RAČUNALNO RAZMIŠLJANJE I PROGRAMIRANJE U SKLOPU NOVOG KURIKULUMA PREDMETA INFORMATIKA U OSNOVNOJ ŠKOLI

Akademik Leo Budin

Državni stručni skup za voditelje ŽSV učitelja i nastavnika Informatike/Računalstva

Zagreb, 4. travnja 2018.

Domena
Računalno razmišljane i programiranje
u kurikulumu Informatike

Dana 16. studenoga 2017. godine Vlada Republike Hrvatske donijela je odluku o uvođenju obaveznog predmeta Informatike u 5. i 6. razrede od školske godine 2018./2109.

Dana 12. veljače 2018. godine ministrica znanosti i obrazovanja profesorica Blaženka Divjak donijela je Odluku o donošenju kurikuluma za nastavni predmet Informatike za osnovne škole i gimnazije u Republici Hrvatskoj.

Odluka je objavljena u Narodnim novinama br. 22 od 6. ožujka 2018. godine

Na temelju članka 26. stavka 3. Zakona o odgoju i obrazovanju u osnovnoj i srednjoj školi (»Narodne novine«, broj 87/08, 86/09, 92/10, 105/10, 90/11, 16/12, 86/12, 94/13, 152/14 i 7/17) ministrica znanosti i obrazovanja donosi

ODLUKU

O DONOŠENJU KURIKULUMA ZA NASTAVNI PREDMET INFORMATIKE ZA OSNOVNE ŠKOLE I GIMNAZIJE U REPUBLICI HRVATSKOJ

I.

Ovom odlukom donosi se kurikulum za nastavni predmet Informatika za osnovne škole i gimnazije u Republici Hrvatskoj.

Predviđeno je da se obavezna nastava u 5. i 6. razredu, te (za sada) izborna nastava u 7. i 8. razredu, obavlja u skladu s odrednicama prijedloga novog kurikuluma predmeta Informatika.

KURIKULUM NASTAVNOGA PREDMETA INFORMATIKA ZA OSNOVNE I SREDNJE ŠKOLE

A. OPIS NASTAVNOGA PREDMETA INFORMATIKA

U posljednjih nekoliko desetljeća razvoj računalne znanosti omogućio je stvaranje informacijske i komunikacijske tehnologije koja je snažno i temeljito promijenila svijet oko nas. Primjena računala u svim područjima današnjega života mijenja i način shvaćanja svijeta u kojemu živimo. Digitalna pismenost danas je neophodna svakomu pojedincu kako bi mogao upotrebljavati računala i različite računalne sustave pri obavljanju svakodnevnih obveza.

Uz tradicionalne znanstvene discipline kao što su matematika, fizika ili kemija, informatika se nameće kao dodatno područje koje je nužno izučavati. Poznavanje temeljnih informatičkih koncepata kao što su programiranje, algoritmi ili strukture podataka postaje neophodno kako ne bismo bili samo korisnici informacijske i komunikacijske tehnologije (IKT) nego i stvaratelji.

Većina poslova 21. stoljeća zahtijeva razumijevanje i primjenu računalne znanosti s ciljem što veće produktivnosti i konkurentnosti. Informatičke kompetencije nužne su u rješavanju različitih izazova u svim područjima ljudskoga djelovanja i u svim područjima znanosti.

Pod nazivom Informatika u obrazovnom sustavu podrazumijeva se

– stjecanje vještina za uporabu informacijske i komunikacijske tehnologije (digitalna pismenost) kojom se oblikuju, spremaju, pretražuju i prenose različiti multimedijski sadržaji;

– uporabu informacijske i komunikacijske tehnologije u obrazovnom procesu (edukacijska tehnologija, e-učenje);

– rješavanje problema računalom uporabom nekog programskog jezika pri čemu su prepoznatljivi sljedeći koraci: specifikacija i raščlamba problema, analiza problema i odabir postupaka za njegovo rješavanje, priprema i izrada programa, ispitivanje programa i uporaba programa (rješavanje problema i programiranje).

Težište obrazovnog procesa u predmetu Informatika treba biti na rješavanju problema i programiranju kako bi se poticalo razvijanje računalnog načina razmišljanja koje omogućuje razumijevanje, analizu i rješavanje problema odabirom odgovarajućih strategija, algoritama i programskih rješenja. Takvi se načini razmišljanja trebaju prenositi i u druga područja posebice matematičko i prirodoslovno, kao i u svakodnevni život.

Učenje Informatike priprema učenika za mnoga područja djelovanja, osobna i poslovna. Osobiti doprinos učenja predmeta Informatika očituje se u razvoju računalnoga načina razmišljanja koje uključuje i tehnike rješavanja problema:

– prikazivanje informacija apstrakcijama

– logičko povezivanje i analizu podataka

– automatizaciju rješenja uporabom algoritamskoga razmišljanja

– prepoznavanje, analizu i primjenu mogućih rješenja s ciljem postizanja učinkovitoga rezultata vodeći računa o dostupnim resursima

– formuliranje problema načinom primjerenim uporabi računala i računalnih alata

– generalizaciju procesa rješavanja problema primjenjivoga na čitav niz sličnih problema.

Te su tehnike alat za rješavanje različitih problema i u ostalim disciplinama pa su veoma važne svima, a ne samo informatičkim stručnjacima.

Sadržaji iz predmeta Informatika trebaju se usvajati tijekom cijeloga školovanja, pri čemu bi se trebalo koristiti načelom spiralnoga modela prema kojemu se znanje stečeno na nižim stupnjevima obrazovanja proširuje i produbljuje na višima. Znanja, vještine i stavovi usvojeni u Informatici podrška su svim ostalim predmetima i međupredmetnim temama.

Od četiri domene kurikuluma predmeta Informatika domena Računalno razmišljanje i programiranje zaslužuje posebnu pažnju !

Mala rasprava o nazivlju

Mnogi nesporazumi u raspravama nastaju zbog neusklađenosti naziva.

- Riječ informatika prvi puta se spominje 1957. godine u Njemačkoj u naslovu eseja ***Informatik: Automatische Informationsverarbaitung.***
- Godine 1966. *l'Academie française* službeno odobrava riječ ***l'informatique*** koja označava ***„science du traitement de l'information".***
- U Njemačkoj se 1968. službeno uvodi naziv ***Informatik*** kao novo znanstveno područje i otvaraju se prvi sveučilišni studiji koji se njime bave.
- U Italiji i Španjolskoj koristi se naziv ***Informatica***,
- Engleski naziv ***Informatics*** rabi se u Velikoj Britaniji i diljem Europe kada se govori i piše engleskim jezikom.
- U S.A.D. se umjesto naziva ***Informatics*** rabi naziv ***Computer Science***. Taj se naziv susreće i u europskim okruženju kao sinonim naziva ***Informatics***.

