Zbornik radova Fakulteta tehničkih nauka, Novi Sad

**UDK: 004.9**
**DOI: https://doi.org/10.24867/10BE42Perlinac**

# ANALIZA PROGRAMSKOG JEZIKA PYTHON

# PYTHON PROGRAMMING LANGUAGE ANALYSIS

Tamara Perlinac, *Fakultet tehničkih nauka, Novi Sad*

**Oblast – RAČUNARSTVO I AUTOMATIKA**

**Kratak sadržaj –** *U prvom delu rada ukratko su objašnjeni osnovni pojmovi o programskim jezicima uopšteno i napisane neke opšte stvari o Python-u, a zatim je izvršena sintaksna analiza programskog jezika Python, i opisani su osnovni elementi i koncepti ovog programskog jezika. U drugom delu rada implementirana je studija slučaja, u okviru koje je obađen primer rada sa fajlovima, otvaranja fajla i isčitavanja podataka iz njega, a zatim smeštanja isčitanih podataka u kolekcije programa i kasniju manipulaciju podacima i pristup istim. Studija slučaja implementirana je u programskim jezicima Python i JAVA, analizirane su obe implementacije i povučena paralela između njih.*

**Ključne reči:** *programski jezik, Python, JAVA, sintaksa, manipulacija podacima, kolekcije, datoteke*

**Abstract** – *The first part of this paper briefly explains basic terms of programming languages in general and writes some general things about Python, then performs a syntatic analysis of the Python programming language and describes the basic elements and concepts of this programming language. The second part of this paper is case study implementation , which contains an example of working with files, opening a file and reading data from it, then placing the read data in program collections and subsequent data manipulation and access. This case study was implemented in programming languages Python and JAVA, both implementations are analyzed and compared with each other.*

**Keywords:** *programming language, Python, JAVA, syntax, data manipulation, collections, files*

## 1. UVOD

Kada se govori o programskim jezicima, kao jedna od definicija može se izdvojiti ta da programski jezik predstavlja formalni jezik, koji sadrži set instrukcija koje proizvode različite vrste izlaza, i koji se koriste za pisanje programa pomoću kojih se vrši implementacija raznih algoritama. Opis programskog jezika deli se u dve osnovne komponente: sintaksu (formu) i semantiku (značenje). Postoje razne podele programskih jezika, prema različitim kriterijumima – programski jezici niskog i visokog nivoa (prema stepenu zavisnosti od računara), interpretirani i kompajlirani (prema načinu izvršavanja), jezici opšte namene i specifični za domen (prema opsegu korišćenja). Programski jezik Python predstavlja interpretirani jezik visokog nivoa i opšte namene, osmišljen krajem 80-ih godina XX veka, čija je implementacija započeta 1989. godine u Holandiji, od strane Gvida van Rosuma. Python omogućava razvrstavanje programa u module, koji mogu biti ponovo iskorišćeni u drugim Python programima.

On sadrži veliku kolekciju standardnih modula, koji se mogu koristiti kao osnova novih programa, ili kao primeri za početak učenja programskog jezika Python. Neki od standardnih modula omogućavaju čitanje/pisanje datoteka, sistemske pozive, kreiranje GUI-ja, vezu ka relacionim bazama podataka, itd. Python je proširiv, što znači da je moguće dodati novu funkciju ili modul, koje je potrebno napisati u nekom drugom programskom jeziku (obično je to programski jezik C). Podržava više paradigmi programiranja – u prvom redu imperativnu, objetno-orijentisanu i funkcionalnu paradigmu, i može biti iskorišćen u različite svrhe. Tema prvog dela ovog rada je sintaksna analiza programskog jezika Python – obrada ugrađenih i ostalih tipova podataka, sintakse komentara, kontrole toka programa (petlje), definisanje funkcija, klasa, obrada grešaka i izuzetaka, unosa i ispisa podataka, čitanja i pisanja datoteka i rad sa kolekcijama. U drugom delu rada implementirana je studija slučaja, koja za cilj ima da prikaže upotrebu ovih obrađenih elemenata na primeru, u programskim jezicima Python i JAVA, i upoređivanje dveju implementacija.

