Programski paket Astropy

Pranjić, Ivan

Master's thesis / Diplomski rad

2016

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:809170

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**



Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb







SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

SMJER: FIZIKA I INFORMATIKA, NASTAVNIČKI

Ivan Pranjić

Diplomski rad

Programski paket Astropy

Voditelj diplomskog rada: izv. prof. dr. sc. Goranka Bilalbegović

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____
2. _____
3. _____
4. _____

Datum polaganja: _____

Zagreb, 2016.

Sažetak

U astronomiji su potrebni alati s kojima se može upravljati mnoštvom podataka koji se prikupljaju o raznim objektima u svemiru. Zbog toga je nastao projekt Astropy. To je paket rutina koje su napisane u programskom jeziku Python. Primarna namjena paketa Astropy je uporaba u astrofizici, ali se može koristiti i u drugim granama fizike. Radi se o projektu otvorenog koda koji se pojavio 2013. godine. Od tada se znatno razvio, ali na njemu se i dalje radi. Astropy je primjer uspješnog znanstveno-računalnog projekta svjetskih razmjera.

Ovaj rad čitatelja provodi kroz kratki uvod u programski jezik Python i njegove značajnije pakete i sučelja, kao što su NumPy, SciPy, matplotlib i Jupyter. Opisan je Github koji, kao kolaboracijska platforma, čini programske pakete u Pythonu lako dostupnim svima koji žele dati doprinos njihovom razvoju. Čitatelja se nakon toga uvodi u projekt Astropy i njegove mogućnosti. Opisani su se neki osnovni potpaketi, kao što su podrška za fizičke jedinice i njihovu konverziju, konstante, koordinatni sustavi, kozmološki modeli, tablice, specijalni formati datoteka, itd. Opisan je i paket Astropy projekta poznat kao Astroquery. Kroz primjere pretraživanja baza primjenom paketa Astroquery dobiveni su podaci za nekoliko izabranih astrofizičkih objekata. Na kraju se, koristeći potpaket za koordinate, navodi primjer jednog zadatka koji bi bio primjeren višim razredima prirodoslovno-matematičke gimnazije.

Software package Astropy

Abstract

In astronomy there is a need for tools that could manage large amount of data about objects in the universe. That was the main reason for the creation of the project Astropy. Astropy is a package written in the Python programming language. Its primary purpose is the field of astrophysics, but it can be used in other parts of physics as well. The project Astropy is an open source project that started in 2013. It has grown into a powerful tool, but it is still being developed. It is also a good example of worldwide scientific collaboration.

In this thesis, a short introduction to the programming lanuage Python is presented. The most important packages, such as NumPy, SciPy, matplotlib and Jupyter are described. Software packages in Python are easy accessible and everyone can contribute to them. This is achieved by using a collaboration platform Github. Then the project Astropy is introduced. Basics are covered, such as: units and units conversion support, constants, coordinate systems, cosmological models, tables, special file formats, etc. Astroquery, the subpackage of Astropy, is described. Several examples of using Astroquery to search for specific objects in astrophysical bases are also presented. In the end, using the coordinates subpackage, an example is given that could be used in teaching of a high school senior year class.

Sadržaj

1. Uvod ... 2
2. Programski jezik Python ... 3
   2.1 NumPy ... 5
   2.2 SciPy ... 7
   2.3 IPython ... 10
   2.4 Jupyter ... 12
      2.4.1 Julia ... 14
      2.4.2 R ... 15
   2.5 matplotlib ... 17
   2.6 GitHub ... 19
3. Astropy ... 22
   3.1 Mjerne jedinice (astropy.units) ... 24
   3.2 Vrijeme (astropy.time) ... 26
   3.3 Nebeske koordinate (astropy.coordinates) ... 27
   3.4 Tablice i podaci u rešetci (astropy.table i astropy.nddata) ... 30
   3.5 Formati datoteka (astropy.io) ... 32
   3.6 World Coordinate System (astropy.wcs) ... 34
   3.7 Kozmologija (astropy.cosmology) ... 35
4. Astroquery: primjena paketa Astropy za pristup podacima u bazama ... 36
   4.1 SIMBAD ... 36
   4.2 VizieR i Open Exoplanet ... 44
   4.3 SIMBAD i VizieR na primjeru jednog objekta ... 45
5. Astropy u nastavi ... 49
   5.1 Python i cjeloživotno učenje ... 49
   5.2 Primjena Astropy programskog paketa u nastavi ... 50
      5.2.1 Primjer korištenja Astropy paketa – astropy.coordinates ... 51
6. Zaključak ... 54
7. Literatura ... 55

1. Uvod

U znanstvenoj zajednici ključ napretka je oduvijek bila međusobna suradnja znanstvenika. Ne isključujući pojedinačni doprinos svakog znanstvenika ponaosob, najbrži napredak u znanosti se odvijao kada su znanstvenici međusobno izmjenjivali mišljenja. Pored prenosa informacija na znanstvenim skupovima, u 21. stoljeću suradnja je moguća i putem raznih kolaboracijskih platformi na internetu. Suvremeni znanstvenici ne mogu biti samo izvrsni u području struke, već moraju imati i barem osnovnu digitalnu pismenost u smislu aktivnog služenja raznim računalnim rješenjima. Na taj način znanstvenici sami mogu modelirati postojeće rješenje prema vlastitim potrebama, nadograditi postojeći, ili napisati neki novi program. To je jedan od glavnih razloga zašto mnogi znanstvenici danas, neovisno o struci, poznaju i neki programski jezik. Danas se u velikoj se mjeri koristi programski jezik Python koji ima kvalitete koje privlače programere početnike i zadržava one kojima je potreban moćan programski jezik. Međusobna suradnja znanstvenika i zajednička izrada i nadogradnja programskih rješenja štedi dragocjeno vrijeme. Zbog toga se širi mreža sudionika u raznim znanstvenim računalnim projektima diljem svijeta.

2. Programski jezik Python

Python je programski jezik koji je svakim danom sve popularniji u svijetu računalnih inženjera, znanstvenika, vrhunskih programera, amatera, školaraca i ostalih. Na prvom mjestu ga odlikuje jednostavnost sintakse (vidi Sliku 2.1), a druga najveća odlika mu je snaga, odnosno mogućnost da širok spektar problema riješi brzo, jednostavno i efikasno.

Ideja o programskom jeziku tipa Pythona je nastala 80.-ih godina 20. stoljeća. Prva verzija programa izašla je 1991. Njegov autor i začetnik je Guido van Rossum, nizozemski računalni programer, koji je sebi dodijelio doživotnu ulogu nadglednika procesa razvoja programskog jezika Python [1]. Programski jezik nije dobio ime prema zmiji pitonu nego prema komičarskoj grupi Monty Python. Glavna nit vodilja u stvaranju ovog jezika je bila to da bude modularan, tj. da ga bude lako nadograđivati te da podržava više platformi. Tijekom 90.-ih godina prošlog stoljeća Python se razvijao, a van Rossum je gledao u budućnost kako bi unaprijedio taj programski jezik. Kao posljedica toga nastala je verzija 2.0 koja je svjetlo dana ugledala 2000. godine. Glavne odlike su joj bile to što je ta verzija bila puno više otvorena, pogotovo za zajednicu koja je pomagala razvoj Pythona. To je bilo lakše kad se izvorni kod, odnosno kompletan repozitorij Pythona, prebacio na SourceForge gdje je svatko mogao doprinijeti razvoju. Ta verzija je dogurala do sedme iteracije, odnosno do 2.7, kada van Rossum i suradnici obustavljaju rad na toj verziji te se posvećuju radu na verziji 3 [1]. Glavni razlog je bio taj jer je bilo previše modula koji su radili iste stvari te je dolazilo do zagušenja. Verzija 3 izlazi 2008. godine. Sve što je bilo napisano u Pythonu verzije 2 nije bilo kompatibilno s verzijom 3. To je izazvalo revolt zajednice te se jako puno korisnika i dan danas služi posljednjom verzijom druge generacije Pythona, verzijom 2.7. Službena podrška za tu verziju ističe 2020. Tvorci Pythona se nadaju da će svi prijeći na treću generaciju jezika. Aktualna verzija u trenutku pisanja ovog teksta je verzija 3.5.

```
print("Hello, World!")
Hello, World!
```

Slika 2.1: Hello World program i njegov rezultat u Pythonu

Ono što mnogi navode kao prvu karakteristiku je to da je Python interpreterski jezik. Programski kod se kompilira u bytecode koji se pohranjuje u posebne datoteke (.pyc) koje se mogu izvoditi interpretacijom tog međukoda na bilo kojoj platformi na kojoj je Python instaliran. Modularnost je također bitno svojstvo ovog programskog jezika. Naime, programski kod se zapisuje u module koje je moguće izvršiti nakon što ih napišemo, a to vodi k tome da kombinacijom više modula možemo lakše „složiti“ cijeli program. Najjednostavniji primjer pozivanja programa u Pythonu možemo vidjeti na Slici 2.2 gdje gornji dio slike prikazuje program, dok je donji dio slike ispis rezultata istog programa u terminalu.

```
ULAZ:
program mojprogram.py
print (2 ** 8) # ispisuje koliko je 2^8
tekst ='ovaj program ispisuje koliko je dva na osmu'

IZLAZ:

256
ovaj program ispisuje koliko je dva na osmu
```

Slika 2.2: Prikaz jednostavnog koda u Pythonu

Python je programski jezik visoke razine. Ima odlike modernih programskih jezika kao što su objektno orijentirano programiranje s višestrukim nasljeđivanjem, upravljanje i korištenje modulima i paketima, korištenje i obrada iznimki i izuzetaka te proširivost [2]. Jedan od razloga zašto je Python dobar za tako širok spektar ljudi je i mnoštvo biblioteka koji dolaze prilikom instalacije gdje je uključeno oko 200 modula. Novi moduli se mogu

dodatno instalirati ovisno o potrebi korisnika. Velika prednost Pythona je i to što ima široku zajednicu korisnika i razvojnih stručnjaka, potpuno je open-source tipa i besplatan je. Kada uz to još nadodamo i po mnogima najjednostavniju sintaksu za kodiranje, jasno je zašto je Python mnogima prvi i najbolji izbor za njihove projekte.

Kod u Pythonu možemo izvoditi :

-interaktivno: interpreter se pokreće u konzoli neovisno o operacijskom sustavu na kojem se nalazi, pojavljuje se prompt >>> nakon kojeg korisnik utipkava željenu naredbu. Obilježje ovakvog načina izvođenja naredbe je to što se ona izvršava odmah iza unosa, ispisuje se njen rezultat i očekuje slijedeća naredba. Ovaj način je zgodan za analizu koda dio po dio gdje možemo vidjeti što se točno događa u pojedinom dijelu modula programa.

-skriptno: programski kod s naredbama možemo pohraniti u datoteku oblika .py koju možemo pozvati iz sistemske linije. Tada se počnu izvoditi naredbe, a na ekranu se ispisuju rezultati, kao što je prikazano na Slici 2.2.

-umetnuto: pozivajući Python runtime API (aplikacijsko programersko sučelje) moguće je izvršavati kod unutar programa napisanog u nekom drugom programskom jeziku.

2.1 NumPy

NumPy je biblioteka koja se može koristiti u Pythonu. Kao što se iz samog imena da naslutiti, radi se o modulu koji pospješuje rad s numeričkim vrijednostima. NumPy sadrži puno napisanih funkcija za razne numeričke operacije. Najveći doprinos proizlazi iz podrške za rad s višedimenzionalnim poljima i velikim matricama. NumPy sadrži veliki broj kompleksnijih matematičkih funkcija za rad s matricama i poljima.

NumPy je nastao iz dva matematička modula koja su bila namijenjena za rad s poljima: Numeric i Numarray. NumPy je spoj najboljeg iz ta dva navedena modula, odnosno pisan je na kodu Numerica sa svim značajkama Numarraya [3]. Ono što čini NumPy efikasnim je lako korištenje broadcast funkcija (objašnjene niže u tekstu), korištenje za linearnu algebru, Fourierove transformate, rad s nasumičnim brojevima te niz alata za laku

integraciju s kodom pisanim u C, C++ i Fortranu [3]. Kako se Python koristi za niz primjena, ista stvar se događa i s NumPy paketom. NumPy je jako dobar za rad s više-dimenzionalnim poljima, a znamo da se unutar Pythona mogu definirati proizvoljni tipovi podataka. To dovodi do toga da se NumPy koristi kao jako učinkoviti višedimenzionalni kontejner generičkih podataka što u konačnici dozvoljava da se NumPy brzo i lako integrira s velikim brojem baza podataka.

U svijetu numeričkog programiranja NumPy je na razini komercijalnog programa MATLAB-a i ne samo to, već ga nadmašuje u nekim osobinama. Kako bi proširili bazu korisnika ovog besplatnog i open-source dodatka za isto tako besplatni Python, na službenim stranicama postoje i upute o tome kako korisnik MATLAB-a može prijeći na NumPy[1]. Kako rad napravljen u MATLAB-u ne bi bio uzaludan i kako ne bi program morali iznova tipkati u Pythonu, NumPy može učitati MATLAB tipove datoteka i spremati u taj isti tip, što smanjuje broj onih koji se opiru prelasku na rad u Pythonu i uporabu biblioteke NumPy.

Ako postoji nešto po čemu je NumPy poznat onda je to svakako rad s poljima. To ni ne čudi iz razloga što je brzina izvođenja naredbi velika, a rad s poljima lagan. No, ono po čemu je NumPy paket najviše poznat, to je potpaket ndarray koji je svojevrstan pojam za rad s višedimenzionalnim poljima [3]. Kako se stvara polje možemo vidjeti na Slici 2.3 gdje je prikazano i kako se dobivaju neke od osnovnih informacija o tom polju.

[1] Upute se nalaze na: https://docs.scipy.org/doc/numpy-dev/user/numpy-for-matlab-users.html

```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.dtype.name
'int64'
>>> a.itemsize
8
>>> a.size
15
```

Slika 2.3: Stvaranje polja i dobivanje nekih osnovnih informacija o tom polju

Kako možemo vidjeti na prethodnoj slici, prvom naredbom uvozimo NumPy i to pod imenom np, čisto iz praktičnih razloga kako bi pisali manje koda. Zatim dajemo naredbu gdje želimo 15 članova u polju koje će imati dimenzije 3 x 5 što odgovara dvodimenzionalnoj matrici. Pozivanjem atributa shape dobivamo dimenzije matrice, pozivanjem ndim dobivamo rang matrice, pozivanjem dtype.name dobivamo informaciju koje su vrste vrijednosti u matrici, pozivanjem itemsize dobivamo informaciju koliko zauzima pojedini element bajtova u memoriji, a pozivanjem size dobivamo broj elemenata matrice, odnosno dvodimenzionalnog polja.

