Poželjne karakteristike jezika za poučavanje programiranja u osnovnoj školi

Radošević, Matea

Master's thesis / Diplomski rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:096758

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-13**

Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Matea Radošević

POŽELJNE KARAKTERISTIKE JEZIKA ZA POUČAVANJE PROGRAMIRANJA U OSNOVNOJ ŠKOLI

Diplomski rad

Voditelj rada:
doc. dr. sc. Goranka Nogo

Zagreb, 2017.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Hvala doc. dr. sc. Goranki Nogo na pomoći i strpljivosti tijekom izrade rada.
Hvala mojoj obitelji, prijateljima, kolegama, profesorima i svim dragim ljudima koji su bili uz mene tijekom mog dosadašnjeg obrazovanja.

Sadržaj

Uvod

Poučavanje programiranja u osnovnim školama postaje sve važniji dio nastave informatike, ali i nastave u osnovnim školama općenito. Također, učenje programiranja nastoji se popularizirati među učenicima. Dok učenicima najviše problema zadaje razvijanje algoritamskog načina mišljenja, nastavnicima, između ostalog, probleme zadaje i odabir programskog jezika. Taj je odabir važan jer je programski jezik bitan čimbenik koji utječe na to hoće li se učenik zainteresirati za programiranje. Programski jezik u kojem učenici uče programirati trebao bi biti jednostavan i primjeren učenicima osnovnih škola, ali bi morao predstavljati i prvi korak ka stvarnom programiranju te bi trebao učiniti prijelaz s početničkog na ozbiljnije programiranje što bezbolnijim. Prema važećem Katalogu obveznih udžbenika i pripadajućih dopunskih nastavnih sredstava za osnovnu školu [2] koriste se programski jezici Logo, Python i QBasic. Na natjecanjima se koriste i programski jezici Basic, C, C++ i Pascal. U nekim školama, učenici svoje prve programe pišu u programskom jeziku Scratch.

U radu će biti navedene i pojašnjene poželjne karakteristike jezika za poučavanje programiranja. Cilj ovog rada je usporediti programske jezike Logo, Scratch i Python te argumentirati njihovu primjerenost za poučavanje programiranja u osnovnoj školi.

Rad se sastoji od tri poglavlja. U prvom poglavlju su detaljno opisane poželjne karakteristike jezika za poučavanje programiranja te su opisani programski jezici Scratch, Logo i Python. Drugo poglavlje sadrži rezultate nekih dosad provedenih istraživanja, a koja se odnose na utjecaj na motivaciju učenika, utjecaj programskog jezika na percepciju, učenje i ishode učenja te prijelaz s vizualnog na tekstualni programski jezik. Treće poglavlje opisuje karakteristike odabranih programskih jezika te navodi koji programski jezici imaju poželjne karakteristike jezika za poučavanje programiranja navedene u prvom poglavlju.

Poglavlje 1

Jezici za poučavanje programiranja

Postoje barem tri vrste programskih jezika: slikovni jezici (*engl. image or picture languages*), blokovski jezici (*engl. block or bubble languages*) i pravi programski jezici (*engl. "real" programming languages*) [15]. Slikovni programski jezici koriste se slikama i drugih raznim vizualnim prikazima kako bi se učenicima približile osnovne ideje programiranja i algoritamskog mišljenja. Blokovski jezici koriste blokove s naredbama, a ponekad i stvarni kod, kako bi se riješili zadani problemi. Primjer blokovskog programskog jezika je Scratch. Blokovski jezici predstavljaju prijelaz sa slikovnog programskog jezika na tradicionalni programski jezik. Pravi programski jezici su zapravo tekstualni programski jezici, a i oni se, ako se objašnjavaju korak po korak, mogu koristiti kako programski jezici za poučavanje programiranja.

Programiranje se smatra načinom na koji učenici mogu razvijati svoje intelektualne sposobnosti rješavajući izazovne probleme. Dolaskom programskog jezika Logo očekivalo se da će učenici koji su učili programirati u Logu postizati puno bolje rezultate i biti vještiji u programiranju. Međutim, istraživanja nisu pokazala znatnu razliku u postignutim rezultatima. Učenici koji su učili programirati u Logu su najviše poteškoća imali zbog ograničenog programerskog okruženja. Primjerice, prilikom programiranja, svaka linija koda mora biti sintaktički savršeno ispravna. Zbog toga učenici svoju pažnju prilikom pisanja programskog koda usmjeravaju gotovo isključivo na sintaksu, a ne koncentriraju se na semantiku.

Funkcionalnost Scratcha kao programskog jezika je slična Logu. Ipak, glavna razlika je u tome što se dijelovi programskog koda stavljaju na određeno mjesto u sintaktički ispravnom obliku pa su pogreške uvijek semantičke, a nikad nisu zbog greške u tipkanju ili zato što su učenici krivo zapamtili sintaksu. Scratch je vizualni programski jezik osmišljen tako da bi kompliciraniji elementi poput petlji i postavljanja uvjeta učenicima bili što prirodniji [13].

Python sve više ulazi kao jezik za poučavanje programiranja na svim razinama obra-

zovnog sustava. Python nudi interaktivno sučelje u kojem učenici mogu istraživati funkcionalni, proceduralni i objektno orijentirani način rješavanja problema. Strukture podataka i jednostavna sintaksa čine Python dobrim jezikom za učenje programiranja, a postojeće knjižnice (*engl. Libraries*) omogućavaju da se pomoću Pythona riješe razni kompleksni problemi [1].

1.1 Poželjne karakteristike jezika za poučavanje programiranja

Kreatori jezika za poučavanje programiranja, Seymour Papert (kreator Loga), Niklaus Wirth (kreator Pascala), Guido van Rossum (kreator Pythona) predložili su kriterije koje trebaju zadovoljavati programski jezici za poučavanje programiranja. [14] Ukupno 17 kriterija grupirano je u smislene cjeline koje su prikazane u Tablici 1.1.

Učenje	**Dizajn i okruženje**	**Podrška i dostupnost**	**Programiranje**
- pogodan za poučavanje - jednostavna sintaksa i prirodna semantika - fizička analogija - općenita slika o programiranju - novi pristup poučavanju	- interaktivno i jednostavno okruženje - povratna informacija o radu - ispravni programi - mogućnost podjele koda na manje dijelove - razlika između ideje i programskog koda	- podrška korisnicima - dostupan izvorni kod - rad na različitim platformama - obrađen u udžbeniku - besplatan i lako dostupan	- različite razine znanja - programiranje u druge svrhe - kompliciraniji programi

Tablica 1.1: Poželjne karakteristike jezika za poučavanje programiranja

Prva skupina odnosi se na učenje programskog jezika. Bitno je da je programski jezik pogodan za poučavanje. Za ispunjavanje ovog kriterija programski jezik treba biti osmišljen tako da se razmišljalo i o nastavi. Uz to, treba imati jednostavnu sintaksu i prirodnu semantiku te izbjegavati kratice, a pridruženi alati trebaju biti jednostavni za korištenje. Nadalje, da bi zadovoljio kriterije za učenje, programski jezik treba se moći koristiti za primjenu fizičke analogije. Kako bi se zadovoljio taj kriterij, programski jezik bi trebao omogućiti učenicima da mogu uvidjeti utjecaj napisanog programskog koda na stvarne situacije. Potrebno je uložiti napor kako bi učenici bili na razini na kojoj mogu iskoristiti taj potencijal i primjenjivati svoje znanje u različitim okruženjima. Primjerice, programiranje fizičkih objekata jedan je od načina gdje učenici primjenjuju programski jezik za fizičku analogiju i odmah uočavaju rezultat svog rada, čak i s relativno malo programerskog znanja. Također, programski jezik treba davati općenitu sliku o programiranju.

Programski jezik treba omogućiti učenje osnova i temeljnih načela programiranja, koji će kasnije poslužiti kao odličan temelj za kasnije programiranje u drugim programskim jezicima. Isto tako, programski jezik treba promovirati novi pristup poučavanju. Ne smije se ograničiti samo na implementaciju već treba pokriti i druge aspekte procesa razvoja softvera. Trebao bi biti metodički osmišljen kao niz procesa potrebnih za izgradnju softvera koji se uključuje sami programski jezik te skup načela, alata i knjižnica.

Sljedeća skupina kriterija opisuje dizajn i okruženje poželjno za programske jezike za učenje programiranja. Oni trebaju biti interaktivni i omogućavati brzi razvoj programskog koda. Učenicima, koji su novi u programiranju, bitno je da mogu biti kreativni i napisati program koji žele unatoč tome što ne poznaju programski jezik koji koriste do u detalje. Mogućnost da se može jednostavno započeti pisanje prvih programa je motivirajuće i inspirativno za učenike. Kako bi zadovoljili ovaj kriterij, programski jezik za učenje programiranja treba ohrabriti učenike da napišu svoje prve programe iako još nisu sasvim upoznati s programskim jezikom. To se može ostvariti ako je okruženje za pisanje programa interaktivno, jednostavno i intuitivno. Također, treba davati povratnu informaciju o učeničkom radu. Sljedeći kriterij koji bi programski jezik za učenje programiranja trebao zadovoljiti je da potiče učenike da programiraju ispravne (točne) programe. Cilj je da se izbjegne programiranje na temelju pokušaja i pogreške. Dakle, programski jezik treba osigurati da učenici znaju da je programski kod koji su napisali ispravan i da ne sadrži greške u kodu. Nadalje, pomoću programskog jezika učenici trebaju moći riješiti određene problemske zadatke tako da su podijeljeni u manje smislene podzadatke. Osoba koja piše programski kod treba se koncentrirati na svaki pojedinačni dio zadatka kojeg treba riješiti prije nego prijeđe na sljedeći. a svi pojedinačni dijelovi zajedno čine cjelinu. Zato je jedna od poželjnih karakteristika programskog jezika za poučavanje programiranja i mogućnost podjele programa na manje dijelove, vodeći računa o hijerarhijskoj strukturi tih dijelova. Da bi se ostvario taj kriterij, programski jezik treba podržavati modeliranje problema pomoću funkcija, procedura itd. Za učenike koji su pišu svoje prve programe važno je da razumiju svaki dio napisanog programa te da budu svjesni zašto je svaki od dijelova potreban da bi program bio ispravan. Neki programski jezici koriste kratice i simplificiraju kod, što je dobro ako programe pišu iskusni programeri. Programski jezik za učenje programiranja trebao bi pomoći učenicima da shvate razliku između ideje za rješavanje programa i samog programskog koda. Primjerice, to se može ostvariti pomoću dijagrama toka (bilo prije pisanja programskog koda bilo kao pojašnjenje nakon što je kod već napisan). Zato je još jedna od poželjnih karakteristika programskog jezika za poučavanje programiranja i intuitivno grafičko sučelje za dizajn i implementaciju koji omogućava jednostavno dohvaćanje knjižnica i ostalih opcija koje nudi programski jezik.

