



CS324 - SKRIPTING JEZICI

Uvod u Python 3

Lekcija 02

PRIRUČNIK ZA STUDENTE

CS324 - SKRIPTING JEZICI

Lekcija 02

UVOD U PYTHON 3

- ✓ Uvod u Python 3
- ✓ Poglavlje 1: Uvod u Python 3
- ✓ Poglavlje 2: Istorija Python programskog jezika
- ✓ Poglavlje 3: Razlika između verzija Python programskog jezika
- ✓ Poglavlje 4: Primene Python programskog jezika
- ✓ Poglavlje 5: Python 3 razvojno okruženje (IDE)
- ✓ Poglavlje 6: Pokazna vežba #2
- ✓ Poglavlje 7: Individualna vežba #2
- ✓ Poglavlje 8: Domaći zadatak #2
- ✓ Zaključak

Copyright © 2017 - UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

U drugoj lekciji govorićemo o Python programskom jeziku kao jednom od najpopularnijih skripting jeziku današnjice.

U drugoj lekciji govorice se o Python programskom jeziku kao jednom od najpopularnijih skripting jeziku današnjice. Upravo zbog toga se samo Python obraduje na ovom predmetu.

O popularnosti Python jezika govori činjenica da su sajtovi poput **Google**-a i **reddit**-a pisani u Python-u, a ali i **Netflix**, **Spotify**, **Pinterest**, **Uber** i **Lyft**, kao i **Instagram**.

Python se trenutno koristi u trećoj verziji, dok je prethodna verzija, python 2, i dalje zastupljena.

Najbitnije razlike jesu te da Python 3 ima jednostavniju sintaksu. Srećom, sintaksa je dovoljno slična i kod se može lako protumačiti.

O drugim razlikama između verzija Python-a biće reči u nastavku lekcije.

Rečeno je da se Python, kao i svi skripting jezici, primenjuje najviše u razvoju web aplikacija, ali to nije jedina primena.

Pored Web-a, Python se dosta koristi i u Data Science, u algoritmima veštačke inteligencije, prvenstveno u mašinskom učenju, u NLP-u, u razvoju video igara, ali i u inženjerskim i naučnim primenama.

Na kraju lekcije biće reči kako o najčešće korišćenim okruženjima za razvoj aplikacija u Python verzije 3.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 1

Uvod u Python 3

UVOD U PYTHON 3 PROGRAMSKI JEZIK

Primenom Python programskog jezika, programer se fokusira na rešavanje problema, a ne na sintaksu.

Python programski jezik je programski jezik otvorenog koda (en.[open-source](#)) koji se najviše koristi u [web programiranju](#), u [data science](#)-u, [veštačkoj inteligenciji](#), i u mnogim [naučnim primenama](#).

Učenje Python programskog jezika omogućava programeru za se fokusira na rešavanje problema, a ne da se fokusira na samu sintaksu jezika.

Jednostavna sintaksa daje Python programskom jeziku prednost u odnosu na druge jezike kao što su C++ ili Java, dok sa druge strane, pregršt biblioteka omogućava Python jeziku da rešava mnoge probleme.

Opis Python programskog jezika

Python predstavlja interpretirani, objektno-orientisani programski jezik visokog nivoa sa dinamičkim sistemom tipiziranja podataka.

Zbog osobina visokog nivoa koje su ugrađene u strukture podataka, zajedno sa dinamičkim sistemom tipiziranja i dinamičkim povezivanjem, veoma je pogodan za brz razvoj aplikacija (en. [Rapid Application Development](#)), za primene kao skripting jezik, ali i za primene kao lepljiv jezik koji povezuje postojeće komponente zajedno.

Sintaksa Python jezika koja je jednostavna i laka za korišćenje povećava samu čitljivost programa, i na taj način smanjuje cenu održavanja programa pisana u Python jeziku.

Python interpreter i bogata biblioteka standardnih funkcija dostupni su u izvornom kodu ili u binarnom obliku besplatno za sve (glavne) računarske platforme, i mogu se besplatno distribuirati.

PRODUKTIVNOST U PYTHON PROGRAMSKOM JEZIKU

Produktivnost Python programskog jezika ogleda se u brzom ažuriraj-testiraj-debuguj ciklusu

Programeri često biraju, i ostaju sa Python programskim jezikom zbog povećanja produktivnosti koji Python pruža. Budući da ne postoji korak kompajliranja, ciklus ažuriraj-

testiraj-debaguj (en. [edit-test-debug](#)) je neverovatno brz, pogotovo u porečenju sa programskim jezicima koje sadrže kompjajler.

[Debagovanje](#) u Python programskom jeziku je lako: [bag](#) ili loš unos neće izazvati grešku segmentacije.

Podsetnik:

Greška segmentacije dešava se kada program pokuša da pristupi memorijskoj lokaciji kojoj ne treba da pristupi, ili da pokuša da pristupi memorijskoj lokaciji na način na koji ne treba da pristupi (primer: da piše na read-only lokaciji, ili da upiše preko dela operativnog sistema).

Kada Python interpreter pronađe grešku, podiće će izuzetak. Kada program ne uhvati izuzetak, interpreter štampa [trag steka](#) (en. [stack trace](#)).

Debager na izvornom nivou (en. [source level debugger](#)) dozvoljava pregled lokalnih i globalnih promenljivih, evaluaciju proizvoljnih izraza, postavljanja [breakpoint](#)-ova, kao i pregled koda liniju po liniju. Sam debager je pisan takođe u Python jeziku.

Često je najkraći način da se debaguje program tako što se dodaju nekoliko print izraza u izvornom kodu - zbog brzog [ažuriraj-testiraj-debaguj](#) ciklusa, ovaj jednostavni pristup je jako efektivan.

▼ Poglavlje 2

Istorija Python programskog jezika

POČETAK PYTHON JEZIKA

Tvorac Python jezika, Guido van Rossum, razvojao je Pyzhon jezik od druge polovine 80tih godina prošlog veka.

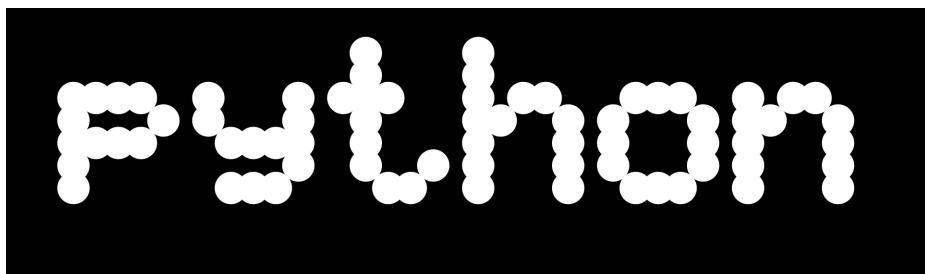
Programski jezik Python nastao je kao ideja kasnih 80tih godina 20. veka, a njegova prva implementacija počela je 1989. godine.

Tvorac Python jezika je Guido van Rossum, koji je radio u CWI (nl. [Centrum Wiskunde & Informatica](#)), razvojnom centru za matematiku i teoretsku računarsku nauku, koji pripada Holandskoj Organizaciji za Naučna Istraživanja (en. [Netherlands Organization for Scientific Research](#), NWO) u Amsterdamu.

Python predstavlja naslednika programskog jezika ABC, ali koji je mogao da obrađuje izuzetke i da ima interfejs ka Amoeba operativnim sistemom (koji je razvio Andrew Tanenbaum u Vrije univerzitetu u Amsterdamu). Van Rossum je nastavio da razvija Python programski jezik sve do 2018. godine.

Zanimljivost:

Python je dobio ime po seriji Leteći Cirkus Montija Pajtona (en. *Monty Python's Flying Circus*)



Slika 2.1 Logo Python jezika iz devedesetih godina prošlog veka. [Izvor: [wikipedia.com/python](https://en.wikipedia.org/wiki/Python_(programming_language))]

Rana istorija Python programskog jezika

Februara 1991. godine, van Rossum je objavio kod (v0.9.9) koji je već u toj fazi razvoja imao funkcionalnosti kao što su podrška za klase sa naslednošću, obrada izuzetaka, podrška za funkcije, ali i tipove podataka kao što su *list*, *dict*, *str*.

U ovom, inicijalnom izdanju, postojao je i sistem modula po ugledu na Modula-3 programski jezik. Van Rossum je opisao modul kao "jedan od glavnih osobina Python jezika".

PYTHON PROGRAMSKI JEZIK V1

Van Rossum je nastavio razvoj Python programskog jezika i nakon odlaska iz CWI.

Već januara 1994. godine Python je izdat u verziji 1.0.

Glavne osobine koje su bile uključene u ovom izdanju jesu vili alati za funkcionalno programiranje (konkretno *lambda*, *map*, *filter*, *reduce*).

Zanimljivost:

*Van Rossum je izjavio da je "Python 'peuzeo' funkcije *lambda()*, *filter()*, i *map()* zahvaljujući Lisp kajeru koji ih je zaboravio i poslao kao zakrpe".*

Poslednja verzija koja je izdata dok je van Rossum radio u CWI bila je Python 1.2.

Nakon toga, van Rossum se seli u Reston, Virdžinija, SAD, gde dobija posao u Korporaciji za Nacionalna Istraživanja (en. **Corporation for National Research Initiatives**, CNRI). Odatle je izdao još nekoliko verzija Python jezika.

Verzija 1.4 donela je nove opcije, za koje je Modula-3 jezik ponovo bio inspiracija.

Najbitniji noviteti bili su imenovani parametri (en. **named parameters**, **named arguments** ili **keyword arguments**), kao i podršku za kompleksne brojeve.

Podsetnik:

Imenovani parametri opisuju podršku za pozive funkcije koji jasno opisuju ime svakog parametra unutar poziva funkcije.

Razvojni tim Python programskog jezika se 2000. godine preselio BeOpen.com da bi oformili BeOpen PythonLabs razvojni tim. Poslednja verzija pri CNRI bila je Python 1.6

Pošto je Python 1.6 bio pod licencom CNRI, klauzula u licenci je opisivala da je zakon o licenciranju bio podložan zakonima u saveznoj državi Virdžiniji. Zbog toga, Python verzija 1.6 nije bila kompatibilna sa *GNU General Public Licence*, pa je zbog toga naknadno izdata verzija 1.6.1, koja je suštinski bila ista kao 1.6, iz manje korekcije grešaka, ali bitnije, sa novom *GNU GPL* licencom.