U hrvatskom se jeziku se pod nazivom **informatika** uobičajeno podrazumjevaju (kako je to uvodno navedeno u kurikularnom dokumentu):

- *vještine uporabe informacijske i komunikacijske tehnologije (**digitalna pismenost**);*
- *uporaba informacijske i komunikacijske tehnologije u obrazovnom procesu (**edukacijska tehnologija**, **e-učenje**);*
- *rješavanje problema računalom uporabom nekog programskog jezika (**računalno razmišljanje i programiranje**).*

U engleskom jeziku se za vještine uporabe informacijske i komunikacijske tehnologije rabi naslov ***digital literacy*** (digitalna pismenost)

Digitalna pismenost stječe se upoznavanjem ***informacijske tehnologije*** (***Infomation Technology, IT***) odnosno ***informacijske i komunikacijske tehnologije*** (***Information and Communication Technology, ICT***).

Sve je više u uporabi i naziv ***computing*** koji obuhvaća kako teorijsku podlogu tako i načela izgradnje i primjenu suvremenih računalnih i komunikacijskih sustava.

U Hrvatskoj je naziv ***computing*** preveden kao ***računarstvo.***

U znanstvenom području ***tehničkih znanosti*** postoje polje ***računarstvo***.

U znanstvenom području ***društvenih znanosti*** postoji polje ***informacijske i komunikacijske znanosti***
(s granama: ***informacijski sustavi i informatologija***, ***organizacija i informatika***, ***informacijsko i programsko inženjerstvo***).

U znanstvenom području ***prirodnih znanosti*** u polju ***matematika*** postoji grana ***matematička logika i računarstvo.***

U svojevrsnom terminološkom kaosu u raznim zemljama rabe se različiti nazivi za istovrsne ili vrlo slične sadržaje. To su:

Computing,
Computational thinking,
Informatics,
Computer science,
Algorithmitic Thinking,
Coding,

Međutim, pod tim različitim nazivima pojavljuju se vrlo slični sadržaji:

- ***specifikacija i raščlamba problema (dekompozicija i apstrakcija),***
- ***odabir postupaka za njegovo rješavanje (algoritamsko razmišljanje),***
- ***priprema i izrada programa (programiranje),***
- ***ispitivanje i ispravljanje programa,***
- ***poopćavanje rješenja.***

To su sadržaji koje obuhvaća domena
Računalno razmišljanje i programiranje
u našem kurikulumu informatike

Najnovija zbivanja u Europi i svijetu

Kurikulum nastavnog predmeta Informatika za osnovne i srednje škole sadrži *Popis izvora i literature* sa šezdeset navoda.

Iz tog je popisa vidljivo da je stručna radna skupina pažljivo pregledala i uzela u obzir svjetske i posebice europske preporuke za unapređenja informatičkih kurikuluma.

Međutim, zadnjih se nekoliko godina u svijetu intenzivno razrađuju okviri za kurikulume u kojima se uočava sve veći naglasak na domenu računalnog razmišljanje i programiranja.

Ti novi dokumenti nisu mogli nisu mogli biti uzeti u obzir u pripremi našeg kurikuluma pa ih i nema u popisu izvora i literature. No oni samo dodatno potkrepljuju postavke kurikuluma. U nastavku ove prezentacije komentirat će se neki od njih.

Informatics Europe je udruženje od preko 120 sveučilišnih i istraživačkih institucija iz 30 europskih zemalja. Članice udruženja iz Hrvatske su Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu i Odjel za informatiku Sveučilišta u Rijeci.

Governments and the public all too often satisfy themselves that teaching digital literacy is enough to prepare the citizenry for the "Information Society" that Europe has decided to become. **It is not**. Digital literacy is a practical skill, not a scientific topic or an adequate intellectual preparation for the challenges of a digital world.

For a nation or a group of nations to compete in the race for technology innovation, the general population must in addition to digital literacy understand the basics of the underlying discipline, *informatics*[1]. On the road to an information society, informatics plays the same enabling role as mathematics and physics in previous industrial revolutions.

[1] *informatics* ili *computer science* poistovjećuje se s domenom *Računalno razmišljanje i programiranje* u našem kurikulumu predmeta *Informatika*.

Digital technologies are not "just another technology" like the steam engine, the telegraph, the aeroplane, and penicillin. All other technologies invented by mankind, are technologies that stretch physical abilities. Digital technology and its scientific basis, Informatics, radically challenge the way we think about, understand, and organise the World. The impact on society is pervasive and profound, e.g. politically, economically, legally, medically, scientifically, and educationally.

Therefore, it is of profound importance that Informatics becomes part of general education so that all children are educated to become critical, competent and reflective citizens who can contribute in the broadest sense to shaping the future of our society.

Issued: November 2017 DES4633
ISBN: 978-1-78252-297-3

studeni 2017. godine

RECOMMENDATION 5

Governments should introduce quality-assured computing conversion courses for existing teachers, equivalent to those in physics and mathematics. Individual teachers or schools should not have to contribute to the costs of this training.

RECOMMENDATION 6

Governments should work with higher education providers and the British Computer Society to develop and accredit pre-service subject content courses to enable more people from a wider variety of backgrounds to become computing teachers.

Existing initiatives to support and develop computing degree courses with qualified teaching status should be continued and, if successful, expanded.

RECOMMENDATION 7

Higher education providers need to promote careers in computing education to a wide range of students.

RECOMMENDATION 8

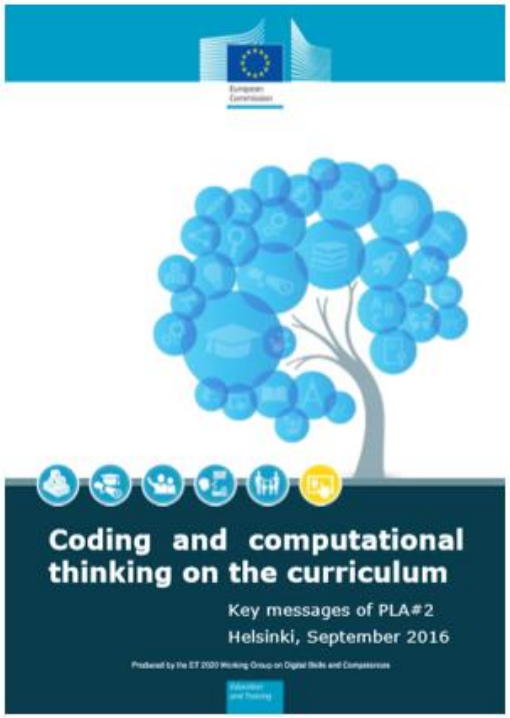
Industry and academia should support and encourage braided[9] careers for staff who want to teach as well as work in another setting.

COMPUTER PROGRAMMING

Computer programming is the process of developing and implementing various sets of instructions to enable a computer to perform a certain task, solve problems, and provide human interactivity. These instructions (source codes which are written in a programming language) are considered computer programs and help the computer to operate smoothly.

In this report the terms computer programming and coding are used interchangeably, and are in general referred to as coding. They refer to activities that enable children not only to know how to use specific programmes but to learn how to programme computers, tablets, or other electronic devices.

Computational thinking is typically associated with coding and computer programming, but is more than that, involving "solving problems, designing systems, and understanding human behaviour", according to the Carnegie Mellon University.

