Priprema meteoroloških prognostičkih karata u programskom jeziku Python

Šterpin, Alen

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:217:621678

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-07-17**

Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Alen Šterpin

PRIPREMA METEOROLOŠKIH PROGNOSTIČKIH KARATA U PROGRAMSKOM JEZIKU PYTHON

Diplomski rad

Zagreb, 2016.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

SMJER: FIZIKA I INFORMATIKA, NASTAVNIČKI

Alen Šterpin

Diplomski rad

Priprema meteoroloških prognostičkih karata u programskom jeziku Python

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

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: _____

Zagreb, 2016.

Sažetak

Programski jezik Python postaje sve značajniji za obradu i prikaz podataka u znanosti. Pored velikih mogućnosti koje ima Python, mogu se koristiti paketi kao što su Numpy, Scipy i Matplotlib za numerički rad i vizualizaciju rezultata. Dodatni paketi koji su prilagođeni za rad u određenom užem znanstvenom području se također koriste. U ovom diplomskom radu primjenjuje se Python te dodatne biblioteke za analizu i prikaz meteoroloških podataka. Specijalno, napisani su i opisani programi za prikaz osnovnih meteoroloških elemenata u budućnosti, kao što su temperatura zraka, tlak zraka, smjer i brzina vjetra, ukupna naoblaka te količina oborine. Prognostički model koji se prikazuje na kartama je GFS kojeg razvija NOAA. Podaci su ucrtani na karti Hrvatske i dostupni su u formatu pogodnom za prikazivanje na web stranicama. Te meteorološke prognostičke karte dostupne su u intervalima od svaka tri sata, za sljedećih 10 dana, na internet stranicama udruge Istramet: `www.istramet.hr`.

Preparation of the weather forecast maps in programming language Python

Abstract

The programming language Python becomes increasingly important for analysis and display of scientific data. In addition to the big capabilities of Python, packages such as Numpy, Scipy and Matplotlib can be used for a numerical work and visualization of results. Additional packages are applied in various scientific fields. This thesis uses Python and additional packages for analysis and display of meteorological data. In particular, programs for display of basic meteorological parameters, such as air temperature, air pressure, direction and wind speed, total cloudness, as well as precipitations, are written and described here. Forecast model used for display weather forecast maps is GFS developed by NOAA. Data are displayed on the map of Croatia, suitable for display on web pages. These weather forecast maps are available at an interval of every three hours, through the next 10 days, on webpages of the association Istramet: `www.istramet.hr`.

Sadržaj

1 Uvod **1**

2 Osnove meteorološke prognoze **2**

3 Python i dodatne biblioteke **7**
3.1 Programski jezik Python 7
3.2 Numpy 10
3.3 Scipy 13
3.4 Matplotlib 16
3.5 Basemap 18
3.6 Pygrib 27
3.7 Itertools 30

4 Priprema meteoroloških prognostičkih karata **33**
4.1 GFS prognostički model 33
4.2 Učitavanje modula i podataka 35
4.3 Temperatura zraka 37
4.4 Tlak zraka 43
4.5 Smjer i brzina vjetra 45
4.6 Ukupna naoblaka 47
4.7 Količina oborine 49

5 Zaključak **51**

Dodatak **53**

A Meteorologija u školama **53**

1 Uvod

Razvoj računala i informatike omogućio je brzi razvoj znanosti. Računala nam pružaju brzu obradu i pregled podataka, rješavanje jednadžbi i drugih matematičkih problema. Posebno snažna računala, tzv. superračunala, najčešće se koriste u specifičnim znanostima i područjima gdje je potrebna izuzetno velika brzina obrade ogromne količine podataka. Jedna od takvih znanosti je meteorologija.

Možemo predočiti Zemlju kao tijelo na kojem se nalazi mnoštvo točaka smještenih na različitm geografskim širinama i dužinama, nadmorskim visinama i različitim oblicima reljefa, odnosa mora i kopna, itd., koje je obavijeno plinovitim omotačem atmosferom. Svaka točka atmosfere u nekom određenom trenutku ima neko stanje koje možemo opisati meteorološkim elementima. U svakoj takvoj točki termometar će pokazivati drukčiju temperaturu, drukčiji će biti iznos relativne vlažnosti, tlaka zraka, vjetra, a i količina oborina koja je pala u posljednih 24 sata također će se uvelike razlikovati od točke do točke na Zemljinoj površini. Vidimo da već do sada imamo mnoštvo informacija koje treba obraditi i uzeti u obzir sve specifičnosti svake pojedine točke. Takvim svakodnevnim promatranjem prizemnih atmosferskih podataka i bilježenjem vremena u duljim vremenskim intervalima, a minimalno 30 godina, stvara se velika baza podataka koju treba obraditi da bi na kraju opisali klimu neke lokacije.

Čovjek je oduvijek pratio i bilježio vremenske pojave i pokušao uočiti nekakvu pravilnost kako bi mogao unaprijed predvidjeti događaje. Pri tome, vrlo je značajan problem predviđanja vremena. Danas vremensku prognozu dobivamo koristeći aktualne i povijesne meteorološke podatke na Zemlji.

Cilj ovog rada je izrada meteoroloških prognostičkih karata koristeći programski jezik Python i njegove dodatne pakete kao pomoć pri čitanju podataka i njihovoj vizualizaciji. Za izradu prognostičkih karata koristit će se izračuni globalnog prognostičkog modela.

U drugom poglavlju ovog diplomskog rada opisane su osnove meteorološke prognoze. Nakon kratkog povijesnog pregleda, prikazan je sustav jednadžbi pomoću kojih možemo numerički opisati stanje u atmosferi. Ukratko su opisane i osnovne postavke prognostičkih modela.

U trećem poglavlju obrađen je programski jezik Python i sve njegove biblioteke koje će biti potrebne za crtanje prognostičkih karata.

Četvrto poglavlje konkretizira problematiku diplomskog rada. Koristeći podatke prognostičkog modela i programski jezik Python obrađeno je i prikazano nekoliko primjera osnovnih prognostičkih karata za područje Hrvatske.

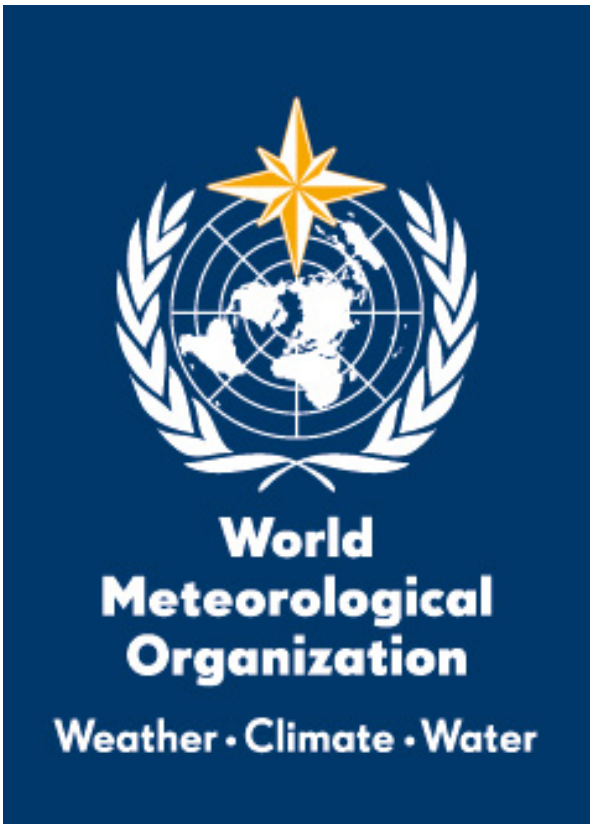
2 Osnove meteorološke prognoze

U početku je čovjek pokušavao predvidjeti vrijeme koristeći isključivo svoje osobno iskustvo. To je bilo posebice izraženo u pomorstvu, pošto je plovidba morima i oceanima uvelike ovisila o vremenskim (ne)prilikama. Nepovoljne vremenske prilike na moru često su rezultirale ljudskim žrtvama i potonućem brodova. Još prije postojanja organiziranih prognoza vremena, ribari su uz obalu predviđali vrijeme koristeći lokalne predznake vremena. Tako je primjerice poznato da nastanak oblačne "kape" na obalnim planinama (Učka, Velebit, Biokovo) označava skori početak puhanja bure u podnožju planina. Nadalje, crvenkasto nebo prilikom zalaska Sunca najvjerojatnije je uvod u sunčani sutrašnji dan, dok je crvenkasto nebo ujutro uvod u pogoršanje vremena. "Sunce u kaladu - Bog će dat pljukadu" označava da kad Sunce navečer zalazi u tmasti oblak, valja uskoro očekivati kišu [1]. Ovi i drugi predznaci i prognoze formirani su na temelju lokalnog praćenja vremena kada su ljudi cijelog dana radili na poljima i imali vizualan uvid u vanjske uvjete. U prošlosti su ljudi često formirali poslovice na temelju opaženih pojava, a nerijetko su ih povezivali i s Bogom. No, crvenkasti zalazak nije baš uvijek uvod u sutrašnje stabilno vrijeme. Danas znanost ima precizna objašnjenja tih pojava, zašto se javlja crvenilo na zalasku ili izlasku Sunca i zašto se javlja oblak na planinama prije početka puhanja bure. No znanost nam danas nudi i znatno više od toga.

Razvitak meteorologije kao znanosti započinje tijekom 17. stoljeća kada su konstruirani prvi meteorološki instrumenti. Tada je osim vizualnog opažanja bilo moguće i izravno mjeriti meteorološke elemente. Meteorološki instrumenti omogućili su početak sustavnog mjerenja tih elemenata i praćenja njihovih promjena.

Snažniji razvoj meteorologija doživljava tijekom 19. stoljeća, do kada su se u mnogim mjestima već obavljala sustavna praćenja vremena i organizirale meteorološke službe. U Krimskom ratu (1854. godine) na Crnom moru nevrijeme uništava francusku i britansku flotu, a Francuz Le Verrier dokazuje da se organiziranom meteorološkom službom, uz primjenu telegrafa, to nevrijeme moglo predvidjeti [2]. U Beču je 1873. godine osnovana Međunarodna meteorološka organizacija (IMO - akronim od engl. *International Meteorological Organization*), koja je kasnije preobražena u Svjetsku meteorološku organizaciju (WMO - akronim od engl. *World Meteorological Organization*). Današnji logo Svjetske meteorološke organizacije prikazan je na Slici 2.1. S razvitkom matematike i fizike teorijski su objašnjene mnoge meteorološke pojave što je dovelo do još bržeg razvoja meteorologije u 20. stoljeću.

Događaji u atmosferi vrlo su složeni pa je potrebno izdvojiti nekoliko najvažnijih zakona. V. Bjerknes je 1904. godine izdvojio 6 temeljnih jednadžbi u kojima su uzeti u obzir: sila Zemljine teže, Coriolisova sila, sila gradijenta tlaka, zakon o neuništivosti mase, prvi zakon termodinamike i veličine koje prikazuju svojstva plina. To je 6 jednadžbi sa 6 zavisno promjenljivih veličina u atmosferi [2]. Taj se sustav jednadžbi nadopunjavao ovisno o dodatnim saznanjima i mogućnostima računanja.



Slika 2.1: Logo Svjetske meteorološke organizacije. Preuzeto s: `http://wmo.int`

Nakon višegodišnjeg računanja, 1922. godine Richarson je prvi objavio rezultate prognoze dobivene rješavanjem jednadžbi. Time su postavljeni temelji objektivne, odnosno numeričke prognoze [2]. Proces računanja bio je vrlo spor zbog nepostojanja računala, a još uvijek nisu bile poznate neke važne metode numeričkog rješavanja diferencijalnih jednadžbi. Upravo iz tog razloga sve do 60-ih i 70-ih godina dvadesetog stoljeća bila je u uporabi subjektivna prognoza u kojima je meteorolog izdavao procjene temljene na promatranju aktualnih i povijesnih podataka. Tada je već bila raširena mreža meteoroloških postaja te su se podaci prikupljali i crtali na kartama, a uspoređivanje najnovijih karata u odnosu na prethodne dovodili su meteorologe do zaključka o premještanju meteoroloških procesa. Tako su mogli predvidjeti kretanje oblačnih masa, kao i ostalih meteoroloških elemenata, ali točnost takve prognoze nije bila zadovoljavajuća.

