



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-orijentisanim 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. Ocenjuju 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 ispitu 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 PREDISMITNIH OBAVEZA

Student dobija poene tokom semestra na predismitnim 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 PROJEKTOG 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 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%.**

Odbrana 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.

Odbrana 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 programskih jezika spada u discipline informatike, zavisi ali i utiče na matematiku, softversko inženjerstvo i lingvistiku.

Teorija programskih jezika je grana informatike koja se bavi dizajnom, implementacijom, analizom i klasifikacijom programskih jezika i njihovim pojedinačnim karakteristikama.

Postoje različiti tipovi jezika kao što su:

- Funkcionalni jezici i proceduralni jezici
- Imperativni jezici i deklarativni jezici
- Dinamički jezici i statički jezici
- Jaki jezici i slabi jezici

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?

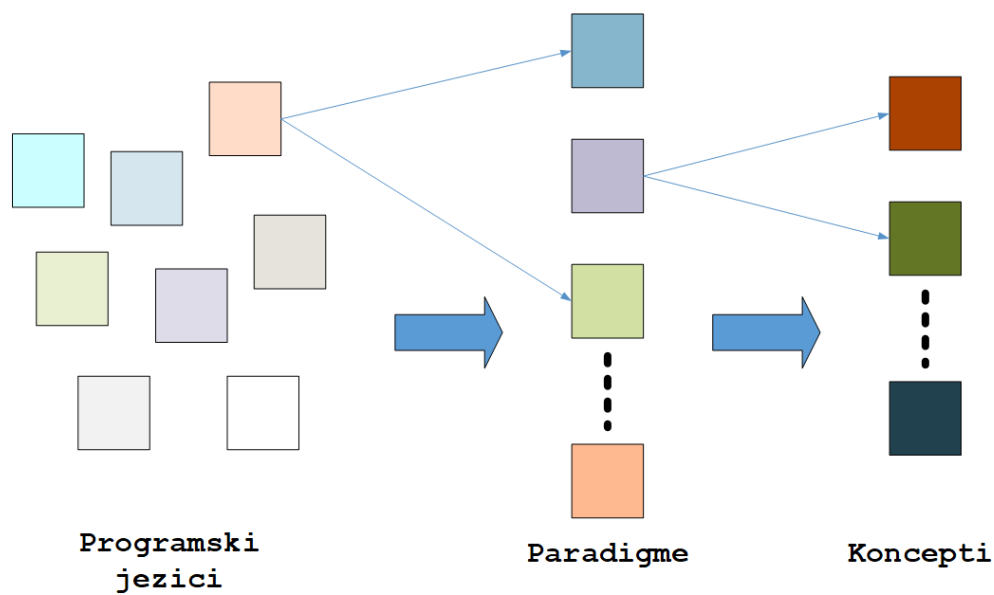
Programski jezik je veštački jezik koji se može koristiti za kontrolu ponašanja računara.

Programski jezik (en. **programming language**) je veštački jezik koji se može koristiti za kontrolu ponašanja računara.

Definisan je preko sintaksnih i semantičkih pravila koja opisuju njihovu strukturu.

Svaki jezik realizuje jednu ili više paradigmi (en. **paradigm**), a svaka paradigma ima jedan koncept (en. **concept**), ili češće, sastoji se od skupa koncepata.

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



Slika 1.1 Odnos programskih jezika, paradigmi, i koncepata. [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-**orjentisano**, dok se deklarativno deli na **logičko** i **funkcionalno**.

Podela 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

Karakteristike programskih 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 programskih jezika su:

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

▼ Poglavlje 2

Imperativno programiranje

DEFICIJA 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 sekvence 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.

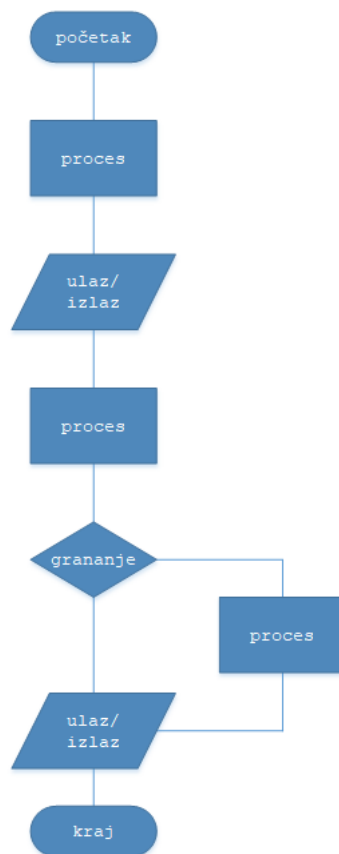
Hardverske implementacije računara su većinom imperativne, to znači da je hardver napravljen da izvršava mašinski jezik napisan u imperativnom stilu.