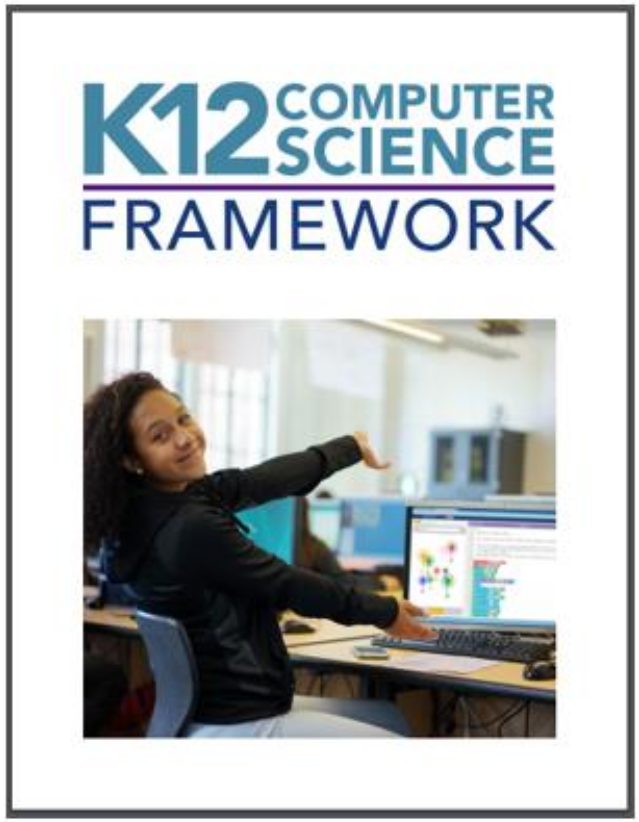


Computational Thinking is understood as shorthand for "thinking like a computer scientist", i.e. using concepts of computer science to formulate and solve problems. In the past decade Computational Thinking has increasingly gained attention in the educational field for its potential to teach logical thinking, problem-solving and digital competence.

Teachers need a good understanding of what CT is and how to teach it. Introducing CT requires new training, possibly at large scale, as CT does so so far not often feature in teachers' initial training. Support services for teachers that provide concrete advice and examples can support teachers to use coding and computational thinking in class.

In delivering CT education, teachers' knowledge and starting point should be considered; e.g. the similar focus on logical steps between mathematics and CT can be exploited.

Teachers need examples and good practices relevant for their specific teaching context in order to confidently approach CT.[2] Teaching CT may require new pedagogical approaches that put students at the centre of the learning process.



travanj 2016. godine

K–12 Computer Science Framework Steering Committee

Association for Computing Machinery

CODE

CSTEACHERS.ORG
COMPUTER SCIENCE TEACHERS ASSOCIATION

cyber innovation center
Collaboration • Research • Technology

NATIONAL MATH + SCIENCE INITIATIVE

Computational thinking is essentially a problem-solving process that involves designing solutions that capitalize on the power of computers; this process begins before a single line of code is written. Computers provide benefits in terms of memory, speed, and accuracy of execution. Computers also require people to express their thinking in a formal structure, such as a programming language. Similar to writing notes on a piece of paper to "get your thoughts down," creating a program allows people to externalize their thoughts in a form that can be manipulated and scrutinized. Programming allows students to think about their thinking; by debugging a program, students debug their own thinking (Papert, 1980).

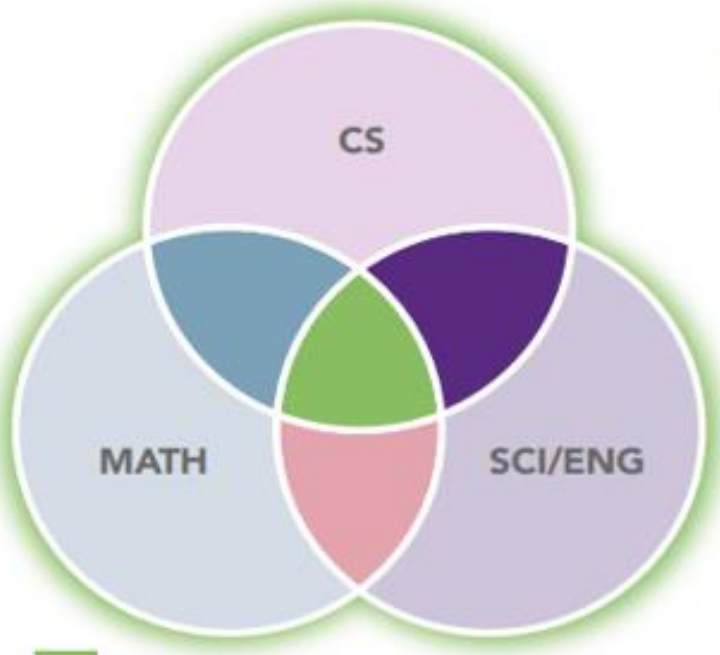
Computational thinking refers to the thought processes involved in expressing solutions as computational steps or algorithms that can be carried out by a computer.

Computer Science Practices and Other Subject Areas

The framework is grounded in the belief that computer science offers unique opportunities for developing computational thinking and that the framework's practices can be applied to other subjects beyond computer science. As Barr and Stephenson (2011) have noted, the "computer science education community can play an important role in highlighting algorithmic problem solving practices and applications of computing across disciplines, and help integrate the application of computational methods and tools across diverse areas of learning" (p. 49).

While computational thinking is a focus in computer science, it is also included in standards for other subjects. For example, computational thinking is explicitly referenced in the practices of many state science standards[1] and implicitly in state math standards.[2] Additionally, the recent revision to the International Society for Technology in Education Standards for Students (ISTE, 2016) describes computational thinking in a similar way as the framework. All of these documents share the vision that computational thinking is important for all students.

> ***Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability (Wing, 2006, p. 33).***



CS + Math

- **Develop and use abstractions**
  M2. Reason abstractly and quantitatively
  M7. Look for and make use of structure
  M8. Look for and express regularity in repeated reasoning
  CS4. Developing and Using Abstractions
- **Use tools when collaborating**
  M5. Use appropriate tools strategically
  CS2. Collaborating Around Computing
- **Communicate precisely**
  M6. Attend to precision
  CS7. Communicating About Computing

CS + Sci/Eng

- **Communicate with data**
  S4. Analyze and interpret data
  CS7. Communicating About Computing
- **Create artifacts**
  S3. Plan and carry out investigations
  S6. Construct explanations and design solutions
  CS4. Developing and Using Abstractions
  CS5. Creating Computational Artifacts
  CS6. Testing and Refining Computational Artifacts

CS + Math + Sci/Eng

- **Model**
  S2. Develop and use models
  M4. Model with mathematics
  CS4. Developing and Using Abstractions
  CS6. Testing and Refining Computational Artifacts
- **Use computational thinking**
  S5. Use mathematics and computational thinking
  CS3. Recognizing and Defining Computational Problems
  CS4. Developing and Using Abstractions
  CS5. Creating Computational Artifacts
- **Define problems**
  S1. Ask questions and define problems
  M1. Make sense of problems and persevere in solving them
  CS3. Recognizing and Defining Computational Problems
- **Communicate rationale**
  S7. Engage in argument from evidence
  S8. Obtain, evaluate, and communicate information
  M3. Construct viable arguments and critique the reasoning of others
  CS7. Communicating About Computing

Teacher Development

Teacher development is a critical part of the computer science education infrastructure. Teacher development is used here as a broad term that includes preservice teacher preparation, certification, licensure, and ongoing professional development. It concerns stakeholders in higher education, state agencies, school districts, and organizations that provide professional development.

