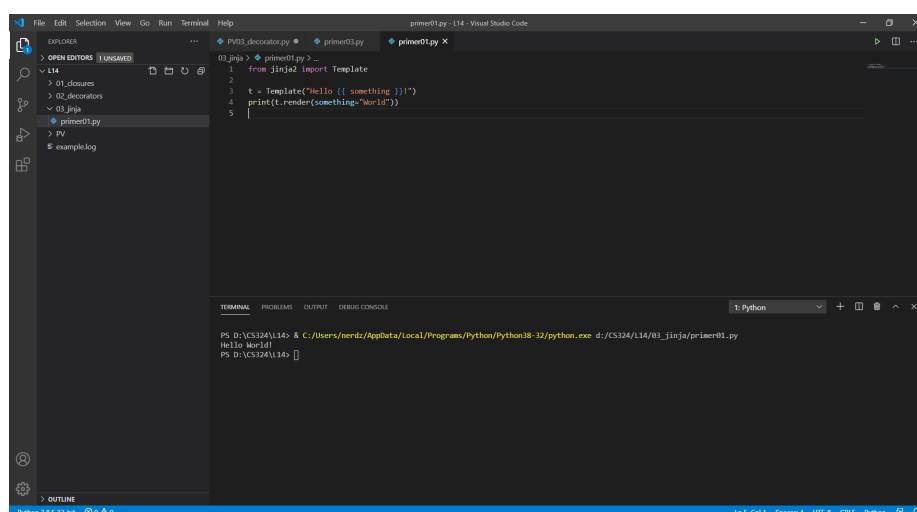
Napisati "Hello world!" aplikaciju koristeći Ninja šablone u Python jeziku, gde je `world` promenljiva `something` koju treba render-ovati.

### Rešenje:

```
from jinja2 import Template

t = Template("Hello {{ something }}!")

print(t.render(something="World"))
```



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files: `03_jinja`, `03_decorators`, `03_jinja`, and `primer01.py`.
- Code Editor:** The `primer01.py` file contains the following code:

```
from jinja2 import Template

t = Template("Hello {{ something }}!")

print(t.render(something="World"))
```
- Terminal:** Shows the command line output:

```
PS D:\CS32A\14> & C:/Users/verdi/AppData/Local/Programs/Python/Python38-32/python.exe d:/CS32A/14/03_jinja/primer01.py
Hello World!
PS D:\CS32A\14>
```

Slika 3.3 Ninja2 Hello world aplikacija. [Izvor: Autor]

## ▼ Poglavlje 4

# Šabloni za Flask mikroservis

## POČETNA FLASK APLIKACIJA (BEZ ŠABLONA)

*Flask podrazumevano pravi serverski proces na localhost-u (IP adresi 127.0.0.1) na portu 5000.*

U nastavku obradiće se korišćenje Jinja šablona unutar Flask veb aplikacije.

Početna `flask` aplikacija jeste sledeća:

```
# $env:FLASK_APP = "flask_primer"
# $env:FLASK_DEBUG = 1
# flask run

from flask import Flask
app = Flask(__name__)

@app.route('/')
@app.route('/home')
def hello_world():
    return "<h2> Pocetna stranica! </h2>"

@app.route('/about')
def about():
    return "About stranica!"
```

Komentarisane linije koda jesu komande za `flask` unutar `powershell` terminala za Windows. Ukoliko se koristi `Linux` ili `macOS`, treba koristiti komandu `set`

Kao i ranije, najpre se importuje `flask`, i pravi objekat klase `Flask`.

Definišu se rute kroz dekoratore, sa nekim početnim vrednostima za početnu (`home`) i `about` stranice.

Flask podrazumevano pravi serverski proces na `localhost`-u (IP adresi 127.0.0.1) na portu 5000.



Slika 4.1 Početna home stranica. [Izvor: Autor]

Slika 4.2 Početna about stranica. [Izvor: Autor]

## POČETNA FLASK APLIKACIJA (SA HTML ŠABLONIMA)

*Efikasniji način upisa HTML-a od direktnog jeste pravljenje šabloni.*

Unutar tela funkcije **home()** moguće je napisati i ceo HTML koji se želi prikazati, ali to nije praktično.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
@app.route('/home')
def hello_world():
    return
    ...
    <!DOCTYPE html>
    <html>
        <body>
            <h1>My First Heading</h1>
            <p>My first paragraph.</p>
        </body>
    </html>
    ...

@app.route('/about')
def about():
    return "About stranica!"
```

Efikasniji način upisa HTML-a jeste **pravljenje šabloni**.

Za početak, treba napraviti direktorijum za šablonе (**templates**) u radnom direktorijumu:

```
└── templates
    └── __pycache__
```

Unutar direktorijuma za šablove, napraviti šablon za svaku od stranica. Šablon predstavlja HTML dokument.

Za home stranicu, datoteka home.html izgledaće:

```
<!DOCTYPE html>
<html>
<head>
    <title>CS324 - Skripting jezici</title>
</head>
<body>

    <h1>Pocetna stranica</h1>

</body>
</html>
```

## PRIKAZ FLASK APLIKACIJE SA HTML ŠABLONIMA

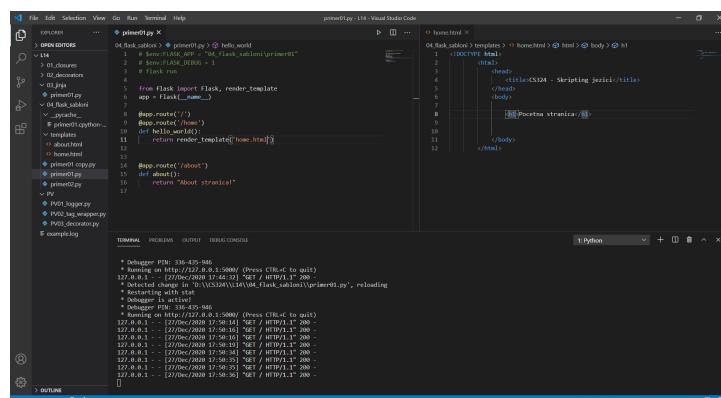
*Modul `render_template` potražiće dokumente u "templates" direktorijumu flask projekta.*

Nakon kreacije HTML šablonu u odgovarajućem direktorijumu, potrebno je i uvesti u modul `render_template` iz `flask` paketa.

Nakon toga, umesto vraćanja direktno kodiranog HMTL-a, funkcija treba da vrati poziv funkcije `render_template`, koji kao string ima HTML dokument. Podrazumevano, ovaj dokument tražiće se u direktorijumu `templates`.

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/')
@app.route('/home')
def hello_world():
    return render_template('home.html')
```



Slika 4.3 Izgled HTML šablonu u Flask aplikaciji. [Izvor: Autor]

Slika 4.4 Izgled početne stranice sa render-ovanim šablonom. [Izvor: Autor]

Slika 4.5 Izgled izvornog koda početne stranice. [Izvor: Autor]

## ▼ Poglavlje 5

# Nastavak rada u izradi Flask veb aplikacije

## DINAMIČKE VEB STRANICE I FLASK

*Tokom rada savremene veb aplikacije, podaci se stalno ažuriraju.*

Pri radu sa savremenim veb aplikacijama statičke stranice nisu dovoljne.

Tokom rada aplikacije, podaci se stalno ažuriraju.

U primeru, veb aplikacija koja je u izradi predstavlja blog. Recimo da je pozivom iz baze podataka dobijena lista imenika koji sadrži sledeće informacije

```
posts_var = [
    {
        'author': 'Autor 1',
        'title': 'Blog post 1',
        'content': 'Sadrzaj prvog posta',
        'date_posted': datetime.now()

    },
    {
        'author': 'Autor 2',
        'title': 'Blog post 2',
        'content': 'Sadrzaj drugog posta',
        'date_posted': datetime.now()

    }
]
```

Najpre neka ovaj sadržaj bude direktno kodiran u .py datoteci.

```
# $env:FLASK_APP = "04_flask_sabloni\primer01"
# $env:FLASK_DEBUG = 1
# flask run

from flask import Flask, render_template
app = Flask(__name__)
from datetime import datetime

posts = [
    {
```

```
'author': 'Autor 1',
'title': 'Blog post 1',
'content': 'Sadrzaj prvog posta',
'date_posted': datetime.now()

},
{
    'author': 'Autor 2',
    'title': 'Blog post 2',
    'content': 'Sadrzaj drugog posta',
    'date_posted': datetime.now()

}
]

@app.route('/')
@app.route('/home')
def hello_world():
    return render_template('home.html', posts=posts_var)

@app.route('/about')
def about():
    return render_template('about.html')
```

U funkciji koja vraća glavnu stranicu, u **render\_template** funkciji dodat je argument **posts=posts\_var**, koji označava da u šablonu, kada nađe na **posts**, treba da zameni sa promenljivom **posts\_var**.

