Programiranje u osnovnoj školi pomoću besplatnih alata na primjeru Pythona

Turčin, Dora

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Teacher Education / Sveučilište u Zagrebu, Učiteljski fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:147:098374

SVEUČILIŠTE U ZAGREBU

UČITELJSKI FAKULTET

ODSJEK ZA UČITELJSKE STUDIJE

DORA TURČIN

DIPLOMSKI RAD

PROGRAMIRANJE U OSNOVNOJ ŠKOLI POMOĆU BESPLATNIH ALATA NA PRIMJERU *PYTHONA*

Zagreb, rujan 2020.

SVEUČILIŠTE U ZAGREBU

UČITELJSKI FAKULTET

ODSJEK ZA UČITELJSKE STUDIJE

(Zagreb)

DIPLOMSKI RAD

Ime i prezime pristupnika: Dora Turčin

TEMA DIPLOMSKOG RADA: Programiranje u osnovnoj školi pomoću besplatnih alata na primjeru Pythona

MENTOR: Izv. prof. dr. sc. Predrag Oreški

Zagreb, rujan 2020.

ZAHVALA

Zahvaljujem svom mentoru izv. prof. dr. sc. Predragu Oreškom na vodstvu, podršci i strpljenju tijekom pisanja mog diplomskog rada.

Zahvaljujem svojoj obitelji; mami Ružici, tati Srećku, sestrama Matei i Ivi, bratu Dominiku, zaručniku Filipu te široj rodbini, na velikoj podršci i ljubavi koju su mi pružali tijekom cijeloga moga obrazovanja.

Zahvaljujem svojim nećacima Antonu i Tomi te kumčetu Magdaleni koji su, unatoč tome što su djeca, uspjeli pridonijeti mom akademskom rastu i mojoj motivaciji za daljnjim usavršavanjem u svom pozivu.

Zahvaljujem svojim prijateljima koji su bili uz mene tijekom studija, bodrili me te podržavali u stjecanju kompetencija za moju buduću profesiju, odnosno moj poziv.

Naposljetku želim zahvaliti Bogu koji mi je dao snagu da nakon svih nedaća, koje su me ove 2020. godine zadesile, ne posustanem u svom naumu.

Tada Isus reče svojim učenicima: »Hoće li tko za mnom, neka se odrekne samoga sebe, neka uzme svoj križ i neka ide za mnom.« Mt, 16-24

SADRŽAJ

SAŽETAK

U današnje se vrijeme informacijske i komunikacijske tehnologije sve više i brže razvijaju, a samim time sve više ulaze u živote ljudi, pa tako mijenjaju i način shvaćanja svijeta u kojem ljudi žive. Opće je poznato da u 21. stoljeću većina zanimanja zahtijeva razumijevanje i primjenu informacijskih i komunikacijskih tehnologija radi veće produktivnosti i efikasnosti. Djeca suvremenog svijeta sve su više izložena informacijskim i komunikacijskim tehnologijama, budući da se rađaju u takvom vremenu i okruženju. Stoga bi valjalo informacijske i komunikacijske tehnologije u školama integrirati na način da učenici uče kako ih učinkovito koristiti. U Hrvatskoj je na snagu stupila nova kurikularna reforma kojom bi se informatika trebala uvesti kao obavezan nastavni predmet u osnovne škole već od 1. razreda. Nastavni predmet Informatika u hrvatskim školama podrazumijeva četiri domene: digitalnu pismenost i komunikaciju, računalno razmišljanje i programiranje, informacije i digitalnu tehnologiju te e-društvo. Smatra se da težište odgojno-obrazovnog procesa u nastavi Informatike treba biti postavljeno na rješavanje problema i programiranje, zato što bi se time poticalo razvijanje računalnog, odnosno apstraktnog, načina razmišljanja koje omogućava razumijevanje, analizu i rješavanje problema, što može biti iznimno korisno u ostalim područjima, pa tako i u svakodnevnom životu. Kao što je u svakodnevnom životu korisno razumijevanje i primjena osnovnih alata za rad na računalu, tako je korisno i poznavanje osnova programiranja, zato što ono potiče razvoj stvaralaštva, inovativnosti te poduzetnosti, a samim time i usvajanje vrijednih znanja koje mogu pomoći i u profesionalnome razvoju. Budući da je danas velik izbor programskih jezika, često je teško odabrati najpogodniji za početno učenje programiranja, posebice u osnovnoškolskom obrazovanju. Programski jezik *Python* iznimno je jednostavan za početnike radi svoje jednostavne sintakse, iako podržava objektno-orijentirano programiranje. Važno je istaknuti i da je to programski jezik s iznimno širokim područjem primjene, što u nastavi Informatike omogućava korelaciju sa ostalim nastavnim predmetima, čime se učenike može dodatno motivirati.

Ključne riječi: *programiranje, računalno razmišljanje, Python*

SUMMARY

Nowadays, information and communication technologies are developing more and more rapidly, and as a result, they are increasingly entering the lives of people, thus changing the way people understand the world they live in. It is widely known that in the 21st century, most professions require the understanding and implementation of information and communication technologies for greater productivity and efficiency. Children of the modern world are increasingly exposed to information and communication technologies as they are born in such a time and environment. Therefore, information and communication technologies in schools should be integrated so that students learn how to use them effectively. A new curricular reform has been put into effect in Croatia, which should introduce computer science as a compulsory subject in primary schools since the 1st grade. Teaching subject Informatics in Croatian schools covers four domains: digital literacy and communication, computer thinking and programming, information and digital technology, and e-society. It is considered that the focus of the educational process in the teaching of Informatics should be placed on problem solving and programming, because this would encourage the development of computer, or abstract way of thinking that enables understanding, analysis, and problem solving, which can be extremely useful in other areas, including everyday life. Competence in basic programming is useful in everyday life, as well as understanding and implementation of basic computer tools, because it raises development of creativity, innovation and busyness which leads to acquisition of knowledge useful for professional development. Due to the existence of large selection of programming languages, nowadays is often difficult to choose the most suitable one for beginner programming learning, especially in elementary education. Programming language *Python* is extremely simple for beginners because of its simple syntax, although it supports object-oriented programming. It is also important to emphasize that this is a programming language with an extremely wide field of implementation, which enables in the teaching of Informatics correlation with other subjects, and it can further motivate students.

Key words: *programming, computer thinking, Python*

1. UVOD

U odabiru teme diplomskoga rada prevagnulo je područje koje me najviše zanima i smatram ga važnim za svoj budući posao, s obzirom na to da završavam Učiteljski studij s modulom informatika. Stoga sam odabrala temu iz informatike.

Današnja su djeca od sve ranije dobi podložna tehnologiji, sve je više konzumiraju. Budući da sam tijekom studija određeni broj sati provela u školama, u okruženju djece, uočila sam koliko zapravo djeca ne provode vrijeme igrajući se vani, već igrajući igrice na mobitelima. Zato je nastavni predmet Informatika važan kako bi učenici spoznali informatičke pojmove, naučili kako se efikasno i produktivno tehnologije mogu koristiti u razne svrhe, a ne samo igrajući igrice.

U hrvatskim se osnovnim školama nastavni predmet Informatika do 2018. godine izvodio isključivo kao izvannastavna aktivnost. Donošenjem novog kurikuluma informatike 2018. godine, od školske godine 2018./2019. Informatika je postala obvezan predmet samo u 5. i 6. razredu osnovne škole, a u 7. i 8 razredu izborni predmet. Informatika će postati izboran predmet od 1. do 4. razreda osnovne škole, no tek od školske godine 2020./2021. (MZO, 2018).

Na mrežnim stranicama Škole za život (2019) navodi se kako su prije samog donošenja novog kurikuluma, za učitelje i nastavnike održane brojne edukacije 2018. godine, a za one koji u njima nisu sudjelovali u prosincu iste godine otvorena je virtualna učionica Informatika CKR (Cjelovita kurikularna reforma) II. U njoj su se ponovile teme važne za odgojno-obrazovni rad učitelja i nastavnika informatike. Ista je učionica zatvorena 14. lipnja 2019. godine.

Kurikulum nastavnoga predmeta Informatika sadržava odgojno-obrazovne ishode, razradu ishoda, razine usvojenosti i preporuke za ostvarivanje odgojno-obrazovnih ishoda po razredima i domenama, prikaz godišnjeg broja sati i oblik izvođenja nastavnog predmeta Informatika u osnovnoj školi i gimnazijama te popis preporučenih kvalifikacija za učitelje i nastavnike Informatike. U kurikulumu je detaljno opisan nastavni predmet Informatika. U obrazovnom sustavu pojam Informatika podrazumijeva digitalnu pismenost, e-učenje te rješavanje problema i programiranje (MZO, 2018).

Predmetni kurikulum Informatike svoje ciljeve realizira kroz četiri domene: e-Društvo, Digitalna pismenost i komunikacija, Računalno razmišljanje i programiranje te Informacije i digitalnu tehnologiju. Težište odgojno-obrazovnog procesa Informatike treba biti postavljeno na rješavanje problema i programiranje. Time bi se poticalo razvijanje računalnog načina razmišljanja koje omogućava razumijevanje, analizu i rješavanje problema. Takav način razmišljanja, osim što je koristan u području informatike, važan je i koristan u svakodnevnom životu. Stoga ću se u ovom radu baviti time što je programiranje, računalno razmišljanje te kako programiranje u Pythonu primijeniti u nastavi Informatike.

2. PROGRAMIRANJE I RAČUNALNO RAZMIŠLJANJE

„Programiranje je rješavanje problema, otklanjanje grešaka, razvijanje logičkog razmišljanja i računalnog razmišljanja“ (Bubica, Mladenović, Boljat, 2013: 1). Ono podrazumijeva alat pomoću kojeg se potiče razvoj učenikovih metakognitivnih sposobnosti, koje čine temelj uspješnosti u obrazovanju. Programiranje omogućava razvoj samopouzdanja, preciznosti u ispravljanju pogrešaka, upornosti, sposobnosti komunikacije te zajedničkog rada. Bitno je istaknuti da programiranje ne podrazumijeva samo rješavanje problema, već iziskuje ovladavanje višim vještinama, kao što je uočavanje i ispravljanje pogrešaka („bug-ova“) zbog kojih programi ne funkcioniraju kako bi trebali. Tijekom samog programiranja valja voditi brigu o tome je li program popravljiv, a ne je li on dobar ili nije (Bubica, Mladenović, Boljat, 2013).

Većina autora stručne literature navodi da je računalno razmišljanje misaona aktivnost formuliranja i rješavanja problema na način kako bi to izvelo računalo, iako ne nužno uz pomoć računala (Tomljenović, 2018). Računalno razmišljanje sadrži velik broj karakteristika, a one najznačajnije su: formuliranje problema uz mogućnost korištenja računala i drugih alata, logičku organizaciju i analizu podataka, reprezentaciju podataka modelima i simulacijama, izvođenje rješenja kroz algoritam, identifikaciju, analizu i primjenu mogućih rješenja kako bi postigli najefikasnije rješenje, generalizaciju i prijenos postupka na rješavanje drugih problema (Tomljenović, 2018).

U svojoj knjizi *Mindstorms* Papert je predstavio LOGO kao temelj pristupa razmišljanja u učenju i poučavanju te je pokazao kako računalo može esencijalno doprinijeti mentalnim procesima (Bubica, Mladenović, Boljat, 2013). Računalo se u današnje vrijeme vrlo često koristi kao alat koji potiče učenje, posebice jer su jeftina i svima dostupna.

U knjizi *Znanost za sve Amerikance* koju je izdala grupa *American Association for the Advancement of Science* iznosi jednu važnu tvrdnju: „…učenici uglavnom uče od konkretnog prema apstraktnom.“ (Bubica, Mladenović, Boljat, 2013: 3). Sposobnost apstraktnog mišljenja, zaključivanje, generaliziranje samo su neke od vještina koje se razvijaju odrastanjem, a potrebno ih je često vježbati. Iako je programiranje samo po sebi apstraktno i teško razumljivo, posebice djeci, istodobno

je i odličan alat za razvoj prethodno navedenih sposobnosti te računalnog razmišljanja. No, u današnje vrijeme osmišljeno je i rješenje za tu apstraktnost programiranja, pa su tako osmišljeni vizualni programski jezici koji omogućavaju učenje programiranja u konkretnom okruženju (Bubica, Mladenović, Boljat, 2013).

Prema MZO (2018), u nastavi Informatike težište se treba postaviti na rješavanje problema i programiranje jer bi se time poticalo razvijanje računalnog razmišljanja. Računalno razmišljanje jedna je od četiri domene koje obuhvaća predmetni kurikulum Informatike. Ona predstavlja temeljni pristup kojim se razvija sposobnost rješavanja problema i programiranja. Takav pristup učenicima pruža da postanu stvaratelji različitih računalnih alata, a ne samo njihovi korisnici. Aktivnosti i sadržaji ishoda te domene razvijaju stvaralaštvo, inovativnost i poduzetnost te pružaju vrijedna znanja koja se mogu primijeniti u budućem poslovnom životu (MZO, 2018).

Novim kurikulumom predmeta Informatika nije specificiran niti jedan obavezan programski jezik, već je učitelju dan njegov slobodan izbor, u skladu s njegovim kreativnim zamislima (Tomljenović, 2018). Pa tako učitelji imaju na izbor hoće li se u nastavi služiti komercijalnim vlasničkim softverima ili besplatnim i slobodnim softverima (FOSS), o kojima će biti posvećeno sljedeće poglavlje.

3. BESPLATNI I SLOBODNI ALATI (*Free and Open Source Software*)

U svojoj knjizi *Slobodan softver u obrazovanju* (2013), autori Oreški i Šimović navode nekoliko kategorija softvera: softver u javnom vlasništvu (eng. *Public Domain Software*), slobodan softver otvorenog izvornog kôda (eng. *Free Open Source Software*), besplatan softver (eng. *Freeware*), softver za podjelu (eng. *Shareware*) i vlasnički softver (eng. *Commercial Propriatary Software*).

Richard Stallman začetnik je ideje slobodnoga softvera otvorenog izvornoga kôda. Stallman je 1984. godine pokrenuo projekt GNU koji je imao zadatak razviti slobodan operacijski sustav temeljen na UNIX-u. Godinu dana kasnije osnovana je *Free Software Foundation* (FSF) koja započinje razvoj niza softverskih proizvoda baziranih na zamisli slobodnog softvera (eng. *free software*). Značenje riječi *free*, u

ovom slučaju, FSF podrazumijeva kao „slobodan pristup“ (Oreški, Šimović, 2013). Osnovne slobode slobodnog softvera jesu: slobodno izvođenje programa za bilo koju namjenu; slobodno analiziranje rada programa i mijenjanje prema vlastitim potrebama; slobodno kopiranje i razmjena programa kako bi mogli pomoći svom susjedu; slobodno poboljšavanje programa i slobodno ponovno izdavanje promijenjenog programa kako bi cijela zajednica imala koristi (Oreški, Šimović, 2013).

„Neki od najznačajnijih projekata FOSS-a jesu operacijski sustav GNU, jezgra operacijskog sustava Linux, grafičko sučelje XFree86, web poslužitelj Apache, baza podataka MySQL, skriptni programski jezik PHP, grafičko korisničko sučelje GNOME, web preglednik Mozilla Firefox te skup uredskih alata OpenOffice“ (Oreški, Šimović, 2013: 28).

Mrežne stranice Debian-a (2019) navode da je Linux operacijski sustav, odnosno niz programa koji omogućava interakciju s računalom i pokretanje drugih programa. Najvažniji dio operacijskog sustava jest njegova jezgra, a kada govorimo o GNU/Linux sustavu (Slika 1.), tu jezgru čini Linux. Ostatak samog sustava čine drugi programi. GNU je za upotrebu u operacijskim sustavima koji su slični UNIX-u, poput Linuxa, razvio skup besplatnih softverskih alata kako bi korisnicima omogućili izvršavanje svakodnevnih zadataka, ali i onih nešto zahtjevnijih.