Jedna od skupina poželjnih karakteristika je i podrška i dostupnost programskog jezika te materijala za poučavanje. Tu se ubraja podrška korisnicima. Korištenje programskog jezika u svrhu poučavanja ovisi o podršci tom jeziku u programerskoj zajednici. Izvori ma-

terijala i podrška korisnicima mogu biti ograničavajući faktor za nastavnike i za učenike. Da bi se zadovoljio taj kriterij, potrebna je podrška za nastavnike i učenike u različitim oblicima. To mogu biti *web*-stranice, knjige, dodatni zadaci, riješeni primjeri, dokumentacija, forumi itd. Također, poželjno je da programski jezik ima softver čiji je izvorni kod i/ili dizajn dostupan javnosti za korištenje, uvid, izmjene i daljnje poboljšanje. To je bitno jer se smanjuju troškovi, ali i zbog toga što se može uvijek raditi na poboljšanju programskog jezika kako bi bio što bolji za poučavanje. To je još jedna poželjna karakteristika programskog jezika za poučavanje programiranja, a da bi bila ostvarena bilo bi dobro kad bi programski jezik bio proizvod grupe ljudi koja nema za cilj unovčiti svoj proizvod te prihvaća savjete zainteresiranih za taj programski jezik. Dakako, poželjno je da programski jezik funkcionira u različitim okruženjima, tj. treba biti dostupan za rad na različitim platformama, te da bude besplatan i lako dostupan svugdje u svijetu uz pripadajuće materijale za poučavanje programiranja u tom programskom jeziku. Programski jezik za poučavanje programiranja trebao bi biti obrađen u udžbeniku te bi se u tom programskom jeziku učenicima trebali pojasniti osnovni pojmovi.

Sljedeća skupina opisuje karakteristike koje se ne odnose isključivo na poučavanje programiranja početnika. Važno je koliko je programski jezik primjenjiv za poučavanje na različitim razinama znanja učenika jer nije praktično ni ekonomično da se za svaku višu razinu znanja prelazi na poučavanje u drugom programskom jeziku. Također, dobro je ukoliko se programski jezik koristi i u druge svrhe, a ne samo kao programski jezik za poučavanje programiranja. To je naročito važno zbog učenika koji su napredni i žele učiti više. Tada oni mogu probati napisati program kojim će realizirati neku svoju ideju i vidjeti stvarnu svrhu programiranja ili se mogu okušati u programiranju fizičkih objekata pišući programske kodove u programskim jezicima koje ih poučavaju nastavnici u školi. Zato su poželjne karakteristike programskog jezika da se ne koristi samo za poučavanje i da se njegova primjena može proširiti na programiranje u specifičnim granama. Primjerice, ukoliko učenici trebaju napisati program u sklopu referata iz biologije tada bi bilo korisno da programski jezik ima implementirane određene funkcije potrebne koje se koriste u biologiji. Također, programski jezik treba moći izvoditi i nešto kompliciranije programe u razumnom vremenu upravo zbog učenika koji žele napisati kompliciraniji program i vidjeti stvarnu korist naučenoga na nastavi.

1.2 Jezici za poučavanje programiranja u osnovnoj školi

U osnovnim školama u Hrvatskoj učenici uče programirati ponajviše u programskim jezicima Logo, Pascal, Phyton, Basic i Scratch. Dakle, ne postoji standard, tj. ne postoji obavezni programski jezik u kojem učenici trebaju naučiti programirati u osnovnoj školi. U ovoj sekciji navest će se neke od osnovnih karakteristika programskih jezika Scratch, Logo i Python.

Scratch

Scratch je besplatni vizualni programski jezik. Razvila ga je grupa Lifelong Kindergarten, dio MIT (Massachusetts Institute of Technology) Media Laba. Grupa Lifelong Kindergarten surađivala je s tvrtkom Lego na izradi robota Lego Mindstorms koji se koristi za poučavanje programiranja. Uočilo se da rad s Lego kockama potiče kreativnost kod djece te da ima puno mogućnosti kako djeca mogu napraviti što god zamisle pa su osmislili programski jezik koji podsjeća na slaganje kockica. Blokovi s naredbama su napravljeni u obliku slagalica tako da je vizualno jasno kako se naredbe mogu složiti. Blokovi s naredbama su grupirani tematski, a razlikuju se po obliku i bojama.

Scratch koriste učenici, studenti, znanstvenici, nastavnici i roditelji kako bi na jednostavan način napravili vlastite animacije, igre itd. Iako je prvenstveno namijenjen poučavanju učenika od 8 do 11 godina starosti, Scratch predstavlja prvi korak prema naprednijem programiranju. Može se koristiti u edukativne i znanstvene svrhe, naročito u području matematike i drugih znanstvenih projekata. Primjerice, može se koristiti za simulaciju i vizualizaciju eksperimenata. Na službenoj stranici može se pronaći niz projekata koji se mogu modificirati, a registracija nije potrebna. Scratch je dostupan online i kao aplikacija za operacijske sustave Windows, MacOS i Linux. Prvi put se pojavio 2002. godine kao probni programski jezik, a službeno postoji od 2015. godine [4].

Ime programskog jezika Scratch dolazi od engleske riječi ”scratching” što u doslovnom prijevodu znači ”grebanje” dok u ovom slučaju predstavlja ponovno korištenje programskog koda koji se može lako kombinirati, mijenjati i nadopunjavati. Upravo to je i glavna značajka Scratcha, činjenica da korisnici mogu besplatno preuzeti gotove programske kodove koje mogu modificirati. Scratch je postao popularan najprije u Velikoj Britaniji zahvaljujući Code Clubovima. Koristi se kao programski jezik za poučavanje programiranja jer je kreiranje zanimljivih programa relativno jednostavno, a naučene vještine mogu se primijeniti na druge temeljne programske jezike poput Phytona i Jave. Ipak, ne služi samo za kreiranje igrica. Koristeći vizualizacije, može se služiti za kreiranje animiranih priča, informativnih tekstova itd. Već postoji više programa koje učenici mogu koristiti kako bi više naučili o pojedinim cjelinama iz matematike, povijesti pa čak kako bi naučili više i o fotografiranju. Fleksibilnost Scratcha omogućuje nastavnicima stvaranje konceptualnih i vizualnih predavanja i zadataka za učenike. Predstavlja koristan alat za kreiranje animacija koje pomažu učenicima da vizualiziraju komplicirane koncepte poput stanične mitoze, Galileovog termometra i ostalih. Također, nastavnici mogu kreirati kvizove, igre i predavanja kako bi potaknuli učenike na razmišljanje, ali i kako bi poboljšali interakciju s učenicima. Korištenje Scratcha potiče učenike da bolje shvate logiku programiranja te kako na kreativan način nadograđivati postojeće programske kodove i surađivati s drugim učenicima. Neki predavači sa Sveučilišta Harvard preferiraju korištenje Scratcha za uvođenje programiranja u nastavu. Ipak, Scratch nije namijenjen fakultetskoj razini znanja pa nakon prvog tjedna nastave u Scratchu predavači poučavaju programiranje u programskom jeziku C.

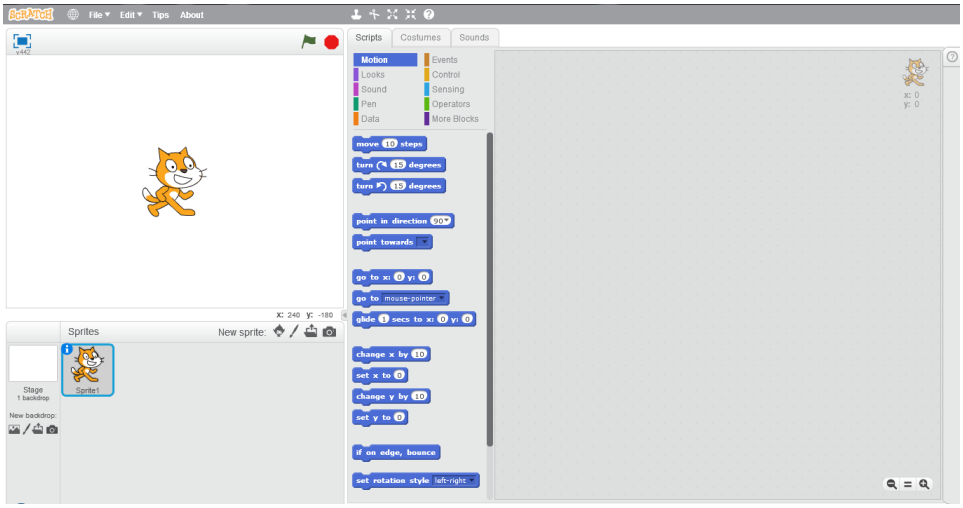
Postoji online zajednica ScratchEd koja okuplja nastavnike koji dijele svoje priče iz učionice, razmjenjuju nastavne materijale i postavljaju pitanja [3]. ScratchEd je proizvod Harvard Graduate School of Education koji i dalje podupire rad te zajednice. Također, organiziraju se okupljanja (Scratch Educator Meetups) nastavnika koji žele učiti zajedno i učiti jedni od drugih te dijeliti svoje ideje i strategije kako bi poduprli kreativnost programiranja u svim oblicima.

Program u Scratchu (Scratch projekt) čine objekti koji se zovu likovi (*engl. sprites*). Izgled lika moguće je mijenjati dodavanjem različitih kostima (*engl. costumes*). Lik može izgledati poput osobe, vlaka, leptira ili bilo čega drugoga. Glavni lik je mačak koji je i logo programskog jezika Scratch. Za kostim je moguće koristiti bilo koju sliku. Učenici mogu izraditi crtež u Paint Editoru, mogu učitati sliku s računala ili metodom uhvati-povuci-pusti (*engl. drag-and-drop*) mogu preuzeti sliku s internetske stranice. Lik se ponaša prema danim uputama: može se kretati, razgovarati, reproducirati glazbu ili komunicirati s drugim likovima. Za izdavanje naredbi potrebno je složiti blokove s naredbama u smislene cjeline, nazvane skripte.