2.2 SciPy

SciPy je još moćnije proširenje Pythona koje sadrži NumPy, ali i mnogo drugih matematičkih metoda [3-5]. Paket pod imenom Numeric je nastao još 1995. godine suradnjom više autora (Jim Fulton, David Ascher, Paul DuBois, Konrad Hinsen). Za potrebe svojih znanstvenih istraživanja neki korisnici paketa Numeric su pisali dodatne module. Travis Oliphant, Eric Jones i Pearu Peterson su spojili svoje module te je tako

nastao SciPy. Usporedno s time razvoj Numerica stagnira, ponajviše iz dva razloga. Prvi razlog je da su korisnici tog koda, mahom znanstvenici, uvidjeli nedostatke Numerica te su isti htjeli nadograditi no način na koji je Numeric kodiran to nije dozvoljavao. Drugi razlog je to što izvorni autor Pythona Guido van Rossum nije htio uključiti Numeric u standardnu distribuciju Pythona zbog toga što izvorni kod za Numeric nije bilo moguće održavati na odgovarajući način.

Kao reakcija na to nastao je numarray koji je trebao zamjeniti Numeric. Autori numarraya su znanstvenici sa SSTI (Space Science Telescope Institute): Todd Miller, Rick White i Perry Greenfield. Praktična primjena novog paketa je bila obrada astronomskih slika. Ono što možda zvuči paradoksalno je jedan od većih problema koji je numarray imao: bio je izuzetno brz za obradu velikih polja, ali jako spor prilikom obrade malih polja. Pored toga ideja o SciPy paketu koji bi objedinio sve potrebno za znanstvenu obradu podataka u Pythonu je padala u vodu zbog toga što API nije bio s njim kompatibilan. To se očitovalo u tome što nije bilo moguće prebaciti izvorni kod SciPy-a sa korištenja Numerica na numarray, a rezultat je bio da su se korisnici podijelili na pola: jedni su pisali kod za numarray vidjevši ga kao pravog nasljednika Numerica, dok su drugi ostali vjerni pisanju koda za Numeric jer se nisu mogli odreći svih blagodati tadašnjeg paketa SciPy.

Na scenu se tada vraća jedan od glavnih autora SciPy-a Travis Oliphant zbog toga što je podijeljena zajednica bila suprotno od onoga što je želio. Oliphant je u potpunosti refaktorirao kod Numerica kako bi se isti mogao lakše održavati i nadograđivati te na kraju i implementirati sve dobre osobine numarray paketa. Tom svom pothvatu je nadjenuo ime SciPy core kao polazni modul za stvaranje još većeg i sveobuhvatnijeg znanstvenog paketa SciPy. Taj čin je naišao na pohvale cijele zajednice, no u početku se javljao problem što krajnji korisnici očito baš i nisu shvatili što je tim potezom Oliphant napravio. Naime, većina korisnika kojima su bile potrebne elementarne funkcije Numerica instalirali su cijeli SciPy paket misleći kako Oliphant na neki način neraskidivo vezan uz Numeric. Tada se povela javna rasprava o imenovanju novog Numerica kako bi ga jasno znali razlikovati te se na taj način rodio NumPy, koji kao što vidimo ima zajedničku povijest sa paketom SciPy [3-5].

SciPy paket se najčešće koristi za [5]:

- Optimizaciju: služi optimizaciji postojećeg koda
- Naprednu analizu podataka: sa svojim dodatkom RPy, analizira podatke dobivene najpoznatijim programskim jezikom za tu vrstu posla imenom R
- Bazu podataka: hijerarhijski usmjeren paket za bazu podataka PyTables koji upravlja velikim količinama podataka prema HDF5 formatu
- U interaktivnoj okolini: uz IPython kao dodatak, stvara se interaktivna okolina koja gotovo sve značajke identične kao MATLAB
- Posebni dodaci javljaju se u obliku scikit-ova koji služe kako bi se u NumPy ili Python koristili posebni paketi poput scikit-image za obradu slike ili scikit-learn za machine learning [2]

Mnoge funkcije pisane za SciPy su naslijedjene iz drugih programskih jezika koji su se pokazali učinkovitim na tom polju, a najčešće se radi o funkcijama i algoritmima koji su prvotno napisani u programskom jeziku Fortran. Najčešće korišteni i najpoznatiji moduli, odnosno potpaketi u SciPy-u su [5]:

- constants: sadrži veliki broj matematičkih i fizikalnih konstanti
- special: sadrži specijalne funkcije za matematičke metode fizike poput Kelvinove funkcije, Beta funkcije, Gama funkcije, Besselovih funkcija, Kugline funkcije, itd.
- integrate: funkcije potrebne za integriranje koristeći trapezoidnu metodu, Simpsonovu metodu, Rombergovu metodu
- io: unos i ispis podataka
- lib: koristi se za dohvat vanjskih biblioteka
- misc: razni alati, primjerice pregled ili stvaranje neke slike
- ndimage: razne funkcije za obradu višedimenzionalnih slika
- optimize: algoritmi za optimizaciju korisničkih funkcija
- linalg: velik broj metoda i funkcija za rješavanje problema u linearnoj algebri poput dijagonalnih matrica (računanje normi, inverza, vlastitih vrijednosti i vlastitih vektora te razne druge operacije s matricama)
- sparse: rutine za rad s velikim raspršenim matricama

[2] Machine learning - područje računalne znanosti koje se bavi razvojem algoritama kojima računalo može učiti iz podataka i dati određena predviđanja na temelju stečenih informacija

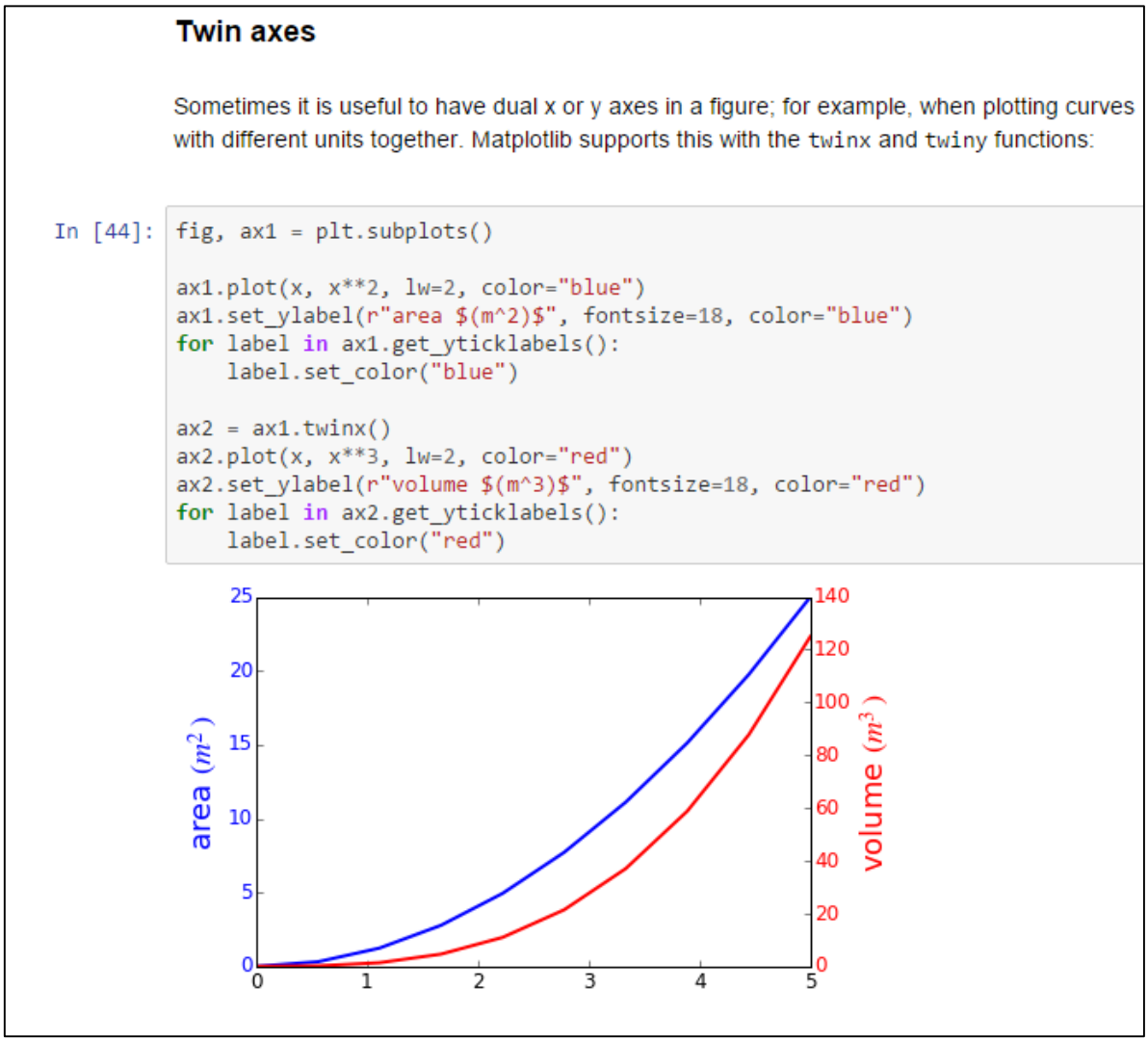
- interpolate: rutine i klase za interpolaciju objekata koji se mogu iskazati diskretnim brojevima, dostupno za jednodimenzionalne i dvodimenzionalne skupove podataka
- fftpack: obrada i rutine Fourierovih transformata
- signal: rutine za obradu signala kao što su filtriranje, korelacija, konačni Fourierovi transformati, itd.
- stats: biblioteka raznih statističkih raspodjela i funkcija koje se mogu koristiti na skupovima podataka

2.3 IPython

IPython (Interactive Python) je interaktivna ekstenzija Pythona koja se najčešće koristi kao programčić (widget) u nekom od preglednika u obliku bilježnice [6,7]. Njegov autor je Fernando Perez koji je htio poboljšati rad znanstvenika prilikom obrade podataka u istraživanju. IPython je računalno sučelje temeljeno na Pythonu koje u sebi prikazuje sve elemente koje smatramo općeprihvaćenim prilikom rada s modernim preglednicima. Takav interaktivni dokument se može sastojati od izračuna, tekstualnih pojašnjenja zajedno sa svim formulama (koje mogu biti napisane u LaTeXu), raznih skica, grafikona, slika, video zapisa i slično. To se sve sprema u JSON dokument koji se kasnije, ukoliko postoji potreba za time, može prebaciti u LaTeX, HTML ili u skriptu za Python. Mnogi IPython uspoređuju sa SageMath paketom koji se paralelno razvijao uz IPython. Oni imaju puno sličnosti, a jedna od najupečatljivijih je to da oba sučelja imaju mogućnost rada u bilježnici u pregledniku [7]. Primjer bilježnice u IPythonu je prikazan na Slici 2.4. Osim za navedeno, IPython se koristi i kao alat za paralelno izvršavanje računalnih operacija, bilo da se radi o paralelizmu podataka, paralelizmu zadataka, SPMD (Single Program Multiple Data) paralelizmu, MIMD (Multiple Instruction Multiple Data) paralelizmu ili nekoj od kombinacija. U prijevodu to znači da IPython omogućuje da se paralelizirane aplikacije razvijaju, debugiraju pokreću te interaktivno motre [6].

Ovo sučelje je posebno i po tome što se može koristiti i u drugim programskim jezicima

poput F#, Ruby, Go, Scala, itd. što znači da se u IPythonu može sastaviti jedan projekt koji sadrži kod pisan u više jezika. Snaga i brzina IPythona se očituje i u situaciji gdje je u IBM-u vršena promjena izvornog koda jednog njihovog machine-learning projekta od 8000 linija koda pisanih u JavaScriptu, HTML5 i Javi. U prvoj verziji je trebalo dvije minute po upitu, a to je zamijenjeno sa svega dvjesto linija koda u Pythonu i dvije sekunde po upitu. Upravo taj primjer i odgovara na pitanje zašto je IPython danas zanimljiv i stručnjacima koji se bave Big Data projektima, a ne samo znanstvenicima.

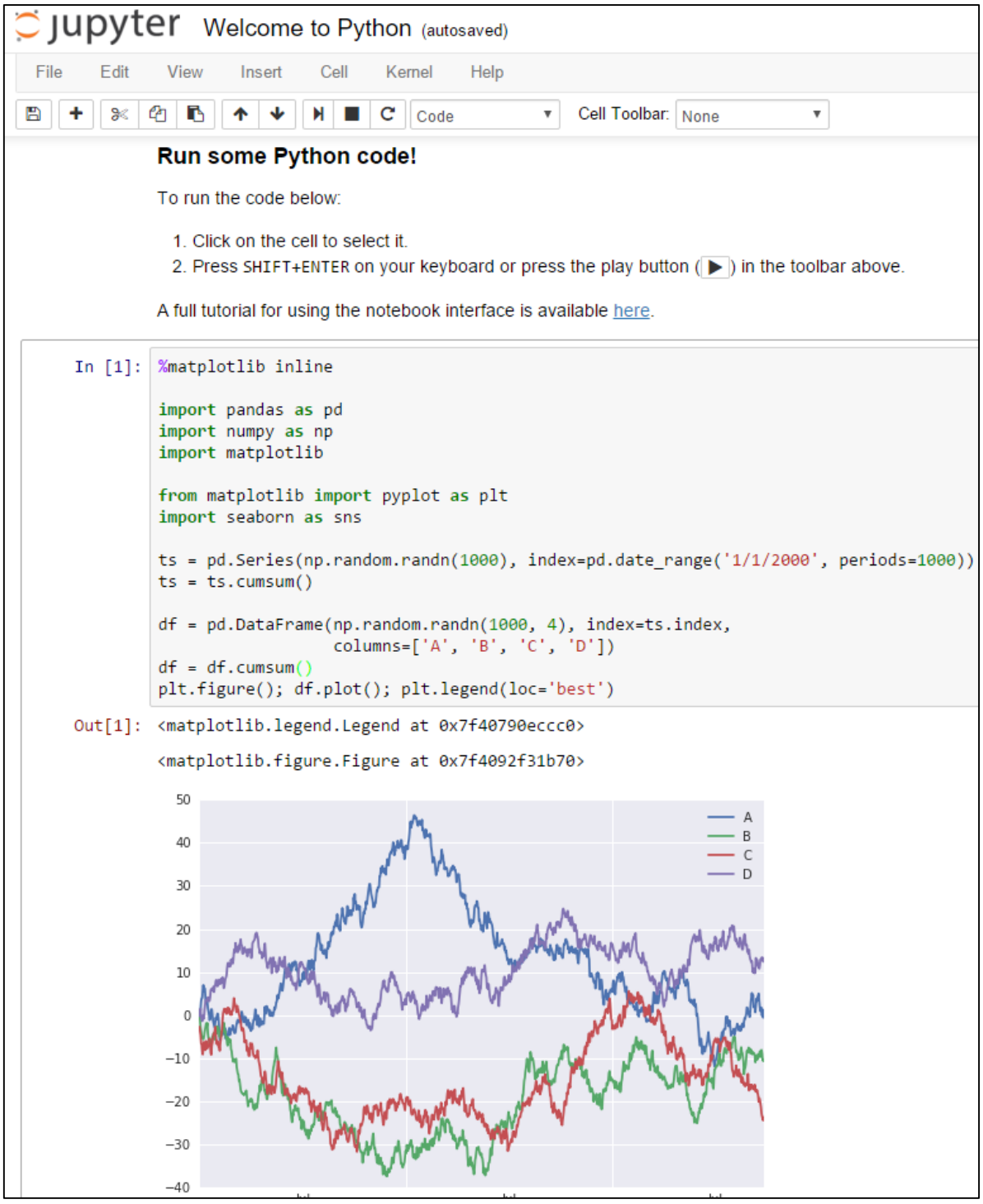


Slika 2.4: Primjer IPython web bilježnice. Preuzeto iz priručnika za matplotlib autora J.R. Johanssona: https://github.com/jrjohansson/scientific-python-lectures/blob/master/Lecture-4-Matplotlib.ipynb

2.4 Jupyter

Projekt Jupyter je međunarodni projekt koji je izrastao iz IPythona. Ime Jupyter dolazi od tri glavna jezika za koje je pripremljen: Julia, Python i R. Jupyter koriste znanstvenici i IT stručnjaci za rad u više od 40 programskih jezika. Glavni sponzori su velike korporacije poput Microsofta i Google-a te neke zaklade. Partneri u projektu su poznata imena poput Sveučilišta Berkeley, Bloomberga i Continuum Analyticsa, a članovi tih sveučilišta i tvrtki se nalaze u savjetodavnom odboru projekta Jupyter. Uz navedeni odbor glavne odluke donosi Fernando Perez, začetnik IPythona te projekta Jupyter. Na službenim internetskim stranicama projekta, nalaze se i neke dinamične bilježnice (notebooks), koje je moguće vidjeti kroz preglednik bilježnica (notebook viewer) [8]. U tom pregledniku moguće je vidjeti prethodno napravljene bilježnice, a smisao tih bilježnica je da korisnici naprave sami interaktivni prikaz nekog problema i njegovo rješenje. Konkretno, na službenim stranicama[3] dostupan je priručnik za IPython, priručnik za programsko rješenje IRuby (Ruby putem komandne linije IPythona), knjiga s primjerima na temu programiranja vjerojatnosti i Bayesianske metode, primjeri za Python u korištenju za pregled vida, knjiga o korištenju Pythona za obradu signala, i mnoge druge. Primjer bilježnice u Jupyteru je prikazan na Slici 2.5.