Slika 1. Logo operacijskog sustava *GNU/Linux*

U Republici Hrvatskoj niti jednim odgojno-obrazovnim dokumentom za osnovnu školu nisu propisani softveri za korištenje u nastavi. Udžbenici iz nastavnog predmeta Informatika uglavnom predstavljaju softvere komercijalnog vlasništva poput

operacijskog sustava Windows, Microsoft Office paketa namijenjenog uredskoj obradi podataka i sl. (Oreški, Šimović, 2013).

Oreški i Šimović (2012) proveli su istraživanje *Razlozi za i protiv korištenja slobodnog softvera otvorenog izvornoga kôda u osnovnom obrazovanju u Republici Hrvatskoj* kojim su dobili odgovore učitelja koji su glavni razlozi za i protiv korištenja FOSS-a. Glavni razlozi za korištenje FOSS-a jesu to što je besplatan, lako dobavljiv, kvalitetan softver koji zadovoljava njihove potrebe te im pruža veću sigurnost od komercijalnog vlasničkog softvera jer ne sadržava skrivene zlonamjerne softvere (engl. malware), te to što mogu pristupiti njegovu izvornom kôdu i modificirati ga u skladu sa svojim potrebama.

Unatoč brojnim prednostima koje pruža FOSS, postoje i određeni rizici korištenja takvih softvera. Prvi takav razlog jest to što imaju slabu korisničku podršku ili je uopće niti nema, jer FOSS razvija skupina programera, a ne postoji vlasnik. Nadalje, postoji mogućnost da se FOSS neće dalje razvijati ili ispravljati greške, što može zakomplicirati život ljudima koji ga koriste. Ako se u školi koristi FOSS i učenike učimo da ga koriste, može doći do problema, jer učenici najčešće na svojim osobnim računalima kod kuće koriste softvere komercijalnog vlasništva. To može dovesti do nekompatibilnih formata datoteka pri izvršavanju domaćih zadaća i slično. Također, može se dogoditi da ne postoji specifični obrazovni softver za platformu FOSS (Oreški, Šimović, 2012).

Važno je napomenuti da je tržišni udio FOSS-a u stalnom porastu, budući da ima stalan porast broja korisnika. Cilj pokrovitelja FOSS-a jest postići što veći broj korisnika, a to bi se moglo postići njihovom promidžbama, kojih gotovo i nema (Oreški, Šimović, 2013).

4. PROGRAMSKI JEZICI I ALATI

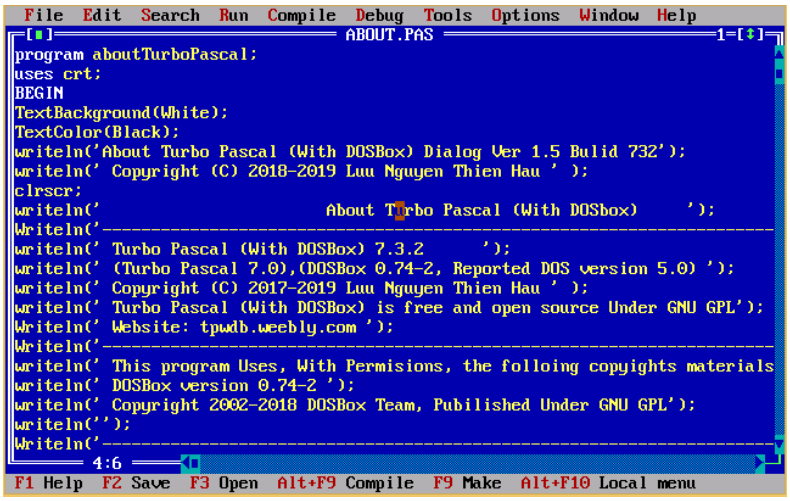
Programski jezici su „umjetni jezici kojima su pisani programi za rad na elektroničkom računalu.“ (Hrvatska enciklopedija, 2020). Postoje dvije vrste programskih jezika, viši i niži programski jezici. Niži programski jezici još se nazivaju i strojnim jezicima ili asemblerima. Budući da računalo izvršava isključivo programe napisane nižim programskim jezicima, svi programi koji su napisani višim

programskim jezicima moraju biti dodatno obrađeni kako bi ih računalo moglo izvršiti. To zahtijeva dodatno vrijeme, što predstavlja mali nedostatak za više programske jezike (Downey, 2014). Za prevođenje viših programskih jezika u niže koristi se jedna od dvije vrste programa, a to su interpreteri ili kompajleri. Unatoč tome, korištenje viših programskih jezika ima mnogobrojne prednosti, kao što su brže pisanje, lakše čitanje, programi su kraći te je veća vjerojatnost njihove ispravnosti. Također, programi napisani višim programskim jezicima, radi svoje prenosivosti, mogu se izvršavati na različitim računalima, dok se oni napisani nižim programskim jezicima mogu izvršavati isključivo na jednoj vrsti računala, odnosno na drugim vrstama računala mogu se izvršiti uz određene prepravke ili ponovnim pisanjem (Downey, 2014). Neki od viših programskih jezika su Python, C, C++, Perl i Java.

Sve do današnjeg dana, stvoren je velik broj programskih jezika. Kroz povijest su se u poučavanju programiranja isticali Pascal, Logo te QBASIC. S obzirom da su Pascal, Logo i QBASIC iznimno zastarjeli programski jezici, u podučavanju programiranja danas se radije koriste Scratch i Python.

4.1. *Pascal*

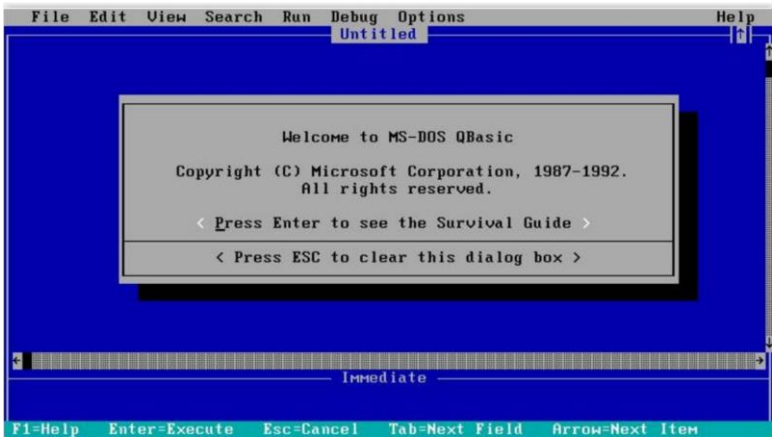
Niklaus Wirth razvio je programski jezik Pascal početkom 1970-ih godina, za podučavanje programiranja kao sustavnu disciplinu i za razvijanje učinkovitih i pouzdanih programa. Pascal je programski jezik opće namjene i visoke razine, a temeljen je na Algolu te uključuje mnoge njegove konstrukcije. Programi napisani u Pascalu lako su razumljivi i jednostavni za održavanje. Ovom jeziku popularnost je najviše porasla radi njegovih glavnih karakteristika i prednosti, kao što su jednostavnost njegova učenja, strukturiranost, transparentnost, učinkovitost i pouzdanost programa, opsežna provjera pogrešaka, nudi više vrsta podataka, datoteka, programskih struktura, podupire strukturno programiranje pomoću funkcija i procedura, a podupire i objektu orijentirano programiranje. Kako navodi Tutorialspoint (2015) u svom priručniku *Pascal Programming*, ovaj je programski jezik naziv dobio prema francuskom matematičaru Blaiseu Pascalu. U samom je početku program prevoditelj bio iznimno skup te si ga pojedinac nije mogao priuštiti, no, kada je na tržište izašao *Turbo Pascal*, cijena programa prevoditelja naglo je pala. Od svih programa prevoditelja dostupnih za *Pascal*, najčešći i najrašireniji su *Turbo Pascal Borland International Inc.*-a te *Microsoft*-ov *Quick Pascal* (Ganeshan, 1999).



Slika 2. Sučelje programa *Turbo Pascal 7.0*

4.2. *BASIC*

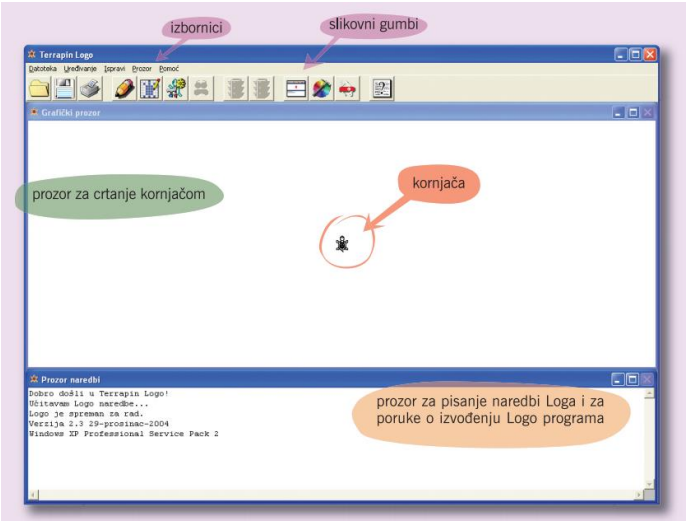
Program *BASIC* (*Beginner's All-Purpose Symbolic Instruction Code*) razvili su 1963. godine matematičari s Dartmoutha, John George Kemeny i Tom Krtzas (Rouse, 2011). Razvili su ga kao alat za podučavanje studenata kako bi mogli pisati jednostavne računalne programe (Christensson, 2006). Encyclopaedia Britannica (2016) navodi da je taj program trebao služiti za povećanje utjecaja računala u poslovnim i znanstvenim područjima. *BASIC* je najčešće bio korišten kao uvod u korištenje računalnih programskih jezika, radi svoje jednostavnosti. Bitna karakteristika jest to što se može brzo naučiti, a podrška je dostupna na većini operacijskih sustava (Rouse, 2011). Najpoznatija verzija *BASIC*-a danas je *QBASIC*. Microsoft je također razvio svoju verziju BASIC-a, nazvanu Visual BASIC nadodavši standardnoj verziji BASIC-a objektno orijentirane značajke te grafičko sučelje (Rouse, 2011).



Slika 3. Sučelje programa *QBasic*

4.3. *Logo*

Prva verzija programa *Logo* stvorena je 1968. godine u Cambridgeu, a stvorila ga je grupa ljudi pod vodstvom Wallacea Feurziga, no u samom radu i razvoju *Logoa* sudjelovao je i matematičar Seymour Papert (Đurđević, 2014). *Logo* je programski jezik opće namjene pa se može učiti na više načina te na više obrazovnih razina, stoga se uči od predškolske dobi do akademske razine. No, *Logo* je prvenstveno stvoren kao alat za učenje. Zahvaljujući svojoj jednostavnoj grafici mogu ga kreativno koristiti i djeca rane dobi crtajući razne poligone. Najvažnije karakteristike *Logoa* jesu njegova modularnost, proširivost, interaktivnost i fleksibilnost. *Logo* se najviše proslavio pojavom grafičkog pokazivača, *Logo* kornjače (Glavan, 2000). Djeca toj kornjači daju određene naredbe kuda treba ići, kada treba početi crtati i slično, kako bi nacrtali svoj crtež (Đurđević, 2014). Po uzoru na *Logo* napravljeni su novi alati za vizualno programiranje (Bubica, N., Mladenović, M., Boljat, I., 2013). Programsko okruženje *Logo*-a ima sposobnost poticanja učenike na razvoj samopouzdanja, zanimanja za računalno programiranje te razumijevanje konstrukcije petlje (Lewis, C.M., 2010).

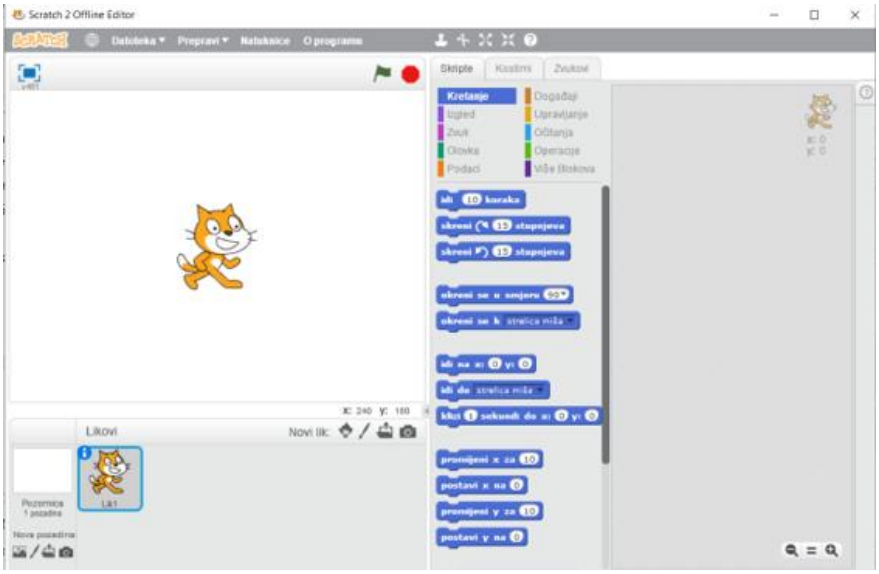


Slika 4. Sučelje programa *Terrapin Logo*

4.4. *Scratch*

Scratch je besplatan program otvorenoga kôda (Đurđević, 2014). To je tek nedavno izumljen jezik vizualnog programiranja, kojima se složeniji elementi programiranja, poput petlji i uvjeta, čine prirodnijima. Programski jezik *Scratch* je novije programsko okruženje, koje pruža gotovo iste funkcionalnosti kao i *Logo*. *Scratch* omogućava rad u višejezičnom okruženju, pa tako i na hrvatskom jeziku

(Đurđević, 2014). U *Scratchu* se blokovi naredbi slažu samo na sintaktički valjane načine, stoga su greške koje se u njemu pojavljuju uvijek semantičke pogreške, a nikada nisu rezultat pogreške tipkanja ili pogrešno upisane sintakse jezika (Lewis, C.M., 2010). Autori programa *Scratch* navode da je ovaj program nastao kao pomoć za razvijanje osnovnih vještina kao što su kreativno razmišljanje, sistematična analiza, jasna komunikacija, efikasna elaboracija i kontinuirano učenje (Đurđević, 2014).



Slika 5. Sučelje programa *Scratch*

5. PROGRAMSKI JEZIK *PYTHON*

Python je jedan od popularnijih programskih jezika. Stvorio ga je Guido van Rossum 1991. godine. *Python* je proceduralni i objektu orijentirani programski jezik. U kasnijim poglavljima bit će opisano što podrazumijeva proceduralnost, a što objektnu orijentiranost programskoga jezika. Vrlo je važno istaknuti da je *Python* u potpunosti besplatan programski jezik koji se može besplatno preuzeti na mrežnoj stranici http://www.python.org/download/ te jednostavno instalirati.

Prednost programskog jezika *Pythona* jest u tome što funkcionira na različitim platformama (Windows, Mac, Linux, Raspberry Pi itd.), te ima sintaksu sličnu engleskom jeziku. Za razliku od ostalih programskih jezika, sintaksa *Pythona* omogućava programerima da napišu programe u samo nekoliko linija kôda. Budući da *Python* radi preko interpretera, kod napisan tom sintaksom izvršava se čim je napisan,

što omogućava brzu izradu prototipa. Mrežna stranica w3schools (2020) također navodi da se *Python* može tretirati na tri načina: proceduralni, objektu orijentirani i funkcionalni način.

Kao što je prethodno spomenuto, *Python* ima neke sličnosti s engleskim jezikom, jer je dizajniran da bude lako čitljiv. Za razliku od ostalih programskih jezika koji se za dovršavanje naredbi koriste zarezom ili zagradom, *Python* se koristi novim retkom. Također, kod definiranja opsega npr. petlje, funkcija i klasa, oslanja se na uvlačenje, koristeći razmaknice, dok se u ostalih programskih jezika često koriste vitičaste zagrade.

