



Univerzitet u Novom Sadu
Prirodno - matematički fakultet,
Departman za fiziku

UPOTREBA PROGRAMSKI JEZIK PYTHON U ANALIZI METEOROLOŠKIH PODATAKA

-MASTER RAD-

Mentor: Prof.dr Ilija Arsenić

Kandidat: Martin Petraš

Novi Sad, 2018.

Sadržaj

1	Korišćeni alati	3
1.1	Python	3
1.1.1	Korišćenje biblioteka	3
1.1.2	Scipy	4
1.1.3	Numpy	4
1.1.4	Pandas	4
1.1.5	Matplotlib	5
1.1.6	MetPy	5
1.2	Django	6
1.2.1	Django arhitektura	6
2	Postupak za instaliranje	8
2.1	Instaliranje Python3 verzije	8
2.2	Podešavanje virtuelnog okruženja	9
2.3	Instaliranje Djanga	10

Zahvalnica:

mami,tati,.....

Uvod

Predmet ovog rada je izrada web aplikacije u Django razvojnom okruženju za analizu meteoroloških podataka. Web aplikacija koristi podatke sa Carpatoclim projekta, projekat koji je prikupljao meteorološke podatke u vremenskom periodu od 1961-2010 godine. Obuhvaćena oblast, kao što i samo ime projekta kaže, je oblast pružanja planinskog lanca Karpatskih planina. Aplikacija će korisniku omogućiti prikaz podataka u vizuelnom grafičkom obliku ili u vidu izlazne dadoteke sa podacima za navedeni vremenski interval.

1 Korišćeni alati

1.1 Python

Python je dinamički i objektno orijentisan programski jezik. Spada u interpreterske programske jezike visokog nivoa. Nastao je krajem devedesetih godina prošlog veka i njegov autor je Gvido van Rosum. Broj funkcija u samom jeziku je skroman, pa zahteva relativno malo uloženog vremena i napora kako bi se napravio prvi programi. Pythonova sintaksa je dizajnirana da bude čitljiva i jednostavna, što ga čini idealnim nastavnim jezikom i omogućava početnicima brzo napredovanje. Programeri provode više vremena razmišljajući o problemu koji pokušavaju da reše, a manje vremena razmišljaju o kompleksnosti jezika. Python se može izvoditi na svim važnijim operativnim sistemima (Windows, Linux/Unix, OS X), na Linux i OS X sistemima nije potrebna posebna instalacija. Za pokretanje Python programa potreban je Python interpreter, koji je pisan u programskom jeziku C. Najosnovnije korišćenje Pythona je kao jezik skriptovanja i automatizacije, koristi se za nauku o podacima i mašinsko učenje, za web usluge, metaprogramiranje. Njegova popularnost, sem čiste i pregledne sintakse, ogleda se i u velikom broju standardnih biblioteka koje su najčešće pisane u C i samom Pythonu. Što se tiče nedostataka istakli bi njegovu brzinu, međutim brzina kojom se može napisati funkcionalan program je daleko brži nego u nekom drugom jeziku.

1.1.1 Korišćenje biblioteka

Mnoge python funkcije su sadržane u specijalizovanim bibliotekama, tzv. modulima. Učitavanje modula se postiže naredbom *import*. Ova mogućnost je pythonu donela popularnost jer postoje mnogi moduli koji jako pomažu prilikom rada. Python je svojom popularnošću stekao pri izradi veb programa međutim usavršavanjem podrške za dodatne module otvorilo je vrata pythonu i u drugim oblastima. Moduli poput *numpy* i *pandas*, koji se koriste za analizu i vizuelno prikazivanje podataka, su python svrstali uz rame sa ostalim kako komercijalnim programima tako i sa programima otvorenog koda kao što su R, MATLAB, SAS, Stata i ostali. Kada pogledamo da python spada u besplatni programski jezik, njegova popularnost nije slučajnost. Jedan deo ove popularnosti u obradi podataka pripada i laka integracija sa C, C++ i FORTRAN kodom. Pošto su moduli jako značajni za python istaći ćemo neke od najznačajnijih. Samo da napomene, sve nabrojane mogućnosti koje pružaju biblioteke je moguće izvesti koristeći isključivo samo Pythonov kod. Prednost biblioteka je u efikasnosti i lakoći kojim se postižu isti rezultati, uz manje utrošenog vremena. Jedna od značajnih biblioteka otvorenog koda je i SciKit. Ona sadrži

različite klase objekata za klasifikaciju, algoritme mašinskog učenja (engl. *Machine Learning*). U SciKit paketu se nalaze NumPy i SciPy biblioteke koje sadrže klase za razne standardne matematičke i numeričke funkcije.