## 2. SINTAKSNA ANALIZA JEZIKA

### 2.1. Tipovi podataka

Python poseduje određeni broj ugrađenih tipova podataka, koji se mogu podeliti u sledeće kategorije: tekstualni tipovi (str), numerički tipovi (int, float, complex), tipovi sekvence (list, tuple, range), tip mapiranja (dict), tipovi skupa (set, frozenset), boolean tip (bool), binarni tipovi (bytes, bytearray, memoryview). Tip podatka promenljive u Python-u ne mora biti eksplicitno naveden, ali može. Prilikom dodeljivanja vrednosti promenljivoj, Python joj sam dodeli i tip – u slučaju x = „apple“ Python će promenljivoj x dodeliti tip str, dok u jednakosti x = 6, promenljiva x dobija tip int.

Prilikom inicijalizacije promenljive x može se i eksplicitno navesti njen tip, i u tom slučaju navedene jednakosti bi izgledale ovako: x = str(„apple“) i x = int(6). Python dozvoljava i da se promenljivoj koja je inicijalizovana kao promenljiva jednog tipa, kasnije dodeli vrednost drugog tipa, i to dovodi i do same promene tipa podatka koji nosi promenljiva (tipa promenljive). Tip podatka svakog objekta može se preuzeti pozivom type() funkcije.

**NAPOMENA:**
**Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Petar Marić.**

### 2.2. Komentari

Komentar je razumljivo objašnjenje ili napomena koje se dodaje izvornom kodu programa radi lakšeg razumevanja od strane osobe koja čita programski kod. Programeri takođe koriste komentare kao napomene sebi, obično da opišu logiku pri pisanju nekih komplikovanih funkcija i sl., kako bi se lakše snašli i podsetili posle nekog vremena, kada dođe na red održavanje programa. Oni mogu biti jednolinijski i višelinijski (blok) komentari, i deo programa koji predstavlja komentar ignorisan je od strane interpretera. Jednolinijski komentar u Python-u obeležava se sa # (tarabom), dok višelinijski komentari tehnički ne postoje u ovom programskom jeziku. Postoje docstring-ovi – string koji služi za dokumentaciju Python-a, i on predstavlja literal string koji se obično koristi u klasama, modulima, funkcijama ili definicijama metoda, koji im se pridružuju kao tehnička dokumentacija, i olakšavaju razumevanje funkcionalnosti svake veće celine koda. Početak i kraj docstring-a obeležavaju se sa tri dvostruka navodnika “””, i smešteni su u __doc__ atributu klase, funkcije, modula ili metode za koju su vezani.

### 2.3. Kontrola toka programa

Upravljanje tokom programa predstavlja redosled po kome se naredbe, instrukcije i pozivi funkcija izvršavaju. U programskom jeziku Python nema blokova, koji se u većini ostalih programskih jezika koriste za grupisanje iskaza, već se grupisanje vrši uvlačenjem redova – svi iskazi koji se nalaze u istom bloku treba da imaju isti količinu belog prostora od početka reda. Izjave pomoću kojih se može vršiti kontrola toka programa u Python-u su sledeće: IF...ELIF...ELSE – definišu kontrolnu strukturu, koja može sadržati više blokova izjava od kojih će se izvršiti maksimalno jedna i to ona koja zadovoljava uslov naveden nakon neke od reči IF ili ELIF, ili nakon reči ELSE ukoliko nijedan uslov nije zadovoljen, FOR petlja – služi za iteriranje kroz neku sekvencu, i pomoću nje može se izvršiti jedna ili više izjava, jednom za svaku stavku sekvence kroz koju se iterira ovom petljom, range() funkcija – generiše aritmetičku progresiju i olakšava iteriranje sekvencom brojeva, while petlja – uslovna petlja koja se izvršava sve dok je uslov koji sledi nakon ključne reči while zadovoljen. Za kontrolu toka programa koriste se i break izjava – služi za prekidanje daljih iteracija for ili while petlje u okviru koje se napiše i zaustavlja izvršavanje petlje pre nego što se završi iteracija kroz sve stavke, continue izjava – prekida samo trenutnu iteraciju kroz petlju, i nastavlja iteriranje sa sledećom iteracijom, pass izjava – koristi se da bi sintaksa jezika bila zadovoljena, na mestima gde program ne zahteva nikakvu akciju i else klauzula na petljama – označava deo programskog koda koji će se izvršiti kada se petlja završi.