Atmosfera se opisuje sustavom jednadžbi koje se moraju rješavati numerički. Model atmosfere prikazan je mrežom točaka i nije kontinuiran, već diskretan. Temelj opisivanja svakog prognostičkog modela je niz sačuvanja [2]: sačuvanje mase (jednadžba kontinuiteta) (2.1), sačuvanje topline (termodinamička jednadžba) (2.2), sačuvanje količine gibanja (jednadžba gibanja) (2.3), sačuvanje vode u svim agregatnim stanjima (2.4) i jednadžba stanja (idealnog) plina (2.5). Dodatno model može sadržavati i jednadžbe sačuvanja atmosferskih primjesa poput plinovitih, tekućih i

krutih sastojaka (2.6).

$$\frac{d\rho}{d} + \rho\nabla \boldsymbol{V} = 0 \tag{2.1}$$

$$\frac{1}{T}\frac{dQ}{dt} = c_p\frac{d\ln T}{dt} - R\frac{d\ln p}{dt} \tag{2.2}$$

$$\frac{d\boldsymbol{V}}{dt} = -2\boldsymbol{\Omega V} - \frac{1}{\rho}\nabla p + \boldsymbol{g} + \boldsymbol{T} \tag{2.3}$$

$$\frac{dq_n}{dt} = \frac{S_n}{\rho}; \qquad (n = 1, 2, 3) \tag{2.4}$$

$$p = \rho RT \tag{2.5}$$

$$\frac{d\mu_m}{dt} = S_m; \qquad (m = 1, 2, ..M) \tag{2.6}$$

U svim jednadžbama uočavamo totalni (potpuni) diferencijal, kojeg kad raspišemo dobivamo (2.7).

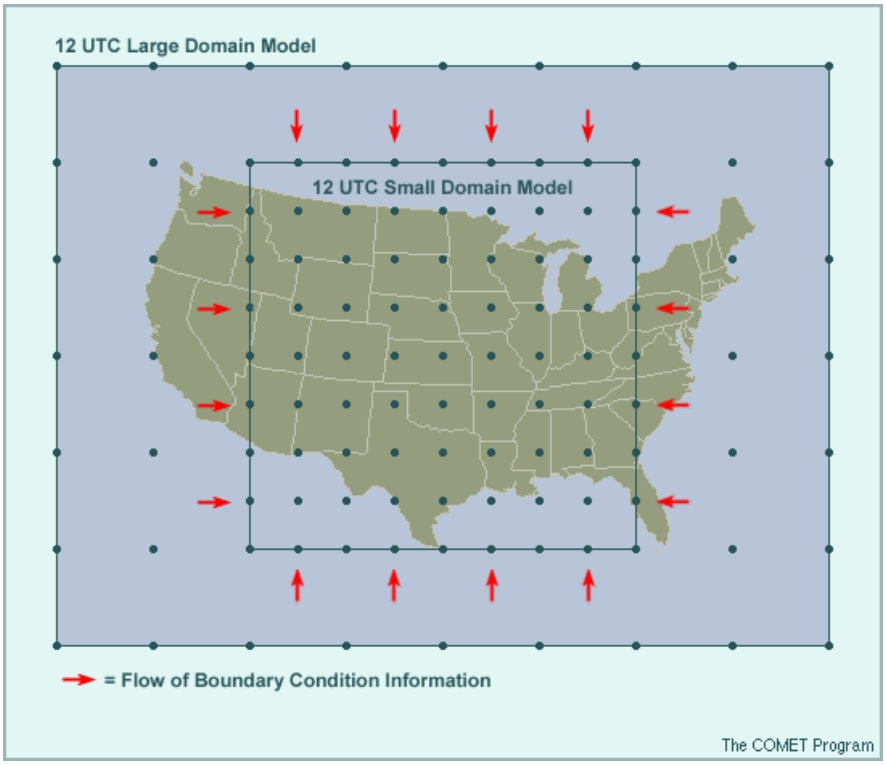
$$\frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial w} \tag{2.7}$$

U jednadžbi (2.7) prvi član označava lokalnu (mjesnu) promjenu, a ostala tri člana označavaju advekcijsku promjenu, odnosno promjenu nastalu premještanjem vremenskih procesa u određenom smjeru. Tako se primjerice temperatura zraka tijekom dana mijenja zbog grijanja Sunčevim zračenjem, i to je lokalna ili mjesna promjena. No, temperatura zraka se može tijekom dana promijeniti i zbog dolaska druge zračne mase neke druge temperature (advekcijska promjena).

Prethodne se jednadžbe prikazuju u različitim koordinatnim sustavima, što ovisi o potrebama i namjeni modela. U procesu računanja vremenske prognoze račun se provodi za jedan vremenski korak unaprijed (jedinica od 10 minuta do 1 s). Prognoza za 24 h zahtjeva 200 do 500 vremenskih koraka, nekad i znatno više, te ovako zahtjevan proces traži računala velikih mogućnosti [2]. Osim vremenskih koraka (razlučivosti), važna je i prostorna razlučivost, odnosno veličina koraka mreže. Veličina vremenskog koraka mora biti usklađena s veličinom koraka mreže kako bi se izbjegla numerička nestabilnost.

Da bi model započeo s računanjem potrebni su nam početni i rubni uvjeti. Kao početni uvjeti uzimaju se podaci motrenja i mjerenja, odnosno podaci meteoroloških elemenata u numeričkom obliku koji se preračunavaju na mrežu modela. Kod globalnih modela, odnosno modela koji izračunom pokrivaju cijeli svijet, nema rubnih uvjeta. Rubni se uvjeti koriste u ograničenim modelima koji zahvaćaju neko određeno područje, npr. područje Hrvatske ili neke regije unutar Hrvatske. Ti su rubni uvjeti

obično uzeti iz nekog drugog modela koji obuhvaća šire područje (Slika 2.2) ili globalnog modela koji obuhvaća cijeli svijet.

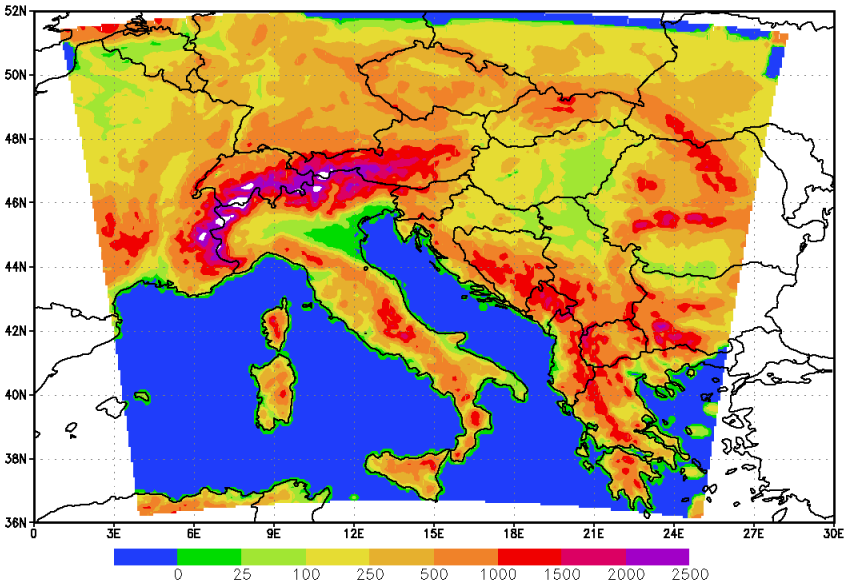


Slika 2.2: Ubacivanje rubnih uvjeta iz modela širokih razmjera u model ograničenih razmjera. Preuzeto s: `https://www.meted.ucar.edu`

Orografija, odnosno reljef značajno utječe na razvoj meteoroloških procesa unutar promatranog područja. Orografija Aladina, prognostičkog modela Državnog hidrometeorološkog zavoda prikazana je na Slici 2.3. Primjerice, znamo da temperatura zraka u troposferi u prosjeku pada s porastom visine, kao i da je vjetar uvelike ovisan o obliku reljefa, što nam je poznato po buri koja puše podno Velebita. Orografske prepreke (npr. planine) utječu i na raspodjelu količine oborina, što znamo na primjeru Gorskog kotara koji u Hrvatskoj dobiva najviše oborina. Također, vodene površine imaju drukčije djelovanje na procese u atmosferi u odnosu na kopnene. Dnevni hod temperature zraka nad morem ima znatno manju amplitudu nego nad kopnenim predjelima.

Rezolucija (udaljenost točaka mreže) modela govori nam koliko su udaljene točke u mreži za koje se izračunava prognoza. Globalni modeli danas imaju prostornu razlučivost od 50 km ili manju. Lokalni modeli, koji su ograničeni na neko manje područje, imaju gušću mrežu točaka. Primjerice, model Aladin Državnog hidrometeorološkog zavoda (Slika 2.3) ima prostornu razlučivost od 8 km, a u izradi je verzija s razlučivosti od 2 km. Gustoća točaka lokalnih modela može biti i znatno veća, tada su točke međusobno udaljene čak nekoliko desetaka metara. Gušća mreža točaka zahtjeva dulje vrijeme izračunavanja rezultata modela.

Prognostički modeli koriste se za dobivanje prognostičkih karata koje prognostičari



Slika 2.3: Orografija Aladina prostorne razlučivosti 8 km. Preuzeto s: `http://radar.dhz.hr/~moho/aladin/`

koriste prilikom izrade prognoze vremena. Uobičajeno možemo podijeliti prognoze s obzirom na razdoblje prognoziranja, područje prognoziranja te ovisno o vrsti i načinu na koji se daje prognoza.

3 Python i dodatne biblioteke

3.1 Programski jezik Python

Programski jezik Python dobio je ime po grupi komičara Monty Python. Uveo ga je Nizozemac Guido van Rossum 1989. godine kao programski jezik koji je prikladan za poučavanje programiranja i savladavanje proceduralnog i objektno orijentiranog pristupa [3]. Jezik je osmišljen tako da se može upotrebljavati bez poznavanja složenijih struktura i sintaksi, ali i s ciljem da se uz daljnje učenje izvode i vrlo složeni programi. Logo programskog jezika Python prikazan je na Slici 3.4.

