SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

DIPLOMSKI RAD

ANALIZA UČENJA OSNOVNIH KONCEPATA PROGRAMIRANJA

Ana Milošević

Split, studeni 2021.

Temeljna dokumentacijska kartica

Diplomski rad

Sveučilište u Splitu
Prirodoslovno-matematički fakultet
Odjel za informatiku
Ruđera Boškovića 33, 21000 Split, Hrvatska

ANALIZA UČENJA OSNOVNIH KONCEPATA PROGRAMIRANJA

Ana Milošević

SAŽETAK

Cilj ovog rada je dobiti uvid u razlike učenja osnovnih koncepata programiranja u odnosu na prethodno učenje kroz osnovnu i/ili srednju školu. Ispitani su stavovi studenata o početnom učenju programiranja, mogućim zaostacima obzirom na kolege koje su prethodno učili programiranje te samo zalaganje studenata na kolegiju Programiranje 1. Svrha istraživanja bila je prikupljenim informacijama ponuditi moguća rješenja za uspješnije shvaćanje osnovnih koncepata studentima koji se nisu tijekom ranijeg školovanja susretali s programiranjem. Stavovi studenata po pitanju „olakšanja“ studentima koji nisu prethodno učili programiranje i prikladnoj dobi za učenje se razlikuju, jednima nije bilo potrebno prethodno predznanje za savladavanje gradiva na kolegiju Programiranje 1, drugi pak smatraju da su bili u zaostatku naspram drugih kolega koji su prethodno se susretali s osnovnim principima programiranja. Također, u radu su obrađeni programski jezici prikladni za poučavanje programiranja, komparativna analiza istih, mogući problemi programera početnika te druge teme vezane uz poučavanje programiranja.

Ključne riječi: Informatika, programiranje, osnovni koncepti, istraživanje, programski jezici, Python, Scratch, Logo, komparativna analiza, obrazovanje, učenik

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 43 stranice, 16 grafičkih prikaza, 8 tablica i 21 literaturnih navoda.

Izvornik je na hrvatskom jeziku.

Mentor: **Dr.sc. Monika Mladenović**, docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Neposredni voditelj: **Dr.sc. Divna Krpan**, viši predavač Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Ocjenjivači: **Dr.sc. Monika Mladenović**, docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Dr.sc. Divna Krpan, viši predavač Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Dr.sc. Goran Zaharija, docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad prihvaćen: Studeni, 2021

Basic documentation card

Thesis

University of Split
Faculty of Science
Department of Computer Science
Ruđera Boškovića 33, 21000 Split, Croatia

ANALYSIS OF LEARNING BASIC PROGRAMMING CONCEPTS

Ana Milošević

ABSTRACT

The aim of this paper is to gain differences in learning basic programming concepts in relation to previous learning through primary and / or secondary school. Student's point of view about the initial learning of programming, possible lags in relation to colleagues who previously studied programming, also the commitment of students to the course Programming 1 was researched. The purpose of the research was to gather information to offer possible solutions for a more successful understanding of basic concepts to students who did not encounter programming during their previous education. The issue of "alleviation" for students who have not previously studied programming and the appropriate age for learning varies from student to student, some did not need prior knowledge to master the material in the course Programming 1, while others believe that they were struggling unlike colleagues who previously encountered the basic principles of programming. Also, the paper deals with programming languages suitable for teaching programming, their comparative analysis, possible problems of beginner programmers and other topics related to teaching programming.

Key Words: Informatics, programming, basic concepts, research, programming languages, Python, Scratch, Logo, comparative analysis, education, student

Thesis deposited in library of Faculty of science, University of Split.

Thesis consist of: 43 pages, 16 figures, 8 tables and 21 references

Original language: Croatian

Supervisor: Monika Mladenović, Ph.D. Assistant Professor of Faculty of Science, University of Split

Co-Supervisor: Divna Krpan, Ph.D. Senior Lecturer of Faculty of Science, University of Split

Reviewers: **Monika Mladenović, Ph.D.** Assistant Professor of Faculty of Science, University of Split

Divna Krpan, Ph.D. Senior Lecturer of Faculty of Science, University of Split

Goran Zaharija, Ph.D. Assistant Professor of Faculty of Science, University of Split

Thesis accepted: November, 2021

ZAHVALE

Prvo, zahvaljujem se mentorici dr.sc. Divni Krpan koja me je svojim znanjima i savjetima usmjeravala na putu do konačne izrade diplomskog rada. Zahvaljujem se svim studentima koji su sudjelovali u istraživanju.

Posebno se zahvaljujem mom Milanu na stalnoj i bezrezervnoj podršci tijekom studija. Dragi moj MG, hvala na svemu, bez tebe bi sve ovo bilo nezamislivo!

Veliko hvala mojoj obitelji na potpori i razumijevanju, posebno mojoj majci koja je uvijek vjerovala u mene. Od srca hvala mojoj kolegici i prijateljici Nikolini. Draga Nik, s tobom je uvijek bilo ljepše ići na predavanja!

Sadržaj

Uvod

Živimo u svijetu u kojem prosječnom čovjeku život bi bio nezamisliv bez svakodnevnog korištenja mobilnih telefona, računala, raznih konzola, bez korištenja Interneta – brzog i jednostavnog pretraživanja informacija, općenito tehnologije koja je daleko prodrla u grane života i djela današnjice. Izuzetno važno je biti informatički pismen jer informatička pismenost više nije prednost, već je imperativ uspješnog čovjeka. Razvojem tehnologije dešavaju se promjene diljem svijeta, susrećemo se s izumiranjem raznih zanimanja, ali i svjedočimo razvoju novih. Prema tome poučavanje predmeta Informatike sve više dobiva na značaju. Investiranje u pojedinca započinje edukacijom.

Djeca od najranije dobi su izložena tehnologiji te smatraju se digitalnim domorocima, ali još uvijek su aktualne rasprave trebaju li uopće biti izložena, a kamoli usvajati koncepte računalne znanosti te učiti koncepte programiranja. Zaista, djeca predškolske dobi su u mogućnosti shvatiti osnovne koncepte programiranja uz dovoljno jasne i pojednostavljene instrukcije, a osim toga brojnim istraživanjima se pokazalo da kroz učenje programiranja razvijaju se mnoge kognitivne vještine.

Svatko od nas želi djetetu pružiti najbolju edukaciju bilo da polazimo od uloge roditelja ili učitelja. Programiranje, kao i primjerice matematika, potiče jedan novi način razmišljanja i bez obzira na to hoće li se dijete jednog dana baviti kiparstvom, kuhanjem, ekonomijom ili će pak biti liječnik, kod djece potiče određene misaone procese koji će im u budućem životu jako dobro doći.

Time se daje poseban naglasak učiteljima, voditeljima obrazovnog procesa, kako odabrati programski jezik koji je prikladan za poučavanje bilo osnovnoškolcima ili općenito programerima početnicima. Jako je važno odabrati programski jezik koji zadovoljava više kriterija prilagođenosti. Potrebno je konstantno približavati koncepte na učenicima shvatljiv način, poticati pozitivnu atmosferu i pružati stalnu motivaciju u radu.

1. Informatika u Republici Hrvatskoj

U Republici Hrvatskoj trenutno na snazi je kurikulum za nastavni predmet Informatike za osnovnu školu i gimnazije iz 2018. godine. Razvojem tehnologije i računalne znanosti stvorila se sve veća potreba za poučavanjem Informatike kao redovnog predmeta, samim time Informatika je predmet kojim se unosi najviše promjerna kurikularnom reformom. Uvođenjem spomenutog kurikuluma, Informatika je obvezni predmet za peti i šesti razred osnovne škole, a već od prvog razreda djeca mogu imati Informatiku kao izborni predmet. Prirodoslovno-matematičke gimnazije i strukovne škole i dalje imaju mogućnost realizacije programa s više sati godišnje. Godišnja satnica u svim razredima je 70 sati.

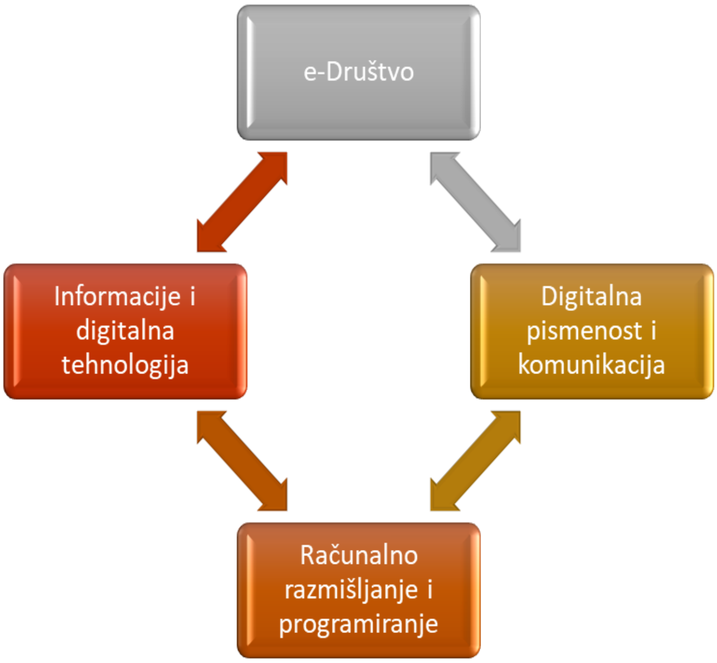
Pod nazivom Informatika u obrazovnom sustavu podrazumijeva se:

- stjecanje vještina za uporabu informacijske i komunikacijske tehnologije (digitalna pismenost) kojom se oblikuju, spremaju, pretražuju i prenose različiti multimedijski sadržaji
- uporabu informacijske i komunikacijske tehnologije u obrazovnom procesu (edukacijska tehnologija, e-učenje)
- rješavanje problema računalom uporabom nekog programskog jezika pri čemu su prepoznatljivi sljedeći koraci: specifikacija i raščlamba problema, analiza problema i odabir postupaka za njegovo rješavanje, priprema i izrada programa, ispitivanje programa i uporaba programa (rješavanje problema i programiranje) (Ministarstvo znanosti i obrazovanja, 2018.).

Sadržaji se trebaju usvajati kroz cijelu školsku godinu, a preporučen je spiralni model koji označava postepeno razvijanje i usvajanje nastavnih sadržaja i područja prvo učenjem na nižim razinama obrazovanja te kasnijim detaljnijim učenjem na višim razinama. Odgojno-obrazovni ishodi, odnosno ciljevi učenja su definirani na način da se učiteljima pruži sloboda odabira sadržaja i metoda poučavanja koje su prilagođene određenom području i sposobnostima samih učenika. Od učenika se očekuje da postanu informatički pismeni kako bi se mogli pripremiti na učenje, život i rad u digitalno razvijenom društvu. Ishodi su napisani kao informatičke kompetencije učenika koje se razvijaju kroz različite obrazovne situacije.

1.1. Domene realizacije predmeta Informatike

Četiri su domene kojima će se realizirati ciljevi predmeta Informatika: e-Društvo, Digitalna pismenost i komunikacija, Računalno razmišljanje i programiranje te Informacije i digitalna tehnologija. Domene su međusobno povezane, određeni sadržaji se mogu razmatrati u više njih. Domena e-Društvo se bavi područjem zaštite podataka i sigurnosti na mreži te elektroničkim nasiljem – jako bitna stavka i u samom odgoju djece. Cilj je oblikovati odgovornog, kompetentnog i pouzdanog korisnika digitalne tehnologije. Nadalje, domena Digitalna pismenost i komunikacija je središte svih domena, tj. svaka domena je povezana s njom. Nastoji pružiti podlogu u razvijanju komunikacijskih vještina te stvoriti pozitivnu svijest prema budućoj tehnologiji. Informacije i digitalna tehnologija bavi se područjima kao što su pohrana i prijenos podataka uporabom računala, digitalnih uređaja i mreža. Naposlijetku, domena Računalno razmišljanje i programiranje dobila je najveći naglasak u ovoj cijeloj reformi. Ciljevi su razvijati računalno razmišljanje, poticati kritičko mišljenje, kreativnost i inovativnost, razvijanje sposobnosti rješavanja problema i usvajanje vještina programiranja.



Slika 1.1 Odnos domena

1.2. Informatika u drugim zemljama

Na području Europe osnovana je neprofitna organizacija učenog društva *Informatics Europe* koja predstavlja udruženje sveučilišnih odjela i istraživačkih laboratorija na području informatike. Sjedište se nalazi u Zurichu, u Švicarskoj. Formirana je 2006. godine i sastoji se od 140 i više institucija, a od toga su 32 zemlje. Glavni cilj je obrazovanje, istraživanje te prijenos znanja na polju informatike.

Navedena organizacija potiče više radnih grupa koje se bave određenim temama kako bi oblikovale strateške prioritete unutar europske zajednice pa tako, između ostalih, ima i posebna grupa žena u informatičkom obrazovanju. Svake godine dodjeljuju se dvije nagrade kojima se ističu inicijative koje unapređuju kvalitetu istraživanja i obrazovanja iz informatike u Europi. (ME Caspersen, 2018)

Pitanje osnovnog obrazovanja, digitalne pismenosti i općenito obuke samih učitelja područja informatike je, također, u domeni rada organizacije. Doneseno je izvješće o navedenim disciplinama u kojem se pokazalo kako je velik nedostatak nastave Informatike, bilo obvezne ili izborne, diljem europskih škola. Ne nudeći osnovno informatičko obrazovanje narušava se edukativno i ekonomsko stanje učenika. Naime, organizacija se zalaže za definiranje kurikuluma predmeta Informatike na nacionalnoj razini, ali usprkos tome još je mali postotak zemalja koje definiraju vlastite kurikulume na razinama pojedinih škola.

2. Programiranje

Što je to programiranje? Da bi se započela bilo kakva daljnja priča potrebno je odrediti neke definicije programiranja. Dakle, programiranje je računalom potpomognuta vještina rješavanja problema, razvijanja logičkog i računalnog razmišljanja te otklanjanja greški. Najvažniji dio programiranja je kontrola tijeka programa. Ona obilježava na koji način će se program ponašati u zadanim okolnostima. Jedan od osnovnih elemenata kontrole tijeka su petlje, to su strukture koje omogućuju višestruko ponavljanje određenog dijela programskog koda.

Istodobno, može se reći i da je programiranje umijeće stvaranja programa na računalu. Pojam programer označava osobu koja stvara program. Učenje programiranja znači izgradnju mehanizama i objašnjenja. Svaki program ima dva čitatelja, a to su računalo i čovjek, odnosno to znači da naredbe programa pretvaraju računalo u mehanizam koji diktira kako riješiti problem, dok programer treba objasniti zašto i kako program rješava i izvršava određeni problem. Soloway i Ehrlich su definirali da je znanje programiranja kao set struktura blokovskog oblika koje nazivamo planovima. (E Soloway, 1988.)