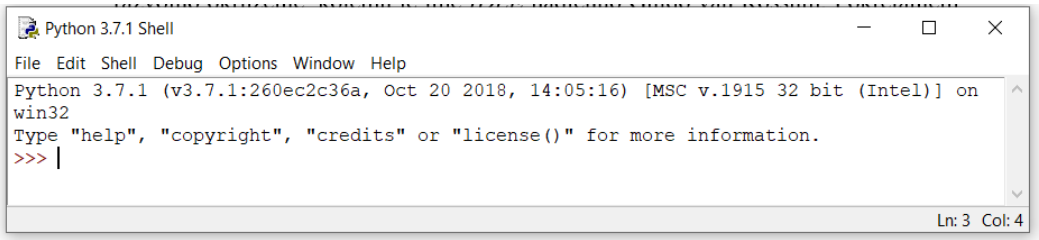
Python je prikladan za početno podučavanje programiranja jer se na samome početku može koristiti bez složenih detalja, koji se mogu uvoditi postupno i po potrebi. Vrlo je jednostavan programski jezik koji omogućava iznimno brzo svladavanje svih načela proceduralnog i objektu orijentiranog programiranja (Budin, Brođanac, Markučić, Perić, 2012).

Osim za podučavanje, programski se jezik *Python* koristi i za razvoj weba, softvera, skriptiranje sustava te u matematici. Kada govorimo razvoju weba, podrazumijevamo da se *Pythonom* možemo služiti na serverima za izradu web aplikacija. Na mrežnim stranicama *w3schools*-a (2020) navodi se da se *Python* također može povezati na sustave baza podataka te čitati i modificirati datoteke. Uz to, može podržati velik broj podataka te se koristiti u kompleksnoj matematici. Kao jedan od dokaza za to da se *Python* može upotrebljavati i u profesionalne svrhe Budin i suradnici (2012: 5) navode: „..kao potvrda te tvrdnje može poslužiti činjenica da je autor *Pythona* Guido van Rossum od 2005. godine zaposlen u tvrtki *Google*, te da je *Python* jedan od tri službena programska jezika kojim se koristi u toj tvrtki“.

5.1. PROGRAMSKO OKRUŽENJE ZA PRIPREMANJE PROGRAMA – *IDLE*

U kojem god da se programskom jeziku programira, za pisanje i ispitivanje programa koriste se posebni programi koje nazivamo programskim pomagalima. Skupina takvih pomagala predstavlja integrirano razvojno okruženje (engl. *Integrated Development Environment – IDE*). Pa tako programski jezik *Python* koristi svoje razvojno okruženje, kojemu je ime *IDLE* nadjenuo Guido van Rossum (Budin,

Brođanac, Markučić, Perić, 2012). Pokretanjem *IDLE*-a, otvara se prozor koji predstavlja interaktivno sučelje s *Pythonom*, a njegov naslov je *Python Shell* (slika 6).



Slika 6. Interaktivno sučelje *Pythona*

Ispod naslovne trake nalazi se traka izbornika sa pripadajućim nazivima padajućih izbornika. Na slici 6. vidljivi su sljedeći izbornici: *File, Edit, Shell, Debug, Options, Window* i *Help*. Na vrhu samog radnog okvira nalaze se informacije o inačici *Pythona* koji se koristi te uputa za upisivanje riječi *help, copyright, credits* ili *license* kako bi se doznale dodatne informacije o *Pythonu*.

5.2. PRIKAZIVANJE PODATAKA U PYTHONU

Svaki računalni program obrađuje podatke, a svaki taj podatak predstavlja određenu vrijednost. U programskom jeziku *Pythonu* svaki je podatak nekog određenog tipa. Pojmovi vrijednost i tip upotrebljavaju se u proceduralnim jezicima. Budući da je *Python* i proceduralni i objektu orijentirani programski jezik, u njemu se osim tih pojmova koriste i pojmovi klasa umjesto tipa, te objekt umjesto vrijednosti. *Python* razlikuje nekoliko osnovnih tipova podataka, a to su: cijeli broj, broj s pomičnom točkom (realni broj), logički ili Booleov tip te string ili znakovni niz (Budin, Brođanac, Markučić, Perić, 2012). Mogućnost prikaza brojeva važna je značajka programskih jezika, jer se oni često koriste u raznim primjenama kroz računske postupke. Programski jezik *Python* razlikuje cijele, realne i kompleksne brojeve.

5.2.1. Cijeli brojevi

Za cjelobrojni tip podataka u *Pythonu* se koristi naziv *int*. Za razliku od ostalih programskih jezika, kod *Pythona* ne postoji ograničenje veličine, odnosno broja znamenaka cijelog broja (Budin, Brođanac, Markučić, Perić, 2012). Primjer takvog zapisa prikazan je na slici 7.

```
>>> 78
78
>>> -2458
-2458
>>> 1234567891234567891234567891234567891234567 89
1234567891234567891234567891234567891234567 89
>>>
```

Slika 7. *Ispis cijelih brojeva bez ograničenja*

U rijetkim slučajevima potrebno je određene cijele brojeve zapisati u binarnom obliku. Tada se, prilikom njegova upisivanja, ispred binarnog zapisa broja upisuje 0b, a *Python* će ispisati taj broj u dekadskom obliku. Također, ako je potrebno zapisati brojeve u heksadekadskom obliku, ispred heksadekadskog zapisa upisuje se 0x, a *Python* ga ispisuje u dekadskom obliku. Bez obzira na to u kojem se obliku upiše broj, bilo u binarnom ili heksadekadskom obliku, *Python* će ga uvijek pretvoriti i ispisati u dekadskom obliku (Budin, Brođanac, Markučić, Perić, 2012). Primjer takvog zapisa prikazan je na slici 8.

```
>>> 0b01010101
85
>>> 0b111000111
455
>>> 0xA
10
>>> 0xFA
250
>>>
```

Slika 8. *Ispis binarnog i heksadekadskog zapisa*

U slučaju kada želimo da nam određeni broj iz dekadskog oblika ispiše u binarnom ili heksadekadskom obliku, koristimo ugrađene funkcije *bin()* ili *hex()*. Pritom se u zagrade upisuje broj u dekadskom obliku, a *Python* ga ispisuje u binarnom ili heksadekadskom obliku kao znakovni niz, ovisno o tome koja se funkcija koristila (Budin, Brođanac, Markučić, Perić, 2012). Primjer takvog zapisa prikazan je na slici 9.

```
>>> bin (50)
'0b110010'
>>> hex (337)
'0x151'
>>>
```

Slika 9. *Ispis dekadskog zapisa u binarnom i heksadekadskom obliku*

5.2.2. Realni brojevi

U matematici se skup realnih brojeva dijeli na racionalne i iracionalne brojeve. Racionalni se brojevi mogu prikazati u obliku razlomka te kao decimalan zapis broja. Kada se prikazuje kao razlomak važno je da njegov brojnik čini cijeli broj, a nazivnik neki prirodan broj. Kada je riječ o decimalnom prikazu racionalnog broja, onda je to broj koji ima konačan broj decimalnih mjesta ili ima periodičko ponavljanje nakon određenog decimalnog mjesta. Za razliku od racionalnih brojeva, iracionalni se brojevi ne mogu tako prikazivati, jer oni imaju beskonačan broj decimalnih mjesta te u njih ne postoji periodičko ponavljanje znamenaka. Tek kada se iracionalni brojevi dodaju skupu racionalnih brojeva i stvore skup realnih brojeva, mogu se zapisati u računalu na isti način kao i racionalni brojevi (Budin, Brođanac, Markučić, Perić, 2012).

Prema Budinu i suradnicima (2012), racionalni se brojevi u računalu zapisuju formatom s pomičnom točkom, a taj se tip podataka naziva *float*. Pri upisivanju racionalnog broja u programu, obavezno je pisanje decimalne točke, jer će samo tako *Python* prepoznati da je riječ o *float* tipu podataka. Racionalni se brojevi još mogu zapisati i znanstvenim načinom zapisivanja, tako da se zapiše kao racionalni broj s točkom, zatim slovo e te eksponent kao cijeli broj. Postoji određena vrijednosna granica u sklopu koje će *Python* brojeve ispisivati u obliku brojeva s decimalnom točkom. Ukoliko brojevi prelaze tu vrijednosnu granicu, *Python* će ih zapisati u eksponencijalnom obliku (Budin, Brođanac, Markučić, Perić, 2012). Primjeri zapisa racionalnih brojeva prikazani su na slici 10.

```
>>> 89
89
>>> 89.
89.0
>>> 8.9
8.9
>>> 0.89
0.89
>>> 0.089
0.089
>>> 0.0000089
8.9e-06
>>> 0.0000895486
8.95486e-05
>>> 8e2
800.0
>>> 8e6
8000000.0
>>> 8e16
8e+16
>>>
```

Slika 10. *Zapisi racionalnih brojeva u Pythonu*

“Broj koji bi imao eksponent veći od +308 je za *Python* beskonačno velik broj i *Python* ga zapisuje kao *inf* (engl. Infinite - beskonačno).” (Budin, Brođanac, Markučić, Perić, 2012: 20). Primjer zapisa takvog broja nalazi se na slici 11.

```
>>> 1.1845486486465316486486 48e309
inf
>>>
```

Slika 11. *Zapis beskonačnog broja u Pythonu*

5.2.3. Logički ili Booleov tip podataka

Prema Budinu i suradnicima (2012), pojam, odnosno naziv, logički tip potječe iz logike, čiji je osnovni pojam logički sud. Logički sud podrazumijeva svaku tvrdnju koja može biti istinita ili lažna. U svrhu istraživanja sudova razvila se posebna grana matematike, pod nazivom matematička logika, čiju osnovu čini logička algebra ili drugim nazivom Booleova algebra. Logički ili Booleovi tipovi podataka u programskom jeziku *Pythonu* nazivaju se *bool*. Takvi tipovi podataka poprimaju samo dvije moguće vrijednosti, a to su: *True* za istinitost i *False* za lažnost (Budin, Brođanac, Markučić, Perić, 2012). Prilikom upisivanja vrijednosti *True* ili *False*, važno je pripaziti da se te vrijednosti napišu velikim početnim slovom, u suprotnom će program izbaciti pogrešku (Slika 12.).

```
>>> True
True
>>> False
False
>>> true
Traceback (most recent call last):
  File "<pyshell#105>", line 1, in <module>
    true
NameError: name 'true' is not defined
>>> false
Traceback (most recent call last):
  File "<pyshell#106>", line 1, in <module>
    false
NameError: name 'false' is not defined
>>>
```

Slika 12. *Ispravan i pogrešan ispis vrijednosti True i False*

Pomoću *Pythonovih* ugrađenih funkcija *int* () i *bool* () može se ustanoviti cjelobrojna vrijednost neke logičke vrijednosti. Pritom se očituje da ekvivalent vrijednosti *True* iznosi 1 ili bilo koja druga cjelobrojna vrijednost, a vrijednosti *False* iznosi isključivo 0 (Budin, Brođanac, Markučić, Perić, 2012). Primjer na slici 13. prikazuje cjelobrojnu vrijednost logičkih vrijednosti *True* i *False* pomoću ugrađene

funkcije *int* () te logičku vrijednost cjelobrojnih vrijednosti pomoću ugrađene funkcije *bool* ().

```
>>> int (True)
1
>>> int (False)
0
>>> int (1)
1
>>>
>>>
>>>
>>>
>>> int (True)
1
>>> int (False)
0
>>> bool (1)
True
>>> bool (0)
False
>>> bool (-1)
True
>>> bool (-100)
True
>>>
```

Slika 13. *Prikaz cjelobrojnih i logičkih vrijednosti pomoću ugrađenih funkcija int () i bool ()*

5.2.4. String – znakovni niz

Osim brojeva, u programiranju su važni i tekstovi, pogotovo u današnje vrijeme kada se rad računala uglavnom bazira na stvaranju i pohranjivanju tekstova, njihovu oblikovanju, pretraživanju te prijenosu. Za prikaz teksta osnovni je tip podataka znakovni niz ili string (Budin, Brođanac, Markučić, Perić, 2012). U *Pythonu* se stringovi zovu str, a njihova se vrijednost označava jednostrukim ili dvostrukim navodnicima na početku i kraju niza. Pritom se navodni znakovi trebaju pažljivo upotrebljavati. Preporuča se korištenje jednostrukih navodnika, a dvostruke samo u slučajevima kada se unutar teksta upotrebljavaju navodni znakovi. Također, prilikom pisanja dvostrukih navodnika, važno je shvaćanje postojanja posebne tipke za njihovo upisivanje, jer to nije isto kao i upisivanje jednostrukih navodnika uzastopno dva puta. Ukoliko tekst upisujemo isključivo koristeći navodne znakove, *Python* će ih i ispisati zajedno s korištenim navodnicima. No, postoji funkcija *print* () koja služi za ispisivanje teksta, odnosno znakovnog niza bez rubnih navodnika. Na slici 14. prikazani su primjeri zapisivanja znakovnih nizova pomoću prethodno navedenih znakova i funkcije *print ()*.

```
>>> 'Programiranje u osnovnoj školi'
'Programiranje u osnovnoj školi'
>>> 'Nauči programirati u Pythonu uz knjigu "Naučite Python".'
'Nauči programirati u Pythonu uz knjigu "Naučite Python".'
>>> "Kako biste napisali tekst u programskom jeziku Python, morate ga napisati na ovaj način: 'Učim programirati.'"
"Kako biste napisali tekst u programskom jeziku Python, morate ga napisati na ovaj način: 'Učim programirati.'"
>>> print ('Programiranje u osnovnoj školi')
Programiranje u osnovnoj školi
>>> print ('Nauči programirati u Pythonu uz knjigu "Naučite Python".')
Nauči programirati u Pythonu uz knjigu "Naučite Python".
>>>
```

Slika 14. *Zapisivanje znakovnih nizova*

Osim navedenih znakova, postoji nekolicina posebnih znakova čija je svrha oblikovanje teksta, a prikazani su u Tablici 1. (Budin, Brođanac, Markučić, Perić, 2012). Primjena nekih posebnih znakova iz Tablice 1. prikazana je kroz primjere na slici 15.