PYTHON PROGRAMSKI JEZIK V2

Python v2 je imao životni vek od 20 godina, zaključno sa aprilom 2020. godine

Python 2.0 je objavljen oktobra 2000. godine, i uveo je nove mogućnosti, prvenstveno razumevanje lista (en. [list comprehensions](#)), osobina koja se koristi u funkcionalnom programiranju, prvenstveno u jezicima SETL i Haskell.

Python 2.0 je takođe uvek i sistem za skupljanje otpada (en. [garbage collection system](#)).

Podsetnik:

Razumevanje lista je konstrukt u pojedinim programskim jezicima koji služi za pravljenje listi na osnovu postojećih listi.

Podsetnik:

Sistem sakupljanja otpada jeste oblike automatskog upravljanja memorijom. Sakupljač otpada (collector), pokušava da povrati otpad, tj. memoriju koju su zauzeli objekti koje program više ne koristi.

Python 2.1 je izašao pod licencom Python Software Foundation Licence, po ugledu na Apache Software Foundation.

Python 2.1 je uveo podršku za ugnježdene oblasti vidljivosti (en. [nested scope](#))

Python 2.2 je izdat decembra 2001. godine, i glavni novitet bio je objedinjenje tipova podataka i klasa u jedinstvenu hijerarhiju. Na ovaj način Python model objekata je postao zaista objektno-orientisan.

Novitet koji je uveo Python 2.5 (izdat septembra 2006. godine) jeste uvođenje ključne reči [with](#), koja enkapsulira deo koda unutar kontekstnog menadžera.

Od Python 2.6 krenula su paralelna izdanja 2.x i 3.x verzija

Python 2.6 je trebalo da bude izdat paralelno sa Python 3, sa nekim funkcionalnostima koje je trebalo da budu u novom izdanju.

Slično se desilo i sa 2.7, koja je izdata paralelno sa 3.1, u junu 2009. godine.

Od tada, prestala su paralelna izdanja dve verzije, i trenutno je Python 2.7 poslednja verzija 2.x

Novembra 2014. najavljeno je da će podrška za Python 2.x prestati 2020. godine uz preporuku korisnicima da pređu na Python v3.

Poslednja verzija Python v2, verzija 2.7.18 je izdata aprila 2020 godine i predstavlja kraj životnog ciklusa Python 2.x programskog jezika.

PYTHON PROGRAMSKI JEZIK V3

Python 3 je projektovan u cilju da popravi fundamentalne greške u dizajnu jezika.

Python 3.0 izdat je decembra 2008. godine, uz kodne nazine Python 3000 ili Py3K.

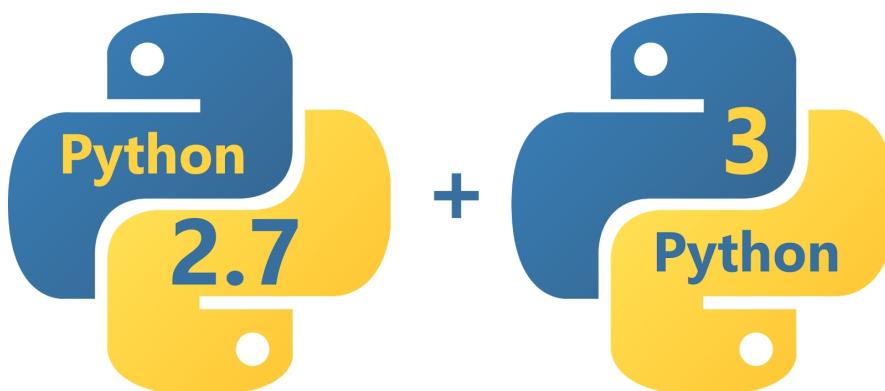
Python 3 je projektovan u cilju da popravi fundamentalne greške u dizajnu jezika.

Ove promene s nisu mogle uraditi a da se zadrži kompatibilnost unazad (en. **backwards compatibility**) sa izdanjima iz serije 2.x.

To je jedan od razloga zašto se sa pojavom Python 2.6 javio i Python 3.

Razvojni princip pri projektovanju Python jezika 3 je bio "ukloniti duplikaciju osobina uklanjanjem starih načina rada"-

Python 3 je razvijen sa istom filozofijom kao i u prethodnim izdanjima. Međutim, u skladu sa razvojnim principom, Python 3.0 je stavio akcenat na uklanjanje više istih modula, sa ciljem da "treba da bude jedan i samo jedan način da se nešto uradi, i da taj način bude očigledan".



Slika 2.2 Python 2.7 naspram Python 3. [Izvor: wikipedia.com/python]

Python 3.0 je ostao programski jezik sa više paradigm.

Programeri su i dalje mogli da pišu programe koji iz proceduralne, objektno-orientisane, ali i funkcionalne paradigm. Python 3.0 je nastojao da odabir paradigmne pri pisanju bude još jasniji u odnosu na prethodne 2.x verzije.

Kao što je rečeno, Python 3. je prestao sa kompatibilnošću unazad, i programi koji su napisani u Python 2.x programskom jeziku ne mogu raditi u Python 3.0, ukoliko se prvenstveno ne modifikuju.

Dinamičko tipiziranje podataka u kombinaciji sa promenama semantike pojedinih metoda učinilo je da je prevođenje (ne u smislu kompajliranje) koda iz Python 2.x u Python 3 jako teško.

GLAVNE OSOBINE PYTHON 3.0

Python 3.0 programski jezik je uvek mnoge funkcionalnosti, i doveo do prestanka razvoja 2.x verzija

Glavne osobine Python 3.0 jezika jesu sledeće:

- Menjanje print komande da sada postaje funkcija, a ne izjava. Na ovaj način lakše se mogu koristiti druge funkcije za štampanje ukoliko je potrebno,
- Uklanjanje `input` funkcije (iz prethodnih verzija Pythona), i preimenovanje `raw_input` (iz prethodnih verzija) u `input` (u verziji 3.x),
- Dodata podrška za opcionu anotaciju funkcija koje se mogu koristiti za deklaraciju neformalnih tipova,
- Unificiranje `str` / `unicode` tipova podataka koji predstavljaju tekst, i dodavanje `bytes` i `bytearray` tipova, koji predstavljaju nizove bajtova,
- Uklanjanje kompatibilnost unazad,
- Promena u deljenju celih brojeva.

Sa svakom novom verzijom nakon 3.0, Python je uveo nove funkcionalnosti

Verzija Python programskog jezika u vreme pisanja lekcije je 3.8, i puna dokumentacija se može naći na:

<https://docs.python.org/3/whatsnew/3.8.html>

Primer: Deljenje celih brojeva

Napomena: simulira se izlaz konzole, gde je `>>>` prompt koji Python izbacuje.

```
#Python v2.X
>>> 5/2
>>> 2
```

```
#Python 3.0
>>> 5 / 2
>>> 2.5
```

Za celobrojni rezultat, koristi se isključivo `//`

```
#Python 3.0
>>> 5 // 2
>>> 2
```

▼ Poglavlje 3

Razlika između verzija Python programskog jezika

DA LI TREBA I DALJE KORISTITI PYTHON 2.X?

Od aprila 2020. godine Python 2.x ne dobija podršku od razvojnog tima.

Prema preporukama razvojnog tima Python programskog jezika, svaki novi razvoj softvera koji se piše u Python-u, treba pisati u Python-u 3.x.

Kao što je već rečeno, najavljeno je da će početkom 2020. godine Python 2.x izgubiti podršku razvojnog tima, tj. doći će do kraja svog životnog ciklusa (en. [End of Life](#)). Ovo se desilo aprila 2020. godine, i od tada Python 2.x ne dobija nova ažuriranja.

Ažuriranja se odnose na bagove, ali i na bezbednosne ispravke. Zbog nedostatka podrške, moguće je naći nove bezbednosne ranjivosti u programiranim napisanim u Python 2.X verziji, što predstavlja veliki propust.

Zanimljivost:

Poslednja verzija Python 2.x programskog jezika jeste 2.7.18 iz 20. aprila 2020. godine.

<https://www.python.org/downloads/release/python-2718/>

U nastavku objekta učenja biće dati konkretni primeri sa razlikama između [Python 2.x](#) i [Python 3.x](#)

Neke od funkcija koje se spominju još uvek nisu uvedeni kroz predmet, ali su dovoljno jednostavne da se razumeju imajući u vodu znanje iz sličnih predmeta.

PYTHON 2.X VS 3.X - PRINT

U Python 2.x print je ključna reč, dok je u 3.x funkcija

Najpoznatija promena, i najtrivijalnija, ali i dalje bitna jeste promena u sintaksi print funkcije.

[Python 2.x](#) je imao print ključnu reč (en. statement), dok je u [Python 3.x](#) jeziku potrebno ubaciti zagrate, što pretvara u funkciju.

Python 2.x:

```
print 'Hello, World!'
print('Hello, World!')
print "text", ; print 'print more text on the same line'
```

```
Hello, World!
Hello, World!
text print more text on the same line
```

Python 3.x:

```
print('Hello, World!')
print("some text,", end="")
print(' print more text on the same line')
```

```
Hello, World!
some text, print more text on the same line
```

Ukoliko bismo hteli da koristimo sintaksu 2.x u 3.x, dobili bismo sledeće:

Python 3.x:

```
print 'Hello World!'
```

```
File "<stdin>", line 1
    print 'Hello World!'
    ^

```

```
SyntaxError: Missing parentheses in call to 'print'. Did you mean print('Hello
World!')?
```

PYTHON 2.X VS 3.X - DELJENJE CELIH BROJEVA

Kada se izvršava Python 3.x kod koji ima deljenje celih brojeva u 2.x interpretalu, može doći do bitnih promena i grešaka.

Jedna od bitnijih promena kod [Python 3.x](#) u odnosu na [Python 2.x](#) jeste rezultat deljenja celih brojeva.