### 2.4. Definisanje funkcija

Funkcija predstavlja blok organizovanog koda za višekratnu upotrebu, koji se koristi za izvršavanje jedne akcije. Pomoću njih se programski kod može podeliti u organizovane blokove i učiniti bolje organizovanim i ponovo iskoristivim. Što se tiče programskog jezika Python, ključnom rečju def nagoveštava se da će uslediti definicija funkcije. Nakon nje, sledi naziv funkcije, a posle njega u okviru malih zagrada lista parametara funkcije. Linija definicije funkcije završava se dvotačkom, i od naredne linije kreću izjave koje sačinjavaju telo te funkcije (moraju biti pravilno uvučene). Izvršenje funkcije uvodi novu tablicu simbola koja se koristi za lokalne promenljive te funkcije. Tačnije, svaka dodela vrednosti promenljivoj u okviru funkcije čuva se kao nova vrednost u njenoj lokalnoj tablici simbola, i kad god se koristi neka promenljiva, prvo se pretražuje da li postoji u lokalnoj tablici simbola funkcije u okviru koje se zahteva njena vrednost, pa onda dalje. Definicija funkcije uvodi naziv funkcije u tabelu simbola programa. Return izjava služi za vraćanje vrednosti iz funkcije.

Funkcije koje nemaju povratnu vrednost u Python-u vraćaju None – ugrađeno ime za povratnu vrednost funkcije kada nema return izjave. Postoje različite vrste parametara funkcije: obavezni (pozicioni), parametri sa podrazumevanom vrednošću, parametri ključnih reči. Pri pozivu funkcije, neophodno je proslediti sve obavezne parametre, dok se parametri koji imaju podrazumevanu vrednost mogu, ali i ne moraju proslediti. Lambda izrazi su male, anonimne funkcije čija je sintaksa: lambda argumenti: izraz.

Mogu imati proizvoljan broj parametara, a izraz može biti samo jedan. Obično se koriste za manje, jednostavnije delove programskog koda, kako bi uprostile i skratile kod.

### 2.5. Klase

Klasa predstavlja proširivi šablon programskog koda koji služi za kreiranje objekata, podešavanje početnih vrednosti za stanje klase (atribute) i implementaciju ponašanja (funkcije ili metode). Klase predstavljaju sredstvo za objedinjavanje podataka i funkcionalnosti. Svaka instanca klase (objekat) može imati pridružene atribute klase, čije vrednosti određuju trenutno stanje instance, i metode definisane nad klasom, koje se pozivaju nad instancom klase i uglavnom definišu ponašanje koje vrši promenu stanja instance nad kojom su pozvane. Definicija klase počinje ključnom rečju class, zatim sledi naziv klase. U okviru definicije klase pišu se izjave koje najčešće predstavljaju definicije funkcija (ali nije obavezno da budu). Funkcije koje se definišu u okviru klase su njeni članovi i nazivaju se metodama klase – funkcije koje pripadaju nekoj klasi i može im se pristupiti samo preko objekta te klase. Kada se definiše funkcija u okviru klase (metod), kao prvi parametar se prosleđuje self – ključna reč koja označava samu instancu klase (nad kojom je metod pozvan), i pomoću nje pristupa se svim atributima i drugim metodama klase. Konstruktor klase jeste funkcija __init__ koja služi za kreiranje objekta i inicijalizaciju početnih vrednosti atributa. Python podržava i mehanizme nasleđivanja i višestrukog nasleđivanja – prilikom definicije klase potrebno je nakon naziva klase u malim zagradama proslediti jednu ili više klasa koje su roditeljske toj klasi. Sve klase nasleđuju Object klasu.