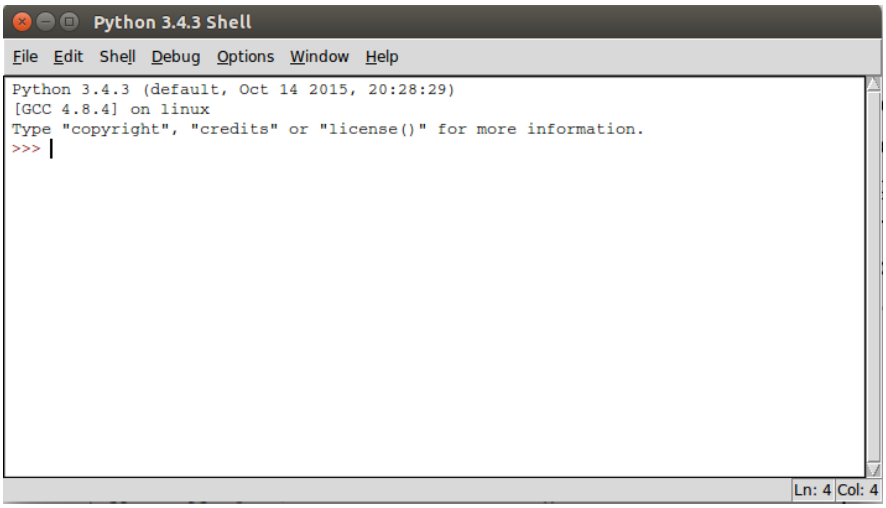
Slika 3.4: Logo programskog jezika Python. Preuzeto s: `https://www.python.org`

Proceduralni programski jezici su jezici koji se sastoje od više manjih cjelina koje nazivamo funkcijama i oblikujemo njihov rad na taj način. Objektno orijentirano programiranje je pokušaj dodatnog olakašanja izrade programa u kojem spajamo strukture podataka i funkcije u objekt. Objektno programiranje je namijenjeno izradi profesionalnih programa. Zbog toga je njihova sintaksa nešto složenija čak i prilikom izrade najjednostavnijih programa. Programski jezik Python postaje sve popularniji iz nekoliko razloga:

- potpuno je besplatan
- jednostavno se instalira na sve poznatije operativne sustave
- ima vrlo jednostavnu sintaksu (pravila pisanja)
- mogućnost korištenja u profesionalne svrhe i u znanosti
- veliki broj dodataka (modula) koji se mogu uključiti u program
- sveprisutna podrška za korištenje i učenje Pythona

Postoji više inačica Pythona. Danas je još uvijek najkorištenija inačica Python 2, iako je od 2009. dostupna inačica Python 3. Za inačicu Python 2 postoji više dodataka i gotovih rješenja nego za inačicu Python 3. Python 3 doživio je značajne izmjene, a nije kompatibilan sa starijim inačicama. Možemo reći da će u bliskoj budućnosti ipak prevladati najnovija inačica Pythona.

Svaki programski jezik dolazi sa svojim pomagalima, a skup svih pomagala nazivamo integrirano razvojno sučelje (IDE - akronim od engl. *Integrated Development Environment*) [3]. Prilikom instalacije Pythona na računalo istovremeno će se instalirati integrirano razvojno sučelje za Python koje nazivamo IDLE. Kad se IDLE pokrene, otvara se prozor koji predstavlja interaktivno sučelje s Pythonom (Slika 3.5). Na početku prozora ispisuje se verzija Pythona uz kratak opis kao pomoć za korištenje. Tri strelice na početku reda označavaju da se od korisnika traži upisivanje neke naredbe. Korisnik odgovara nekim nizom znakova i na kraju pritiskom tipke Enter na tipkovnici. S obzirom na uneseni sadržaj, Python će obaviti neku akciju. Sučelje nam omogućuje interaktivan rad i izvođenje naredbi (programa) unutar njega.



Slika 3.5: Sučelje Pythona (IDLE).

Interaktivno sučelje vrlo je prikladno za izvođenje pojedinih naredbi programa koje su dio neke veće programske cjeline. Također pogodno je za eksperimentiranje s konstruktima jezika, ili za testiranje modula od kojih se grade složeniji programi [4]. Primjer unosa jednostavnih naredbi prikazan je na Slici 3.6. Pritiskom na tipku Enter nakon svake unesene naredbe prikaže se rezultat izvršenja u novom redu. Nakon zatvaranja IDLE sučelja, sve naredbe koje smo unijeli trajno su izgubljene.

```
>>> print('Prognosticke karte')
Prognosticke karte
>>> 2+5
7
>>> 2**5
32
```

Slika 3.6: Primjer izvođenja jednostavnih naredbi u interaktivnom sučelju Pythona.

Naredbe koje želimo trajno sačuvati moramo pohraniti u datoteke koje zatim smještamo na disk. To možemo ostvariti u uređivačkom dijelu sučelja IDLE kojem

pristupamo koristeći padajući izbornik na vrhu klikom na *New File*. Otvara nam se nova prazna datoteka bez znaka upita (bez tri strelice). Nakon što unesemo program, klikom na *Save As* pohranjujemo datoteku. Datoteci pridjelimo određeno ime (naziv programa) po kojem ćemo raspoznavati naš program i dodamo sufiks *.py*. Takve datoteke nazivamo i modulima [3, 4]. Pokrećemo ih koristeći naredbu *Run* u padajućem izborniku, ili kraće pritiskom tipke *F5*. Primjer jednostavnog programa (modula) prikazan je na Slici 3.7. Nakon spremanja datoteke, naše su naredbe trajno pohranjene.

```
1 # modul za diplomski rad
2
3 # ispisujemo Prognosticke karte
4 print('Prognosticke karte')
5 # ispisujemo rezultat zbrajanja 2+5
6 print(2+5)
7 # ispisujemo rezultat potenciranja dva na petu
8 print(2**5)
```

Slika 3.7: Primjer jednostavnog programa napisanog u Pythonu.

Spomenimo da smo u prvom retku našeg programa upotrijebili simbol ljestve (engl. *hash*), koji u Pythonu označava početak komentara. Komentari nisu obavezni, ali nam pomažu prilikom programiranja, posebice prilikom izrade složenijih programa gdje služe najčešće kao podsjetnik na ono što smo napisali u tom dijelu programa. Nakon što pokrenemo program (modul), u interaktivnom dijelu ispisuje se rezultat (Slika 3.8).

```
1 >>> ========== RESTART ==========
2 >>>
3 Prognosticke karte
4 7
5 32
6 >>
```

Slika 3.8: Rezultat pokretanja modula sa Slike 3.7.

Osim što ih možemo pokrenuti u interaktivnom sučelju, module možemo učitati i u neki drugi program. Time se postiže bolja organizacija biblioteka i alata. Module učitavamo naredbom *import* i *from* u drugi program koji onda ima pristup cjelokupnom modulu. Prilikom učitavanja modula u program, nije potrebno pisati nastavak *.py*.

Kao što je ranije već navedeno, prednost Pythona je u velikom broju dodataka, od kojih je većina potpuno besplatna za korištenje. Moduli olakšavaju posao i ubrzavaju cijeli proces programiranja te se mogu jednostavno instalirati i učitati u program.

Velik broj modula dolazi službeno uz Python, a mnoge su pisali korisnici i stavili na slobodno raspolaganje [3]. U ovom diplomskom radu korišteni su neki od najpoznatijih modula, ali također i neki specifični koji su namijenjeni točno određenoj problematici. Svi korišteni moduli su opisani u sljedećim potpoglavljima.

3.2 Numpy

Numpy je jedan od besplatnih modula otvorenog koda namjenjenog za korištenje u Pythonu. Ime mu dolazi od skraćenica dviju engleskih riječi, a to su *Numeric* i *Python*. Numpy sadrži funkcije za upravljanje velikim poljima i matricama te postaje jedan od najboljih programa za manipulaciju numeričkim podacima, poput komercijalnog MatLaba [5]. Numpy možemo uključiti u naš program naredbom *import numpy* kao što je to ranije i opisano.

```
1  # ucitavamo modul numpy
2  import numpy as np
3
4  # kreiramo python listu
5  lista = [1, 2, 3, 4]
6  # ispisujemo listu
7  print('Lista:', lista)
8  # kreiramo numpy polje od python liste
9  polje = np.array(lista, float)
10 # ispisujemo numpy polje
11 print('Polje:', polje)
12 # ispisujemo tip polja
13 print('Tip polja:', type(polje))
14 # ispisujemo prva dva elementa polja
15 print('Prva dva elementa polja:', polje[:2])
16 # ispisujemo zadnji element polja
17 print('Zadnji element polja:', polje[3])
18 # mijenjamo zadnji element polja
19 polje[3] = 999
20 # ispisujemo polje nakon promjene zadnjeg elementa
21 print('Izmijenjeno polje:', polje)
```

Slika 3.9: Programske naredbe za kreiranje polja u Pythonu i nekoliko osnovnih operacija izvršenih nad njim.

Kao glavna i najvažnija mogućnost paketa Numpy ističe se rad s objektima tipa polje (engl. *array*). Polja su slična listama u Pythonu, a mogu raditi s jako velikom količinom podataka i mnogo brže nego se to ostvaruje u listama, što daje tom programskom paketu veliku prednost. Polje može biti kreirano pomoću liste, a za pristupanje elementima polja koriste se iste naredbe kao i za listu. Polja mogu biti i višedimenzionalna, stoga je programski paket prilagođen i brzom radu s matricama.

Za razliku od liste koja može sadržavati nekoliko elemenata različitih tipova (npr. float, int, itd.), Numpy polje zahtjeva da svi elementi budu istog tipa.

```
Lista: [1, 2, 3, 4]
Polje: [ 1.  2.  3.  4.]
Tip polja: <class 'numpy.ndarray'>
Prva dva elementa polja: [ 1.  2.]
Treci element polja: 4.0
Izmijenjeno polje: [   1.    2.    3.  999.]
```

Slika 3.10: Ispis programskih naredbi sa Slike 3.9.

Na Slici 3.9 prikazan je primjer korištenja polja iz paketa Numpy. Paket smo učitali naredbom *import numpy as np*, što označava da ćemo funkcije koje koristimo iz paketa pozvati s prefiksom np. To se koristi iz praktičnih razloga, kako bi se povećala preglednost i smanjila količina koda. Kreirali smo Python listu koju smo kasnije umetnuli u argument funkcije za kreiranje Numpy polja. Drugi argument funkcije je tip elemenata polja, koji je u našem slučaju postavljen na *float*. U nastavku je izvršeno nekoliko osnovnih operacija nad poljem. Lako možemo pristupati pojedinim elementima polja koristeći uglate zagrade. Znak dvotočka unutar uglatih zagrada označava "od do" elemente kojima želimo pristupiti i ispisati ih, u našem slučaju *a[:2]* ispisuje prva dva elementa polja. Valja primijetiti kako smo zadnjem elementu polja pristupili naredbom *a[3]*, iako je zadnji element polja četvrti po redu. To je iz razloga što elementi polja (i liste) počinju od nule, dakle prvi element polja ima indeks 0, drugi element polja indeks 1, itd. Pojedine elemente polja možemo lako promijeniti pridružujući tom elementu neku drugu vrijednost. Ispis programskih naredbi sa Slike 3.9 prikazan je na Slici 3.10.

Program za kreiranje dvodimenzionalnog polja prikazan je na Slici 3.11. Naredbom *polje.shape* možemo provjeriti dimenzije tog polja, dok naredba *polje.transpose()* pretvara stupce u retke, a retke u stupce (transponiranje matrice). Numpy ima vrlo razvijene i pristupačne metode obrade polja: tako možemo provjeriti koja je minimalna ili maksimalna vrijednost unutar polja, kao i sumu svih vrijednosti elemenata polja. Naredba *np.sqrt(transponirano)* vrši operaciju korijenovanja na svakom elementu transponiranog polja. Rezultat izvršenja programskih naredbi sa Slike 3.11 prikazan je na Slici 3.12.