Imperativno programiranje se deli na proceduralno i objektno-orjentisano 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/](https://en.wikipedia.org/wiki/List_of_programming_languages_by_type#Imperative_languages)

[List_of_programming_languages_by_type#Imperative_languages](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.

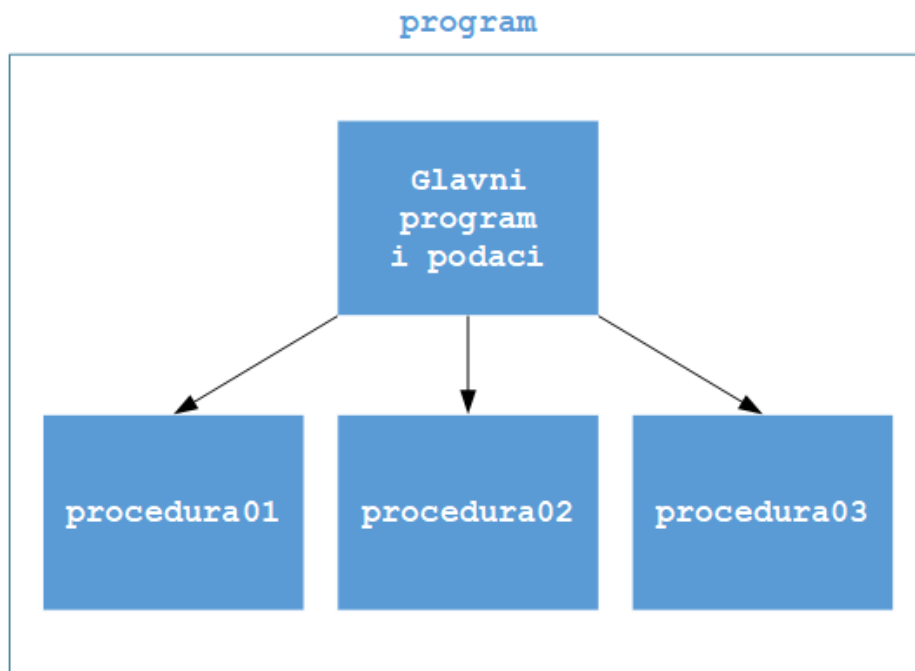
Proceduralno programiranje (en. **procedural programming**) je programska paradigma koja se zasniva na procedurama i njihovom izvršavanju, i to na principu *pozivanja procedure*.

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



Slika 2.2 Proceduralno programiranje. [Izvor: Autor]

DEFINICIJA OBJEKTNO-ORIJENTISANOG PROGRAMIRANJA

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

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

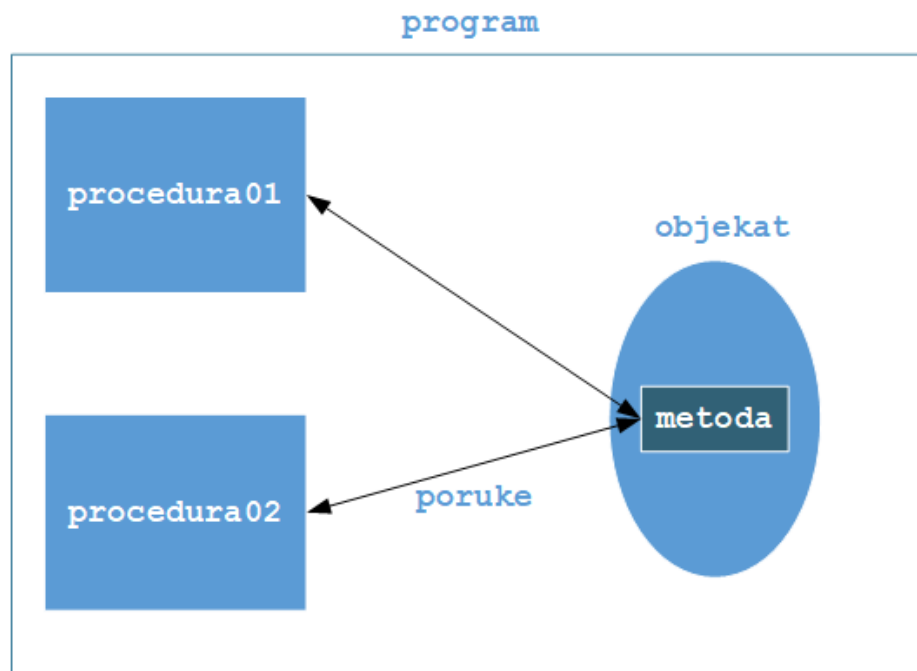
Klase se sastoje od polja podataka i metoda zajedno sa njihovom interakcijom i služe za kreiranje aplikacija i računarskih programa.