Professional development should attend to novice teachers' anxiety over their lack of content knowledge.

Given the introduction of computer science into many education systems, it is natural that many teachers attending professional development may not already have a background in computer science. While not diminishing the importance of pedagogical content knowledge or general pedagogical practice for teaching computer science, professional development providers should attend to teachers' anxiety about content knowledge by helping them see that many teachers are in the same situation. Professional development can instill a growth mindset in participants, in which learning builds over time, during a workshop as well as the school year while teachers deliver instruction. Professional development should be viewed as a safe space to try new or difficult things.

EUROPEAN
COMMISSION

Brussels, 17.1.2018
COM(2018) 22 final

COMMUNICATION FROM THE COMMISSION TO THE EUROPEAN PARLIAMENT, THE COUNCIL, THE EUROPEAN ECONOMIC AND SOCIAL COMMITTEE AND THE COMMITTEE OF THE REGIONS

on the Digital Education Action Plan

The digital revolution will continue to dramatically change the way Europeans live, work and study. While this offers tremendous opportunities, there are also significant risks if digital competences are not developed. As part of the Skills Agenda, the Upskilling Pathways initiative recommends that Member States introduce coherent provision to improve the digital (and literacy and numeracy) skills of the many millions of low-skilled or low-qualified adults - the group in most urgent need. Moreover, an estimated 90 % of jobs nowadays require some level of digital skills[21] and one significant threat is that Europe will lose its most competitive edge — a highly-skilled and educated workforce — should we fail to teach digital competences to Europeans of all ages.

Acquiring digital skills needs to start at early age and carry on throughout life. This can happen as part of educational curricula or through after-school classes. Young Europeans are avid users of the web, apps and games but they also need to learn about underlying structures and basic algorithms, and become digital creators and leaders. An example of a successful grassroots movement is the EU codeweek.eu initiative, which reached nearly a million people around the world in 2016. Based on this experience, the initiative will be scaled up to encourage all schools in Europe to participate in **EU Code Week** by collaborating with authorities in EU Member States, Code Week ambassadors, the eTwinning network, the Digital Skills and Jobs Coalition[22] and related actions.


Legal notice
Cookies
Contact on Europa
Search on Europa
English (en)
European Commission
EDUCATION AND TRAINING
Supporting education and training in Europe and beyond
Search
Search
Education and training > European Commission > Education and training >
Home
Policies
Initiatives
Updates
Resources
Contact
ET2020 working group on Digital Skills and Competences

Definition of digital competence

2006 Definition

Digital competence involves the confident and critical use of Information Society Technology (IST) for work, leisure and communication. It is underpinned by basic skills in ICT: the use of computers to retrieve, assess, store, produce, present and exchange information, and to communicate and participate in collaborative networks via the Internet.

2018 Definition

Digital competence involves the confident, critical and responsible use of, and engagement with, digital technologies for learning, at work, and for participation in society. It includes information and data literacy, communication and collaboration, digital content creation (including programming), safety (including digital well-being and competences related to cybersecurity), and problem solving.

Heads of State and Government discussed education and training at the Gothenburg Social Summit on 17 November 2017, guided by the Commission's Communication 'Strengthening European Identity through Education and Culture'. This resulted in the European Council conclusions of 14 December 2017 calling on Member States, the Council and the Commission to take the agenda discussed in Gothenburg forward. The Recommendation follows up on that political agreement.

IMPROVING 8 KEY COMPETENCES

1- Literacy

Strengthening literacy as a basis for further learning and communication in different societal and cultural contexts

2 - Languages

Enhancing the ability to use a variety of languages to be active and better cope with the challenges of today's multilingual and diverse societies

3 - Science, technology, engineering and mathematics (STEM)

Focusing on improving acquisition of these competences to nurture scientific understanding

4 - Digital

Strengthening the confident and critical use of digital technology , including coding and programming, safety and citizenship related aspects

5 - Personal, social and learning

Improving the skills necessary to participate in an active social life

6 - Civic

Stressing the importance of democratic participation, European values, sustainable development and media literacy

7 - Entrepreneurship

Enhancing entrepreneurial attitudes to unlock personal potential, creativity and self-initiative

8 - Cultural awareness and expression

Increasing intercultural skills and the ability to express ideas in a variety of ways and contexts

The proposed Recommendation especially calls for:

- Raising levels of achievement in **basic skills** (literacy, numeracy and basic digital skills)
- **Promoting entrepreneurial education**, notably by providing one entrepreneurial experience in primary or secondary school
- Boosting **digital competences** including programming and cybersecurity aspects
- Supporting the development of and the interest in **science, technology, engineering and maths** (STEM) competence and making STEM careers more attractive
- Increasing **language competences** and number of languages learned.

The Commission will:

- Provide support to educational staff, for example by helping them develop **innovative teaching methods** using new technologies
- Enhance **work-based and project-based learning**, and promoting **learning mobility**
- **Assess key competences** to allow people to measure their competence development, but also to make competences visible through tools such as **Europass** that makes skills and qualifications easily understood, or **Youthpass that helps demonstrate learning outcomes from youth work activities**
- **Reinforce systematic collaboration** between education, training and employment learning settings to support lifelong learning pathways

To this end the Commission will:

- Make it easier for Member States to learn from each other about promising learning approaches, support for educational staff and the assessment and validation of competence development
- Develop reference material and tools in cooperation with Member States
- Monitor the provision of competence-oriented education, training and learning and competence development in the EU
- Analyse and report actions taken through the Education and Training 2020 framework, including the **Education and Training Monitor**, a snapshot of the state of education and training systems in Member States published by the Commission every year

U našem se kurikulumu predmeta Informatika prepoznaju opisani svjetski i europski trendovi, što je vidljivo iz sljedećeg (već spomenutog) citata:

> Težište obrazovnog procesa u predmetu Informatika treba biti na rješavanju problema i programiranju kako bi se poticalo razvijanje računalnog načina razmišljanja koje omogućuje razumijevanje, analizu i rješavanje problema odabirom odgovarajućih strategija, algoritama i programskih rješenja. Takvi se načini razmišljanja trebaju prenositi i u druga područja posebice matematičko i prirodoslovno, kao i u svakodnevni život.

Odabir programskog jezika

Tradicionalni pristup poučavanju programiranja polazio je uobičajeno od opisa svojstava računalnog sustava, upoznavanja internog prikaza načina pohranjivanja osnovnih tipova podataka i detaljnog upoznavanja sintakse programskog jezika. Tek nakon toga pripremali su se jednostavni programi koji su služili su prvenstveno za ilustraciju svojstava pojedinih elemenata programskog jezika.