[3] http://www.jupyter.org



Slika 2.5: Primjer web bilježnice u projektu Jupyter. Preuzeto sa: https://try.jupyter.org

2.4.1 Julia

Julia je relativno nov dinamički programski jezik visoke razine koji od 2012. godine služi za obradu numeričkih i znanstvenih podataka, dok se u puno rjeđe koristi za svakodnevno programiranje ili razvoj web sučelja [9,10]. Primjer koda u programskom jeziku Julia je prikazan na Slici 2.6. Glavne odlike ovog programskog jezika su korištenje vlastitih biblioteka za rješavanje problema linearne algebre, generiranje nasumičnih brojeva, brzih Fourierovih transformacija, mogućnost paralelnog izvođenja više operacija, pozivanje i korištenje postojećih biblioteka napisanih u programskim jezicima C ili Fortran. Julia (kao i Python) ima i sakupljač smeća (GC – garbage collector) što u programerskom svijetu znači da pametno raspolaže resursima, odnosno da se uz zadani naputak rješava nepotrebnih podataka, varijabli i objekata koje više ne služe ničemu u izvođenju programa. Briše ih i na taj način oslobađa dragocjene računalne resurse pozitivno utječući na brzinu i kvalitetu izvođenja programa.

```
ULAZ:
julia> s = "\u2200 x \u2203 y"

IZLAZ:
"∀ x ∃ y"

ULAZ:
julia> for i = 1:endof(s)
         try
           println(s[i])
         catch
           # ignore the index error
         end
       end
∀

x

∃

y
```

Slika 2.6: Primjer koda u programskom jeziku Julia: rad s podacima tipa string i Unicode formatiranim simbolima

Julia također može raditi i sa funkcijama iz programskog jezika Python koje se koriste uz pomoć PyCall paketa, funkcijama iz programskog jezika C, a koristi i Unicode način formatiranja teksta tako da može prihvatiti i posebne matematičke simbole te ih na pravilan način iščitati, primjerice simbol za presjek skupova.

2.4.2. R

R je programski jezik namijenjen za statističko računanje, a najviše ga koriste statističari te programeri koji razvijaju vlastito programsko rješenje za statistiku i analizu podataka [11]. Ovaj programski jezik razvijan je pomoću programskih jezika C i Fortran. Kao i većina ovakvih programskih rješenja nema grafičko sučelje, budući da je fokus na kvaliteti izvedbe pa se interakcija odvija putem komadne linije. Postoje i neke izvedenice s grafičkim sučeljem.

Dovoljno je primjerice upisati 5+3 u komandnu liniju i pritiskom na tipku Enter u slijedećoj liniji dobivamo rezultat 8. Iako je ovaj rezultat skalar, u programskom jeziku R ne postoji struktura podataka u obliku skalara te se isti vodi kao vektor duljine jedan. R podržava i proceduralno programiranje s funkcijama, a djelomice i objektno orijentirano programiranje s nekim generičkim funkcijama. Zanimljiv je podatak da programski jezik R mnogi koriste kao alat za računanje s matricama budući da su mu performanse vrlo slične poznatom komercijalnom rješenju MATLAB. Primjer koda u jeziku R je prikazan na Slici 2.7.

Funkcionalnosti R programskog jezika moguće je pozivati iz drugih programskih jezika u obliku skripte, a neki od njih su Python, Perl, Ruby, F# i Julia. R ima i mogućnost korištenja paketa s posebnim statističkim tehnikama, grafičkih paketa, paketa za razne izvještaje, itd. Ti paketi su najčešće pisani u R programskom jeziku, ali postoje i paketi pisani u programskim jezicima Java, C, C++ te Fortran. Sa instalacijom programskog jezika dolaze neki paketi. Postoji oko 8000 paketa koje su korisnici stvorili za posebne namjene.

```
ULAZ:
a <- 42
A <- a * 2  # R raspoznaje mala i velika slova
print(a)
cat(A, "\n") # "84" je odvojen sa "\n"
if(A>a) # ako je istina tada je ispis: 84 > 42
{
  cat(A, ">", a, "\n")
}

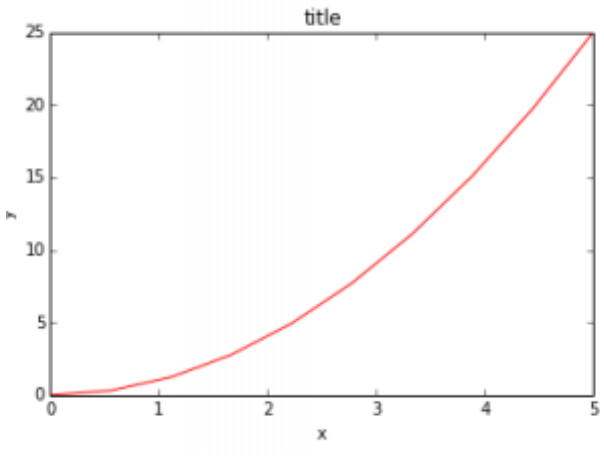
IZLAZ:

[1] 42
84
84 > 42
```

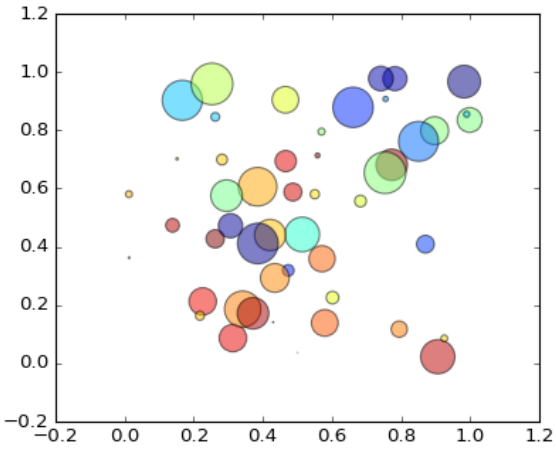
Slika 2.7: Primjer koda u programskom jeziku R

2.5 matplotlib

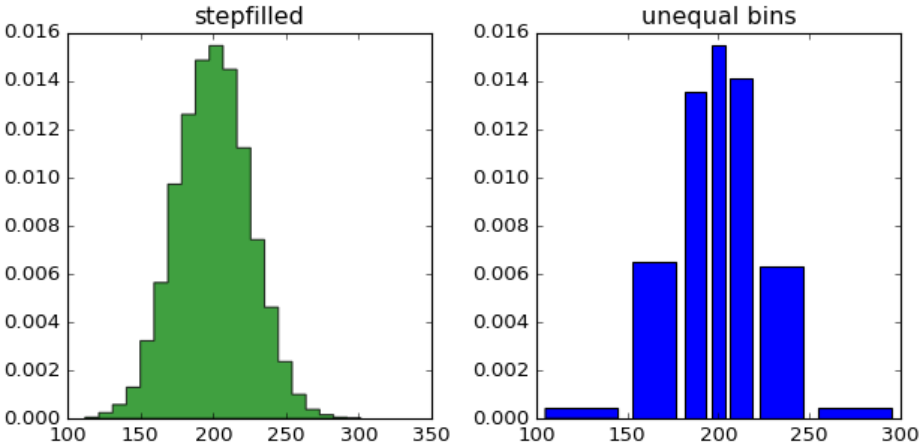
Matplotlib je biblioteka koja služi za crtanje grafikona [12]. Ova biblioteka se služi raznim GUI (Graphical User Interface – grafičko korisničko sučelje) alatima, prvenstveno wxPython, Qt ili GTK+. Značajka koju mnogi znanstvenici cijene je uporaba LaTeX-a odnosno umetanje formula i tekstova formatiranih LaTeX-om. Najveća motivacija za napuštanje komercijalnih rješenja poput MATLABA i MATHEMATICE je to što je matplotlib kao projekt open-source i besplatan. Na slici se može upravljati svakim elementom, ili se mogu naknadno obraditi, a kao izlazni formati su u ponudi svi visoke kvalitete poput SVG, PDF, EPS, PNG i PGF. Puno je znanstvenika naviklo na MATLAB sintaksu, no i to je riješeno u korištenju matplotlib-a budući da je sintaksa gotovo istovjetna kao u MATLAB-u. Otvara se i pitanje korisničke podrške za pomoć pri korištenju i održavanju sustava koji je i jedan od glavnih razloga zašto se neki odlučuju za komercijalna rješenja. Zajednica znanstvenika koji koriste i razvijaju matplotlib je velika i uvijek spremna pomoći [12]. Neki od primjera rezultata dobivenih u matplotlibu su prikazani na slijedećim Slikama 2.8-2.11.



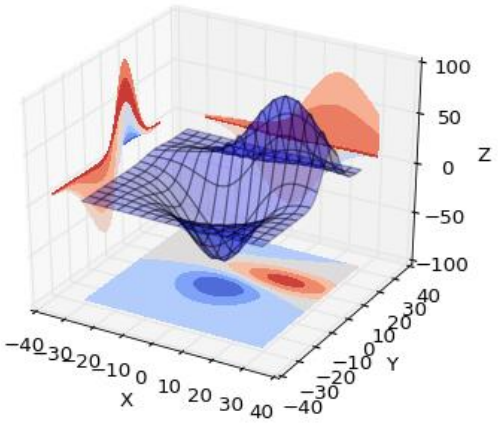
Slika 2.8: Linijski graf. Preuzeto sa: http://matplotlib.org/examples



Slika 2.9: Točkasti graf. Preuzeto sa: http://matplotlib.org/examples



Slika 2.10: Histogram. Preuzeto sa: http://matplotlib.org/examples

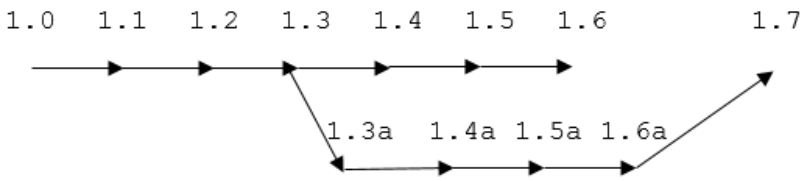


Slika 2.11: 3D graf. Preuzeto sa: http://matplotlib.org/examples

2.6 GitHub

GitHub je jedan od najpoznatijih alata za kolaboraciju pri razvoju programskog koda [13-15]. Ova platforma se razvija od 2008. godine. Koristi se za reviziju i kontrolu koda, a odlikuju ga brzina te podrška za nelinearnu kontrolu toka. Git radi s repozitorijem koji ima povijest mijenjanja, nadodavanja i brisanja koda kako bi se mogao pratiti razvoj software-a. To olakšava posao razvoja kao i brzo i precizno uklanjanje eventualnih grešaka. U današnjim implementacijama najčešće se radi o klijentsko-poslužiteljskom načinu rada. Korisnik (klijent) svaku promjenu koda snimi u repozitorij. U tom istom repozitoriju nalazi se kod u izvornom obliku s vremenskom oznakom kada je određena promjena izvršena u repozitoriju.

Git sustav kontrole koda možemo zamisliti kao drvo s čvorovima (to su verzije programa) na koje se možemo vratiti. Pri razvoju programa često dolazimo do točke kada bi se htjeli vratiti u neki prethodni čvor. Na tom mjestu možemo napraviti novu granu (branch) od koje ćemo krenuti razvijati program s drugim postavkama. Ako nam ta verzija odgovara više od prethodne, onda nju uključujemo u projekt. Isto tako, moguće je i učiniti tzv. fork na projektu što je slično grananju. Naredbom fork se također stvara nova grana, ali ukoliko se glavna grana izbriše projekt koji je nastao fork-om se ne gubi za razliku od onog koji je nastao branch-om. Grafičku prezentaciju jednoga grananja te spajanja te grane možemo vidjeti na Slici 2.12.