Osim ranije spomenutih osobina i prikazanih osnovnih operacija paketa, spomenimo da Numpy ima mogućnosti obrade podataka i rada s [5]:

- složenim operacijama linearne algebre
- spajanjem polja (matrica)
- promjenama veličine polja

```
# ucitavamo modul numpy
import numpy as np

# kreiramo dvodimenzionalno polje
polje = np.array([[1, 2, 3], [4, 5, 6]], float)
# ispisujemo dvodimenzionalno polje
print('Dvodimenzionalno polje:', polje)
# ispisujemo dimenziju polja
print('Dimenzije 2D polja:', polje.shape)
# pretvaramo stupce u retke, a retke u stupce
transponirano = polje.transpose()
# ispisujemo transponirano polje
print('Transponirano polje:', transponirano)
# ispisujemo minimalnu vrijednost polja
print('Minimalna vrijednost polja:', polje.min())
# ispisujemo maksimalnu vrijednost polja
print('Maksimalna vrijednost polja:', polje.max())
# ispisujemo sumu elemenata polja
print('Suma elemenata polja:', np.sum(polje))
# ispisujemo korijenovano transponirano polje
print('Korijenovano transponirano polje:', np.sqrt(transponirano))
```

Slika 3.11: Kreiranje dvodimenzionalnog polja i nekoliko osnovnih operacija izvršenih nad njim.

```
Dvodimenzionalno polje: [[ 1.  2.  3.]
                         [ 4.  5.  6.]]
Dimenzije 2D polja: (2, 3)
Transponirano polje: [[ 1.  4.]
                      [ 2.  5.]
                      [ 3.  6.]]
Minimalna vrijednost polja: 1.0
Maksimalna vrijednost polja: 6.0
Suma elemenata polja: 21.0
Korijenovano transponirano polje: [[ 1.          2.        ]
                                   [ 1.41421356  2.23606798]
                                   [ 1.73205081  2.44948974]]
```

Slika 3.12: Rezultat izvršenja programa sa Slike 3.11.

- Fourierevim transformacijama
- naprednom statističkom obradom podataka
- posebnim tipovim podataka
- ostalim matematičkim funkcijama poput Besselovih funkcija

U ovom diplomskom radu programski paket Numpy koristi se za rad s poljima prilikom kreiranja jednostavnih jednodimenzionalnih polja, složenih dvodimenzionalnih polja, zbrajanja polja i korijenovanja elemenata polja.

3.3 Scipy

Scipy je još razvijeniji dodatak za Python, a sadržava i sve mogućnosti ranije opisanog paketa Numpy. Većina Scipy mogućnosti dostupna je učitavanjem Scipy modula koristeći naredbu *import scipy*, ali treba napomenuti da dio Scipy pod-modula zahtjeva eksplicitno učitavanje. Mnoge funkcije paketa Scipy mogu se lako implementirati u program i nude brzi pristup svim numeričkim algoritmima [5]. Prikaz nekih pod-modula Scipy paketa prikazan je u Tablici 3.1.

Modul	**Namjena modula**
scipy.constants	Sadrži mnogo matematičkih i fizikalnih konstanti.
scipy.special	Specijalne funkcije za matematiku i fiziku, poput gamma funkcija, Besselovih funkcija, paraboličnih funkcija, Kelvinovih funkcija i ostalih.
scipy.integrate	Sadrži funkcije za provedbu numeričke integracije koristeći više različitih metoda.
scipy.optimize	Optimizacija korisnički definiranih funkcija.
scipy.linalg	Sadrži veliku bazu funkcija za linearnu algebru, veću nego Numpy. Primjerice sadrži funkciju Choleskyjeve dekompozicije matrice, Schurove dekompozicije matrice i mnogo drugih matematičkih operacija za matrice.
scipy.sparse	Za rješavanje velikih rijetkih matrica (engl. *sparse matrices*)
scipy.interpolate	Sadrži funkcije za interpolaciju objekata koji sadrže diskretne numeričke podatke.
scipy.stats	Velika biblioteka za različite statističke operacije na skupu podataka.

Tablica 3.1: Prikaz nekih modula Scipy paketa i njihovih namjena.

Scipy paket najčešće se koristi za optimizaciju koda, za naprednu analizu podataka, bazu podataka ili za obradu slike. Tako Scipy posjeduje nekoliko osnovnih funkcija za rad sa slikama. Primjerice, postoji funkcija za čitanje slike s diska u Numpy polje, za pisanje Numpy polja na disk kao slika, ili za smanjivanje veličine slike. Na Slici 3.13 učitana je izvorna slika loga programskog jezika Python sa Slike 3.4 i smanjena je s 203x601 piksela na 101x300 pixela.

U prvoj naredbi na Slici 3.13 učitali smo pojedine funkcije Scipy pod-modula koristeći naredbe *from* i *import*. Te dvije naredbe koristimo kada želimo učitati samo neke dijelove određenih modula. Tako smo u ovom slučaju, iz paketa *scipy.misc*

```
# ucitavamo pojedine funkcije scipiy.misc pod-modula
from scipy.misc import imread, imsave, imresize

# ucitavamo sliku s diska u polje
logo = imread('/putanja/python.png')
# ispisujemo dimenziju originalnog loga
print ('Dimenzija originalnog loga', logo.shape)
# smanjujemo dimenziju slike na 101x300 pixela
mali_logo = imresize(logo, (101,300))
# spremamo novi logo u obliku slike na disk
imsave('/putanja/python_mali.png', mali_logo)

# ucitavamo novi logo i provjeravamo njegove dimenzije
print ('Dimenzija smanjenog loga', imread('/putanja/python_mali.
    png').shape)
```

Slika 3.13: Program za smanjivanje dimenzija loga prikazanog na Slici 3.4.

```
Dimenzija originalnog loga: (203, 601, 4)
Dimenzija smanjenog loga: (101, 300, 4)
```

Slika 3.14: Ispisane dimenzije originalnog i smanjenog loga.

Slika 3.15: Smanjeni logo Pythona nakon izvršenja programa sa slike 3.13.

učitali funkcije koje će nam biti potrebne tijekom izvođenje programa. Koristeći funkciju *imread* učitali smo sliku s diska, a pomoću *logo.shape* provjerili smo dimenzije učitane slike. Slika se prilikom učitavanja pretvara u Numpy polje, a *shape* je funkcija Numpy paketa koja nam daje informacije o dimenziji polja. Pomoću funkcije *imresize* (koja ima dva argumenta: učitanu originalnu sliku i dimenziju nove slike) smanjili smo dimenziju slike na 101x300 piksela. Funkcija *imsave* pretvorila je polje nove smanjene slike u format slike i spremila ju na disk pod nazivom *python_mali.png* (Slika 3.15).

Scipy ima još dodatnih pod-modula za obradu Slike. Spomenimo dodatak *scipy.ndimage* koji sadrži više različitih funkcija za obradu i procesiranje višedimenzionalnih slika [6]. Te funkcije možemo grupirati s obzirom na njihovu primjenu:

- filtriranje (s obzirom na os, s obzirom na vrijednost, ...)
- Fourierevo filtriranje (gausijanski ili eliptoidni Fourierov filter, ...)

- interpolacija (rotacija polja, pomak, proširivanje polja, ...)
- mjerenja (suma vrijednosti polja, pronalazak najveće vrijednosti, ...)
- morfologija (erozija elementa, popunjavanje rupa danog elementa, ...)
- ostale mogućnosti (npr. učitavanje slike)

```
# ucitavamo modul numpy
import numpy as np
# ucitavamo funkciju zoom iz paketa scipy.ndimage
from scipy.ndimage import zoom

# kreiramo polje s dva elementa
polje = np.array([1,2], float)
# ispisujemo dimenziju polja
print('Dimenzija polja:', polje.shape)
# prosirujemo polje funkcijom zoom za faktor 2
prosireno_polje = zoom(polje, 2)
# ispisujemo dimenziju novog polja
print('Dimenzija prosirenog polja:', prosireno_polje.shape)
# ispisujemo prosireno polje
print('Prosireno polje: ', prosireno_polje)
```

Slika 3.16: Program za interpolaciju i proširivanje polja za faktor 2.

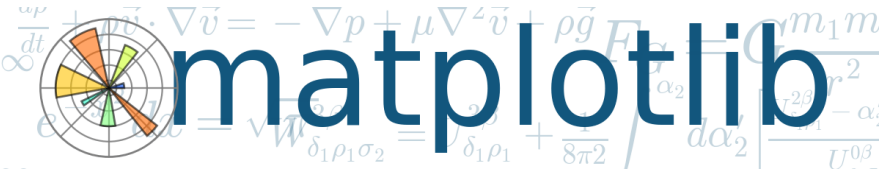
```
Dimenzija polja: (2,)
Dimenzija prosirenog polja: (4,)
Prosireno polje:  [ 1.   1.25925926  1.74074074  2.     ]
```

Slika 3.17: Ispis nakon pokretanja programa sa Slike 3.16.

U ovom diplomskom radu, radi bolje interpolacije, korištena je funkcija *zoom* podmodula *scipy.ndimage*. Ta se funkcija koristi za proširivanje polja tako da se uz svaki element polja kreira još neki određeni broj elemenata koji su ustvari samo interpolacija nastala od izvornih elemenata. Time možemo značajno povećati dimenziju polja, ovisno o faktoru proširenja kojeg definiramo. U programu na Slici 3.16, nakon učitavanja potrebnih modula, kreirali smo jednostavno polje od dva elementa i ispisali njegovu dimenziju. Funkcija *zoom* ima dva argumenta: prvi argument je naše polje, a drugi argument faktor proširivanja polja, u našem slučaju je to dva. Nakon proširivanja polja, ispisujemo njegovu dimenziju. Uočavamo da novo polje sada ima 4 elementa, odnosno prošireno je za faktor 2 te smo dobili dva nova elementa koja su nastala interpolacijom originalnih elemenata polja (Slika 3.17).

3.4 Matplotlib

Matplotlib je Python biblioteka namijenjena crtanju grafova. Glavna karakteristika je velika moć i brzina obrade podataka, kao i sama jednostavnost korištenja. Matplotlib može se koristiti u Python programima, Python ljuskama poput Matlaba, web aplikacijama i još nekim drugim grafičkim sučeljima [7]. Korisnici mogu jednostavno kreirati nacrte, histograme, stupčaste grafove, točkaste grafove i mnoge druge u svega nekoliko linija koda. Matplotlib omogućava potpunu kontrolu fonta, boja, oblika linija, osi, prikaza legendi itd. koristeći objektno orijentirano sučelje sa setom poznatih funkcija. Logo Matplotliba prikazan je na Slici 3.18.



Slika 3.18: Logo Matplotliba. Preuzeto s: `http://matplotlib.org/`

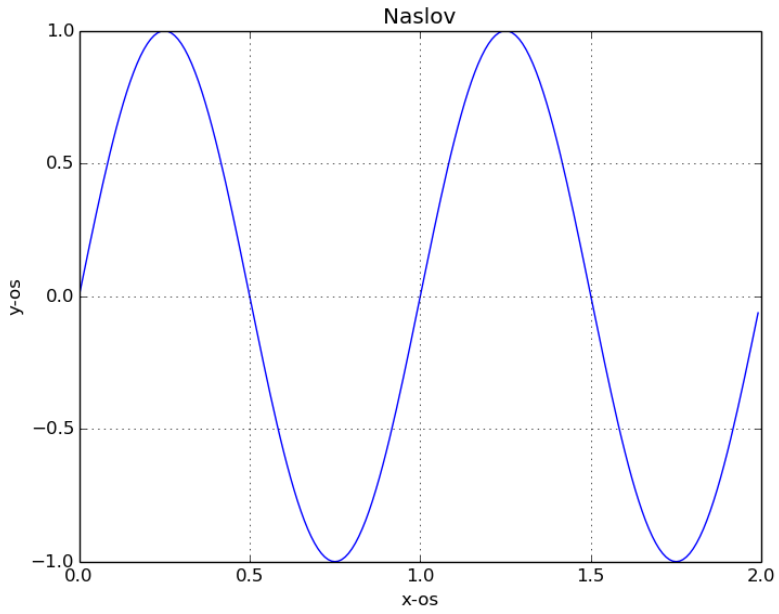
Matplotlib je besplatan program otvorenog koda. Kreirao ga je John D. Hunter koji je nažalost preminuo u ljeto 2012. godine u svojoj 44. godini života. Danas Matplotlib razvija veliki broj korisnika koji žele doprinijeti znanosti i znanstvenoj zajednici [7]. Potpuna dokumentacija za Matplotlib dostupna je na službenim web stranicama, na kojima se mogu pronaći stotine primjera korištenja koda uz brojna pitanja i odgovore. Zadnja stabilna verzija je 1.5.1, a u razvoju je nova verzija 2.0.