Ova promena se često previdi, i ukoliko se ne vodi računa može izazvati velike greške (dok se ne pronađe i ispravi).

Python 2.x:

```
print '3 / 2 =', 3 / 2
print '3 // 2 =', 3 // 2
print '3 / 2.0 =', 3 / 2.0
print '3 // 2.0 =', 3 // 2.0
```

```
3 / 2 = 1
3 // 2 = 1
3 / 2.0 = 1.5
3 // 2.0 = 1.0
```

Python 3.x:

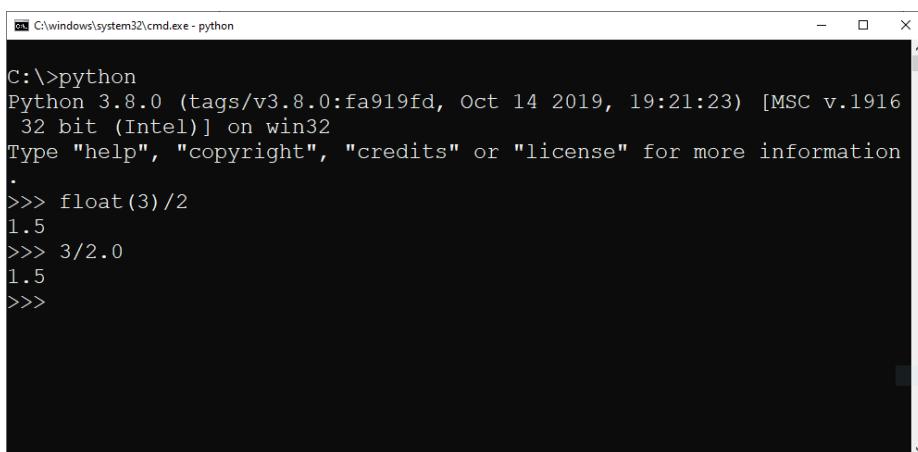
```
print('3 / 2 =', 3 / 2)
print('3 // 2 =', 3 // 2)
print('3 / 2.0 =', 3 / 2.0)
print('3 // 2.0 =', 3 // 2.0)
```

```
3 / 2 = 1.5
3 // 2 = 1
3 / 2.0 = 1.5
3 // 2.0 = 1.0
```

U Python 3.x se može koristiti `float(3)/2` ili `3/2.0` da bi se kod mogao koristiti i u Python 2.0 jeziku.

```
float(3)/2
3/2.0
```

```
1.5
1.5
```



Slika 3.1 Ukoliko se program koristi i u Python 2.x, sigurnije je koristiti float(). [Izvor: Autor]

PYTHON 2.X VS 3.X - UNICODE FORMATIRANJE I BYTE TIP PODATAKA

U Python 2.x jeziku razdvojeni su str i unicode tipovi podataka, dok su u Python 3.x jeziku oba formata u str tipu.

U Python 2.x jeziku razdvojeni su str i unicode tipovi podataka, dok su u Python 3.x jeziku oba formata u str tipu.

Python 2.x:

```
print type(unicode('this is like a python3 str type'))
```

```
<type 'unicode'>
```

Python 3.x:

```
print('strings are now utf-8 \u03BCnico\u0394Irish!')
```

```
strings are now utf-8 μnicoΔIrish!
```

Takođe, ne postoji tip podataka byte i bytearray u Python 2.x jeziku

Python 2.x:

```
print type(b'byte type does not exist')
```

```
<type 'str'>
```

Python 3.x:

```
print('Python 3.x has', type(b' bytes for storing data'))
```

```
Python 3.x has <class 'bytes'>
```

PYTHON 2.X VS 3.X - OBRADA IZUZETAKA

U Python 3.x programskom jeziku ubačena je ključna reč as pri obradi izuzetaka.

Obrada izuzetaka je promenjena u Python 3 programskom jeziku. Za razliku od [Python 2.x](#), u [Python 3.x](#) jeziku treba da se koristi ključna reč [as](#).

Python 2.x:

```
try:  
    let_us_cause_a_NameError  
except NameError, err:  
    print err, '--> our error message'
```

```
name 'let_us_cause_a_NameError' is not defined --> our error message
```

Python 3.x:

```
try:  
    let_us_cause_a_NameError  
except NameError as err:  
    print(err, '--> our error message')
```

```
name 'let_us_cause_a_NameError' is not defined --> our error message
```

PYTHON 2.X VS 3.X - PARSIRANJE PODATAKA U INPUT() FUNKCIJI

U Python 3.x jeziku [input\(\)](#) funkcija uvek parsira u str tip potadaka

U [Python 2.x](#) jeziku funkcija [input\(\)](#) je parsirala ulaz u različite tipove podataka, dok se za parsiranje u [str](#) tipove podataka sada koristila [raw_input\(\)](#) funkcija. U [Python 3.x](#) jeziku, [input\(\)](#) uvek parsira ulaz kao [str](#) tip podataka.

Python 2.x:

```
>>> my_input = input('enter a number: ')  
enter a number: 123
```

```
>>> type(my_input)
<type 'int'>

>>> my_input = raw_input('enter a number: ')

enter a number: 123

>>> type(my_input)
<type 'str'>
```

Python 3.x:

```
>>> my_input = input('enter a number: ')

enter a number: 123

>>> type(my_input)
<class 'str'>
```

Možemo primetiti da se bilo koji ulaz parsira kao string.

Slika 3.2 Parsitanje pomoću input() funkcije. [Izvor: Autor]

▼ Poglavlje 4

Primene Python programskog jezika

PRIMENE PYTHON PROGRAMSKOG JEZIKA - WEB I GAME DEVELOPMENT

Python programski jezik je poznat po svojoj opštoj primeni u svakom domenu razvoja softvera.

Python programski jezik je poznat po svojoj opštoj primeni u gotovo svakom domenu razvoja softvera. Python kao takav predstavlja koji je doživeo najveći uspon za razvoj aplikacija.

U nastavku su date najpoznatije primene Python programskoj jeziku

Web development

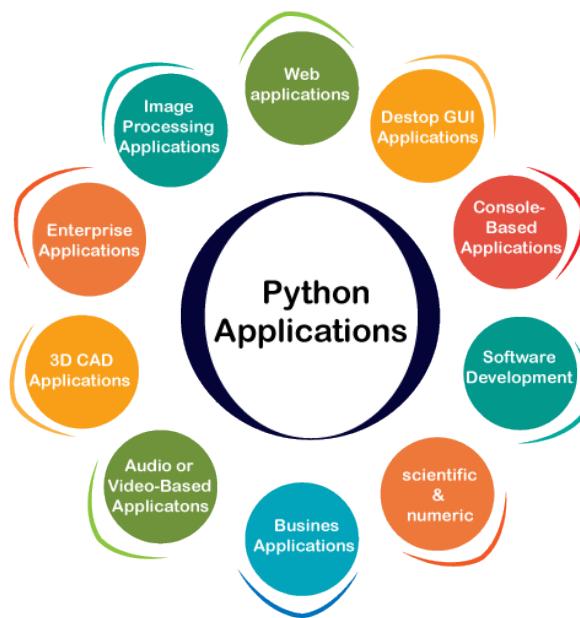
Python programski jezik se može koristiti za brz razvoj web aplikacija. Python koristi framework-ove i za *common-backend* logiku i biblioteke koji mogu pomoći za integriranje protokola kao što su HTTPS, FTP, SSL i drugi. Takođe može procesuirati JSON, XML, e-mail i druge tipove.

Najpoznatiji framework-ovi za Python jesu Django i Pyramid za "teže" aplikacije, dok se Flask koristi kao mikro-framework. Korišćenjem ovih framework-a dobija se na sigurnosti, skalabilnosti i ujedno i na jednostavnosti, pogotovo u poređenju sa razvojem web-stranice od nule.

Razvoj igara

Python se koristi u razvoju interaktivnih video igara. Korišćenjem biblioteka kao što su PySoy koji predstavlja 3D endžin (en. *engine*) sa podrškom za Python 3, PyGame koji pruža funkcionalnost. Poznate igre koje su pisane (bilo u celosti ili delom) u Python jeziku:

- Battlefield 2
- Civilization IV
- Sims 4
- World of Tanks
- EVE Online



Slika 4.1 Primene Python programskog jezika. [Izvor: <https://www.javatpoint.com/python-applications>]

PRIMENE PYTHON PROGRAMSKOG JEZIKA - AI/MAŠINSKO UČENJE, DATA SCIENCE, KONZOLNE APLIKACIJE

Python koristi mnoge biblioteke za konkrene primene: Pandas, Scikit-Learn, NumPy, Matplotlib, REPL

Veštačka inteligencija i mašinsko učenje

Mnoge Python biblioteke kao što su Pandas, Scikit-Learn i NumPy se koriste u oblasti veštačke inteligencije, pogotovo u mašinskom učenju. Posebne lekcije obrađuju ovu temu, kao i predmet Mašinsko učenje.

Računar koristi algoritme mašinskog učenja da "uči" na osnovu svog iskustva sa prethodnim skupom podataka, ili da samostalno uči bez poznavanja ranijih izlaza.

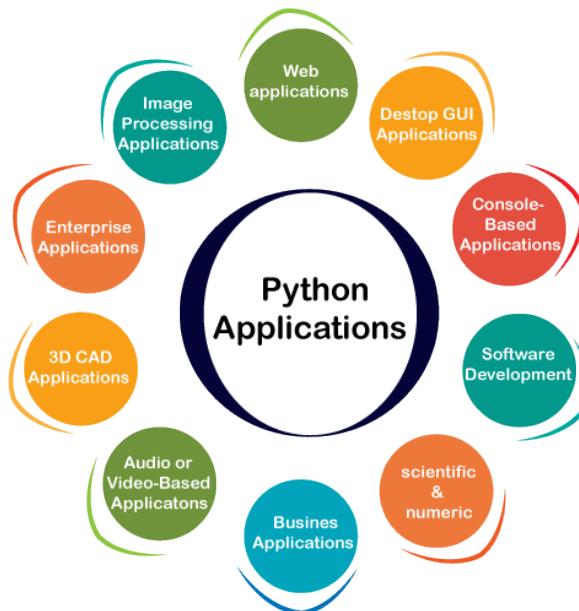
Data Science i vizuelizacija podataka

Data Science postaje jedna od vodećih oblasti istraživanja kako u akademskim krugovima tako i u privredi. Mogućnost obrade velike količine podataka postao je imperativ u današnjem svetu.