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



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 formalnoj logici. Svaki program napisan na jeziku logičkog programiranja predstavlja skup izraza u logičkoj formi koji daju činjenice i pravila za programski domen.

Najpopularniji logički programski jezici 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

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 programski jezici 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

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

Struktorno programiranje

Struktorno 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 prevođenja 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 direktne 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 podstrukture (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, ljske operacionih sistema i nekoliko programskih jezika opšte namene.

Skripta se može biti napisati i odmah izvršiti, tj. "*on-the-fly*", bez eksplicitnih koraka prevođenja (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 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 deklarisan 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 deklarirati 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 interpretera 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 memorijskom upravljačkom sistemu u delu za izvršavanje (en. **runtime**), tj. programskom interpreteru.

Više programa može deliti jedan interpreter, 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 integrisan 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 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.

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 identifikuje 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` dodeljujemo vrednost `"a string"`, i upisujemo sledeći komentar.

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

Konačno, promenljivoj `c` dodeljujemo 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 ljuske (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 ljuska 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 ljuske, **bash**, **csh**, **ksh**, **sh**, kao i na Apple **MPW**.

Korisnici ovih ljuski 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 korisničkih 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 ljuške se mogu lakše pisati, razumeti i modifikovati. Mnogi od standardnih UNIX alata su pisani kao skript ljuške.

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 igrama.

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

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.pcmag.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. **Tool Command Language**) 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 jezik.

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 programskog jezika na računar sa Windows operativnim sistemom. Nakon instalacije, biće dostupna komandna linija Python 3 jezika.

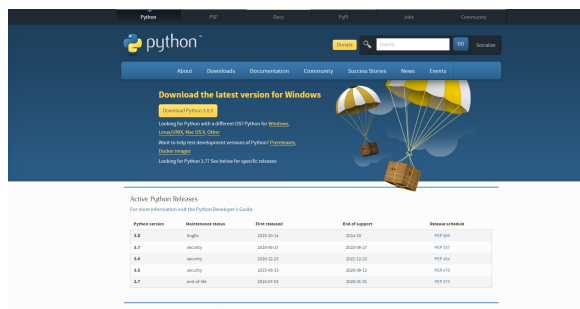
Ovo predstavlja osnovnu instalaciju, dok će u kasnijim lekcijama biti obrađeni konkretno razvojno okruženje za razvoj aplikacija u Python 3 programskom jeziku.

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]

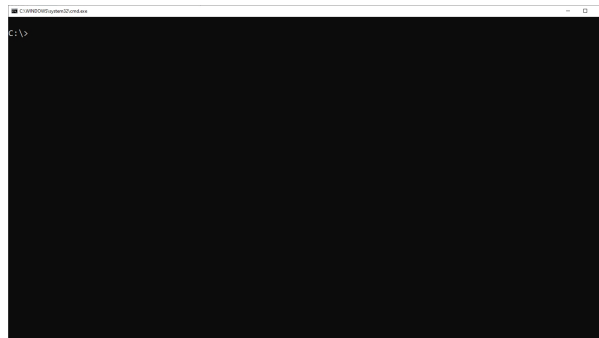
Treba obratiti panju na **Add Python 3 to PATH** koje treba štiklirati. Ostala podešavanja mogu ostati kako jesu. Nakon toga, može se kliknuti na **Install Now**, ili ukoliko želite podesiti dodatne opcije, onda na **Customize installation**

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]

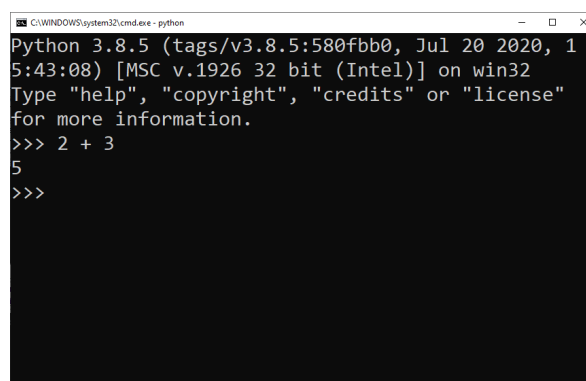
Odmah prilikom pokretanja može se videti koja je i trenutna verzija Python jezika, a i prepoznaje se prompt koji je (po default-u) tri znaka veće: **>>>**