Primjer crtanja jednostavnog grafa koristeći Matplotlib prikazan je na Slici 3.19. Najprije smo učitali potrebne module, a to su Numpy i Matplotlibov Pyplot. Numpy koristimo za definiranje polja x od vrijednosti 0.0 do vrijednosti 2.0, u koraku od 0.01. Polje x predstavljat će nam x-os na grafu. Numpy također upotrebljavamo i za neke specijalne funkcije; tako smo definirali polje y kao sinus svakog elementa iz x polja. Time smo dobili y-os koja je ovisna o x-osi. Da bi nacrtali graf, korisimo naredbu *plt.plot(x, y)* koja kreira novi objekt u Pythonu. Korištenjem ostalih funkcija grafu dodajemo opis x-osi, opis y-osi, naslov grafa i mrežu između osi. Tekst koji želimo da se prikaže uz osi ili kao naslov grafa unesemo unutar zagrada pojedine funkcije kao što je to prikazano na Slici 3.19. Prikaz slike (grafa) dobivamo nakon upisivanja *plt.show()* što otvara novi prozor u Python interaktivnom sučelju s pripadajućim grafom (Slika 3.20).

Matplotlib posjeduje kvalitetan alat za rad s bojama. Postoji više mogućih načina kako možemo definirati našu skalu boja. Matplotlib dolazi s već nekim unaprijed definiranim skalama boja, a dio njih prikazan je na Slici 3.21. Na službenoj web stranici Matplotliba postoje detaljne upute za korištenje boja, kako unaprijed definiranih skala tako i mogućnost vlastitog definiranja skale. Kako bi Matplotlib i Python općenito bili vrlo pristupačni korisniku, omogućuje se definiranje boja na više načina. Tako možemo primjerice sivu boju definirati upisom engleskog naziva sive boje (*co-*

```
# ucitavamo Matplotlibov Pyplot modul
import matplotlib.pyplot as plt
# ucitavamo Numpy modul
import numpy as np

# definiramo vrijednosti polja x
x = np.arange(0.0, 2.0, 0.01)
# definiramo vrijednosti polja y (koje su ovisne o x)
y = np.sin(2*np.pi*x)
# kreiramo graf (objekt)
plt.plot(x, y)
# dodajemo opis x-osi
plt.xlabel('x-os')
# dodajemo opis y-osi
plt.ylabel('y-os')
# dodajemo naslov grafa
plt.title('Naslov')
# ucrtavamo mrezu izmedju osi
plt.grid(True)
# prikazujemo graf
plt.show()
```

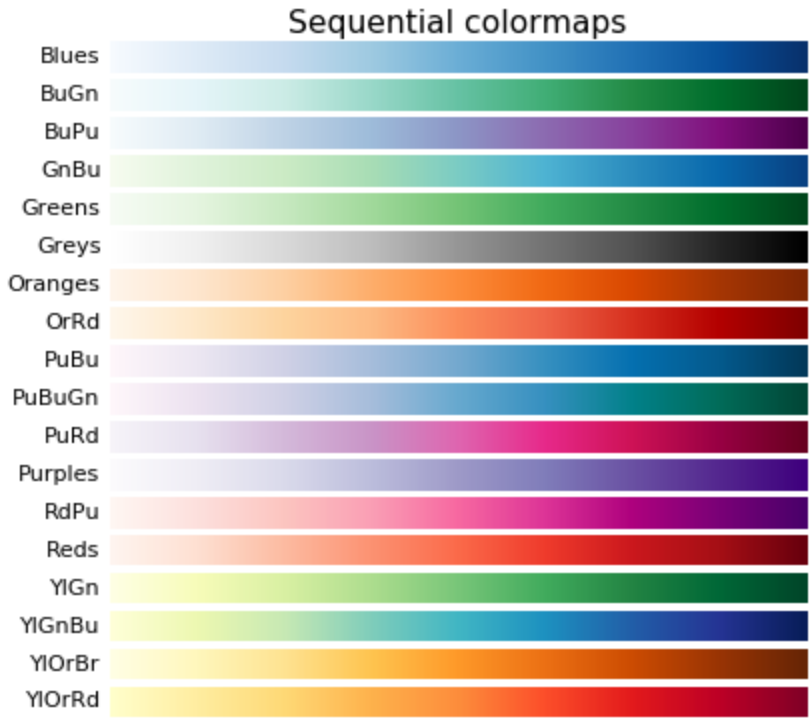
Slika 3.19: Programski kod za dobivanje jednostavnog grafa u Matplotlibu.



Slika 3.20: Graf kao rezultat izvršenog programskog koda sa Slike 3.19.

lor = 'gray'), pomoću HTML oznake sive boje (*color = '#eeefff'*), pomoću crno-bijele skale (*color = '0.75'*) ili koristeći R-G-B vrijednosti boje [7].

Matplotlib ima još nekoliko dodatnih mogućnosti (engl. *add-on toolkits*) koje se



Slika 3.21: Neke od definiranih skala boja u Matplotlibu. Preuzeto s: `http://matplotlib.org/devdocs/examples/color/colormaps_reference.html`

mogu instalirati. Dodaci povećavaju moć Matplotlib paketa, a koriste se najčešće prilikom izrade složenijih grafova, karata ili mapa, kao i za 3D crtanje [7]. U idućem poglavlju opisan je jedan od dodataka koji se koristi u diplomskom radu, odnosno predstavlja bazu u crtanju prognostičkih karata.

3.5 Basemap

Basemap je dodatak Matplotlib biblioteci i upotrebljava se za crtanje 2D podataka na kartama u Pythonu [8]. Ima slične funkcionalnosti poput paketa Matlab ili GrADS. Namijenjen je znanstvenicima, posebno oceanografima i meteorolozima. Basemap je osmislio i napisao Jeffrey Whitaker, u početku kao pomoć pri njegovom istraživanju klime i vremenske prognoze. Do onda je CDAT paket bio jedini dostupan u Pythonu za crtanje podataka na geografske karte. Kroz godine, potreba i namjena Basemap paketa se povećala i proširila u druge znanstvene discipline poput biologije, geologije i geografije.

Basemap ima mogućnost crtanja koordinata u čak 24 različite projekcije koje su dostupne unutar paketa. Neke od poznatijih projekcija su: Lambertova, Mercatorova, Hammerova i Mollweideova. Rijeke, jezera, mora i političke granice već su dostupne unutar paketa te postoje specijalne metode za njihovo crtanje. Za crtanje kontura, linija, slika, vektora i točaka koristi se Matplotlib.

Ako želimo prikazati zakrivljenu površinu Zemlje na dvodimenzionalnoj karti, moramo se poslužiti nekom od geografskih projekcija. Basemap omogućava oda-

bir 24 različite geografske projekcije, od kojih svaka ima svoje prednosti i mane. Projekcije se obično odabiru s obzirom na namjenu karata i s obzirom na Zemljinu površinu koju želimo prikazati. Neke su projekcije globalne, a neke prikazuju samo dio Zemljine površine. Geografska projekcija karte mora biti specificirana zajedno s informacijama o geografskom području koje se želi prikazati. Postoje dva osnovna načina kako možemo definirati geografsko područje koje želimo prikazati na karti: prvi je da definiramo geografsku dužinu i geografsku širinu krajnjih granica (kutova) karte koju želimo prikazati, a drugi je da definiramo geografsku dužinu i geografsku širinu centra karte (tada nema prostornog ograničenja).

Na Slici 3.22 prikazan je program za izradu karte svijeta u Hammerovoj projekciji. Zadana je geografska dužina i geografska širina središta karte.

```
# ucitavamo Basemap modul
from mpl_toolkits.basemap import Basemap
# ucitavamo Matplotlibov Pyplot modul
import matplotlib.pyplot as plt
# ucitavamo Numpy modul
import numpy as np

# kreiramo kartu u Hammerovoj projekciji
# karta je centrirana na 0 stupnjeva geo. duzine i sirine
# postavljena gruba rezolucija
m = Basemap(projection = 'hammer', lon_0 = 0,
            lat_0 = 0, resolution = 'c')

# ucrtavamo obalnu crtu
m.drawcoastlines()
# ucrtavamo boju kontinentima i jezerima
m.fillcontinents(color = 'coral', lake_color = 'aqua')
# ucrtavamo paralele
m.drawparallels(np.arange(-90.,120.,30.))
# ucrtavamo meridijane
m.drawmeridians(np.arange(0.,360.,60.))
# ucrtavamo rub karte i boju oceana
m.drawmapboundary(fill_color = 'aqua')
# ucrtavamo naslov
plt.title("Hammerova projekcija")
# prikazujemo kartu
plt.show()
```

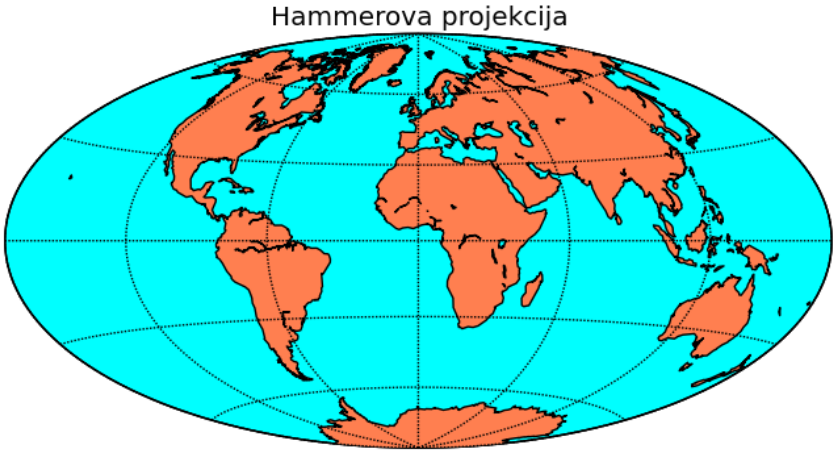
Slika 3.22: Program za izradu karte svijeta u Hammerovoj projekciji.

Na početku programa (Slika 3.22) učitali smo potrebne module, a to su Numpy, Matplotlib i Basemap. Kartu kreiramo naredbom *Basemap* koja sadrži četiri argumenta. Prvi argument je projekcija koju želimo odabrati, u našem slučaju je to Hammerova projekcija. Drugi i treći argumenti označeni s $lon_0 = 0$ i $lat_0 = 0$ označavaju

da će u središtu karte biti prikazana geografska dužina i geografska širina od 0 stupnjeva. Četvrti argument je rezolucija karte, odnosno broj detalja koje želimo prikazati na karti (u našem slučaju rezolucija je gruba, 'c'). Postoji nekoliko mogućih razina rezolucije koje možemo postaviti. Za karte gdje prikazujemo cijeli svijet nije potrebno previše detalja, stoga koristimo grubu rezoluciju, dok ćemo za neko manje određeno područje na Zemlji koristiti finiju rezoluciju. Objašnjenja ostalih naredbi korištenih u programu nalaze se u Tablici 3.2. Rezultat izvršenja programa sa Slike 3.22 prikazan je na Slici 3.23.

Naredba	**Opis**
m.drawcoastlines	Upotrebljava se za ucrtavanje obale, odnosno obalne linije.
m.fillcontinents	Ispunjava kontinente (kopnene površine Zemlje) bojom koju odaberemo. Također postoji mogućnost prepoznavanja rijeka i jezera te postavljanje njihovih boja.
m.drawparallels	Ucrtava na karti paralele, koje smo definirali koristeći Numpy polje.
m.drawmeridians	Ucrtava na karti meridijane, koje smo definirali koristeći Numpy polje.
m.drawmapboundary	Ucrtava granicu karte (obrub) i ispunjava površine koji nisu kontinenti (mora i oceane) određenom bojom.
plt.title	Ucrtava naslov na vrh karte.
plt.show	Prikazuje kreiranu kartu u obliku slike.