Python nudi biblioteke za laku obratu i vizuelno predstavljanje podataka. Najčešće korišćene biblioteke jesu Matplotlib i Seaborn. Sve što mogu konkretne aplikacije za vizuelno predstavljanje podataka (uključujući i Excel), može se postići i sa Matplotlib bibliotekom za iscrtavanje grafova.

Konzolne aplikacije

Konzolne aplikacije mogu se pokrenuti sa komandne linije ili shell-a. Python može razviti ovakav tip aplikacija vrlo lako i brzo. **REPL** (en. **Real-Eval-Print-Loop**) interakcijom sa Python interpretatom mogu se odmah videti izlazi programa.



Slika 4.2 Primene Python programskog jezika. [Izvor: <https://www.javatpoint.com/python-applications>]

PRIMENE PYTHON PROGRAMSKOG JEZIKA - DESKTOP GUI, WEB SCRAPING, BIZNIS, AUDIO/ VIDEO APLIKACIJE

Python koristi mnoge biblioteke za konkrene primene: Tkinter, BeautifulSoup, Tryton

Desktop GUI

Python se može koristiti i za razvoj desktop aplikacija i korisničkih interfejsa. Za razvoj desktop grafičkog korisničkog interfejsa (en. **Graphics User Interface**, GUI) najpopularnija je **Tkinter** biblioteka. Ostali alati uključuju wxWidgets, Kivy, PYQT, koji se mogu koristiti za razvoj aplikacija za različite platforme.

Aplikacije za preuzimanje podataka sa Interneta

Python se može koristiti za preuzme (en. **pull**) veliku količinu podataka s web stranica koje se nakon toga mogu koristiti u primenama kao što su poređenje cena, oglasa, razvoj i sl. Ovakav

tip aplikacija nazivaju se aplikacije za preuzimanje podataka sa Interneta (en. **Web Scraping Applications**). Python biblioteka koja se najčešće koristi za ovu primenu jeste [BeautifulSoup](#).

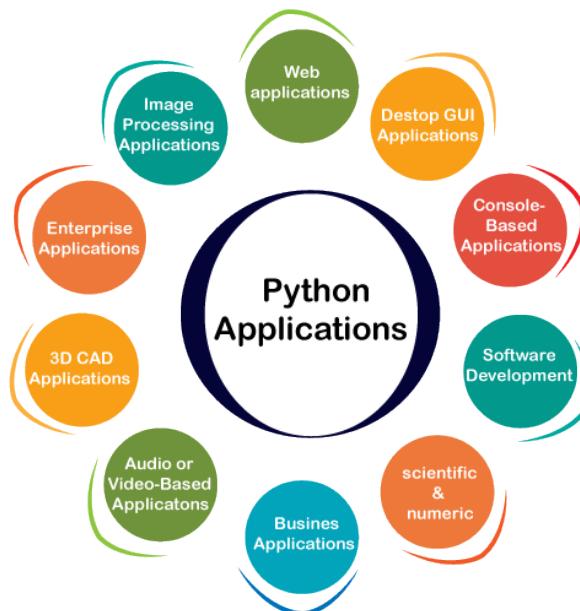
Biznis aplikacije

Biznis aplikacije su nešto različite u odnosu na standardne aplikacije, i uglavnom se bave e-commerce-om, planiranjem resursima u korporacijama (en. Enterprise Resource Planning, ERP), i sl. Aplikacije ovakvog tipa moraju biti skalabilne, proširive, i čitljive. Sve ovo Python programski jezik može da pruži kroz platformu za razvojanje biznis aplikacija kao što je [Tryton](#).

Audio & Video aplikacije

Python programski jezik se može koristiti u razvoju aplikacija koje podržavaju multitasking i na svojim izlazima imaju audio i/ili video formate datoteka.

Primeri uspešnih Audio & Video aplikacija napisanih u Python programskom jeziku uključuju TimPlayer i Cplay.



Slika 4.3 Primene Python programskog jezika. [Izvor: <https://www.javatpoint.com/python-applications>]

PRIMENE PYTHON PROGRAMSKOG JEZIKA - CAD

Python koristi mnoge biblioteke za konkrene primene: Pillow, OpenCV, SimpleITK

CAD Aplikacije

CAD (en. **Computer Aided Desing**) aplikacije mogu biti veoma komplikovane zbog mnogih komponente koje sadrže i o kojima stalno treba voditi računa (objekti i njihova reprezentacija, funkcije i sl.)

Najpoznatija aplikacija CAD tipa razvijena u Python programskom jeziku jeste Fandango.

Embedded aplikacije

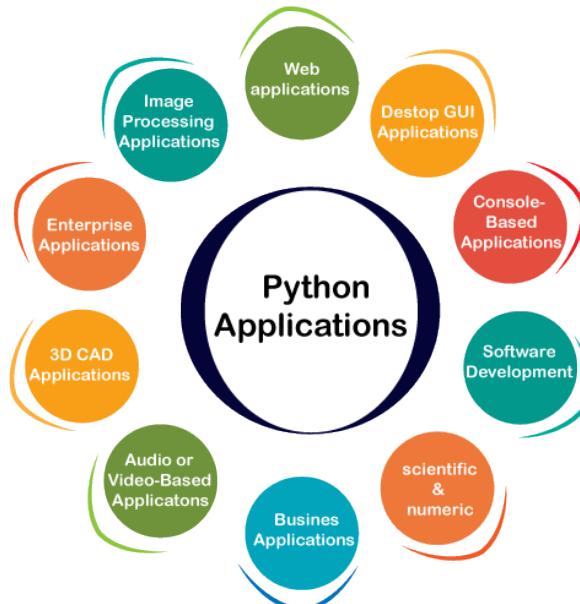
Python jezik je baziran na C jeziku pa se može iskoristiti za se napravi Embedded C softver za embedded aplikacije. Ovo omogućuje primenu aplikacija visokog nivoa na manjim uređajima koje mogu da vrše izračunavanja na Python jeziku.

Najpoznatija embedded platforma, Raspberry Pi, upravo koristi Python za svoja izračunavanja. Može se koristiti kao računar, ali i kao mikrokontroler, tj. embedded ploča za izračunavanje (en. high-level computations)

Aplikacije za procesiranje slika

Python programski jezik sadrži mnoge biblioteke koje služe za procesiranje i obradu slika. Slike se mogu manipulisati prema zahtevima korisnika.

Najpoznatije biblioteke za rad sa slikama jesu Pillow, OpenCV, SimpleITK.



Slika 4.4 Primene Python programskog jezika. [Izvor: <https://www.javatpoint.com/python-applications>]

▼ Poglavlje 5

Python 3 razvojno okruženje (IDE)

INTEGRISANO RAZVOJNO OKRUŽENJE (IDE) I EDITORI KODA

Za razvoj ozbiljnih aplikacija, potrebno je koristiti integrisano razvojno okruženje - IDE

Do sada je bilo reči o razvoju Python aplikacija preko komandne linije, tj. o pisanju programa direktno u komandnu liniju iz prozora terminala.

Ovaj pristup jeste dovoljan za vrlo jednostavne aplikacije, ali ukoliko želite da se bavite ozbiljnijim razvojem, potrebno je koristiti [integrisano razvojno okruženje](#) (en.[Integrated Development Environment](#), IDE).

Osobine IDE okruženja

IDE predstavlja skup softvera koji sadrži sve komponente za razvoj i testiranje softvera. Programer može koristiti neke ili sve alate u okviru jednog IDE okruženja.

Alati IDE okruženja uključuju editore teksta, biblioteke, kao i platforme za kompajliranje, debagovanje i testiranje.

IDE okruženje pomaže da se automatizuju zadaci programera tako što smanjuju broj zadataka koje bi inače ručno programer ratio, i ujedno spaja sve alate u jedan zajednički framework.

Da ne postoji IDE okruženje, programer bi morao ručno da obavi procese selekcije, integracije i postave (en. [deployment](#)) programa.

Editor koda

Za razliku od IDE okruženja, neki programeri preferiraju [editor koda](#) (en. [code editor](#)), koji predstavlja editor teksta u koje programer može pisati kod za razvoj bilo kog softvera. Editor obično dozvoljavaju programeru da sačuvaju te tekstualne datoteke.

Razlike između IDE okruženja i editora koda

I IDE i editor koda se mogu koristiti za razvoj softvera. Editor teksta koji sadrže obe varijante mogu pomoći programeru u pisanju skripti, modifikovanju koda i drugih operacija nad tekstrom koda.

Međutim, ukoliko programer koristi IDE, onda može obavljati i druge funkcije koje editor koda uglavnom ne poseduje.

Ove funkcije koda obično uključuju izvršenje koda, kontrolu verzije, debagovanje, interprataciju i kompajlovanje, funkcije auto-complete, auto-linting, i sl.

IDE obično poseduje i integrisani sistem za upravljanje datotekama.

Editor koda se može smatrati kao običan tekstualni editor sa nekim dodatim funkcijama (može prepoznati programski jezik).

PREGLED NAJŠEŠĆE KORIŠĆENIH EDITORA KODA: NOTEPAD++

Notepad++ je jedan od najprepoznatljivijih editora koda, nastao 2003. godine, i još uvek je aktivan, prvenstveno među korisnicima Windows operativnih sistema.

Kao što je rečeno, editor koda je tekstualni editor koji uglavnom ima neke dodatne funkcionalnosti koje olakšavaju programerima da pišu izvorni kod softvera.

Zanimljivost:

Kada bi se sve dodatne funkcionalnosti uklonile sa editora koda, dobio bi se Notepad.

Na tržištu komercijalnog softvera postoji mnogo rešenja za editore koda, od kojih su neke besplatne opcije, a neke se plaćaju, ili na bazi preplate, ili jednokratno.