Razvoj računalnog razmišljanja i učenje programiranja su vještine koje pridonose procesu učenja kod djece te im pomažu da se suoče sa situacijama u stvarnom životu. (FJ García Peñalvo, 2016.) Također, učenje programiranja podrazumijeva primjenu naučenog za rješavanje problema u novonastalim situacijama. Neki od glavnih razloga zašto bi se tijekom obrazovanja trebalo obratiti pažnju i bazirati se na programiranju bili bi: cjeloživotno učenje i interes, razvoj organizacijskih vještina, stjecanje novih znanja i vještina koje su potrebne za kompetentnost pri razvoju tehnologije te općenito širenje zajednice korisnika tehnologije. (Passey, 2017)

Postoje tri perspektive interesa u poučavanju programiranja, a to su:

- inženjeri – oni žele poboljšati programsku produktivnost
- nastavnici – oni poboljšavaju početno razumijevanje programiranja i početničke sposobnosti
- znanstvenici – oni pomažu pri rješavanju kompleksnih problema pomoću vještina

Sedamdesetih i osamdesetih godina prošlog stoljeća programiranje je izazivalo velik entuzijazam, dok danas stav je drugačiji. Konstantnim razvojem tehnologije današnja djeca i općenito nove generacije su u dodiru s jačim uređajima, poput mobilnog telefona, nego što je to nekad bilo snažno računalo koje se koristilo na sveučilištima. Samim time, pojam programiranja bio je vezan uz računanje, tj. na programiranje se gledalo kao na tehničku aktivnost koja je primjerena tek manjoj grupi ljudi. Nešto kasnije programiranje simbolizira aktivnost koja označava oblikovanje ponašanja računala koja pomaže ljudima ili umjesto njih obavlja neke intelektualne zadatke.

2.1. Programski jezici prikladni za poučavanje programiranja

Stav učenika, općenito programera početnika, prema programiranju jest da je programiranje teško. Većina „velikih“ stvari u životu koje radimo po prvi put su zaista teške i može se reći apstraktne – tako možemo shvatiti i programiranje. Mnogo njih odustane nakon prvih problema koji nastaju učenjem programiranja i iz tog razloga razvila se potreba olakšanja prvog susreta s programiranjem. Tijekom brojnih rasprava uvijek je bilo prisutno pitanje koji jezik koristiti u uvodnom programiranju. George Milbrandt je predložio slijedeće karakteristike (M McCracken, 2001.) koje su poželjne za programske jezike koji se koriste u srednjim školama:

- jednostavan za korištenje
- strukturiran dizajn
- velike računalne sposobnosti
- jednostavna sintaksa
- deklaracija varijabli
- jednostavan ulaz – izlaz i oblikovanje izlaza
- dopuštanje izražajnih imena varijablama
- neposredan povrat informacija
- dobri alati za testiranje i ispravljanje pogrešaka

Manilla i de Raadt su u svom istraživanju *An Objective Comparison of Languages for Teaching Introductory Programming* postavili popis od 17 kriterija koji povezuju programski jezik s aspektima učenja programiranja. (L Mannila, 2006.) Kao prvi kriterij navode da jezik treba biti pogodan za poučavanje. Jezici koji se koriste u široj upotrebi ne znači da su i nužno najbolji za poučavanje. Programeri početnici trebaju se posvetiti učenju koncepata, a ne same sintakse jezika i time jezik bi trebao biti nenametljiv, jednostavne sintakse. Jezik bi trebao omogućavati korištenje za primjenu fizičke analogije što znači da bi trebao osigurati multimedijalne mogućnosti bez dodatnih proširenja. Nadalje, jezik bi trebao pružati opći okvir. Primarno je upoznati učenike s programiranjem i time im pružiti mogućnost učenja drugih i složenijih jezika. Jezik bi trebao promovirati novi pristup poučavanja *software-a,* primarni cilj je podrška razmišljanju o problemima i njihovim rješenjima. (L Mannila, 2006.)

Između ostalog, treba obratiti pažnju i na aspekte jezika koji se odnose na dizajn i okruženje u kojem se jezik može koristiti. Jezik bi trebao biti interaktivan i olakšavati razvoj koda, omogućavati implementaciju ideja bez utvrđivanja konteksta cijele primjene. Jezik bi trebao promicati pisanje ispravnih programa što označava davanje osiguranja da je napisani kod točan i da je bez greški. Također, jezik bi omogućavao rješavanje problema u manjim dijelovima, trebao bi podržavati modularnost (funkcije, procedure…). Jezik bi trebao pružati besprijekorno razvojno okruženje. Učenici, početnici u programiranju, trebaju razumjeti proces koji izvorni kod prevodi u izvršni program. Prednost je da razvojno okruženje ima intuitivni GUI (eng. *graphical user interface*). (L Mannila, 2006.)

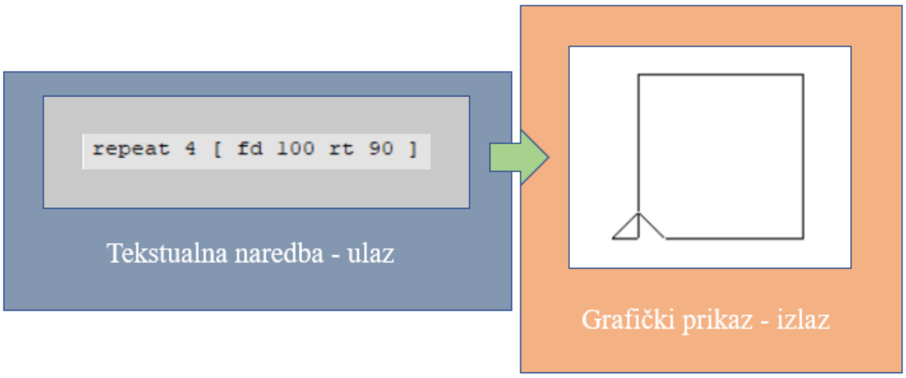
Zatim, kriteriji podrške i dostupnosti se odnose na to da jezik ima podršku zajednice korisnika. Jezik postaje dugovječan širom zajednicom korisnika poput profesora, studenata i drugih zainteresiranih korisnika kao njegovom podrškom. Jezik bi trebao biti otvorenog koda tako da svatko može doprinijeti njegovu razvoju. Obično je to rezultat rada neke grupe koja ne želi u konačnici stvoriti komercijalni proizvod. Jezik je dosljedno podržan kroz različita okruženja, treba biti dostupan na različitim platformama. Nadalje, još jedan nezanemariv kriterij je da bi jezik trebao biti lako dostupan širom svijeta bez ograničenja. Jezik bi trebao biti podržan dobrim nastavnim materijalima. Udžbenici trebaju biti dostupni za korištenje. (L Mannila, 2006.)

Osim obrazovanja, jezik se može koristiti i u drugim područjima, npr. industriji. Jezik bi trebao biti proširiv te pouzdan i učinkovit. Trebao bi omogućiti stvaranje aplikacija velike brzine. Također, jezik ne bi trebao biti primjer *QWERTY* fenomena da pokaže kako ima korisnost i u budućnosti. (L Mannila, 2006.)

2.2. Podjela programskih jezika za poučavanje

Za potrebe poučavanja programiranja nastali su pojednostavljeni programski jezici. Primjer takvog jezika jest Logo, *Kornjačina grafika*. Nastao je još 60-ih godina prošlog stoljeća od strane *Massachusetts Institute of Technology*, namijenjen djeci. (Watt, 1982.) Konstruira se na ideji da učenik upravlja kretanjima glavnog lika, kornjače. Kornjača se može kretati po ekranu pomoću naredbi poput *fd* (eng. *forward*), *bk* (eng. *back*), *lt* (eng. *left*) te *rt* (eng. *right*).

Ulaz programa su tekstualne naredbe, a izlaz je grafički prikaz. Takvi jezici se svrstavaju kao **vizualno-tekstualni programski jezici**. Na slijedećoj slici se može vidjeti primjer crtanja kvadrata u programskom jeziku Logo:

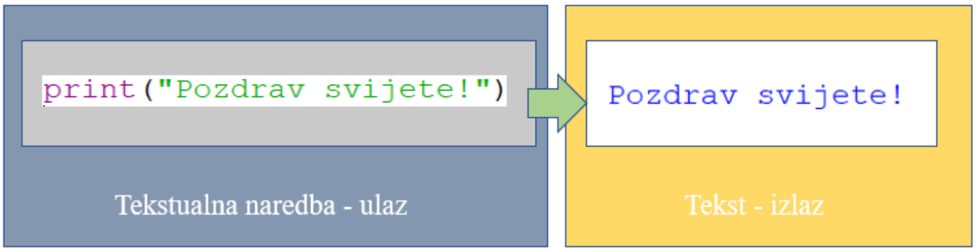


Slika 2.1 Primjer ulaza i izlaza u programskom jeziku Logo

Ministarstvo znanosti i obrazovanja Republike Hrvatske preporučuje upravo Logo, kao prvi programski jezik s kojim se djeca susreću.

Slijedeća grupacija programskih jezika bila bi **proceduralno-tekstualni programski jezici**. Osnovna karakteristika takvih jezika jest ulaz u obliku tekstualne naredbe te izlaz, također u tekstualnom obliku. Najbolji primjer takvih jezika koji je postao najrašireniji jezik za početno učenje programiranja u višim razredima osnovne škole – Python. Tvorac Python-a je Guido van Rossum, 1990. godine. Zanimljiva činjenica o Python-u jest da je ime dobio po televizijskoj seriji *Monty Python's Flying Circus*.

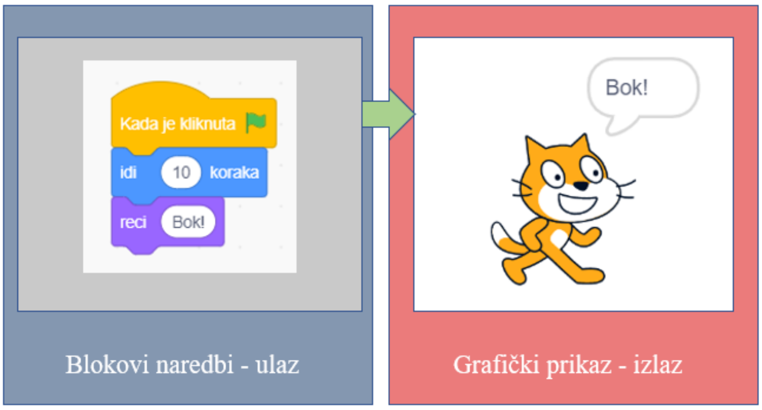
Prilikom kodiranja u Pythonu potrebno se usredotočiti na uvlačenje koda i korištenje razmaka, a kao velika prednost, zahtjeva minimalnu upotrebu posebnih znakova. (Zelle, 2004.) Uglavnom zadaci i problematika kojom se učenici bave jest rješavanje matematičkih problema. Pruža mogućnost razdiobe problema na više manjih te u konačnici njihovo povezivanje u cjelinu. Na slijedećoj slici se može vidjeti najjednostavniji primjer programa pozdrava u Python-u:



Slika 2.2 Primjer ulaza i izlaza u programskom jeziku Python

Nadalje, **vizualno-blokovski programski jezici** su jezici u kojima su ulaz i naredbe prezentirani blokovima koji se spajaju u obliku slagalice, a izlaz je grafički prikazana kretnja lika. Takav princip rada naziva se *drag & drop*, u prijevodu *povuci & ispusti*. (D Krpan, 2014.) Takvi jezici imaju poveznicu s prethodno opisanim jezikom, Logo, a razlikuju se po ulaznim naredbama. Primjer vizualno-blokovskih programskih jezika je Scratch. Scratch je nastao 2003. godine pod vodstvom Mitch Resnika.

Podržava stvaranje interaktivnih priča, igara i animacija. Blokovi naredbi podijeljeni su u više skupina koje predstavljaju pokret, izgled, zvuk, događaje, upravljanje, očitanja, operacije, varijable i u konačnici pruža izradu vlastitih blokova. (O'Neill, 2018.) Razlikuju se po boji čime je učenicima uvelike olakšan postupak pronalaska i spajanja blokova. Problematika rješavanja zadataka u Scratch-u nije izravno rješavanje matematičkih problema, iako su prisutni, prekriveni su u obliku kreiranja određenih priča. Na slijedećoj slici se može vidjeti najjednostavniji primjer programa kretnje i pozdrava u Scratch-u:



Slika 2.3 Primjer ulaza i izlaza u programskom jeziku Scratch

Naravno, programskih jezika za poučavanje ima mnogo. Logo, Python i Scratch su izabrani kao reprezentativni primjer jezika koji se najčešće koriste u školama na području Republike Hrvatske i šire.

2.3. Komparativna analiza programskih jezika za poučavanje

Na temelju prethodno objašnjenih kriterija u prošlom poglavlju o prikladnosti programskih jezika za poučavanje programiranja prikazana je usporedba programskih jezika Logo, Scratch te Python. (L Mannila, 2006.)

Tablica 2.1 Usporedni prikaz programskih jezika za poučavanje programiranja

Kriterij / Programski jezik	**Logo**	**Scratch**	**Python**
pogodan za poučavanje	jednostavna sintaksa i semantika jezika pogodan za početnike jednostavna instalacija	jednostavan *drag & drop* princip dovlačenja blokova naredbi spajanje blokova nalik slagalici, pogodan za početnike	jednostavna sintaksa i semantika jezika pogodan za početnike jednostavna instalacija

		jednostavna instalacija, a može se koristiti i iz preglednika	
pruža opći okvir	temelji se na osnovnim konceptima programiranja nema mogućnost poučavanja OOP pristupom	temelji se na osnovnim konceptima programiranja nema mogućnost poučavanja OOP pristupom	temelji se na osnovnim konceptima programiranja mogućnost poučavanja OOP pristupom
interaktivan i olakšava razvoj koda	intuitivno razvojno okruženje	intuitivno razvojno okruženje	intuitivno razvojno okruženje
promiče pisanje ispravnih programa	program daje povratnu informaciju da ne prepoznaje dio koda u slučaju sintaksnih pogreški, ali bez detaljnijih pojašnjenja	gotovi blokovi eliminiraju sintaksne pogreške	program javlja sintaksne pogreške
modularnost	podržava modularnost procedurama	mogućnost izrade vlastitih blokova	podržava modularnost funkcijama
podrška zajednice korisnika	globalna zajednica korisnika	globalna zajednica korisnika, izrađen posebno za dob od 8 do 16 godina, ali koriste ga ljudi svih	globalna zajednica korisnika

		dobnih skupina	
otvorenog koda	da	da	da
podržan kroz različita okruženja	dostupan je na različitim platformama	dostupan je na različitim platformama	dostupan je na različitim platformama
dostupnost širom svijeta bez ograničenja	besplatan ima jaku međunarodnu zajednicu korisnika s više od desetljeća iskustva u školstvu	besplatan koristi se u više od 150 različitih zemalja i dostupan je na više od 60 jezika	besplatan dostupan svima, potiče se na sudjelovanje korisnika diljem svijeta
podržan nastavnim materijalima	službena dokumentacija programa postoje razni školski udžbenici, priručnici i vodiči pisani na više jezika	službena dokumentacija programa postoje razni školski udžbenici, priručnici i vodiči pisani na više jezika	opsežna službena dokumentacija postoje školski udžbenici i priručnici, pisani na više jezika
može se koristiti i u drugim područjima	pruža podršku za istraživanje različitih disciplina uključujući matematiku, inženjering, umjetnost, glazbu i robotiku	namijenjen je za edukacijske svrhe pomaže mladim ljudima kreativno i sustavno razmišljati te međusobno surađivati	sadrži mnoge biblioteke koje osim edukacije pružaju razvoj aplikacija za analizu i modeliranje podataka, strojno učenje napredni sustavi za upravljanje sadržajem, okviri za programiranje asinkrone mreže,

			razvoj igara često se koristi kao jezik za podršku programerima za kontrolu i upravljanje izgradnjom te testiranje također se koristi za izgradnju ERP i sustava e-trgovine
proširiv, pouzdan i učinkovit	eventualna buduća proširenja i nadogradnje su besplatne	eventualna buduća proširenja i nadogradnje su besplatne	eventualna buduća proširenja i nadogradnje su besplatne starije verzije rade bez obzira na nova izdanja

2.4. Problemi početnog programiranja

Prvo, krenimo od tehničkih problema izvođenja samog predmeta Informatike u osnovnim i/ili srednjim školama. Škole nisu sve podjednako opremljene, a ni grupe učenika nemaju određeni fiksni broj. Time može doći do toga da za svakog učenika nema predviđeno odgovarajuće računalo pa se obično jedno računalo zajednički koristi među dva ili više učenika. Takvi problemi na koje učenici ne mogu utjecati mogu negativno utjecati na motivaciju. Na fakultetima spomenuti problemi su rjeđi ili pak nisu slučaj.