## JINJA2 ŠABLONI U FLASK APLIKACIJI

*Često se koriste ista imena za promenljive koje se nalaze u šablonu i koje se prosleđuju šablonu.*

Unutar šablonu za home stranicu (`templates\home.html`) treba ubaciti sadržaj promenljive **posts\_var**.

Po pozivu funkcije **render\_template**, sadržaj datoteke **posts\_var** nalazi se u posts datoteci (**posts=posts\_var**).

Često se koriste ista imena za promenljive koje se nalaze u šablonu i koje se prosleđuju šablonu. Međutim, ovde je namerno odvojeno da bi se razumelo koja se promenljiva gde navodi.

Izgled HTML datoteke sada je:

```
<!DOCTYPE html>
<html>
    <head>
        <title>CS324 - Skripting jezici</title>
```

```
</head>
<body>

  {% for post in posts %}
    <h1>{{ post.title }}</h1>
    <p> By {{ post.author }} on {{ post.date_posted }}</p>
    <p>{{ post.content }}</p>
  {% endfor%}

</body>
</html>
```

U okviru HTML datoteke sada se koristi **Jinja notacija** za šablove, i to:

**Naredba:**

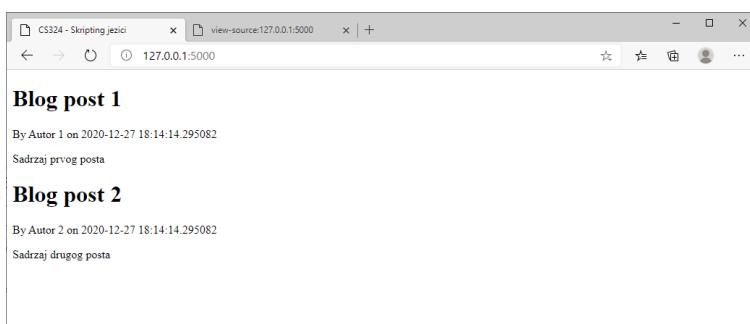
```
{% for post in posts %}
...
{% endfor %}
```

prolazi kroz sve elemente promenljive posts. U Python datoteci to je promenljiva **posts\_var**.

Zatim, **izrazi**:

```
<h1>{{ post.title }}</h1>
<p> By {{ post.author }} on {{ post.date_posted }}</p>
<p>{{ post.content }}</p>
```

Ovi izrazi nalaze se unutar **for petlje** i pišu se samo u dvostrukim velikim zagradama. Kada se sačuva HTML dokument, stranica izgleda:



Slika 5.1 Izgled stranice nakon ubacanja Jinja notacije. [Izvor: Autor]

## USLOVNO GRANANJE U JINJA NOTACIJI

*Moguće je dodati uslovno grananje u Jinja notaciji za šablove.*

Unutar šablonu za about stranicu, dodaće se još jedna promenljiva, **title**:

```
@app.route('/about')
def about():
    return render_template('about.html', title='About')
```

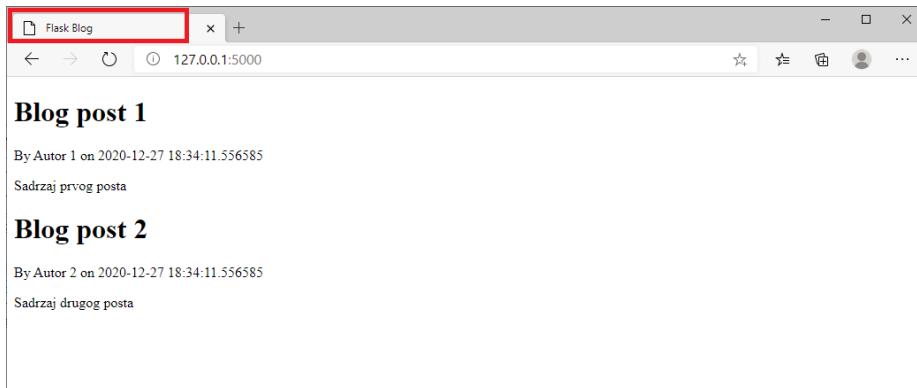
U HTML šablonu obe stranice, može se postaviti uslov da ukoliko postoji promenljiva title, da sadržaj promenljive title bude naslov stranice, a ukoliko nije ispunjen uslov, da bude neko podrazumevani naslov.

```
<!DOCTYPE html>
<html>
    <head>
        {% if title %}
            <title>Flask Blog - {{ title }}</title>
        {% else %}
            <title>Flask Blog</title>
        {% endif %}
    </head>
    <body>

        <h1>About stranica</h1>

    </body>
</html>
```

Na ovaj način, about stranica imaće naslov "Flask blog - About", dok će početna stranica imati podrazumevani naslov.



Slika 5.2 Početna stranica sa podrazumevanim naslovom. [Izvor: Autor]

Slika 5.3 About stranica sa novim naslovom. [Izvor: Autor]

## NASLEĐIVANJE ŠABLONA

*Moguće je naslediti opšti šablon, dok pojedinčani šabloni menjaju samo određene blokove*

Primećuje se da je šablon koji se koristi za home i about stranicu jako sličan, samo se razlikuje deo sa tagom **<body>**. Zbog toga, moguće je napraviti opšti šablon, kojeg će ostali šabloni naslediti, dok će određene blokove zadržati.

Najpre, unutar templates direktorijuma treba napraviti novu datoteku **layout.html**. Ovo će biti datoteka opšteg šablonu.

```
<!DOCTYPE html>
  <html>
    <head>
      {%- if title %}
        <title>Flask Blog - {{ title }}</title>
      {%- else %}
        <title>Flask Blog</title>
      {%- endif %}
    </head>
    <body>
      {%- block content %} {%- endblock %}
    </body>
  </html>
```

Svi identični delovi jesu isti, samo je dodato da se u telu HTML strane nalazi blok **content**:

```
{%- block content %} {%- endblock %}
```

Sada, u pojedinačnim šablonima treba dodati sledeće:

### About:

```
{% extends "layout.html" %}

{%- block content %}
  <h1>About stranica</h1>
{%- endblock %}
```

### Home:

```
{% extends "layout.html" %}

{%- block content %}

  {%- for post in posts %}

    <h1>{{ post.title }}</h1>
    <p> By {{ post.author }} on {{ post.date_posted }}</p>
    <p>{{ post.content }}</p>
  {%- endfor %}

{%- endblock %}
```

Na ovaj način funkcionalnost se ne gubi, a sve što je bilo zajedničko za šablove nasledilo se od glavnog šablonu **layout.html**, komadom:

```
{% extends "layout.html" %}
```

## ✓ Poglavlje 6

### Pokazne vežbe

#### ZADATAK #1

*Zadatak #1 odnosi se na funkcije prve klase i okvirno se radi 15 minuta.*

##### **Zadatak #1 (15 minuta)**

Napisati program koji pravi aktivnosti pozvanih funkcija. Definisati funkciju logger, koja uzima kao parametar funkciju func.

Unutar ove funkcije, definisati novu funkciju log\_func, sa proizvoljnim brojem argumenata, koja vraća text koji kaže koja se funkcija pokreće i sa kojim parametrima, i štampa pokrenutu funkciju func. Zati, vraća log\_func kao povratnu vrednost.

Definisati jednostavne funkcije za sabiranje i oduzimanje dva broja, napraviti odgovarajuće logger funkcije, i isprobati funkcionalnost.