Tablica 1. Posebni znakovi za oblikovanje teksta u *Pythonu*

Posebni znak	Svrha
\n	prijelaz u novi red
\t	tabulator
\\	ispisati lijevo ukošenu crtu
\’	ispisati jednostruki navodnik
\”	ispisati dvostruki navodnik

```
>>> '1. \n2. \n3. \n4. \n5.'
'1. \n2. \n3. \n4. \n5.'
>>> print ('1. \n2. \n3. \n4. \n5.')
1.
2.
3.
4.
5.
>>> '1. \t2. \t3. \n4. \t5. \t6.'
'1. \t2. \t3. \n4. \t5. \t6.'
>>> print ('1. \t2. \t3. \n4. \t5. \t6.')
1.      2.      3.
4.      5.      6.
>>> print ('Nauči programirati u Pythonu uz knjigu \"Naučite Python\".')
Nauči programirati u Pythonu uz knjigu "Naučite Python".
>>>
```

Slika 15. *Primjena posebnih znakova za ispis znakovnih nizova*

Uz posebne znakove, navedene u Tablici 1., postoje i četiri operatora koji se koriste u izrazima sa stringovima (Budin, Brođanac, Markučić, Perić, 2012). Ti operatori su: +, *, *in* i *not in*. Kod znakovnih nizova može se koristiti operator +, no

on neće imati ulogu zbrajanja, nego pridruživanja jednog znakovnog niza drugom (Downey, 2014). Drugim riječima, operator +, odnosno operator nadovezivanja, koristi se za dobivanje novog stringa koji će biti sastavljen od dvaju zadanih stringova. Ukoliko se koristi operator * sa zadanim stringom, dobit će se novi string koji će biti sastavljen od zadanog niza ponovljenog zadani broj puta. „Operatori *in* i *not in* ispituju je li prvi string u izrazu sadržan u drugom znakovnom nizu i to počevši od bilo kojeg mjesta.“ (Budin, Brođanac, Markučić, Perić, 2012: 168). Prilikom korištenja operatora *in* ili *not in*, moguće je dobiti jedan od dva rezultata, a to je *True* ili *False*. Primjeri s korištenjem navedenih operatora prikazani su na slici 16.

```
>>> 'program' + 'iranje'
'programiranje'
>>> 'ABC' * 4
'ABCABCABCABC'
>>> 'gra' in 'programiranje'
True
>>> 'gra' not in 'programiranje'
False
>>> 'miro' in 'programiranje'
False
>>> 'miro' not in 'programiranje'
True
>>>
```

Slika 16. *Primjeri korištenja operatora sa stringovima*

Važno je istaknuti da postoje i ugrađene funkcije sa stringovima, a neke od njih su len (), min () i max () (Budin, Brođanac, Markučić, Perić, 2012). Prilikom njihova korištenja praktičnije je prethodno imenovati string. String 'programiranje' imenujemo primjerice sa z, pa tako dobivamo z = 'programiranje'. Korištenjem funkcije print(len (z)), kao rezultat se ispisuje duljina tog stringa, u ovom slučaju broj 13. Iako se riječ programiranje u hrvatskom jeziku sastoji od 12 slova (znakova), Python slovo, odnosno znak „nj“ razlikuje kao dva zasebna znaka, „n“ i „j“. Ukoliko želimo saznati koji znak iz zadanog znakovnog niza ima najmanju kodnu vrijednost, koristimo funkciju min (z). Suprotno tomu, ako želimo saznati koji znak iz zadanog stringa ima najveću vrijednost, koristimo funkciju max (z) (Budin, Brođanac, Markučić, Perić, 2012). Primjeri realizacije nekih od prethodno navedenih ugrađenih funkcija sa stringovima prikazani su na slici 17, a na slici 18. nalazi se rezultat ispisa korištenja ugrađenih funkcija.

```
z = 'programiranje'

print (len (z))

print (min (z))

print (max (z))
```

Slika 17. *Primjer korištenja ugrađenih funkcija sa stringovima*

```
13
a
r
>>>
```

Slika 18. *Rezultat ispisa korištenja ugrađenih funkcija sa stringovima s prethodne slike*

Osim svega navedenog, *Python* ima mogućnost davanja stringova pomoću trostrukih navodnika. Ti znakovi omogućavaju dokumentiranje napisanih programa, stoga se takav niz naziva *docstring*, odnosno dokumentacijski niz. Unutar trostrukih navodnika tekst se upisuje onako kako se očekuje da bude ispisan (Budin, Brođanac, Markučić, Perić, 2012). Budući da se znakovi pohranjuju u računalu brojevnim prikazom, potrebno ih je kodirati, odnosno zapisati pod određenim kodom. U tu svrhu, *Python* koristi kodni sustav Unicode inačicu UTF-8, čije se kodiranje znakova provodi s jednim do šest bajtova.

5.2.5. Aritmetički izrazi

Budin i suradnici (2012) navode da, kao i u matematici, *Python* razlikuje operande i operatore. Operatori određuju operaciju koja se treba provesti s vrijednostima operanada kako bi se dobio određeni rezultat, odnosno neka nova vrijednost. Kada je riječ o operandima u aritmetičkim izrazima, kod *Pythona* su to cijeli brojevi ili brojevi s pomičnom točkom, odnosno tipovi podataka int i float. Kao što je navedeno i objašnjeno u prethodnom potpoglavlju o stringovima, odnosno znakovnim nizovima, matematičke se operacije ne mogu se izvoditi sa znakovnim nizovima, barem ne na onaj način na koji bi se to moglo očekivati (Downey, 2014).

U tablici 2. prikazani su operatori koje *Python* razlikuje, te njihova uloga (Budin, Brođanac, Markučić, Perić, 2012).

Tablica 3. Aritmetički operatori

Operator	Uloga operatora
+	zbrajanje
-	oduzimanje ili negacija
*	množenje
**	potenciranje
/	dijeljenje
//	cjelobrojno dijeljenje
%	računanje ostatka dijeljenja (modulo)

Korištenjem prva tri operatora iz tablice 2., rezultat će biti onog tipa kojeg su i sami operandi. Pa tako, ako su oba operanda tipa *int*, rezultat će također biti tipa *int*. Ukoliko su operandi tipa *float*, rezultat će isto biti tipa *float*. No, ako su operandi različitog tipa, jedan je tipa *int*, a drugi tipa *float*, rezultat će biti tipa *float*. Kada se koristi operator dijeljenja (/), rezultat će uvijek biti tipa *float*, bez obzira na to kojeg su tipa operandi (Budin, Brođanac, Markučić, Perić, 2012). Posebni operatori su operator za cjelobrojno dijeljenje (//) te operator za računanje ostatka dijeljenja (%). Operatorom cjelobrojnog dijeljenja kao rezultat se dobiva broj koji označava koliko se puta djelitelj može oduzeti od djeljenika. Ukoliko je samo jedan od operanada tipa *float*, rezultat će biti također tipa *float*, a ukoliko su oba operanda tipa *int*, rezultat će također biti tipa *int*. Operator za računanje ostatka dijeljenja često je usko vezan uz operator cjelobrojnog dijeljenja. Kao što mu i samo ime govori, on ispisuje broj koji čini ostatak nakon dijeljenja dvaju brojeva (Budin, Brođanac, Markučić, Perić, 2012). Primjena prethodno navedenih aritmetičkih operatora prikazana je na slici 19.

```
>>> 5 + 3
8
>>> 5 - 3
2
>>> 5.2 + 3.1
8.3
>>> 5.2 - 3.1
2.1
>>> 8 * 2
16
>>> 2 ** 3
8
>>> 8 / 2
4.0
>>> 8 // 3
2
>>> 8 % 3
2
>>>
```

Slika 19. *Primjeri korištenja aritmetičkih operatora u Pythonu*

5.2.6. Varijable i znak pridruživanja

Kako bi se omogućilo pamćenje različitih vrijednosti, uvode se varijable. "Varijable su imena koja će biti pridružena pojedinim vrijednostima." (Budin, Brođanac, Markučić, Perić, 2012: 31). Varijable su praktične jer se kasnije tijekom pisanja programa mogu dohvaćati i upotrebljavati njihove vrijednosti samim njihovim navođenjem. U matematici znak = predstavlja znak jednakosti, no u *Pythonu* to nije tako. Znak = naziva se znakom pridruživanja. Taj znak koristi se za pridruživanje određene vrijednosti, koja se nalazi tom znaku s desne strane, varijabli s lijeve strane tog znaka (Budin, Brođanac, Markučić, Perić, 2012). Na slici 20. prikazano je pridruživanje određene vrijednosti varijabli „a" s lijeve strane, a s desne strane nalazi se ispis rezultata.

```
a = 5

print ('a = ', a)
```

```
a =  5
>>>
```

Slika 20. *Primjer pridruživanja vrijednosti varijabli i rezultat ispisa*

Ukoliko nismo sigurni kojeg je tipa neka određena vrijednost, interpreter nudi mogućnost ispisivanja tipa te vrijednosti upisivanjem, npr. type('Pozdrav!'), tada će nam interpreter ispisati odgovor: <type 'str'> (Downey, 2014).

5.3. STRUKTURNO PROGRAMIRANJE

Strukturno programiranje sastavnica je proceduralnog programiranja. Razvijeno je u svrhu olakšavanja pisanja programa korištenjem programskih struktura kao što su procedure, funkcije, petlje te uvjetni izrazi, o kojima će se detaljno govoriti dalje u radu (Budin, Brođanac, Markučić, Perić, 2012). Proceduralni se način programiranja temelji na funkcijama, odnosno blokovima programskoga kôda. U slijedećih nekoliko potpoglavlja bit će pobliže objašnjene sastavnice strukturnog programiranja.

5.3.1. Upravljanje tokom izvođenja programa

Izvršavanje programa najčešće je slijedno izvršavanje koje je samo po sebi logično jer je linearno. No, ako se isključivo na takav način izvršava neki program, on će u svakom slučaju imati isti ishod. Stoga je često potrebno taj linearni niz izvršavanja programa promijeniti. To se često vrši ponavljanjem blokova naredbi, izvršavanjem novih blokova naredbi kroz razne funkcije ili preskakanjem određenih naredbi ili blokova naredbi (Maleš, Redžepagić, Rakijašić, 2018). Tijekom narednih potpoglavlja bit će objašnjeni načini promjene tijeka izvršavanja programa korištenjem grananja, petlji te određenih naredbi.

5.3.1.1. Uvjetno izvođenje

Prema Budinu i suradnicima (2012), prilikom pisanja programa, gotovo je uvijek nužna mogućnost provjere ispunjenosti uvjeta i promjene izvođenja programa s obzirom na ispunjenost uvjeta. Za to služe uvjetni iskazi, a najjednostavniji takav oblik jest iskaz *if*. Na slici 21. prikazan je najjednostavniji primjer korištenja iskaza if u kojem će se ako je uvjet zadovoljen, izvršit uvučeni iskaz, no, ukoliko uvjet nije zadovoljen, neće se dogoditi ništa.

```
x = 5

if x > 0:
    print ("x je pozitivan broj!")
```

```
x je pozitivan broj!
>>>
```

Slika 21. *Jednostavan primjer korištenja iskaza if te rezultat njegova ispisa*

Osim navedenog načina izvođenja iskaza *if* u kojem se uvjet izvršava ili prekida rad programa, postoji i iskaz u kojem postoje dvije mogućnosti njegova izvođenja,

ovisno u uvjetu. Ako je uvjet uz iskaz *if* zadovoljen, izvršava se njegov uvučeni iskaz. Ukoliko uvjet uz *if* nije zadovoljen, izvršava se drugi skup iskaza koji slijedi nakon iskaza konstrukcije *else* (Downey, 2014). Na slici 22. prikazan je primjer alternativnog izvođenja iskaza *if*. U prikazanom primjeru uvjet glasi x%2==0. Budući da se u uvjetu koristi operator modulo (%), to znači da se ispituje je li ostatak dijeljenja broja x brojem 2 jednak 0, ako je, izvršava se naredba unutar *if* iskaza, ukoliko taj ostatak nije jednak 0, izvršava se naredba konstrukcije *else*.

```
x = 15

if x%2 == 0:
   print ('x je paran broj')
else :
   print ('x je neparan broj')
```

```
x je neparan broj
>>>
```

Slika 22. *Primjer alternativnog izvođenja iskaza if*

Također, u nekim je situacijama potrebno koristiti više od dviju navedenih grana. U takvom se slučaju koriste ulančani uvjetni iskazi. U tim se iskazima osim iskaza *if* i *else*, koristi i iskaz *elif*, odnosno *else if*. Na slici 23. prikazan je primjer korištenja ulančanih uvjetnih iskaza. Primjena *elif* iskaza nema količinsko ograničenje, odnosno može se koristiti više takvih iskaza jedan za drugim. U takvim će se situacijama provjeravati jedan po jedan uvjet, sve dok se ne dođe do prvog zadovoljenog, odnosno točnog uvjeta (Downey, 2014).

```
x = 6

y = 9

if x > y :
   print ('x je veći od y')
elif x < y :
   print ('x je manji od y')
else :
   print ('x i y su jednaki')
```

```
x je manji od y
>>>
```

Slika 23. *Primjer korištenja ulančanih uvjetnih iskaza*

Osim navedenih mogućnosti njihova korištenja, uvjetne je naredbe moguće ugnijezditi unutar neke druge uvjetne naredbe (Kalafatić, Pošćić, Šegvić, Šribar, 2016), što je prikazano na slici 24.

```
x = 6

y = 9

if x > y :
   print ('x je veći od y')
else:
   if x < y:
      print ('x je manji od y')
   else:
      print ('x i y su jednaki')
```

```
x je manji od y
>>>
```

Slika 24. *Primjer ugniježđene uvjetne naredbe*

5.3.1.2. Programske petlje

U programiranju često dolazi do potrebe za ponavljanjem određenog dijela kôda. Najlakši način kojim bi se to ostvarilo jest kopiranje tog dijela kôda onoliko puta koliko je to potrebno. No, to donosi mnoge nedostatke. Najvažniji nedostatak je nepotrebna redundancija. A uz to, kao veliki nedostatak je velika vjerojatnost buduće promjene ili ispravka naredbi koje se ponavljaju, jer će programeri time morati promijeniti sve kopije kôda koji se ponavlja (Kalafatić, Pošćić, Šegvić, Šribar, 2016). Kako bi se spriječilo takvo kompliciranje brojni programski jezici, pa tako i sam *Python*, nude jasniju mogućnost izražavanja takvih ponavljajućih odnosno iterativnih postupaka, a to su programske petlje. Nazivaju se petlje zato što se nakon uspostavljanja istinitosti njihova uvjeta izvršavanje zadatka vraća na početak (Downey, 2014). Programski jezik Python koristi se petljama *for* i *while*.

5.3.1.2.1.Petlja *while*

Često je potrebno određene zadatke ponavljati tako dugo, dok je zadovoljen određeni uvjet. To se može vršiti petljom *while*. Tijekom izvršavanja petlje *while*, prvo se vrednuje istinitost uvjeta. Ukoliko uvjet nije ispunjen, petlja *while* se prekida, te se izvršavanje zadatka nastavlja sljedećim iskazom. Ukoliko je uvjet zadovoljen, izvršavaju se naredbe unutar petlje *while*, te se vraća na prvi korak, odnosno slijedi ponovno vrednovanje istinitosti uvjeta (Downey, 2014). Naredbe koje se nalaze unutar tijela petlje izvodit će se dokle god je uvjet zadovoljen. Primjer programa koji koristi petlju *while* prikazan je na slici 25. U ovom primjeru nalazi se operator '+=' čija je uloga da zadanu varijablu zbraja s vrijednošću koja se nalazi s desne strane operatora te joj dodjeljuje vrijednost, odnosno u ovom slučaju x+=1, isto je kao i izraz x=x+1.