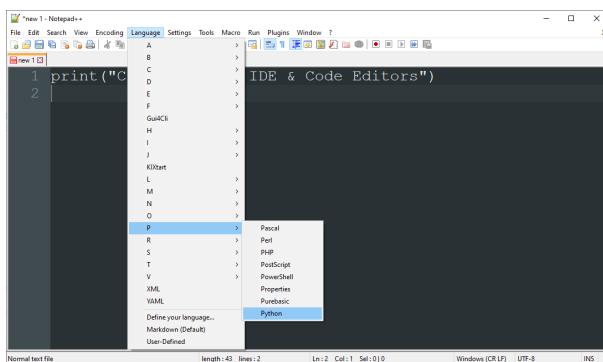
U nastavku objekta učenja biće reči o najpopularnijim editorima koda koji su besplatni.

Notepad++

Notepad++ je jedan od najprepoznatljivijih editora koda, nastao 2003. godine, i još uvek je aktivan, prvenstveno među korisnicima Windows operativnih sistema.

Notepad++ zaista podseća na Notepad, ali podržava više tabova. Najbitnija opcija jeste prepoznavanje sintakse mnogih programskeh jezika, (ključne reči ili neke standardne funkcije).

<https://notepad-plus-plus.org>



Slika 5.1 Odabir programskog jezika u Notepad++ [Izvor: Autor]

Slika 5.2 Izgled Notepad++ kada se ne selektuje jezik. [Izvor: Autor]

Slika 5.3 Izgled Notepad++ kada se selektuje jezik. [Izvor: Autor]

PREGLED NAJŠEŠĆE KORIŠĆENIH EDITORA KODA: VISUAL STUDIO CODE

Visual Studio Code predstavlja editor koda koji ima ugrađene napredne funkcije i čini ga bliži pravom IDE okruženju.

Visual Studio Code

Visual Studio Code je besplatan editor koda kojeg je razvio Microsoft 2015. godine i koji pruža pregršt mogućnosti u odnosu na mnoge editora koda (poput Notepad++), ali nije potpuni IDE kao što je puni Visual Studio IDE.

Visual Studio Code dolazi sa ugrađenom podrškom za JavaScript, TzpeScript i Node.js jezike, dok se mogu instalirati ekstencije za druge programske jezike poput C++, C#, Java, Python, PHP, Go) ali i za runtime-ove (kao što su .NET i Unity).

Visual Studio Code takođe pored prepoznavanja sintakse sadrži i IntelliSense, opciju za (polu)automatsko kompletiranje koda, koje uključuje promenljive, metode ali i uvežene module.

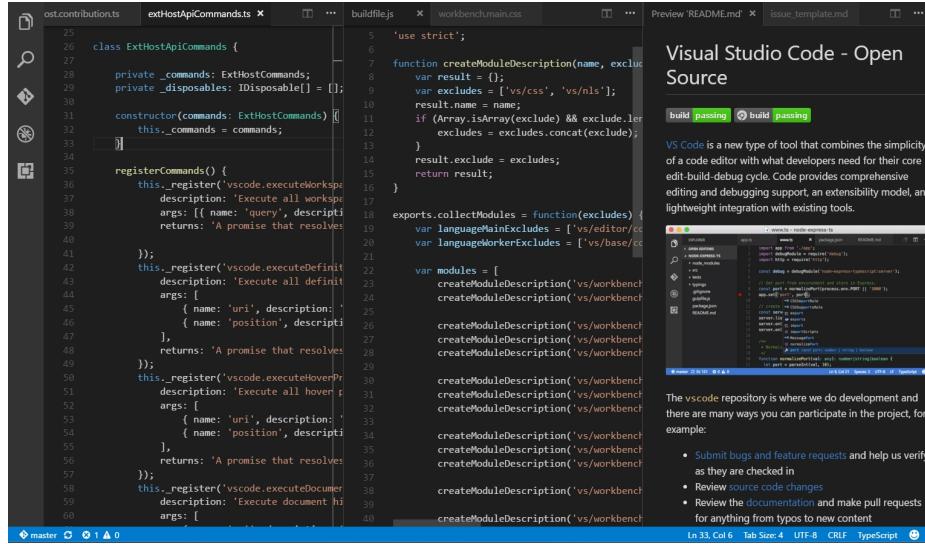
Pruža i opciju navigacije do tražene reči, preimenovanje reči odjednom na više mestu, opcije lintovanja i višestruke selekcije.

Pored osnovnih opcija editora koda, Visual Studio Code pruža i opcije debagovanja sa alatima terminala.

Konačno, Visual Studio Code nudi integraciju sa softverom za kontrolu verzije poput Git-a.

<https://code.visualstudio.com>

Treba napomenuti da Visual Studio Code, kao i ostali editori koda ne sadrži kompjajlere i interpretare, i treba njih providno instalirati i integrisati sa ovim editorom.



Slika 5.4 Izgled Visual Studio Code prozora. [Izvor: Autor]

PREGLED NAJŠEŠĆE KORIŠĆENIH IDE: SPYDER

Python Spyder IDE predstavlja jedan od najčešće korišćenih okruženja, prvenstveno za naučno programiranje.

Spyder, kao deo Anaconda distribucije Python i R programske jezike, predstavlja IDE okruženje, prvenstveno za naučno programiranje, i projektovano je u vidu za inženjere, naučnike i analitičare podataka.

Spyder je IDE otvorenog koda tj. besplatan je za preuzimanje. Može se koristiti na svim popularnim operativnim sistemima, tj. Windows, Linux i macOS.

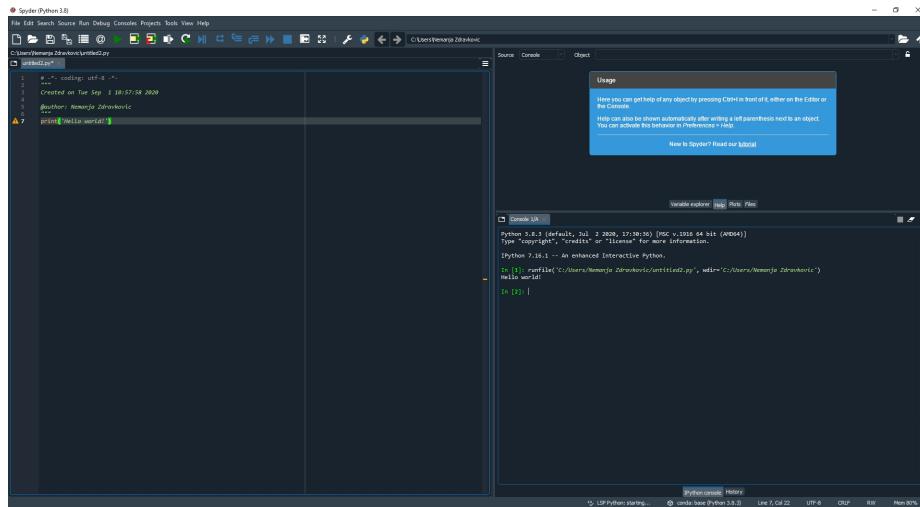
Kao pravo IDE okruženje, Spyder poseduje alate za ažuriranje koda, analizu, debagovanje i profilisanje funkcionalnosti, uz jasan pregled i korisnički interfejs.

Spyder se može preuzeti sa <https://www.spyder-ide.org>, ali i sa Anaconda distribucijom <https://www.anaconda.com/products/individual>

Spyder se može isprobati i online na sajtu MyBinder.

Izgled Spyder IDE je dat na slici.

Pored samog editora koji zauzima najveći deo prostora, konzola je uvek vidljiva i uvek se može direktno kucati u konzolu, ali takođe se interfejs može i personalizovati.



Slika 5.5 Korisnički interfejs Spyder IDE okruženja. [Izvor: Autor]

PREGLED NAJŠEŠĆE KORIŠĆENIH IDE: JUPYTER

Jupyter predstavlja interaktivno IDE okruženje bazirano na web stranicama, koje objedinjuje pisanje koda i dodavanje dodatnih elemenata

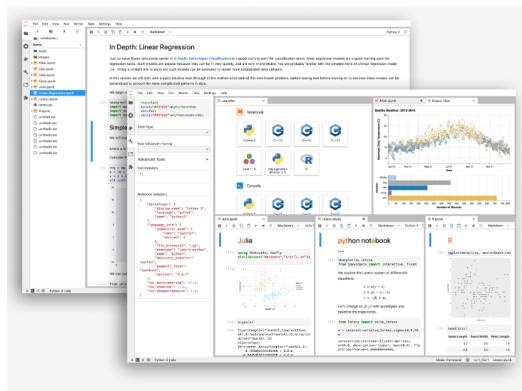
Jupyter predstavlja tzv. **notebook interface**, odnosno interativno okruženje namenjeno kako pisanju koda, tako i prikazu i vizuelizaciji podataka.

Slično kao i Spyder, Jupyter se koristi najpre u projektima naučne svrhe, najčešće za mašinsko učenje, jer podržava laku integraciju izlaznih podataka.

Jupyter je modularno okruženje koje podržava se u postojeće **sveske** lako dodaju nove komponente i da se integrišu u postojeće.

Jupyter se može isprobati i online na <https://jupyter.org>

Jupyter je deo Anaconda 3 distribucije, i automatski se instalira prilikom instalacije ove platforme.



Slika 5.6 Izgled Jupyter svezaka. [Izvor: jupyter.com]

Izgled Spyder IDE je dat na slici.

Pored samog editora koji zauzima najveći deo prostora, konzola je uvek vidljiva i uvek se može direktno kucati u konzolu, ali takođe se interfejs može i personalizovati.

Prilikom pokretanja Jupyter-a pokreće se web strana gde se mogu otvoriti .py datoteke i Jupyter sveske.



Slika 5.7 Izgled Jupyter web okruženja. [Izvor: Autor]

Slika 5.8 Ulazak u .py datoteku kroz Jupyter web okruženje. [Izvor: Autor]

PREGLED NAJŠEŠĆE KORIŠĆENIH IDE: PYCHARM

PyCharm IDE okruženje nudi opštu funkcionalnost i bliže je okruženjima za razvoj softvera na nižim programskim jezicima.