Koristiti sledeći paket za logovanje:

```
# uvoz paketa
import logging
logging.basicConfig(filename='example.log', level=logging.INFO)

# pravljenje log datoteke:
logging.info(
    'Running "{}" with arguments {}'.format(func.__name__, args))
```

```
# Pokazna vezba Zadatak #1

import logging
logging.basicConfig(filename='example.log', level=logging.INFO)

def logger(func):
    def log_func(*args):
        logging.info(
            'Running "{}" with arguments {}'.format(func.__name__, args))
        print(func(*args))
    return log_func

def add(x, y):
```

```
return x+y

def sub(x, y):
    return x-y

add_logger = logger(add)
sub_logger = logger(sub)

add_logger(3, 3)
add_logger(4, 5)

sub_logger(10, 5)
sub_logger(20, 10)
```

## ZADATAK #2

*Zadatak #2 odnosi se na zatvorene funkcije i okvirno se radi 15 minuta.*

### Zadatak #2 (15 minuta)

Napisati funkciju **html\_tag(tag)**, koja ima ulazni parametar string **tag**.

Unutar ove funkcije, definisati funkciju **wrap\_text(msg)**, koja štampa string **msg** okružen otvorenim i zatvorenim **html** tagovima.

Funkcija **html\_tag** vraća (bez pozivanja) funkciju **wrap\_text**.

U glavnom programu, napraviti funkcije za štampanje **h1** i **p** tagova, i ispitati funkcionalnost.

```
def html_tag(tag):

    def wrap_text(msg):
        print('<{0}>{1}</{0}>'.format(tag, msg))

    return wrap_text

print_h1 = html_tag('h1')
print_h1('Test Headline!')
print_h1('Another Headline!')

print_p = html_tag('p')
print_p('Test Paragraph!')
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ZADATAK #3

*Zadatak #3 odnosi se na dekoratore i okvirno se radi 15 minuta.*

### Zadatak #3 (15 minuta)

Napisati funkciju koja računa i štampa faktorijal unetog broja. Može se koristiti math modul.

Zatim, napraviti dekorator **vreme\_izvrsenja** koji računa vreme izvršenja unete funkcije. Koristiti time modul za dobijanje razlike u vremenu.

```
def vreme_izvrsenja(orig_f):
    import time

    def wrapeer_func(*args, **kwargs):
        start = time.time()
        orig_f(*args, **kwargs)
        stop = time.time()

        print(f'Vreme izvrsenja funkcije {orig_f.__name__} je {stop - start}')

    return wrapeer_func

@vreme_izvrsenja
def factorial(num):
    import math, time
    print(math.factorial(num))

factorial(52)
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ✓ Poglavlje 7

### Individualne vežbe

#### INDIVIDUALNE VEŽBE #14

*Individualne vežbe #14 rade se okvirno 90 minuta.*

Individualne vežbe #14 sastoje se iz dva do tri zadatka.

Vreme za izradu svih zadataka okvirno je 90 minuta.

Studenti u dogovoru sa predmetnim nastavnikom i asistentom dobijaju zadatke individualnih vežbi.

## ✓ Poglavlje 8

### Domaći zadatak

#### DOMAĆI ZADATAK #14

*Domaći zadatak #14 okvirno se radi 2h*

Domaći zadatak #14 daje se u dogovoru sa predmetnim nastavnikom i/ili asistentom.

**Predaja domaćeg zadatka:**

**Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije 7 dana nakon predavanja, za 100% poena. Nakon toga poeni se umanjuju za 50%.

**Internet studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganje ispita. Domaći zadaci se brane!

Domaći zadatak poslati dr Nemanji Zdravkoviću: nemanja.zdravkovic@metropolitan.ac.rs

Obavezno koristiti uputstvo za izradu domaćeg zadatka.

Uz .doc dokument (koji treba sadržati i screenshot svakog urađenog zadatka kao i komentare za zadatak), poslati i izvorne i dodatne datoteke.

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Zaključak lekcije #14*

#### **Rezime:**

U ovoj lekciji najpre su objašnjeni koncepti funkcija prve klase, zatvorenja funkcija, kao i dekoratera kod Python jezika.

Nakon toga, objašnjena je jinja2 notacija za šablone koja se koristi pri izradi Flask veb aplikacija.

Konačno, nastavljen je rad sa Flask mikroservisom, ovog puta dodavanjem šablona i prikazom promenljivih iz Python dela u HTML deo aplikacije.

#### **Literatura:**

- Tarek Ziade;, Python - Razboj mikroservisa, Kompjuter biblioteka, 2017. (prevod iste knjige Packt Publishing-a)
- David Beazley, Brian Jones, Python Cookbook: Recipes for Mastering Python 3, 3rd edition, O'Reilly Press, 2013.
- Mark Lutz, Learning Python, 5th Edition, O'Reilly Press, 2013.
- Andrew Bird, Lau Cher Han, et. al, The Python Workshop, Packt Publishing, 2019.
- Al Sweigart, Automate the boring stuff with Python, 2nd Edition, No Starch Press, 2020.



## CS324 - SKRIPTING JEZICI

Pregled predmeta

Lekcija 15

PRIRUČNIK ZA STUDENTE

# CS324 - SKRIPTING JEZICI

## Lekcija 15

### ***PREGLED PREDMETA***

- ✓ Pregled predmeta
- ✓ Poglavlje 1: Python i razumevanje listi
- ✓ Poglavlje 2: Dobre prakse Python programiranja
- ✓ Poglavlje 3: Česti propusti pri programiranju u Python jeziku
- ✓ Poglavlje 4: Ostali popularni Python paketi
- ✓ Poglavlje 5: Pregled predmeta
- ✓ Poglavlje 6: Pokazne vežba #15
- ✓ Poglavlje 7: Individualne vežbe
- ✓ Poglavlje 8: Domaći zadatak
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

## ✓ Uvod

# UVOD

### *Uvod u lekciju #15*

U petnaestoj lekciji najpre je predstavljen pojam **razumevanja listi** kao napredni način kreacije listi i imenika.

Zatim su predstavljene dobre prakse u Python programiranju, kao i neke česte greške koje se javljaju, najviše jer Python ne predstavlja prvi naučeni programski jezik.

Tokom predmeta predstavljeni su najpopularniji paketi u Python jeziku za razne primene. Naravno, pregršt biblioteka, paketa i modula jeste ono što čini Python popularnim. Upravo zbog velikog broja paketa neki od njih nisu obrađeni, ali su nabrojeni, zajedno sa linkovima i dokumentacijom.

Konačno, dat je pregled predmeta sa Google formom koju je poželjno popuniti zbog unapređenja predmeta.

U pokaznim vežbama dati su opisi pitanja kakva studenti mogu očekivati na ispitu.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

# ▼ Poglavlje 1

## Python i razumevanje listi

### UVOD U RAZUMEVANJE LISTI

*Razumevanje listi jeste jedan sintaksni oblik pisanja listi i sličnih iterabilnih promenljivih.*

Razumevanje listi (en. **list comprehension**) jeste metoda pisanja listi i sličnih iterabilnih promenljiva koja je bliža matematičkoj notaciji konstrukciji skupova.

#### Primer:

Matematički formulisati niz koji je skup svih brojeva **2 puta  $X$** , gde je  $X$  element skupa **prirodnih brojeva**, i  $X^2$  je veće od 3.

$$S = \{2 \cdot x \mid x \in \mathbb{N}, x^2 > 3\}$$

U pojedinim programskim jezicima, razumevanje listi jeste sintaksni oblik pisanja listi i sličnih iterabilnih promenljivih koje pojednostavljaju pisanje koda, često izbegavajući više linija koda i for petlje.

U Python jeziku, ovakva notacija iterabilnih promenljiva olakšava pisanje listi koja je ujedno bliža i ljudskom govoru, što čini Python sintaksu još lakšom.