1.1.2 Scipy

Scipy (engl. *Scientific Python*) proširuje funkcionalnost Numpy-a sa značajnom zbirkom algoritama za Fourijeovu transformaciju, regresiju i druge matematičke tehnike. Sadrži neke od sledećih modula :

- `scipy.integrate` - integracija funkcija
- `scipy.special` - specijalne funkcije
- `scipy.optimize` - optimizacija
- `scipy.interpolate` - interpolacija

Nama od značaja će biti modul za interpolaciju.

1.1.3 Numpy

NumPy (*Numerical Python*) predstavlja fundamentalnu open source Python biblioteku kada su obzir uzmu numerički proračuni. Sadrži matematičke funkcije zadužene za operaciju na podacima, koje se veoma brzo i efikasno izvršavaju. Poziva se komandom :

```
import numpy as np
```

Za numeričke proračune, nizovi kod NumPy su dosta efikasniji prilikom njihove manipulacije nego bilo koja druga izgrađena struktura unutar Pythona. Biblioteka je pisana u C-u i Fortranu.

1.1.4 Pandas

Pandas je dizajniran kako bi se ubrzao rad sa strukturiranim ili tabelarnim podacima i kao takav, postao jako moćna i produktivna alatka za analizu podataka. Ono što krasi ovu biblioteku je i velika podrška od strane zajednice, njen aktivna razvoj, odličan rad sa ostalim bibliotekama, izgrađena je na osnovama *numpy* pa je takođe krasi brzina. Neke od mogućnosti koje pruža ova biblioteka su:

- ulaz-izlaz podataka u različitim formatima (csv, txt, SQL...)
- indeksiranje, sortiranje, rangiranje

- čišćenje podataka
- grupisanje
- vizuelizacija

Sačinjenja je od dve strukture podataka, *dataframe* i *series*. *Series* predstavlja jednodimenzioni objekat sačinjen od tabele sa vrednostima i njihovim indeksima. Skraćenjica za *pandas* je *pd*, poziva se korišćenjem komande:

```
import pandas as pd
```

1.1.5 Matplotlib

Najviše korišćena i najstarija biblioteka koja se koristi za vizuelno predstavljanje podataka. U kombinaciji sa *numpy* i *pandas* čine jako moćne alate, koji se mogu nositi sa komercijalnim i dosta skupim alatima kao što su *Matlab* i *Mathematica*. Mana ove biblioteke je potreba za pisanjem većeg broja linija koda kako bi se dobila naprednija vizuelizacija podataka. Potreba za povećanim kvalitetom vizuelizacije, nastale su savremenije biblioteke koje se u većoj ili manjoj meri oslanjaju na *matplotlib*. Jedna od takvih biblioteka je *seaborn* čiji je fokus na atraktivnijoj izradi statističkih grafika. *Seaborn* se potpuno oslanja na *matplotlib*. Naveo bih još i *bokeh* i *altair* koje nisu zavisne od *matplotlib*, a takođe se koriste za vizuelizaciju podataka.

1.1.6 MetPy

Metpy predstavlja kolekciju alata u Pythonu za pregled, vizuilizaciju i proračuna meteoroloških podataka. Paket je nastao od strane *Unidata* udruženja, koja se bavi izradom alata za naučne svrhe. MetPy paket sadrži dosta opcija, koristi se za vizelizaciju radarskih slika, rad sa netCDF podacima, vizuelizaciju radio-sondažnih merenja, vizuelizaciju meteoroloških podataka i ostalo. Mi ćemo koristiti ovu biblioteku u svrhu interpolacije polja temperature. Za instaliranje MetPy u terminal kucamo:

```
conda install -c -conda-forge metpy
```

Instaliranjem ovog paketa dobijamo razne alate, nama će najviše značiti mogućnost interpolacije za polje temperature kao i negove vizuelizacije. Za vizuilizaciju se koristi *Cartopy*, Pythonov paket za crtanje mapa.