JetBrains distribucije za razvoj softvera su jako popularne kako među naučnim, tako i među komercijalnim krugovima programera.

JetBrains okruženje za razvoj Python aplikacija jeste PyCharm, koje nudi opštu funkcionalnost, ne samo za naučne svrhe (kao što je bio slučaj sa Spyder-om), već nudi opcije za lako postavljanje web aplikacija, integracija sa web servisima i druge opcije.

Međutim, PyCharm je licencirano okruženje, tj. plaća se.

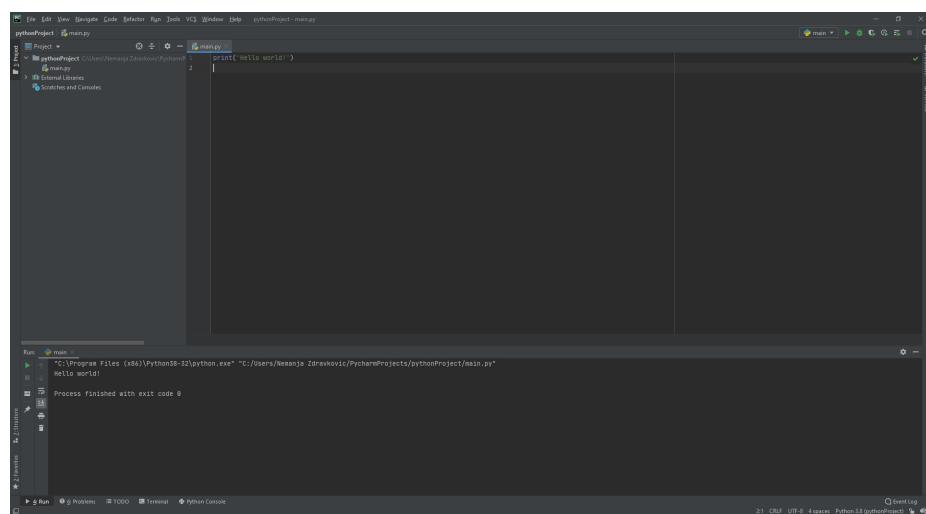
Mesečna licenca za individualnu upotrebu iznosi oko 9€, ili oko 90€ godišnje, dok za firme cena je nešto veća, oko 200€ godišnje.

Srećom, za univerzitete, odnosno studente i nastavnike, JetBrains nudi besplanu licencu sa validnom akademskom e-mail adresom, koja traje godinu dana i koja se nakon toga obnavlja.

PyCharm se može preuzeti sa sledeće stranice:

<https://www.jetbrains.com/pycharm/>

PyCharm okruženje je bliže okruženjima za razvoj softvera u nižim programskim jezicima, i takođe nudi opciju za pregled izlaza, kako kroz *run* tab, tako i kroz *terminal* tab, gde možemo kucati direktno u konzoli.



Slika 5.9 Izgled PyCharm IDE okruženja. [Izvor: Autor]

▼ Poglavlje 6

Pokazna vežba #2

UVOD U POKAZNU VEŽBU #2

Editori koda se vrlo lako instaliraju, jer u opštem slučaju ne zahtevaju već instaliran interpreter.

U pokaznoj vežbi izvršiće se instalacija Notepad++ editora koda kao i Visual Studio Code editora koda na računar sa Windows operativnim sistemom.

Nakon instalacije Visual Studio Code-a, povezaće se sa već instaliranim Python interpretrom i na taj način se može ovaj editor koda koristiti kao razvojno okruženje.

Procenjeno trajanje pokazne vežbe iznosi 45 minuta.

INSTALACIJA NOTE PAD++

Notepad++ se vrlo lako instalira, i lako i koristi jer se integriše u kontektsni meni.

Notepad++ se može preuzeti sa sajta <https://notepad-plus-plus.org/>



Slika 6.1 Preuzimanje Notepad++ editor koda. [Izvor: Autor]

Nakon jednostavne instalacije, Notepad++ možete dodatno konfigurisati u pogledu izgleda interfejsa.

Najbolja opcija Notepad++ jeste što podržava prepoznavanje sintakse mnogih programskih jezika.

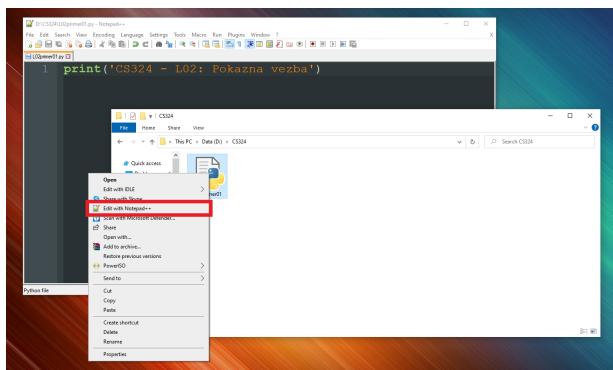
Slika 6.2 Podrška za mnoge programske jezike. [Izvor: Autor]

Nakon odabira jezika, u Notepad++ možemo da vidimo sintaksu jezika, ali ne postoje naprednije opcije kao *auto-complete* ili *IntelliSense*

Slika 6.3 Prepoznavanje sintakse u Notepad++ [Izvor: Autor]

Jedna od najčešće korišćenih "osobina" Notepad++ jeste ta da se integriše u kontekstni meni Windows-a, i bilo koja datoteka se može otvoriti i pregledati desnim klikom korišćenjem Notepad++.

Ova osobina je možda i glavni razlog zašto se Notepad++ koristi i pored postojanja ozbiljnijih editora koda.



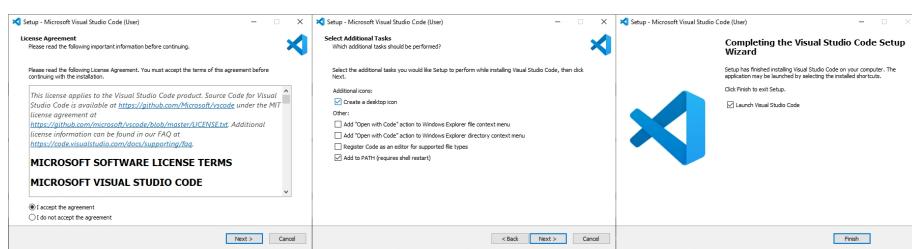
Slika 6.4 Integracija u kontekstni meni Windows-a. [Izvor: Autor]

INSTALACIJA VISUAL STUDIO CODE EDITORA KODA

Visual Studio Code se lako instalira, ali je neophodno dodati ekstenzije za Python.

Visual Studio Code se može naći na <https://code.visualstudio.com>

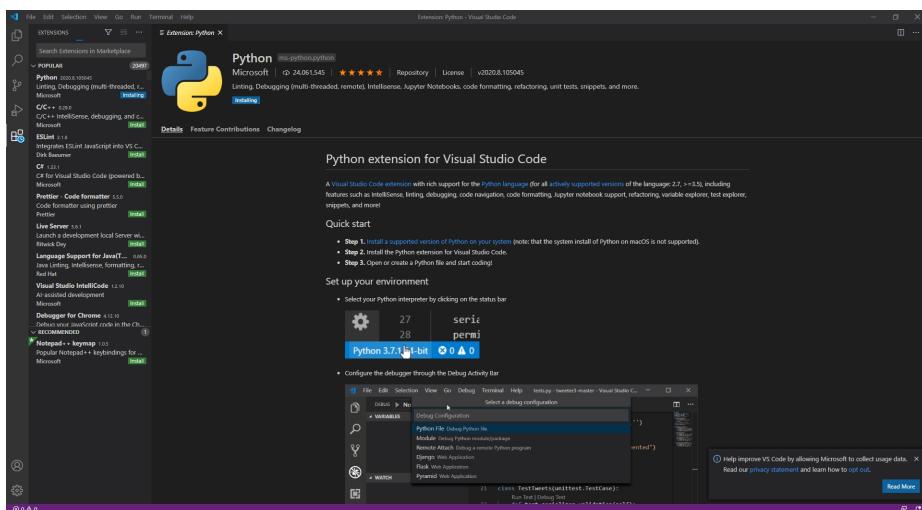
Nakon pokretanja instalacije, pratiti sve korake, i štiklirati *Add to PATH*.



Slika 6.5 Instalacija Visual Studio Code editora koda. [Izvor: Autor]

Nakon instalacije (eventualno i restarta) može se pokrenuti Visual Studio Code koji omogućava ažuriranje datoteka u raznim programskim jezicima.

Da bi Visual Studio "prepoznao" jezik, potrebno je preuzeti odgovarajuću ekstenziju sa njegove prodavnice.



Slika 6.6 Instalacija Python ekstenzije. [Izvor: Autor]

Ova ekstenzija, osim prepoznavanja sintakse jezika, ima opcije IntelliSense, linting, debagovanja, navigaciju i formatiranje koda, podršku za Jupyter sveske, pretraživač promenljivih, kao i mnoštvo drugih opcija.

ODABIR PYTHON INTERPRETERA U VISUAL STUDIO CODE-U

Pre pokretanja koda u Visual Studio Code-u, potrebno je ubaciti interpreter.

Ukoliko već postoji instaliran Python interpreter (a instaliran je direktnom instalacijom Python-a u vežbi #1), dok se može direktno i pokrenuti.

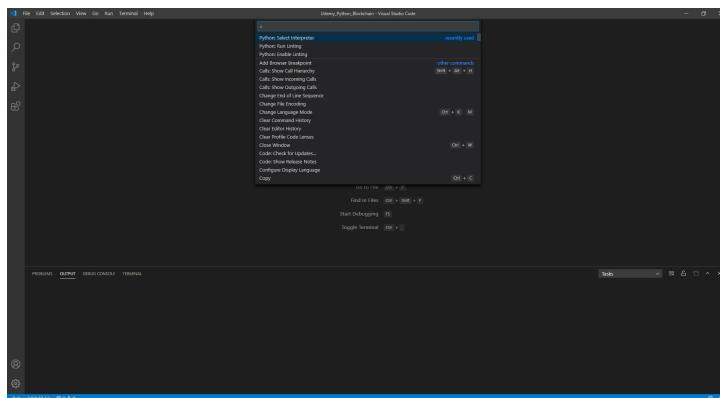
Podsetnik:

Editori koda obično ne sadrže interpreter/kompajler, i nije moguće pokrenuti program napisan u jeziku ukoliko ne postoji (i ukoliko nije povezan) interpreter ili kompajler.