Moderni programski jezici i njihovo radno okruženje omogućuju da poučavanje programiranja može početi s izučavanjem pojedinih naredbi i postupnom izgradnjom vlastiti ***misaonog modela*** računalnog sustava koji objašnjavanja djelovanje te naredbe.

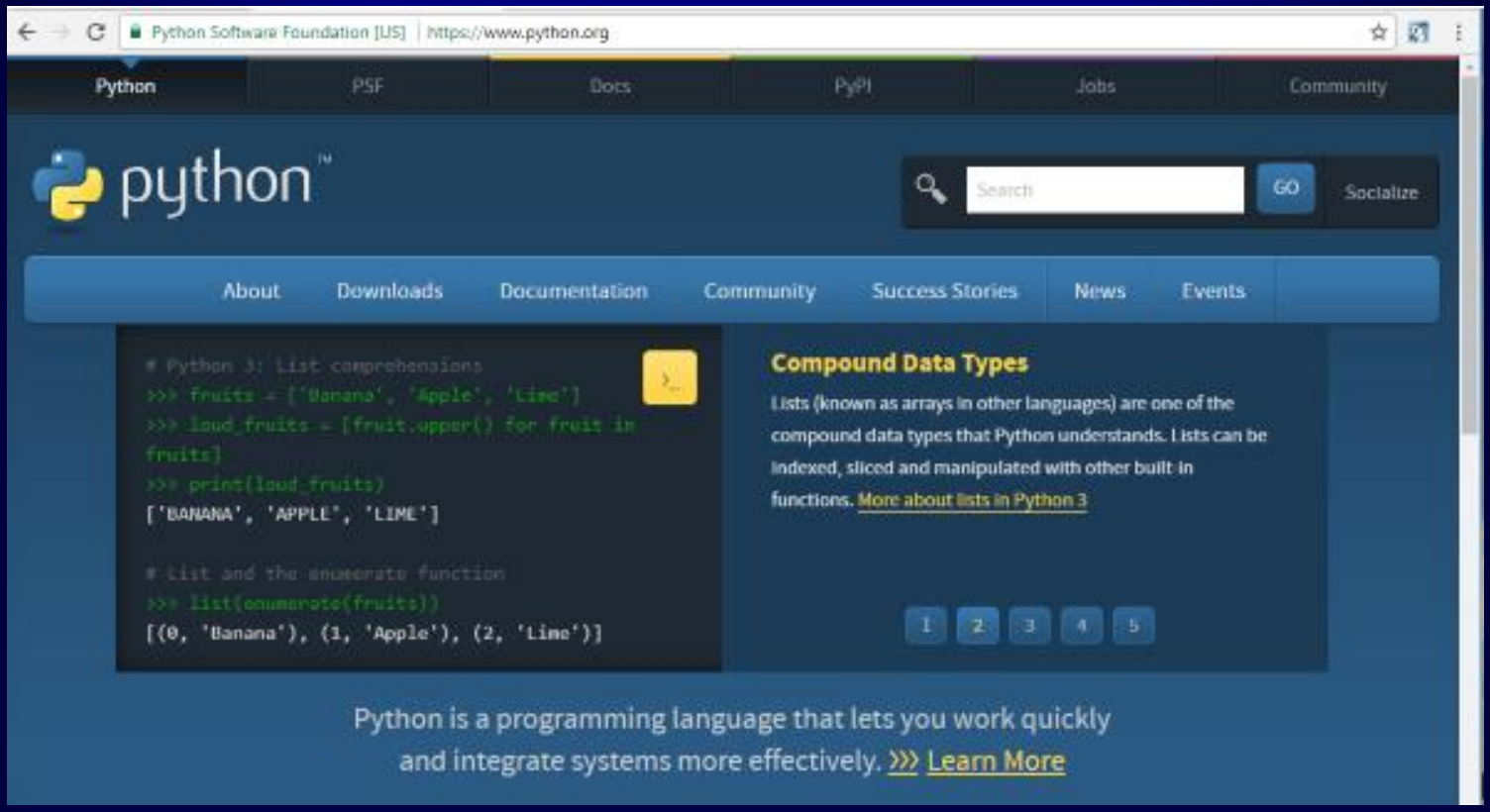
Takav misaoni model računalnog sustava (engl. ***notional machine***) postepeno se izgrađuje u glavi svakog učenika i prilagođava razini složenosti problema.

S obzirom na to da misaoni model ovisi o svojstvima programskog jezika, prikladno je za početno učenje programiranja odabrati jedan programski jezik i sustavno ga rabiti u cijelom razdoblju učenja.

Bilo bi vrlo korisno da odabir jezika bude takav da se on može rabiti i kasnije u višim razinama obrazovanja.

Za poučavanje programiranja u višim razredima osnovne škole, zadnjih se nekoliko godina prikladnim pokazao programski jezik *Python*. On je postao i najrašireniji jezik za početno učenje programiranja.

Programsko okruženje za pripremu i izvođenje programa dobavlja se slobodno i besplatno s mrežne stranice **http://www.python.org**



U gimnazijama i u većini ostalih srednjih škola razumno je zadržati programski jezik *Python* jer će iskustvo stečeno u osnovnoj školi olakšati računalno razmišljanje pri rješavanju problema više razine složenosti.

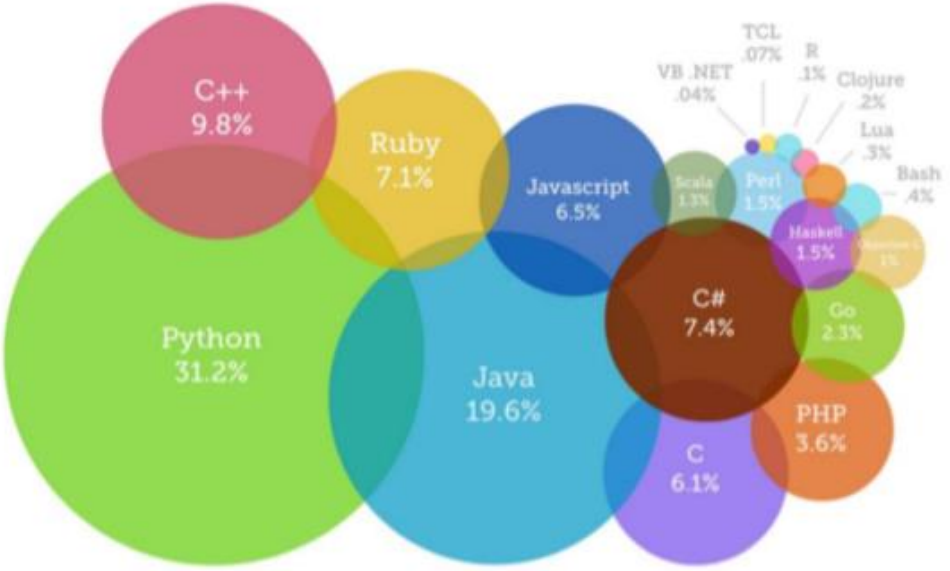
Python se pritom može nadograđivati slobodno dostupnim posebnim specijaliziranim modulima za pojedine vrste problema kao što su:

- numpy – modul za numeričku matematiku
- scipy – modul s naprednim matematičkim postupcima
- pandas – modul za statističku obradu podataka
- matplotlib – modul za pripremu dvodimenzionalnih grafičkih prikaza

Taj cijeli ekosustav Pythona može se dobaviti s mrežnog mjesta https://www.anaconda.com/download/.