### 2.6. Greške i izuzeci

Postoje najmanje dve vrste grešaka: sintaksne greške – greške koje nastaju pri kompajliranju programa usled nepoštovanja definisane sintakse jezika (pogrešnog redosleda sekvence znakova ili tokena) i prekidaju izvršenje programa i izuzeci – greške koje nastaju u toku izvršenja programa i nisu bezuslovno fatalne greške –

njima se može rukovati, pošto su oni predvidivi. Nekima od izuzetaka je već rukovano od strane programa, ali većinom nije pa je potrebno da programer sam obradi izuzetak i rukuje njime na željeni način. Postoje različiti tipovi izuzetaka, a kao deo poruke o grešci uvek je ispisano o kom tipu izuzetka se radi. Izuzecima se u Python-u rukuje pomoću try-except klauzule, gde su u okviru try bloka piše deo koda u kom može doći do nekog tipa izuzetka, a nakon ključne reči except navode se tipovi izuzetaka kojima se rukuje, a zatim i način na koji se to može uraditi. Moguće je napisati i korisnički definisane izuzetke – potrebno je naslediti klasu Exception.

### 2.7. Input i output

Podaci koji predstavljaju izlaz programa mogu biti ispisani u ljudski-čitljivoj formi ili upisani u neki fajl, radi trajnog čuvanja istih. Funkcija za ispis podataka jeste funkcija print(), dok se za upis podataka u fajlove koristi funkcija write(). Formatiranje ispisa može se vršiti pomoću formatiranih string literala, format() metode ili ručno. Računarska datoteka ili fajl, predstavlja često korišćen resurs za trajno skladištenje podataka. Python pruža ugrađene metode koje rade sa fajlovima – open(), pomoću koje se otvara datoteka, kojoj se pri pozivu proslede dva argumenta: prvi sadrži ime datoteke koja se otvara, dok drugi predstavlja režim otvaranja datoteke (govori da li se datoteka otvara za čitanje/pisanje/oba, ili pak za dodavanje novog sadržaja na kraj datoteke ili eksluzivno kreiranje). Režim nije obavezan argument poziva, i ukoliko se ne prosledi koristi se režim čitanja. Metod open() vraća fajl objekat.

Druge značajne metode za rad sa datotekama su: read() – čitanje sadržaja otvorene datoteke, readline() – čitanje jedne linije iz datoteke, write(string) – upisivanje prosleđenog stringa u datoteku, tell() – vraća trenutnu poziciju u fajl objektu, seek(offset, whence) – vrši promenu trenutne pozicije otvorene datoteke. U Python-u se često koristi ključna reč with u kombinaciji sa pozivom metode open(), i prednost njenog korišćenja jeste u tome što je datoteka ispravno zatvorena automatski od strane programa kada se rad sa datotekom završi, čak i ako u nekom momentu dođe do nekog izuzetka.