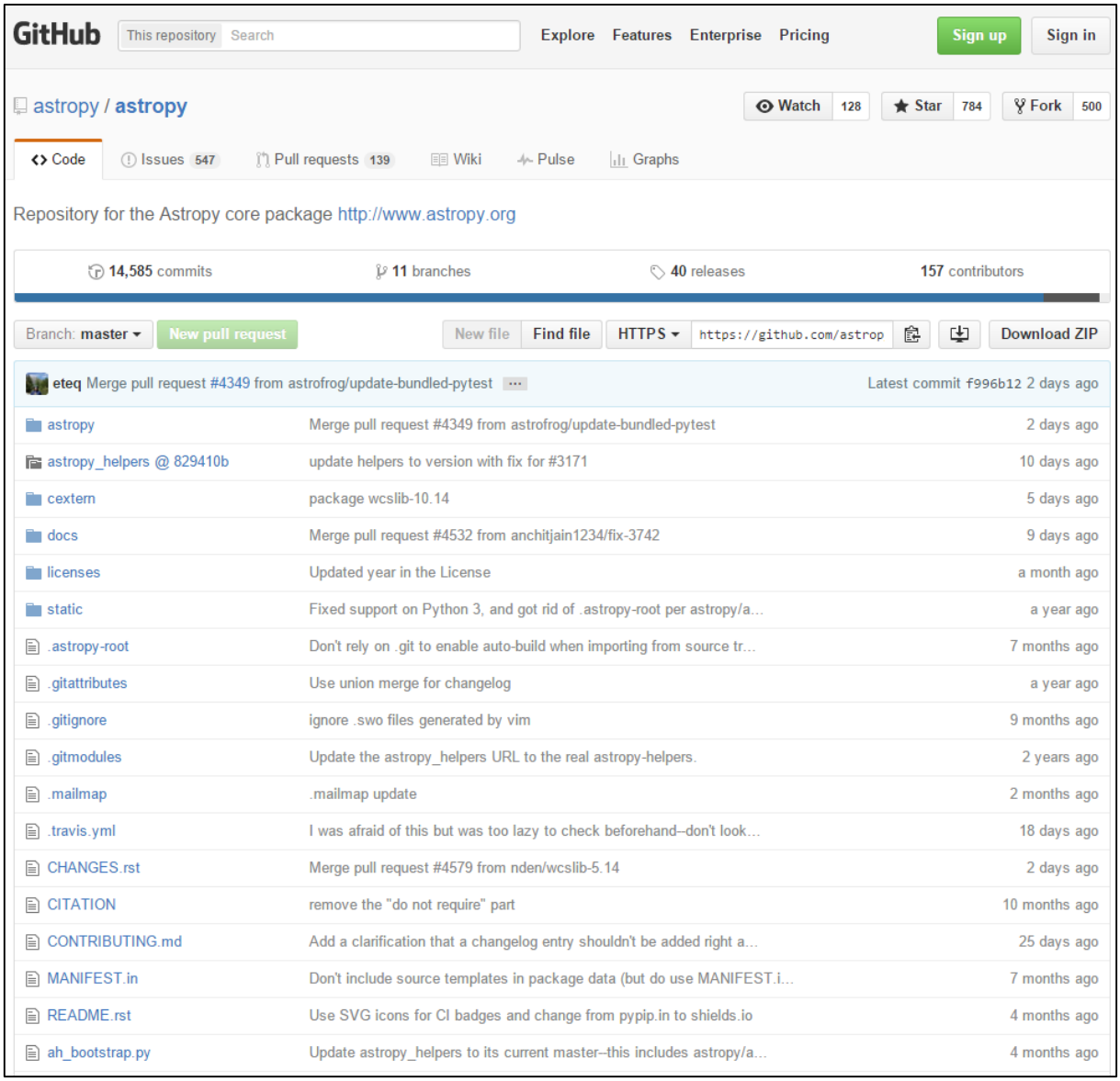
Slika 2.12: Grafički prikaz grananja (branch) te naknadnog spajanja (merge) koda

Git je nastao 2005. kada je Larry McVoy kao vlasnik prava na BitKeeper (do tad najrašireniji alat za upravljanje kodom) isti odlučio komercijalizirati. Linus Torvalds je odlučio napraviti bolji alat za upravljanje kodom. Uveo je nelinearnu kontrolu toka te postavljanje visokih standarda kako bi se cijeli sustav očuvao od potencijalnih slučajnih ili zlonamjernih grešaka. Tako je nastao Git: skalabilan sustav kojeg odlikuje brzina i pouzdanost. Među ostalim vrline su mu i jaka podrška za nelinearan razvoj programa, mogućnosti raščlanjenog razvoja (svatko razvija određen dio programa), kompatibilnost s postojećim sustavima i protokolima, lako upravljanje velikim projektima, kriptirana povijest koda, upravljanje raznim strategijama grananja koda, itd. [13].

Ono što ističe GitHub u usporedbi s ostalim sličnim servisima, poput SourceForge-a i Google Code-a je sigurnost koju pruža korisniku [14]. Iako je GitHub nastao prvenstveno kao repozitorij za kod, danas nudi puno širi spektar usluga kao što su: dokumentacija koda uz automatsko generiranje README datoteka, prijavljivanje i praćenje prijavljenih smetnji i grešaka u kodu uz pripadajući tražilicu, stranice za pomoć s informacijama kako koristiti GitHub, grafikoni koje prate kako, tko i kada je postavio određeni kod u repozitorij, liste sa zadacima, mogućnost držanja manjih web stranica, pregled Photoshop PSD datoteka, itd. U trenutku pisanja GitHub svojim korisnicima pruža besplatnu uslugu korištenja neograničenog broja repozitorija i isto toliko suradnika na istima. Postoji i komercijalni dio GitHub-a gdje se nalaze privatni repozitoriji skriveni od javnosti. Repozitoriji korisnika koji koriste besplatni GitHub su javno dostupni i vidljivi.

GitHub u otvorenim javnim projektima (kao što su NumPy, SciPy, Astropy) služi zajedničkom radu. Svatko tko ima nekakvo rješenje ili dodatak može doprinijeti projektu. Netko napiše kod i pošalje ga u taj javni repozitorij, odnosno napravi comit. Tada taj kod

ide na testiranje i reviziju od ljudi koji su zaduženi za projekt. Ako novi kod doprinosi rješenju, tada se on uključuje u repozitorij kao sastavni dio paketa [15].



Slika 2.13: Prikaz službenog GitHub repozitorija za projekt Astropy. Slika je prikaz stranice https://github.com/astropy/astropy

Važno je napomenuti i to da se Python kao programski jezik otvorenog koda, odnedavno također razvija na platformi GitHub [16]. Na Slici 2.13 je prikazan GitHub projekta Astropy koji je tema ovog rada.

3. Astropy

Programski paket Astropy se razvija od 2011. godine [17-20]. U javnost je izašao u verziji 0.2, a u trenutku pisanja ovog rada Astropy se nalazi u verziji 1.1.2. Osnovna ideja je bila razvoj jednog programskog paketa u Pythonu koji sadrži metode rješavanja problema koji se javljaju u astrofizičkim istraživanjima. Prije pojave paketa Astropy astrofizičari su za te iste zadatke morali instalirati veći broj programa u Pythonu i drugim jezicima.

Od pokretanja Astropy projekta do danas napisano je preko 600 000 linija koda i učinjeno preko 7000 podnesaka koda za unaprjeđenje paketa. O popularnosti unutar astronomske zajednice govori i podatak da web stranica Astropy projekta ima u prosjeku oko 600 jedinstvenih posjeta dnevno. U dane kada izlazi nova verzija, ili neka dugo očekivana nadogradnja, posjet web stranici prelazi 1500 jedinstvenih dnevno. Oko 100 znanstvenika redovno pridonose kodu, a ono što je možda najljepše je to da se većina tih ljudi nikada u životu nije upoznala.

Budući da kodu svatko može doprinositi i na taj način dodatno obogatiti Astropy projekt, potrebno je imati i ljude koji će sav taj kod i održavati. Naime 30-ak programera stalno radi na ovom projektu. Njihov posao je, uz dodavanje prethodno odobrenih novih dodataka i proširenja, održavanje stabilnosti koda. Svoj kod korisnici mogu dodati u repozitorij Astropy projekta na već spomenutom GitHub-u koji je javno dostupan i otvoren svima. Tada voditelji projekta testiraju kod kako isti ne bi narušio rad ostatka Astropy paketa, ili ga značajno usporio. Za testiranje koda programeri koriste py.test i Travis Cl kako bi se utvrdila brzina izvršavanja koda, resursi potrebni za izvršavanje istog te razne greške koje su najčešće rezultat loše programiranog koda. Za dokumentaciju koda koriste servise readthedocs.org i Sphinx-Python documentation generator. Uvjeti da bi kod bio prihvaćen od strane ljudi koji vode Astropy projekt je odsustvo grešaka te navodjenje novih značajki dobivenih tim kodom i objašnjenje potrebe za dodavanjem istog u projekt. Budući da se radi o velikom projektu, često se mogu otkriti greške u kodu. Korisnici putem GitHub platforme mogu prijavljivati greške koje su uočili i čak slati svoja rješenja za otklanjanje tih grešaka.

Kako je sam Python kod dosta čitak i jasan čak i onima koji ga prvi put susreću, istim načelom su se vodili i začetnici Astropy projekta. To dokazuje velik broj korisnika, ali i broj onih koji s lakoćom utvrde grešku u kodu te istu prijave. Ljudi koji vode ovaj projekt

se trude kako bi se svi potrebni astronomski alati uključili u Astropy projekt, ali to nekad nije moguće. Zato je i omogućena podrška te instalacija paketa koji se službeno ne nalaze u Astropy projektu zbog licenci, ili zbog toga jer sam kod tog paketa nije dovršen ili nije po standardu Astropy paketa. Zanimljiva je činjenica da STSI (Space Telescope Science Institute) koristi Astropy u radu Hubble teleskopa. Popis projekata koji koriste Astropy vidljiv je u Tablici 3.1.

Naziv projekta	Područje uporabe
The National Virtual Observatory	Korištenje Astropy VOTable klasa
The Hyper Suprime-Cam – 900 megapikselna ultra širokokutna kamera	Korištenje u korekciji rada kamere
ALMA i CARMA	Korišten kao alat za obradu velikih količina podataka u radioastronomiji
pcigale	pretvorba u Python koda CIGALE (Code Investigating Galaxy Emission) koji je bio napisan u Fortranu
High Energy Astrophysical Phenomena	Analiza optičkog naknadnog sjaja snopa gama zraka
HEASARC (The High Energy Astrophysics Science Archive Research Center)	Koriste Astropy kao "jedino rješenje za astronomiju u Pythonu"
Projekt PANOPTES (Panoptic Astronomical Networked Optical Observatory for Transiting Exoplanets Survey)	U radu projekta je izražena velika uporaba Astropy paketa
Astrophysics Source Code Library	Uvršten u ovu biblioteku programskog koda za astrofiziku

Tablica 3.1: Neki projekti koji koriste Astropy. Preuzeto sa: https://en.wikipedia.org/wiki/Astropy

Astropy paket se sastoji od većeg broja modula. Neki od njih su: apsolutno vrijeme i datum, mjerne jedinice i njihova konverzija, obrada skupa podataka u obliku tablica, obrada najčešće korištenih astronomskih datotečnih formata i spremanje u njih, transformacije svjetskog koordinatnog sustava (World Coordinate System) te kozmološki alati.

3.1 Mjerne jedinice (astropy.units)

Potreba za ovim paketom se javlja zbog velikog broja mjernih jedinica koje znanstvenici koriste. Više jedinica se koristi zato jer je često puno zgodnije raditi s određenim mjernim jedinicama za određena mjerenja i obradu rezultata. Ono što uistinu krati vrijeme prilikom obrade rezultata i pisanja programa u Astropy paketu je to što se mjerne jedinice pozivaju bez da se prethodno moraju inicijalizirati putem neke varijable kako je to inače običaj u programskim jezicima. Njihovo pozivanje je vrlo jednostavno i vrlo slično kao u LaTeX-u [17,21,22]. Značajka koja također ubrzava posao je da unutar paketa postoji veza među različitim mjernim jedinicama što sam program prepoznaje. To je vrlo zgodno kada radimo s podacima poput frekvencije i valne duljine kod svjetla. Ta značajka je izuzetno bitna za Astropy projekt jer dotadašnja programska rješenja nisu imala tu mogućnost. Primjer pretvorbe u Astropy paketu možemo vidjeti na Slici 3.1.

```
Define a quantity from scalars and units:
>>> from astropy import units as u
>>> 15.1 * u.m / u.s
<Quantity 15.1 m / (s)>

Convert a distance:
>>> (1.15e13 * u.km).to(u.pc)
<Quantity 0.372689618289 pc>
```

Slika 3.1: Pridruživanje mjerne jedinice skalaru i pretvorba iz jedne mjerne jedinice u drugu. Preuzeto sa: http://arxiv.org/pdf/1307.6212v1.pdf

Kako Astropy nastoji sve držati na jednom mjestu i biti jedan alat za sve, tako se moralo pronaći i rješenje za standard prikaza rezultata s mjernim jedinicama. Budući da smisao Astropy projekta nije ograničiti ljude, Astropy podržava sve standarde s mogućnošću konverzije istih, a ti standardi su FITS standard i VOUnit standard. Najčešće korištena značajka u ovom paketu za mjerne jedinice je direktno vezanje skalara ili polja iz NumPy uz određenu mjernu jedinicu, a to se radi preko Quantity objekta s kojim je moguće izvoditi sve aritmetičke radnje pritom zadržavajući mjerne jedinice. Taj objekt se dakako može konvertirati i u druge mjerne jedinice, no ne samo objekt već i cijeli skupovi podataka. Razlog tome je već prije spomenuta mogućnost rada s NumPy poljima [17,21,22]. Korisnost ove značajke se već nameće sama po sebi, a to je dimenzijska analiza, koja izgleda kao bezazlena početnička greška, ali u rezultatima može napraviti veliku pomutnju. Primjer kako se može reprezentirati jedna takva greška nalazi se na Slici 3.2.

```
try:
    grav_force(20*u.m, 500 * u.kg)
except u.UnitsError, e:
    print e


'm4 solMass / (kg3 s2)' and 'N' (force) are not
convertible
```

Slika 3.2: Primjer kako detektirati grešku prilikom dimenzijske analize. Preuzeto sa: http://arxiv.org/pdf/1307.6212v1.pdf

U toj Slici možemo vidjeti da je program otporan na greške prilikom konverzije mjernih jedinica. Česta greška se javlja i pri pretvaranju u stupnjeve ili radijane prilikom korištenja trigonometrijskih funkcija, no i za to postoji rješenje koje možemo vidjeti u primjeru na Slici 3.3.

```
phi = 30 * u.deg
print "sin(%.1f %s) = %.4f" % (phi.value, phi.unit, np.sin(phi))
phi = phi.to(u.rad)
print "sin(%.5f %s) = %.4f"" % (phi.value, phi.unit, np.sin(phi))
phi = phi.to(u.arcsec)
print "sin(%.0f %s) = %.4f"" % (phi.value, phi.unit, np.sin(phi))
sin(30.0 deg) = 0.5000
sin(0.52360 rad) = 0.5000
sin(108000 arcsec) = 0.5000
```

Slika 3.3: Pretvorba iz stupnjeva u radijane. Preuzeto sa: http://joergdietrich.github.io/python-physics-equations.html

Astronomske i neke fizičke konstante su standardni dio ovoga paketa. Prilikom njihovog pozivanja dobiju se osnovni podaci o konstanti uz mjernu jedinicu i procijenjenu grešku uz dodatne podatke o konstanti, naravno tamo gdje su oni dostupni. Korisnik isto tako može dodati konstante u paket [17].