Osim tehničkih problema, postoje određene miskoncepcije koje su specifične za programere početnike. Ranije stečena znanja mogu omesti novo učenje. U tom slučaju postojeće znanje može interferirati s novim. (Ž Žanko, 2019.) Prema konstruktivističkoj teoriji učenja znanje se izgrađuje iz osobnih iskustava. Učenje je rezultat socijalnih interakcija time znači da staro znanje utječe na novo. Primjerice, matematičko znanje može dovesti do zabluda pri programiranju. Ako pogledamo izraz $x=x+1$, matematička je besmislica dok u programiranju itekako ima smisla.

Postoje određeni problemi semantike jezičnih konstrukcija, mnoge su konstrukcije programskog jezika imenovane prirodnim jezikom, pa se početnici zbune, npr. *while* – ima različito značenje. Početnici pretpostavljaju da će računalo razviti slične interpretacije kao što sami žele. (J Gal-Ezer, 2004) Također, javlja se problem nekonzistentnosti kada učenici, početnici, pretpostavljaju da će određena konstrukcija funkcionirati u različitim situacijama, ako je radila za jednu situaciju.

Ciljno-planska analiza programa se temelji na cilju – što se mora ostvariti za rješavanje problema te planu – koji su načini rješavanja. Naime, problemi koji stvaraju poteškoće pri sastavljanju ispravnog plana su:

- problem paralelnosti - zbrajanje složenih kombinacije planova u terminima nekih primarnih funkcija previđajući implikacije složenih sekundarnih funkcija na kasnijim kompozicijama plana
- problem optimizacije – da li je moguća optimizacija uopće
- problem prijašnjih iskustava – recikliranje planova temeljenih na prijašnjim iskustvima
- problem prilagodbe – nepravilna prilagodba u novoj situaciji
- problem prirodnog jezika – pridruživanje prirodnog jezika u programski stvara pogreške
- problem interpretacije – ovisi o postojećem znanju
- granični problemi – poteškoće u odlučivanju o odgovarajućim graničnim točkama
- problem neočekivanih slučajeva – problemi koji rade točno za neke obične slučajeve, ali ne i generalno
- kognitivni problemi – mali, ali važni dijelovi planova mogu biti izostavljeni

Nezanemariva činjenica jest da neki učenici i/ili studenti ponekad sami namjeste ulazne parametre kako bi program radio. Važan je sam proces programa, ne samo krajnji rezultat izvođenja. Dakle, očito je da je glavni nedostatak programera početnika posjedovanje manje programskog znanja i nepostojanje vještina.

2.4.1. Vještine čitanja i pisanja koda kod programera početnika

Prema studiji za procjenu vještina programiranja studenata prve godine informatičkih znanosti iz 2001. godine istraženo je koliko su zapravo studenti vješti, sposobni i koliko posjeduju znanja nakon završenih uvodnih kolegija informatičkih znanosti. Istraživanje je provedeno na nekoliko sveučilišta. (M McCracken, 2001.)

Rezultati su bili poražavajući, odnosno ispod očekivanja. Naime, pokazalo se da nekolicina studenata ipak ne zna programirati na očekivanoj razini. Shodno tome, dana su moguća objašnjenja, a to su da studentima nedostaju sposobnosti rješavanja problema, generalno shvaćanje, raščlamba te da studenti posjeduju slaba znanja o osnovnim konceptima i principima programiranja. Nailaze na probleme u rutinskim zadacima. Studenti se više fokusiraju na sintaksu programskog jezika, a najčešća greška je nemogućnost izdvajanja problema u zadatku.

Zašto se studenti muče s programiranjem? Što utječe na nedostatak znanja? Znači, cilj nakon odslušanih uvodnih kolegija programiranja je steći studente koji su sposobni naučiti proces u rješavanju problema. Taj se proces sastoji od pet koraka prema McCrackenu:

1. definiranje problem iz opisa
2. raščlamba problema na manje probleme
3. rješavanje manjih problema
4. sastavljanje u cjelovito rješenje
5. ocjenjivanje postupka i ponavljanje

Također, 2004. godine ITiCSE (eng. *Innovation and Technology in Computer Science Education*) provela je istraživanje u sedam zemalja. Studentima su dana dva tipa zadataka u kojima su trebali predvidjeti izlaz izvršavanja određenog dijela koda te nadopuna programa s dijelom koji nedostaje. Uglavnom su bolje bili riješeni zadaci prvog tipa. Analizom bilješki studenata tijekom ispitivanja ustanovljeno je da su bolje prolazili studenti koji su pisali bilješke u obliku tablica i grafova, koji su zapisivali sve promjene vrijednosti varijabli, a najlošije su prolazili oni studenti koji su rješavali „napamet“, odnosno oni koji nisu pravili bilješke tijekom ispitivanja. (R Lister, 2004.)

Dakle, zaključeno je da studenti nemaju problema s pisanjem koda nego s praćenjem istog. Ključno je razviti vještine analize i čitanja koda kako bi ostvarili preduvjet za rješavanje problema. Pravilan način učenja programiranja jest automatizacija analize i čitanja koda pa tek onda krenuti na rješavanje problema.

2.4.2. Kognitivni procesi učenika prilikom programiranja

Već je ranije spomenuto u radu da programiranje označava rješavanje problema, otklanjanje grešaka te razvijanje logičkog i računalnog razmišljanja. Ne znači samo pisanje programa na računalu i upravo zbog toga se može reći da programiranje mijenja način razmišljanja. Isto tako, programiranjem učenici razvijaju i kognitivne vještine. Procesom učenja kroz koji učenici prolaze u dodiru s programiranjem, nezanemarivo je kognitivno opterećenje. Prema Teoriji kognitivnog opterećenja tumači se da višak informacija može ometati učenika i da se poteškoće stvaraju pri prisutnosti neiskorištenih elemenata. Spomenutu teoriju razvio je Sweller 1988. godine te postavio tri glavna čimbenika na kojima se temelji: razina prethodnog znanja pojedinca, složenost sadržaja koju pojedinac treba usvojiti i vrsta materijala za učenje. (Sweller, 1988.)

Također, teorija polazi od pretpostavke da se proces pohrane informacija dijeli na kratkotrajno i dugoročno pamćenje. Kratkotrajno pamćenje se još naziva i radno pamćenje ili radna memorija, ono govori s koliko informacija se može baratati te izrazito je ograničeno. Drugim riječima, nakon što se informacija obradi u mozgu, ako se s njom ništa ne upravi ona nestaje. Dugoročno pamćenje je daljnja faza procesa pohrane informacija, što znači da bilo koja informacija koja je stigla do ove faze ujedno tu ostaje trajno. Naime, sve čega se možemo prisjetiti nakon dvadeset sekundi nalazi se u dugoročnom pamćenju. Kapacitet i trajanje praktično su neograničeni. Kako dolazi do kognitivnog opterećenja učenika? Dakle, nastaje elementima kratkotrajnog pamćenja koji su potrebni za učenje određenog zadatka, tj. nastaje mentalno opterećenje koje može biti prevladano sadržajem i strukturom zadatka. (Sweller, Cognitive load theory, 2011.)

2.4.3. Metode poučavanja programiranja

Metoda poučavanja je obrazac ponašanja koji se može sustavno primjenjivati u različitim nastavnim područjima s ciljem olakšavanja i poboljšanja ishoda učenja. Jedna od danas najzastupljenijih metoda poučavanja je izravno poučavanje koje se odnosi na aktivnosti koje zahtijevaju neposrednu uključenost nastavnika, sudjelovanje i interakciju sa studentima i to uz upotrebu izlaganja, opisivanja, objašnjavanja, ispitivanja, modeliranja i prikazivanja. (S Mohorovicic, 2011.)

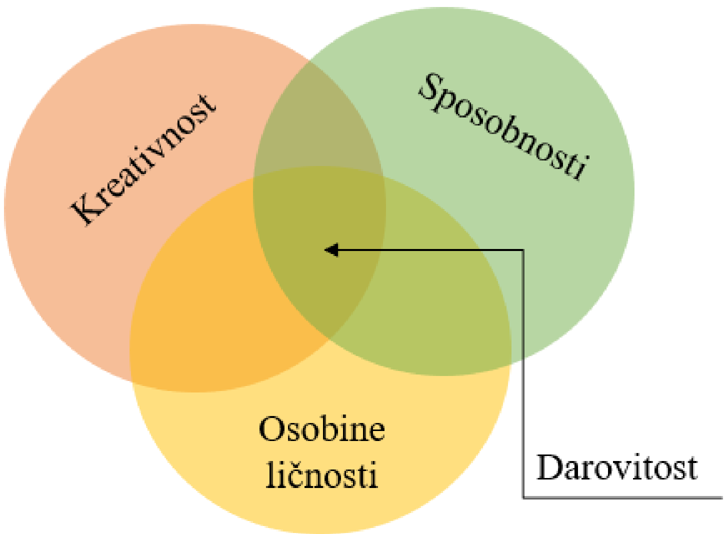
Osim izravnog poučavanja, postoje i druge razne metode kao što su: *Drill and practice*, metoda indirektnog poučavanja, metoda eksperimentalnog poučavanja, metoda poučavanja putem analogija, metoda simulacije, metoda poučavanja pomoću računala…

Prilikom odabira metode poučavanja važno je uzeti u obzir na koje područje se odabrana metoda primjenjuje. Za poučavanje gradiva programiranja širok je spektar metoda kojima se nastava i sam sadržaj može prenijeti na novu, zanimljiviju razinu. Jedan od primjera takvih metoda bila bi metoda poučavanja pomoću kreativnih igara. Temelj kreativnosti je kreativno mišljenje, a objašnjava se kao nov način razmišljanja koji navodi na otkrivanje, eksperimentiranje, zamišljanje i istraživanje. (Sternberg, 2006.) Karakteristike su: radoznalost, težnja za uspjehom, želja za različitošću i unutarnje zadovoljstvo pojedinca.

Zašto je baš odabrana metoda poučavanja pomoću kreativnih igara u ovom radu? Može se vidjeti na slijedećoj slici odnos, tj. ulogu kreativnosti u darovitosti.

„Darovitost je sklop osobina koje omogućuju pojedincu da dosljedno postiže izrazito iznad prosječan uradak u jednoj ili više aktivnosti kojima se bavi." (Bulaja A. , 2016.)

Darovita djeca zahtijevaju minimum strukturirane podrške, sposobna su za samostalna otkrića i pronalaženje novih putova razumijevanja te imaju veliku potrebu za ovladavanjem nekim područjem, po čemu se razlikuju od djece koja su samo jako marljiva i trude se.



Slika 2.4 Definicija darovitosti

Motivacija je najčešće prisutna kod djece jer djeca uvijek nešto istražuju tragajući za nepoznatim i zagonetnim. Tradicionalna nastava izostavlja izgrađivanje kreativnih stavova učenika. Igra čija je svrha poučiti, povećava motivaciju i traži aktivno sudjelovanje, čini učenje zanimljivijim i povećava pažnju. Igru treba prilagoditi dobi i mogućnostima učenika. Kada se koristi igra u nastavi potrebno je znati njenu svrhu. (Mladenović, 2019.) Svaki učenik je aktivni sudionik kojem se potiče natjecateljki duh.

3. Učenje osnovnih koncepata programiranja

Provedeno je istraživanje na uzorku od 58 studenata prve godine koji su odslušali kolegij Programiranje 1 na Prirodoslovno – matematičkog fakulteta u Splitu (studijske grupe: Informatika, Informatika i tehnika, Matematika i informatika; smjer: nastavnički, Matematika; smjer: matematički, računarski, primijenjena matematika). Cilj je bio dobiti uvid u razlike učenja osnovnih koncepata programiranja u odnosu na prethodno učenje kroz osnovnu i/ili srednju školu.

Anketom su ispitani stavovi studenata o početnom učenju programiranja, mogućim zaostacima obzirom na kolege koje su prethodno učili programiranje te samo zalaganje studenata na kolegiju Programiranje 1. Svrha istraživanja bila je prikupljenim informacijama ponuditi moguća rješenja za uspješnije shvaćanje osnovnih koncepata studentima koji se nisu tijekom ranijeg školovanja susretali s programiranjem.

Obzirom na moguće poteškoće prilikom početnog učenja programiranja, ispitani su stavovi studenata o osnovnim konceptima, načinima poučavanja i konkretno problemima sa kojima se susreću kao programeri početnici. Jesu li uopće imali predmet Informatiku u prethodnom školovanju, da li su se susretali s programiranjem u sklopu predmeta i da li se itko od ispitanih studenata bavim programiranjem u slobodno vrijeme.

U navedenoj anketi poseban naglasak je bio na studente koji se nisu susretali s programiranjem kroz osnovnu i/ili srednju školu te smatraju li da su u zaostatku naspram ostalih kolega u učenju gradiva na kolegiju Programiranje 1. Kolegij Programiranje 1 je odabran prvenstveno jer je prvi programerski predmet na studiju.

Također, ispitani su i subjektivni dojmovi ispitanika o vlastitom zalaganju i učenju na spomenutom kolegiju te koliko često su tražili konzultacije kod profesora. Nadalje, ispitano je koliku ulogu u jasnijem učenju i rješavanju problema imaju vizualni programski jezici, pisanje pseudokoda prije samog kodiranja i kojim se programskim jezicima rado služe ukoliko programiraju u slobodno vrijeme. Na samom kraju ankete dan je slobodan prostor za komentar u kojem su ispitanici izrazili vlastito mišljenje o učenju osnovnih koncepata programiranja, prikladnoj dobi za početak učenja te ideje kojima bi se moglo olakšati budućim studentima koji nisu prethodno učili programiranje.

3.1. Istraživanje

Postavljeni cilj istraživanja je dobiti uvid u razlike učenja osnovnih koncepata programiranja u odnosu na prethodno učenje kroz osnovnu i/ili srednju školu i u skladu s tim postavljene su pretpostavke:

1. Studenti se uglavnom nisu susretali s programiranjem u sklopu predmeta Informatika kroz osnovnu i/ili srednju školu.
2. Postoje razlike u stavovima studenata s obzirom na prikladnu dob učenja programiranja.