### 2.8. Strukture podataka (kolekcije)

Struktura podataka predstavlja oblik organizacije, upravljanja i skladištenja podataka, koji omogućava efikasan pristup podacima i modifikaciju istih. Najčešće korišćene strukture podataka su liste, dok u Python-u postoje i setovi, rečnici, sekvence itd. Liste predstavljaju uređene kolekcije koje su podložne izmenama i koje mogu sadržati duplikate. U Python-u se prazna lista kreira pomoću uglastih zagrada, a ukoliko je potrebno kreirati listu sa nekim inicijalnim vrednostima, onda se one navode razdvojene zarezom u uglastim zagradama. Elementima liste moguće je pristupiti pomoću indeksa – navođenjem naziva objekta liste i broja u uglastim zagradama koji predstavlja indeks, pristupa se elementu liste koji se nalazi na navedenoj poziciji (indeksiranje ide od nule). Python podržava i negativne indekse – ukoliko je indeks negativan broj, onda se pristupa elementima od kraja liste. Postoji velik broj metoda za rad sa listama, od kojih su najvažnije: append(x) – dodaje element x na kraj liste, insert(i,x) – dodaje element x na poziciju određenu indeksom i, remove(x) – uklanja iz liste element čija je vrednost jednaka x ili uzrokuje izuzetak ValueError (ukoliko element ne postoji u listi), pop([i]) – uklanja element liste koji se nalazi na prosleđenom indeksu ili uklanja poslednji element liste (ukoliko indeks nije prosleđen) i vraća ga, clear() – prazni listu, copy() – kreira kopiju liste, reverse() – preokreće mesta elemenata u listi (element sa poslednjeg mesta prebacuje se na prvo, tj. indeks 0, i tako redom), sort(key, reverse) – koristi se za sortiranje liste u rastućem redosledu, a argumenti određuju po kom ključu se sortira lista i da li rastuće ili opadajuće. Ove metode omogućavaju da se liste po potrebi mogu koristiti i kao LIFO i kao FIFO strukture. Korišćenjem list comprehensions mehanizma omogućeno je kreiranje nove liste u kojoj je svaki element rezultat primene operacija na svaki član druge iterabilne kolekcije. Rečnici su strukture podataka koje su indeksirane pomoću ključeva, koji mogu biti bilo kog nepromenljivog tipa podatka. Rečnik zapravo predstavlja skup ključ:vrednost parova, gde je vrednost ključa jedinstvena u okviru tog rečnika. Najčešće korišćen tip podatka za ključeve u rečnicima su stringovi i brojevi. Prazan rečnik se u Python-u kreira pomoću para praznih vitičastih zagrada, a ukoliko je potrebno ubaciti inicijalne vrednosti u rečnik, onda se one navode razdvojene zarezom kao ključ:vrednost parovi u okviru vitičastih zagrada. Vrednostima iz rečnika pristupa se isto kao u listi, samo što se umesto indeksa šalje vrednost ključa u uglastim zagradama. Izraz recnik[“k1”] = “v1” će u rečnik dodati novi par (“k1”, “v1”) – ukoliko vrednost ključa “k1” ne postoji u rečniku ili će pregaziti postojeću vrednost za ključ “k1”, ukoliko vrednost već postoji, dok se uređeni par iz rečnika briše pomoću ključne reči del – del recnik[“k1”]. Prilikom iteriranja kroz rečnike, ključu i njegovoj vrednosti može se pristupiti zajedno, ako se koristi items() metoda nad rečnik objektom (for (key, value) in recnik.items()).

## 3. STUDIJA SLUČAJA – POREĐENJE UČITAVANJA PODATAKA IZ CSV DATOTEKE I MANIPULACIJE PODACIMA U PROGRAMSKIM JEZICIMA PYTHON I JAVA

CSV (Comma Separated Values) fajl je obična tekstualna datoteka koja sadrži određene podatke, obično razdvojene zarezima. Ove datoteke se koriste kao skladišta podataka, ali isto tako često se koriste za razmenu podataka između različitih aplikacija. Prva linija CSV fajla je linija zaglavlja, koja sa sobom nosi opis semantika podataka koji se nalaze od druge do poslednje linije fajla. Studija slučaja obuhvata primer pisanja programa, koji kao ulaz ima books.csv fajl – fajl koji sadrži podatke o knjigama: autoru, naslovu, godini izdanja, oceni i ceni knjiga, u Python-u i Javi. Program otvara ovaj fajl, učitava podatke iz njega i smešta ih u kolekcije – rečnik, koji kao ključ ima string vrednost autor (ime i prezime autora), a kao vrednost ima listu knjiga koje je taj autor napisao, a zatim prikazuje i opisuje primere iteriranja kroz kolekcije, pristupa i manipulacije podacima, i rad sa njima nakon smeštanja u kolekcije. Python nudi csv modul za rad sa ovim tipom fajla, i on sa sobom donosi mnoge metode i klase koje olakšavaju rad. Otvaranje fajla u Python-u vrši se pomenutom kombinacijom ključne reči with i open metode, bez prosleđivanja režima (otvara se za čitanje što