3.2 Vrijeme (astropy.time)

Modul unutar Astropy paketa zadužen za vrijeme ima funkcionalnost upravljanja vremenom i datumom i to u oblicima:

- TAI - International Atomic Time
- UTC - Coordinated Universal Time
- TCB - Barycentric Coordinate Time
- TCG - Geocentric Coordinate Time
- TDB - Barycentric Dynamical Time
- TT - Terrestrial Time
- UT1 - Universal Time

Vrijeme se može prikazivati u formatima JD (Julian Day), MJD (Modified Julian Date) ili ISO 8601 (prikaz klasičnog vremenskog formata), a samo vrijeme se u paketu kreira kao objekt uz željeni format i vremensku skalu. Cijeli paket za vrijeme u paketu Astropy je

napravljen prema SOFA (Standards of Fundamental Astronomy) vremensko-datumskoj biblioteci i kao posljedica toga koriste se dvije 64-bitne vrijednosti s dinamičkim rasponom 30 redova veličine kako bi iskazani rezultat bio što precizniji kao i sama konverzija [18,22]. Primjer jedne takve konverzije moguće je vidjeti na Slici 3.4.

```
#Prijenos datuma/vremena u ISO format na UTC skali
>>> from astropy.time import Time
>>> t = Time("2010-06-01 00:00:00", ... format="iso", scale="utc")
>>> t
<Time object: scale='utc' format='iso' vals=2010-06-01 00:00:00.000>
#Vrijeme u Julian Date formatu
>>> t.jd 2455348.5
#Vrijeme u year:day:time formatu
>>> t.yday '2010:152:00:00:00.000'
#Pretvorba vremena u TT skalu
>>> t.tt <Time object: scale='tt' format='iso' vals=2010-06-01
00:01:06.184>
#Prikaz Julian Date na TT skali
>>> t.tt.jd 2455348.5007660184
```

Slika 3.4: Primjer konverzije vremena iz ISO formata na UTC skali u JD format na TT skali. Upit napravljen prema izvoru: http://arxiv.org/pdf/1307.6212v1.pdf

3.3 Nebeske koordinate (astropy.coordinates)

Ovaj paket je nastao kombinacijom više postojećih, a to su redom: kapteyn, astropysics, palpy, pyast i pyephem. Naglasak ovog paketa stavlja se na jednostavno sučelje za upis i ispis koordinata, upravljanje njima te njihovu pretvorbu u druge koordinatne sustave. Kao i u većini potpaketa u projektu Astropy i ovdje je moguće unositi vlastite podatke u smislu vlastitih koordinatnih sustava. Za unos koordinata i upravljanje njima potrebno je stvoriti objekt za koordinate, a oni koji se službeno koriste imaju već svoje predefinirane klase. Tako da ako želimo koordinate u ICRS (International Celestial Reference System) koordinatnom sustavu one se pozivaju stvaranjem objekta naslijeđenog iz klase ICRSCoordinates. Primjer pozivanja nekih koordinata te transformacija istih u drugi sustav

možemo vidjeti na Slici 3.5. Koordinate se mogu naći i provjeriti u SIMBAD astronomskoj biblioteci koja sadrži osnovne podatke, literaturu i mjere za astronomske objekte van Sunčevog sustava. Moguće je pronaći i koordinate nekog tijela ukoliko znamo njegovo ime što je prikazano na Slici 3.5 za primjer galaksije Messier 32 (M32). To je patuljasta eliptična galaksija koja je satelit galaksije Andromeda (M31). Kako smo već i prije spomenuli, razni koordinatni sustavi imaju vlastite klase unutar Astropy paketa. U paketu su sadržani ekvatorijalni koordinatni sustavi ICRS, FK4 i FK5, galaktički koordinatni sustav, horizontalni (Alt/Az) koordinatni sustav, moderni IAU 2006/200A modeli za pripadne sustave, supergalaktički koordinatni sustav, eliptični sustav te još neki potrebni za konverziju iz eliptičnog u horizontalni sustav [18,22,23].

```
>>> import astropy.coordinates as coords
>>> c = coords.ICRSCoordinates
("00h42m44.3s +41d16m9s")


>>> c.ra <RA 10.68458 deg>
>>> c.dec
<Dec 41.26917 deg>
>>> c.ra.degrees 10.68458333333333
>>> c.ra.hms
(0.0, 42, 44.299999999999784)
#Pretvorba u galaktičke koordinate
>>> c.galactic.l <Angle 121.17431 deg> >>> c.galactic.b
<Angle -21.57280 deg>


>>> from astropy import units as u
>>> g = c.transform_to(coords.GalacticCoordinates) >>>
g.l.format(u.degree, sep=":", precision=3)
'121:10:27.499'


>>> c.distance = coords.Distance(770., u.kpc)
>>> c.x 568.7128882165681
>>> c.y 107.30093596881028
>>> c.z 507.8899092486349
#dobivanje koordinata pozivanjem imena objekta u sustavu SIMBAD
>>> m = coords.ICRSCoordinates.from_name("M32")
>>> m
<ICRSCoordinates RA=10.67427 deg, Dec=40.86517 deg>
```

Slika 3.5: Pretvorba u neke od koordinatnih sustava. Upit napravljen prema izvoru: http://arxiv.org/pdf/1307.6212v1.pdf

Korisna je i mogućnost nadodavanja vlastitih koordinatnih sustava unutar paketa, budući da postoji transformacijski graf koji prati međusobnu povezanost koordinatnih sustava i transformacija među njima. Navedeni graf također prati i koje se sve transformacije moraju odraditi kako bi se iz jednog koordinatnog sustava prešlo u drugi utvrđujući najkraći i

najbrži put. Ako transformiramo iz sustava A u sustav B, graf će odrediti kombinaciju za tu pretvorbu koja je najbrža i koja uključuje najmanje međupretvorbi.

3.4 Tablice i podaci u rešetci (astropy.table i astropy.nddata)

Podaci smješteni u tablicu odnosno u neku vrst rešetke su najčešći oblici u kojima se pojavljuju podaci u astronomiji. Najčešće se koriste NumPy strukturirana polja poput ndarray. Možemo postaviti pitanje zašto se onda ne uzima ndarray nego se izmišlja novi oblik za manipulaciju podacima. Problem kod ndarray strukturiranih polja je što u izvornom obliku ona u sebi ne mogu sadržavati dodatne informacije potrebne u astronomiji kao što su primjerice mjerne jedinice, dodatna polja za opis, dodatna svojstva podataka, FITS zaglavlja, itd. Korištenje potpaketa astropy.table i astropy.nddata rješava taj problem te omogućuje korisniku pohranu podataka u obliku tablica ili n-dimenzionalnih rešetkastih skupova podataka uključujući sve dodatne informacije o pojedinom podatku.

Kako NumPy strukturirana polja imaju heterogeni tip podataka te stupce i redove je malo teže modificirati. Klasa Table u ovom potpaketu stvara neku vrstu omotača na ista ta polja kako bi se mogle stvarati tablice iz stupaca te bez komplikacija micali uklanjali ili dodavali dodatni redovi ili stupci. Same tablice se zapisuju koristeći metode Table.write i Table.read i to u korisniku poznate datotečne formate, jer su te metode povezane s potpaketom astropy.io zaduženim za unos i ispis koji omogućuje čitanje i pisanje u razne formate. Primjer osnovnog rada s tablicom nalazi se na Slici 3.6. Kako bi se dodatno olakšalo upravljanje objektima u tablici, klasa Table omogućava da se pojedinim stupcima pridijele različite mjerne jedinice koristeći prethodno spomenuti potpaket za mjerne jedinice te omogućuje pohranu dodatnih podataka o objektu koristeći Table.meta metodu [18,22].

```
#kreiranje prazne tablice i dodavanje stupaca
>>> from astropy.table import Table, Column
>>> t = Table()
>>> t.add_column(Column(data=["a", "b", "c"], ... name="source"))
>>> t.add_column(Column(data=[1.2, 3.3, 5.3], ... name="flux"))
>>> print t
source flux
------ ---
     a 1.2
     b 3.3
     c 5.3
#iščitavanje tablice iz datoteke
>>> t1 = Table.read("catalog.vot")
>>> t1 = Table.read("catalog.tbl", format="ipac")
>>> t1 = Table.read("catalog.cds", format="cds")
#označavanje svih redova u t1 gdje je vrijednost flux veća od 5
>>> t2 = t1[t1["flux"] > 5.0]
#upravljanje stupcima
>>> t2.remove_column("J_mag")
>>> t2.rename_column("Source", "sources")
#ispis tablice u datoteku
>>> t2.write("new_catalog.hdf5", path='/table')
>>> t2.write("new_catalog.rdb")
>>> t2.write("new_catalog.tex")>
```

Slika 3.6: Neke osnovne radnje s tablicama u Astropy paketu. Upit napravljen prema izvoru: http://arxiv.org/pdf/1307.6212v1.pdf

Vrlo slično se ponaša i klasa NDData koja je napravljena prema NumPy ndarray klasi, a služi pohrani podataka u n-dimenzionalna polja. Podaci su spremljeni u klasično polje tipa ndarray što omogućava lakšu obradu i čitanje u drugim znanstvenim paketima za obradu podataka, dok se sve ostale dodatne informacije o podacima kao što su mjerne jedinice, koordinatni sustav, greške i ostalo spremaju u n-dimenzionalno polje iste veličine kao i originalna tablica ndarray oblika. Dodatne značajke NDData nisu usmjerene krajnjem korisniku nego obradi podataka za daljnje korištenje od strane potklasa višeg stupnja čiji rezultati primjerice služe prikazu spektra ili astronomskim slikama [18].

3.5 Formati datoteka (astropy.io)

Već smo prije spomenuli ovaj potpaket u kontekstu čitanja određenih formata datoteka i zapisa te spremanja u iste. Jedan od najraširenijih tipova datoteka u astronomiji je FITS (Flexible Image Transport System). Prema imenu se može zaključiti kako se radi o formatu vezanom za slike što je djelomično istina budući da uz samu sliku takav tip podatka sadrži mnoštvo drugih podataka poput fotometrijske i prostorne kalibracije zajedno sa svim ostalim bitnim podacima. U astronomiji je češća primjena ovog formata za podatke koji nisu slike. Tako se u FITS format mogu spremiti općeniti podaci, ili čak strukturirani podaci poput baze podataka s više tablica, a jedan podatak u FITS formatu može sadržavati i više informacija o više raznih objekata. Naime, svaki podatak u FITS formatu može imati svoju ekstenziju koja može sadržavati novi objekt, što čini vrlo praktičnim spremanje više različitih vrsta podataka u jednu datoteku. Na slici 3.7 možemo vidjeti primjer elementarnih operacija na jednom podatku koji se nalazi u FITS formatu [18,22,23].

```
#čitanje FITS datoteke sa diska
>>> from astropy.io import fits
>>> hdus = fits.open("sample.fits")
#pristup zaglavlju prvog HDU
>>> hdus[0].header
SIMPLE    =           T
BITPIX    =         -32
NAXIS     =           3
NAXIS1    =         200
NAXIS2    =         200
NAXIS3    =          10
EXTEND    =           T
#pristup obliku podatka u prvom HDU:
>>> hdus[0].data.shape
(10, 200, 200)
#promjena/dodavanje ključnih riječi u zaglavlje
>>> hdus[0].header["TELESCOP"] = "Mt Wilson"
>>> hdus[0].header["OBSERVER"] = "Edwin Hubble"
#množenje podataka sa 1.2
>>> hdus[0].data *= 1.2
#zapis na disk
>>> hdus.writeto("new_file.fits")
```

3.7: Neke osnovne operacije nad podatkom u FITS formatu. Upit napravljen prema izvoru: http://arxiv.org/pdf/1307.6212v1.pdf

Budući da se FITS tip podataka koristi jako dugo, pojavio se veliki broja njegovih izvedenica. Neke od njih djeluju na prvu ruku nejasno budući da podržavaju enkapsulaciju datoteke u FITS formatu na način da je moguće postojanje više datoteka u FITS formatu u postojećoj datoteci u FITS formatu. Astropy može raditi i s takvima, no korisniku je bitno znati s čim točno radi pa zato i postoji funkcija fitscheck koja provjerava odstupa li neki podatak koji bi načelno trebao biti u FITS formatu od standarda postavljenih za takvu vrstu podataka. Isto tako, postoji i funkcija fitsdiff koja uspoređuje dva podatka u FITS formatu prema više ključeva usporedbe.

Jedan, također široko rasprostranjen format podatka je i ASCII tablični format. Naime, postoji i mnoštvo formata temeljenih na ASCII standardu, a možda najpoznatiji prikazi takvih tablica su one kojima su stupci odvojeni razmakom, zarezom ili nekim drugim simbolom. No, postoje i druge tablice temeljene na ovom formatu a neke od njih su CDS, IPAC, IRAF DAOphot te širokoj zajednici dobro poznat LaTeX. Dodatna mogućnost koju ovaj potpaket pruža je i stvaranje vlastitih čitača i pisača za razne ASCII formate [18].

Treći među najkorištenijim vrstama podataka su tablični podaci vrste Virtual Observatory, odnosno format VOTable. Slično kao u podacima FITS formata, glavni dio podataka je spremljen u polje tipa NumPy, ali ono što mu je velika prednost nad ostalima je taj da se tablica nikad ne učitava cijela nego dio po dio što značajno poboljšava performanse računala na kojem se izvodi neka radnja s takvim tipom podataka, smanjuje zagušenje memorije i znatno ubrzava cijeli proces obrade podataka. Bez obzira na to što se tablica u ovom formatu možda ne učita cijela, moguće ju je urediti putem Table objekta iz ovog potpaketa te izmjene spremiti u originalnu datoteku bez gubitka podataka. Problem koji se najčešće javlja prilikom čitanja i obrade podataka ovog formata je da pisači VOTable formata ne paze na standard što čitanje i obradu čini znatno težim te je zato u Astropy uvedena podrška i za takve podatke. Kako bi se ubuduće izbjeglo širenje takvih vrsta VOTable podataka, unutar potpaketa astropy.io.votable nalazi se alat volint koji provjerava koliko određeni podatak odstupa od standarda [18].

3.6 World Coordinate System (astropy.wcs)

Malen ali značajan dio Astropy paketa nalazi se upravo u ovom potpaketu. Naime, ovaj potpaket služi za upravljanje World Coordinate System (WCS) transformacijama u podacima formata FITS. Konkretno, mape nastale transformacijom jasno povezuju neki piksel na slici sa stvarnim položajem na nebu. Biblioteka wcslib je u neku ruku postala standard za ovu transformaciju budući da se parsiranje FITS zaglavlja koristi isključivo tom bibliotekom. Ovaj potpaket sadrži i alat koji provjerava transformaciju kako bi ukazao na neke greške i odstupanja u podacima u FITS formatu [18].

3.7 Kozmologija (astropy.cosmology)

Kozmologija je još jedna grana astrofizike koju paket Astropy pokriva i to putem potpaketa astropy.cosmology. Koristi se prilikom kozmoloških izračuna jer sadrži funkcije za računanje veličina ovisnih o pojedinim kozmološkim modelima, klase za najčešće korištene kozmologije te svojevrstan okvir za rjeđe korištene kozmologije koje imaju vremenski ovisan omjer pritiska i gustoće w za tamnu energiju. Neki od primjera su kritična gustoća, udaljenost svjetlosti, Hubble-ov parametar i ostali. Kako je ova grana fizike složena, autori su se trudili zadržati ovaj paket jednostavnim za korištenje pa je tako svaka kozmologija prikazana određenom klasom u paketu. Kad netko koristi određenu kozmologiju, mora unijeti podatke poput Hubble-ovog parametra, CMB temperature, barionske hladne tamne tvari, i sl. [18].