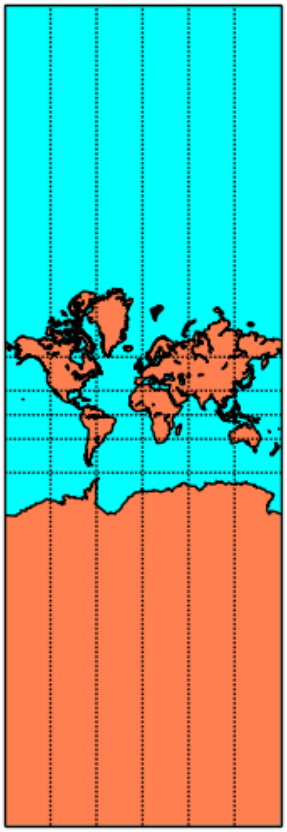
Tablica 3.2: Ostale naredbe korištene u programu na Slici 3.22.



Slika 3.23: Karta svijeta kreirana u Hammerovoj projekciji.

Ukoliko izvršimo slične programske naredbe kao na Slici 3.22, ali odaberemo Mercatorovu projekciju dobivamo Sliku 3.24. Valja primijetiti kako je za prikaz cijele

Zemlje zgodnija Hammerova projekcija, dok je Mercatorova pogodnija ako želimo prikazati neki manji dio Zemljine površine. ”Prirodniji” prikaz Mercatorove projekcije možemo dobiti tako da definiramo geografske dužine i geografske širine kutova karte (krajnjih granica) koju želimo kreirati. Time bi izbjegli problematične dijelove karte koji u Mercatorovoj projekciji čine krajnji sjeverni i južni pol. U Tablici 3.3 kratko su opisani argumenti objekta Basemap kojima definiramo rubove karte.



Slika 3.24: Karta svijeta kreirana u Mercatorovoj projekciji.

Argument	**Opis**
llcrnrlat	Geografska širina donjeg lijevog kuta karte (engl. *lower left corner*).
urcrnrlat	Geografska širina gornjeg desnog kuta karte (engl. *upper right corner*).
llcrnrlon	Geografska dužina donjeg lijevog kuta karte (engl. *lower left corner*).
urcrnrlon	Geografska dužina gornjeg desnog kuta karte (engl. *upper right corner*) geografske dužine.

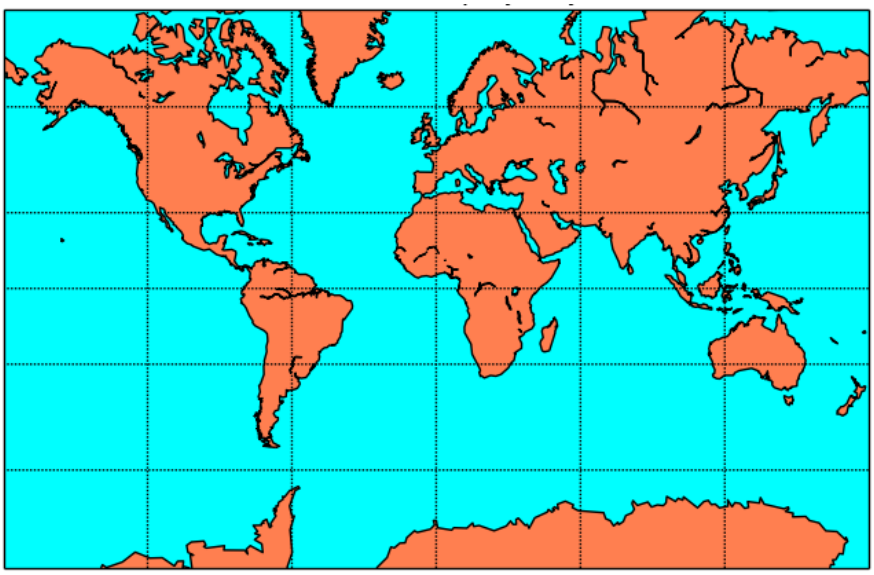
Tablica 3.3: Argumenti Basemap naredbe za definiranje kutova karte.

Izmjenjenom naredbom kao što je to prikazano na Slici 3.25 definirali smo krajnje geografske dužine i geografske širine (kutove) karte, umjesto geografskih koordinata središta karte kao do sada. Za područje karte koju želimo kreirati odabrali smo sve geografske dužine (svih 360 stupnjeva geografske dužine), ali smo ih prikazali na

malo izmjenjeni način (od -180 do 180 stupnjeva) kako bi centrirali naš dio Zemlje (ako postavimo od 0 do 360 stupnjeva centrirat će se Tihi ocean, dok će se naši krajevi naći na rubu karte). Geografsku širinu ograničili smo na jugu na -75 stupnjeva (minus označava južnu geografsku širinu), a na sjeveru na 75 stupnjeva (pozitivna vrijednost označava sjevernu geografsku širinu). Time smo izbjegli prikaz problematičnih dijelova krajnjeg sjevera i juga Zemlje. Karta koju kreiramo nalazi se na presjeku definiranih koordinata kutova, i prikazana je na Slici 3.26.

```
# kreiramo kartu u Mercatorovoj projekciji
# definirani su kutovi karte
# postavljena gruba rezolucija
m = Basemap(projection = 'merc', resolution = 'c',
            llcrnrlon = -180, llcrnrlat = -75,
            urcrnrlon = 180, urcrnrlat = 75)
```

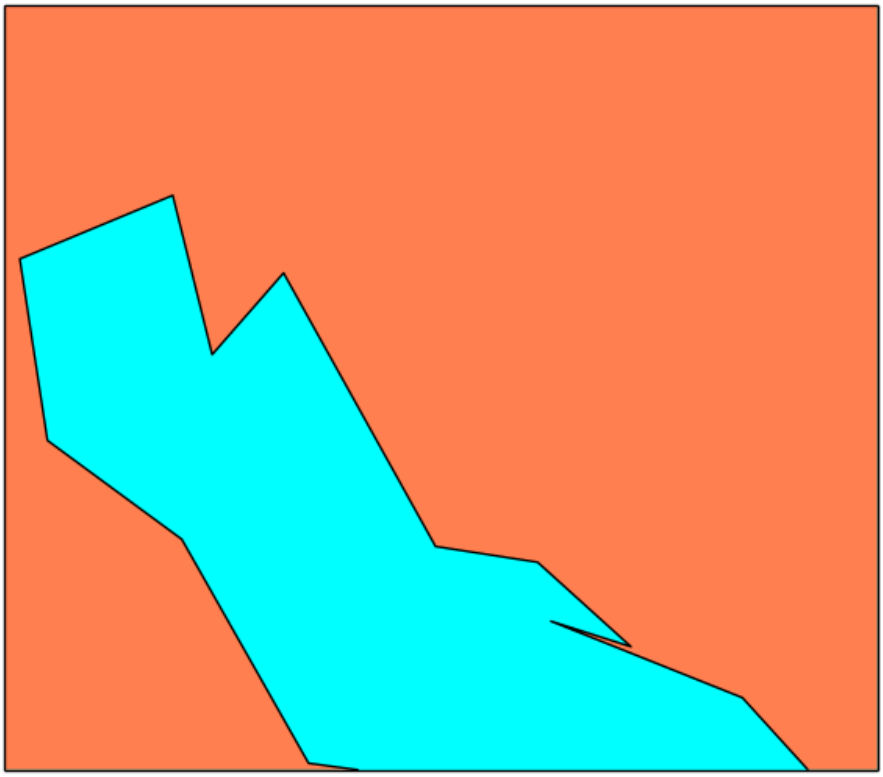
Slika 3.25: Izmijenjena naredba za kreiranje karte s određenim kutovima.



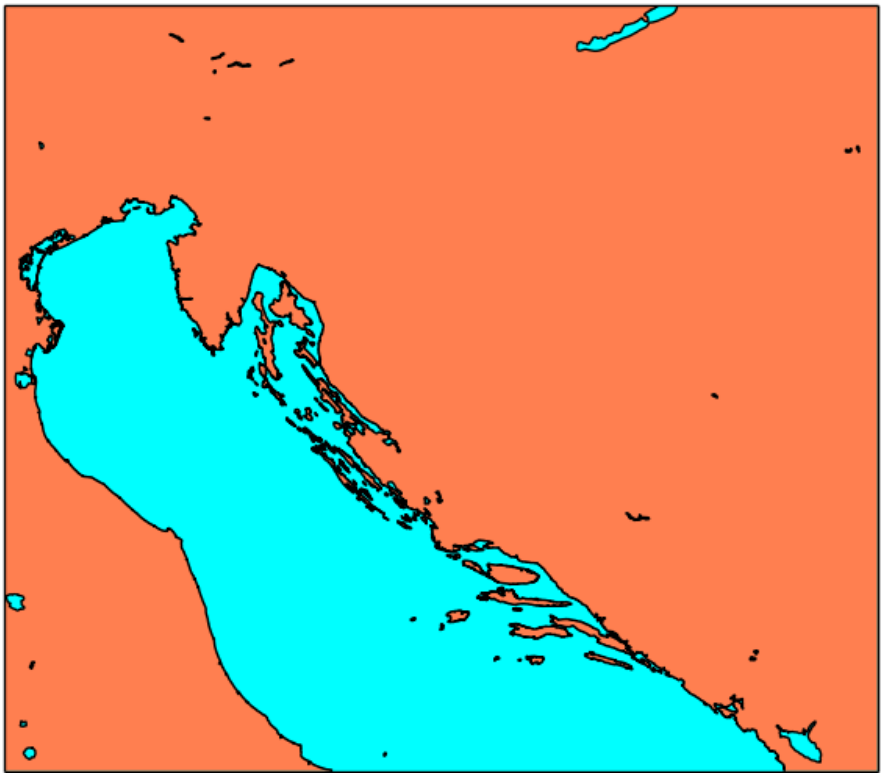
Slika 3.26: Karta svijeta u Mercatorovoj projekciji s određenim kutovima.

Da bi kreirali kartu Hrvatske, potrebno je modificirati rubne koordinate na Slici 3.25, stoga kreiramo kartu čija je donja granica na 42. stupnju, a gornja na 47. stupnju sjeverne geografske širine. Geografska dužina koja će se prikazivati nalazi se između 12. i 20. stupnja istočne geografske dužine. Karta koju smo dobili prikazana je na Slici 3.27.

Karta na Slici 3.27 prikazana je u istoj (gruboj) rezoluciji kao i karte svijeta u prethodnim primjerima. S obzirom da smo sada na karti sličnih dimenzija prikazali znatno manji dio Zemljine površine, javlja se potreba za korištenjem finije rezolucije. To možemo napraviti tako da izmjenimo vrijednost argumenta *resolution* u naredbi na Slici 3.25. Slovo 'c' oznaka je grube (engl. *crude*) rezolucije kao što je ranije već



Slika 3.27: Karta Hrvatske grube rezolucije.