Istraživanje je provedeno putem online ankete na Prirodoslovno-matematičkom fakultetu u Splitu u akademskoj godini 2020./2021. Sudjelovalo je 58 studenata prve godine koji su odslušali kolegij Programiranje 1 na studijskim grupama: Informatika, Informatika i tehnika, Matematika i informatika; smjer: nastavnički, Matematika; smjer: matematički, računarski, primijenjena matematika. Uzorak populacije istraživanja je ciljan i neslučajan, upitnik je proveden u potpunosti anoniman te za ispunjavanje bilo je potrebno izdvojiti 5 – 10 minuta.

Tablica 3.1 Utvrđivanje cilja istraživanja

Problem istraživanja	Razlike u učenju osnovnih koncepata programiranja u odnosu na prethodno učenje kroz osnovnu i/ili srednju školu.
Cilj istraživanja	Prikupljenim informacijama ponuditi moguća rješenja za uspješnije shvaćanje osnovnih koncepata studentima koji se nisu tijekom ranijeg školovanja susretali s programiranjem.
Uzorak ispitanika	Studenti prve godine koji su odslušali kolegij Programiranje 1 na PMFST.
Hipoteze istraživanja	Studenti se uglavnom nisu susretali s programiranjem u sklopu predmeta Informatike kroz osnovnu i/ili srednju školu. Postoje razlike u stavovima studenata s obzirom na prikladnu dob učenja programiranja.
Instrument istraživanja	Za prikupljanje odgovora kreirana je online anketa napravljena pomoću Google obrasca.

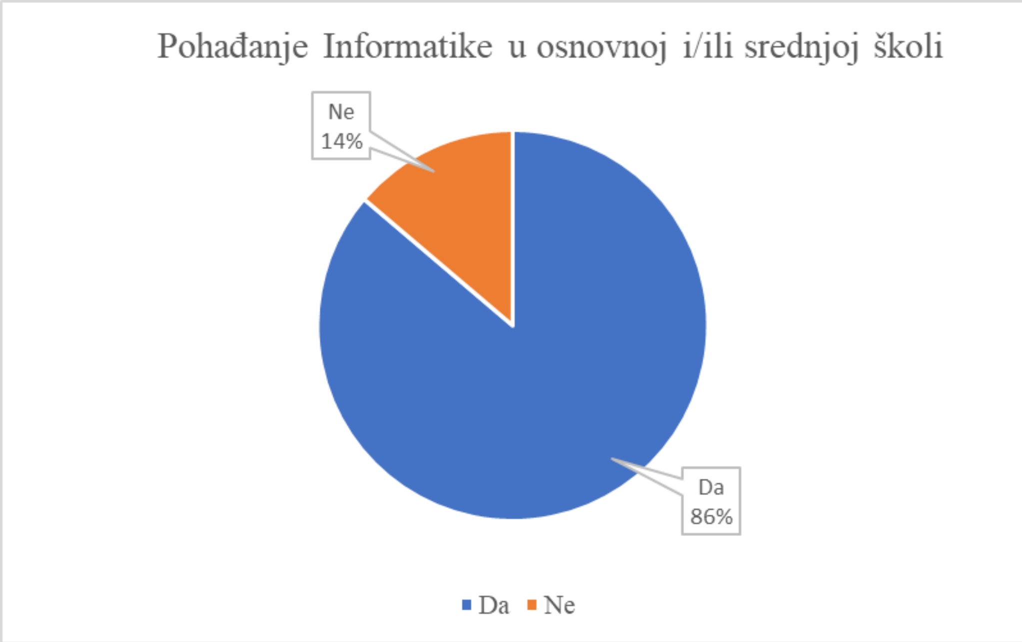
Upitnik je započeo pitanjima o ranijem iskustvu s predmetom Informatika tijekom školovanja kroz osnovnu i/ili srednju školu kako bi se izvršila generalna podjela ispitanika.

Tablica 3.2 Popis pitanja o Informatici tijekom ranijeg školovanja

R.br.	Pitanje	Tip pitanja	Obvezno
1.	Jeste li u osnovnoj ili srednjoj školi slušali predmet Informatika? a) Da b) Ne	Pitanje višestrukog izbora – moguć jedan odgovor	*
2.	Ako da, jeste li se u sklopu predmeta Informatika susretali s programiranjem? a) Da b) Ne	Pitanje višestrukog izbora – moguć jedan odgovor	*
3.	Ako da, navedite s kojim ste se programskim jezicima susreli (moguće je odabrati više odgovora). a) Logo b) Python c) Scratch d) Pseudokod e) Ostalo	Pitanje višestrukog izbora – moguće više odgovora	*

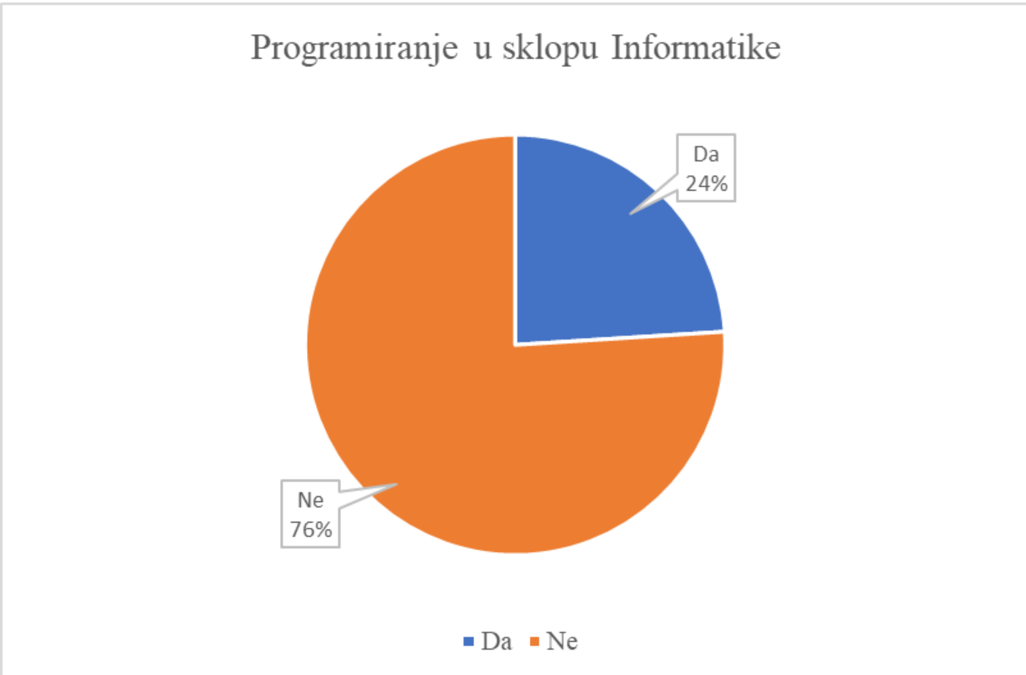
Ispitanici su podijeljeni u dvije grupe:

- **Grupa koja nije imala doticaj s programiranjem ni kroz školu ni u slobodno vrijeme** (*oznaka* ***1***) – nisu imali informatiku u školi, imali su informatiku u školi, ali bez programiranja/ne bave se programiranjem u slobodno vrijeme
- **Grupa koja je imala doticaj s programiranjem kroz školu ili slobodno vrijeme** (*oznaka* ***2***) – nisu imali informatiku u školi, ali se bave u slobodno vrijeme programiranjem, imali su informatiku i programiranje u školi (ili sve tri kategorije)

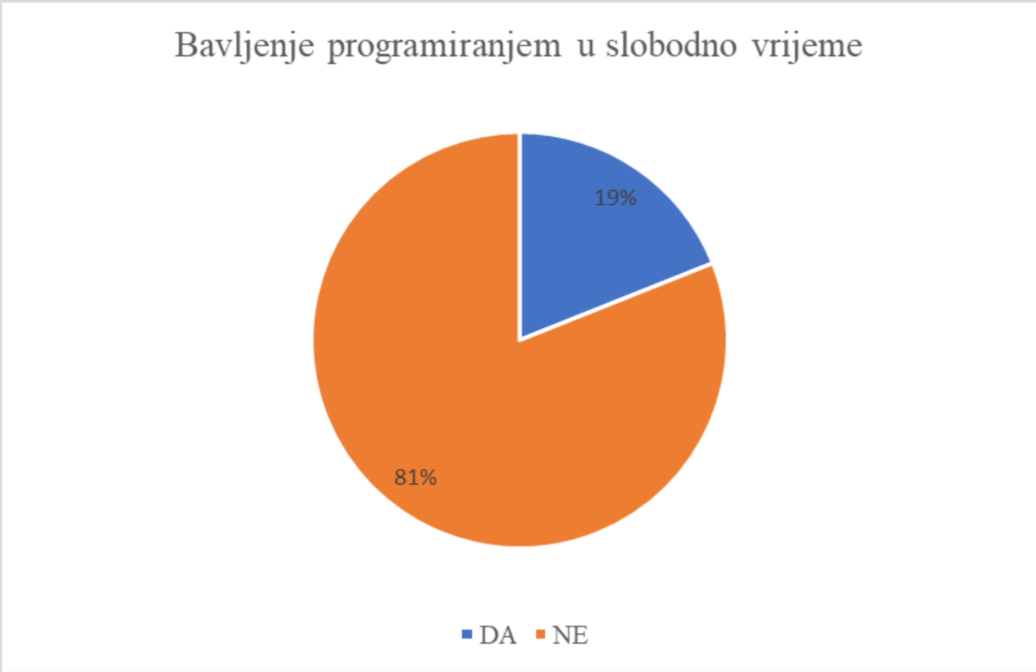


Slika 3.1 Pohađanje Informatike

Kao što je i očekivano, visok broj ispitanika je imao predmet Informatiku u sklopu osnovne ili srednje škole, a jako malen broj ispitanika nije imao taj predmet u školi. S druge strane, malen broj ispitanika susreo se s programiranjem u sklopu tog predmeta.



Slika 3.2 Programiranje u Informatici



Slika 3.3 Programiranje u slobodno vrijeme

Velik broj ispitanika ne bavi se programiranjem u slobodno vrijeme (81%), dok njih 19% programira u slobodno vrijeme.

3.2. Usporedba prve i druge grupe ispitanika

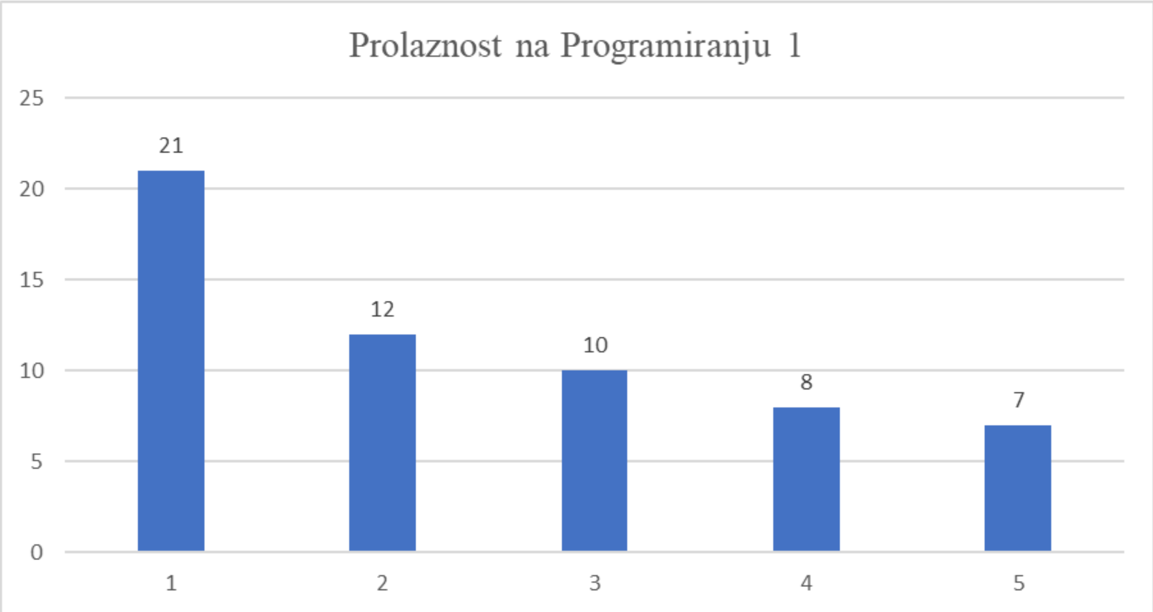
Tablica 3.3 Popis pitanja vezanih uz kolegij Programiranje 1

R.br.	Pitanje	Tip pitanja	Obvezno
4.	Koja je bila Vaša konačna ocjena iz kolegija Programiranje 1? a) Nepoloženo (1) b) Dovoljan (2) c) Dobar (3) d) Vrlo dobar (4) e) Odličan (5)	Skala od 1 do 5, ocjene od nepoloženo (1) do odličan (5) – moguć jedan odgovor	*
5.	Koliko redovito ste učili/ponavljali naučeno gradivo?	Likertova skala od 5	*

	a) Nikada … e) Iznimno redovito	stupnjeva	
6.	Koliko su Vam bili jasni koncepti naučeni tijekom predavanja? a) Nimalo jasni … e) Potpuno jasni	Likertova skala od 5 stupnjeva	*
7.	Koliko su Vam bili jasni koncepti naučeni tijekom vježbi? a) Nimalo jasni … e) Potpuno jasni	Likertova skala od 5 stupnjeva	*
8.	Koliko često ste tražili konzultacije kod profesora tijekom semestra? a) Nikada b) Tjedno c) Mjesečno d) Ostalo	Pitanje višestrukog izbora – moguć jedan odgovor	*
9.	Jesu li Vam naučeni koncepti postali razumljiviji nakon konzultacija? a) Ostali su mi nejasni … e) Sve sam razumio/la	Likertova skala od 5 stupnjeva	

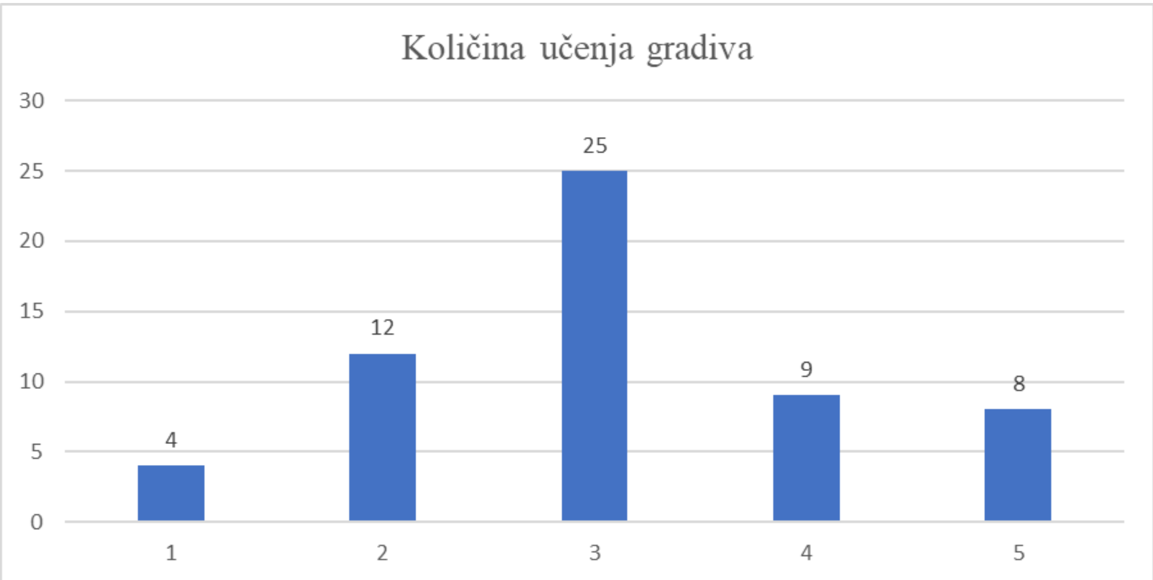
Za usporedbu prve i druge grupe gledale su se krajnje ocjene iz kolegija Programiranje 1, količina učenja gradiva za Programiranje 1, razumijevanje koncepata predstavljenih na predavanjima i vježbanja te učestalost traženja konzultacija za ovaj kolegij.