4. Astroquery: primjena paketa Astropy za pristup podacima u bazama

Namjena potpaketa Astroquery je stvaranje upita prema astronomskim bazama podataka te upravljanje tim upitima [24]. Ovaj dio Astropy portfelja je vrlo često korišten. Postoji mnoštvo primjena ovoga paketa, a ovdje ćemo prikazati samo neke od njih budući da je područje primjene vrlo široko. To široko područje unaprijed je uvjetovano količinom objekata koji se nalaze u raznim bazama podataka te njihovom kompleksnošću posebice ako imamo na umu da se najčešće koristimo prethodno spomenutim višedimenzionalnim poljima koji u sebi mogu sadržavati još takvih polja. Imajući sve navedeno na umu jasno nam je koliko kompleksno rješenje mora biti za upravljanje takvim podacima. Izvedba tog rješenja nikako ne smije biti napravljena nauštrb brzine izvođenja, jednostavnosti i efikasnosti uporabe. Jasno je da su tu i dodatni uvjeti poput toga da takav proizvod mora biti rješenje za sve vrste problema te da je po mogućnosti besplatan, konstantno održavan i unaprjeđivan.

Kako se upiti u Astropy-u pa tako i u Astroquery-u unose i izvršavaju u terminalu odnosno komandnoj liniji, tako je i rezultat tih upita prikazan na isti način. Nebitno je radi li se o klasičnom terminalu ili o nekoj bilježnici u Internet pregledniku koja koristi Jupyter, Astroquery kao dio Astropy paketa nema grafičko sučelje. Predstavit ćemo primjere upita u Simbadu, jednom od najkorištenijih astronomskih servisa.

4.1 SIMBAD

SIMBAD je astronomska baza podataka o objektima koji se nalaze van Sunčevog sustava, a nastao je 1979. spajanjem CSI (Catalog of Stellar Identifications) i Bibliographic Star Index [25]. Trenutno ga održava astronomski centar u Strasbourgu, Francuska. U trenutku pisanja ovog rada baza sadrži preko 8 milijuna objekata, više od 22 milijuna raznih imena, preko 300 tisuća bibliografskih referenci i više od 13 milijuna bibliografskih citata.

MAIN_ID	Glavni identifikator astronomskog objekta
RA	Rektascenzija
DEC	Deklinacija
RA_PREC	Preciznost rektascenzije (0:1/10deg, ..., 8: 1/1000 arcsec)
DEC_PREC	Preciznost deklinacije (0:1/10deg, ..., 8: 1/1000 arcsec)
COO_ERR_MAJA	Velika poluos elipse pogreške
COO_ERR_MINA	Mala poluos elipse pogreške
COO_ERR_ANGLE	Kut elipse pogreške
COO_QUAL	Kvaliteta (A: astrometrička, …, E: nepoznata)
COO_WAVELENGTH	Tip spektra mjerenja (radio, infracrveni, vidljivi, UV, X, gama)
COO_ BIBCODE	Bibliografska referenca

Tablica 4.1: Tablica izlaznih varijabli koristeći SIMBAD[4]

Tablica 4.1 prikazuje neke od izlaznih varijabli baze SIMBAD. U primjeru upita pokušat ćemo nešto saznati o objektu znanom kao M64, što je ime prema klasifikaciji Messierovih objekata, odnosno listi od 110 astronomskih objekata koje je u katalog organizirao francuski astronom Charles Messier. Njegova inicijalna verzija kataloga je 1774. godine sadržavala 45 objekata, a zadnji objekt je uvršten u taj katalog 1967. M64 objekt, poznat i kao Crnooka galaksija [26], je prikazan na Slici 4.1. To je spiralna galaksija koja je ime dobila po obliku oblaka prašine i plina koji je okružuju. Udaljena je 24 milijuna svjetlosnih godina te se nalazi u sazviježđu Berenikina kosa. Primjer upita za objekt M64 može se vidjeti na Slici 4.2.

[4] Tablica izrađena prema podacima sa službene tablice SIMBAD-a: http://simbad.u-strasbg.fr/simbad/sim-help?Page=sim-fscript



Slika 4.1: Messier 64 (M64), Crnooka galaksija. Preuzeto sa:
https://upload.wikimedia.org/wikipedia/commons/c/c4/Blackeyegalaxy.jpg

```
>>> from astroquery.simbad import Simbad
>>> result_table = Simbad.query_object("M64")
>>> print(result_table)
MAIN_ID      RA           DEC       ... COO_WAVELENGTH     COO_BIBCODE
          "h:m:s"      "d:m:s"     ...
------- ------------ ------------ ... -------------- -------------------
  M  64 12 56 43.696 +21 40 57.57 ...              I 2006AJ....131.1163S
>>>
```

Slika. 4.2: Upit za objekt M64

Kao što je prije spomenuto, ovdje možemo vidjeti da je rezultat našeg upita prikazan u obliku tablice, ali isključivo u obliku teksta. Uzmimo za primjer i objekt M27. To je objekt koji također pripada Messierovom katalogu, a označava Maglicu bućica, na Slici 4.3, koja je planetarna maglica te se nalazi u zviježđu Lisica. Ta maglica je udaljena oko 1360

svjetlosnih godina, a upravo ta udaljenost ju čini dobrim materijalom za promatranje od strane astronoma amatera [27].



Slika 4.3: Messier 27 poznatija i kao Maglica bućica. Preuzeto sa: https://upload.wikimedia.org/wikipedia/commons/e/e8/M27_-_Dumbbell_Nebula.jpg

Uporabom baze SIMBAD možemo istražiti koji se objekti nalaze u području oko nekog određenog objekta. Ukoliko ne zadamo radijus u kojem istražujemo prostor oko nekog objekta, automatski zadana vrijednost iznosi 20 kutnih minuta. Primjer pretrage gdje sami zadajemo radijus, moguće je vidjeti na Slici 4.4 gdje pretragu vršimo oko objekta M27.

```
>>> from astroquery.simbad import Simbad
>>> import astropy.units as u
>>> result_table = Simbad.query_region("m27", radius='0d6m0s')
>>> print(result_table)
        MAIN_ID              RA       ... COO_WAVELENGTH     COO_BIBCODE
                           "h:m:s"     ...
---------------------- ------------ ... -------------- -------------------
                  M  27 19 59 36.379 ...              I 2006AJ....131.1163S
              [ML93] G  19 59 36.83 ...
         GB6 B1957+2234  19 59 36.90 ...                1996ApJS..103..427G
              [ML93] J  19 59 35.74 ...
              [ML93] I  19 59 37.62 ...
              [ML93] H  19 59 38.06 ...
              [ML93] K  19 59 35.54 ...
              RRF  518   19 59 35.8 ...
              [ML93] A  19 59 33.46 ...
              [ML93] B  19 59 32.64 ...
              [ML93] C  19 59 33.02 ...
              [ML93] E  19 59 33.46 ...
              [ML93] D  19 59 32.65 ...
              [ML93] F  19 59 32.44 ...
             NSV 24959  19 59 29.73 ...              I 2003yCat.2246....0C
        TYC 2141-1400-1 19 59 25.196 ...              O 2000A&A...355L..27H
2MASS J19595101+2242352 19 59 51.018 ...              I 2003yCat.2246....0C
        NAME VAR VUL 05 19 59 51.283 ...                2005IAUC.8591....2R
        TYC 2141-1086-1 19 59 26.962 ...              O 2000A&A...355L..27H
        IRAS 19578+2236   20 00 00.4 ...              F 1988NASAR1190....1B
>>>
```

Slika 4.4: Upit za pretragu po polju oko objekta M27 sa unaprijed željenim radijusom.

Područje je također moguće istražiti oko određenih koordinata te za tu priliku koristimo astropy.coordinates potpaket kako bi uveli željene koordinate. Isto tako, možemo definirati epohu i ekvinocij kako bi na taj način dodatno filtrirali željene podatke. Primjer gdje koristimo ove dvije značajke možemo vidjeti na Slici 4.5.

```
>>> from astroquery.simbad import Simbad
>>> import astropy.coordinates as coord
>>> import astropy.units as u
>>> result_table = Simbad.query_region(coord.FK5(ra=11.70, dec=10.90,
...                                    unit=(u.deg, u.deg)),
...                                    radius=0.5 * u.degree,
...                                    epoch='B1950',
...                                    equinox=1950)
>>> print(result_table)

        MAIN_ID               RA      ...     COO_BIBCODE
----------------------- ------------ ... -------------------
              PHL  6696       00 49.4 ...
           BD+10    97 00 49 25.4553 ... 2007A&A...474..653V
         TYC  607-238-1  00 48 53.302 ... 2000A&A...355L..27H
              PHL  2998       00 49.3 ...
2MASS J00492121+1121094  00 49 21.219 ... 2003yCat.2246....0C
        TYC  607-1135-1 00 48 46.5838 ... 1998A&A...335L..65H
2MASX J00495215+1118527  00 49 52.154 ... 2006AJ....131.1163S
           BD+10    98 00 50 03.4124 ... 1998A&A...335L..65H
                   ...           ... ...                 ...
         TYC  607-971-1 00 47 38.0430 ... 1998A&A...335L..65H
         TYC  607-793-1  00 50 35.545 ... 2000A&A...355L..27H
USNO-A2.0 0975-00169117  00 47 55.351 ... 2007ApJ...664...53A
         TYC  607-950-1  00 50 51.875 ... 2000A&A...355L..27H
           BD+10   100 00 51 15.0789 ... 1998A&A...335L..65H
          TYC  608-60-1  00 51 13.314 ... 2000A&A...355L..27H
         TYC  608-432-1  00 51 05.289 ... 2000A&A...355L..27H
```

Slika 4.5: Upit koji koristi koordinate, epohu i ekvinocij kako bi dodatno filtrirali rezultate upita. Preuzeto sa: https://astroquery.readthedocs.org

Ukoliko želimo pretraživati po nekom određenom katalogu, dovoljno je navesti samo ime kataloga i tada će nam rezultat upita izbaciti tablicu sa svim objektima iz određenog kataloga kako možemo vidjeti u primjeru na Slici 4.6.

```
>>> from astroquery.simbad import Simbad
>>> limitedSimbad = Simbad()
>>> limitedSimbad.ROW_LIMIT = 6
>>> result_table = limitedSimbad.query_catalog('eso')
>>> print(result_table)


MAIN_ID              RA      ... COO_WAVELENGTH     COO_BIBCODE
---------------------- ------------ ... -------------- ---------------
----
2MASS J08300740-4325465  08 30 07.41 ...              I
2003yCat.2246....0C
NGC  2573 01 41 35.091 ...              I 2006AJ....131.1163S
ESO   1-2   05 04 36.8 ...                1982ESO...C......0L
ESO   1-3 05 22 36.509 ...              I 2006AJ....131.1163S
ESO   1-4 07 49 28.813 ...              I 2006AJ....131.1163S
ESO   1-5 08 53 05.006 ...              I 2006AJ....131.1163S
```

Slika 4.6: Pretraga objekata iz određenog kataloga, primjer za katalog ESO (European Southern Observatory). Preuzeto sa: https://astroquery.readthedocs.org

U astronomiji se koristi bibcode, odnosno Bibliographic Reference Code. To je jedinstvena oznaka od 19 simbola koja se koristi kao standard u označavanju literature u astronomskoj bibliografiji. Koristeći bibcode, pravilnim upitom, možemo izvući pojedinu referencu. Primjer možemo vidjeti na Slici 4.7.

```
>>> from astroquery.simbad import Simbad
>>> result_table = Simbad.query_bibcode('2005A&A.430.165F')
>>> print(result_table)


References
---------------------------------------------------------------------
---------------------------------------------------------------------
---------------------------------------

2005A&A...430..165F  --  ?
Astron. Astrophys., 430, 165-186 (2005)
FAMAEY B., JORISSEN A., LURI X., MAYOR M., UDRY S., DEJONGHE H. and
TURON C.
Local kinematics of K and M giants from CORAVEL/Hipparcos/Tycho-2 data.
Revisiting the concept of  superclusters.
Files: (abstract)
Notes: <Available at CDS: tablea1.dat notes.dat>
```

Slika 4.7: Ispis reference uz zadani bibcode. Preuzeto sa: https://astroquery.readthedocs.org

No, bibcode se vrlo često koristi i u pretrazi objekata koji se spominju u određenoj literaturi kao što je primjerom pokazano na Slici 4.8.

```
>>> from astroquery.simbad import Simbad
>>> result_table = Simbad.query_bibobj('2005A&A.430.165F')
>>> print(result_table)


MAIN_ID       RA           DEC       RA_PREC DEC_PREC COO_ERR_MAJA
COO_ERR_MINA COO_ERR_ANGLE COO_QUAL COO_WAVELENGTH     COO_BIBCODE
--------- ------------ ------------ ------- -------- ------------ -----
------- ------------- -------- -------------- -------------------
NGC   524 01 24 47.707 +09 32 19.65       7        7          nan
nan             0        B              I 2006AJ....131.1163S
NGC  3593 11 14 37.002 +12 49 04.87       7        7          nan
nan             0        B              I 2006AJ....131.1163S
NGC  4138 12 09 29.788 +43 41 07.14       7        7          nan
nan             0        B              I 2006AJ....131.1163S
NGC  4550 12 35 30.612 +12 13 15.44       7        7          nan
nan             0        B              I 2006AJ....131.1163S
NGC  5179 13 29 30.875 +11 44 44.54       7        7          nan
nan             0        B              I 2006AJ....131.1163S
NGC  5713 14 40 11.528 -00 17 21.16       7        7          nan
nan             0        B              I 2006AJ....131.1163S
```

Slika 4.8: Pretraga objekata pomoću bibcode-a. Preuzeto sa: https://astroquery.readthedocs.org

4.2 VizieR i Open Exoplanet

Na Slici 4.9 je prikazana uporaba astronomskog web servisa pod nazivom VizieR. Postoje velike sličnosti s uporabom servisa Simbad. VizieR se od 1996. godine koristi kao pristupna točka velikom broju objavljenih podataka u astronomskim katalozima [28]. Broj kataloga obuhvaćenih u ovom servisu u trenutku pisanja iznosi preko 14 000. VizieR je nastao u Astronomskom podatkovnom centru u Strasbourgu - Centre de données astronomiques de Strasbourg. Zanimljiva je činjenica da ovaj servis korijene vuče iz ESIS-ova (European Space Infromation System) projekta koji je kao svojevrsna umrežena baza trebao omogućiti pristup heterogenom skupu kataloga i podataka. U tome je kronološki pretekao World Wide Web. Kao i kod Simbada, pozivanjem upita prema ovom servisu vraćaju se najbitniji podaci o nekom objektu koji se nalazi u određenom katalogu, a u primjeru na Slici 4.9 tražimo zvijezde tipa bijeli patuljak.

```
>>> from astroquery.vizier import Vizier
>>> v = Vizier(keywords=['stars:white_dwarf'])
>>> from astropy import coordinates
>>> from astropy import units as u
>>> c = coordinates.SkyCoord(0,0,unit=('deg','deg'),frame='icrs')
>>> result = v.query_region(c, radius=2*u.deg)
>>> print len(result)
31
>>> result[0].pprint()
   LP    Rem Name  RA1950   DE1950  Rmag l_Pmag Pmag u_Pmag spClass
pm   pmPA _RA.icrs _DE.icrs
-------- --- ---- -------- -------- ---- ------ ---- ------ ------- ---
--- ---- -------- --------
584-0063          00 03 23 +00 01.8 18.1        18.3              f
0.219   93 00 05 57 +00 18.7
643-0083          23 50 40 +00 33.4 15.9        17.0              k
0.197   93 23 53 14 +00 50.3
584-0030          23 54 05 -01 32.3 16.6        17.7              k
0.199  193 23 56 39 -01 15.4
```