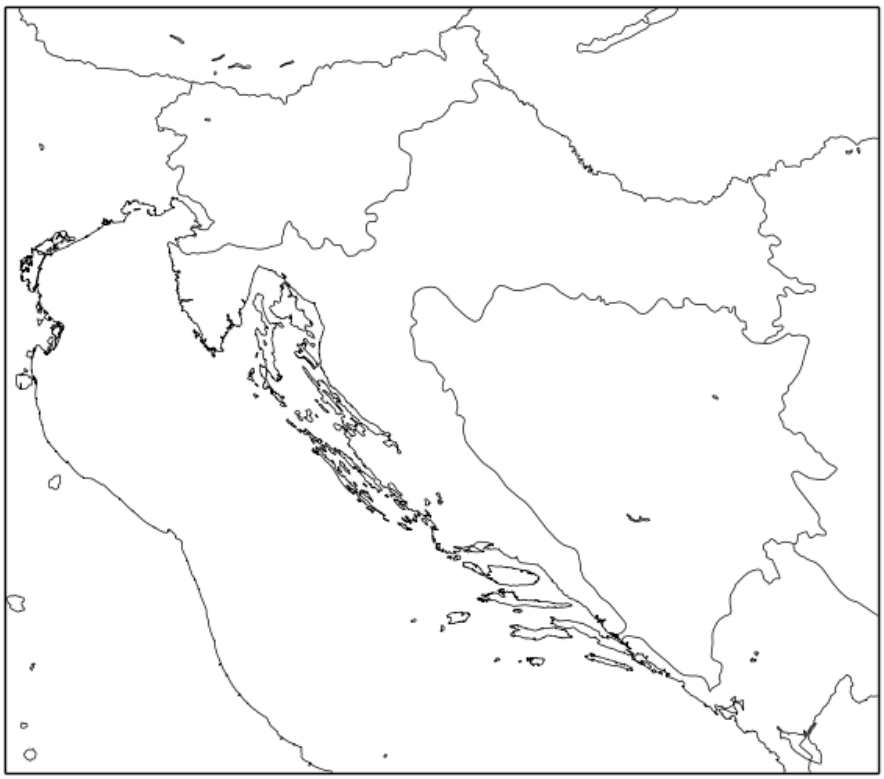


Slika 3.28: Karta Hrvatske fine rezolucije.

i navedeno. Za dobivanje manje grube rezolucije koristimo slovo 'i' (od engl. *intermediate*), ili slovo 'f' (od engl. *full*) za najfiniju rezoluciju. Karta Hrvatske najfinije

```
# ucitavamo Basemap modul
from mpl_toolkits.basemap import Basemap
# ucitavamo Matplotlibov Pyplot modul
import matplotlib.pyplot as plt
# ucitavamo numpy modul
import numpy as np

# kreiramo kartu u Mercatorovoj projekciji
m = Basemap(projection='merc', resolution = 'f',
            llcrnrlon=12, llcrnrlat=42,
            urcrnrlon=20, urcrnrlat=47)

# ucrtavamo obalnu crtu
m.drawcoastlines()
# ucrtavamo politicke granice
m.drawcountries()
# prikazujemo kartu
plt.show()
```

Slika 3.29: Program za kreiranje slijepe karte Hrvatske.



Slika 3.30: Slijepa karta Hrvatske.

rezolucije prikazana je na Slici 3.28.

Ubacivanjem još argumenata u naredbu *Basemap* možemo dodatno prilagoditi prikaz naše karte. Tako primjerice dodavanjem argumenta *area_thresh = 10* defini-

```
# transformiramo koordinate
x, y = m(lon, lat)
```

Slika 3.31: Programska naredba za transformaciju koordinata.

```
# ucitavamo Basemap modul
from mpl_toolkits.basemap import Basemap
# ucitavamo Matplotlibov Pyplot modul
import matplotlib.pyplot as plt
# ucitavamo numpy modul
import numpy as np

# kreiramo kartu u Mercatorovoj projekciji
# karta je definiran za podrucje Europe
m = Basemap(projection='merc', resolution = 'h',
            llcrnrlon=12, llcrnrlat=42,
            urcrnrlon=20, urcrnrlat=47)

# ucrtavamo obalnu crtu
m.drawcoastlines()
# ucrtavamo politicke granice
m.drawcountries()
# geografska duzina Zagreba
lon = 15.97
# geografska sirina Zagreba
lat = 45.81
# transformiramo koordinate
x, y = m(lon, lat)
# ucrtaj oznaku Zagreba
m.plot(x, y, 'bo', markersize=12)
# ucrtaj naziv grada
plt.text(x, y, 'Zagreb')
# prikazujemo kartu
plt.show()
```

Slika 3.32: Program za ucrtavanje pozicije grada Zagreba.

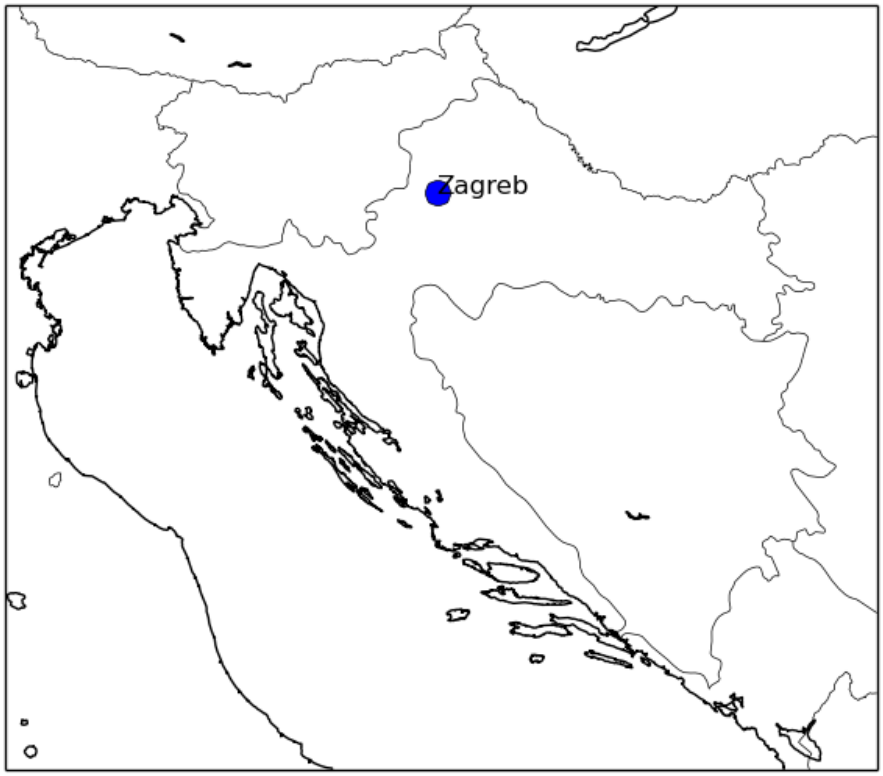
ramo da se na karti ne iscrtavaju obalne linije i jezera koja su površinom manja od 10 kilometara kvadratnih.

Za potrebe diplomskog rada korištena je karta Hrvatske, bez pozadinskih boja i bez oznaka (tzv. slijepa karta), ali s označenim političkim granicama radi lakšeg snalaženja u prostoru prilikom iščitavanja prognostičkih karata. Program za kreiranje slijepe karte Hrvatske prikazan je na Slici 3.29, dok je kreirana karta dostupna na Slici 3.30.

Prije crtanja podataka na kartu Hrvatske potrebno je transformirati geografske

koordinate podataka na koordinate ucrtane mape. To radimo pomoću naredbe prikazane na Slici 3.31. Basemap instanci *m* pridijelimo dva argumenta: geografsku dužinu i geografsku širinu. Ona nam vraća x i y koordinate u metrima, prilagođene za crtanje na karti [8].

Sada možemo na slijepoj karti ucrtati poziciju nekog grada, npr. Zagreba. Sve što nam treba je geografska dužina i geografska širina Zagreba. Pomoću funkcije *plot* ucrtat ćemo krug kao oznaku pozicije grada, a pomoću funkcije *text* upisati naziv grada. Program za ucrtavanje pozicije grada Zagreba i ispisivanje teksta pored oznake prikazan je na Slici 3.32, a rezultat na Slici 3.33.



Slika 3.33: Slijepa karta Hrvatske s ucrtanom pozicijom grada Zagreba.

Unutar Basemap modula postoje još brojne dodatne mogućnosti za izradu geografskih karata. Primjerice, korištenjem programske naredbe sa Slike 3.34 dobivamo topografsku kartu Europe koja je prikazana na Slici 3.35. Prije izvođenja programske naredbe potrebno je, isto kao i ranije, učitati sve potrebne module i kreirati kartu određene projekcije, u našem slučaju Mercatorove.

```
1 # ucrtavamo topografiju
2 m.etopo()
```

Slika 3.34: Programska naredba za crtanje topografske karte Europe.

Slika 3.35: Topografska karta Europe za čije je dobivanje potrebna programska naredba sa Slike 3.34.

3.6 Pygrib

U meteorologiji postoji puno podataka koje treba obraditi, analizirati ili prikazati, a pri tome se koriste posebni formati datoteka namijenjeni isključivo tome. GRIB je standardni format datoteke koji se koristi u meteorologiji za razmjenu meteoroloških podataka [9]. Prihvaćen je i od Svjetske meteorološke organizacije. Svaka GRIB datoteka ima dvije komponente: jedna koja opisuje podatke (sadržaj) datoteke (tzv. zaglavlje), a druga koja pohranjuje te podatke u binarnom obliku.

Postoji veliki broj softverskih paketa i programskih jezika koji mogu upotrebljavati GRIB datoteke. U Pythonu je razvijen Pygrib modul za čitanje i pisanje GRIB datoteka. Prije instalacije samog Pygrib modula, moraju se instalirati dodatni moduli kako bi se Pygrib mogao uspješno pokrenuti. Većina tih modula spomenuta je u prethodnim poglavljima: Numpy, Matplotlib i Basemap. Dodatni modul koji je potrebno instalirati prije instalacije Pygriba je Grib-api, modul koji je napisan u programskom jeziku C i koristi se za kodiranje i dekodiranje GRIB datoteka. Grib-api je razvijen (a i dalje se razvija) u Europskom centru za srednjoročne vremenske prognoze (ECMWF, akronim od engl. *European Center for Medium range Weather Forecasting*).

GRIB datoteke obično možemo preuzeti sa servera velikih meteoroloških organzacija, poput američke NOAA-e (akronim od engl. *National Oceanic and Atmospheric Administration*), NCEP-a (akronim od engl. *National Centers for Environmental Prediction*) ili europskog ECMWF-a. Dio tih organizacija daje većinu podataka besplatno

(npr. NCEP), dok dio organizacija (npr. ECMWF) omogućuje besplatno preuzimanje tek manjeg dijela meteoroloških podataka. Nakon preuzimanja GRIB datoteke, spremamo ju na disk i učitavamo pomoću Pygrib modula u Python. Kasnije se unutar samog programa može postaviti da se datoteka preuzima automatski, sprema na disk i učita u program.

```
1 # ucitavamo Pygrib modul
2 import pygrib
3 # otvaramo GRIB datoteku
4 grib = Pygrib.open('/putanja/gfs.t12z.pgrb2.0p25.f006')
5 # petljom ispisujemo sadrzaj GRIB datoteke
6 for sadrzaj in grib:
7     print(sadrzaj)
```

Slika 3.36: Učitavanje Pygrib modula i čitanje sadržaja GRIB datoteke.

Učitavanje Pygrib modula i čitanje GRIB datoteke prikazano je na Slici 3.36. Pygrib modul učitamo u Pythonu koristeći naredbu *import pygrib*. GRIB datoteka naziva *gfs.t12z.pgrb2.0p25.f006* preuzeta je sa stranica američkog centra za prognoze (NCEP) i sadrži numeričke podatke meteoroloških elemenata za određeni termin u budućnosti. Budući da GRIB datoteke američkog centra za prognoze sadrže podatke za cijelu Zemlju, prije preuzimanja datoteke definirana je pod-regija ograničena geografskom dužinom i geografskom širinom koja je nama potrebna za prikaz podataka na karti (Slika 3.30). Definiranjem pod-regije smanjujemo količinu podataka koju ćemo preuzeti i kasnije obrađivati, što nam ubrzava čitav proces.

Za učitavanje datoteke koristimo naredbu *pygrib.open()*, gdje kao argument postavljamo naziv GRIB datoteke (s putanjom ako je potrebno). Da bi ispisali sadržaj GRIB datoteke, koristimo se for petljom kao što je to i prikazano na Slici 3.36.