Na pitanje *„Koja je bila vaša konačna ocjena iz kolegija Programiranje 1?“*, najveći broj ispitanika odgovorio je kako još nisu položili kolegij, dok je najmanji broj ispitanika položio ovaj kolegij s ocjenom *„Odličan“*.



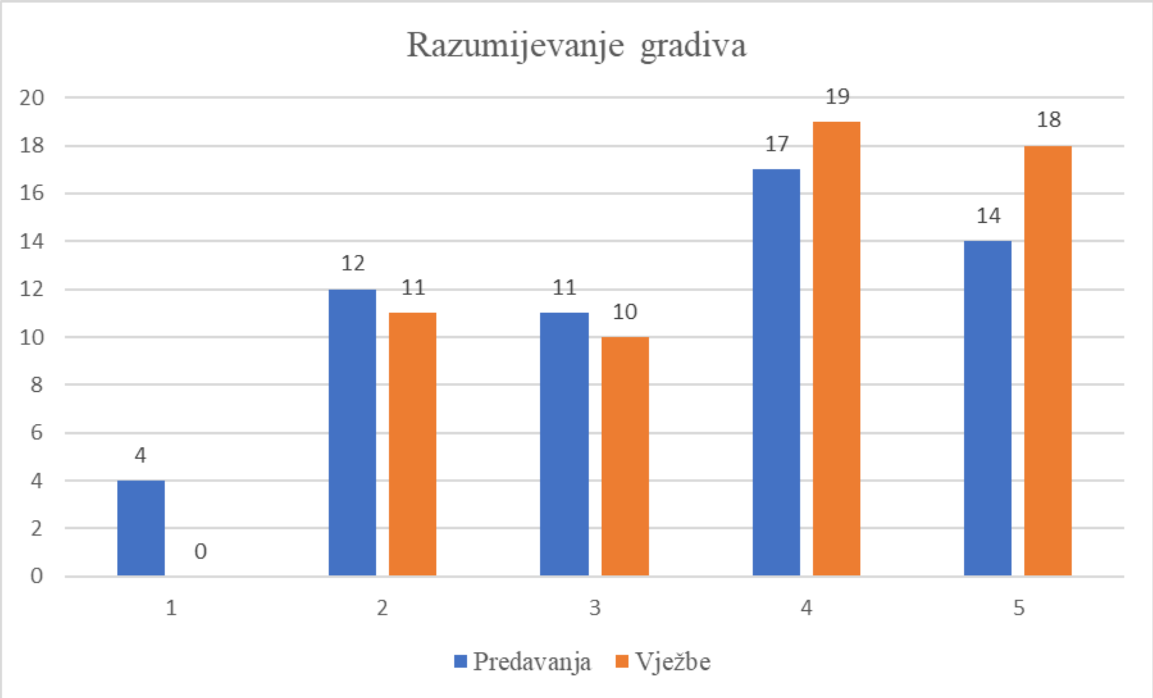
Slika 3.4 Konačna ocjena na kolegiju po svim ispitanicima

Na pitanje *„Koliko ste redovno učili/ponavljali naučeno gradivo?“*, većina studenata je na ljestvici od *„Nikada (1)“* prema *„Iznimno redovito (5)“* dala sebi ocjenu 3.



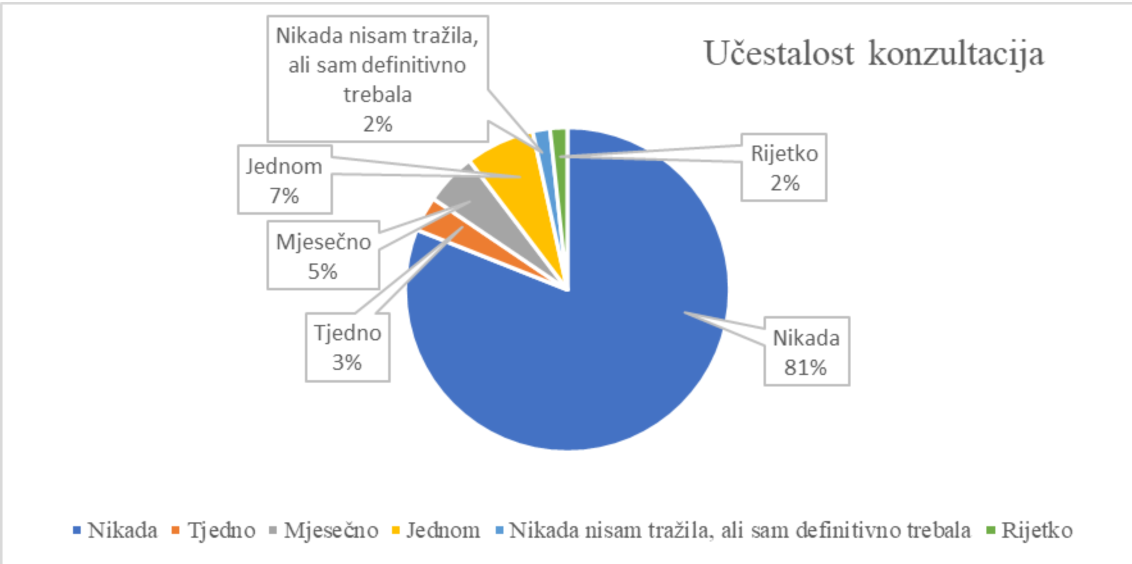
Slika 3.5 Količina uloženog vremena za učenje za sve ispitanike

Na pitanja *„Koliko su Vam bili jasni koncepti naučeni tijekom predavanja?“* i *„Koliko su Vam bili jasni koncepti naučeni tijekom vježbi?“*, na skali od *„Potpuno nejasni“* do *„Potpuno jasni“* ispitanici su rekli da su nešto bolje razumjeli koncepte koji su izloženi na vježbama nego koncepte izložene na predavanjima.

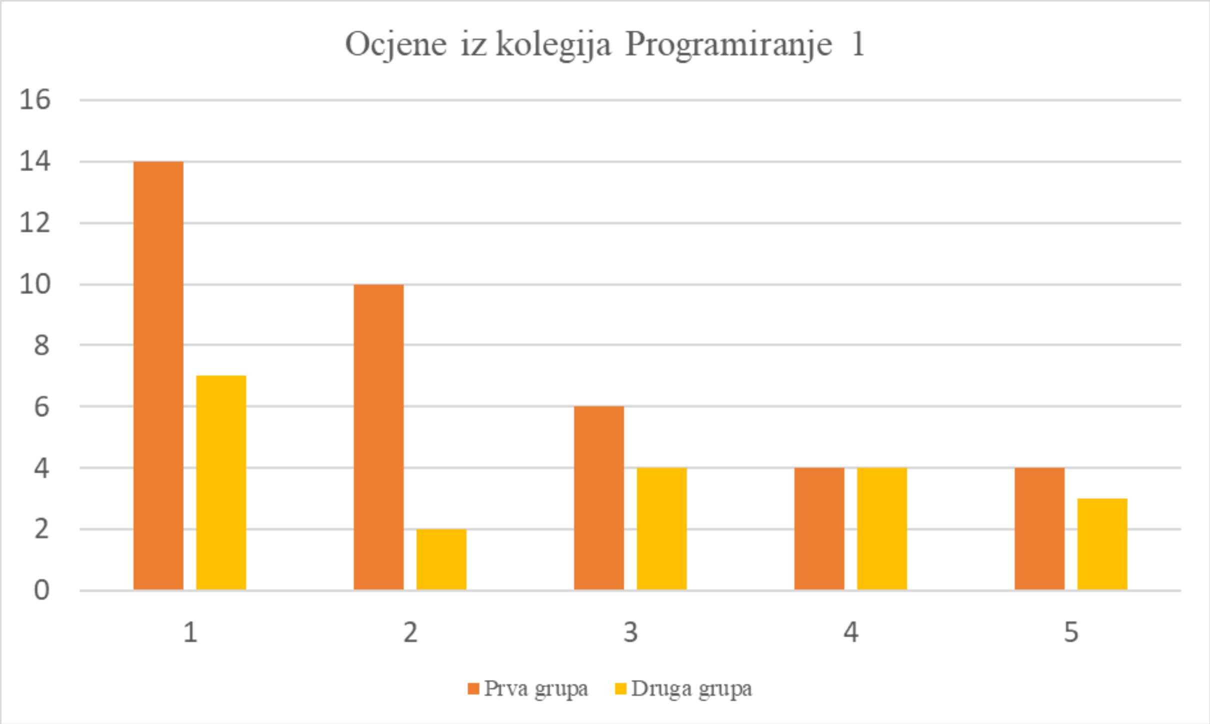


Slika 3.6 Razumijevanje gradiva kod svih ispitanika

Na pitanje *„Koliko ste često tražili konzultacije kod profesora tijekom semestra?“*, većina ispitanika odgovorila je da nikad nije tražila konzultacije, a oni koji su zatražili konzultacije, to su uglavnom zatražili jedanput ili na mjesečnoj bazi.

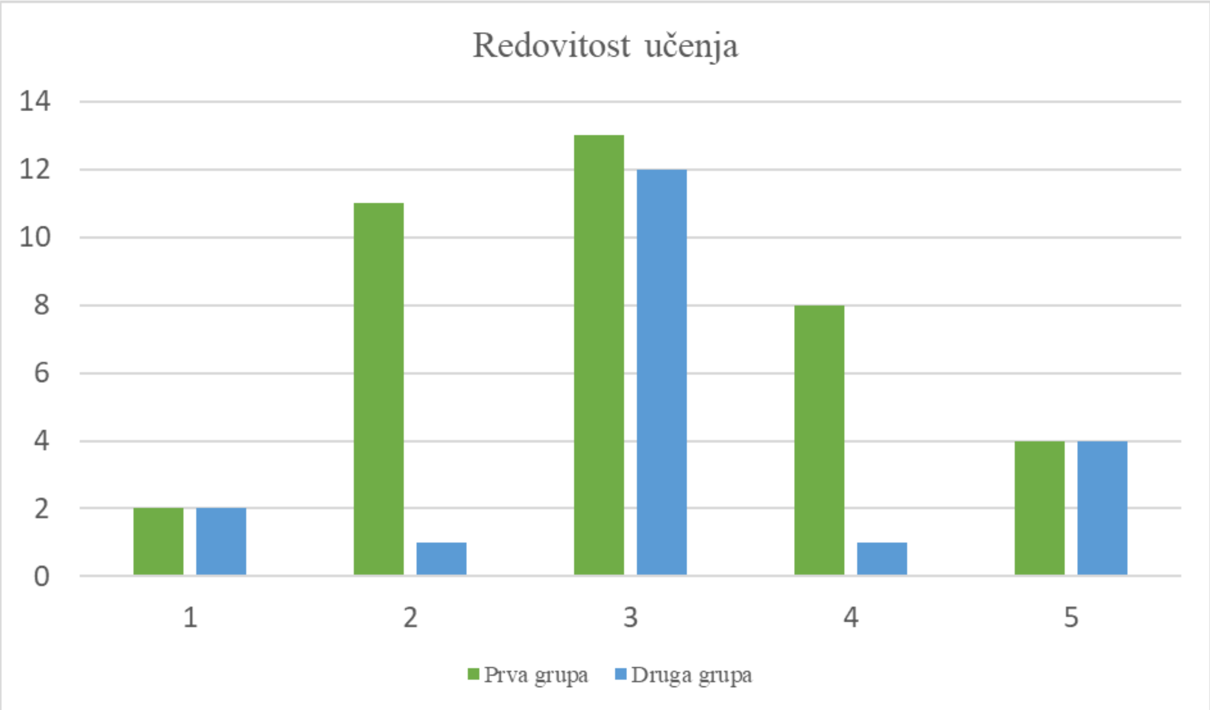


Slika 3.7 Učestalost traženja konzultacija kod svih ispitanika



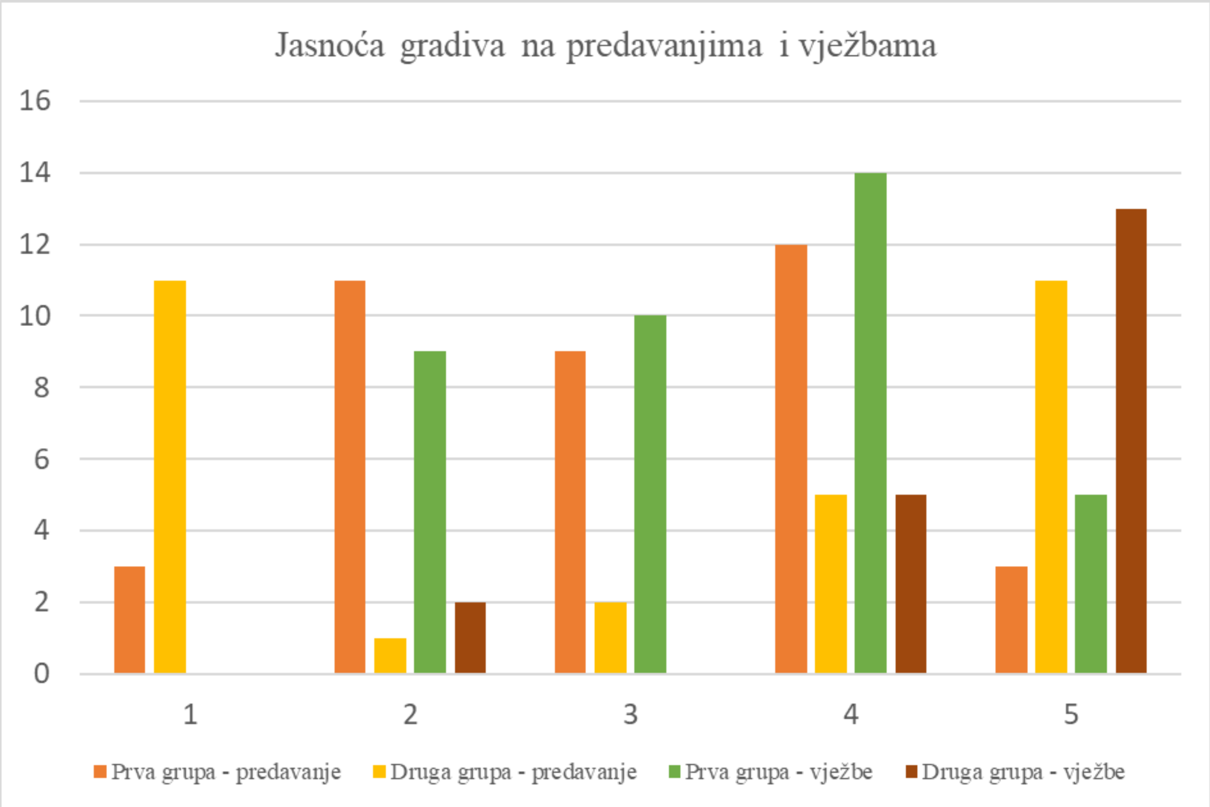
Slika 3.8 Ocjene iz kolegija Programiranje 1 za prvu i drugu grupu

I za jednu, i za drugu grupu visok je postotak ispitanika koji nisu položili kolegij Programiranje 1, dok za ostale ocjene, grupa koja je imala doticaja s programiranjem postigla je nešto bolje ocjene od grupe koja nije imala doticaja s programiranjem.