#### Primer #1

Lista **nums** sadrži sve prirodne brojeve od 1 do 10. Popuniti promenljivu **my\_list** ovim brojevima, bez i sa korišćenjem razumevanja listi.

```
nums = [1,2,3,4,5,6,7,8,9,10]

# zadatak:
# Popuniti my_list sa n, za svako n u nums

# bez razumevanja listi
my_list = []
for num in nums:
    my_list.append(num)
print(my_list)

# sa razumevanjem listi
my_list = [n for n in nums]
print(my_list)
```

#### Izlaz:

```
>>>[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>>[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Može se primetiti da je kod razumevanja listi **for** petlja sadržana unutar jedne komande, kao i da nema inicijalne kreacije prazne liste.

Takođe se može primetiti da je sama sintaksa sličnija govornom jeziku zadakta: "Popuniti **my\_list** sa **n**, za svako **n** u **nums**"

## PRIMERI ZA RAZUMEVANJE LISTI BEZ DODATNIH USLOVA

*Nekada je moguće koristiti map funkciju umesto for funkcije pri sintaksi razumevanja listi.*

### Primer #2

Lista **nums** sadrži sve prirodne brojeve od 1 do 10. Popuniti promenljivu **my\_list** kvadratima ovih brojeva, najpre bez, pa sa korišćenjem razumevanja listi, kao i korišćenjem **map** funkcije.

```
nums = [1,2,3,4,5,6,7,8,9,10]

# zadatak:
# Popuniti my_list sa n^2, za svako n u nums

# bez razumevanja listi
my_list = []
for num in nums:
    my_list.append(num*num)
print(my_list)

# sa razumevanjem listi
my_list = [n*n for n in nums]
print(my_list)

# sa map+lambda funkcijom
my_list = map(lambda n: n*n, nums)
print(list(my_list))
```

### Izlaz:

```
>>>[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>>[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>>[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

**Napomena:**

**Funkcija `map()` pravi iterator koji računa funkciju koristeći argumente svih iterabilnih promenljivih. Zaustavlja se kada se iskoristi najkraći iterator.**

```
numbers1 = [1, 2, 3]
numbers2 = [4, 5, 6]

result = map(lambda x, y: x + y, numbers1, numbers2)
print(list(result))
```

Izlaz:

```
>>>[5, 7, 9]
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## PRIMERI ZA RAZUMEVANJE LISTI SA DODATNIM USLOVIMA

*Nekada je moguće koristiti filter funkciju umesto for funkcije pri sintaksi razumevanja listi.*

### Primer #3

Lista `nums` sadrži sve prirodne brojeve od 1 do 10. Popuniti promenljivu `my_list` samo parnim brojevima iz prve liste, najpre bez, pa sa korišćenjem razumevanja listi, kao i korišćenjem filter funkcije.

```
nums = [1,2,3,4,5,6,7,8,9,10]

# zadatak:
# Popuniti my_list sa n, za svako n u nums koje je parno

# bez razumevanja listi
my_list = []
for num in nums:
    if num % 2 == 0:
        my_list.append(num)
print(my_list)

# sa razumevanjem listi
my_list = [n for n in nums if n % 2 == 0]
print(my_list)

# sa filter+lambda funkcijom
my_list = filter(lambda n: n % 2 == 0, nums)
print(list(my_list))
```

**Izlaz:**

```
>>>[2, 4, 6, 8, 10]
>>>[2, 4, 6, 8, 10]
>>>[2, 4, 6, 8, 10]
```

**Napomena:**

**Funkcija `filter()` vraća iterabilnu promenljivu sa onim elementima koji ispunjavaju uslov.**

Return an iterator yielding those items of iterable for which function(item) is true. If function is None, return the items that are true.

```
# function that filters vowels
def fun(variable):
    letters = ['a', 'e', 'i', 'o', 'u']
    if (variable in letters):
        return True
    else:
        return False

# sequence
sequence = ['g', 'e', 'e', 'j', 'k', 's', 'p', 'r']

# using filter function
filtered = filter(fun, sequence)

print('The filtered letters are:')
for s in filtered:
    print(s)
```

**Izlaz:**

```
>>>e
>>>e
```

## PRIMERI ZA RAZUMEVANJE LISTI SA UGNJEŽDENIM PETLJAMA

*I ugnježdene petlje se mogu napisati u jednoj liniji koda korišćenjem sintakse razumevanja listi.*

### Primer #4

Vratiti par (slovo, broj) za svako slovo u "abcd" i za svaki broj u "1234", bez, i sa razumevanjem listi.

```
# zadatak:  
# Popuniti my_list sa parovima (letter, num) za slova "abcd" i brojeve "1234"  
  
# bez razumevanja listi  
my_list = []  
for letter in 'abcd':  
    for num in range(4):  
        my_list.append((letter, num))  
print(my_list)  
  
# sa razumevanjem listi  
my_list = [(letter, num) for letter in 'abcd' for num in range(4)]  
print(my_list)
```

Izlaz:

```
>>>[('a', 0), ('a', 1), ('a', 2), ('a', 3), ('b', 0), ('b', 1), ('b', 2), ('b', 3),  
('c', 0), ('c', 1), ('c', 2), ('c', 3), ('d', 0), ('d', 1), ('d', 2), ('d', 3)]  
>>>[('a', 0), ('a', 1), ('a', 2), ('a', 3), ('b', 0), ('b', 1), ('b', 2), ('b', 3),  
('c', 0), ('c', 1), ('c', 2), ('c', 3), ('d', 0), ('d', 1), ('d', 2), ('d', 3)]
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## RAZUMEVANJE IMENIKA

*Razumevanje imenika prati sličnu sintaksu kao kod razumevanja lista*

**Primer 5:**

Date su dve liste:

```
names = ['Bruce', 'Clark', 'Peter', 'Logan', 'Wade']  
heroes = ['Batman', 'Superman', 'Spiderman', 'Wolverine', 'Deadpool']
```

Napraviti imenik **my\_dict** sa parovima **ključ: vrednost** = **name: hero** iz datih listi, najpre bez, a posle sa razumevanjem liste (razumevanjem imenika).

```
# Razumevanje listi - imenici  
  
names = ['Bruce', 'Clark', 'Peter', 'Logan', 'Wade']  
heroes = ['Batman', 'Superman', 'Spiderman', 'Wolverine', 'Deadpool']  
  
# Bez razumevanja imenika  
my_dict = {}  
for name, hero in zip(names, heroes):  
    my_dict[name] = hero
```

```
print(my_dict)

# Sa razumevanjem imenika
my_dict = {name: hero for name, hero in zip(names, heroes)}
print(my_dict)
```

**Izlaz:**

```
>>>{'Bruce': 'Batman', 'Clark': 'Superman', 'Peter': 'Spiderman', 'Logan': 'Wolverine', 'Wade': 'Deadpool'}
>>>{'Bruce': 'Batman', 'Clark': 'Superman', 'Peter': 'Spiderman', 'Logan': 'Wolverine', 'Wade': 'Deadpool'}
```

Napomena:

Funkcija `zip()` vraća tuple-ove elemenata pojedinačnih iterabilnih promenljivih koje se ubace kao lista parametra. Funkcija staje kod najkraće iterabilne promenljive

```
# primer za zip:
list(zip('abcdefg', range(3), range(4)))

# zip objekat se mora pretvoriti u listu da bi se stampali tuple-ovi
>>>[('a', 0, 0), ('b', 1, 1), ('c', 2, 2)]
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ▼ Poglavlje 2

# Dobre prakse Python programiranja

## UVOD U DOBRU PRAKSU PROGRAMIRANJA

*Uvek treba imati na umu da postoji dobra praksa programiranja koja može učiniti kod pisan na Python programskom jeziku boljim i optimalnijim.*

Jedna od najvećih prednosti Python programskog jezika jeste jednostavna sintaksa i čitljivost koda.

Međutim, treba imati na umu da postoji [dobra praksa programiranja](#) (en. [good coding practices](#)) koja može učiniti kod pisan na Python programskom jeziku boljim i optimalnijim.

U nastavku jesu neke od teme dobrih praksi pri radu sa Python jezikom:

- Ternarna uslovna izjava (en. [ternary conditionals](#))
- Dodavanje donjih crta (en. [underscore placeholders](#))
- Kontekstni menadžeri (en. [context managers](#))
- Funkcija [enumerate\(\)](#)
- Funkcija [zip\(\)](#)
- Funkcije [help\(\)](#) i [dir\(\)](#)

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## TERNARNI USLOV

*Kod ternarnog uslova (ternarnog operatora) moguće je izvršiti ispitivanje uslova u jednom redu.*

Kod jednostavnih if/else naredbi, moguće je istu naredbu napisati kroz [ternarni uslov](#), koji je jednostavnije sintakse.

Primer:

Napisati jednostavni program za proveru uslova [condition](#). Ukoliko je [condition=True](#), vrednost promenljive [x](#) postaje 1, ili postaje 0 ukoliko nije.

Uraditi program i kao [ternarni uslov](#).