Scratch blokovi razvrstani su u deset skupina različitih boja: Kretanje, Izgled, Zvuk, Olovka, Podaci, Događaji, Upravljanje, Očitanja, Operacije i Varijable. Svaka skupina blokova ima drugačiji izgled koji sugerira na koji se način blokovi kombiniraju s blokovima iz drugih skupina (ili iz iste skupine). Svojim izgledom blokovi određuju na koji način se više blokova povezuje u cjelinu te se na taj način izbjegavaju pogreške (tj. pogreške u sintaksi zbog tog načina rada gotovo nisu moguće). Iako nema pogrešaka u sintaksi, program može raditi krivo. Točnije, moguće je da program ne radi ono što učenik očekuje, ali eksperimentiranjem se lako dolazi do željenog rezultata. Klikom na skriptu, program se izvršava od početka do kraja, tj. prolazi kroz blokove od prvog (najvišeg) do zadnjeg (najnižeg). Ako učenici žele provjeriti što će se dogoditi nakon jedne naredbe, klikom na blok s tom naredbom lik na ekranu će naredbu odmah izvršiti. Ako se nalaze dodatni blokovi s naredbama izvan skripte, skripta (program) će se izvršiti neovisno o tim naredbama.

Likovi koji izvršavaju naredbe nalaze se na pozornici (*engl. stage area*). Likovi se kreću po pozornici, a izgled pozornice moguće je promijeniti. Širina pozornice je 480 piksela, a visina 360 piksela. Pozicioniranje na pozornici ostvaruje se pomoću koordinatnih osi (os x i os y). Središte pozornice ima x-koordinatu 0 i y-koordinatu 0, tj. nalazi se u (0,0). Krajnje desna točka ima s-koordinatu 240, a krajnje lijeva -240, dok točke na vrhu ekrana imaju y-koordinatu 180, a na samom dnu ekrana -180. Kako bi saznali koja je pozicija na pozornici, potrebno je pomicati kursor miša i pratiti x–y prikaz koji se nalazi na dnu pozornice. Slika 1.1 prikazuje izgled grafičkog sučelja programskog jezika Scratch.

U Scratchu su prisutne neke uobičajene naredbe poput naredbe If te naredbe If-Else. Također, u skupini Upravljanje su i petlje Repeat i Forever. Učenici u ovom programskom okruženju mogu dobro razumjeti važnost petlje Forever, pogotovo prilikom programiranja fizičkih objekata. Primjerice, naredbe za robot mBot pišu se u mBlocku, grafičkom



Slika 1.1: Grafičko sučelje programskog jezika Scratch

programskom okruženju koje se bazira na Scratchu 2.0, a čini programiranje na Arduino platformi jednostavnim i lako shvatljivim učenicima.

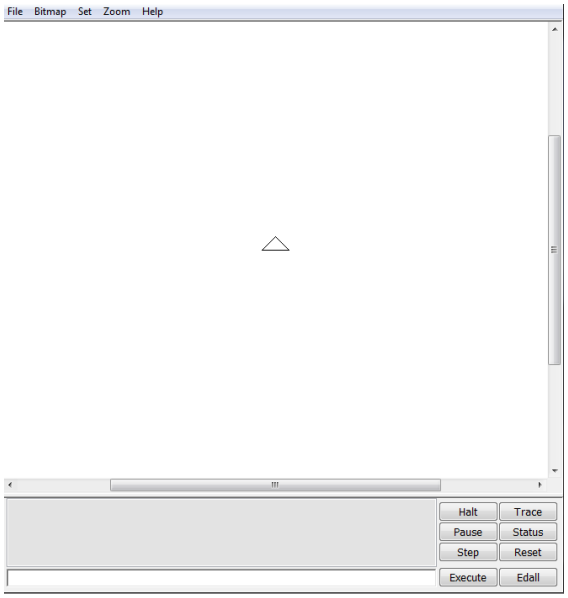
Zanimljivo je nadodati kako također postoji i Scratch kutak za izumitelje (Scratch For Developers). Budući da je Scratch zapravo i osmišljen kako bi poticao mlade osobe na kreativno razmišljanje, sustavno rezoniranje i surađivanje, postoje načela učenja i načela dizajniranja kojih se potrebno pridržavati pri nadopunjavanju Scratch okruženja. Što se tiče načela učenja, bitno je da se Scratch sastoji od više projekata jer smatraju da se radom na projektima najbolje uči, zatim bitno je da se programeri fokusiraju na one stvari do kojih im je posebno stalo jer će tada najviše doprinijeti i biti motivirani za svoj rad, važno je surađivanje s drugima i da svi projekti i zadaci budu poput igre, ali dovoljno izazovni i uvijek iznova s novim problemima koje je potrebno riješiti. Načela dizajna zahtijevaju da programski jezik bude prvenstveno jednostavan kako bi ga mogli koristiti učenici koji još ne znaju ništa o programiranju, ali ipak bi trebao pružati mogućnost i za rad na kompleksnijim zadacima. Scratch treba biti jednostavan za korištenje i intuitivan, treba biti primjenjiv i u drugim znanostima te u nastavni iz drugih predmeta u školi, a treba poticati učenike da isprobavaju nove stvari i da proces provjere ispravnosti programa bude brz.

Logo

Logo je jezik za poučavanje programiranja, kojeg su 1967. godine dizajnirali Wally Feurzeig, Seymour Papert i Cynthia Solomon. Intelektualni korijeni Loga su povezani s umjetnom inteligencijom, matematičkom logikom i razvojnom psihologijom. Poučavanje

programiranja pomoću programskog jezika Logo najbolje se može opisati kao konstruktivizam (teorija učenja koju se formulirao Jean Piaget). Konstruktivizam predstavlja učenje u kojem učenici dobivaju nova znanja u interakciji sa stvarima i ljudima oko sebe. Naziv Logo dolazi od grčke riječi *logos* koja u prijevodu na hrvatski znači *riječ* ili *um*, a Feurzeig ističe kako taj naziv služi i kako bi se programski jezik Logo razlikovao od drugih programskih jezika koji su orijentirani na brojeve, a ne na grafiku ili logiku [6]. Prva implementacija programskog jezika Logo bila je napisana u LISP-u na PDP-1. Cilj je bio kreirati matematičko okruženje u kojem će učenici imati priliku da se igraju s riječima i rečenicama uobličenim u naredbe. Još jedan od ciljeva dizajna Loga bio je da učenici mogu na neki način komunicirati s programskim jezikom te da dobivaju povratne informacije o pogreškama. Neke verzije Loga omogućavaju i 3D grafiku te rad s više kornjača istovremeno (npr. MSWLogo).

Unatoč tome što je Logo programski jezik opće namjene, najpoznatiji je po korištenju kornjačine grafike u kojoj se naredbe za pokretanje i crtanje izvršavaju na zaslonu ili s malim robotom koji se zove kornjača. Slika 1.2 prikazuje grafičko sučelje programskog jezika Logo.



Slika 1.2: Grafičko sučelje programskog jezika Logo

Dok pišu programske kodove u programskom jeziku Logo, učenici se stavljaju u poziciju kornjače kako bi mogli razumjeti i predvidjeti kako će se pokretati kornjača. Dakle, kornjača se pomiče u ovisnosti o svojoj trenutnoj poziciji (npr. RIGHT 90 znači da se kor-

njača treba okrenuti za 90 stupnjeva u desno u odnosu na položaj u kojem se u tom trenutku nalazi).

Programski jezik Logo je osmišljen kao programski jezik za poučavanje programiranja. Slijedeći taj cilj, Logo ima sljedeće značajke: interaktivan je, modularnost, rastezljivost i fleksibilnost tipova podataka. Interaktivnost se može uočiti pri javljanju pogrešaka. Iako neke verzije Loga imaju kompajler, Logo je osmišljen kao jezik u kojem će se prevoditi jedna po jedna naredba. Tako učenici dobivaju detaljne i opisne povratne informacije za svaku pojedinu instrukciju, a na taj način uče kako ispravljati greške i bolje razumiju poruke koje dobivaju od računala pri programiranju. Rastezljivost i modularnost podrazumijeva da se kompleksni projekti (zadaci) grade pomoću jednostavnih naredbi. Pisanje programskih kodova u Logu svodi se na dodavanje novih naredbi u vokabular Loga (nove naredbe sastoje se od niza bazičnih naredbi) tako da Logo uči nove riječi pomoću riječi koje već poznaje. Tako imitira način na koji ljudi uče strane jezike. Logo radi s riječima i listama. Riječ u Logu je string koji se sastoji od znakova, dok je lista u Logu niz riječi i/ili lista. Čak su i brojevi riječi, ali se ipak razlikuju jer se s njima može računati. Većina programskih jezika traži da programer jasno definira s kojim tipom podataka će raditi. To olakšava izvođenje programa, ali je kompliciranije za programera. U programskom jeziku Logo, nije potrebno navesti o kojem tipu podataka je riječ, a kad treba računati sa znakovima, to se obavi bez problema.

Učenici u Logu koriste uobičajene naredbe odluke poput If i If-Else te petlju For, While i Repeat. Sučelje programskog jezika Logo omogućava da je rezultat napisanog programskog koda odmah vidljiv. Također, u početnom prozoru učenici mogu vidjeti i sve poruke o pogreškama i objašnjenja što je netočno napisano. Postoji i Editor u kojem učenici mogu pisati programski kod, a zatim ga kompajlirati. Nakon toga potrebno je pozvati program u početnom prozoru, u kojem će biti vidljiv konačni rezultat programa.

Python

Tvorac Pythona je Nizozemac Guido van Rossum, a programski jezik Python nastao je 1991. godine [1]. Švicarac Niklaus Wirth dvadesetak godina ranije kreirao je programski jezik Pascal. Osnovna ideja obojice bila je kreiranje programskog jezika za učenje programiranja. Jedno od načela dizajna Pythona je čitljivost. Čitljiviji je od većine drugih programskih jezika jer ne koristi npr. vitičaste zagrade nego su pojedini dijelovi programskog koda uvučeni. Također, sintaksa omogućava programerima da u manjem broju linija koda naprave što su naumili (sintaksa je kraća i jednostavnija nego kod programskih jezika kao što su C++ ili Java).