U srednjim školama koje pripremaju učenike za informatičke struke treba prema potrebi uvoditi programske jezike koji se pretežito koriste za pripremu profesionalnih programskih rješenja (*C*, *C++*, *C#*, Java, Javascript, HTML i druge).

Pregled uporabe programskih jezika u nastavi programiranja (2017. godina)



Python se sve više koristi i za profesionalne primjene. Tako je *Python* jedan od tri službena programska jezika u tvrtki *Google*. *Youtube* je u velikom svom dijelu razvijen u *Pythonu*, a u svemirskoj agenciji NASA mnogi su primjenski programi također pripremljeni u *Pythonu*.

Stručno usavršavanje učitelja

Učenje programiranja potiče konstrukcijski pristup učenju

Pokazalo se da učenjem programiranja uporabom prikladnog programskog jezika i radnog okruženja za pripremu programskih rješenja učenici razvijaju načine razmišljanja kompatibilne s općim osnovnim postavkama konstrukcijskog modela učenja primjenljivog u ostalim problemskim domenama.

Pokazalo se da aktivno učenje programiranja potiče:

- temeljito razmišljanje, precizno izražavanje i formalni opisa problema (jer se programi zasnivaju na dobro razrađenim algoritmima) ;
- razumijevanje osnovnih koncepata kao što su: formalne procedure, funkcije i varijable (jer se na tim konceptima izgrađuju računalni programi) ;
- heurističke pristupe rješavanju problema kao što su: izrada plana rješavanja, prepoznavanje sličnosti s nekim već riješenim problemima, dekompozicija složenih problema na manje dijelove (jer se u postupcima programiranja postupa na taj način) ;

- postupno pronalaženje rješenja problema metodom pokušaja i pogrešaka (jer se priprema računalnog programa obavlja upravo na taj način – do uspješnog računalnog programa dolazi se najčešće tako da se postupno otklanjaju pogreške u prvotno skiciranoj varijanti programa) ;
- pronalaženje rješenja problema na način da se međusobno povezuju prethodno razrađeni i provjereni manji dijelovi rješenja (jer se računalni programi grade tako da se međusobno povezuju prethodno pripremljene i ispitane funkcije) ;
- prepoznavanje mogućnosti da se neki problem može riješiti na više načina te da se odabir najboljeg rješenja može obaviti temeljem odgovarajućih kriterija usporedbe (jer se pri programiranje između više mogućih rješenja odabire ono koje je po nekim svojstvima – najčešće po trajanju izvođenja ili potrošnji memorije – najprikladnije).

Zbog svega navedenog ustanovljeno je da se učenjem programiranja potiče razvitak računalno-algoritamskog načina razmišljanja (engl. *computational thinking*) koje omogućuje razumijevanje, analizu i rješavanje problema odabirom odgovarajućih strategija i programskih rješenja.

Takav način razmišljanja nadovezuje se na matematički način razmišljanja (engl. *mathematical thinking*) koji se sustavno mora razvijati u matematici. Takvi se načini razmišljanja moraju prenositi i u druga područja, posebice u područje prirodoslovlja, kao i u praktični život.

Inovativni pristup oblikovanju nastave informatike, posebice u osnovnoj školi, svakako bi se trebao odraziti i na poboljšavanje PISA rezultata hrvatskih učenika u području matematike i u području prirodoslovlja.

Zbog toga je potrebno stručno usavršavanje provesti ne samo za učitelje informatike već i za učitelje tehničke kulture, matematike i fizike koji moraju steći potrebna znanja i vještine u rješavanju problema iz svojih domena računalnim razmišljanjem programiranjem.

Preporuka
Povjerenstva za uvođenje informatike kao obaveznog predmeta u osnovnoškolski odgoj i obrazovanje

Plan stručnog usavršavanja postojećih učitelja

Plan stručnog usavršavanja treba načiniti za dvije skupine postojećih učitelja:

- učitelje informatike,
- učitelje ostalih premeta STEM područja.

Za učitelje informatike stručno usavršavanje treba obuhvatiti:

1) upoznavanje s novim kurikulumom iz nastavnog predmeta Informatika i preporuke oko organizacije i pripreme za provedbu u svim domenama ,
2) cjelovitu edukaciju u domeni *Računalno razmišljanje i programiranje,*
3) pripremu učitelja za mentoriranje učenika na informatičkim natjecanjima.

Za sve učitelje ostalih predmeta iz STEM područja stručno usavršavanje treba obuhvatiti:

4) cjelovitu edukaciju u domeni *Računalno razmišljanje i programiranje* u okviru koje učitelje treba pripremiti za primjenu računalnog razmišljanja i programiranja u konstrukcijskom pristupu poučavanju u njihovim predmetima.

U praktičnoj provedbi stručnog usavršavanja mogu se objediniti točke 2) i 4). Štoviše, takva organizacija stručnog usavršavanja može imati vrlo pozitivan sinergijski učinak.

Povjerenstvo preporuča da Ministarstvo znanosti i obrazovanja i Agencija za odgoj i obrazovanje (u suradnji sa sveučilišnim institucijama) pripremi detaljni plan provedbe posebnog projekta stručnog usavršavanja.

Preporuke za provedbu stručnog usavršavanja

Radna skupina *Povjerenstva za uvođenje Informatike kao obaveznog predmeta u osnovnoškolski odgoj i obrazovanje* u sastavu:

- prof. dr. sc. Nina Begičević Redep, Fakultet organizacije i informatike Sveučilišta u Zagrebu, Varaždin
- akademik Leo Budin, Hrvatska akademija znanosti i umjetnosti
- prof. dr. sc. Tomislav Jakopec, Filozofski fakultet Sveučilišta u Osijeku
- prof. dr. sc. Vedran Mornar, Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu
- dr. sc. Jelena Nakić, Prirodoslovno matematički fakultet Sveučilišta u Splitu
- doc. dr. sc. Goranka Nogo, Prirodoslovno matematički fakultet Sveučilišta u Zagrebu

utvrdila je sljedeće aktivnosti provedbe stručnog usavršavanja:

- odabir i priprema obrazovnih sadržaja ;
- priprema edukatora ;
- stručno usavršavanje učitelja.