```
# ternarni uslov
condition = True

# prvi nacin
if condition:
    x = 1
else:
    x = 0

print(x)

# drugi nacin
x = 1 if condition else 0

print(x)
```

Primećuje se da je, slično kao kod razumevanje listi, ternarni uslov bliži govornom jeziku.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## DODAVANJE DONJIH CRTA

*Python može ispisivati brojeve sa zapetom za razdvajanje hiljada.*

Pri radu sa veoma velikim ili veoma malim brojevima, standardni izlaz ne grupiše izlaz po hiljadama.

**Primer:**

Broj milion (1000000) se može lakše prepoznati ukoliko se napiše (1.000.000), tj. ako se hiljade razdvajaju tačkom.

**Napomena:**

***U srpskom jeziku, razlomljeni brojevi se razdvajaju decimalnom zapetom, dok u engleskom se razdvajaju decimalnom tačkom.***

Izlaz u Python-u (ukoliko se ne koristi prikaz brojeva po naučnoj notaciji:  $10^6$  ) prikazivaće brojeve bez grupisanja:

```
>>> 1000000
```

Međutim, u Python jeziku moguće je pri pisanju vrednosti dodati donju crtu (en. **underscore**) koja neće menjati vrednost promenljive, ali je lakše programerima pisati velike ili male brojeve.

Primer: Napisati 10,000,000,000 korišćenjem donjih crta.

```
x = 10_000_000_000
print(x)
>>> 100000000000
```

Python takođe može ispisivati brojeve sa zapetom za razdvajanje hiljada ukoliko se štampa kao *f-string* sa dodatnim parametrom `{:,}`

```
# razdvajanje hiljada

x = 10_000_000_000
y = 100_000_000_000
z = x + y
print(f'{z:,}')

# izlaz
>>> 110,000,000,000
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## KONTEKSTNI MENADŽERI

*Upotrebom kontekstnih menadžera lakše se upravljaju resursi koji bi se posle korišćenja morali zatvoriti.*

Pri radu sa kontrolom resursa, kao što su datoteke ili niti (en. **thread**), treba voditi računa o resursima da se resursi zatvore nakon korišćenja.

Ukoliko se koriste kontekstni menadžeri, onda se unutar bloka kod kontekstnog menadžera izvršavaju operacije nad resursima, a čim se izade iz tog bloka koda, resurs se zatvara.

**Primer:**

Otvoriti datoteku "test\_file.txt" najpre bez, a nakon toga sa kontekstnim menadžerom. U ova slučaja pročitati liniju teksta i štampati

```
# bez kontekstnog menadzera

f = open('test_file.txt', 'r')
x = f.readline()
f.close()
print(x)

# sa kontekstnim menadzerom

with open('test_file.txt', 'r') as f:
    y = f.readline()

print(y)
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## FUNKCIJA ENUMERATE

*Upotrebom funkcije enumerate moguće je dobiti i indeks i promenljivu unutar petlje.*

Za razliku od programskih jezika koji su bazirani na C familiji jezika, za prolazak kroz for petlju nije neophodan brojač kao eksplisitna promenljiva, koji ujedno sliži i kao indeks elemenata unutar iterabilne promenljive.

Korišćenjem funkcije **enumerate()** može se odmah dobiti indeks neke iterabilne promenljive prilikom prolaska kroz petlju.

**Primer:**

Napisati program koji će prolaziti kroz petlju za sve elemente liste **predmeti**. Unutar petlje ispisivati indeks i vrednost elementa.

Najpre uraditi zadatku bez **enumerate** funkcije, a potom sa **enumerate** funkcijom.

```
predmeti = ['CS324', 'CS220', 'CS225', 'SE325', 'IT331']

# bez enumerate
index = 0
for item in predmeti:
    print(f'{index} {item}')
    index += 1

# sa enumerate

for index, item in enumerate(predmeti):
    print(f'{index} {item}')
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## ZIP FUNKCIJA

*Zip objekat nije moguće direktno odštampati, već je potrebno pretvoriti u listu.*

Kada se potrebno da grupišemo elemente više listi, tako da se  $n$ -ti element jedne liste mapira sa  $n$ -tim elementom druge liste, može se koristiti funkcija **zip()**.

Funkcija zip napraviće zip objekat sa **tuple**-ovima koji sadrže elemente listi koje su parametri funkcije. Mapiranje se vrši dok se ne istroše elementi najkraće liste.

Zip objekat nije moguće direktno odštampati, već je potrebno pretvoriti u listu.

**Primer:**

Napraviti dve liste sa istim brojem elemenata. Štampati pojedinačne elemente kroz jednu *for petlju* korišćenjem **zip** funkcije nad listama.

```
# zip funkcija

predmeti = ['CS324', 'CS220', 'CS225', 'SE325', 'IT331']
ime_predmeta = ['Skripting jezici', 'Arhitektura racunara', 'Operativni sistemi',
'Upravljanje projektima razvoja softvera', 'Racunarke mreze i komunikacije']

# bez enumerate
for predmet, ime in zip(predmeti, ime_predmeta):
    print(f'{predmet} - {ime}')

print(zip(predmeti, ime_predmeta))
print(list(zip(predmeti, ime_predmeta)))
```

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## FUNKCIJE HELP() I DIR()

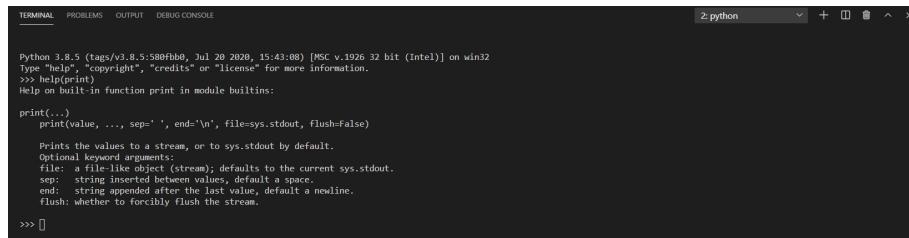
*Funkcija help() omogućavaju programeru da nauče kako se radi sa nekim objektom, dok funkcija dir() ispisuje koje su funkcije dostupne u okviru modula ili paketa.*

### Funkcija help()

U Python konzoli moguće je potražiti pomoć oko specifičnog objekta.

Potrebno je u Python konzoli (ne u standardnom terminalu) ukucati:

```
>>> help(object)
```



```
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 2:python + x

Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> help(print)
Help on built-in function print in module builtins:

print(*, sep=' ', end='\n', file=sys.stdout, flush=False)
    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep:  string inserted between values, default a space.
    end:  string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.

>>> [
```

Slika 2.1 Funkcija help(). [Izvor: Autor]

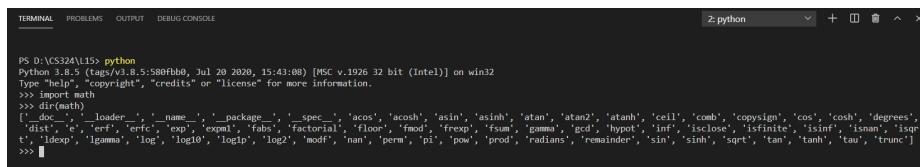
Kao izlaz, vratiće se objašnjenje objekta (ukoliko postoji)

### Funkcija dir()

U Python konzoli moguće je naći koje su sve funkcije dostupne u okviru nekog paketa ili modula.

Potrebno je najpre učitati (vesti) željeni paket i/ili modul, i onda pozvati funkciju dir():

```
>>> import module
>>> dir(module)
```



The screenshot shows a terminal window with the title '2: python'. The command 'dir(math)' is entered, and the output lists numerous mathematical functions and constants. The list includes: \_\_doc\_\_, \_\_loader\_\_, \_\_name\_\_, \_\_package\_\_, \_\_spec\_\_, 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'cos', 'cosh', 'copysign', 'degrees', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqr', 'ldexp', 'lgamma', 'log', 'log10', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc'.

Slika 2.2 Funkcije dir(). [Izvor: Autor]

Izlaz ove funkcije ispisaće kao listu sve funkcije koje je moguće pozvati kada se modul ili paket učita.