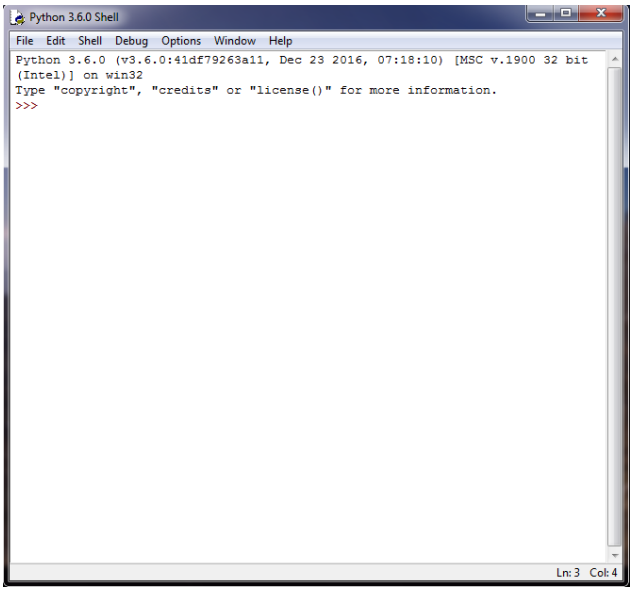
Glavni nedostatak Pascala jest to što ne podržava objektno-orijentirani princip programiranja. Guido van Rossum je napravio je jezik za učenje programiranja koji će po potrebi biti strukturni, a može biti i objektno-orijentirani. Upotreba Pythona u poučavanje je u

sve većem porastu u hrvatskim školama. Razlog rastu popularnosti je upravo činjenica da se radi o lako čitljivom jeziku s čistom i jednostavnom sintaksom. Na taj način učenici mogu usmjeriti svoju pažnju prema algoritamskom razmišljanju, a manje prema sintaksi programskog jezika [10].

Iako je tvorac Pythona osmislio Python kao programski jezik za učenje programiranja, Python je postao i popularan jezik za izradu poslovnih aplikacija. Mogućnosti Pythona prepoznate su u najpoznatijim svjetskim IT kompanijama, kao što su Google, Yahoo i sl. Trenutno je Python jedan od 5 najpopularnijih programskih jezika na svijetu [5].

Kao što je već naglašeno, Python je jezik s čistom i jednostavnom sintaksom. Same naredbe su jednostavne i intuitivne bez dodatnih naredbi ili simbola (vitičaste zagrade, begin, end, endif,...). To omogućava učenicima koji su početnici u programiranju da se posvete algoritmu problema koji trebaju riješiti, a pritom ne moraju previše razmišljati o samoj sintaksi. Za većinu programskih jezika potrebno je naučiti osnovna sintaktička i semantička pravila (tipovi podataka, deklaracija varijabli itd.).

U interpreterskom sučelju Pythona učenici pišu naredbe i odmah uočavaju što se točno događa nakon koje naredbe, a kasnije te naredbe postaju dio kompleksnijeg programa. Slično kao u programskom jeziku Logo, učenici mogu izvoditi jednostavne naredbe i odmah vidjeti rezultate njihova izvođenja. U interpreterskom sučelju, interpreter ispisuje trostruku strelicu ⋙ kad očekuje unos nove naredbe. Slika 1.3 pokazuje izgled grafičkog sučelja programskog jezika Phyton, verzija 3.6, tj. Python Shell.



Slika 1.3: Grafičko sučelje programskog jezika Python

Slično kao u programskim jezicima Logo i Scratch, u kojima se svaki pomak lika/kornjače vidi na ekranu, tako se i u Pythonu odmah može vidjeti što je rezultat izvršavanja pojedine naredbe. Kasnije kompleksniji programi (funkcije) nastaju kombinacijom jednostavnih naredbi, njihovim višestrukim izvođenjem, uvjetnim izvođenjem itd. Dakle, radi se o planski osmišljenom programskom jeziku za poučavanje programiranja. Također, i Python je programski jezik u kojem nema deklaracije varijabli što uvelike olakšava programiranje učenicima jer nema problema s različitim tipovima podataka i radom s tim podacima. Upravo problem varijable i tipa podataka je vrlo apstraktan učenicima koji tek počinju programirati, ali ipak je i neophodan ako kasnije žele programirati i rješavati kompleksnije probleme. U Pythonu su, kao i u Logu i Scratchu prisutne naredbe grananja (If, If-Else, Elif) i petlje (For, Repeat, While), a gotovo su identične kao i u drugim programskim jezicima.

Programski jezik Python dolazi na svim Linux i Macintosh računalima, a na Windows ga je računala moguće instalirati preuzimanjem programa sa službenih stranica. Instalacija programa je jednostavna, kao i instalacija programskih jezika Logo i Scratch. Nakon instaliranja softvera, Python se može pokrenuti iz komandne linije (naredbenog retka) ili korištenjem jednostavnog interaktivnog sučelja nazvanog IDLE. Poruke o greškama koje Python prijavljuje sadrže detaljnije objašnjenje (slično kao u Logu), no učenicima je najbitnije uočiti što piše u posljednjem retku u kojem interpreter prijavljuje vrstu greške.

Python, kao i Logo, koristi kornjačinu grafiku. Prilikom pisanja programskih kodova, često se lakše koristiti Python Editorom u kojem će pisanje programa za rješavanje kompleksnijih problema biti preglednije, a kasnije će se u sučelju IDLE pozivati programi ili pojedine funkcije.

Poglavlje 2

Rezultati istraživanja

U ovom poglavlju kratko će se predstaviti rezultati nekih istraživanja o primjerenosti korištenja nekih programskih jezika i njihov utjecan na interes učenika za programiranjem.

2.1 Utjecaj na motivaciju

Provedeno je istraživanje o utjecaju programskog jezika Alice na motivaciju učenika i kasniji izbor karijere. Programski jezik Alice se do sad nije spominjao u radu, ali ovo istraživanje je relevantno za temu koja se obrađuje jer je programski jezik Alice sličan programskom jeziku Scratch. On je bio dominantan programski jezik za poučavanje programiranja [8]. Kasnije je osmišljen programski jezik Scratch koji se, načinom na koji funkcionira, može usporediti s programskim jezikom Alice.

Programski jezik Alice omogućuje animiranje objekata u virtualnom svijetu. Njegovo stvaranje sastoji se od čitanja scenarija te dizajniranja, pisanja i testiranja programa. Scenarij daje sve potrebne detalje za postavljanje početne scene i planiranje niza naredbi, definira potrebne objekte koji imaju glavnu ulogu u priči i objekte u pozadini te akcije koje će se odvijati. Knjiga snimanja sastoji se od kronološkog niza metoda potrebnih da bi program radio ono što se od njega očekuje. Naredbe se slažu po principu uhvati-povuci-pusti, a tipkovnica je potrebna samo za definiranje vrijednosti svojstava. Dakle, slaganje programa odvija se na način sličan kao u programskom jeziku Scratch. Dijelovi programskog koda sastoje se od naredbi kojima objekti vrše neku radnju, izraza kojima označavamo matematičke operacije nad podacima te naredbi za uvjetno izvođenje If, If-Else i petlji. Ugrađene funkcije omogućavaju informacije o blizini objekata, njihovoj veličini, prostornim odnosima itd.

Uz istraživanja provedena u inozemstvu, naročito su zanimljiva istraživanja provedena u Hrvatskoj. Istraživanje je provedeno u jednoj splitskoj osnovnoj školi i općoj gimnaziji. Istraživanje je provedeno na uzorku od 18 učenika osmog razreda osnovne škole koji su

u izbornoj nastavi informatike prethodno učili programski jezik Logo i 28 učenika trećeg razreda opće gimnazije koji su na izbornoj nastavi uobičajeno pisali programske kodove u Pascalu. Odmah valja primijetiti da je istraživanje provedeno na malom proju učenika i u samo dvije škole. Iz tog razloga, rezultate treba oprezno tumačiti. Rezultati mogu ovisiti, ne samo o učenicima, već i o ranijem angažmanu nastavnika na nastavi informatike. Istraživanje je bilo podijeljeno na nekoliko dijelova: ispunjavanje ulazne ankete, demonstracija razvoja programa u okruženju Alice 2, samostalan rad učenika, provjera usvojenosti osnovnih koncepata i izlazna anketa.

Pokazalo se da su gotovo svi učenici naučili faze izrade programa u programskom jeziku Alice. Učenici osnovne i srednje škole su, podjednako uspješno, svladali pojmove klase i objekata. Učenici osnovne škole pokazali su veći angažman i entuzijazam za učenjem programiranja u programskom jeziku Alice. Ipak, učenici osnovne i srednje škole su jednako procijenili prikladnost programskog jezika Alice za poučavanje programiranja. Ugodno okruženje, mogućnost da se pogreške brzo i lako detektiraju i otklone, učinile su programiranje u programskom jeziku Alice manje zastrašujućim za 44% učenika osnovne škole i 28% gimnazijalaca. Motivacija za učenje programiranja prije rada s Alice značajno je veća kod učenika osnovne škole nego kod gimnazijalaca. Iako je autor predvidio drugačije rezultate, tijekom istraživanja motivacija za učenje programiranja se smanjila u obje skupine, a posebno kod učenika osnovne škole. Autor napominje da bi se objašnjenje možda moglo pronaći u Vroomovoj teoriji očekivanja koja govori da jačina i usmjerenje djelovanja na određen način ovisi o razini očekivanja da će djelovanje biti popraćeno određenim ishodom i o privlačnosti toga ishoda za pojedinca. Više o rezultatima istraživanja može se pronaći na poveznici [8].