je po default-u). Prilikom obrade podataka iz csv fajla potrebno je preskočiti liniju zaglavlja. Zatim se iz csv modula koristi klasa DictReader – kojoj se prosledi fajl objekat koji vraća poziv metode open(), i nazivi polja u .csv fajlu. Ovaj objekat mapira informacije u svakom redu .csv fajla u jednu kolekciju tipa rečnik, po ključevima zadatim kroz nazive polja. Dakle, pomoću DictReader-a se od svakog reda .csv fajla kreira jedan rečnik, čiji su ključevi zadati parametrom, a vrednosti se kupe iz svakog reda posebno, i svi oni se smeštaju u listu objekata tipa rečnik. Od ove liste kreira se glavni objekat tipa rečnik, koji će kao vrednost ključa imati autora, a kao vrednost listu knjiga koje je on napisao, i ovo se radi pomoću itertools.groupby funkcije. Ova funkcija kreira iterator koji vraća uzastopne ključeve i grupe za te ključeve iz iterabilne kolekcije koja joj je prosleđena kao prvi argument poziva, i u ovom slučaju će to biti lista objekata tipa rečnik. Drugi parametar poziva funkcije jeste funkcija koja izračuneva ključnu vrednost za svaki element, i upravo ova funkcija zahteva da se iterabilna kolekcija koja se grupiše pomoću groupby funkcije prethodno sortira na osnovu iste ključne funkcije. Zbog ovoga, pre grupisanja liste objekata tipa rečnik i kreiranja glavnog rečnika od njega, potrebno je prvo sortirati listu objekata tipa rečnik po vrednosti za ključ „autor“. Nakon ovoga, postoje razne varijacije na temu kako se može pristupati i manipulisati podacima iz rečnika – prikazati knjige autora Meše Selimovića, ispisati broj knjiga autora Fjodora Dostojevskog, ispisati knjige autora Ive Andrića sortirane po oceni, ili knjige Ive Andrića sortirane po ceni sa ocenom većom od 8 (razni filteri su mogući i dozvoljeni i razlike su minimalne), kao i maksimalno ocenjenu knjigu autora Čarlsa Bukovskog. Ključ rečnika je ime autora, a kako jedna knjiga predstavlja jedan rečnik, tako je vrednost zapravo lista objekata tipa rečnik (lista knjiga) za tog autora. Listi knjiga autora (vrednosti iz rečnika za zadati ključ) pristupa se pomoću get metode – npr listi knjiga autora Meše Selimovića – recnik_knjiga.get(„Mesa Selimovic“), zatim se for petljom prođe kroz sve njegove knjige (objekte tipa rečnik) i ispišu se sve potrebne vrednosti. Ispis broja knjiga određenog autora vrši se pozivom len funkcije nad dobavljenom listom knjiga tog autora, dok se sortiranja u Python-u vrše pomoću ugrađene sorted() funkcije, kojoj se prosleđuju lista koja se sortira, i ključ za sortiranje (zadat pomoću lambda izraza). Sortiranje je podrazumevano rastuće, ukoliko je potrebno uraditi opadajuće onda se sortirana lista u rastućem redosledu samo prosledi reversed funkciji. Primena filtera vrši se tako što se od postojeće liste kreira nova, korišćenjem list comprehensions mehanizma – iz postojeće liste u novu se prebacuju samo oni elementi koji zadovoljavaju određene uslove tj. filtere. Ispis informacija o knjizi je odvojen u posebnu funkciju i poziva se na više mesta u kodu. Za pronalaženje knjige sa najvećom ocenom koristi se max funkcija, kojoj se prosledi lista iz koje traži najveću vrednost (lista knjiga Čarlsa Bukovskog), i ključ po kome to radi – u ovom primeru je to ocena knjige.