## ▼ Poglavlje 3

# Česti propusti pri programiranju u Python jeziku

## UVOD U ČESTE PROPUSTE

*Čest greške pri programiranju u Python jeziku jesu posledica prelaska sa drugog programskog jezika u Python, ili prelazak sa Python 2.x u Python 3.x.*

Pri prelasku sa drugog programskog jezika na Python, ili pri prelasku sa Python 2.x na Python 3.x jezik, treba obratiti pažnju na greške koje se često dešavaju, a možda nisu odmah uočljive.

U nastavu su dati neki od čestih propusta pri programiranju u Python 3.x programskom jeziku:

- Konflikti u imenovanju
- Promenljivi podrazumevani argumenti
- Uvoz kompletног модула ili paketa u lokalni imenski prostor

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

## KONFLIKTI U IMENOVANJU DATOTEKA

*Nekad python interpreter javlja grešku ukoliko je ime datoteke isto kao i ime modula standardne biblioteke.*

Python standardna biblioteka sadrži veliki broj modula koji je moguće uvesti u radnu datoteku.

Jedan od čestih problema koji se javlja prilikom rada sa dodatnim modulima jeste čuvanje same datoteke kao ime modula koji se uvozi.

Primer:

U radnu datoteku uvesti modul math, i funkcije sin() i radians(). Napisati program koji računa sinus od datog ugla (unetog u stepenima), i sačuvati datoteku kao math.py. Diskutovati rezultate.

**Napomena:**

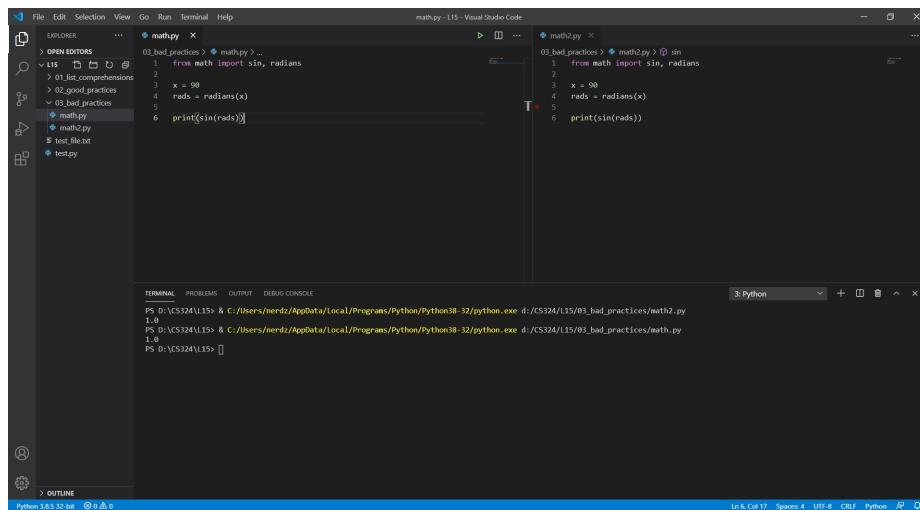
**Python neće uvek javiti grešku o konfliktnim imenima. Zavisiće i od IDE-a i od verzije interpretera.**

```
# datoteka se cuva kao math.py

from math import sin, radians

x = 90
rads = radians(x)

print(sin(rads))
```



Slika 3.1 Konflikti u imenovanju datoteka. [Izvor: Autor]

## PROMENLJIVI PODRAZUMEVANI ARGUMENTI

*Podrazumevani argumenti funkcije se izvršavaju samo jednom, i to u trenutku kreacije same funkcije.*

Treba voditi računa o funkcijama koje imaju prazne iterabilne objekte kao podrazumevani argument. Ovakvi objekti jesu promenljivi ili mutabilni (en. **mutable**)

### Primer:

Napisati funkciju **stampaj\_predmet(predmet, spisak=[])** koja dodaje **predmet** na **spisak** i štampa **spisak**.

Napraviti listu sa nekoliko predmeta i pozvati funkciju sa novim predmetom.

Zatim, pozvati funkciju više puta bez drugog argumenta. Diskutovati rezultate.

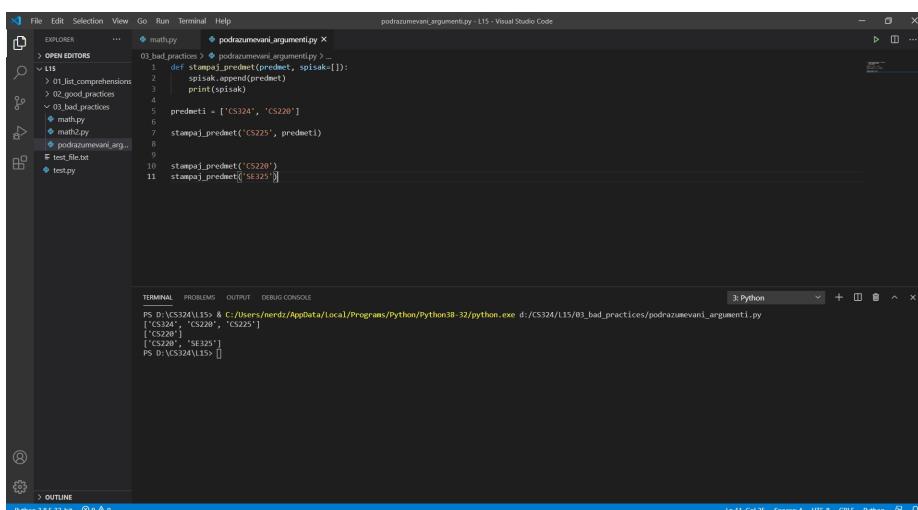
```
def stampaj_predmet(predmet, spisak=[]):
    spisak.append(predmet)
    print(spisak)
```

```
predmeti = ['CS324', 'CS220']

stampaj_predmet('CS225', predmeti)

stampaj_predmet('CS220')
stampaj_predmet('SE325')
```

U Python programskom jeziku, podrazumevani argumenti (argumenti funkcije koji imaju podrazumevanu vrednost) se izvršavaju **jednom**, u trenutku kreacije same funkcije.



Slika 3.2 Mutabilni podrazumevani argumenti. [Izvor: Autor]

## UVOZ CELOG MODULA U LOKALNI IMENSKI PROSTOR

*Uvoz celog modula ili paketa u lokalni imenski prostor treba koristiti samo kada je neophodno.*

Prilikom uvoza paketa i modula u radnu datoteku, treba biti obazriv ukoliko se ceo paket ili modul uveze u imenski prostor.

Komanda za uvoz celog paketa ili modula je:

```
from module import *
```

Ukoliko se uvozi veći broj paketa ili modula, nije uvek jednostavno naći koja funkcija ili klasa je iz kog paketa ili modula.

Takođe, pojedini moduli imaju iste funkcije, i može se desiti greška da se želi koristiti funkcija iz jednog paketa, a zapravo se koristi funkcija drugog.

**Primer:**

Uvesti sve iz paket numpy, pa onda iz modula math.

Napraviti numpy niz **a**, sa elementima koji su od 1 do 10 sa korakom 1. Napraviti novi niz **b**, čiji su elementi **sqrt()** od niza **a**. Stampati niz **b**. Diskutovati rezultate.