Odabir i priprema obrazovnih sadržaja

Priručnik *Računalno razmišljanje i programiranje u Pythonu* (Element, Zagreb, 2017) sadrži prikladno edukacijsko gradivo koje može poslužiti kao edukacijsko gradivo za domenu Računalno razmišljanje i programiranje. Izdavač je suglasan da se dijelovi priručnika upotrijebe za pripremu digitalnih obrazovnih sadržaja.

prof. dr. sc. Marko Rosić

Sveučilište u Splitu

marko.rosic@unist.hr

Recenzija knjige *Računalno razmišljanje i programiranje u Pythonu*

Autori: Leo Budin, Predrag Brođanac, Zlatka Markučič, Smiljana Perić, Dejan Škvorc i Magdalena Babić

Ovaj priručnik je izvorno djelo na hrvatskom jeziku te ima svoje mjesto kako u sadašnjoj strukturi programa osnovne škole (pokriva gradivo od 5. do 8. razreda osnovne škole) tako i u predloženom trećem obrazovnom ciklusu nove organizacije obveznog školovanja čime je postignuta neupitna relevantnost teme ovog djela. S obzirom na relevantnost teme smatram da je objavljivanje ovog djela od posebnog društvenog značenja. U pitanju je priručnik za učitelje i nastavnike unutar područja rješavanja problema i programiranja što upravo treba biti okosnica predmeta *Informatika*. Odabir programskog jezika *Python* predstavlja rješenje koje je u brojnim kako domaćim tako i inozemnim radovima predstavljeno kao optimalno rješenje za ovo područje.

https://esavjetovanja.gov.hr/ECon/MainScreen?entityId=6720

Mario Miličević 01.02.2018 17:12 1 0

Pozdravljam napore oko donošenja novog kurikuluma predmeta Informatika, naročito u dijelu koji se odnosi na osnovne škole. Krajnje je vrijeme da se s načelnih strategija, primjerice o STEM području kao EU i nacionalnom prioritetu, prijeđe na konkretne korake.

Za prvo upoznavanje s programiranjem u nižim razredima osn.škole svakako je preporuka da se to odradi kroz igru i programski jezik Scratch.
Nakon toga treba slijediti programski jezik Python, možda prvo kroz kornjačinu grafiku kao logični nastavak na programiranje kroz "igru" u Scratch-u. Python je trenutno optimalan izbor jer s jedne strane ima sintaksu koja nije stresna za početnike, a s druge strane u nastavku školovanja omogućuje usvajanje i naprednijih koncepata kao što je primjerice objektno orijentirano programiranje. Da ne spominjem da se u Pythonu mogu raditi i ozbiljniji primjenski programi.

Konačno, nije za zanemariti što za Python već postoji i vrlo kvalitetna literatura na hrvatskom jeziku, primjerice kao što je Računalno razmišljanje i programiranje u Pythonu.

Predvidivo bi se u obliku prezentacija mogli pripremiti obrazovni sadržaji sa sljedećim temama:

- Upoznavanje s radnim okruženjem Pythona, programska varijable (poglavlje 2. priručnika);
- Osnovne funkcije modula kornjačine grafike (modul `turtle`), programske petlje (dio 3. poglavlja priručnika);
- Definiranje vlastitih funkcija, pisanje programa s više funkcija, generiranje nasumičnih brojeva (dio 4. poglavlja priručnika);
- Ispitivanje uvjeta i donošenje odluka u programima i uređivanje ispisa (dio 5. poglavlja priručnika);
- Ostvarenje programskih petlji ispitivanjem uvjeta (dio 5. poglavlja priručnika);
- Zbirke podataka: liste i n-torke (dio 6. poglavlja priručnika);
- Rastavljanje problema na manje dijelove, priprema funkcija i programskih modula (dio 7. poglavlja priručnika);
- Programski modul za operacije s prirodnim brojevima (dio 7. poglavlja priručnika);
- Napredna uporaba modula `turtle`, vizualizacija, optičke iluzije (dio 8. poglavlja priručnika)
- Modul za računanje s razlomcima (`fractions`) i modul za računanje s decimalnim brojevima (`decimal`) (dio 9. poglavlja priručnika);
- Strojni oblik pohranjivanja racionalnih brojeva, tip podataka `float` (dio 9. poglavlja priručnika).

Priprema edukatora

U skladu s procijenjenim brojem edukacijskih skupina i njihova regionalnog rasporede treba pronaći odgovarajući broj edukatora. Potencijalni edukatori mogli bi se odabrati iz sveučilišnih institucija odnosno među učiteljima osnovnih škola i nastavnicima srednjih škola.

Od sveučilišnih institucija mogli bi se u prvom redu kontaktirati one koje su na neki način sudjelovale u raspravama oko preobrazbe nastave informatike, kao i u svim ostalim institucijama koje djeluju u tom području.

Agencija za odgoj i obrazovanje bi uz pomoć voditelja županijskih stručnih vijeća mogla pomoći u izboru edukatora iz redova učitelja i nastavnika

Predvidivo bi se priprema edukatora mogla obaviti na jednom dvodnevnom skupu na kojem bi se obradili pripremljeni nastavni materijali i utvrdili načini daljnje suradnje.

Na tom bi se dvodnevnom skupu obradili prethodno pripremljeni obrazovni sadržaji.

Na temelju primjedaba edukatore i provedenih rasprava mogla bi se obaviti eventualna dorada obrazovnih sadržaja.

Stručno usavršavanje učitelja

Radna skupina predložila je da se stručno usavršavanje svih učitelja informatike, matematike, fizike i tehničke kulture raspodijeljenih edukacijske skupine provede prema pripremljenom rasporedu u mjesecu lipnju 2018. u terminima predviđenim za stručno usavršavanje učitelja.

Naknadna analiza u Ministarstvu znanosti i obrazovanja pokazala je da je stručno usavršavanje prikladnije obavit u dva jednodnevna seminara s tim da se jedan od njih održi u mjesecu lipnju i drugi krajem mjeseca kolovoza 2018. godine.

Osnovni nastavni materijali mogu biti (po potrebi dorađeni) digitalni obrazovni sadržaji priređeni za pripremu edukatora.

Probna provjera stručnog usavršavanja

Na prijedlog ministrice prof. dr. sc. Blaženka Divjak utvrđeno je da se tijekom mjeseca travnja održe dva pilotska jednodnevna seminara za skupine od po 150 sudionika kako bi se stekla stanovita iskustva za organizaciju velikog ljetnog edukacijskog poduhvata.

U tu će svrhu:

- MZO istražiti mogućnost da Fakultet elektrotehnike i računarstva iz Zagreba i Fakultet organizacije i informatike iz Varaždina ustupe odgovarajuće predavaonice (s dostupnom *Wi-Fi* mrežom);
- Agencija za odgoj i obrazovanje provesti pripremu, najaviti održavanje seminara i na uobičajeni način prikupiti podatke i pripremiti popis sudionika seminara (učitelja Informatike, Matematike, Fizike i Tehničke kulture);
- MZO u dogovoru s koautorima priručnika *Računalno razmišljanje i programiranje u Pythonu* odrediti edukatore za ta dva seminara (tijekom seminara provjerit će se obrazovni sadržaji pripremljeni kao *PP* prezentacije).