2.2 Utjecaj programskog jezika na percepciju, učenje i ishode učenja

Sljedeće istraživanje provedeno je u Sjedinjenim Američkim državama među učenicima od 10 do 12 godina starosti [13]. Učenici su upisali ljetni tečaj "Stvaranje glazbe, filmova i igara uz pomoć računala" (*engl. "Making Music, Movies, and Games with Computers"*) u sklopu programa Academic Talent Development 346 (ATDP) na Sveučilištu u Kaliforniji, Berkeleyu. Tečaj je trajao 36 sati raspoređeno u 12 dana. Svi polaznici tečaja završili su peti razred osnovne škole. Polaznici su mogli birati žele li imati nastavu prijepodne ili poslijepodne. Također, bili su podijeljeni u dvije grupe. Jedna grupa prvo je učila programirati u Scratchu 6 dana, a zatim u Logu, dok je druga grupa je prvo 6 dana učila programirati u Logu, a nakon toga u programskom jeziku Scratch. Ovo istraživanje se fokusira na prvih 6 dana učenja, percepcije i učenje prije početka učenja drugog programskog jezika. S učenicima su radili isti nastavnici, autor istraživanja i dva pomoćna nastavnika. U

istraživanju su sudjelovale djevojčice i dječaci (jednoj grupi bilo je 16 dječaka i 10 djevojčica, a u drugoj 17 dječaka i 7 djevojčica).

Polaznici su svaki dan pisali testove kojima se ispitivalo koliko toga novog su naučili, ali i kakav je njihov stav prema programiranju. Pokazalo se da Logo utječe na razvoj samopouzdanja učenika kad je u pitanju programiranje, povećava interes za programiranjem i pomaže da učenici shvate pojam petlje, dok Scratch bolje utječe na razumijevanje naredbi grananja. Izgled Scratcha i način na koji učenici grade programe u Scratchu sugeriraju da bi učenici trebali bolje shvatiti osnovne pojmove, ali polaznici koji su učili programirati u Scratchu su postigli odlične rezultate u ispitima u kojima se provjeravalo razumijevanje petlji, čak bolje od polaznika koji su učili programirati u Logu. Isto tako, pretpostavljalo se da će polaznici koji uče programirati u Logu imati problema s pisanjem kodova zbog nepoznavanja sintakse, ali moguće je da su upravo zbog toga što su se morali koncentrirati na sve detalje prilikom programiranja o njima više i promišljali. Zato su, koncentrirajući se na svaki liniju koda, bolje shvatili i pojam petlji. Moguće je da je upravo to što su se fokusirali na detalje prilikom pisanja programa nadomjestilo mogući nedostatak vizualnog prikaza programskog koda. Tek u usporedbi programskog jezika Scratch s programskim jezikom poput Loga, u kojem učenici trebaju samostalno pisati kod, može se uočiti kako na učenike utječu vizualni programski jezici. Ovim istraživanjem nisu se potvrdile hipoteze da vizualni programski jezici potiču učenike da se bave programiranjem ili da im daju više samopouzdanja što se tiče programiranja. Štoviše, neznatne su razlike u rezultatima koji pokazuju koliko učenici vjeruju u vlastite sposobnosti.

Treba uzeti u obzir da je istraživanje provedeno na malom uzorku te da su u istraživanju sudjelovali isključivo učenici koji su dobrovoljno upisali tečaj. Dakle, to su vjerojatno učenici koji su i inače zainteresirani za programiranje i motivirani za rad. Moguće je da bi rezultati bili drugačiji da je istraživanje provedeno u školi, u razredima u kojima su učenici s različitim interesima i sposobnostima.

2.3 Prijelaz s vizualnog na tekstualni programski jezik

Britanski znanstvenici su, potaknuti pojavom sve više vizualnih jezika za poučavanje programiranja, odlučili otkriti je li učenicima koji uče programirati u vizualnom programskom jeziku kasnije jednostavnije programirati u tekstualnim programskim jezicima [11]. Istraživanje je provedeno tako da su uključeni i nastavnici drugih nastavnih predmeta, a naročito matematike jer su zadaci bili povezani s geometrijskim oblicima, simetrijama i pravilima.

Britanski nacionalni kurikulum [7] [12] ne daje mišljenje o korištenju vizualnog jezika za poučavanje programiranja, jedino je bitno da učenici u srednjoj školi programiraju u dva različita programska jezika od kojih barem jedan mora biti tekstualni programski jezik. Nacionalni kurikulum propisuje samo minimalna znanja koja učenik treba steći, a

posebno naglašava da nastavnici trebaju dobro osmisliti način na koji će učenici prijeći s vizualnog na tekstualni programski jezik. Također, naglašavaju korelaciju informatike s drugim nastavnim predmetima, kako bi se obogatila nastava svih predmeta. Iako su prije provedenog istraživanja predviđali drugačije rezultate, zaključili su da se vizualni programski jezik treba koristiti za poučavanje programiranja, ali ne umjesto tekstualnog jezika, već uz tekstualni programski jezik (kao način pisanja pseudokoda). Kombinacija vizualnog i tekstualnog programskog jezika trebala bi učenicima povećati samopouzdanje i učiniti ih sposobnijima napisati programske kodove u tekstualnom programskom jeziku, u osnovnim i srednjim školama. Nadalje, zaključili su da bi uz podršku stručnih suradnika, metodički osmišljenu nastavu i strategiju potpore svim učenicima dok prelaze s vizualnog na tekstualni programski jezik, učenici mogli već u osnovnoj školi naučiti programirati u tekstualnom programskom jeziku. Treba uzeti u obzir da u Velikoj Britaniji, gdje je istraživanje provedeno, osnovna škola (*engl. primary school*) obuhvaća prvih pet razreda, nakon toga učenici upisuju srednju školu (*engl. secondary school*). Istraživanje je također pokazalo da bi učenici u srednjoj školi trebali moći samostalno napisati programe u tekstualnom programskom jeziku, a vizualni programski jezik, poput Scratcha, je dobar uvod u programiranje u osnovnim i srednjim školama.

Valja primijetiti da nije navedeno koje su točno dobi učenici koji su sudjelovali u istraživanju, a samo istraživanje provedeno je tako da su proučavali isključivo kako učenici rješavaju probleme vezane za geometriju. Zato se i rezultati ovog istraživanja trebaju uzeti s dozom opreza.

Također, jedno istraživanje o prijelazu s vizualnog na tekstualni programski jezik provedeno je i u jednoj gimnaziji u Hrvatskoj. Istraživanje je opisano u radu profesora Brođanca [9], međutim rad još nije publiciran. Provedeno istraživanje temeljilo se na korištenju različitih alata za uvod u programiranje i promatranje njihovog utjecaja na nastavak učenja programiranja u programskom jeziku Python. Za istraživanje je korišten Scratch kao blokovski programski jezik te pseudojezik kao tekstualni jezik. Istraživanje je trebalo utvrditi postoji li razlika u transferu znanja iz Scratcha i pseudojezika na programski jezik Python te preferiraju li učenici Scratch kao primjer vizualnog programskog jezika ili pseudojezik kao primjer tekstualnog jezika. U istraživanju je sudjelovalo 50 učenika prvog razreda prirodoslovno-matematičke gimnazije. Nastava se odvijala u dvije faze: uvod u programiranje i programiranje u programskom jeziku Python. Učenici su bili podijeljeni u dvije grupe, eksperimentalnu i kontrolnu. Eksperimentalna grupa je za uvod u programiranje koristila Scratch, a kontrolna grupa pseodujezik. Nakon uvoda u programiranje u trajanju od 8 školskih sati, obje grupe su nastavile nastavu programiranjem u programskom jeziku Python. Nakon uvodna dva sata, učenici su pristupili anonimnoj anketi. Anketa se sastojala od tri dijela: opća pitanja, transfer znanja i pitanja stava o Scratchu/pseudojeziku. Što se tiče tranfera znanja, bolje rezultate postigla je grupa koja je kao uvod u programiranje radila pseudojezik. U pitanjima otvorenog tipa su učenici kao najveću prednost Scratcha

naveli jednostavnost, grafičko sučelje i preglednost naredbi. Sučelje su učenici naveli i kao osnovni nedostatak Scratcha jer ga smatraju predjetinjastim. Glavna prednost pseudojezika također je jednostavnost, ali i to što su naredbe na hrvatskome jeziku.

Provedeno istraživanje je pokazalo da će učenici prvog razreda prirodoslovno-matematičke gimnazije, koji nemaju iskustva ili imaju vrlo malo iskustva u programiranju uspješnije prenijeti znanje iz pseudojezika u Python nego iz Scratcha u Python. Važno je napomenuti da su učenici koji su sudjelovali u istraživanju motivirani za programiranje. Također, sami učenici smatraju da će im za buduće programiranje više koristiti programiranje u pseudojeziku nego u Scratchu. Ovim istraživanjem je pokazano da je, kod učenika koji su motivirani za programiranje, za uvod u programiranje korisnija uporaba tekstualnog nego vizualnog programskog jezika.

I rezultati ovog istraživanja trebaju se promatrati uz dozu opreza jer je u istraživanju sudjelovao relativno mali broj učenika. Ipak, istraživanje je detaljno provedeno, a rezultati će sigurno koristiti nastavniku u budućem radu s učenicima te će mu pomoći da odluči na koji način će učenike uvesti u programiranje.

Poglavlje 3

Karakteristike odabranih programskih jezika

U ovom poglavlju komentirat će se prikladnost programskih jezika Scratch, Logo i Python za poučavanje programiranja u osnovnim školama uzimajući u obzir karakteristike koje je poželjno da programski jezik za poučavanje programiranja ima [14]. Dakle, dat će se kratak pregled kriterija koje zadovoljavaju navedeni programski jezici.

3.1 Učenje programiranja

Programski jezici Scratch, Logo i Python pogodni su za poučavanje. Sva tri programska jezika osmišljena su za učenje programiranja. Scratch je naročito pogodan za poučavanje programiranja kod učenika mlađih uzrasta zbog izgleda sučelja (likovi, naredbe različitih oblika, boja itd.). Također, prednost programskog jezika Scratch je u tome što prilikom programiranja učenici ne moraju znati sami napisati sve naredbe i ne trebaju pamtiti točnu sintaksu već među naredbama odabiru one koje su im potrebne. Iz tog razloga ne može doći do pogrešaka u sintaksi, a istovremeno i učenici mlađih uzrasta nemaju problema s tipkanjem i pisanjem programskog koda jer je pisanje programskih kodova znatno brže nego što je to kod programiranja u drugim programskim jezicima. Još jedna prednost programskog jezika Scratch kao jezika za učenje programiranja jest ta što su naredbe zapisane na blokovima koje su poput slagalice nadopunjuju i na taj način učenici lako pamte kako se, primjerice, postavljaju uvjeti kod naredbi grananja ili kako se ugnježđuje više petlji. Logo i Python su također pogodni za poučavanje, također imaju jednostavnu sintaksu, a nazivi naredbi su jasni. Nema previše kratica, a dodatne opcije su na jasno vidljivim mjestima. Ukoliko učenici naprave pogrešku, dobit će jasno i detaljno objašnjenje što je netočno u njihovom kodu.