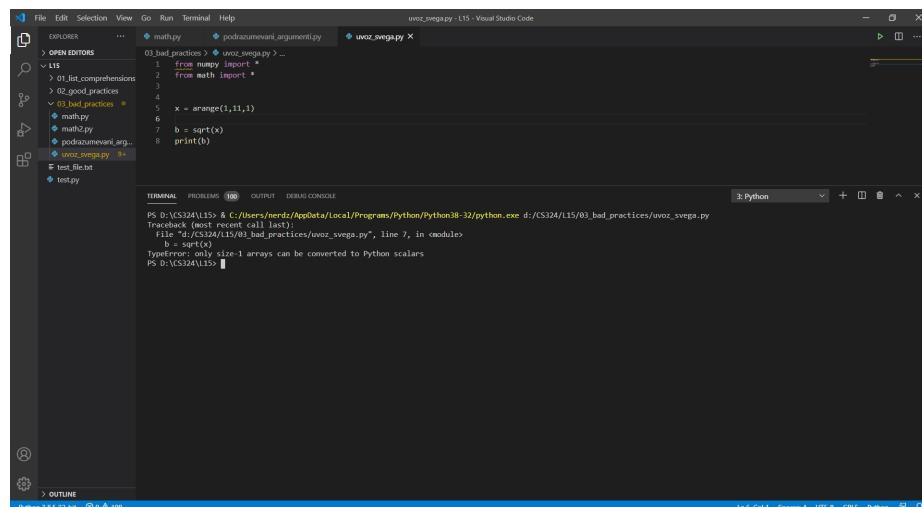
### ***Napomena:***

**Neki IDE javiće da se ne koriste sve funkcije koje su uvezene iz datog modula ili paketa.**

```
from numpy import *
from math import *

x = arange(1,11,1)

b = sqrt(x)
print(b)
```



Slika 3.3 Greška pri uvozu svih funkcija u imenski prostor. [Izvor: Autor]

## ▼ Poglavlje 4

### Ostali popularni Python paketi

#### MODULARNOST = POPULARNOST

*Osobina modularnosti ono što čini Python jezik toliko popularnim.*



Slika 4.1 Modularnost Python jezika. [Izvor: python.org]

Python poseduje pregršt modula i paketa za različitu primenu. Upravo je osobina modularnosti ono što čini Python jezik toliko popularnim.

U nastavku biće reči o popularnim paketima koji nisu obrađivani u predmetu.

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

**Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.**

#### PAKETI ZA MAŠINSKO UČENJE

*Najpopularniji paketi za mašinsko učenje i neuronske mreže jesu tensorflow, keras i pytorch*



Slika 4.2 TensorFlow logo. [Izvor: <https://www.tensorflow.org>]

### TensorFlow

TensorFlow je biblioteka koju je razvio Google Brain Team, i deo je svake aplikacije mašinskog učenja koju je Google razvio.

TensorFlow radi na principu velikog broja tenzorskih operacija, prvenstveno zbog neuronskih mreža koje se mogu lako predstaviti kao niz tenzorskih grafova.

<https://www.tensorflow.org>



Slika 4.3 Keras. [Izvor: <https://keras.io>]

### Keras

Keras pruža mehanizam za predstavljanje modela neuronskih mreža, kao i alate za predstavljanje modela, procesiranja skupova podataka, vizualizaciju i prikaz podataka.

Keras se koristi u tandemu sa TensorFlow paketom, koji je zadužen za logiku mašinskog učenja.

<https://keras.io>



Slika 4.4 PyTorch. [Izvor: <https://pytorch.org>]

### PyTorch

PyTorch prestavlja najveću biblioteku za mašinsko učenje za Python koje dozvoljava korišćenje grafičkog procesora (GPU) za ubrazano dobijanje rezultata. PyTorch podržava kreaciju dinamičkih grafova i automatko računanje gradijenata u algoritmima mašinskog učenja.

<https://pytorch.org>

## PAKET ZA VEŠTAČKU INTELIGENCIJU

*Popularni paketi za veštačku inteligenciju i duboko učenje jesu pybrain, caffe2 i hebel.*



Slika 4.5 PyBrain. [Izvor: <http://pybrain.org>]

### PyBrain

PyBrain sadrži algoritme za neuronske mreže koje se lako mogu koristiti i istrenirati bez velikog predznanja o načinu funkcionisanja neuronskih mreža. Cilj PyBrain paketa jestu napraviti jednostavne ali moćne algoritme za mešinsko učenje sa već generisanim okruženjem za testiranje i poređenje.

<http://pybrain.org>



Slika 4.6 Caffe2. [Izvor: <https://caffe2.ai>]

### Caffe2

Caffe2 predstavlja radni okvir za duboko učenje koji je skalabilan i modularan. Pruža jednostavan način za eksperimentisanje sa algoritmima dubokog učenja i poseduje API-je za C++ i Python jezike.

<https://caffe2.ai>



Slika 4.7 Hebel. [Izvor: <https://pypi.org/project/Hebel/>]

## Hebel

Hebel predstavlja Pyzhon biblioteku za duboko učenje koje koristi GPU ubrzanje sa CUDA platformom i pyCUDA paketom. Hebel implementira neuroneke mreže za algoritme klasifikacije i regresije na jedan ili na više zadataka.

<https://pypi.org/project/Hebel/>

# MATEMATIČKI PAKETI

*Najpopularniji matematički paketi koji nisu obrađivani jesu statsmodels i seaborn, kao i theano.*



Slika 4.8 Theano.[Izvor: <https://pypi.org/project/Theano/>]

## Theano

Theano predstavlja biblioteku koja omogućava efikasno definisanje, optimizaciju i evaluaciju matematičkih izraza koji sadrže  $n$ -dimenzionalne nizove.

Theano prepoznae nestabilne izraze i izvršava ih stabilnim algoritmima, što drugi paketi, kao što je NumPy, ne nude.

<https://pypi.org/project/Theano/>



Slika 4.9 Statsmodels. [Izvor: <https://www.statsmodels.org>]

## Statsmodels

Statsmodels predstavlja modul koji pruža klase i funkcije za estimaciju različitih statističkih modela, kao i za statističke testove.

<https://www.statsmodels.org>



Slika 4.10 Seaborn. [Izvor: <https://seaborn.pydata.org>]

### Seaborn

Seaborn predstavlja paket za vizuelizaciju statističkih veličina, koji pruža više mogućnosti od paketa od kojeg je izveden - matplotlib. Seaborn je blisko integriran sa Pandas paketom i pandas DataFrame strukturama.

<https://seaborn.pydata.org>

## WEB PAKETI

*Najpopularniji web radni okviri jesu django, tornado i pyramid.*



Slika 4.11 Django. [Izvor: <https://www.djangoproject.com>]

### Django

Django predstavlja najpopularniji radni okvir za razvoj veb aplikacija u Python jeziku. Django poseduje sve potrebne elemente za razvoj kompletne veb aplikacije, uključujući autentifikaciju, rute, šablone, migraciju baza kao i mapiranje objekata.

<https://www.djangoproject.com>



Slika 4.12 Tornado. [Izvor: <https://www.tornadoweb.org/en/latest/>]

### Tornado

Tornado predstavlja radni okvir za razvoj veb aplikacija koji je asinhronog tipa, i predstavlja mikroservis koji omogućava veliki broj istovremenih konekcija. Tornado se zasniva na mogućnosti obrade velikog saobraćaja.

<https://www.tornadoweb.org/en/latest/>



Slika 4.13 [Pyramid. Izvor:<https://trypyramid.com>]

## Pyramid

Pyramid predstavlja drugi po popularnosti radni okvir za razvoj veb aplikacija. Cilj pyramid-a jeste da se učini što više, sa što manje kompleksnosti. Pyramid zahteva Python 3 i ne može raditi na ranijim verzijama.

<https://trypyramid.com>

## GUI PAKETI

*Među popularnijim GUI paketima jesu tkinter i pyqt.*



Slika 4.14 Tkinter. [Izvor: <https://docs.python.org/3/library/tkinter.html>]

### Tkinter

Tkinter predstavlja podrazumevani radni okvir za razvoj aplikacija sa GUI-jem. Tkinter je popularan zvog svoje jednostavnosti i korisničkog interfejsa.