DODATAK

Neke posebnosti programskog jezika Python

- *Python* ima jasna pravila oblikovanja te se programi pisani u *Pythonu* lako čitaju i imaju veliku pedagošku vrijednost. Radi se o vrlo intuitivnom jeziku koji kod učenika razvija smisao za sistematičnost i preciznost.
- Programsko okružje potiče pisanje programa s jasno izraženom logičkom strukturom programa.
- Podjela problema na manje dijelove i njihovo međusobno povezivanje u cjelovito rješenje obavlja se vrlo jednostavno. Razrađena rješenja mogu se lako pohraniti i kasnije upotrebljavati pri rješavanju drugih problema.
- Programsko okružje *Pythona* omogućuje jednostavan i brz prijelaz iz faze pisanja u fazu ispitivanja programa i obrnuto.

Modularna građa programa

definicija funkcije →

definicija funkcije →

definicija funkcije →

glavni program →

```
#Modul: prirodni.py
'''
   modul prirodni
   sadrži funkcije za operacije s prirodnim brojevima
'''

def djelitelji(n):
    '''Funkcija vraća listu svih djelitelja broja n'''
    djel = []
    for d in range(1, n + 1):
        if n % d == 0:
            djel.append(d)
    return djel

def prost(n):
    '''Funkcija vraća vrijednosti:
         True   ako je broj prost
         False  ako je broj složen
    '''
    d = 2
    while d ** 2 <= n:
        if n % d == 0:
            return False
        d += 1
    return True

def prosti_brojevi(n):
    '''Funkcija vraća listu prostih brojeva manjih ili jednakih n'''
    return  [i for i in range(2, n + 1) if prost(i)]

if __name__ == '__main__':
    n = int(input('Upiši broj n = '))
    print('U intervalu [2, {}] su prosti: \n{}'.format(n, prosti_brojevi(n)))
    print('Djelitelji broja {} su: \n{}'.format(n, djelitelji(n)))
```

dokumentacijski string → modula

dokumentacijski string → funkcije

dokumentacijski string → funkcije

dokumentacijski string → funkcije

```
#Modul: prirodni.py
'''
   modul prirodni
   sadrži funkcije za operacije s prirodnim brojevima
'''

def djelitelji(n):
    '''Funkcija vraća listu svih djelitelja broja n'''
    djel = []
    for d in range(1, n + 1):
        if n % d == 0:
            djel.append(d)
    return djel

def prost(n):
    '''Funkcija vraća vrijednosti:
         True   ako je broj prost
         False  ako je broj složen
    '''
    d = 2
    while d ** 2 <= n:
        if n % d == 0:
            return False
        d += 1
    return True

def prosti_brojevi(n):
    '''Funkcija vraća listu prostih brojeva manjih ili jednakih n'''
    return  [i for i in range(2, n + 1) if prost(i)]

if __name__ == '__main__':
    n = int(input('Upiši broj n = '))
    print('U intervalu [2, {}] su prosti: \n{}'.format(n, prosti_brojevi(n)))
    print('Djelitelji broja {} su: \n{}'.format(n, djelitelji(n)))
```

prazna lista →

dodavanje djelitelja →

n nije prost →

n je prost →

konstrukcija liste →

```python
#Modul: prirodni.py
'''
   modul prirodni
   sadrži funkcije za operacije s prirodnim brojevima
'''

def djelitelji(n):
    '''Funkcija vraća listu svih djelitelja broja n'''
    djel = []
    for d in range(1, n + 1):
        if n % d == 0:
            djel.append(d)
    return djel

def prost(n):
    '''Funkcija vraća vrijednosti:
         True   ako je broj prost
         False  ako je broj složen
    '''
    d = 2
    while d ** 2 <= n:
        if n % d == 0:
            return False
        d += 1
    return True

def prosti_brojevi(n):
    '''Funkcija vraća listu prostih brojeva manjih ili jednakih n'''
    return  [i for i in range(2, n + 1) if prost(i)]

if __name__ == '__main__':
    n = int(input('Upiši broj n = '))
    print('U intervalu [2, {}] su prosti: \n{}'.format(n, prosti_brojevi(n)))
    print('Djelitelji broja {} su: \n{}'.format(n, djelitelji(n)))
```

```python
#Modul: prirodni.py
'''
   modul prirodni
   sadrži funkcije za operacije s prirodnim brojevima
'''

def djelitelji(n):
    '''Funkcija vraća listu svih djelitelja broja n'''
    djel = []
    for d in range(1, n + 1):
        if n % d == 0:
            djel.append(d)
    return djel

def prost(n):
    '''Funkcija vraća vrijednosti:
         True   ako je broj prost
         False  ako je broj složen
    '''
    d = 2
    while d ** 2 <= n:
        if n % d == 0:
            return False
        d += 1
    return True

def prosti_brojevi(n):
    '''Funkcija vraća listu prostih brojeva manjih ili jednakih n'''
    return  [i for i in range(2, n + 1) if prost(i)]

if __name__ == '__main__':
    n = int(input('Upiši broj n = '))
    print('U intervalu [2, {}] su prosti: \n{}'.format(n, prosti_brojevi(n)))
    print('Djelitelji broja {} su: \n{}'.format(n, djelitelji(n)))
```

ulazni string s pretvorbom u broj →

```python
#Modul: prirodni.py
'''
   modul prirodni
   sadrži funkcije za operacije s prirodnim brojevima
'''

def djelitelji(n):
    '''Funkcija vraća listu svih djelitelja broja n'''
    djel = []
    for d in range(1, n + 1):
        if n % d == 0:
            djel.append(d)
    return djel

def prost(n):
    '''Funkcija vraća vrijednosti:
         True   ako je broj prost
         False  ako je broj složen
    '''
    d = 2
    while d ** 2 <= n:
        if n % d == 0:
            return False
        d += 1
    return True

def prosti_brojevi(n):
    '''Funkcija vraća listu prostih brojeva manjih ili jednakih n'''
    return  [i for i in range(2, n + 1) if prost(i)]

if __name__ == '__main__':
    n = int(input('Upiši broj n = '))
    print('U intervalu [2, {}] su prosti: \n{}'.format(n, prosti_brojevi(n)))
    print('Djelitelji broja {} su: \n{}'.format(n, djelitelji(n)))
```

priprema i ispis izlaznih stringova →

Pokrenemo li modul kao program i upišemo broj 50, ispis će izgledati ovako:

```
Upiši broj n = 50
U intervalu [2, 50] su prosti:
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
Djelitelji broja 50 su:
[1, 2, 5, 10, 25, 50]
```

Za svaki modul koji importiramo u interaktivno sučelje s `import ime_modula` možemo naredbom `help(ime_modula)` ispisati opis modula i popis funkcija s njihovim dokumentacijskim stringovima. Za naš modul `prirodni` taj ispis izgleda ovako:

```
>>> import prirodni
>>> help(prirodni)
Help on module prirodni:

NAME
prirodni

DESCRIPTION
modul prirodni
sadrži funkcije za operacije s prirodnim brojevima

FUNCTIONS
djelitelji(n)
    Funkcija vraća listu svih djelitelja broja n

prost(n)
    Funkcija vraća vrijednosti:
    True   ako je broj prost
    False  ako je broj složen

prosti_brojevi(n)
    Funkcija vraća listu svih prostih brojeva od 2 do n
```

HVALA NA PAŽNJI !