1.2 Django

Nagla ekspanzija interneta praćenja je i povećanjem popularnosti web aplikacija. Google, Facebook, YouTube su samo od nekih popularnih web aplikacija koje čine svakodnevnicu pretraživanja. Web aplikacija je u stvari program, kojim pristupamo pomoću internet pretraživača(browser). Ovakav pristup omogućava veću bezbednost, sigurnost bez potrebu za instaliranjem na operativni sistem. Sve što vam je potrebno kako bi pristupili programu je uređaj koji ima pristup internetu. Usled spomenutog naglog rasta popularnosti web aplikacija, javila je potreba za brzim i automatizovanim načinom njihove izrade. Tako je nastao Django, web framework napisan u Python-u i služi za izradu web stranica i aplikacija.

Razvijen je od 2003. godine od strane novinara, koji su zahtevali brz razvoj web stranica. Kao slobodni softver(eng. *open source*) objavljen je 2005. godine, kada je i dobio ime Django po jazz gitaristi Djangu Reinhardt. Od 2008. godine za razvoj Django frameworka zadužena je *Django Software Foundation*. Django je zapravo skup alata i biblioteka koje programerima web aplikacija olakšavaju izradu i brigu o povezivanju svih njenih delova. Dizajniran je kako bi se skratilo vreme potrebno za izradu web aplikacije, držeći se strogih zahteva iskusnih web programera koji su ga stvorili. Web programiranje zahteva ponavljanje isti zadataka kao što su :

- korisnička prava
- registracija korisnika
- konfigurisanje URL-ova
- validacija unesenih podataka
- administracija sajta

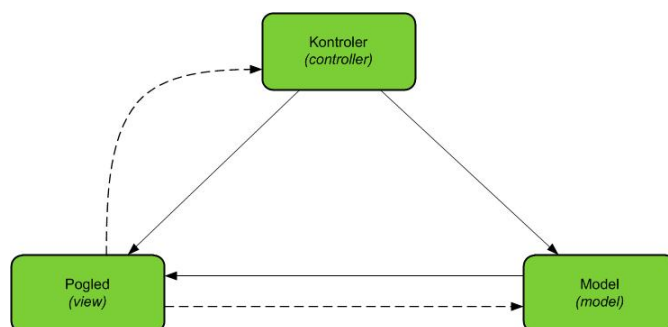
Pošto se Django bazira na Python-u pisanje ovakvog koda sadrži nekoliko linija. U drugim programskim jezicima takvi zadaci predstavljaju mukotrpan proces zahtevajući dosta vremena.

1.2.1 Django arhitektura

Kao što je rečeno, Django služi za brzu i jednostavnu izradu web stranica i aplikacija. Ovu osobinu mu omogućava arhitektura MVC(*model-view-controller*) na kojoj se bazira, i koja razdvaja logiku same aplikacije od prikazanog dela aplikacije. Možemo je podeliti na tri dela:

- Model

- View (Pogled)
- Controller(Kontroler)



Slika 1. Koncept modela MVC

Pune linije predstavljaju vezu izmedju navedenih elemenata za razmenu podataka, dok isprekidane linije predstavljaju vezu kod koje neki drugi objekt vrši tu funkciju. Django koristi drugačije nazive elemenata MVC-a. Tako *view* predstavlja kontroler, a *template* (šablon) predstavlja *view* iz MVC-a pa implementaciju MVC-a možem za Django zvati kao MTV (*model-template-view*). Ova promena je objašnjena na takav način što kod Djanga pogled predstavlja podatke koji se prezentuju dok šablon način na koji se podaci prikazuju.

Model je zadužen za komunikaciju sa bazom podataka. Model je u Django okruženju klasa (engl. *class*) i određuje varijable i metode pridružene određenim tipovima podata. Pridružene varijable predstavljaju kolone u tablici dok metode definišu relacije između varijabli. Ulog modela je da dohvati tražene podatke i prosledi ih ka pogledu. Model nema informacija o šablonima i funkcijama izvedenih u pogledima.

Uloga pogleda je prikazivanje podataka dohvaćenih iz baze podataka u internet pretraživaču. Priliko izrade web stranice, svaka kreirana aplikacija ima svoj pogled. Pogled u stvari sačinjavaju funkcije u Python programskom jeziku. Ove funkcije imaju ulogu prenosnika podataka koji će biti prikazani.