Nadalje, programski jezik Scratch odlično povezuje programiranje i pokretanje fizičkih

objekata. U samom grafičkom sučelju učenici mogu, koristeći jednostavne naredbe, uočiti kako izvršavanje tih naredbi utječe na kretanje lika po ekranu. U programskom jeziku Logo učenici, također, korištenjem osnovnih naredbi mogu pokretati kornjaču po ekranu. Prednost oba programska jezika je laka i jasna vizualizacija i neposredno uočavanje kako djeluju naredbe na promatrani objekt. Tako učenici lako mogu uočiti i kako im naredbe grananja i petlje pomažu te čemu služe. Npr. ukoliko učenici žele likom u Scratchu nacrtati kvadrat na ekranu ne moraju 4 puta za redom napisati isto naredbu već mogu koristeći petlju For ili Repeat zadati da se ista naredba (idi naprijed, okreni se za 90 stupnjeva) ponovi četiri puta. Programski jezik Python ne može toliko jasno predočiti učenicima na koji način programiranje utječe na neke uređaje, objekte u stvarnosti koje žele programirati. Bitno je da učenici koji su početnici u programiranju budu svjesni da pisanje programskih kodova ima primjenu u stvarnosti i da nije toliko apstraktno koliko im se isprva može činiti. O tome će biti riječi kasnije u ovom radu. Kako bi i najmlađi učenici shvatili upravo tu primjenu, bitno je naglasiti da se recimo programski jezik Scratch u nešto modificiranom obliku koristi upravo kao osnovni jezik za programiranje fizičkih objekata (robot mBot). Tek programiranjem fizičkih objekata učenici mogu u potpunosti shvatili neke pojmove, poput beskonačne petlje (ako učenici žele konstantno mjeriti temperaturu okoline potrebno je u beskonačnu petlju ugnijezditi naredbu za mjerenje temperature). Python se također može koristiti za programiranje fizičkih objekata. Ipak, sam izgled sučelja programskih jezika Scratch i Logo omogućavaju da pokretanje objekata postane učenicima jasnije.

Sva tri programska jezika omogućuju učenicima da dobiju dojam o tome što je zapravo programiranje. Iako je programski jezik Scratch nešto jednostavniji za korištenje od Loga i Pythona, učenici koriste blokove s naredbama koje koriste i u drugim programskim jezicima. U ovom slučaju, nedostatak programskog jezika Scratch je u tome što učenici ne moraju pamtiti sve naredbe i kojim redoslijedom ih je potrebno pisati. Ipak, nakon što učenici dulje vrijeme rješavaju probleme programiranjem u Scratchu, zapamte nazive naredbi i sintaksu. Budući da su Scratch, Logo i Python i osmišljeni kao programski jezici za poučavanje programiranja, oni koriste naredbe i neke implementirane funkcije koje se koriste i u drugim programskim jezicima u kojima će učenici možda jednog dana programirati. Ipak, Scratch, Logo i Python su jednostavniji i pregledniji što je neizbježan kriterij koji programski jezik za poučavanje programiranja treba zadovoljiti. Ipak, omogućavaju da učenici nauče osnove i temeljna načela programiranja, što im je izvrstan temelj za kasnije programiranje u drugim programskim jezicima. Teži se tomu da učenici tijekom svog obrazovanja ne mijenjaju previše okruženja za programiranje [10] i upravo to se i postiglo s programskim jezikom Python. Programski jezici Scratch i Logo su ipak primjereniji za rješavanje manje kompleksnih problema.

Scratch, Logo i Python su metodički osmišljeni programski jezici. Zato nisu ograničeni samo na učenje sintakse već omogućavaju učenicima da koriste i više od osnovnih naredbi. Scratch ima primjenu i u drugim nastavnim predmetima, naročito u predmetima iz priro-

doslovnog područja, kao što je ranije opisano. Logo i Python na jednostavan način učenike potiču da kreiraju nove funkcije koje će zapravo biti sastavljene od niza osnovnih naredbi.

3.2 Dizajn i okruženje

Programski jezici za poučavanje programiranja trebaju zadovoljavati određene kriterije vezane za dizajn grafičkog sučelja i okruženje u kojem učenici programiranju. Što se tiče Scratcha, on je posebno privlačan učenicima mlađeg uzrasta. Zbog intuitivnog sučelja, naredbi koje su razvrstane u skupine i označene različitim bojama te blokova s naredbama koje svojim oblicima sugeriraju redoslijed naredbi, Scratch je jednostavan za upotrebu i lako shvatljiv početnicima. Interakcija s učenicima nije toliko izražena jer ne dolazi do javljanja o pogreškama (nije moguće pogriješiti u sintaksi), osim što učenici neposredno mogu dobivati povratnu informaciju, tj. vidjeti rezultat onoga što su programirali. Logo i Python nemaju toliko privlačno sučelje jer se ono svodi na bijelu podlogu i crna slova. Ipak, jasno je i učenici ne bi trebali imati problema pri pronalaženju onoga što im je potrebno (dodatnih alata, knjižnica itd.). Također, sučelje programskog jezika Logo sastoji se i od ekrana na kojem se može pratiti kretanje kornjače. Logo i Python su izuzetno interaktivni programski jezici jer pogreške javljaju na učenicima vrlo jasan način. Detaljno opisuju gdje je pogreška. Na neki način, programski jezik pri javljaju da je došlo do pogreške direktno komunicira s programerom jer programiranje i javljanje da je došlo do pogreške funkcionira kao svojevrsni dijalog. Programski kod se razvija relativno brzo u svim promatranim programskim jezicima. U Scratchu učenici uzimaju one blokove s naredbama koji su im potrebni i slažu ih u željenom redoslijedu (i dopuštenom redoslijedu). S druge strane, naredbe koje učenici koriste programirajući u Logu i Pythonu su naredbe koje se koriste i u većini drugih programskih jezika. Pomoć pri pisanju programskog koda (*Help*) je dostupna i sadrži popis svih naredbi i sintaksu, što može uvelike pomoći učenicima. Budući da su programski jezici intuitivni i laki za korištenje, čak i početnicima, oni djeluju motivirajuće na učenike jer neće morati učiti nove naredbe i njihovu sintaksu kako bi napisali svoj prvi program. Također, daju brzu povratnu informaciju o učeničkom radu. Naredbe se izvršavaju neposredno nakon što su ih učenici zapisali te na taj način oni imaju priliku odmah uočiti što je rezultat njihovog rada. To je naročito jasno i vidljivo u programskim jezicima Scratch i Logo gdje učenici mogu vidjeti kako se kreće lik/kornjača ovisno o naredbama koje su napisali.

Sljedeći kriterij jest da programski jezici potiču učenike da pišu ispravne programske kodove. Logo i Python u potpunosti zadovoljavaju taj kriterij. Oba programska jezika učenicima daju neposrednu povratnu informaciju i upozoravaju ih ako su učinili pogrešku prilikom pisanja programskog koda. Na taj način učenici uviđaju gdje su pogriješili i uče samostalno napisati ispravne programske kodove koji će istovremeno rješavati zadani problem. S druge strane, Scratch ne daje povratnu informaciju o tome jesu li učenici ispravno

napisali programski kod. Iako je nemoguće da učenici naprave pogrešku u sintaksi, ovakav način pisanja programskog koda se temelji na pokušajima i pogreškama što se želi izbjeći dok učenici uče programirati. Ako pišu svoje programe na temelju pokušaja i pogrešaka tada oni ne znaju u čemu su zapravo pogriješili nego samo znaju da njihov program ne radi ili ne radi ono što treba raditi. Ipak, ako program funkcionira na način da radi upravo ono što je zadano kako bi se riješio problem, onda učenici znaju kako izgleda ispravno napisan programski kod. Metodički, ispravnija je komunikacija programskih jezika Logo i Python s učenicima jer ih upozorava na pogreške i potiče ih da zaista znaju napisati ispravni programski kod.

Prilikom rješavanja problema, učenici trebaju moći podijeliti programski kod na manje smislene cjeline tako da modeliraju pomoću funkcija, procedura itd. U Scratchu učenici ne mogu jasno uočiti da se program sastoji od manjih dijelova jer se svaki program sastoji od niza blokova s naredbama koji čine jedan veliki blok, tj. cijeli program. Programski jezici Logo i Python omogućavaju učenicima da, koristeći osnovne naredbe, stvaraju nove funkcije. Te nove funkcije koristit će kako bi riješili neki zadani problem. Na taj način učenici shvaćaju hijerarhijsku strukturu dijelova programa te sami povezuju manje dijelove u jednu veliku smislenu cjelinu.

Nadovezujući se na prethodni kriterij, učenicima je potrebno omogućiti da shvate razliku između ideje za rješavanje problema i samog programskog koda koji je upravo rješenje problema. Tako učenici pišući kod u programskim jezicima Logo i Python prvo osmišljavaju ideju za rješavanje problema i uviđaju koje će im funkcije biti potrebne, nakon čega trebaju zaključiti kako će napisati željene funkcije koristeći osnovne naredbe. U Scratchu učenici ne pišu nove funkcije koje koriste za rješavanje problema, već kad osmisle ideju, oni grade programski kod tako da stavljaju blokove s naredbama na željena mjesta. U ovom slučaju su blokovi s naredbama korisni jer dobro vizualno pokazuju učenicima gdje se nalazi koji dio koda, ali ih također mogu i zbuniti jer grafičko sučelje ne omogućava učenicima da jasno vide od kojih se dijelova sastoji njihov programski kod.

3.3 Podrška i dostupnost

Kao što je ranije istaknuto, bitno je da postoje lako dostupni materijali za učenje i poučavanje, kao i da je prisutna podrška nastavnicima i učenicima prilikom učenja. Također, sam programski jezik treba biti lako dostupan nastavnicima, školama i učenicima.