To znači da svaki puta kada se izvrši petlja while, vrijednost varijable x povećava se za 1, a petlja će se izvršavati dok god bude zadovoljen uvjet.

```
x = 1

while x < 11:
    print (x ** 2)
    x += 1
```

```
1
4
9
16
25
36
49
64
81
100
>>>
```

Slika 25. *Primjer korištenja petlje while*

Važno je da se u tijelu petlje mijenja vrijednost jedne ili više varijabli, zato da se u jednom trenu petlja može prekinuti, jer uvjet neće biti ispunjen. Ukoliko u tijelu petlje ne postoji promjena vrijednosti niti jedne varijable, petlja će se nastaviti ponavljati beskonačno mnogo puta. Tada govorimo o beskonačnoj petlji (Downey, 2014). Primjer beskonačne petlje s rezultatom beskonačnog ispisa varijable n prikazan je na slici 26.

```
n = 9
while n > 0 :
    print (n)
```

```
9
9
9
9
9
9
9
9
9
9
9
9
9
9
9
9
9
9
9
9
9
9
9
9
9
9
```

Slika 26. *Primjer beskonačne petlje*

Kao i kod uvjetnih naredbi *if*, i petlji *while* može se dodati konstrukcija *else*, odnosno njen alternativni stavak, koji se odvija isključivo kada je petlja uredno završena (Kalafatić, Pošćić, Šegvić, Šribar, 2016). Na slici 27. prikazan je primjer petlje *while* sa dodanom *else* konstrukcijom.

```
x = 11

while x < 11:
    print (x ** 2)
    x += 1
else:
    print ('x je veći od 11')
```

```
x je veći od 11
>>>
```

Slika 27. *Primjer petlje while sa else konstrukcijom*

5.3.1.2.2. Petlja *for*

Prema Kalafatiću i suradnicima (2016), petlja *for* najpraktičnija je za korištenje kada je unaprijed poznato koliko će se puta određeni dio kôda ponavljati. Petlja *for* najčešće se upotrebljava u vezi s rasponom *range*, složenim tipom podataka koji podrazumijeva skup cijelih brojeva s konstantnim prirastom (Kalafatić, Pošćić, Šegvić, Šribar, 2016). Na slici 28. prikazan je primjer korištenja petlje for sa rasponom range u programu za ispisivanje kvadrata svih brojeva od 1 do 20.

```
for x in range (1, 21):
    print (x ** 2, end=' ')
    x += 1
```

```
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361 400
>>>
```

Slika 28. *Primjer korištenja petlje for sa rasponom range*

Također, kao i naredba *if*, petlja *for* može sadržavati konstrukciju *else*, odnosno svoj alternativni stavak (Kalafatić, Pošćić, Šegvić, Šribar, 2016). Taj se dio petlje *for* koristi isključivo kada petlja *for* besprekidno izvršava svoj zadatak, odnosno u slučaju kada nije prekinuta naredbom *break*. Prilikom korištenja petlje *for* sa konstrukcijom *else* u sebi, moguća su dva slučaja kao rezultat pretraživanja neke određene strukture podataka (Hruška, 2019). U prvom je slučaju petljom *for* pronađen element pretraživanja, te se poziva naredba *break*, a *else* dio petlje se ne izvršava (Slika 29.). U drugom pak slučaju, element pretraživanja ne postoji, odnosno nije pronađen, stoga petlja *for* završava svojim radom, a program se nastavlja izvršavati kroz *else* konstrukciju (Slika 30.).

```
lista = [1, 2, 3, 4, 5, 6]

for element in lista :
   if element == 5 :
      break
else :
   print ("Nije pronađeno!")
```

```
>>>
```

Slika 29. *Primjer petlje for tijekom koje se ne izvršava else konstrukcija*

```
lista = [1, 2, 3, 4, 5, 6]

for element in lista :
   if element == 10 :
      break
else :
   print ("Nije pronađeno!")
```

```
Nije pronađeno!
>>>
```

Slika 30. *Primjer petlje for tijekom koje se izvršava else konstrukcija*

5.3.1.2.3. Naredba *break*

Naredba break najčešće se koristi u petljama for i while. Pozivanjem ove naredbe napuštaju se te petlje, odnosno dolazi do njihova završetka (Hruška, 2019). Primjerice, možemo omogućiti korisniku da samostalno odredi kada će se pokrenuti naredba break, odnosno da napusti petlju, kada to želi, tako što upisuje da ili ne. Primjer programa u kojem se na taj način koristi naredba break prikazan je na slici 31.

```
while True:
   line = input('> ')
   if line == 'da' :
      break
   print (line)

print ('Gotovo!')
```

```
> da
Gotovo!
>>>
```

Slika 31. *Primjer korištenja naredbe break*

Kalafatić i suradnici (2016) naglašavaju da je važno naredbu *break* staviti posljednju, ukoliko se u tijelu petlje nalazi veći broj naredbi koje se trebaju izvršiti, jer one u protivnom neće nikada biti izvršene.

5.3.1.2.4. Naredba *continue*

U nekim slučajevima prilikom izvršavanja petlji *for* ili *while*, potrebno je nastavljanje petlje, bez izvršavanja nekog njenog dijela. Prethodno je spomenuto, kako se pomoću naredbe *break* prekida izvršavanje petlje. Za razliku od toga, naredba

continue pruža nam mogućnost nastavljanja izvršavanja petlje preskačući određeni dio koda petlje, te vraćanjem na njen početak. Na slici 32. prikazan je primjer korištenja naredbe *continue* u programu kojim se ispisuju svi brojevi od 1 do 50, koji nisu djeljivi sa 5. Važno je napomenuti da se, kao i kod naredbe *break*, mora paziti na smještanje naredbe *continue*, radi izvršavanja svih naredbi u petljama (Kalafatić, Pošćić, Šegvić, Šribar, 2016).

```
for x in range(0, 51):
    if x % 5 == 0:
        continue
    print(x)
```

```
1
2
3
4
6
7
8
9
11
12
13
14
16
17
18
19
21
22
23
24
26
27
28
29
31
32
33
34
36
37
38
39
41
42
43
44
46
47
48
49
```

Slika 32. *Primjer korištenja naredbe continue*

5.3.1.3. Naredba *pass*

U određenim situacijama tijekom programiranja programer se može suočiti s nemogućnosti smišljanja naredbi za jednu granu programa. S obzirom da je Python programski jezik koji ne dopušta grane programa bez naredbi, autori su uveli naredbu *pass* (Kalafatić, Pošćić, Šegvić, Šribar, 2016). Pa tako, kako bi izbjegli prekid rada programa, programeri implementiraju naredbu *pass* u granu za koju nisu sigurni koja će joj biti uloga. Time omogućavaju normalan rad programa za ostale grane. Na slici 33. prikazan je primjer programa u kojem se koristi naredba *pass*.

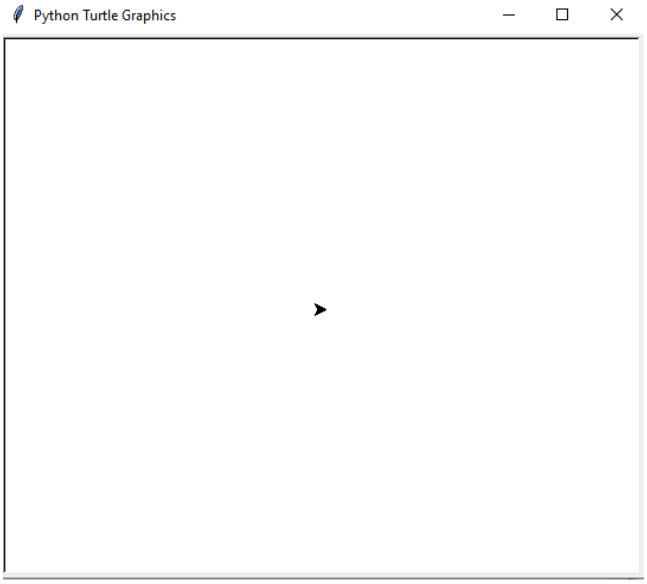
```
x = 2

if x == 0:
    pass

else:
    rezultat = 1 / x
    print ('rezulat =', rezultat)
```

```
rezulat = 0.5
>>>
```

Slika 33. *Primjer korištenja naredbe pass*

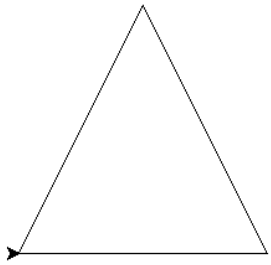
5.3.1.4. Crtanje uz pomoć kornjače

Jedna od brojnih mogućnosti programskoga jezika *Python*, jest i crtanje. Python omogućava crtanje koje se temelji na modulu *turtle*. „Programsko sučelje tog modula temelji se na konceptu kornjače (engl. *turtle*) koja prilikom kretanja po ekranu svojom olovkom može ostavljati trag.“ (Kalafatić, Pošćić, Šegvić, Šribar, 2016: 98). Na slici 34. prikazano je sučelje *Pythonove* kornjačine grafike. Modul *turtle* omogućava nekoliko funkcija, a neke od njih su podizanje i spuštanje olovke i pomicanje kornjače u koordinatnom sustavu. Crtanje započinje postavljanjem kornjače na željeni početak krivulje. Zatim se spušta kornjačina olovka te zadaju koordinate vrhova krivulje, kako bi se kornjača kretala po stranicama te krivulje i usporedno ih iscrtavala olovkom. Primjer crtanja jednakostraničnog trokuta uz pomoć kornjače prikazan je na slici 35.



Slika 34. *Python Turtle Graphics sučelje*

```
import turtle
turtle.clear()
turtle.up()
turtle.goto(-100,0)
turtle.down()
turtle.goto(100,0)
turtle.goto(0,200)
turtle.goto(-100,0)
```



Slika 35. *Program za crtanje jednakostraničnog trokuta uz pomoć kornjače*

5.4. OBJEKTU ORIJENTIRANO PROGRAMIRANJE

Iako je razvoj strukturnog programiranja uvelike olakšao izradu programa, postupno se uvelo i objektu orijentirano programiranje kako bi se dodatno olakšala njihova izrada. „Osnovna je zamisao objektno usmjerenog programiranja spajanje strukture podataka i funkcija koje nad njima djeluju u jedan objekt (engl. *object*).“ (Budin, Brođanac, Markučić, Perić, 2012: 4). Za razliku od proceduralnog programiranja, u kojem se izvođenje programa svodi na pozivanje funkcija, kod objektu orijentiranog programiranja izvođenje programa odvija se stvaranjem i međusobnom suradnjom objekata. Objektu orijentirani programski jezici prvenstveno su namijenjeni izradi profesionalnih programa. Na takvim programima najčešće rade desetci pa čak i stotine programera istovremeno. Stoga se početnicima u programiranju ne preporuča njegovo korištenje, jer bi potrošili puno vremena čak i za one najjednostavnije programe (Budin, Brođanac, Markučić, Perić, 2012). Najpoznatiji objektu orijentirani programski jezici danas jesu *C++*, *Java* i *C#*. Glavna ideja objektu orijentiranog programiranja jest ta da se problem rješava razdjeljivanjem na što manje logičke cjeline. Za to se koriste klase (razredi) i objekti koji će biti pobliže objašnjeni u sljedećem potpoglavlju.

5.4.1. Klase i objekti

Klasa je „osnovna programska cjelina kod objektno orijentiranoga načina programiranja“ (Hruška, 2019: 46). Za kreiranje klase koristi se ključna riječ *class*, nakon koje se navodi ime klase prema vlastitom izboru, a unutar nje upisuju se atributi i metode. Na slici 36. prikazan je osnovni primjer kreiranja klase.

```
class Osoba:
    ime = 'Ana'
```

Slika 36. *Osnovni primjer kreiranja klase*

Unutar svake klase zapisana je shema na temelju kojega se kreiraju objekti. Objekt klase sadrži informacije o toj klasi. Objekti se kreiraju tako što se imenu objekta pridruži ime klase: imeObjekta = imeRazreda (). „Objekt je jedinka koja osim što ima stanje (…) u sebi uključuje i ponašanje – operacije koje je objekt sposoban obaviti.“ (Kalafatić, Pošćić, Šegvić, Šribar, 2016: 324).

Uz to, svaka se klasa sastoji od atributa i metoda. Atributima se objekt opisuje različitim podatcima, a svaki će objekt sadržavati kopiju svih atributa. Dakle, drugim riječima, atributi su svojstva klase. Metode su funkcije koje se pozivaju nad nekim objektom kojem je ta funkcija pridružena. Takvo se pridruživanje vrši pomoću parametra *self* (Hruška, 2019). Metode se pozivaju na sljedeći način: imeObjekta.imeMetode (). Na slici 37. prikazan je primjer kreiranja klase sa pripadajućim objektima i metodama, a na slici 38. rezultat ispisa.

```
class Osoba :
   def hodaj (self) :
      print ('Hodaj!')

   def trci (self) :
      print ('Trci!')

Ana = Osoba ()

Ana.hodaj ()
Ana.trci ()
```

Slika 37. *Kreiranje klase s pripadnim metodama i objektom*

```
Hodaj!
Trci!
>>>
```

Slika 38. *Rezultat ispisa kreiranja klase s pripadnim metodama i objektom*

Nad nekim objektom može se primijeniti i skup metoda. Takav skup metoda naziva se sučelje objekta, a u *Pythonu* se još naziva i protokol (Kalafatić, Pošćić, Šegvić, Šribar, 2016). Protokoli definiraju odgovornosti objekta, odnosno operacije koje objekt može i mora izvršiti.

6. PRIMJENA PYTHONA U OSNOVNOJ ŠKOLI

U poučavanju programiranja važno je obratiti pozornost na to kako su učenici povezani s računalom te izabrati prvi programski jezik kojim će učenici uči u svijet programiranja. Jedni od najvažnijih faktora koji utječu na učenikov pristup učenju jesu učitelj i sadržaji koje učitelji poučavaju (Bubica, Mladenović, Boljat, 2013). Učenici će svakako biti motiviraniji ukoliko im je sadržaj blizak i zanimljiv. U ovom će poglavlju biti prikazani moji autorski programi izrađeni u programskom jeziku Pythonu, a namijenjeni su učenicima 8. razreda osnovne škole. Svi programi namijenjeni su učenju programiranja, a posljednji se program može koristiti i u korelaciji s nastavnim predmetom Matematika.

Prvi program namijenjen je učenju programiranja, a temelji se na usvajanju ugrađenih funkcija sorted() i sort(), odnosno sortiranju podataka u Pythonu. Podatke je vrlo važno ispravno razvrstati, tj. sortirati, jer ćemo na taj način, podatak koji nam je potreban za obrađivanje ili pretraživanje, lakše pronaći i dohvatiti u zadanom nizu ili skupu podataka. Stoga se koriste ugrađene funkcije sorted() i sort() kako bi se određeni skup ili niz podataka sortirao određenim redoslijedom. U ovom primjeru sortiraju se brojevi zadanog polja od najmanjeg prema najvećem, prvo ugrađenom funkcijom sorted(), a zatim ugrađenom funkcijom sort(). Na slici 39. nalazi se postupak izvršavanja programa pseudokodom, na slici 40. prikazan je program koji omogućava rješavanje prethodno navedenog problema, a na slici 41. ekran koji prikazuje izvršavanje tog programa.

```
početak
brojevi := [3,5,2,1]
izlaz(brojevi)
izlaz(sortirani brojevi) #(sorted(brojevi))
izlaz brojevi
sortiraj brojevi
izlaz brojevi
brojevi := [3,5,2,1]
izlaz brojevi
sortiraj brojevi #(brojevi.sort())
izlaz(brojevi)
kraj
```

Slika 39. *Pseudokod programa koji omogućava sortiranje podataka u polju pomoću ugrađenih funkcija sorted() i sort()*

```
*1. Koristenje_ugradenih_funkcija - program.py - C:\Users\Dora\Desktop\DIPLOMSKI\programi\1. Koristenje_...
File  Edit  Format  Run  Options  Window  Help
# ----------------------------------------------------------------------------------------------------
# Autorica: (c) Dora Turčin, 2020.
# ----------------------------------------------------------------------------------------------------

#Korištenje ugrađene funkcije sorted()

#Inicijalizacija polja brojevi sa zadanim vrijednostima
brojevi = [3,5,2,1]
#Ispis brojeva
print(brojevi)
#Ispis sortiranog polja brojevi korištenjem ugrađene funkcije sorted()
#Zadani je način sortiranje od najmanjeg prema najvećem
#Funkcija sorted() vraća sortiranu listu, ali ne dodjeljuje ju automatski varijabli na kojoj se obrađuje sortiranje
print(sorted(brojevi))
#Ovdje je prikazano da su brojevi i dalje sortirani kako je na početku zadano:  [3,5,2,1]
print(brojevi)
#Ukoliko želimo promijeniti stanje naše varijable moramo joj pridodijeliti vrijednost koju funkcija sorted() vraća
brojevi = sorted(brojevi)
print(brojevi)


#Korištenje ugrađene funckije sort()

#Inicijalizacija liste brojevi sa zadanim vrijednostima
brojevi = [3,5,2,1]
#Ispis brojeva
print(brojevi)
#Ispis sortiranog polja korištenjem ugrađene funckije sort()
#Zadani je način sortiranje od najmanjeg prema najvećem
#Funkcija sort() poziva se na razini polja i mijenja raspored samog polja
brojevi.sort()
print(brojevi)
Ln: 1  Col: 0
```

Slika 40. *Program koji omogućava sortiranje podataka u polju pomoću ugrađenih funkcija sorted() i sort()*

```
Python 3.8.5 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Dora\Desktop\DIPLOMSKI\programi\1. Koristenje_ugradenih_funkcija.py
[3, 5, 2, 1]
[1, 2, 3, 5]
[3, 5, 2, 1]
[1, 2, 3, 5]
[3, 5, 2, 1]
[1, 2, 3, 5]
>>>
Ln: 11  Col: 4
```

Slika 41. *Ekran izvršavanja programa sortiranja pomoću ugrađenih funkcija sorted() i sort()*