Slika 3.9 Redovitost učenja za prvu i drugu grupu

Kod redovitosti učenja prema grupama, prva grupa je dala nešto raznolikije odgovore, stoga je bilo dosta ispitanika koji su učili malo rjeđe, a neki su dali više truda, odnosno vremena. Kod druge grupe, ispitanici su uglavnom ulagali srednju razinu vremena za učenje.

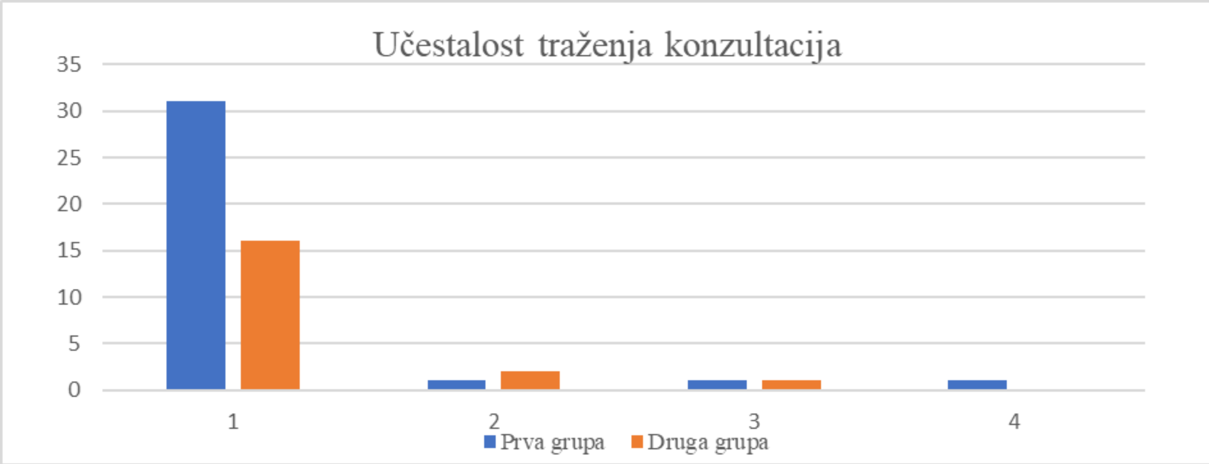


Slika 3.10 Jasnoća gradiva predavanja i vježbi za prvu i drugu grupu

Jasno se vide razlike u razumijevanju koncepata sa predavanja za dvije grupe – grupa koja nije imala doticaja s programiranjem malo je lošije razumijevala koncepte poučavane u sklopu predavanja za kolegij Programiranje 1, dok je grupa koja je imala nekakav doticaj s programiranjem uglavnom odgovarala da su im koncepti poučavani na predavanjima bili iznimno jasni.

I ovdje se mogu vidjeti određene razlike u razumijevanju koncepata naučenih na vježbama između dvije grupe. Za prvu grupu većina ispitanika je svoje razumijevanje koncepata s vježbi ocijenila sa 4, odnosno koncepti su im bili uglavnom jasni, dok je ispitanicima gradivo s vježbi uglavnom bilo posve jasno.

Nitko za koncepte s vježbi nije označio da mu je gradivo naučeno na vježbama posve nejasno, stoga se može reći da obje grupe bolje razumijevaju gradivo izloženo na vježbama nego na predavanjima, što je i očekivano.

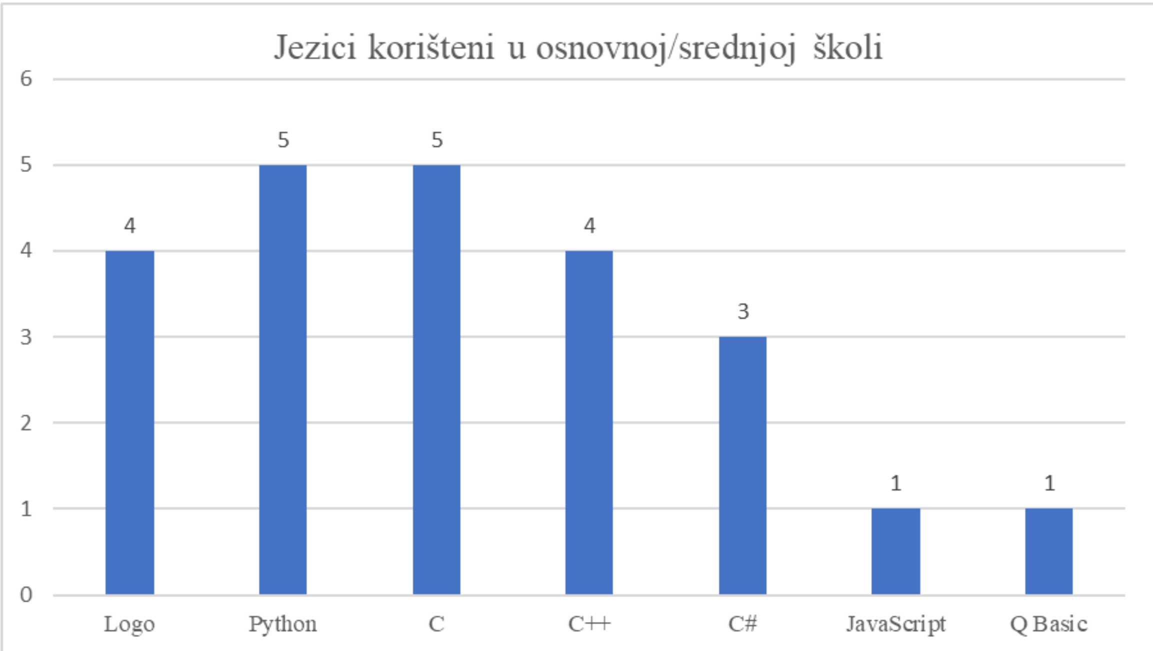


Slika 3.11 Učestalost traženja konzultacija za prvu i drugu grupu

Obje grupe ispitanika većinom nikad nisu tražile konzultacije, a i oni ispitanici koji su zatražili konzultacije to su činili jako rijetko.

3.3. Pregled programskih jezika s kojima su se ispitanici susretali

Studenti koji su se u osnovnoj/srednjoj školi susretali s programiranjem, najčešće su navodili Python, jezike iz C obitelji jezika (C, C++, C#), Logo i pisanje pseudokoda kao jezike s kojima su se susreli.

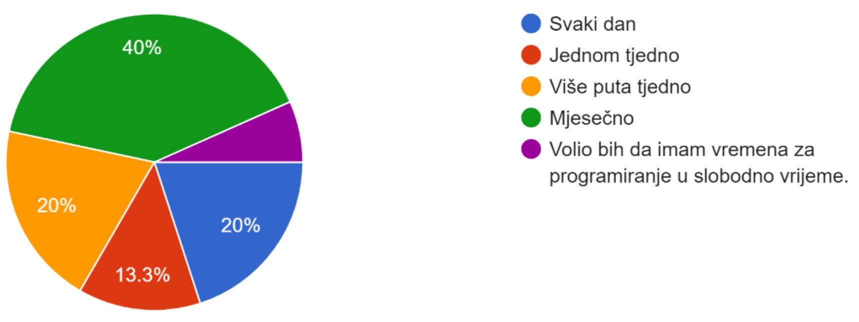


Slika 3.12 Najčešće korišteni jezici

Tablica 3.4 Popis pitanja koja se odnose na programiranje u slobodno vrijeme

R.br.	Pitanje	Tip pitanja	Obvezno
10.	Bavite li se programiranjem u slobodno vrijeme? a) Da b) Ne	Pitanje višestrukog izbora – moguć jedan odgovor	*
11.	Ako da, koliko često programirate u slobodno vrijeme? a) Svaki dan b) Jednom tjedno c) Više puta tjedno d) Mjesečno e) Ostalo	Pitanje višestrukog izbora – moguć jedan odgovor	
12.	Ukoliko se bavite programiranjem u slobodno vrijeme, koje programske jezike koristite?	Tekst kratkog odgovora	

Studenti koji se bave programiranjem u slobodno vrijeme, naveli su za učestalost programiranja u slobodno vrijeme da najčešće programiraju na mjesečnoj bazi (40%), a podjednak broj je naveo da programiraju svaki dan ili barem više puta tjedno (20%). Jedan od studenata naveo je kako bi volio imati više vremena za programiranje u slobodno vrijeme.



Slika 3.13 Učestalost programiranja u slobodno vrijeme

Najčešće korišteni jezici kod ove grupe ispitanika uključivali su C#, a potom JavaScript i Python.

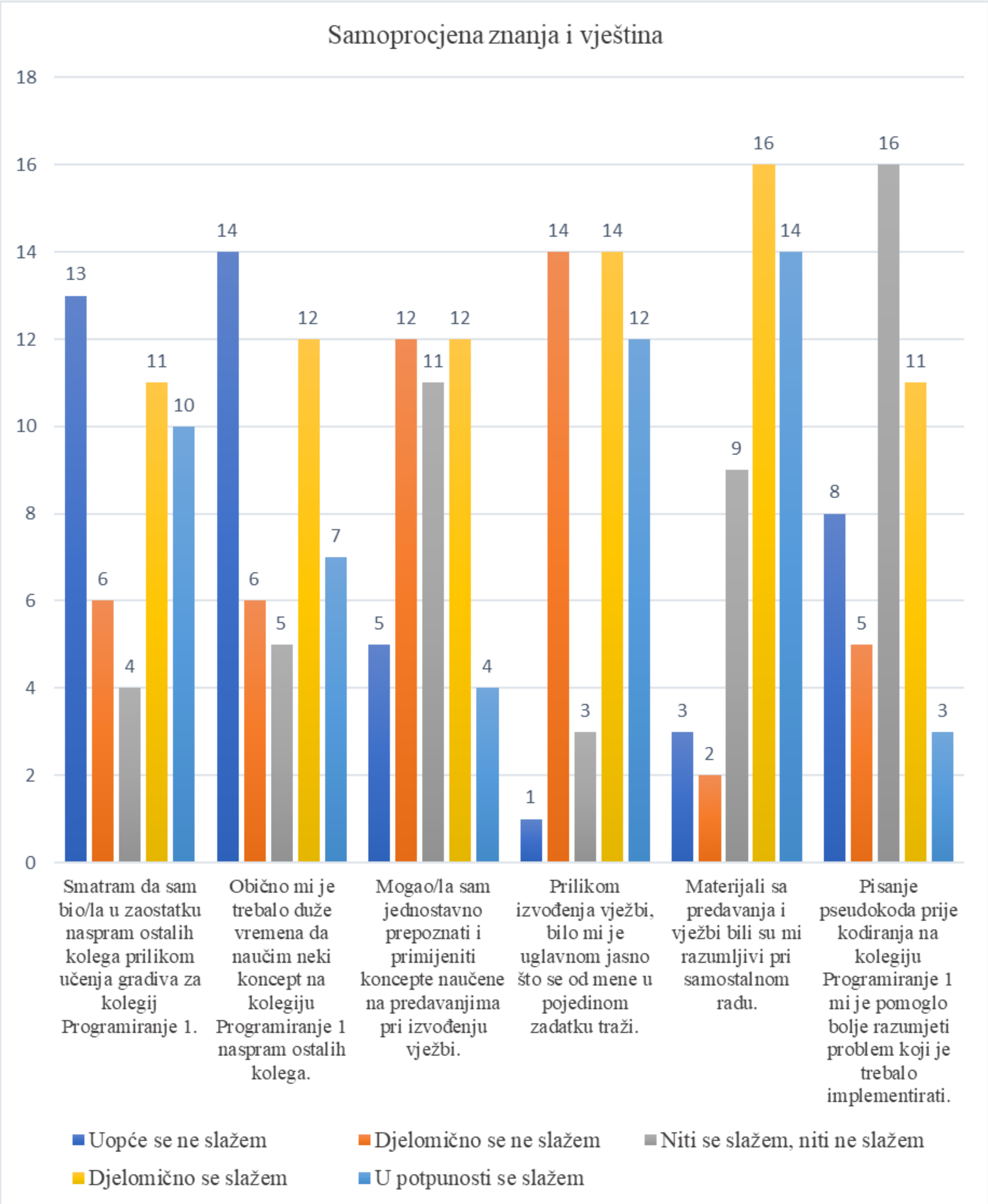
Slika 3.14 Najčešće korišteni jezici kod programiranja u slobodno vrijeme

3.4. Samoprocjena vještina i znanja primijenjenih i stečenih na kolegiju Programiranje 1

Tablica 3.5 Popis pitanja posvećenih studentima koji su prethodno programirali

R.br.	Pitanje (tvrdnja)	Tip pitanja
13.	Mogao/la sam jednostavno prepoznati naučene koncepte iz programiranja u osnovnoj/srednjoj školi.	Likertova skala od 6 stupnjeva, mogući odgovori: a) Ne može se primijeniti b) Uopće se ne slažem c) Djelomično se ne slažem d) Niti se slažem, niti ne slažem e) Djelomično se slažem
14.	Pisanje pseudokoda prije kodiranja mi je pomoglo bolje razumjeti problem koji je trebalo implementirati.	
15.	Korištenje vizualnih programskih jezika (Scratch) mi je pomoglo pri razumijevanju koncepata naučenih na kolegiju Programiranje 1.	

16.	Korištenje drugih programskih jezika pomoglo mi je pri razumijevanju programskog jezika Python korištenog na kolegiju Programiranje 1.	f) U potpunosti se slažem



Slika 3.15 Samoprocjena znanja i vještina

Ovaj dio ankete popunjavali su studenti koji se nisu susreli s programiranjem kroz školu ili kroz slobodno vrijeme. Ova sekcija namijenjena je procjeni vlastitih vještina i znanja koje su studenti primjenjivali u razumijevanju koncepata naučenih na kolegiju Programiranje 1 te usporedbi s kolegama koji su imali nekakvo predznanje prije upisa studija.

Na tvrdnju *„Smatram da sam bio/la u zaostatku naspram ostalih kolega prilikom učenja gradiva za kolegij Programiranje 1“*, najveći broj ispitanika odgovorio je kako se uopće ne slaže s tom tvrdnjom, ali i velik broj ispitanika odgovorio je kako se djelomično ili u potpunosti slaže.

Na tvrdnju *„Obično mi je trebalo duže vremena da naučim neki koncept na kolegiju Programiranje 1 naspram ostalih kolega“*, ponovno je najveći broj ispitanika odgovorio da se uopće ne slaže, a dosta visok broj ispitanika odgovorio kako se u potpunosti ili djelomično slaže.

Na tvrdnju *„Mogao/la sam jednostavno prepoznati i primijeniti koncepte naučene na predavanjima pri izvođenju vježbi“*, najveći broj ispitanika se djelomice slagao i djelomice nije slagao s tom tvrdnjom, dok visok broj njih se nije mogao ni složiti, ni ne složiti s tom tvrdnjom.