Slika 4.9: Korištenje astroquery za web servisom VizieR. Preuzeto sa: https://astroquery.readthedocs.org

Na Slici 4.10 vidimo kako možemo pronaći masu planeta Kepler 68b služeći se podacima iz modula open_exoplanet_catalogue. Primijetimo da je masa ovako ispisana iskazana u

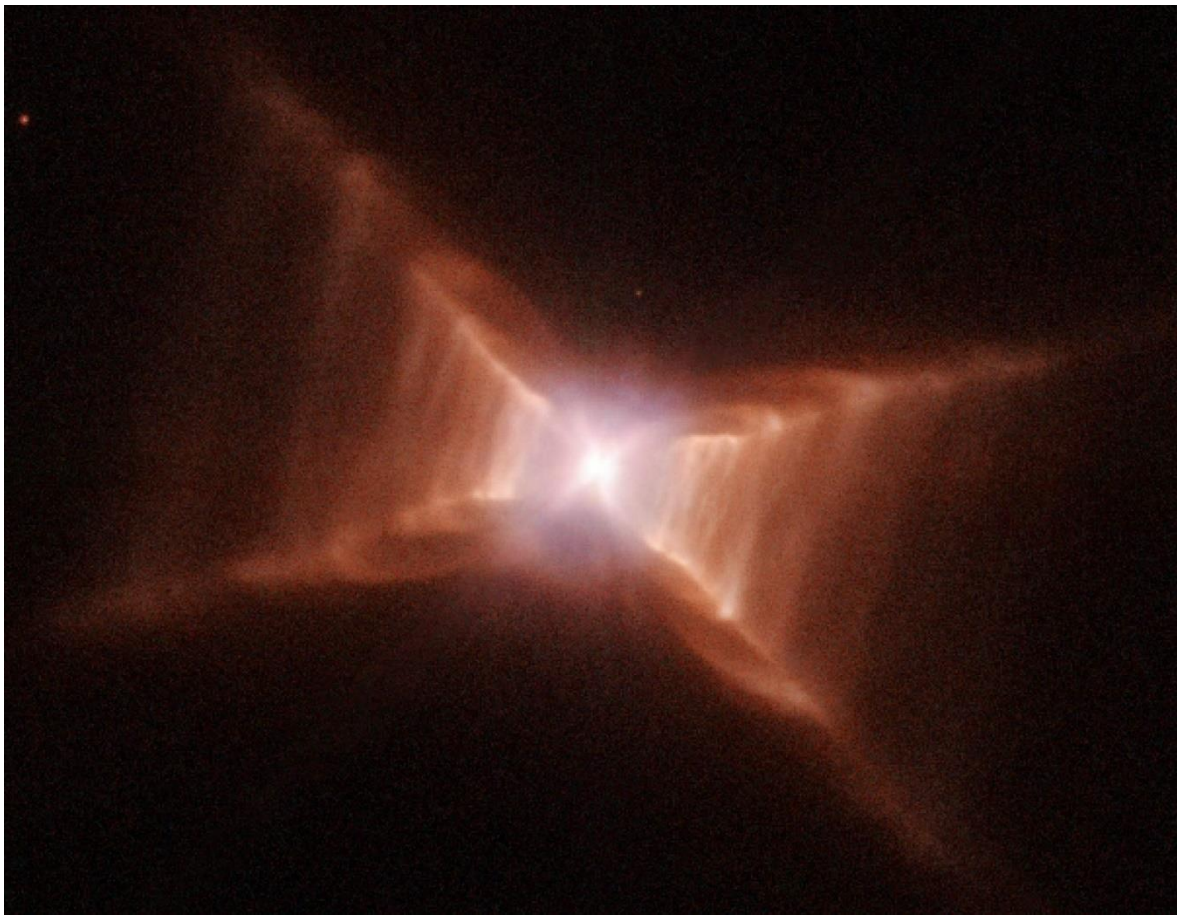
jedinicama mase planeta Jupitera koja iznosi 317.83 mase planeta Zemlje ili $1.898 \cdot 10^{27}$ kg.

```
>>> from astroquery import open_exoplanet_catalogue as oec
>>> from astroquery.open_exoplanet_catalogue import findvalue
>>> cata = oec.get_catalogue()
>>> kepler68b = cata.find(".//planet[name='Kepler-68 b']")
>>> print findvalue( kepler68b, 'mass')
0.02105109
```

Slika 4.10: Ispis mase planeta Kepler 68b. Preuzeto sa: https://astroquery.readthedocs.org

4.3 SIMBAD i VizieR na primjeru jednog objekta

Koristeći Astroquery izvršit ćemo upit prema SIMBAD i ViZieR bazama za jedan određeni objekt. To je HD 44179 dvojna zvijezda u maglici Crvenog Pravokutnika [29]. Ta maglica se nalazi u sazviježđu Jednoroga i udaljena je 2300 svjetlosnih godina. Poznata je zbog svog jedinstvenog oblika. Na prvim fotografijama izgled maglice je podsjećao na pravokutnik crvene boje. Kasnije se ispostavilo da se ne radi o pravokutnom obliku, nego o obliku slova X (vidi Sliku 4.11) koji je rezultat specijalne orijentacije diska kozmičke prašine oko HD 44179 te interakcije zračenja zvijezde i prašine. Ta zvijezda je slična našem Suncu, no nalazi se pri kraju svoga životnog puta te će se uskoro pretvoriti u bijelog patuljka.



Slika 4.11: Maglica Crvenog Pravokutnika. Preuzeto sa:
https://upload.wikimedia.org/wikipedia/commons/b/b7/Redrectangle_hst_full.jpg

Koordinate ovoga objekta i njegovu bibliografsku referencu možemo pronaći slanjem upita prema servisu SIMBAD, kao što možemo vidjeti na Slici 4.12. Upit za informacije o objektima koji se nalaze blizu HD 44179 vidimo na Slici 4.13.

```
>>> from astroquery.simbad import Simbad
>>> result_table = Simbad.query_object("HD44179")
>>> print(result_table)
 MAIN_ID        RA           DEC       ... COO_WAVELENGTH     COO_BIBCODE
             "h:m:s"       "d:m:s"     ...
--------- ------------ ------------ ... -------------- -------------------
HD  44179 06 19 58.2185 -10 38 14.706 ...              0 2007A&A...474..653U
>>> _
```

Slika 4.12: Upit u SIMBAD za objekt HD44179 ili Maglicu Crvenog Pravokutnika

```
>>> from astroquery.simbad import Simbad
>>> result_table = Simbad.query_region("HD44179", radius='0d6m0s')
>>> print(result_table)
    MAIN_ID           RA      ... COO_WAVELENGTH     COO_BIBCODE
                   "h:m:s"    ...
-------------- ------------ ... -------------- -------------------
      HD  44179 06 19 58.2185 ...              0 2007A&A...474..653U
IRAS 06176-1037       06 20.0 ...
    BD-10  1480 06 20 15.0963 ...                1998A&A...335L..65H
    BD-10  1475 06 19 54.3820 ...                1998A&A...335L..65H
>>> _
```

Slika 4.13: Upit u SIMBAD za objekt HD 44179 te područje oko njega u radijusu od 0.1 stupnja

Upit za navedeni objekt potražit ćemo i u VizeR bazi podataka kojom ćemo najprije upitom zatražiti listu kataloga u kojima se nalazi željeni objekt HD 44179 što je vidljivo na Slici 4.14 gdje nam upit vraća 126 različitih tablica. Nakon toga ćemo odabrati jedan od kataloga koji će nam dati više informacija o željenom objektu kako je vidljivo na Slici 4.15.

```
>>> from astroquery.vizier import Vizier
>>> result = Vizier.query_object("HD44179")
>>> print(result)
TableList with 126 tables:
        '0:I/119/sd' with 11 column(s) and 1 row(s)
        '1:I/131A/sao' with 43 column(s) and 1 row(s)
        '2:I/141/yale00' with 22 column(s) and 1 row(s)
        '3:I/171/acrs1' with 33 column(s) and 1 row(s)
        '4:I/193/ppm2' with 26 column(s) and 1 row(s)
        '5:I/195/catalog' with 25 column(s) and 1 row(s)
        '6:I/196/main' with 66 column(s) and 1 row(s)
        '7:I/196/annex1' with 22 column(s) and 2 row(s)
```

Slika 4.14: Pretraga objekta HD44197 po katalozima u VizieRu

```
>>> interesting_table = result['II/225/names']
>>> print(interesting_table)
_RAJ2000 _DEJ2000   _r       name      RAB1950    DEB1950   _RA.icrs   _DE.icrs
  deg      deg     arcm                "h:m:s"    "d:m:s"   "h:m:s"    "d:m:s"
-------- -------- ----- ------------ ---------- --------- ---------- ---------
 94.9718 -10.6444 1.294    FIRSSE 160 06 17 32.0 -10 37 18 06 19 53.2 -10 38 40
 94.9885 -10.6372 0.240 RED RECTANGLE 06 17 36.0 -10 36 52 06 19 57.2 -10 38 14
 94.9914 -10.6345 0.190 RED RECTANGLE 06 17 36.7 -10 36 42 06 19 57.9 -10 38 04
 94.9923 -10.6370 0.033      HD 44179 06 17 36.9 -10 36 51 06 19 58.1 -10 38 13
 94.9923 -10.6356 0.112  HD 44179 5"N 06 17 36.9 -10 36 46 06 19 58.1 -10 38 08
 94.9923 -10.6342 0.195 HD 44179 10"N 06 17 36.9 -10 36 41 06 19 58.1 -10 38 03
 94.9923 -10.6370 0.033 RED RECTANGLE 06 17 36.9 -10 36 51 06 19 58.1 -10 38 13
 94.9927 -10.6372 0.012      AFGL 915 06 17 37.0 -10 36 52 06 19 58.2 -10 38 14
 94.9927 -10.6372 0.012       CRL 915 06 17 37.0 -10 36 52 06 19 58.2 -10 38 14
 94.9927 -10.6372 0.012      HD 44179 06 17 37.0 -10 36 52 06 19 58.2 -10 38 14
 94.9927 -10.6372 0.012     RAFGL 915 06 17 37.0 -10 36 52 06 19 58.2 -10 38 14
 94.9927 -10.6392 0.106 RED RECTANGLE 06 17 37.0 -10 36 59 06 19 58.2 -10 38 21
 94.9927 -10.6397 0.139 RED RECTANGLE 06 17 37.0 -10 37 01 06 19 58.2 -10 38 23
 94.9956 -10.6375 0.178    06176-1036 06 17 37.7 -10 36 53 06 19 58.9 -10 38 15
```

Slika 4.15: Odabir jedne od tablica u kojima se nalazi željeni objekt HD 44197 te njezin ispis

5. Astropy u nastavi

Korištenje IT tehnologija stvar je svakodnevice gotovo svakog građanina Republike Hrvatske. Mnogi ljudi prate trendove, trudeći se biti ukorak s vremenom i razvojem novih tehnologija koje nam olakšavaju život. Uz povećani konzumerizam i sve veću kupovinu potrošačke elektronike, danas mnogi, posebice mladi, samoinicijativno uče razvijati i stvarati nova programska rješenja za koje trebaju učiti razne programske jezike. U doba opće povezanosti putem interneta i niskog ulaznog troška potrebne opreme, svatko ima priliku gotovo besplatno naučiti neki programski jezik, ili rad u nekom određenom alatu. Vrlo se često ljudi na taj način doškoluju ili prekvalificiraju kako bi proširili svoj spektar mogućih zanimanja te povećali šanse za pronalazak posla. U tu sliku se uklapa i formalno obrazovanje koje bi tim istim mladim ljudima i generacijama koje tek dolaze, trebalo dati temelje u programiranju i radu s računalom.

5.1 Python i cjeloživotno učenje

Školovanje u osnovnoj i srednjoj školi je odgojno-obrazovni proces, dok se fakultetsko školovanje ostvaruje u većini slučajeva kao isključivo obrazovni proces. Neovisno o stupnju školovanja nekog učenika, cilj je da danas učenika, sutra odraslog čovjeka, adekvatno pripremimo za tržište rada ostavljajući pritom veći prostor za izbor profesije. Iako srednja škola zakonom nije obavezna, izuzetno rijetki učenici je ne upišu nakon završetka osnovno-školskog obrazovanja. Jedan od razloga tome je što zvanje stečeno nakon srednje škole nije dovoljno za većinu poslova koji se danas nude na tržištu. Čak je i nakon završene strukovne škole poželjno imati završen neki dodatni tečaj, ili steći neki certifikat. Upravo to ide u smjeru cjeloživotnog obrazovanja koje se promiče kroz Europski kvalifikacijski okvir (EQF) [30] iz kojeg je proizašao i Hrvatski kvalifikacijski okvir (HKO) [31]. Proces cjeloživotnog učenja uključuje formalno (osnovna i srednja škola, fakultet) i neformalno obrazovanje (stručna osposobljavanja, tečajevi). Možemo reći da cjeloživotno učenje zadovoljava i pedocentristički i sociocentristički pristup u obrazovanju [32], budući da je rezultat cjeloživotnog učenja dobitna kombinacija za pojedinca i za društvo. Pojedinac stječe znanja, vještine uz pripadajuću samostalnost i odgovornost na taj način izgrađujući sebe i napredujući u karijeri, dok društvo dobiva pojedince koji se stalno razvijaju i uče te stečena znanja i vještine utiskuju u svoj rad

bivajući vrijednim članom kolektiva. Ključne kompetencije za cjeloživotno učenje propisane u Europskom kvalifikacijskom okviru su [30]:

• Komunikacija na materinskom jeziku

• Komunikacija na stranom jeziku

• Matematička kompetencija i temeljne kompetencije u prirodnim znanostima i tehnologiji

• Digitalna kompetencija

• Kompetencija učenja

• Društvene i građanske kompetencije

• Smisao za inicijativu i poduzetništvo

• Kulturološka senzibilizacija i izražavanje

Druga, treća, četvrta i peta točka su kompetencije koje je vrlo lako postići u formalnom dijelu obrazovanja i to korištenjem Pythona kao programskog jezika u nastavi informatike, kako u srednjoj i tako i u višim razredima osnovne škole. Python se u nekim školama pokazuje kao jako dobar izbor za učenje programskog jezika gdje učenici već u osnovnoj školi mogu shvatiti i primijeniti osnovne koncepte programiranja pogotovo zbog jednostavne sintakse.