Bilo da se radi o učenju u školi, samostalnom učenju učenika kod kuće, materijalima za poučavanje i učenje, bitno je da postoji podrška za učenje (poučavanje) programiranja u određenom programskom jeziku. Scratch samo na svojim *web*-stranicama nudi pregršt materijala za učenje, ali i nastavnih materijala. Postoji i ScratchEd kutak za nastavnike koji omogućava nastavnicima da lako dijele svoja iskustva u radu s učenicima, nastavne materijale te da si međusobno pomažu u rješavanju problema. Također, organiziraju se

druženja nastavnika koji poučavaju učenike programiranju u Scratchu. Nadalje, postoji kutak za učenike i kutak za roditelje. Stranica je ažurna i učenici koji zaista žele naučiti programirati u Scratchu imaju mogućnost koristiti se materijalima bez registracije. Što se tiče podrške učenicima, roditeljima i nastavnicima, kreatori Scratcha o tome vode računa. Međutim, na drugim forumima i *web*-stranicama ima puno manje materijala za učenje programiranja u Scratchu nego u jezicima Logo i Python, a i udžbenici iz informatike za osnovne škole u Hrvatskoj su više orijentirane na poučavanje programiranja u Logu i Pythonu. Budući da se programski jezik Python koristi i za pisanje programskih kodova u poslovne svrhe te se pišu kompleksni programi, na forumima je dostupno puno materijala i mogu se pronaći odgovori na pitanja te rješenja raznih problema s kojim su se osobe susretale programirajući u Pythonu.

Nadalje, programski jezici za poučavanje programiranja trebaju biti dostupni javnosti za korištenje, uvid, izmjene i daljnje poboljšanje. Scratch na svojim *web*-stranicama ima kutak za programere koji žele dodatno usavršavati i poboljšavati programski jezik Scratch. Također, programski jezici Logo i Python dozvoljavaju razvojnim programerima da daju prijedloge za poboljšanje i rad na izvornom kodu. Kao što je već napomenuto, sva tri navedena programska jezika su osmišljena za poučavanje programiranja, proizvod su grupe ljudi i besplatni su pa je angažman zainteresiran osoba za poboljšanjem izvornog koda i te kako dobrodošao.

Scratch, Logo i Python funkcioniraju u različitim okruženjima, tj. dostupne su verzije spomenutih programskih jezika u kojima je moguće raditi u različitim operacijskim sustavima (Windows, Linux, Apple). Također, funkcioniraju i na različitim uređajima. Dakle, postoje verzije Scratcha, Loga i Pythona prilagođene za rad na osobnim računalima, tabletima i pametnim telefonima. To je bitno jer u današnje vrijeme se učenici sve više koriste tabletima, a mogućnost pisanja programskog koda na tabletima znatno im olakšava učenje. Budući da učenici posjeduju tablete i pametne telefone, ali i računala, od različitih proizvođača i s instaliranim različitim operacijskim sustavima, odlično je da postoje verzije promatranih programskih jezika koje su napravljene za rad u različitim operacijskim sustavima i na različitim uređajima. Ponekad se izgled sučelja može nešto razlikovati, ali učenici ne bi trebali imati poteškoća pri prelasku s jednog uređaja na drugi ili pri prelasku s jednog operacijskog sustava na drugi.

Sljedeći kriterij koji bi programski jezici za poučavanje programiranja trebali zadovoljavati jest da su lako dostupni svugdje u svijetu te da su besplatni. Scratch, Logo i Python su dostupni u cijelom svijetu. Potrebno ih je preuzeti s internetskih stranica i instalirati na uređaj. Postoje i online verzije programskih jezika u kojima je moguće raditi bez prethodne instalacije na uređaj (računalo, tablet, pametni telefon). Postoje i materijali za poučavanje i učenje programiranja u navedenim programskim jezicima. Na *web*-stranici programskog jezika Scratch postoje dobro osmišljeni i metodički napisani materijali za poučavanje programiranja u tom programskom jeziku, ali i za samostalno učenje programiranja. Postoje i

materijali za poučavanje programiranja u Logu i Pythonu, ali materijali na službenih stranicama tih programskih jezika nisu toliko detaljno i metodički obrađeni.

U udžbenicima koje je propisalo Ministarstvo znanosti i obrazovanja obrađuju se programski jezici Logo i Python [2], ali ne i programski jezik Scratch. To je bitan kriterij koji u ovom slučaju ne zadovoljava programski jezik Scratch. Naime, ne postoje službeni nastavni materijali prilagođeni za poučavanje programiranja u programskom jeziku Scratch. U nekim školama, učenici svejedno uče programirati i pišu svoje prve programske jezike u Scratchu na nastavni informatike. Također, u nekim izvannastavnim i izvanškolskim aktivnostima učenici također uče osnovne pojmove iz programiranja uz programski jezik Scratch.

3.4 Druge karakteristike

Kao što je već ranije istaknuto, nije praktično niti ekonomično da za poučavanje učenika koji su na višoj razini znanja prelazi na poučavanje u drugom programskom jeziku. Programski jezik Python svakako može biti programski jezik za poučavanje programiranja na bilo kojoj razini znanja, od mlađih učenika do poučavanja programiranja na fakultetskoj razini. Sam programski jezik koristi se za programiranje u vodećim svjetskim informatičkim kompanijama. Ipak, dovoljno je jednostavan i intuitivan da bi i početnici mogli usvojiti osnovne pojmove i koncepte programiranja programirajući u tom programskom jeziku. Iako je programski jezik Scratch bolje prilagođen poučavanju mlađih učenika (od 1. do 4. razreda osnovne škole), učenici koji se žele nastaviti baviti programiranjem i rješavati kompleksnije probleme bit će ograničeni mogućnostima koje im pruža programiranje u Scratchu. Isto tako, stariji učenici mogu i samo sučelje programskog jezika Scratch smatrati suviše djetinjastim zbog šarenila i boja. Učenici koji su dobro svladali programiranje i pisanje programskih kodova mogu biti sputani načinom na koji funkcionira programiranje u Scratchu jer su im programski jezici koji im dopuštaju da sami tipkaju kod brži i jednostavniji za korištenje. Programski jezik Logo je, u ovom slučaju, između programskih jezika Scratch i Python. Dovoljno je jednostavan i intuitivan, sučelje i jednostavno, a i kornjača olakšava učenicima shvaćanje nekih naredbi. Opet, na višim razinama učenici mogu naići na kompliciranije probleme te će im pisanje programskih kodova u Logu biti predugačko i presloženo jer je u Pythonu ipak implementirano više funkcija koje mogu koristiti pri rješavanju kompleksnih problema.

Upravo zbog toga, jedino Python omogućava da učenici koji su napredni pišu programske kodove kojima će vidjeti stvarnu svrhu programiranja jer se programski jezik Python koristi u mnoge druge svrhe. Ipak, programski jezik Scratch često se koristi i kao osnovni programski jezik za programiranje fizičkih objekata. Neki učenici na višim razinama (ali ponekad i u osnovnim školama) požele napisati programski kod koji će služiti nekoj kompaniji i za koji mogu biti nagrađeni. Tada je od navedena tri programska jezika

najprikladniji programski jezik Python, koji je od svih programskih jezika za poučavanje programiranja kojima se trenutno koriste nastavnici u osnovnim školama u Hrvatskom jedini primjenjiv i za programiranje u komercijalne svrhe. Ukoliko učenici požele napisati programski kod koji će im poslužiti za nastavu iz nekog drugog predmeta tada im opet najviše mogućnosti pruža Python koji ima implementirane funkcije korisne u raznim područjima. Ipak, i programski jezici Scratch i Logo također se mogu koristiti kao nastavna sredstva u nastavi drugih predmeta. U tom slučaju dolazi do korelacije s drugim nastavnim predmetima čemu se u posljednje vrijeme teži. Učenici tada znanja stečena na nastavi iz jednog predmeta koriste kako bi oplemenili nastavu iz ostalih predmeta.

Programski jezik Python je najbrži za izvođenje programa od svih promatranih programskih jezika. Zato je Python najviše primjenjiv za rješavanje kompleksnijih problema kojima će se baviti učenici zainteresirani za programiranje. Ostali učenici vjerojatno neće niti primijetiti razliku u brzini izvođenja programa na jednostavnijim primjerima. Ponekad nije loše da nastavnici učenicima ukažu na razliku u izvođenju programa ovisno o programskom jeziku u kojem pišu programski kod i ovisno o naredbama koje koriste. To je još jedan od načina kako učenici uče pisati ispravne programske kodove i izabrati primjereni programski jezik za rješavanje problema.

3.5 Zadovoljeni kriteriji

Poželjne karakteristike programskih jezika za poučavanje programiranja, po mišljenju kreatora jezika za poučavanje programiranja, trebaju imati svi programski jezici koji se koriste za poučavanje programiranja u školama. Programski jezici Python i Logo zadovoljavaju najviše navedenih kriterija. Zadovoljeni kriteriji prikazani su u Tablici 3.5.

Jedan od osnovnih kriterija koje ne zadovoljava programski jezik Scratch je nepostojanje nastavnih materijala za poučavanje programiranja. Ipak, dostupni su mnogi materijali na samim *web*-stranicama programskog jezika Scratch. To, međutim, nije dovoljno jer se nastavnici ne mogu osloniti na to da svi učenici imaju pristup internetu niti da dovoljno razumiju engleski jezik, na kojem je većina materijala. Iako zadovoljava sve kriterije za poučavanje najmlađih učenika, oni s vremenom prijeđu na višu razinu programiranja kad im Scratch više ne pruža dovoljno mogućnosti. Programski jezici Logo i Scratch su kompliciraniji u početku jer učenici trebaju pamtiti sintaksi naredbi i trebaju sami tipkati, dok su Scratchu mogu birati blokove s naredbama koje su im u tom trenutku potrebne. To čini Scratch iznimno jednostavnim za korištenje, isto kao i sučelje koje je prilagođeno najmlađim učenicima. Ali upravo to što je najveća prednost za početnike kasnije postaje mana za naprednije učenike.

Nadalje, programski jezik Logo prilagođen je poučavanju mlađih učenika, ali već trebaju razumjeti neke pojmove na engleskom jeziku te trebaju naučiti sintaksu osnovnih naredbi. U početku učenici mogu imati poteškoća sa samim tipkanjem i pisanjem programa,

ali to ih priprema za kasnije pisanje kompliciranijih programskih kodova i rješavanje kompleksnijih problema. Također, Logo omogućava jednostavnu podjelu programa na manje dijelove i kasnije spajanje tih dijelova u smislenu cjelinu. Tako učenici najbolje shvaćaju kako se gradi program od početka do kraja te uočavaju hijerarhiju u dijelovima programa. S druge strane, u sučelju programskog jezika Logo učenici mogu pratiti kretanje kornjače koje će im u početku pomoći da shvate način na koji računalo (kornjača) reagira na naredbe koje učenici daju.