Drugi je program, također, namijenjen učenju sortiranja, no u ovom slučaju uz pomoć algoritma za sortiranje. Prvi algoritam koji se koristi u programu jest algoritam za spoznavanje koji je od zadanih dvaju brojeva manji. U definiranoj funkciji manji(), brojevi x i y uspoređuju se unutar petlje if, te se vraća manji broj. U ovom je programu također definiran i algoritam sortiranja podataka *bubble sort*, koji služi za sortiranje polja rastuće, odnosno od najmanje vrijednosti prema najvećoj vrijednosti. Taj algoritam svojim izvršavanjem može pridonijeti lakšoj obradi i pretraživanju podataka u nekim kompleksnijim programima. Postupak izvršavanja cijelog programa prikazan je pseudokodom na slici 42., a na slici 43. prikazan je program koji omogućava sortiranje podataka pomoću algoritma za sortiranje. Na slici 44. prikazan je ekran izvršavanja programa sortiranja pomoću algoritma za sortiranje.

```
početak

definiraj funkciju manji(x,y)
        ako je x manji od y (x<y)
                vrati x
        inače
                vrati y

a:=2
b:=1

manji_broj=manji(a,b)
izlaz (manji_broj)

definiraj funkciju sortiraj_polje_rastuce(polje)
        za i u rasponu dužine polja
                za j u rasponu dužine polja
                        ako je polje[j] veće od polje[i]
                                tempt:=polje[i]
                                polje[i]:=polje[j]
                                polje[j]=temp
        vrati polje

polje:=[4,2,7,123,7,4,2,1]
izlaz(sortiraj_polje_rastuce(polje))

kraj
```

Slika 42. *Pseudokod programa koji omogućava sortiranje podataka pomoću algoritma za sortiranje*

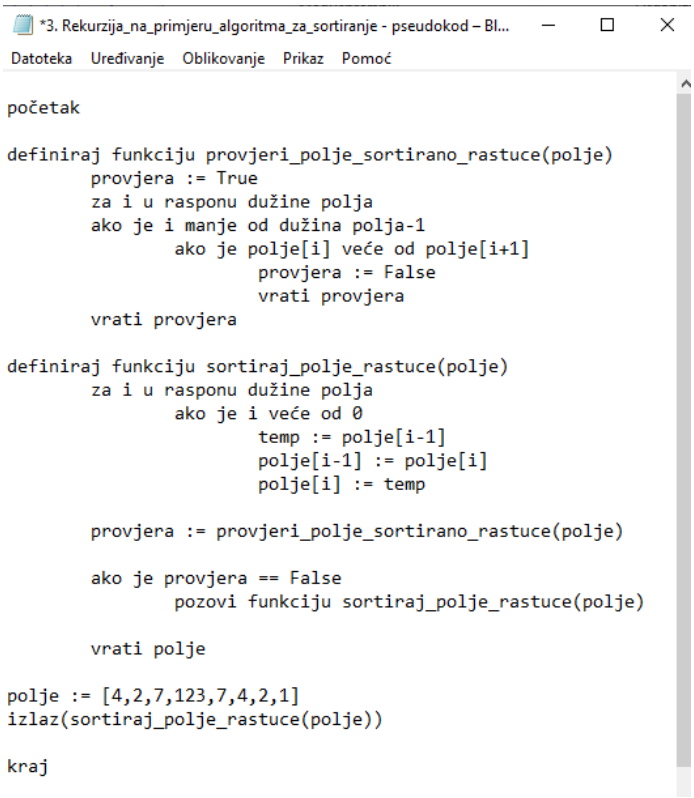
```
*2. Sortiranje_algoritam - program.py - C:\Users\Dora\Desktop\DIPLOMSKI\programi\2. Sorti...
File  Edit  Format  Run  Options  Window  Help
# ----------------------------------------------------------------------------------------------
# Autorica: (c) Dora Turčin, 2020.
# U ovom programu korišten je algoritam sortiranja podataka bubble sort.
# ----------------------------------------------------------------------------------------------

#Algoritam za spoznaju koji od dvaju danih broja je manji.
def manji(x,y):
    if x < y:
        return x
    else:
        return y

#Primjer korištenja algoritma
#Inicijalizacija dviju varijabli s brojčanim vrijednostima
a = 2
b = 1

manji_broj = manji(a,b)
print(manji_broj)

#Algoritam za sortiranje danog polja.
def sortiraj_polje_rastuce(polje):
    for i in range(len(polje)):
        for j in range(len(polje)):
            if polje[j] > polje[i]:
                temp = polje[i]
                polje[i] = polje[j]
                polje[j] = temp

    return polje

#Primjer korištenja algoritma
polje = [4,2,7,123,7,4,2,1]
print(sortiraj_polje_rastuce(polje))
```

Slika 43. *Program koji omogućava sortiranje podataka pomoću definiranog algoritma za sortiranje*

```
Python 3.8.5 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Dora\Desktop\DIPLOMSKI\programi\2. Sortiranje_algoritam.py =
1
[1, 2, 2, 4, 4, 7, 7, 123]
>>>
```

Slika 44. *Ekran izvršavanja programa sortiranja pomoću algoritma za sortiranje*

Treći program temelji se na usvajanju rekurzije na primjeru algoritma za sortiranje. Postupak izvršavanja ovog programa prikazan je na slici 45. Rekurzija omogućava ponavljanje skupine naredbi pomoću funkcija koje pozivaju same sebe, što je vidljivo na ovom primjeru (Slika 46.). U ovom programu zadano je polje koje se pozivom funkcije sortiraj_polje_rastuce(polje) sortira od najmanje vrijednosti prema najvećoj. Kada polje dođe do kraja sortiranja, ono prolazi kroz provjeru pozivom funkcije provjeri_polje_sortirano_rastuce(polje). Ukoliko se prolaskom kroz tu funkciju uspostavi da polje nije u potpunosti sortirano, program se ponovno vraća na funkciju sortiraj_polje_rastuce(polje), te se ponavlja sortiranje. Program će se tako odvijati dokle god provjera ne bude istinita, odnosno jednaka True. U trenu kada provjera bude jednaka True, program ispisuje sortirano polje. Na slici 47. prikazan je ekran izvršavanja programa koji koristi rekurziju u algoritmu za sortiranje.




```
početak

definiraj funkciju provjeri_polje_sortirano_rastuce(polje)
        provjera := True
        za i u rasponu dužine polja
        ako je i manje od dužina polja-1
                ako je polje[i] veće od polje[i+1]
                        provjera := False
                        vrati provjera
        vrati provjera

definiraj funkciju sortiraj_polje_rastuce(polje)
        za i u rasponu dužine polja
                ako je i veće od 0
                        temp := polje[i-1]
                        polje[i-1] := polje[i]
                        polje[i] := temp

        provjera := provjeri_polje_sortirano_rastuce(polje)

        ako je provjera == False
                pozovi funkciju sortiraj_polje_rastuce(polje)

        vrati polje

polje := [4,2,7,123,7,4,2,1]
izlaz(sortiraj_polje_rastuce(polje))

kraj
```

Slika 45. *Pseudokod programa koji koristi rekurziju u algoritmu za sortiranje*

```
*3. Rekurzija_na_primjeru_algoritma_za_sortiranje - program.py - C:\Users\Dora\Desktop\DI
File  Edit  Format  Run  Options  Window  Help
# ------------------------------------------------------------------------------
# Autorica: (c) Dora Turčin, 2020.
# ------------------------------------------------------------------------------

#Funkcija za provjeru je li polje sortirano
def provjeri_polje_sortirano_rastuce(polje):
    provjera = True
    for i in range(len(polje)):
        if i < len(polje) - 1:
            if polje[i] > polje[i+1]:
                provjera = False
                return provjera
    return provjera

#Algoritam za sortiranje danog polja izveden pomoću rekurzije.
def sortiraj_polje_rastuce(polje):
    for i in range(len(polje)):
        if i > 0:
            if polje[i] < polje[i-1]:
                temp = polje[i-1]
                polje[i-1]=polje[i]
                polje[i] = temp

    provjera = provjeri_polje_sortirano_rastuce(polje)

    if provjera == False:
        sortiraj_polje_rastuce(polje)

    return polje

#Primjer korištenja algoritma
polje = [4,2,7,123,7,4,2,1]
print(sortiraj_polje_rastuce(polje))
```

Slika 46. *Program koji koristi rekurziju u algoritmu za sortiranje*

```
Python 3.8.5 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [M
SC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more info
rmation.
>>>
= RESTART: C:\Users\Dora\Desktop\DIPLOMSKI\programi\3.
Rekurzija_na_primjeru_algoritma_za_sortiranje - program.py
[1, 2, 2, 4, 4, 7, 7, 123]
>>>
```

Slika 47. *Ekran izvršavanja programa koji koristi rekurziju u algoritmu za sortiranje*

Sljedeći program temelji se na kornjačinoj grafici. Uvozom modula turtle, omogućava se crtanje u programskom jeziku Python. U ovom programu, riječ je o učenju pisanja i pozivu funkcija koje će crtati trokut, kvadrat, pravokutnik, krug i mnogokut na temelju prethodno unesenih vrijednosti za duljinu stranica i broja vrhova mnogokuta. Na slici 48. prikazan je postupak izvršavanja programa pseudokodom, a slika 49. prikazuje program koji omogućava rješavanje navedenog problema. Prilikom pokretanja programa prvo se otvara maleni prozor koji od korisnika traži unos duljine stranice geometrijskih likova koji će se crtati (Slika 50.). Nakon što se unese duljine stranice, otvara se novi mali prozor koji traži unos broja vrhova mnogokuta (Slika 51.). Nakon što se unio i taj podatak, u većem prozoru započinje crtanje geometrijskih likova redoslijedom kojim su funkcije u programu pozivane (Slika 52.).

```
početak

iz kornjače uvezi sve

x := 400
x := 200

definiraj funkciju trokut()
        podigni olovku
        postavi poziciju (-x,y)
        postavi boju ispune(plava)
        započni ispunu
        spusti olovku
        za i u rasponu od 3
                pomakni se naprijed za a
                zakreni se ulijevo za 120°
        završi ispunu
        podigni olovku

definiraj funkciju kvadrat()
        podigni olovku
        postavi poziciju (-x,-y)
        postavi boju ispune(crvena)
        započni ispunu
        spusti olovku
        za i u rasponu od 4
                pomakni se naprijed za a
                zakreni se ulijevo za 90°
        završi ispunu
        podigni olovku

definiraj funkciju pravokutnik()
        podigni olovku
        postavi poziciju (-x/6,y)
        postavi boju ispune(ljubičasta)
        započni ispunu
        spusti olovku
        za i u rasponu od 2
                pomakni se naprijed za a
                zakreni se ulijevo za 90°
                pomakni se naprijed za 70
                zakreni se ulijevo za 90°
        završi ispunu
        podigni olovku

definiraj funkciju krug()
        podigni olovku
        postavi poziciju (0,-y)
        postavi boju ispune(narančasta)
        započni ispunu
        spusti olovku
        pozovi naredbu circle(a/2)
        završi ispunu
        podigni olovku

definiraj funkciju mnogokut(a,n)
        podigni olovku
        postavi poziciju (x/2,0)
        postavi boju ispune(zelena)
        započni ispunu
        spusti olovku
        za i u rasponu od n
                pomakni se naprijed za a
                zakreni se ulijevo za 360°/n
        završi ispunu
        sakrij kornjaču

ulaz(a)
ulaz(n)

pozovi funkciju trokut()
pozovi funkciju kvadrat()
pozovi funkciju pravokutnik()
pozovi funkciju krug()
pozovi funkciju mnogokut(a,n)

kraj
```

Slika 48. *Pseudokod programa koji korištenjem kornjačine grafike crta trokut, kvadrat, pravokutnik, krug i mnogokut prema unosu duljine stranice i broja vrhova mnogokuta*

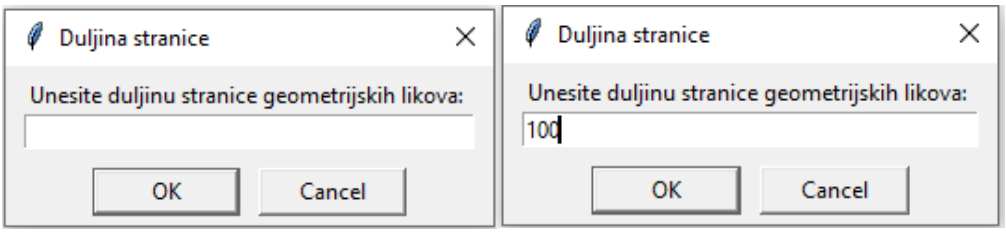
*4. Kornjačina_grafika - program.py - C:\Users\Dora\Desktop\DIPLOMSKI\programi\4. Kornjačina_grat

File Edit Format Run Options Window Help

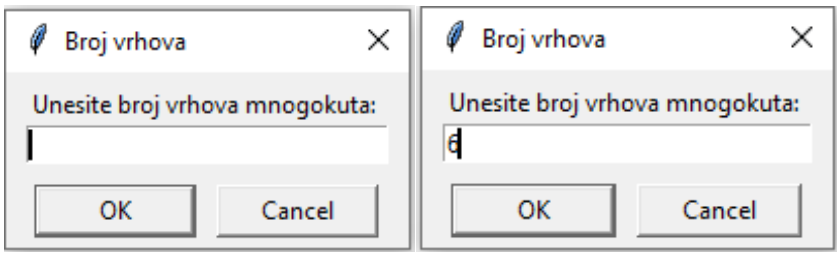
```
# ---------------------------------------------------------------------------------------------
# Autorica: (c) Dora Turčin, 2020.
# ---------------------------------------------------------------------------------------------

from turtle import * # naredba kojom se uvoze sve naredbe modula Turtle

x=400
y=200

# definiramo funkciju kojom će se izvoditi crtanje trokuta
def trokut():
    pu()
    setpos(-x,y)
    fillcolor('blue')
    begin_fill()
    pd()
    for i in  range(3):
        fd(a)
        left(120)
    end_fill()
    pu()

# definiramo funkciju kojom će se izvoditi crtanje kvadrata
def kvadrat():
    pu()
    setpos(-x,-y)
    fillcolor('red')
    begin_fill()
    for i in range(4):
        fd(a)
        left(90)
    end_fill()
    pu()

# definiramo funkciju kojom će se izvoditi crtanje pravokutnika
def pravokutnik():
    pu()
    setpos(-x/6,y)
    fillcolor('purple')
    begin_fill()
    for i in range(2):
        fd(a)
        left(90)
        fd(70)
        left(90)
    end_fill()
    pu()
```

```
# definiramo funkciju kojom će se izvoditi crtanje kruga
def krug():
    pu()
    setpos(0,-y)
    fillcolor('orange')
    begin_fill()
    circle(a/2)
    end_fill()
    pu()

def mnogokut(a,n): # definiramo funkciju kojom će se izvoditi crtanje mnogokuta
    pu() # podižemo olovku
    setpos(x/2,0) # naredba za postavljanje pozicije kornjače, odnosno početne točke crtanja
    fillcolor('green') # postavljamo boju ispune mnogokuta
    begin_fill() # započinjemo ispunjavanje mnogokuta
    for i in range(n): # petljom for definiramo koliko će se puta ponavljati naredbe unutar nje
        fd(a) # naredba za pomicanje kornjače za unesenu vrijednost duljine stranice
        left(360/n) # naredba za rotaciju kornjače s obzirom na broj stranica mnogokuta
    end_fill() # naredba za završavanje ispunjavanja mnogokuta
    hideturtle() # naredba za skrivanje kornjače

# naredba za unos duljine stranice mnogokuta
a=float(textinput('Duljina stranice', 'Unesite duljinu stranice geometrijskih likova: '))

# naredba za unos broja stranica mnogokuta
n=int(textinput('Broj vrhova', 'Unesite broj vrhova mnogokuta: '))

trokut() # pozivamo funkciju za crtanje trokuta
kvadrat() # pozivamo funkciju za crtanje kvadrata
pravokutnik() # pozivamo funkciju za crtanje pravokutnika
krug() # pozivamo funkciju za crtanje kruga
mnogokut(a,n) # pozivamo funkciju za crtanje mnogokuta
```