Odabir Python Interpretera

Odabir Python interpretera u Visual Studio Code-u vrši se kroz sledeće korake:

1. Pokrenuti *Show All Commands* klikom na **Ctrl + Shift + P**
2. Učekati **Python: Select Interpreter**
3. Iz padajuće liste instaliranih interpretera odabratи koji interpreter želite da koristite.



Slika 6.7 Show All Commands. [Izvor: Autor]

Slika 6.8 Odabir Python Interpretera. [Izvor: Autor]

KORIŠĆENJE VISUAL STUDIO CODE EDITORA KODA - PRIMER #1

Visual Studio Code - Primer #1

Nakon odabira interpretera (Visual Studio Code i svoja Python ekstenzija podržava i Python 2.7 kao i Python 3.x) možete u editoru pisati programe.

Primer #1: Površina trougla (10 minuta)

Uneti stranice trougla a, b, i c, i onda izračunati površinu trougla po Heronovm obrascu:

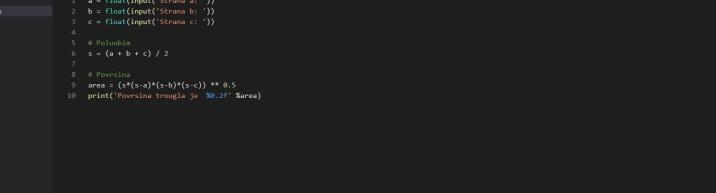
$$s = \frac{a+b+c}{2}$$

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

```
a = float(input('Strana a: '))
b = float(input('Strana b: '))
c = float(input('Strana c: '))

# Poluobim
s = (a + b + c) / 2

# Povrsina
area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
print('Povrsina trougla je %.2f' %area)
```



l02_prime01.py

```
1 #!/usr/bin/python
2
3 a = float(input("Strana a: "))
4 b = float(input("Strana b: "))
5 c = float(input("Strana c: "))
6
7 # Poluhranin
8 s = (a + b + c) / 2
9
10 # Formular
11 area = ((s-a)*(s-b)*(s-c)) ** 0.5
12 print("Povrsina trougla je", area)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\VSCode\l02 & D:\anacelci\python\www\l02\l02_prime01.py
Strana a: 3
Strana b: 5
Strana c: 5
Povrsina trougla je 6.08
PS D:\VSCode\l02
```

File Edit Selection View Go Run Terminal Help

l02_prime01.py - l02 - Visual Studio Code

Slika 6.9 Primer 1 u Visual Studio Code-u. [Izvor: Autor]

Objašnjenje koda:

`input()` funkcija unosi podatke sa tastature, kao string.

`float()` funkcija vrši konverziju podataka u razlomljen broj.

KORIŠĆENJE VISUAL STUDIO CODE EDITORA KODA - PRIMER #2 & #3

Visual Studio Code - Primer #2 i Primer #3

Primer #2: Konverzija stepeni Celzijusa u Farenhajt (10 minuta)

Uneti vrednost temperature u stepenima Celzijusa i pretvoriti u stepene Farenhajta. Konverzija se računa po formuli

$$T_F = (T_C \times 1.8) + 32$$

```
tempC = float(input('Uneti temperaturu u C: '))

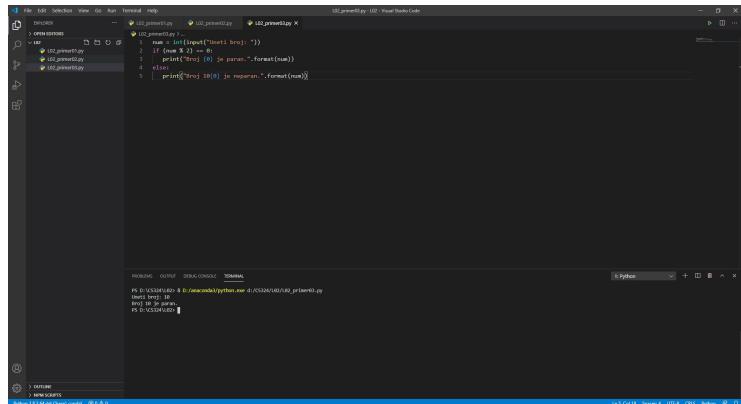
tempF = (tempC * 1.8) + 32
print('%0.1f stepeni Celzijus je %0.1f stepeni Farenhjita.' %(tempC,tempF))
```

Primer #3: Proveriti da li je broj paran ili neparan (10 minuta)

Uneti ceo broj.

Program vraća odgovor da li je broj paran ili neparan.

```
num = int(input("Uneti broj: "))
if (num % 2) == 0:
    print("{0} je paran".format(num))
else:
    print("{0} je neparan".format(num))
```



Slika 6.10 Primer #3 u Visual Studio Code-u. [Izvor: Autor]

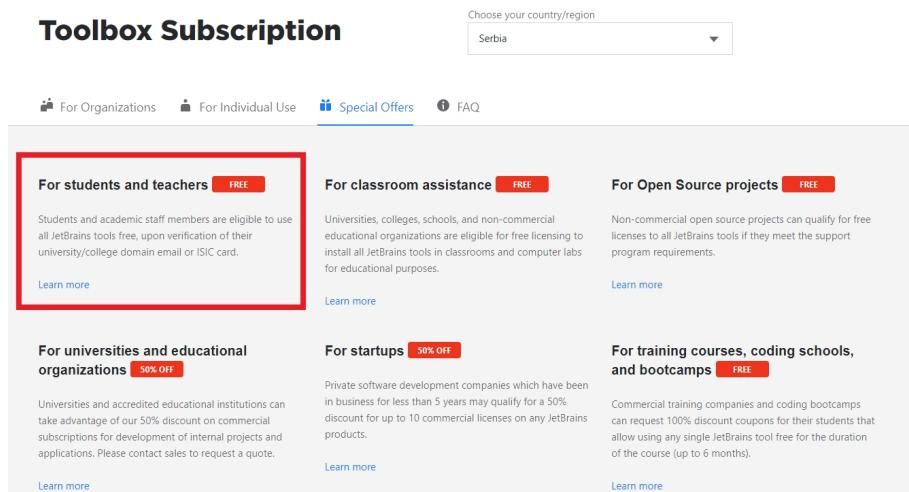


DOBIJANJE JETBRAINS LICENCE I INSTALACIJA PYCHARM OKRUŽENJA

Da bi se preuzeila puna verzija PyCharm okruženja, najpre treba obezbediti licencu.

Da bi se preuzeila puna verzija PyCharm okruženja, najpre treba obezbediti licencu.

Na sajtu <https://www.jetbrains.com/pycharm/buy/#discounts?billing=yearly> nalazi se odeljak za studente i nastavnike.



Slika 6.12 JetBrain licenca za studente i nastavnike. [Izvor: [jetbrains.com](https://www.jetbrains.com/student/)]

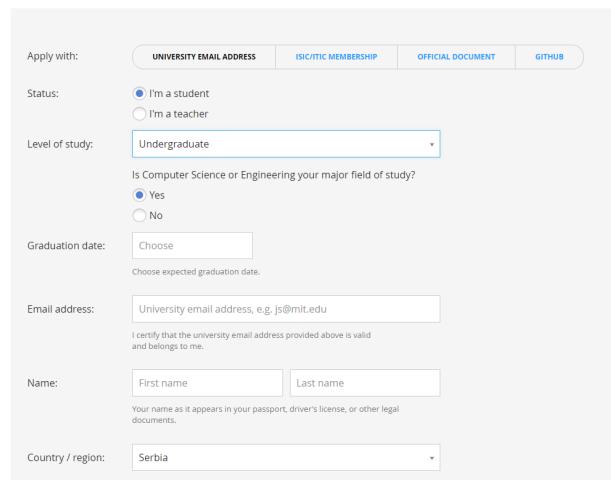
Izabratи tu opciju za studente i nastavnike, i popunitи formular korišćenjem fakultetske email adrese . Nakon toga, moguće je prijaviti se na sajt prema daljem uputstvu.

Podsetnik:

Za prijavu za godišnju licencu koristiti fakultetsku email adresu.

JetBrains Products for Learning

Before you apply, please read the [Educational Subscription Terms and FAQ](#).



The form is a web-based application for applying for a free JetBrains license. It starts with a 'Status' section where 'I'm a student' is selected. Below that is a 'Level of study' dropdown set to 'Undergraduate'. The next section is 'Is Computer Science or Engineering your major field of study?' with 'Yes' selected. The 'Graduation date' field is a 'Choose' button. The 'Email address' field is 'University email address, e.g. js@mit.edu'. Below it is a checkbox for 'I certify that the university email address provided above is valid and belongs to me.' The 'Name' section has 'First name' and 'Last name' fields. Below them is a note: 'Your name as it appears in your passport, driver's license, or other legal documents.' The 'Country / region' dropdown is set to 'Serbia'. At the top, there are tabs for 'UNIVERSITY EMAIL ADDRESS', 'ISIC/ITIC MEMBERSHIP', 'OFFICIAL DOCUMENT', and 'GITHUB', with 'UNIVERSITY EMAIL ADDRESS' being the active tab.