Na tvrdnju *„Prilikom izvođenja vježbi, bilo mi je uglavnom jasno što se od mene u pojedinom zadatku traži“*, najveći broj ispitanika se i djelomice slagao i djelomice nije slagao s tom tvrdnjom, a visok broj ispitanika se u potpunosti slagao, pa se može zaključiti da su zadaci bili donekle razumljivi za rješavanje.

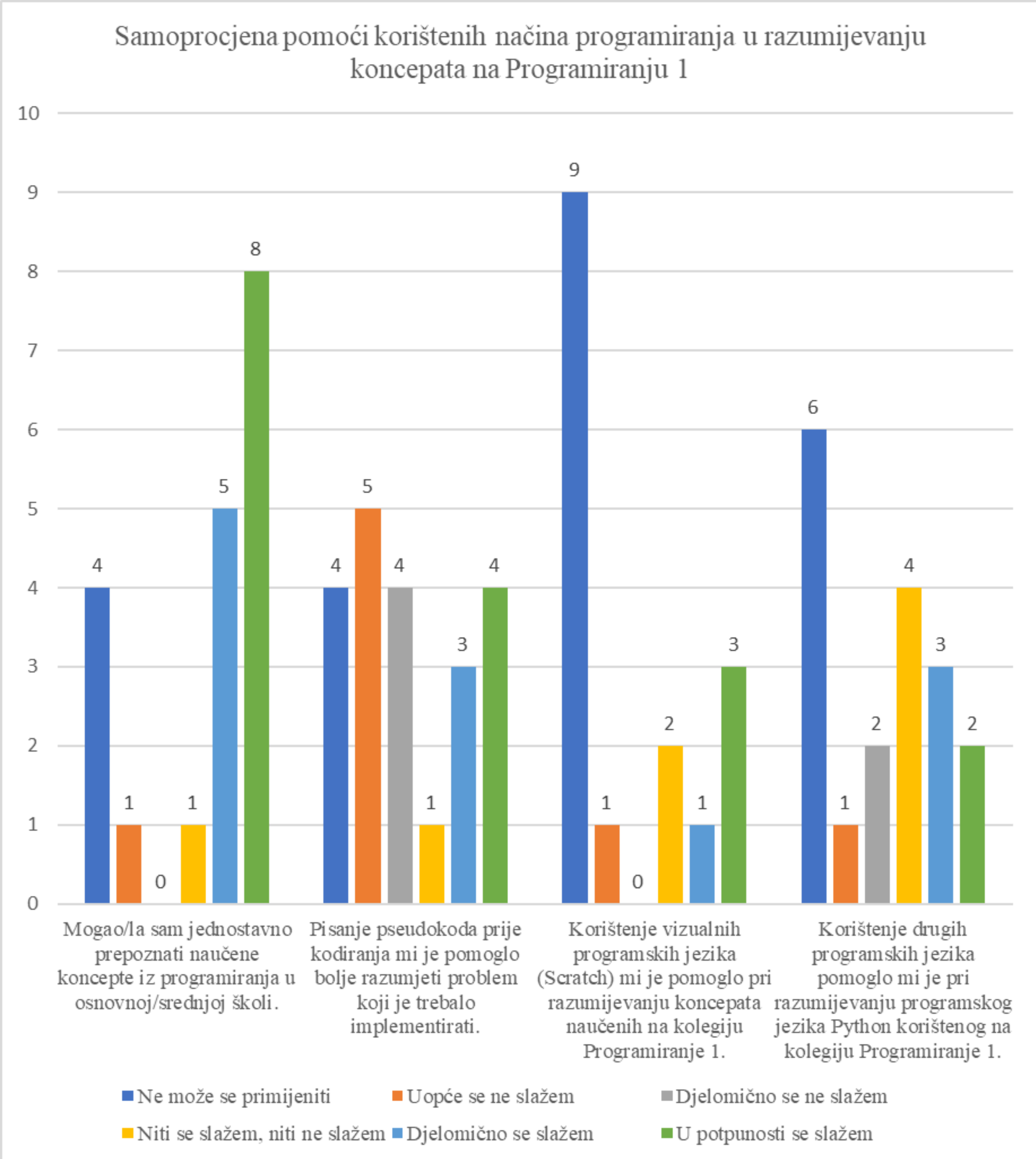
Na tvrdnju *„Materijali sa predavanja i vježbi bili su mi razumljivi pri samostalnom radu“*, najveći broj ispitanika se djelomice složio s tom tvrdnjom, a visok i u potpunosti složio, pa se može zaključiti da su materijali bili primjereni za samostalan rad onima koji se tek susreću s programiranjem.

Na tvrdnju *„Pisanje pseudokoda prije kodiranja na kolegiju Programiranje 1 mi je pomoglo bolje razumjeti problem koji je trebalo implementirati“*, najveći broj ispitanika se nije mogao ni složiti, ni ne složiti s tom tvrdnjom, pa se postavlja pitanje koliko je zapravo korisno pisanje pseudokoda prije kodiranja, te kako bi se taj dio poučavanja trebao oblikovati da bude korisno.

3.5. Samoprocjena pomoći korištenih načina programiranja kroz školu ili u slobodno vrijeme

Tablica 3.6 Popis pitanja posvećenih studentima koji nisu prethodno programirali

<table>
<tr><th>R.br.</th><th>Pitanje (tvrdnja)</th><th>Tip pitanja</th></tr>
<tr><td>17.</td><td>Smatram da sam bio/la u zaostatku naspram ostalih kolega prilikom učenja gradiva za kolegij Programiranje 1.</td><td rowspan="6">Likertova skala od 5 stupnjeva, mogući odgovori:
a) Uopće se ne slažem
b) Djelomično se ne slažem
c) Niti se slažem, niti ne slažem
d) Djelomično se slažem
e) U potpunosti se slažem</td></tr>
<tr><td>18.</td><td>Obično mi je trebalo duže vremena da naučim neki koncept na kolegiju Programiranje 1 naspram ostalih kolega.</td></tr>
<tr><td>19.</td><td>Mogao/la sam jednostavno prepoznati i primijeniti koncepte naučene na predavanjima pri izvođenju vježbi.</td></tr>
<tr><td>20.</td><td>Prilikom izvođenja vježbi, bilo mi je uglavnom jasno što se od mene u pojedinom zadatku traži.</td></tr>
<tr><td>21.</td><td>Materijali sa predavanja i vježbi bili su mi razumljivi pri samostalnom radu.</td></tr>
<tr><td>22.</td><td>Pisanje pseudokoda prije kodiranja na kolegiju Programiranje 1 mi je pomoglo bolje razumjeti problem koji je trebalo implementirati.</td></tr>
</table>



Slika 3.16 Samoprocjena pomoći korištenih načina programiranja u razumijevanju koncepata na Programiranju 1

Ovaj dio ankete bio je namijenjen studentima koji su se susretali s programiranjem kroz školu ili u slobodno vrijeme, te se nastojalo ispitati koliko su im određeni načini programiranja i programiranje općenito pomogli kod razumijevanja koncepata na kolegiju Programiranje 1.

Na tvrdnju *„Mogao/la sam jednostavno prepoznati naučene koncepte iz programiranja u osnovnoj/srednjoj školi“* najveći broj ispitanika se u potpunosti složio, što je bio i očekivan odgovor.

Na tvrdnju *„Pisanje pseudokoda prije kodiranja mi je pomoglo bolje razumjeti problem koji je trebalo interpretirati“* bilo je dosta različitih odgovora, a najviše ih se nije složilo s tom tvrdnjom. Dosta ispitanika također se nije susrelo s pisanjem pseudokoda kroz školovanje ili u slobodno vrijeme.

Na tvrdnju *„Korištenje vizualnih programskih jezika (Scratch) mi je pomoglo pri razumijevanju koncepata naučenih na kolegiju Programiranje 1“*, dosta ispitanika se nije ni susrelo s ovim načinom programiranja, a oni koji jesu najčešće su se u potpunosti složili, ili nisu mogli procijeniti slažu li se ili ne.

Na tvrdnju *„Korištenje drugih programskih jezika mi je pomoglo pri razumijevanju programskog jezika Python korištenog na kolegiju Programiranje 1“*, dosta ispitanika nije moglo odrediti slažu li se ili ne slažu s tom tvrdnjom.

3.6. Komentari ispitanika

Tablica 3.7 Prostor za komentar

R.br.	Pitanje	Tip pitanja	Obvezno
23.	Za kraj, napišite Vaše mišljenje o učenju osnovnih koncepata programiranja (koja je dob prikladna za učenje, na koji način bi se moglo olakšati studentima koji nisu prethodno učili programiranje...).	Tekst dugog odgovora	

„Smatram da je prikladna dob za učenje od srednje škole, a za osnovnu školu samo vizualne programske jezike.“

„Svakako ne u osnovnoj školi, srednja možda 4. srednje“

„Prikladna dob bi po meni bila 15+ godina, znači srednja škola (smatram da bi se u srednjoj školi trebao odraditi nekakav minimum, bar osnove kroz teoriju). Kako bi se olakšalo studentima - mislim da bi bilo jako korisno ako bi se dalo više vremena i truda u

uvodne satove, da student prvo nauči što je uopće taj program u kojem će raditi i kojim jezikom, u koje će ga svrhe najčešće koristiti (meni osobno uvijek pomogne nekakav plan kolegija, da znam što me čeka i kako će to sve izgledati) itd.“

„iz osobnog iskustva nije nužno predznanje kako bi se savladalo gradivo“

„Svaka čast profesoru, ali pojedinci (i ja sama) imaju prevelike rupe da bi se popunile u tako kratkom vremenu. Sada mogu reći da “znam” programirati odnosno razumijem sve što smo radili. Programiranje 1 mi je bio jedan od najtežih kolegija u semestru i zahtijevao je od mene puno truda i vremena za razliku od mojih kolega kojima je to bila zabava jer praktički većinu toga znaju iz srednje.“

„1. Podučavati osnovne koncepte koristeći jezik sa "laganom" sintaksom (primjerice Python). 2. Ne davati rješenje na gotovo, već navoditi do istog. 3. Predavanja ne služe ničemu, vježbe su zakon!“

„Smatram da bi trebalo uvesti programiranje u predmetu informatika koji se uči u srednjoj školi jer bi imali predznanja te bi mnogo naučili u te četiri godine. Na fakultetu bi bolje shvaćali što se od nas traži i kako riješiti problem. Neki od nas se nisu susretali s programiranjem stoga imamo neke zaostatke u odnosu na druge koji su već učili o tome.“

„Mislim da je prikladna dob za učenje osnovnih koncepata programiranja već u startu srednje škole. Studentima koji nisu prethodno učili programiranje bi trebalo zakazati vježbe i konzultacije 3-4 puta tjedno u svrhu lakšeg razumijevanja gradiva.“

„15-16 Uživo raditi ne samo na običnim zadatcima već u grupama na projektu tj. nekom štogod kompliciranijem programu koji bi također se stavljao na web stranicu sto bi omogućilo studentima u potpunosti iskoristiti i sto su naučili za html i css te sto su naučili iz pythona i c# te proširiti to znanje kao i naučiti raditi u timu“

„Ja sve do faksa nisam nikad programirala i prošla sam Programiranje 1.Mislim da nema potrebe za nekim dodatnim olakšanjima nego treba samo uložiti malo veći trud od onih koji su već programirali.“

„Smatram da se programiranje već može početi učiti pri kraj osnovne škole (7./8. Razred) jer onda mogu polako i bez stresa krenuti učiti.“

„Da se razumijemo ja se zalažem da se informatika u srednjim školama stavi kao obavezni predmet. Problem informatičko neobrazovane djece koja jedino sto znaju je biti na instagramu i snapchatu a ne znaju poslati mail dovodi nas u situaciju gdje tu djecu treba

prvo uciti poslati mail pa onda tek programiranje. Jesu li oni uopće sposobni za to? Ja sam prvenstveno bila iznimno nezadovoljna s profesorom iz P1 koji nam je držao vježbe te gradivo nisan razumjela uopće.“

3.7. Statistička analiza podataka

Korištenjem Mann – Whitney U testa, pokušalo se doći do zaključka postoje li značajne razlike kod dvije promatrane grupe na temelju ocjene, redovitosti učenja, razumijevanja gradiva na predavanjima i vježbama te učestalosti traženja konzultacija.

Iz slike je vidljivo kako statistički značajne razlike postoje samo kod jasnoće gradiva na predavanju i vježbama.

Ranks

	Nema/ima doticaj s programiranjem	N	Mean Rank	Sum of Ranks
Ocjena iz programiranja 1	1	38	28.16	1070.00
	2	20	32.05	641.00
	Total	58		
Redovitost ucenja	1	38	28.51	1083.50
	2	20	31.38	627.50
	Total	58		
Jasnoca gradiva na predavanju	1	38	23.88	907.50
	2	20	40.18	803.50
	Total	58		
Jasnoca gradiva na vjezbama	1	38	23.70	900.50
	2	20	40.53	810.50
	Total	58		
Ucestalost trazenja konzultacija	1	38	29.24	1111.00
	2	20	30.00	600.00
	Total	58		

Test Statistics[a]

	Ocjena iz programiranja 1	Redovitost ucenja	Jasnoca gradiva na predavanju	Jasnoca gradiva na vjezbama	Ucestalost trazenja konzultacija
Mann-Whitney U	329.000	342.500	166.500	159.500	370.000
Wilcoxon W	1070.000	1083.500	907.500	900.500	1111.000
Z	-.863	-.645	-3.593	-3.754	-.249
Asymp. Sig. (2-tailed)	.388	.519	.000	.000	.804

a. Grouping Variable: Nema/ima doticaj s programiranjem

Slika 3.17 Testiranje raznih parametara za dvije grupe ispitanika

3.7.1. Traženje povezanosti između ocjene iz Programiranja 1 i drugih varijabli po grupama

Što se tiče ispitivanja povezanosti određenih varijabli kod obje grupe, primijenjen je Spearmannov koeficijent korelacije, kako bi se ustvrdilo postoji li povezanost između:

- Učestalosti bavljenja programiranjem u slobodno vrijeme i konačne ocjene iz Programiranja 1
- Redovitosti učenja i konačne ocjene iz Programiranja 1
- Učestalosti traženja konzultacija i konačne ocjene iz Programiranja 1

Ovim se htjelo vidjeti postoje li nekakve bitne razlike kod povezanosti više varijabli zasebno po grupi.

Nonparametric Correlations

Correlations

	Bave se programiranjem u slobodno vrijeme			Ocjena iz programiranja 1	Ucestalost programiranja u slobodno vrijeme
Spearman's rho	0	Ocjena iz programiranja 1	Correlation Coefficient	1.000	.
			Sig. (1-tailed)	.	.
			N	47	3
		Ucestalost programiranja u slobodno vrijeme	Correlation Coefficient	.	1.000
			Sig. (1-tailed)	.	.
			N	3	3
	1	Ocjena iz programiranja 1	Correlation Coefficient	1.000	.377
			Sig. (1-tailed)	.	.126
			N	11	11
		Ucestalost programiranja u slobodno vrijeme	Correlation Coefficient	.377	1.000
			Sig. (1-tailed)	.126	.
			N	11	11

Slika 3.18 Povezanost programiranja u slobodno vrijeme i ocjene iz Programiranja 1

Promatra se samo druga grupa (prema pitanju *„Bavite li se programiranjem u slobodno vrijeme?“*). Prema koeficijentu korelacije postoji slaba povezanost dobivene ocjene na kolegiju Programiranje 1 i učestalosti bavljenja programiranjem u slobodno vrijeme. Prema rezultatima, nema značajne razlike između ove dvije varijable.