<https://docs.python.org/3/library/tkinter.html>



Slika 4.15 PyQt. [Izvor: <https://riverbankcomputing.com/software/pyqt/intro>]

### PyQT

PyQT predstavlja QT biblioteku za Python, i služi za razvoj QT aplikacija. PyQt je podržan na različite platforme (Windows, Unix/Linux, MacOS)

<https://riverbankcomputing.com/software/pyqt/intro>

## ✓ Poglavlje 5

### Pregled predmeta

## OBRAĐENE TEME

*Teme koje su obrađene na ovom predmetu predstavljaju dovoljnu osnovu za svakog ko želi da nastavi da se bavi programiranjem u Python jeziku,*

Tokom semestra na predmetu [CS324 - Skripting jezici](#), obrađene su sledeće teme:

- Osnove Python 3.x programskog jezika
- Sintaksa Python programskog jezika
  - Promenljive
  - Funkcije
  - Standardni ulaz/izlaz
  - Uslovna grananja
  - Petlje
- Proceduralno programiranje u Python jeziku
  - Korisničke funkcije
  - Strukture podataka
  - Standardni ulaz-izlaz (konzola)
- Objektno-orientisano programiranje u Python jeziku
  - Klase i objekti
  - Atributi i metode
  - Nasleđivanje u Python jeziku
- Rad sa datotekama
- Rad sa bazama podataka
- Obrada izuzetaka
- Debug-ovanje u različitim IDE-ima
- Moduli, paketi i biblioteke u Python jeziku
- Standardna biblioteka
- Ugrađeni paketi u Python jeziku
- Paket NumPy za numeričko programiranje
- Paket pandas za rad sa tabelama
- Paket matplotlib za prikaz podataka
- Paket Pillow za rad sa slikama
- Paket SciPy i SciPy ekosistem
- Python i Data Science
- Python i mašinsko učenje - paket scikit-learn
- Python i razvoj veb aplikacija - paket Flask

Rečeno da je Python jezik koji se može primeniti na veliki broj oblasti upravo zbog svoje modularnosti.

Mnogi od popularnih paketa (prvenstveno za razvoj veb aplikacija) za Python nisu obrađeni na ovom predmetu, jer je nekada obrada jednog paketa toliko obimna da zahteva sopstveni predmet.

Ipak, teme koje su obrađene na ovom predmetu predstavljaju dovoljnu osnovu za svakog ko želi da nastavi da se bavi programiranjem u Python jeziku, i pruža dovoljno znanja za samostalno usavršavanje.

## UNAPREĐENJE PREDMETA

*Popularnost Python programskog jezika čine upravo paketi koji su savremeni i prave razvoj tehnologija.*

Popularnost Python programskog jezika čine upravo paketi koji su savremeni i prave razvoj tehnologija.

Da bi se predmet CS324 - Skripting jezici poboljšao, potrebno je popuniti Google formu.

Iako je potrebna prijava sa Metropolitan nalogom, forma je anonimna.

<https://forms.gle/8F7yikzJm8g8XSs69>

Forma sadrži pitanja o tome koje su teme u okviru predmeta bile više i manje zanimljive, šta je nedostajalo u predmetu, kao i na to na šta treba više, a na šta manje обратити pažnju.

## ✓ Poglavlje 6

### Pokazne vežba #15

#### OPIS TEORETSKOG DELA ISPITA

*Na teoretskom delu ispita mogu se javiti više tipova pitanja.*

- Esejska (duža) pitanja:

Ovakva pitanja zahtevaju od studenata da daju esejski odgovor od jednog do dva pasusa. Ukoliko je potrebno nacrtati i sliku ili dijagram uz odgovor, obično će se naglasiti.

- Esejska (kraća) pitanja:

Ovakva pitanja najčešće zahtevaju od studenata da nabroje sve ili nekoliko stavki vezane za neku temu. Ukoliko je potrebno nacrtati i sliku ili dijagram uz odgovor, obično će se naglasiti.

- Analizirati kod:

Kod ovakvih pitanja od studenata se očekuje da bez računara uspešno analizira kratak kod napisan u Python jeziku i da odgovori na prateća pitanja. Ponekad je potrebno i ručno napisati deo koda ili pseudokoda.

- Pronaći grešku u kodu:

Kod ovakvih pitanja od studenata se očekuje da bez računara pronađe grešku u delu koda koji je zadat, da objasni zbog čega će taj deo koda javiti grešku. Ponekad je potrebno i ručno napisati ispravan kod.

- Pitanja na zaokruživanje:

Kod ovih pitanja student treba zaokružiti tačan odgovor od više ponuđenih odgovora. Zaokruživanje pogrešnog odgovora ne donosi negativne poene.

- Matematičko pitanje ili zadatak iz oblasti rađene u nekoj od lekcija:

Ukoliko je neka od lekcija imala matematički deo (eksplicitna matematička notacija, izvođenje izraza i sl.), od studenta se očekuje da bez korišćenja literature uradi pitanje ili zadatak.

#### OPIS PRAKTIČNOG DELA ISPITA

*Praktični deo ispita sastoji ili iz većeg broja kraćih, ili manjeg broja dužih zadataka.*

Kod praktičnog dela ispita, student radi zadatke u izabranom Python okruženju instaliranom na računaru. Svi zadaci mogu se odraditi u bilo kom razvojnom okruženju.

Studenti mogu koristiti ugrađene opise funkcija u samom IDE-u. Ukoliko se koristi poseban eksterni modul (NumPy, matplotlib, pandas), onda student dobija i spisak najčešće korišćenih komandi iz tog paketa.

Prilikom izrade zadatka, kod treba imati komentare koji prate tok zadatka.

Za pojedine zadatke koje traže daju izlaz, moguće je dati kao primer ulazne podatke, kao i odgovarajuće izlazne podatke.

Datoteke izvornog koda uz odgovarajuće screenshot-ove, treba smestiti u jedan direktorijum, zipovati i poslati predmetnom asistentu i profesoru.

Ukoliko postoji rad sa datotekama treba i sve datoteke poslati zajedno sa izvornim kodom i screenshot-ovima.

- Kratak zadatak koji ispituje jednu funkcionalnost

Kod ovakvih zadatka od studenata se očekuje da napišu program koji će izvršavati datu (najčešće jednu) funkcionalnost.

- Duži zadatak koji ispituje više funkcionalnosti

Kod ovakvih zadatka od studenata se očekuje da napišu program ili programe koji će izvršavati jednu ili više funkcionalnosti, najčešće kao celina.

Praktični deo ispita sastoji ili iz većeg broja kraćih, ili manjeg broja dužih zadatka.

## ✓ Poglavlje 7

### Individualne vežbe

#### INDIVIDUALNE VEŽBE #15

*Individualne vežbe #15 traju 2 časa (90 minuta)*

Individualne vežbe #15 predviđene su za usmenu odbranu projekata, kao i za diskusiju o prethodnim pokaznim i individualnim vežbama.

## ✓ Poglavlje 8

### Domaći zadatak

#### DOMAĆI ZADATAK #15

*Domaći zadatak #15 okvirno se radi 2h*

Domaći zadatak #15 daje se u dogovoru sa predmetnim nastavnikom i/ili asistentom.

**Predaja domaćeg zadatka:**

**Tradicionalni studenti:**

Domaći zadatak treba dostaviti najkasnije 7 dana nakon predavanja, za 100% poena. Nakon toga poeni se umanjuju za 50%.

**Internet studenti:**

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganje ispita. Domaći zadaci se brane!

Domaći zadatak poslati dr Nemanji Zdravkoviću: nemanja.zdravkovic@metropolitan.ac.rs

Obavezno koristiti uputstvo za izradu domaćeg zadatka.

Uz .doc dokument (koji treba sadržati i screenshot svakog urađenog zadatka kao i komentare za zadatak), poslati i izvorne i dodatne datoteke.

## ✓ Poglavlje 9

### Zaključak

## ZAKLJUČAK

### *Zaključak lekcije #15*

#### **Rezime:**

U 15. lekciji najpre obrađene su napredne teme razumevanje listi i imenika. Korišćenjem ovakve sintakse različite for petlje se mogu napisati uglavnom u jednom redu.

Zatim, obrađene su dobre prakse programiranja u Python jeziku, tzv. tips & tricks, koji mogu olakšati svakodnevni rad pri programiranju u Python jeziku.

Nakon dobrih praksi, obrađeno je nekoliko čestih propusta koji programeri u Python jeziku prave.

Konačno, navedeni su popularni paketi u Python jeziku koji nisu detaljno obrađivani u predmetu.

Na kraju je dat pregled predmeta, kao i forma za studente u cilju poboljšanja predmeta.

#### **Literatura:**

- David Beazley, Brian Jones, Python Cookbook: Recipes for Mastering Python 3, 3rd edition, O'Reilly Press, 2013.
- Mark Lutz, Learning Python, 5th Edition, O'Reilly Press, 2013.
- Andrew Bird, Lau Cher Han, et. al, The Python Workshop, Packt Publishing, 2019.
- Al Sweigart, Automate the boring stuff with Python, 2nd Edition, No Starch Press, 2020.