Slika 6.13 Prijava za dobijanje besplatne licence. [Izvor: jetbrains.com]

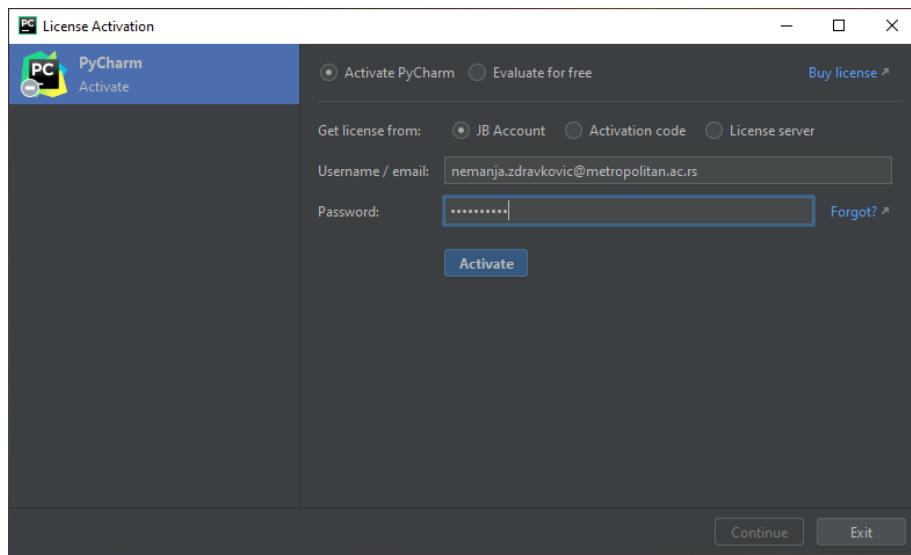
INSTALIRANJE PYCHARM IDE

Instalacija PyCharm IDE je jednostavna i ne bi trebalo da izazove poteškoće.

Nakon prijave i verifikacije email adrese, preuzeti PyCharm instalaciju i pratiti uputstva.

Podsetnik:

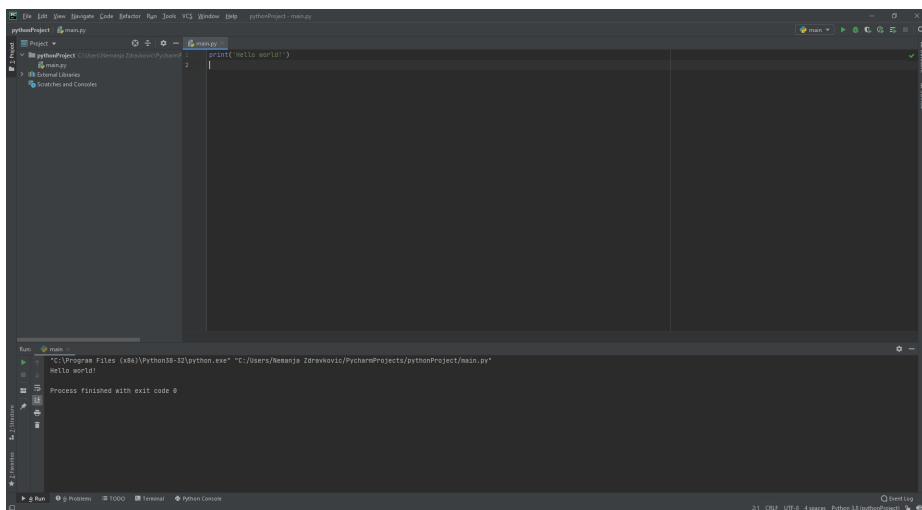
Ukoliko je potrebno, i uPyCharm IDE podesiti Python interpreter.



Slika 6.14 Aktivacija PyCharm IDE-a. [Izvor: Autor]

Nakon uspešne aktivacije, možete i u PyCharm-u pisati projekte u Pythonu.

PyCharm nudi veće mogućnosti nego Visual Studio Code, posebno integraciju sa drugim servisima o kojima će biti reči u daljem toku predmeta.



Slika 6.15 Prikaz PyCharm IDE interfejsa. [Izvor: Autor]

KORIŠĆENJE PYCHARM IDE - PRIMER #4

PyCharm - Primer #4

Primer #4: Provera prostog broja (15 minuta)

Napisati program u PyCharm-u koji proverava da li je broj prost.

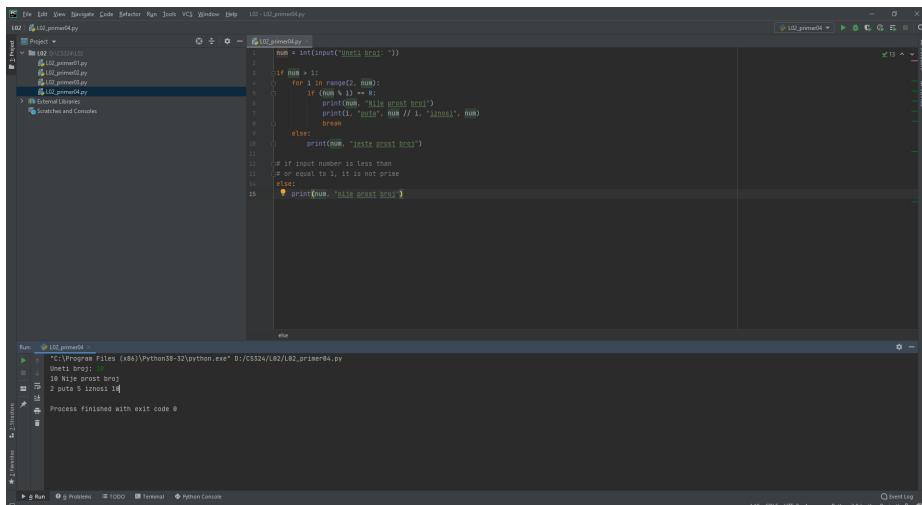
Ukoliko broj nije prost, napisati njegove sadržaoce.

```
num = int(input("Uneti broj: "))

if num > 1:
    for i in range(2, num):
        if (num % i) == 0:
            print(num, "Nije prost broj")
            print(i, "puta", num // i, "iznosi", num)
            break
    else:
        print(num, "jeste prost broj")

# if input number is less than
# or equal to 1, it is not prime
else:
    print(num, "nije prost broj")
```

Prepoznati delove koda koji se odnose na petlje i kontrolu toka. U čemu je razlika u odnosu na C familiju programskih jezika?



Slika 6.16 Primer #4 u PyCharm-u. [Izvor: Autor]

▼ Poglavlje 7

Individualna vežba #2

UVOD U INDIVIDUALNU VEŽBU #2

Nakon dobijanja licence od JetBrains-a, PyCharm se može slobodno koristiti u edukativne svrhe

U individualnoj vežbi #2 proći će se kroz proces dobijanja studenteske licence za PyCharm IDE, samostalnu instalaciju PyCharm IDE okruženja, kao i pisanje jednostavnih programa u Python-u koristeći ovo okruženje.

Procenjeno trajanje individualnih vežbi iznosi 90 minuta.

INDIVIDUALNA VEŽBA #2

Napisati jednostavne programe u Python programskoj jeziku koristeći instalirana okruženja.

Individualna vežba #2 prolazi kroz rad u editorima koda i IDE okruženjima.

Studenti treba samostalno da znaju kada mogu brzo iskoristiti editora koda za brz pregled i ispravke programa, a kada razvijati softver u IDE okruženju.

Na osnovu individualnih zadataka studenti treba da zaključe kada koji alat koristiti.

Uvod u individualnu vežbu #2

Instalirati sledeće editore koda:

- Notepad++
- Visual Studio Code

Nabaviti studentsku licencu JetBrains za PyCharm i instalirati ga.

<https://www.jetbrains.com/pycharm/>

Povezati Python interpretare gde je neophodno.

Zadatak #1 (30 minuta)

Napisati program u Python-u za zamenu dve vrednosti promenljiva tako da:

- Postoji treća promenljiva,
- Ne postoji treća promenljiva.

Program napisati u editoru koda i sačuvati kao L02_zad01.py. Nakon toga, iz konzole pokrenuti program.

Zadatak #2 (30 minuta)

Napisati program koji rešava kvadratnu jednačinu. Koristiti Visual Studio Code ili PyCharm.

$$ax^2 + bx + c = 0$$
$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Zadatak #3 (30 minuta)

Napisati program za pronalaženje faktorijela broja. Faktorijel nule je jedan.

$$n! = n \times (n - 1) \times \cdots \times 2 \times 1$$

✓ Poglavlje 8

Domaći zadatak #2

DOMAĆI ZADATAK

Domaći zadatak #2 se okvirno radi 2h

Domaći zadatak #2

Na kućnom računaru instalirati Notepad++, Visual Studio Code sa Python ekstenzijom, kao i PyCharm IDE okruženje.

Povezati Python interpretare gde je neophodno.

Screenshot-ovima pokazati da su interpretari povezani i da možete pokretati pisane programe.

Napisati program koji će ispisivati sve proste brojeve u proizvoljnom intervalu.

Koristiti uputstvo za izradu domaćeg zadatka dato u prethodnoj lekciji. Uz dokument dostaviti i .py izvorni kod programa.

Predaja domaćeg zadatka:

Tradicionalni studenti:

Domaći zadatak treba dostaviti najkasnije nedelju dana nakon predavanja za 100% poena. Nakon toga poeni se umanjuju za 50%.

Online studenti:

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganja ispita. **Domaći zadaci se brane!**

Svi studenti domaći zadatak poslati dr Nemanji Zdravkoviću:
nemanja.zdravkovic@metropolitan.ac.rs

▼ Poglavlje 9

Zaključak

ZAKLJUČAK

Zaključak lekcije #2

Rezime:

U ovoj lekciji objašnjena je najpre istorija Python jezika, kao i dalji razvoj koji i danas traje.

Zatim objašnjene su razlike dva izdanja Python jezika koja se i dalje koriste, Python 2.x i Python 3.x, kao i razlozi zbog koje treba preći na 3.x.

Opisane su primene Python programskoj jeziku, u kojima se vidi da se Python, iako skripting jezik, može koristiti za opštu upotrebu bez velikih poteškoća.

Konačno, bilo je reči o editorima koda i IDE razvojnim okruženjima za Python.

U vežbi opisan je proces instalacije editora koda i IDE okruženja uz odgovarajuće primere.

Literatura:

1. David Beazley, Brian Jones, *Python Cookbook: Recipes for Mastering Python 3*, 3rd edition, O'Reilly Press, 2013.
2. Mark Lutz, *Learning Python*, 5th Edition, O'Reilly Press, 2013.
3. Andrew Bird, Lau Cher Han, et. al, *The Python Workshop*, Packt Publishing, 2019.
4. Al Sweigart, *Automate the boring stuff with Python*, 2nd Edition, No Starch Press, 2020.