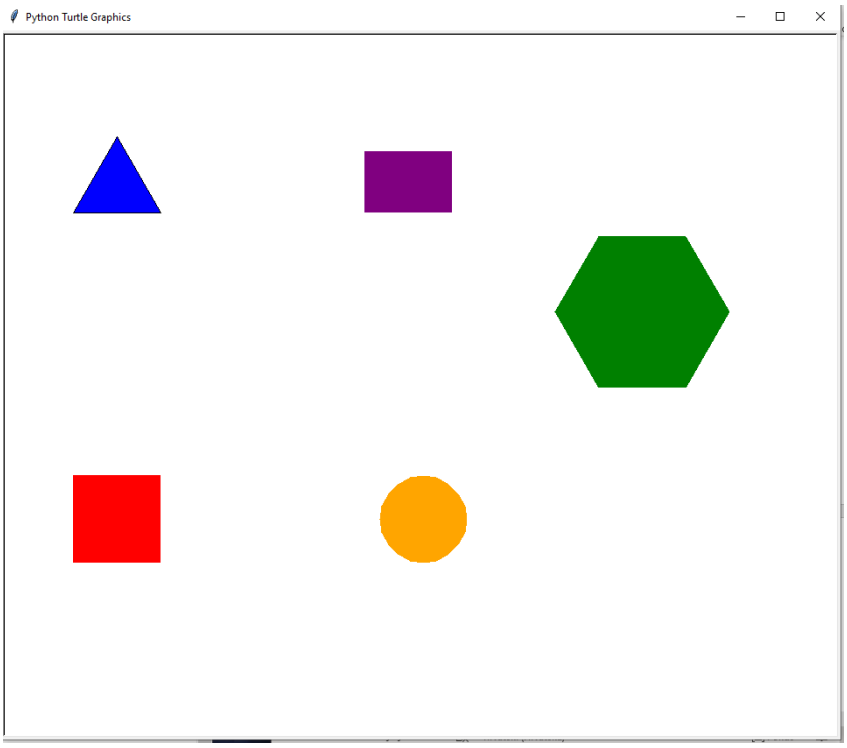
Slika 49. *Program koji korištenjem kornjačine grafike crta trokut, kvadrat, pravokutnik, krug i mnogokut prema unosu duljine stranice i broja vrhova mnogokuta*



Slika 50. *Prozor za unos duljine stranice geometrijskih likova koje će program crtati*



Slika 51. *Prozor za unos broja vrhova mnogokuta*



Slika 52. *Crteži nacrtani pomoću programa na temelju unesenih vrijednosti*

Sljedeći program osmišljen je u svrhu ponavljanja nastavnog sadržaja kvadriranje iz nastavnog predmeta Matematika (Slika 52.), a na slici 51. prikazano je izvršavanje programa pseudokodom. Program sadrži jednostavno grafičko sučelje izrađeno uvozom modula Tkinter u programski jezik Python. Na početnom zaslonu programa (Slika 53.) nalaze se dva gumba: „Kvadriranje racionalnih brojeva“ te „Kvadriranje umnoška“. Pritiskom na prvi gumb, „Kvadriranje racionalnih brojeva“, otvara se novi zaslon (Slika 54.), koji sadrži formulu i objašnjenje izračunavanja kvadrata broja. Osim formule, taj zaslon sadrži i zadatak koji traži unos broja kojeg želimo kvadrirati, polje za unos broja te gumb „Kvadriraj“. Kada se u polje unosa upiše broj koji želimo kvadrirati, mora se pritisnuti gumb „Kvadriraj“, nakon što je gumb pritisnut, ispod će se ispisati postupak kvadriranja upisanog broja te rezultat (Slika 55.). Taj se prozor zatvara pritiskom na „x“ u desnom gornjem kutu. Zatim se vraćamo na početni zaslon. Pritiskom na drugi gumb, „Kvadriranje umnoška“, otvara se novi zaslon (Slika 56.), koji sadrži formulu i objašnjenje izračunavanja kvadrata umnoška dvaju brojeva. Taj zaslon također sadrži zadatak koji traži unos dvaju brojeva, dva polja za njihov unos te gumb „Kvadriraj umnozak“. Kada se u oba polja upiše po jedan broj te pritisne gumb „Kvadriraj umnozak“, program ispod izbacuje postupak izračunavanja kvadrata umnoška te ukupan rezultat (Slika 57.).

```
početak

otvori glavni zaslon ('MATEMATIKA 8 - KVADRIRANJE')
izlaz('Odaberi lekciju koju želiš vježbati:')
ako je klik na gumb Kvadriranje racionalnih brojeva
        onda
                otvori zaslon ('Kvadriranje racionalnih brojeva')
                izlaz('Kvadrat racionalnog broja jednak je umnošku tog broja sa samim sobom.')
                izlaz('a^2=a*a)
                izlaz('Unesi broj koji želiš kvadrirati')
                ulaz(broj)
                kvadrat:=broj**2
                ako je klik na gumb Kvadriraj
                        onda
                                izlaz(broj+'^2='+broj*broj+'='+kvadrat)
ako je klik na gumb Kvadriranje umnoška
        onda
                otvori zaslon ('Kvadriranje umnoška')
                izlaz('Kvadrat umnoška dvaju brojeva jednak je umnošku kvadrata tih brojeva.')
                izlaz('(a*b)^2=a^2*b^2')
                izlaz('Unesi prvi broj')
                ulaz(a)
                izlaz('Unesi drugi broj')
                ulaz(b)
                mnozi=(a*b)**2
                ako je klik na gumb Kvadriraj umnožak
                        onda
                                izlaz('(' + a + '*' + b + ')^2=' + a + '^2 * ' + b + '^2 = ' + mnozi)

kraj
```

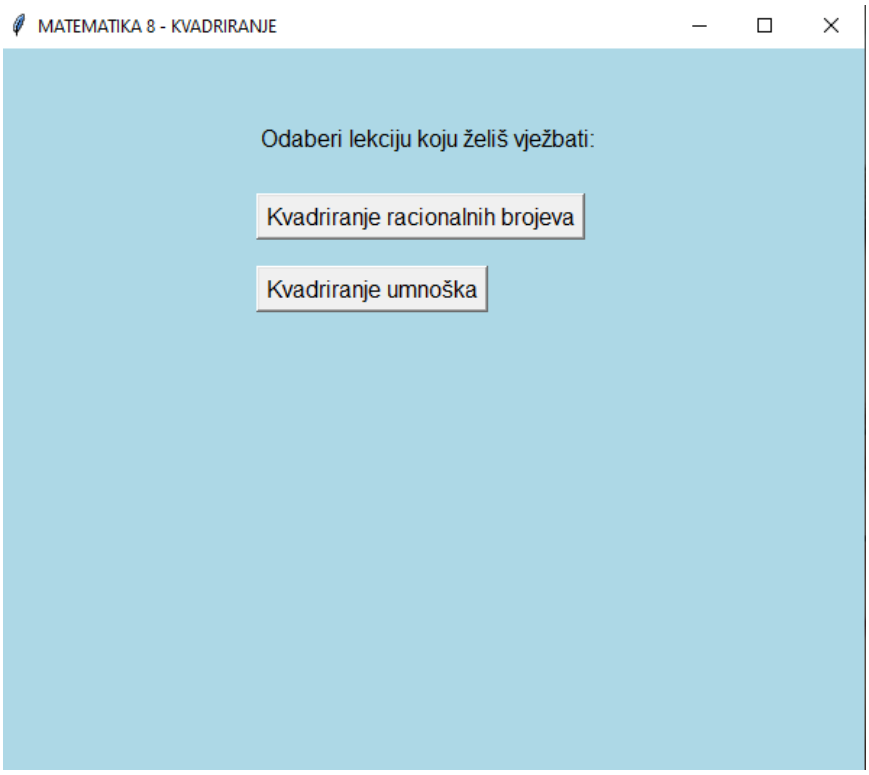
Slika 53. *Pseudokod programa koji sadrži jednostavno grafičko sučelje u Tkinteru te služi za ponavljanje kvadriranja u Matematici*

5. Tkinter - program.py - C:\Users\Dora\Desktop\DIPLOMSKI\programi\5. Tkinter - program.py (3.8.5)

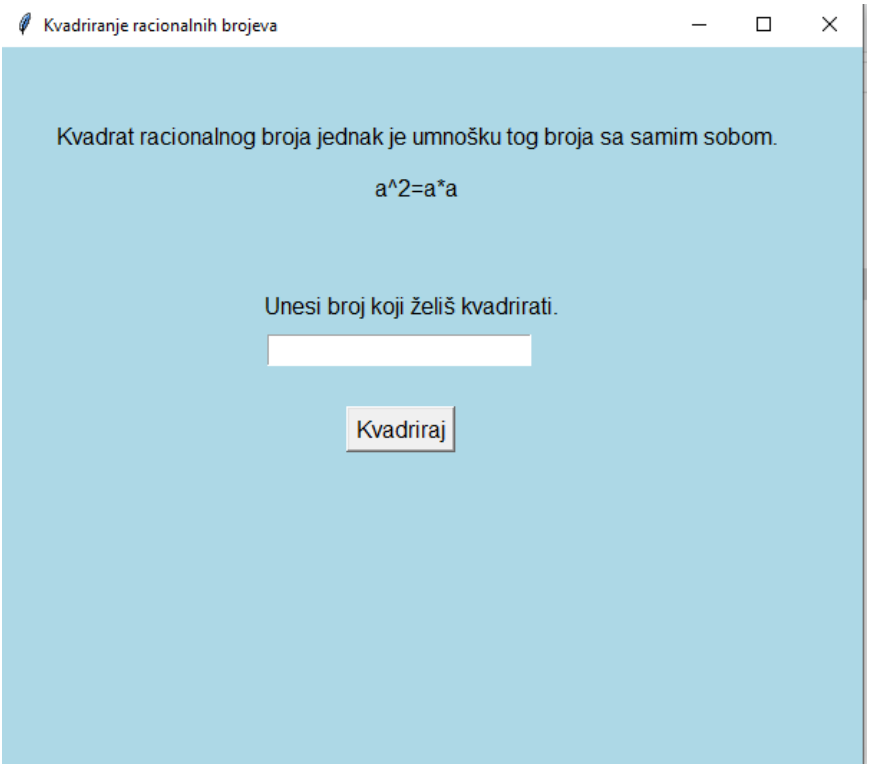
File Edit Format Run Options Window Help

```
# ---------------------------------------------------------------------------------------------------------------
# Autorica: (c) Dora Turčin, 2020.
# ---------------------------------------------------------------------------------------------------------------

from tkinter import *

# definiranje funkcije za kreiranje zaslona sa zadanom konfiguracijom
def zaslon(naslov):
    zaslon=Tk()
    zaslon.geometry('600x500')
    zaslon.config(bg='lightblue')
    zaslon.title(naslov)
    return zaslon

# definiranje funkcije za kreiranje oznake sa zadanom konfiguracijom
def label(zaslon, tekst, x, y):
    label=Label(zaslon, text=tekst, bg='lightblue', font=12)
    label.place(x=x, y=y)

# definiranje funkcije za kreiranje gumba s definiranom konfiguracijom, koji će pritiskom na njega izvršavati određenu funkciju
def gumb(zaslon, tekst, x, y, naredba):
    gumb=Button(zaslon, text=tekst,  font=12, command=naredba)
    gumb.place(x=x,y=y)

# definiranje funkcije za kreiranje polja za unos vrijednosti
def unos(zaslon, x, y):
   unos=Entry(zaslon, font=12)
   unos.place(x=x,y=y)
   return unos

# definiranje funkcije za kreiranje novog zaslona koja će se pokretati pritiskom istoimenog gumba
def kvadriranje_racionalnih_brojeva():
    _zaslon = zaslon('Kvadriranje racionalnih brojeva')
    label(_zaslon, 'Kvadrat racionalnog broja jednak je umnošku tog broja sa samim sobom.\n\na^2=a*a', 35,50)
    label(_zaslon, '\nUnesi broj koji želiš kvadrirati.', 180,150)
    _unos=unos(_zaslon, 185,200)
    # definiranje funkcije za izračunavanje kvadrata unesenog broja, koja će se izvršavati pritiskom na gumb
    def kvadrac():
        broj=int(_unos.get())
        kvadrat=broj**2
        rez=label(_zaslon, str(broj) + '^2 = ' + str(broj) + '*' + str(broj) + ' = ' + str(kvadrat), 230,300)
    gumb(_zaslon, 'Kvadriraj',240, 250, kvadrac)


# definiranje funkcije za kreiranje novog zaslona koja će se pokretati pritiskom istoimenog gumba
def kvadriranje_umnoska():
    _zaslon = zaslon('Kvadriranje umnoška')
    label(_zaslon, 'Kvadrat umnoška dvaju brojeva jednak je umnošku kvadrata tih brojeva.\n\n(a*b)^2=a^2*b^2', 35, 50)
    label(_zaslon, 'Unesi prvi broj:', 130,150)
    _unos=unos(_zaslon, 270,150)
    label(_zaslon, 'Unesi drugi broj:', 130,200)
    _unos2=unos(_zaslon,270,200)
    # definiranje funkcije za izračunavanje kvadrata umnoška dvaju unesenih brojeva, koja će se izvršavati pritiskom na gumb
    def umnozak():
        a=int(_unos.get())
        b=int(_unos2.get())
        mnozi=(a*b)**2
        rez=label(_zaslon,  '(' + str(a) + '*' + str(b) +')^2 = ' + str(a) + '^2 * ' + str(b)+ '^2 = ' + str(mnozi), 270, 300)
    gumb(_zaslon, 'Kvadriraj umnožak', 290, 250, umnozak)

# kreiranje glavnog zaslona 'MATEMATIKA 8 - Kvadriranje' pomoću prethodno definirane funkcije
glavni_zaslon = zaslon('MATEMATIKA 8 - KVADRIRANJE')
# kreiranje tekstualne oznake i gumba koji će se nalaziti na glavnom zaslonu sa zadanim koordinatama
label(glavni_zaslon, 'Odaberi lekciju koju želiš vježbati:', 180,50)
gumb(glavni_zaslon, 'Kvadriranje racionalnih brojeva', 180,100, kvadriranje_racionalnih_brojeva)
gumb(glavni_zaslon, 'Kvadriranje umnoška',180, 150, kvadriranje_umnoska)
```

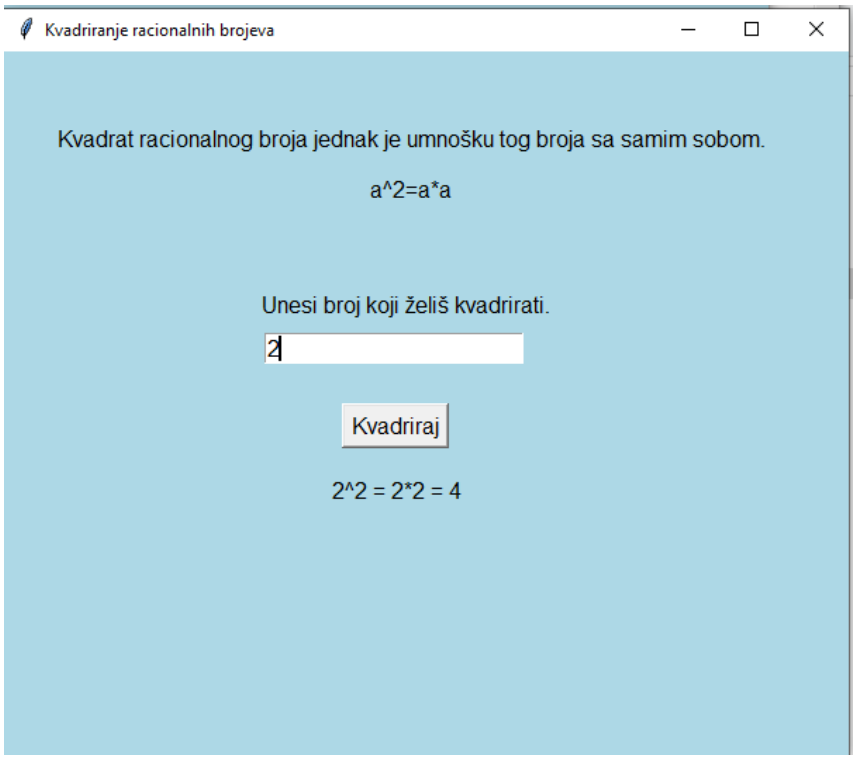
Slika 54. *Program koji sadrži jednostavno grafičko sučelje u Tkinteru te služi za ponavljanje kvadriranja u Matematici*