5.2 Primjena Astropy programskog paketa u nastavi

Python u nastavi možemo koristiti kao interdisciplinarno sredstvo učenja. Astropy kao programski paket pisan u Pythonu je savršen primjer tome: učenici bi na satu informatike mogli provjeravati i proširivati svoje znanje koje su usvojili na redovnoj ili dodatnoj nastavi iz fizike u polju astronomije, odnosno na satu fizike se mogu koristiti ovim programskim rješenjem pritom primjenjujući svoje računalne vještine. Dakako, ovaj model nije primjeren za sve uzraste niti vrste škola. Najprimjereniji bi bio uzrast učenika u višim razredima srednje škole, prvenstveno prirodoslovno matematičke gimnazije koja ima pojačanu satnicu fizike i informatike u odnosu na ostale gimnazijske programe.

Korištenjem ovog programskog paketa učenici će sigurno zadovoljiti barem četiri od osam potrebnih kompetencija potrebnih za cjeloživotno obrazovanje. Komunikacija na stranom

jeziku se može zadovoljiti kroz snalaženje u dokumentaciji za ovaj paket te eventualno sudjelovanje u raspravama na raznim forumima budući da je gotovo sav dostupni materijal na engleskom jeziku. Naravno, ne pretpostavlja se da će učenik aktivno doprinositi razvoju programskog paketa Astropy, no sudjelovanjem u raspravama može pronaći odgovor na mnoga pitanja oko korištenja samog paketa, ili o podacima iz astronomije. Matematičku kompetenciju i temeljne kompetencije u prirodnim znanostima i tehnologiji vrlo je lako zadovoljiti korištenjem Astropy programskog paketa budući da se učenik paralelno razvija na polju fizike kao prirodoslovne znanosti te informatike na polju tehnologije. Digitalna kompetencija podrazumijeva korištenje računala u svrhu pronalaska, stvaranja, pohranjivanja obrade te razmjene informacija te sudjelovanje i komuniciranje u kolaborativnim mrežama preko interneta[5], a učenik se kroz aktivno korištenje Astropy programskog paketa susreće sa svime navedenim. O kompetenciji učenja nije potrebno dodatno govoriti, zbog toga što ovdje raspravljamo o korištenju ovog programskog paketa u nastavi.

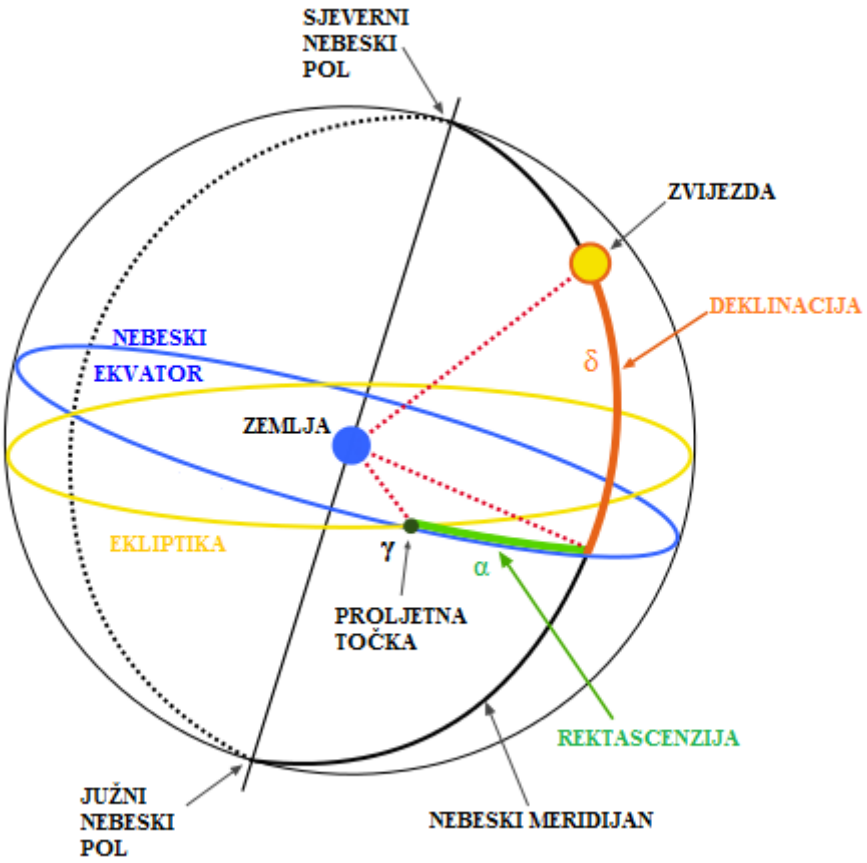
5.2.1 Primjer korištenja Astropy paketa – astropy.coordinates

Primjer koji slijedi bio bi pogodan učenicima koji pohađaju dodatnu nastavu fizike. Obrazovni ishodi postignuti u ovom satu bi bili samostalno pisanje upita u Astroquery, implementacija stečenog znanja iz fizike što bi spadalo pod materijalne ishode. Funkcionalni obrazovni ishodi bi bili: razvijanje sposobnosti uočavanja problema, razvijanje sposobnosti matematičko-logičkog zaključivanja, razvijanje sposobnosti apstraktnog načina razmišljanja, razvijanje sposobnosti primjene usvojenog znanja, razvijanje sposobnosti znanstvenog načina mišljenja te primjena i povezivanje ranije stečenih znanja. Od odgojnih ishoda postigli bi smo razvijanje odgovornosti, temeljitosti i sistematičnosti, uvažavanje tuđih mišljenja i zaključaka te prilagodbu vlastitih uz poticanje na istraživački način razmišljanja. Na taj način bismo ovo izveli kao interaktivnu i istraživački orijentiranu nastavu uz usmjerenu raspravu.

Učenicima bi zadali zadatak koji bi se sveo na pretvaranje jedinica iz jednog koordinatnog sustava u drugi odnosno iz kutnog u njima prirodniji, Kartezijev koordinatni sustav. Prije

[5] Preuzeto s:
Jerbić - Zorc, G., Metodika nastave informatike/tehnike akademska godina 2015/2016. ,predavanje, http://www.phy.pmf.unizg.hr/~gorjana/nastava/Informatika/nastavni materijali/pred_2015.pdf , 05.03.2015.

pristupanja ovom zadatku, učenicima je potrebno razjasniti nebeski koordinatni sustav prikazan na Slici 5.1 budući da ćemo koristiti koordinate izražene kroz deklinaciju (dec - kutnu udaljenost nebeskog tijela od nebeskog ekvatora prema južnom ili sjevernom polu) i rektascenziju (RA - kut od satne kružnice proljetne točke do satne kružnice nebeskog tijela računane u smjeru vrtnje neba).



Slika 5.1: Ekvatorski koordinatni sustav. Preuzeto sa: https://hr.wikipedia.org/wiki/Datoteka:Ekvatorski_koordinatni_sustav_1.png#/media/File:Ekvatorski_koordinatni_sustav_1.png

Također će ih biti bitno upoznati sa standardom za nebeski koordinatni sustav: ICRS – International Celestial Reference System te astronomskim mjernim jedinicama, konkretno parsecima i kiloparsecima, gdje jedan parsec iznosi $3{,}0857 \cdot 10^{16}$ m. Zadatak vidljiv na Slici 5.2 glasi: „Točka u svemiru udaljena je 770 kiloparseca. Njene su koordinate prema ICRS standardu: 00h,42m44.3s, +41d16m9s. Iskaži te koordinate u Kartezijevom koordinatnom sustavu te ispiši svaku koordinatu pojedinačno.“

Rješenje ovog zadatka nije kompliciranо te učeniku daje uvid u služenje programskim paketom Astropy, konkretno njegovim potpaketom astropy.coordinates. Bit ovoga zadatka je da se učenik upozna s osnovama rada u programskom paketu Astropy, korištenjem njegovih potpaketa te promjena načina razmišljanja u smislu da zamjeni korištenje Kartezijevog koordinatnog sustava kutnim koordinatnim sustavom te uvidi da je isti puno prirodniji za korištenje u astronomiji.

```
>>> from astropy import units as u
>>> from astropy.coordinates import Distance
>>> d = Distance(770, u.kpc)
>>> c = ICRSCoordinates('00h42m44.3s +41d16m9s', distance=d)
>>> c
<ICRSCoordinates RA=10.68458 deg, Dec=41.26917 deg, Distance=7.7e+02 kpc>
>>> cp = c.cartesian
>>> cp
<CartesianPoints (568.712888217, 107.300935969, 507.889909249) kpc>
>>> cp.x
568.7128882165681
>>> cp.y
107.3009359688103
>>> cp.z
507.8899092486349
```

Slika 5.2: Zadatak s pretvorbom koordinata korištenjem Astropy programskog paketa. Preuzeto sa: https://astroquery.readthedocs.org

6. Zaključak

Promatranje rada na projektu Astropy, kao primjeru informatičkog projekta u znanstvenoj zajednici, nam može pomoći u shvaćanju važnosti korištenja novih tehnologija. U doba kada su podaci i informacije vrlo važni, te kada podataka u astrofizici ima puno, vrlo je značajno imati rješenje poput Astropy programskog paketa. Taj paket podacima pristupa sustavno, a krajnjem korisniku dopušta upravljanje podacima i njihovu analizu.

Python sve više postaje vrlo traženo znanje u radu znanstvenika jer je s pravom prepoznat kao moćan, jednostavan i lagan jezik koji ima tu odliku da je modularan. Znanstvenici svakodnevno doprinose razvoju projekata poput Astropy jer se pokazalo da takvi alati ubrzavaju proces istraživanja. U zadnje vrijeme se kao trend i stvar opće digitalne pismenosti nameće poznavanje barem jednog programskog jezika, a mnogima je Python prvi i jedini izbor. To za posljedicu ima postojanje sve većeg broja ljudi koji pišu vlastite module i proširenja u Pythonu koja se kasnije mogu uklopiti u veći projekt. Jedan od takvih projekata je programski paket Astropy.

Ohrabrujuća činjenica je da se u zadnje vrijeme radi na popularizaciji znanosti te bavljenje znanošću dobiva vrlo pozitivan odjek u društvu. Znanstvenici bivaju prepoznati radi svojih postignuća koja ostvaruju u suradnji s kolegama koje često osobno nisu upoznali. Bitan faktor u popularizaciji znanosti kod mlađih uzrasta igra i koncept cjeloživotnog učenja koji se nameće kao neminovan, a znanost kao jedna velika igra u kojoj je cilj postići napredak zajedno s drugima. Rezultat je boljitak za čovječanstvo i osobno zadovoljstvo. Ukoliko novi naraštaji budu na taj način shvaćali znanost, preostaje nam samo zaželjeti im da se nikada ne prestanu igrati.

7. Literatura

[1] Groš S., Kalafatić Z., Šegvić S., Uvod u programski jezik Python, http://www.ieee.hr/_download/repository/Skriptni_3_Python%5B2%5D.pdf, 10.1.2016.

[2] Essert M., Python – osnove, http://www.mathos.unios.hr/racnet/Files/Essert-Python.pdf, 10.01.2016.

[3] Scott Shell M., Numpy, http://www.engr.ucsb.edu/~shell/che210d/numpy.pdf , 12.01.2016.

[4] Jones E., Oliphant E., Peterson P., SciPy: Open Source Scientific Tools for Python, (2001.), http://scipy.github.io/old-wiki/pages/History_of_SciPy , 12.01.2016.

[5] Više autora, SciPy.org, http://www.scipy.org , 12.01.2016.

[6] Lorica B., IPython a unified environment for interactive dana analysis, http://radar.oreilly.com/2014/01/ipython-a-unified-environment-for-interactive-data-analysis.html, 29.01.2016.

[7] Perez F., The IPython notebook: a historical retrospective, http://blog.fperez.org/2012/01/ipython-notebook-historical.html, 29.01.2016.

[8] Ingargiola A., What is Jupyter Notebook, http://jupyter-notebook-beginner-guide.readthedocs.org/en/latest/what_is_jupyter.html,14.01.2016.

[9] Bezanson J., Edelman A., Karpinski S., Shah V. B., Julia: A fresh approach to numerical computing, http://arxiv.org/abs/1411.1607, 27.01.2016.

[10] Colvin S., Julia ByExample, http://samuelcolvin.github.io/JuliaByExample/, 27.01.2016.

[11] Više autora, The R Project for Statistical Computing, http://www.r-project.org, 29.01.2016.

[12] Hunter J.D., Matplotlib, http://matplotlib.org/, 29.01.2016.

[13] Straub B., Chacon S., Pro Git, http://git-scm.com/book/en/v2, 03.02.2016.

[14] Todorović A., Uvod u Github, http://www.linuxzasve.com/uvod-u-github, 03.02.2016.

[15] Krajina T., Uvod u Git, http://tkrajina.github.io/uvod-u-git/git.pdf, 04.02.2016.

[16] Cannon B., We will be moving to GitHub, https://mail.python.org/pipermail/core-workflow/2016-January/000345.html, 04.02.2016.

[17] Više autora, Astropy.org, http://www.astropy.org, 21.02.2016.

[18] The Astropy Collaboration, Robitaille, T. P. et al, Astropy: A Community Python Package for Astronomy, http://arxiv.org/pdf/1307.6212v1.pdf , 17.02.2016.

[19] Berriman B., Astropy: A Community Python Package for Astronomy, https://astrocompute.wordpress.com/2013/08/26/astropy-a-community-python-package-for-astronomy/, 17.02.2016.

[20] Tollerud E., The Astropy Project: A "Selfherding cats" Development Model, http://basedonthe.com/cracker/1426091432_6b7826e3da/tollerud_aasjan2014.pdf, 17.02.2016.

[21] Jörg D., Making Your Python Code Looks Like Physics Equations, http://joergdietrich.github.io/python-physics-equations.html, 21.02.2016.

[22] Deil C., Astropy tutorials – Ipython notebooks showing Astropy, http://gammapy.github.io/astropy-tutorials/, 21.02.2016.

[23] Robitaille T., MPIK Astropy Workshop, http://astropy4mpik.readthedocs.org/en/latest/index.html, 21.02.2016.

[24] Ginsburg A., Astroquery, https://astroquery.readthedocs.org, 27.02.2016.

[25] SIMBAD astronomical database, http://simbad.u-strasbg.fr/simbad/, 27.02.2016.

[26] Plotner T., Messier 64, http://www.universetoday.com/37593/messier-64/, 28.02.2016.

[27] Plotner T., Messier 27, http://www.universetoday.com/33035/messier-27/, 28.02.2016.

[28] Ochsenbein F., Bauer P., Marcout J., The VizieR database of astronomical catalouges, Astronomy & Aastrophysics Supplement Series 143, 23 (2000)

[29] Cohen M., et al, The peculiar object HD 44179 («The Red Rectangle»), The Astrophysical Journal 196, 179 (1975)

[30] Borrell Fontelles J., Enestam J.E., Recommendation of the European Parliament and of the Council on key competences for lifelong learning 2006/962/EC, http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32006H0962&from=EN, 04.03.2016.

[31] Leko J., Zakon o Hrvatskom kvalifikacijskom okviru, Narodne novine, http://www.zakon.hr/z/566/Zakon-o-Hrvatskom-kvalifikacijskom-okviru, 04.03.2016.

[32] Jerbić - Zorc, G., Metodika nastave informatike/tehnike akademska godina 2015/2016., http://www.phy.pmf.unizg.hr/~gorjana/nastava/Informatika/nastavni materijali/pred_2015.pdf , 04.03.2015.