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
```

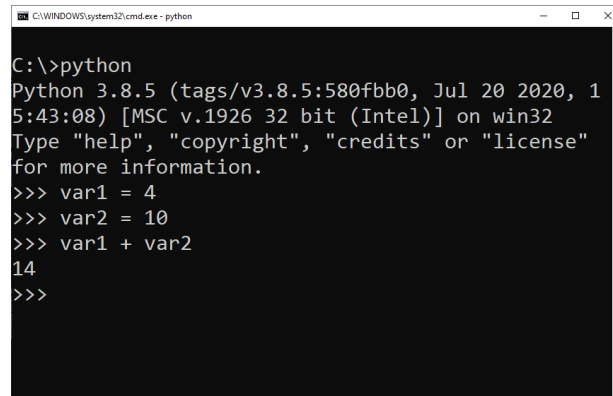


Slika 6.6 Direktno izvršavanje komandi prompt-u. [Izvor: Autor]

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
```



```
C:\>python  
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 1  
5:43:08) [MSC v.1926 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license"  
for more information.  
>>> var1 = 4  
>>> var2 = 10  
>>> var1 + var2  
14  
>>>
```

Slika 6.7 Sabiranje preko promenljivih. [Izvor: Autor]

POKRETANJE PROGRAMA PISANOG U PYTHON JEZIKU PREKO KOMANDNE LINIJE

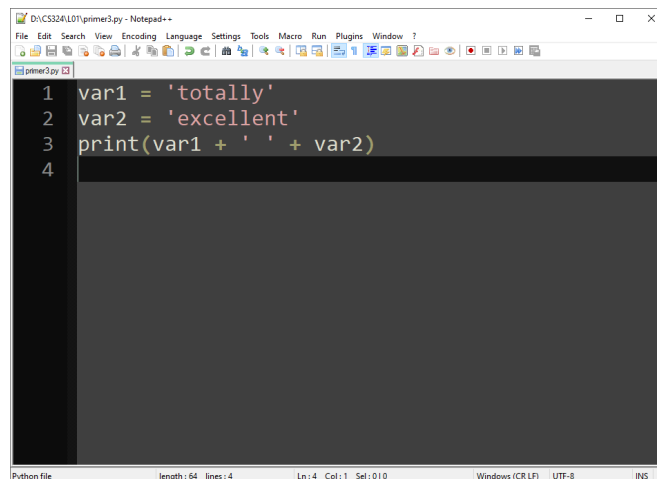
Program koji je već napisan može se direktno pozvati na izvršenje preko komandne linije.

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)
```



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

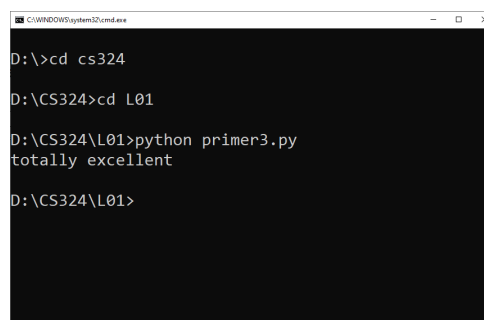
Nakon toga, možemo direktno iz komandnog prozora (bez ulaženja u Python) pokrenuti program tako što ćemo doći do putanje gde se nalazi program (ili ukucati putanju), i pre pokretanja samog programa dodati reč python.

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
```



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

PISANJE PRVOG PROGRAMA U PYTHON 3 JEZIKU - HELLO WORLD!

Vrlo jednostavni 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.

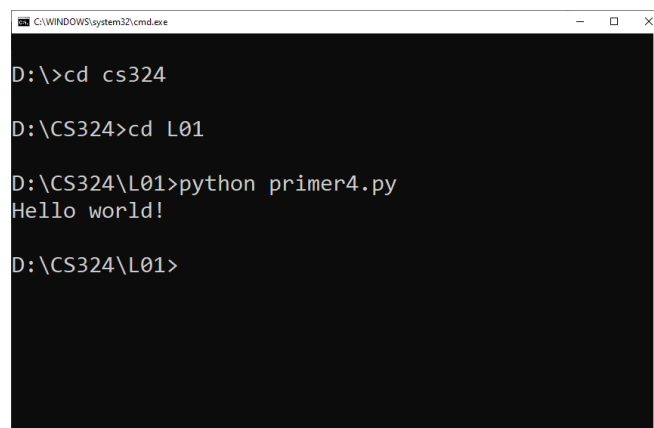
Zatim, isti program napisati preko eksternog tekst editora, sačuvati kao primer4.py, i pozvati na izvršenje direktno iz komande linije.

Rešenje za program:

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

Rešenje za izvršenje preko komandne linije:

```
python primer4.py
```



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

▼ 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.