Pregled je HTML stranica s dodatnim strukturama koje omogućavaju prikaz podataka koji su poslani od pogleda. Uloga pregleda je da sadržaj poslat od strane pogleda ugradi u HTML kod, koji će se prikazati u internet pretraživaču.

2 Postupak za instaliranje

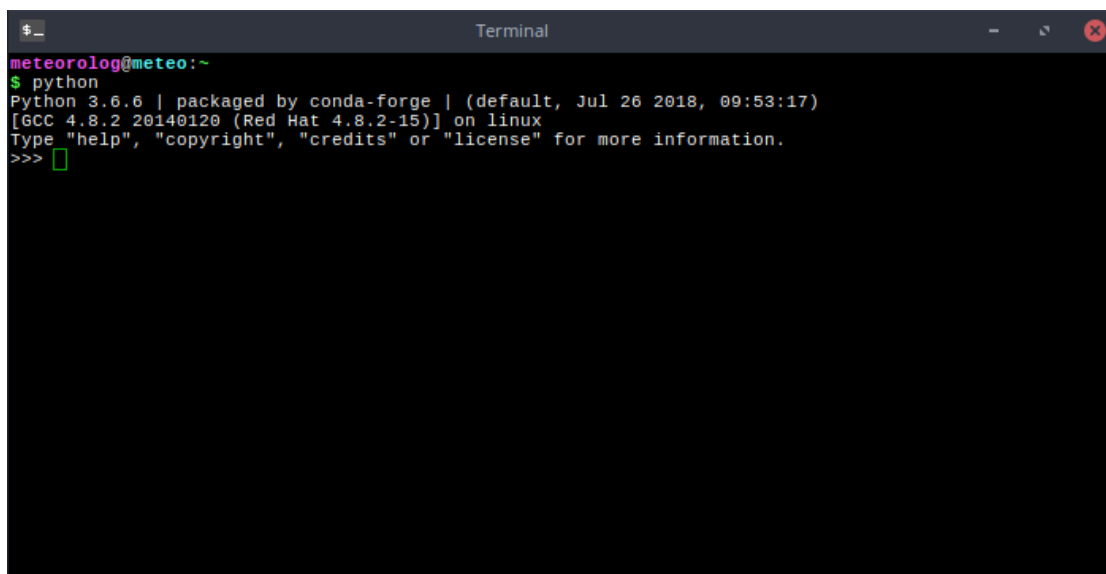
2.1 Instaliranje Python3 verzije

Django je pisan u Python-u i kako bi ga koristili potrebno je imati instaliran Python. Python kao programski jezik postoji za sve tri popularne sistemske platforme Windows, Linux i OSX. Ovde će biti prikazan postupak instaliranja potrebnih paketa za rad pod Linux operativnim sistemom. Gotov sve distribucije linuxa dolaze sa instaliranim Python-om, razlika može biti u verziji. Postoje dve verzije, starija 2.7 verzija koja je podržana do kraja 2020. godine, i novija verzija 3.6. U radu je korišćena verzija 3.5. Proveru verzije radimo tako što otvorimo terminal i kucamo sledeću komandu :

```
python3 --version
```

Ukoliko je Python verzije 3.6 instaliran, pojavi se sledeće

```
Python 3.6.6
```



```
meteorolog@meteo:~  
$ python  
Python 3.6.6 | packaged by conda-forge | (default, Jul 26 2018, 09:53:17)  
[GCC 4.8.2 20140120 (Red Hat 4.8.2-15)] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 
```

Slika 2. Python verzije 3.6 pokrenut iz terminala

Ukoliko nemate instaliran Python verzije 3.6, možete ga instalirati :

```
sudo apt install python3.6  
sudo dnf install python3  
sudo zypper install python3
```

gde je prva komanda za Debian, Fedoru i openSuse redom.