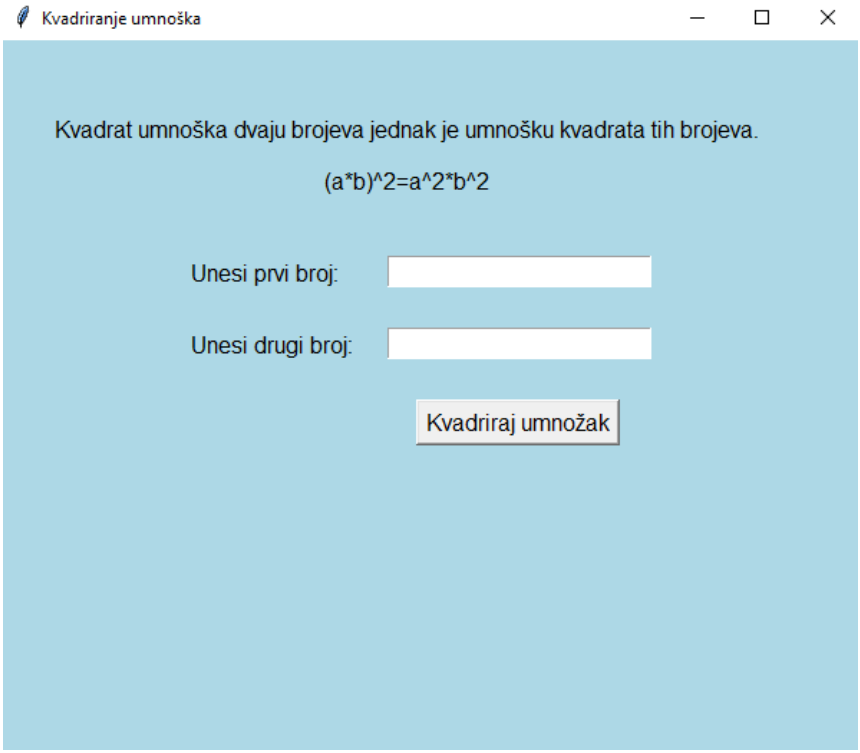
Slika 55. *Početni zaslon programa za ponavljanje kvadriranja u Matematici*



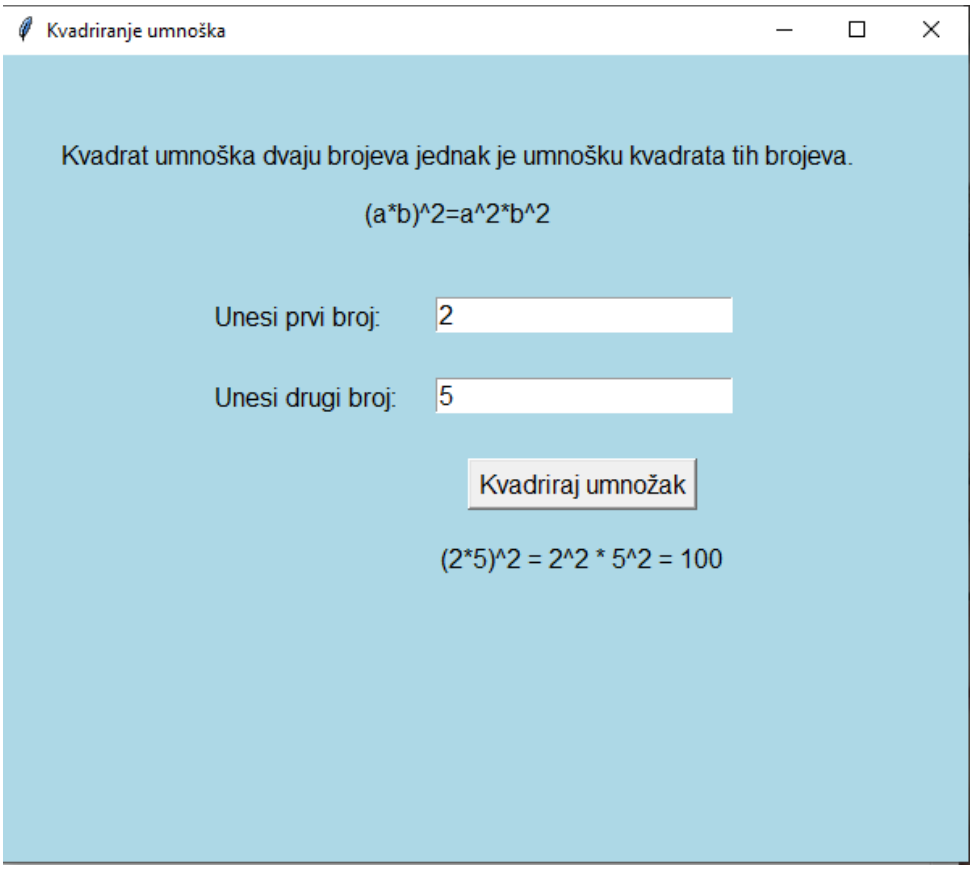
Slika 56. *Zaslon koji se otvori pritiskom na prvi gumb početnog zaslona (Kvadriranje racionalnih brojeva)*



Slika 57. *Prilikom upisa broja i pritiskom na gumb Kvadriraj, program ispisuje rezultat*



Slika 58. *Zaslon koji se otvori pritiskom na drugi gumb početnog zaslona (Kvadriranje umnoška)*



Slika 59. *Prilikom upisa dvaju brojeva i pritiskom na gumb Kvadriraj umnozak, program ispisuje rezultat*

7. ZAKLJUČAK

S obzirom da se u današnje vrijeme djeca od najranije dobi susreću s informacijskom i komunikacijskom tehnologijom te ju sve više koriste, iznimno je važno da ih se što ranije educira o prednostima njena korištenja. Donošenjem novog kurikuluma iz nastavnog predmeta Informatika, omogućava se detaljnija razrada samih ciljeva i sadržaja učenja tog predmeta u sklopu četiri domene. Kao što je već prethodno spomenuto, za ovaj rad najvažnija je domena Računalno razmišljanje i programiranje.

Važno je razvijati računalno razmišljanje jer se samim time omogućava razvoj pristupa rješavanju problema koji se primjenjuje na računalu. To omogućava da učenici ne budu isključivo korisnici, već i stvaratelji raznih računalnih alata. Osim navedenog, računalno razmišljanje smatrano je vještinom koja potiče brojne pozitivne osobnosti, kao što su sustavnost, preciznost i analitičnost, odnosno raščlanjivanje jednog složenog problema, na više jednostavnih. Tijekom programiranja važno je poznavanje standardnih postupaka razvoja programa, no, iznimno je važno i biti inovativan i poduzetan. Te se osobine sve dublje razvijaju, što se više čovjek bavi programiranjem. Osim toga, programiranje dakako utječe i na razvoj samopouzdanja i upornosti u rješavanju problema, pa tako i preciznosti u ispravljanju pogrešaka koje se nerijetko pojavljuju. Računalno razmišljanje i programiranje važno je razvijati jer je njihova primjena široka te seže u gotovo sve aspekte svakodnevnom života.

Kada je riječ o učenju i poučavanju programiranja, važno je pažljivo birati programski jezik kojim će učenici započeti svoj razvoj programerskih vještina i računalnog razmišljanja. U učenju programiranja najčešće se kreće od vizualnih programskih jezika kao što je Scratch, koji se koristi blokovima naredbi. Nakon što se usvoji takav način programiranja, potrebno je uči malo dublje, u samo pisanje kôda. Tu se javlja programski jezik Python koji, osim što je besplatan programski jezik, ima i druge brojne prednosti. Python je programski jezik čija je sintaksa slična engleskom jeziku, stoga je iznimno lagan za početno učenje programiranja. Također, ističe se time što funkcionira na različitim platformama, kao što su Windows, Mac, Linux itd. Python ima široku primjenu, pa se tako koristi u razvoju weba, raznih softvera, u matematici, skriptiranju sustava te u poučavanju. U njemu se programi mogu pisati bez složenih detalja, a oni se mogu uvoditi postupno. Ta je karakteristika iznimno

važna u poučavanju programiranja u osnovnoj školi, kada učenici ne mogu primiti previše apstraktnih i novih informacija odjednom. Stoga se, pri učenju programiranja u Pythonu, njegove funkcionalnosti mogu uvoditi po potrebi i postupno. To je jedna od najvećih prednosti koje Python pruža, kada je riječ o učenju i poučavanju programiranja.

Kao što je prethodno navedeno, Python pruža brojne funkcionalnosti. Neke od funkcionalnosti koje pruža jesu razni tipovi podataka, varijable, uvjetno izvođenje, petlje, crtanje uz pomoć kornjačine grafike te izradu grafičkog sučelja uz pomoć modula tkinter. Sve te funkcionalnosti mogu se kreativno primijeniti u poučavanju programiranja u osnovnoj školi. Osim što se mogu primijeniti u poučavanju i učenju programiranja, mogu se primijeniti i u korelaciji s drugim nastavnim predmetima. Pritom osim što učenici uče programirati, ponavljaju sadržaje drugih predmeta, a nerijetko se i zabave, ako im se programiranje približi na zabavan, njima blizak, način.

LITERATURA

1. Babić, M., Bubica, N., Leko, S., Dimovski, Z., Stančić, M., Mihočka, N., Ružić, I., Vejnović, B. (2020). *#mojportal7 – udžbenik informatike u sedmom razredu osnovne škole.* Zagreb: Školska knjiga https://www.e-sfera.hr/prelistaj-udzbenik/3c89a769-23ee-4b7e-957a-706cd1855373 (7.8.2020.)
2. Bubica, N., Mladenović, M., Boljat, I. (2013). *Programiranje kao alat za razvoj apstraktnog mišljenja*. https://bib.irb.hr/datoteka/702093.Programiranje_kao_alat_za_razvoj_apstraktnog_miljenja-CUC-zbornik.pdf (4.2.2020.)
3. Budin, L., Brođanac, P., Markučić, Z., Perić, S. (2012). *Rješavanje problema programiranjem u Python-u*. Zagreb: Element
4. Christensson, P. (2006). *BASIC Definition*. https://techterms.com (17.2.2020.)
5. Debian https://www.debian.org/releases/buster/amd64/ch01s02.en.html (26.6.2020.)
6. Downey, A.B. (2014). *Naučite Python*. Zagreb: Dobar plan
7. Durđević, I. (2014). *Procjene studenata Učiteljskog studija o tri računalna programa namijenjena malim početnicima u programiranju. Radovi Zavoda za znanstveni i umjetnički rad u Požegi*, No. 3, 93-108 https://hrcak.srce.hr/133854 (17.2.2020.)
8. Ganeshan, P. Radha (1999). *Pascal Programming*. New Delhi: New Age International https://www.academia.edu/31407245/_P._Radha_Ganeshan_Pascal_Programming_BookZZ.org_ (20.2.2020.)
9. Glavan, F. (2000). *MSWLogo: Početnica naprednog programiranja.* Zagreb: Alfej d.o.o.
10. Hruška, M. (2019). *Programiranje u Pythonu*. Zagreb: SRCE https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/d460_polaznik.pdf (14.7.2020.)
11. Hrvatski leksikografski zavod (2020). Hrvatska enciklopedija, mrežno izdanje. https://www.enciklopedija.hr/natuknica.aspx?ID=50558#top (10.8.2020.)
12. Kalafatić, Z., Poščić, A., Šegvić, S., Šribar, J. (2016). *Python za znatiželjne – sasvim drukčiji pogled na programiranje*. Zagreb: Element

13. Maleš, L., Redžepagić, J., Rakijašić, D. (2018). *Priručnik „Programiranje“*. Zagreb: Hrvatska akademska i istraživačka mreža – CARNET https://pilot.e-skole.hr/wp-content/uploads/2018/08/Prirucnik_Programiranje.pdf (30.7.2020.)
14. Ministarstvo znanosti i obrazovanja (2018). *Odluka o donošenju kurikuluma za nastavni predmet informatike za osnovne škole i gimnazije u Republici Hrvatskoj*. Zagreb: Narodne novine https://narodne-novine.nn.hr/clanci/sluzbeni/full/2018_03_22_436.html (4.2.2020.)
15. Lewis, C.M. (2010). *How Programming Environment Shapes Perception, Learning and Goals: Logo vs. Scratch.* http://ims.mii.lt/ims/konferenciju_medziaga/SIGCSE'10/docs/p346.pdf (4.2.2020.)
16. Oreški, P., Šimović, V. (2012). *Razlozi za i protiv korištenja slobodnog softvera otvorenog izvornog kôda u osnovnom obrazovanju u Republici Hrvatskoj*. *Hrvatski časopis za odgoj i obrazovanje*, Vol. 14 No. 1, 11-23 https://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=118373 (17.2.2020.)
17. Oreški, P., Šimović, V. (2013). *Slobodan softver u obrazovanju.* Zagreb: Učiteljski fakultet Sveučilišta u Zagrebu URL: http://ed2.ufzg.hr/press/index.php/UFZG/catalog/book/3 (17.2.2020.)
18. Rouse, M. (2011). *BASIC (Beginner's All-Purpose Symbolic Instruction Code)*. https://whatis.techtarget.com/definition/BASIC-Beginners-All-purpose-Symbolic-Instruction-Code (17.2.2020.)
19. Škola za život https://skolazazivot.hr/informatika-virtualna-ucionica-2018-19/ (4.3.2020.)
20. Encyclopaedia Britannica https://www.britannica.com/technology/BASIC (17.2.2020.)
21. Tomljenović, K. (2018). *Računalno razmišljanje i uloga učenja pomoću igre na njegov razvoj*. https://inf.uniri.hr/images/studiji/poslijediplomski/kvalifikacijski/Kvalifikacijski_rad_Kreso_Tomljenovic.pdf (19.5.2020.)
22. Tutorialspoint (2015). *Pascal Programming – procedural programming.* https://www.tutorialspoint.com/pascal/pascal_tutorial.pdf (20.2.2020.)

23. W3Schools https://www.w3schools.com/python/python_intro.asp (19.5.2020.)

SVEUČILIŠTE U ZAGREBU

UČITELJSKI FAKULTET

ODSJEK ZA UČITELJSKE STUDIJE

ZAGREB

IZJAVA

kojom izjavljujem da sam suglasna da se trajno pohrani i javno objavi moj rad ***Programiranje u osnovnoj školi pomoću besplatnih alata na primjeru Pythona***

u javno dostupnom institucijskom repozitoriju

Učiteljskog fakulteta Sveučilišta u Zagrebu

i javno dostupnom repozitoriju Nacionalne i sveučilišne knjižnice u Zagrebu (u skladu s odredbama Zakona o znanstvenoj djelatnosti i visokom obrazovanju, NN br. 123/03, 198/03, 105/04, 174/04,02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).

U Zagrebu,

Ime i prezime: ***Dora Turčin***

Potpis

_____

SVEUČILIŠTE U ZAGREBU

UČITELJSKI FAKULTET

ODSJEK ZA UČITELJSKE STUDIJE

ZAGREB

IZJAVA O SAMOSTALNOJ IZRADI RADA

Potpisom potvrđujem kako sam ja, Dora Turčin, studentica Učiteljskog fakulteta Sveučilišta u Zagrebu samostalno napisala rad na temu *Programiranje u osnovnoj školi pomoću besplatnih alata na primjeru Pythona* pod vodstvom mentora izv. prof. dr. sc. Predraga Oreškog i kako se nisam koristila drugim izvorima osim onih navedenih u radu.

U Zagrebu,

Ime i prezime: ***Dora Turčin***

Potpis

_____