	Učenje	**Dizajn i okruženje**	**Podrška i dostupnost**	**Programiranje**
Scratch	- pogodan za poučavanje - jednostavna sintaksa i prirodna semantika - fizička analogija - novi pristup poučavanju	- jednostavno okruženje - povratna informacija o radu	- podrška korisnicima - dostupan izvorni kod - rad na različitim platformama - besplatan i lako dostupan	
Logo	- pogodan za poučavanje - jednostavna sintaksa i prirodna semantika - fizička analogija - općenita slika o programiranju - novi pristup poučavanju	-interaktivno i jednostavno okruženje - povratna informacija o radu - ispravni programi - mogućnost podjele koda na manje dijelove - razlika između ideje i programskog koda	- podrška korisnicima - dostupan izvorni kod - rad na različitim platformama - obrađen u udžbeniku - besplatan i lako dostupan	- različite razine znanja
Python	- pogodan za poučavanje - jednostavna sintaksa i prirodna semantika - općenita slika o programiranju - novi pristup poučavanju	- interaktivno i jednostavno okruženje - povratna informacija o radu - ispravni programi - mogućnost podjele koda na manje dijelove - razlika između ideje i programskog koda	- podrška korisnicima - dostupan izvorni kod - rad na različitim platformama - obrađen u udžbeniku - besplatan i lako dostupan	- različite razine znanja - programiranje u druge svrhe - kompliciraniji programi

Tablica 3.1: Zadovoljeni kriteriji

Programski jezik Python omogućava učenicima da u istom okruženju pišu svoje prve programe, ali i programe na višim razinama obrazovanja, a možda i na radnom mjestu. Iako je Python najkompliciraniji za shvatiti od svih navedenih programskih jezika, pruža najviše mogućnosti za učenike koji žele rješavati kompliciranije probleme. Python je također pregledan, čitljiv i jednostavan za korištenje, a poznavanje osnovnih pojmovi koje učenici usvoje učeći programirati u Pythonu mogu primijeniti na programiranje u bilo kojem drugom programskom jeziku.

Kako bi učenici još bolje razumjeli kod, poželjno je da učenici razumiju engleski jezik. Naredbe u programskim jezicima Logo i Python su na engleskom jeziku (ili su skraćenice engleskih riječi). Budući da je programski jezik Scratch lokaliziran na hrvatski jezik, učenicima je samo značenje naredbi jasnije. Ipak, učenicima poznavanje engleskog jezika ne bi trebalo biti ključan faktor i predstavljati problem u programiranju. Jedino bi učenicima u nižim razredima osnovne škole pisanje programskih kodova mogle olakšati naredbe u Scratchu prevedene na hrvatski jezik.

Poglavlje 4

Zaključak

Programski jezici Scratch, Logo i Python koriste se za poučavanje programiranja u osnovnim školama u Hrvatskoj. Iako se u udžbenicima iz informatike za osnovne škole obrađuju programski jezici Logo i Python, smatram da je i programski jezik Scratch pogodan za poučavanje programiranja. Ipak, programski jezik Scratch primjeren je poučavanju mlađih učenika, ali može služiti za pojašnjenje gradiva i starijim učenicima.

Uzevši u obzir rezultate provedenih istraživanja i karakteristike navedenih programskih jezika, zaključuje se da je za učenike osnovnih škola najbolji način poučavanja programiranja kombinacijom vizualnih i tekstualnih programskih jezika. Neki od učenika će se zainteresirati za programiranje te će htjeti pisati kompliciranije programske kodove, zato im je potrebno omogućiti da što lakše savladaju tekstualne programske jezike. Nastava treba biti prilagođena svim učenicima pa je korištenje vizualnih programskih jezika odličan način za objasniti pojedine dijelove gradiva. Svakako, ne smije se zanemariti niti to da programski jezik u kojem učenici uče programirati mora biti obrađen u udžbenicima koje učenici koriste.

Bibliografija

[1] *EDU-SIG: Python in Education*, https://www.python.org/community/sigs/current/edu-sig/.

[2] *Ministarstvo znanosti i obrazovanja*, https://mzo.hr/.

[3] *ScratchEd*, http://scratched.gse.harvard.edu/.

[4] *Scratch*, https://scratch.mit.edu.

[5] *TIOBE Index - The Software Quality Company*, https://www.tiobe.com/tiobe-index/l.

[6] *What is Logo?*, http://el.media.mit.edu/logo-foundation/what_is_logo/index.html.

[7] M. Berry, *Computing in the national curriculum - A guide for primary teachers*, (2013), http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf.

[8] I. Boljat, *Poučavanje programiranja početnika pomoću Alice – utjecaj na motivaciju i izbor karijere*, Život i škola : časopis za teoriju i praksu odgoja i obrazovanja (2014).

[9] P. Brođanac, *Uvod u programiranje: pseudojezik ili Scratch?*, (2016).

[10] P. Brođanac, Z. Markučič, L. Budin i S. Perić, *Python - Jezik za podučavanje algoritamskog pristupa rješavanju problema*, (2008), http://www.mipro.hr/LinkClick.aspx?fileticket=Fq7zZKz0nZw%3D&tabid=134&language=hr-HR.

[11] M. Dorling i D. White, *Scratch: A Way to Logo and Python*, (2015), https://pdfs.semanticscholar.org/00bd/b2fda5f662a0efdd217d22b2ebeeb72b3e53.pdf.

[12] P. Kemp, *Computing in the national curriculum - A guide for secondary teachers*, (2014), http://www.computingatschool.org.uk/data/uploads/cas_secondary.pdf.

[13] C. M. Lewis, *How Programming Environment Shapes Perception, Learning and Goals: Logo vs. Scratch*, Proceedings of the 41st ACM technical symposium on Computer science education (2010), http://ims.mii.lt/ims/konferenciju_medziaga/SIGCSE'10/docs/p346.pdf.

[14] L. Mannila i M. de Raadt, *An Objective Comparison of Languages for Teaching Introductory Programming*, Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling (2006), https://pdfs.semanticscholar.org/dc53/91d339571d7914f2fddf18ce725d64b9abb1.pdf.

[15] T. Slavin, *How to Choose Your First Programming Language*, Kids, Code and Computer Science (2013), https://www.kidscodecs.com/how-to-pick-a-programming-language/.

Sažetak

U radu su navedene i pojašnjene poželjne karakteristike jezika za poučavanje programiranja. Uspoređeni su programski jezici Logo, Scratch i Python. Također, argumentirano je jesu li primjereni za poučavanje programiranja u osnovnoj školi, što postaje sve važniji dio nastave informatike. Učenje programiranja se nastoji popularizirati među učenicima, a najviše im problema zadaje razvijanje algoritamskog načina mišljenja. Nastavnicima probleme zadaje odabir programskog jezika jer programski jezik je jedan od ključnih čimbenika koji utječu na to hoće li se učenici zainteresirati za programiranje. Programski jezik bi trebao biti jednostavan i primjeren učenicima osnovnih škola. Prema važećem Katalogu obveznih udžbenika i pripadajućih dopunskih sredstava za osnovnu školu, za poučavanje programiranja u osnovnim školama se koriste programski jezici Logo, Python i QBasic. Na natjecanjima se koriste i programski jezici Basic, C, C++ i Pascal. U nekim školama, učenici uče programirati u programskom jeziku Scratch. Rad se sastoji od tri poglavlja. U prvom poglavlju su opisane poželjne karakteristike za poučavanje programiranja te su opisani programski jezici Scratch, Logo i Python. Drugo poglavlje sadrži rezultate istraživanja o utjecaju programskog jezika na motivaciju učenika, na percepciju, učenje i ishode učenja te rezultate istraživanja o prijelazu s vizualnog na tekstualni programski jezik. Treće poglavlje prikazuje koje kriterije zadovoljavaju promatrani programski jezici te imaju li poželjne karakteristike jezika za poučavanje programiranja.

Summary

In this thesis it is described which are desirable characteristics of the programming language for the teaching of programming. Programming languages Logo, Scratch and Python are compared. Also, it is explored if they are suitable for teaching programming elementary school, which becomes an increasingly important part of computer science curriculum. Programming is trying to be popularized among students and the biggest problem is learning computational thinking. Teachers are faced with many problems, one of them is choosing a programming language because it is one of the key factors for students to become interested in programming. The programming language should be simple and appropriate for elementary school students. According to the relevant Catalogue of compulsory textbooks and related supplementary materials for elementary schools (*hrv. Katalogu obveznih udžbenika i pripadajućih dopunskih sredstava za osnovnu školu*), programming languages Logo, Python and QBasic are used to teach programming in elementary schools. Also, students use programming languages C, C++ and Basic on competitions. In some schools, students learn to program in the Scratch programming language. This paper consists of three chapters. The first chapter describes desirable features of languages for teaching programming and describes Scratch, Logo and Python programming languages. The second chapter contains the results of the studies about the influence of the programming language on student's motivation, perception, learning and learning outcomes and the results of research on the transition from visual to textual programming language. The third chapter shows which criteria meet the observed programming languages.

Životopis

Rođena sam 19. veljače 1991. u Zagrebu. Školovanje sam počela u Osnovnoj školi kralja Tomislava nakon čega sam upisala V. gimnaziju u Zagrebu. Maturirala sam 2009. godine s odličnim uspjehom. Preddiplomski sveučilišni studij Matematika; smjer: nastavnički, na Prirodoslovno-matematičkom fakultetu Sveučilišta u Zagrebu, završila sam 2015. godine te sam iste godine upisala diplomski sveučilišni studij Matematika i informatika, smjer: nastavnički. Tijekom studija sam bila demonstratorica iz kolegija Elementarna geometrija. Metodičku praksu iz matematike i informatike odradila sam u Osnovnoj školi Otona Ivekovića te u XV. gimnaziji u Zagrebu.

Posjedujem certifikate o znanju engleskog (Cambridge Certificate in Advanced English iz 2009. godine i Cambridge Business English Certificate Higher iz 2015. godine) i njemačkog jezika (Goethe-Zertifikat C1 iz 2009. godine).