Correlations

Nema/ima doticaj s programiranjem				Ocjena iz programiranja 1	Redovitost ucenja
Spearman's rho	1	Ocjena iz programiranja 1	Correlation Coefficient	1.000	.458**
			Sig. (2-tailed)	.	.004
			N	38	38
		Redovitost ucenja	Correlation Coefficient	.458**	1.000
			Sig. (2-tailed)	.004	.
			N	38	38
	2	Ocjena iz programiranja 1	Correlation Coefficient	1.000	-.012
			Sig. (2-tailed)	.	.960
			N	20	20
		Redovitost ucenja	Correlation Coefficient	-.012	1.000
			Sig. (2-tailed)	.960	.
			N	20	20

**. Correlation is significant at the 0.01 level (2-tailed).

Slika 3.19 Povezanost količine učenja i ocjene iz Programiranja 1

Kod prve grupe postoji umjerena povezanost između ocjene iz Programiranja 1 i uloženog vremena kod učenja, dok kod druge grupe postoji vrlo slaba povezanost između te dvije varijable, pa razlika nije statistički značajna. Kod prve grupe razlika jest statistički značajna, pa zaista postoji povezanost između dvije varijable.

Correlations

Nema/ima doticaj s programiranjem				Ucestalost trazenja konzultacija	Ocjena iz programiranja 1
Spearman's rho	1	Ucestalost trazenja konzultacija	Correlation Coefficient	1.000	.238
			Sig. (2-tailed)	.	.150
			N	38	38
		Ocjena iz programiranja 1	Correlation Coefficient	.238	1.000
			Sig. (2-tailed)	.150	.
			N	38	38
	2	Ucestalost trazenja konzultacija	Correlation Coefficient	1.000	.207
			Sig. (2-tailed)	.	.381
			N	20	20
		Ocjena iz programiranja 1	Correlation Coefficient	.207	1.000
			Sig. (2-tailed)	.381	.
			N	20	20

Slika 3.20 Povezanost između traženja konzultacija i ocjene iz programiranja 1

Kod obje grupe korelacija između učestalosti traženja konzultacija i ocjene iz Programiranja 1 je slaba ili vrlo slaba, te ne postoji statistički značajna razlika ni kod jedne ni kod druge grupe.

Treba napomenuti da je broj ispitanika koji je sudjelovao u anketi relativno nizak, stoga je moguće da nema dovoljno podataka da se može izvesti siguran zaključak o povezanosti varijabli za obje grupe.

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.456[a]	.208	-.029	1.474

a. Predictors: (Constant), Ucestalost programiranja u slobodno vrijeme, Ucestalost trazenja konzultacija, Redovitost ucenja

ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	5.710	3	1.903	.876	.486[b]
	Residual	21.719	10	2.172		
	Total	27.429	13			

a. Dependent Variable: Ocjena iz programiranja 1

b. Predictors: (Constant), Ucestalost programiranja u slobodno vrijeme, Ucestalost trazenja konzultacija, Redovitost ucenja

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	.597	1.263		.473	.647
	Redovitost ucenja	.068	.348	.061	.197	.848
	Ucestalost trazenja konzultacija	.670	.681	.298	.983	.349
	Ucestalost programiranja u slobodno vrijeme	.302	.341	.260	.886	.397

a. Dependent Variable: Ocjena iz programiranja 1

Slika 3.21 Linearna regresijska analiza

Kod prve tablice, rezultati R-statistike ukazuju na slabiju korelaciju između navedenih varijabli, a R^2 ukazuje da se oko svega 20% varijacije u zavisnoj varijabli ocjena može objasniti nekom od nezavisnih varijabli. Regresijski model, prema rezultatima statističke značajnosti, ne predviđa značajno varijablu ishoda, dakle trenutni regresijski model nije dobar za predviđanje podataka.

Na kraju, u zadnjoj tablici, pod standardiziranim beta koeficijentom, može se vidjeti koliko značajno svaka od testiranih nezavisnih varijabli utječe na zavisnu varijablu ocjene.

Najmanji utjecaj ima varijabla redovitosti učenja, dok varijable učestalosti programiranja u slobodno vrijeme i učestalosti traženja konzultacija nešto više utječu na varijablu ocjena.

Međutim, ni jedna od ovih varijabli ne utječe značajno na varijablu ocjena, o čemu govori statistička značajnost za svaku od varijabli u zadnjem stupcu tablice. Može se zaključiti da neka druga varijabla bitnije utječe na varijablu ocjena.

Zaključak

Glavni cilj ovog istraživanja bio je ispitati koliko se ustvari učenici, odnosno sada već studenti prve godine susreću s programiranjem prije slušanja i polaganja programerskih kolegija na fakultetu. Rezultatima ankete pokazalo se da je doista mali postotak, tek 24% onih koji su zapravo učili programiranje u sklopu predmeta Informatike u ranijem školovanju ili im je programiranje hobi. Velika su očekivanja stavljena u trenutni kurikulum za predmet Informatiku koji je na snazi u Republici Hrvatskoj. Njime je posebno naglašeno područje računalnog razmišljanja i programiranja. Sve više se pridaje pažnja vještinama programiranja od rane dobi tako da se tijekom višegodišnjeg obrazovanja učenicima potiče kritičko mišljenje, kreativnost i inovativnost te razvijanje sposobnosti rješavanja problema. Time se očekuje povećanje postotka učenika kojima je hobi programiranje.

Zanimljive su razlike u mišljenima studenata po pitanju „olakšanja“ studentima koji nisu prethodno učili programiranje i isto tako o prikladnoj dobi za učenje. Dok jednima nije bilo potrebno prethodno predznanje za savladavanje gradiva na kolegiju Programiranje 1, drugi pak smatraju da su bili u zaostatku naspram drugih kolega koji su prethodno se susretali s osnovnim principima programiranja. Međutim, neki od savjeta budućim kolegama bili bi da krenu samostalno učiti, da ulože dodatan trud, ali isto tako neki smatraju da je potrebno uložiti više vremena i truda za uvodne satove kolegija da studenti prvo nauče što je uopće taj program u kojem će raditi i u koje će ga svrhe najčešće koristiti, ukratko bolji pregled plana kolegija.

Naravno, neće svaki učenik biti programer niti je računalna znanost interes svih, nikako nije cilj stvoriti homogeno društvo, ali je cilj ostvariti pojedinca spremnog za suvremeni život i rad. Svakako, učenje programskih jezika i usvajanje osnovnih koncepata programiranja smatra se potrebom kako bi učenici bili kompetentni jer današnjica bez tehnologije je nezamisliva, a njen razvoj dovodi do svakojakih promjena u različitim područjima života pa tako i u obrazovanju.

Literatura

Bulaja, A. (2016). Darovitost (Doctoral dissertation, University of Zagreb. Faculty of Teacher Education. Chair of Psychology).

Caspersen, M. E., Gal-Ezer, J., McGettrick, A., & Nardelli, E. (2018). Informatics for all the strategy. ACM.

García Peñalvo, F. J., Reimann, D., Tuul, M., Rees, A., & Jormanainen, I. (2016). An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers.J Gal-Ezer, E. Z. (2004). *The efficiency of algorithms—misconceptions.*

Krpan, D., Mladenović, S., & Zaharija, G. (2014). Vizualni programski jezici u visokom obrazovanju. URL: http://www. pmfst. unist. hr/wp-content/uploads/2014/06/Istraziva--kiseminar1-Bubica. pdf (pristupljeno: 19.9. 2021.).

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., ... & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. ACM SIGCSE Bulletin, 36(4), 119-150.

Mannila, L., & de Raadt, M. (2006, February). An objective comparison of languages for teaching introductory programming. In Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006 (pp. 32-37).

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B. D., ... & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In Working group reports from ITiCSE on Innovation and technology in computer science education (pp. 125-180).

Ministarstvo znanosti i obrazovanja. (2018.). Kurikulum nastavnog predmeta Informatika za osnovne škole i gimnazije. Zagreb: Ministarstvo znanosti i obrazovanja.

Mladenović, M. (2019). Poučavanje početnog programiranja oblikovanjem računalnih igara (Doctoral dissertation, University of Split. University of Split, Faculty of science).

Mohorovicic, S., & Strcic, V. (2011). An overview of computer programming teaching methods. In Central European Conference on Information and Intelligent Systems (p. 47). Faculty of Organization and Informatics Varazdin.

O'Neill, J. (2018). SPAE: A Scratch Project Analysis Tool for Educators (Doctoral dissertation, Appalachian State University).

Passey, D. (2017). Computer science (CS) in the compulsory education curriculum: Implications for future research. Education and Information Technologies, 22(2), 421-443.

Soloway, E., Adelson, B., & Ehrlich, K. (1988). Knowledge and processes in the comprehension of computer programs. The nature of expertise, 129-152.

Sternberg, R. J. (2006). The nature of creativity. Creativity research journal, 18(1), 87.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. Cognitive science, 12(2), 257-285.

Sweller, J. (2011). Cognitive load theory. In Psychology of learning and motivation (Vol. 55, pp. 37-76). Academic Press.

Watt, M. (1982). What is Logo?. Creative Computing, 8(10), 112-29.

Zelle, J. M. (2004). Python programming: an introduction to computer science. Franklin, Beedle & Associates, Inc.

Žanko, Ž., Mladenović, M., & Boljat, I. (2019). Misconceptions about variables at the K-12 level. Education and Information Technologies, 24(2), 1251-1268.

Popis tablica

Popis slika

Privitak

Istraživačka anketa: https://forms.gle/Xummi95y4vtK2R8q6

Učenje osnovnih koncepata programiranja

Poštovani,

ovaj upitnik provode studenti završne godine diplomskog studija Informatike na Prirodoslovno-matematičkom fakultetu u Splitu. Želimo dobiti uvid u razlike učenja osnovnih koncepata programiranja u odnosu na prethodno učenje kroz osnovnu i/ili srednju školu.

Upitnik je anoniman, za ispunjavanje potrebno Vam je 5 - 10 minuta. Molimo Vas da odgovorite na sva pitanja.

Zahvaljujemo na Vašem vremenu i suradnji.

---

*Obavezno

1. Jeste li u osnovnoj ili srednjoj školi slušali predmet Informatika? *

   Označite samo jedan oval.

   - Da
   - Ne

2. Ako da, jeste li se u sklopu predmeta Informatika susretali s programiranjem?

   Označite samo jedan oval.

   - Da
   - Ne

3. Ako da, navedite s kojim ste se programskim jezicima susreli (moguće je odabrati više odgovora).

Odaberite sve točne odgovore.

- [ ] Logo
- [ ] Python
- [ ] Scratch
- [ ] pseudokod

Ostalo: [ ] ____________________

4. Koja je bila Vaša konačna ocjena iz kolegija Programiranje 1? *

Označite samo jedan oval.

	1	2	3	4	5	
Nepoloženo	◯	◯	◯	◯	◯	Odličan

5. Koliko redovito ste učili/ponavljali naučeno gradivo? *

Označite samo jedan oval.

	1	2	3	4	5	
Nikada	◯	◯	◯	◯	◯	Iznimno redovito

6. Koliko su Vam bili jasni koncepti naučeni tijekom predavanja? *

Označite samo jedan oval.

	1	2	3	4	5	
Nimalo jasni	◯	◯	◯	◯	◯	Potpuno jasni

7. Koliko su Vam bili jasni koncepti naučeni tijekom vježbi? *

Označite samo jedan oval.

	1	2	3	4	5	
Nimalo jasni	◯	◯	◯	◯	◯	Potpuno jasni

8. Koliko često ste tražili konzultacije kod profesora tijekom semestra? *

Označite samo jedan oval.

◯ Nikada

◯ Tjedno

◯ Mjesečno

◯ Ostalo: ______________________

9. Jesu li Vam naučeni koncepti postali razumljiviji nakon konzultacija?

Označite samo jedan oval.

	1	2	3	4	5	
Ostali su mi nejasni	⬭	⬭	⬭	⬭	⬭	Sve sam razumio/la

10. Bavite li se programiranjem u slobodno vrijeme? *

Označite samo jedan oval.

- ⬭ Da
- ⬭ Ne

11. Ako da, koliko često programirate u slobodno vrijeme?

Označite samo jedan oval.

- ⬭ Svaki dan
- ⬭ Jednom tjedno
- ⬭ Više puta tjedno
- ⬭ Mjesečno
- ⬭ Ostalo: ____________________

12. Ukoliko se bavite programiranjem u slobodno vrijeme, koje programske jezike koristite?

____

Slijedeća pitanja popunjavaju studenti koji su imali programiranje u školi ili su samostalno učili programiranje u slobodno vrijeme.

Možete kliknuti "Dalje" ukoliko niste ranije učili programiranje.

13.

Označite samo jedan oval po retku.

	Ne može se primijeniti	Uopće se ne slažem	Djelomično se ne slažem	Niti se slažem, niti ne slažem	Djelomično se slažem	U potpunosti se slažem
Mogao/la sam jednostavno prepoznati naučene koncepte iz programiranja u osnovnoj/srednjoj školi.	⬭	⬭	⬭	⬭	⬭	⬭
Pisanje pseudokoda prije kodiranja mi je pomoglo bolje razumjeti problem koji je trebalo implementirati.	⬭	⬭	⬭	⬭	⬭	⬭

Korištenje vizualnih programskih jezika (Scratch) mi je pomoglo pri razumijevanju koncepata naučenih na kolegiju Programiranje 1.						
Korištenje drugih programskih jezika pomoglo mi je pri razumijevanju programskog jezika Python korištenog na kolegiju Programiranje 1.						

Slijedeća pitanja popunjavaju studenti koji nisu imali programiranje u školi i nisu samostalno učili programiranje u slobodno vrijeme.

Možete kliknuti "Dalje" ukoliko ste odgovorili na prethodna 4 pitanja.

14.

Označite samo jedan oval po retku.

	Uopće se ne slažem	Djelomično se ne slažem	Niti se slažem, niti ne slažem	Djelomično se slažem	U potpunosti se slažem
Smatram da sam bio/la u zaostatku naspram ostalih kolega prilikom učenja gradiva za kolegij Programiranje 1.	⬭	⬭	⬭	⬭	⬭
Obično mi je trebalo duže vremena da naučim neki koncept na kolegiju Programiranje 1 naspram ostalih kolega.	⬭	⬭	⬭	⬭	⬭

Mogao/la sam jednostavno prepoznati i primijeniti koncepte naučene na predavanjima pri izvođenju vježbi.					
Prilikom izvođenja vježbi, bilo mi je uglavnom jasno što se od mene u pojedinom zadatku traži.					
Materijali sa predavanja i vježbi bili su mi razumljivi pri samostalnom radu.					
Pisanje pseudokoda prije kodiranja na kolegiju Programiranje 1 mi je pomoglo bolje razumjeti problem koji je trebalo implementirati.					

Vaši komentari...

15. Za kraj, napišite Vaše mišljenje o učenju osnovnih koncepata programiranja (koja je dob prikladna za učenje, na koji način bi se moglo olakšati studentima koji nisu prethodno učili programiranje...).