Ceo ovaj program u programskom jeziku JAVA implementira se na sličan način, a najveće razlike su u početku – pri otvaranju .csv fajla i smeštanja podataka u kolekcije, dok se primeri manipulacije podacima i pristupa istim ne razlikuju mnogo od Python-a. Prva razlika je što u Javi nema posebnih reader-a koji direktno mapiraju redove fajla u rečnike, pa je potrebno imati pomoćnu klasu Knjiga, gde će biti smeštane informacije o svakoj knjizi posebno. Otvaranje i čitanje fajla u Javi se vrši pomoću BufferedReader-a i FileReader-a, a popunjavanje glavnog rečnika knjiga se vrši ručno – prvo se napravi lista knjiga za autora, a zatim se u rečnik doda ključ-vrednost par. Koriste se objekti tipa HashMap i ArrayList. Listi knjiga za zadatog autora pristupa se pozivom get() metode nad heš-mapom, broj knjiga za autora se dobija pozivom size() metode nad listom knjiga, za definisanje polja nad kojim se vrši poređenje koriste se Javini Comparator-i, dok se sortiranja, filtriranja i pronalaženja maksimuma rade pomoću ugrađenih Collections.sort funkcije, filter() i max() funkcija nad Stream objektima (potrebno je vršiti prebacivanja iz liste u stream i nazad).

## 4. ZAKLJUČAK

Programski jezici Python i Java pružaju veliki broj metoda, funkcija i klasa za rad sa datotekama i kolekcijama, i olakšavaju isti. U konkretnom primeru rada sa .csv fajlom, Python pruža više klasa i funkcija koje obavljaju veliku količinu posla umesto programera, i jednostavnije su za upotrebu od Javinih klasa koje obavljaju isti posao. Samim tim, količina koda potrebna za implementaciju ovih primera manja je dosta u Python-u nego u Javi, i lakša za razumevanje. Implementacija u Javi zahteva definisanje pomoćne klase Knjiga, dok u Python-u svaka knjiga predstavlja jedan rečnik (posledica prednosti korišćenja DictReader-a). Analizom i izučavanjem programskog jezika Python dolazi se do zaključka da je ovaj jezik veoma pogodan i nudi veliki broj prednosti i razloga za njegovo učenje i korišćenje. Ovaj rad obuhvata samo mali deo mogućnosti i razlika koje ovaj programski jezik može da ponudi u odnosu na ostale. Python nudi ogroman broj biblioteka, modula i framework-a, podržan je od strane velikog broja OS, sintaksa je veoma čitljiva, a kod čist i dobro struktuiran. Podržava nekoliko programskih paradigmi, a primenu nalazi u raznim oblastima: programiranju web aplikacija, desktop aplikacija, mašinskom učenju, programiranju mrežnih servera koji rade sa ogromnim količinama podataka, razvoju interaktivnih kompjuterskih igrica i animacija, itd.

## 5. LITERATURA


[1] https:/docs.python.org/3/ - zvanična dokumentacija programskog jezika Python

[2] https:/docs.oracle.com/en/java/ - zvanična dokumentacija programskog jezika Java

[3] https://www.tutorialspoint.com/python


**Kratka biografija:**


**Tamara Perlinac** rođena je u Novom Sadu 1995. god. Srednju ekonomsku školu završila je 2014. godine, i iste godine upisala OAS na Fakultetu tehničkih nauka, smer Računarstvo i automatika. Diplomirala 2018. god i upisala MAS na istom smeru. Master rad na tehničkih nauka iz oblasti Računarstva i automatike odbranila je 2020.god.
kontakt: tperlinac@gmail.com