2.2 Podešavanje virtuelnog okruženja

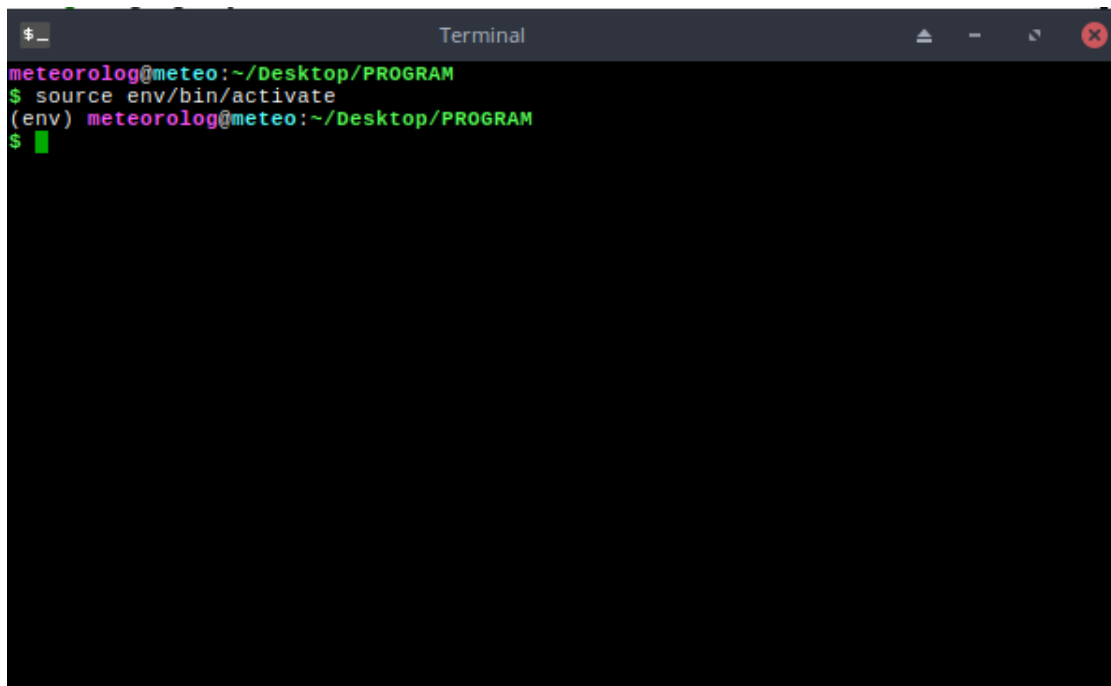
Pre nego što instaliramo Django pokazaćemo kako instalirati veoma zahvalnu alatku, koja nam omogućava održavanje čistoće paketa na sistemu. Virtuelno okruženje (engl. *virtual environment*) izoluje projekte jedan od drugog, držeći pakete vezane za svaki projekat zasebno. Kako bi kreirali virtuelno okruženje zvano *env*, u direktorijum u kome smo započeli projekat, u terminal kucamo :

```
python3 -m virtualenv env
```

[language=bash] Kada ovo odradimo, u dokumentu se pojavi dokument *env* u koji će se instalirati svi paketi koje u buduću budemo koristili. Kada ovo sve odradimo, ostalo nam je još da ovo virtuelno okruženje aktiviramo. Aktiviranje virtuelno okruženja radimo na sledeći način :

```
source env/bin/activate
```

nako čega se u terminalu pojavljuje nastavak (*env*). Ovaj nastavak nam potvrđuje da je virtuelno okruženje aktivirano. Kako bi zatvorili virtuelno okruženje, u terminal kucamo komandu *deactivate*.

A screenshot of a terminal window titled "Terminal". The prompt is "meteorolog@meteo:~/Desktop/PROGRAM". The user enters the command "\$ source env/bin/activate". The prompt changes to "(env) meteorolog@meteo:~/Desktop/PROGRAM". The user then enters a new command "\$" followed by a cursor, and the prompt returns to the original state "meteorolog@meteo:~/Desktop/PROGRAM".

```
meteorolog@meteo:~/Desktop/PROGRAM
$ source env/bin/activate
(env) meteorolog@meteo:~/Desktop/PROGRAM
$ █
meteorolog@meteo:~/Desktop/PROGRAM
$ █
```

Slika 3. Postupak aktiviranje virtuelnog okruženja

2.3 Instaliranje Djanga

Kada smo podesili virtuelno okruženje možemo instalirati Django. Instaliranje radimo pomoću *Python Package Manager* - *pip*. Prvo uradimo nadogradnju pip paketa, kako bi imali poslednju verziju. To radimo na sledeći način :

```
(env)$ python3 -m pip install --upgrade pip
```

Kada smo uradili nadogradnju pip-a, Django instaliramo :

```
pip install Django
```

Kada ovo odradimo možemo uspešno pokrenuti naš prvi Django projekat.