Nakon pokretanja programa ispisuje se cijelokupan sadržaj datoteke koji može biti vrlo velik, a manji dio tog sadržaja prikazan je na Slici 3.37. U sadržaju se među ostalim ispisuje naziv prognoziranog meteorološkog elementa, mjerna jedinica, prognozirani sat (u odnosu na početni termin) i vrijeme izlaza prognostičkog modela.

Određeni meteorološki element možemo učitati koristeći naredbu *grb.select()*, pri čemo se u argumentu nalazi ime tog elementa kojeg želimo učitati (Slika 3.38). Nakon odabira meteorološkog elementa (u našem slučaju maksimalne temperature), učitavamo podatke o maksimalnoj temperaturi u polje te ga ispisujemo. Ispis polja prikazan je na Slici 3.39. Oblik polja prilagođen je geografskim koordinatama koje se koriste u tom prognostičkom modelu, odnosno onim točkama za koje taj prognostički model vrši izračune u odabranoj pod-regiji. Temeperatura zraka zadana je u Kelvinima.

Jednostavno možemo provjeriti dimenziju polja, kao i minimalnu i maksimalnu vrijednost temperature zraka unutar polja (Slika 3.40). Ispis programa prikazan je na Slici 3.41.

```
1:U component of wind:m s**-1 (instant):regular_ll:unknown:level 0
     220:fcst time 6 hrs:from 201606151200
2:V component of wind:m s**-1 (instant):regular_ll:unknown:level 0
     220:fcst time 6 hrs:from 201606151200
3:Ventilation Rate:m**2 s**-1 (instant):regular_ll:unknown:level 0
     220:fcst time 6 hrs:from 201606151200
4:Wind speed (gust):m s**-1 (instant):regular_ll:surface:level 0:
    fcst time 6 hrs:from 201606151200

...

412:Pressure:Pa (instant):regular_ll:potentialVorticity:level
    2.147485648 K m2 kg-1 s-1:fcst time 6 hrs:from 201606151200
413:Vertical speed shear:s**-1 (instant):regular_ll:
    potentialVorticity:level 2.147485648 K m2 kg-1 s-1:fcst time 6
     hrs:from 201606151200
414:Pressure reduced to MSL:Pa (instant):regular_ll:meanSea:level
    0:fcst time 6 hrs:from 201606151200
415:5-wave geopotential height:gpm (instant):regular_ll:
    isobaricInhPa:level 50000 Pa:fcst time 6 hrs:from 201606151200
```

Slika 3.37: Ispis sadržaja GRIB datoteke.

```
# otvaramo GRIB datoteku
grib = pygrib.open('gfs.t12z.pgrb2.0p25.f006')
# odabiremo podatak (maksimalnu temperaturu)
tmax = grib.select(name='Maximum temperature')[0]
# ucitavamo podatke o maksimalnoj temperaturi u polje
tmax_podaci = tmax.values
# ispis polja
print(tmax_podaci)
```

Slika 3.38: Učitavanje podataka o maksimalnoj temperaturi u polje.

Preostaje nam još učitati podatke o geografskoj dužini i geografskoj širini kako bi podaci bili spremni za crtanje na karti. Programske naredbe koje učitavaju podatke geografskih koordinata prikazane su na Slici 3.42. U istom programu provjeravamo i dimenzije stvorenih polja, a ispis je dostupan na Slici 3.43. Valja primijetiti kako su dimenzije sva tri polja jednake (maksimalna temperatura, geografska dužina i geografska širina).

U diplomskom radu modul Pygrib koristi se za učitavanje (čitanje) meteoroloških podataka iz GRIB datoteka, odnosno numeričkih podataka vremenske prognoze za više meteoroloških elemenata (temperature zraka, relativne vlažnost, brzine vjetra i ostalih). Postoji još dodatnih mogućnosti Pygrib modula, tako je primjerice osim čitanja podataka moguće i zapisivanje podataka u novu GRIB datoteku.

```
[[ 295.6   298.6   300.   299.2   294.1   291.9   295.1   295.2   293.8
   296.5   300.2   301.8   299.6   298.1   297.7   296.8   295.6   295.7
    296.1   296.8   297.1   296.9   296.7   296.4   296.2   296.   296.
   295.9   295.5   295.2   297.7   294.5   290.2]

...

 [ 284.3   277.5   281.4   278.8   282.6   282.8   284.5   285.8   286.8
   290.7   290.5   289.7   291.3   295.7   296.   295.5   294.4   293.2
   292.9   292.5   292.3   292.   291.2   290.8   292.4   294.   294.7
   294.8   295.8   296.7   296.6   296.5   296.8]]
```

Slika 3.39: Ispis polja s učitanim podacima o maksimalnoj temperaturi.

```
# ispis dimenzije polja
print('Dimenzija polja:', tmax_podaci.shape)
# ispis minimalne vrijednosti temperature unutar polja
print('Minimalna temperatura:', tmax_podaci.min())
# ispis maksimalne vrijednosti temperature unutar polja
print('Maksimalna temperatura:', tmax_podaci.max())
```

Slika 3.40: Provjera dimenzije polja te minimalne i maksimalne vrijednosti.

```
Dimenzija polja: (21, 33)
Minimalna temperatura: 277.5
Maksimalna temperatura: 303.6
```

Slika 3.41: Ispis programa sa Slike 3.40.

```
# ucitavamo koordinate u polja
lat, lon = tmax.latlons()
# ispis dimenzija polja
print('Dimenzija polja geo. sirine:', lat.shape)
print('Dimenzija polja geo. duzine:', lon.shape)
```

Slika 3.42: Učitavanje geografskih koordinata i provjera dimenzije polja.

```
Dimenzija polja geo. sirine: (21, 33)
Dimenzija polja geo. duzine: (21, 33)
```

Slika 3.43: Ispis dimenzija polja geografskih dužina i širina.

3.7 Itertools

Ako želimo ispisati vrijednost maksimalne temperature zraka koja je pridjeljena

točno određenoj geografskoj dužini i geografskoj širini, moramo ulaziti u tri dvodimenzionalna polja i naći odgovarajuće vrijednosti. Ukoliko želimo ispisati sve vrijednosti maksimalne temperature zraka koje pripadaju određenim geografskim dužinama i širinama moramo koristiti petlje. Programske petlje često se koriste za iteraciju, odnosno prolazak kroz sve ili dio elemenata nekog polja.

Pythonov modul Itertools sastoji se od funkcija koje kreiraju iteratore za prolazak kroz petlju. Prednost modula je u velikoj brzini izvršavanja petlje i malom zauzeću radne memorije [10]. Umjesto korištenja više ugniježdenih programskih petlji, Itertools omogućuje jednostavnije definiranje petlje, čime naš kod u Pythonu postaje čitljiviji.

```
# ucitavamo Itertools modul
import itertools
# sva su polja vec ranije definirana
# i ucitana u prethodnim programima
# polje s vrijednostima maksimalnih temperatura
tmax_podaci
# polje s vrijednostima geografskih duzina
lon
# polje s vrijednostima geografskih sirina
lat
# dimenzije svih polja su (21, 33)
# kreiranje iteratora za petlju
for i, j in itertools.product(range(0, 21), range(0,33)):
    # ispis temperature za svaku geo. duzinu i sirinu
    print(lat[i][j], lon[i][j], tmax_podaci[i][j])
```

Slika 3.44: Korištenje modula Itertools za iteraciju kroz petlju.

```
42.0 12.0 295.6
42.0 12.25 298.6
42.0 12.5 300.0

...

47.0 19.5 296.6
47.0 19.75 296.5
47.0 20.0 296.8
```

Slika 3.45: Rezultat izvođenja programa sa Slike 3.44.

U ovom diplomskom radu Itertools se koristi prilikom ispisivanja vrijednosti (brojeva) meteoroloških podataka na kartu. Pritom moramo proći kroz sve elemente svih polja, i to činimo koristeći Itertools funkciju *product()*. Funkcija je ekvivalentna ug-

nježdenim for petljama. Primjer korištenja Itertools modula prikazan je na Slici 3.44, dok je rezultat programa dostupan na Slici 3.45.

4 Priprema meteoroloških prognostičkih karata

4.1 GFS prognostički model

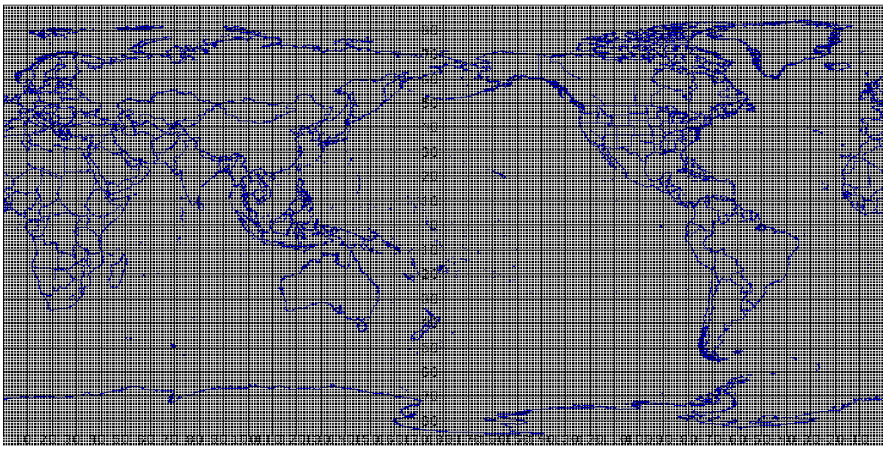
Za potrebe ovog diplomskog rada i izrade prognostičkih karata koristit ćemo američki prognostički model GFS (akronim od engl. *Global Forecast System*). Prognostički model razvija NCEP koji djeluje u sklopu NOAA-e [11].

Podaci kojima raspolaže NCEP obuhvaćaju široki spektar meteoroloških elemenata vezanih za tlo, oceane i atmosferu za područje cijele Zemlje. GFS model sastoji se od kompozicije nekoliko odvojenih modela, a to su: model atmosfere, model oceana, model tla i model leda u oceanima. Promjene na GFS modelu izrađuju se konstantno, a sve u svrhu poboljšanja rezultata i veće preciznosti prognoze [11]. GRIB datoteke koje sadrže podatke o prognozi mogu se preuzeti s internet stranica NCEP-a. Varijable za koje se izrađuje prognoza kreću od osnovnih meteoroloških elemenata poput temperature zraka, relativne vlažnosti, tlaka zraka, vjetrova pa sve do vlažnosti tla i koncentracije ozona u atmosferi. Ispis dijela sadržaja GFS GRIB datoteke prikazan je na Slici 3.37. Mnogi od meteoroloških elemenata ne odnose se samo na prizemni sloj (najčešće 2 metra od tla), već i po visinama počevši od tla pa sve do vrha atmosere, čime se dobiva potpuni uvid u stanje svih slojeva atmosfere.

Nakon poboljšanja prognostičkog modela tijekom 2015. godine povećan je broj točaka za koje se izrađuje prognoza tako da razmak između točaka sada iznosi oko 13 kilometara. Ranije je rezolucija iznosila oko 27 kilometara. Horizontalna mreža točaka za koje GFS prognostički model vrši izračune prikazana je na Slici 4.46. Osim horizontalne rezolucije, konstantno se poboljšava i vertikalna rezolucija modela, tako je i gustoća točaka od tla pa do vrha atmosfere povećana. Vremenski razmaci na koje se prognoza odnosi dostupni su do 240 sati unaprijed (10 dana) u intervalima od 3 sata, te od 240. do 384. sata unaprijed (od 10. do 16. dana) u intervalima od 6 sati [11]. Najnovije poboljšanje modela u 2016. godini omogućilo je dostupnost prognoze do 120. prognoziranog sata (do 5. dana) u intervalima od samo 1 sat.

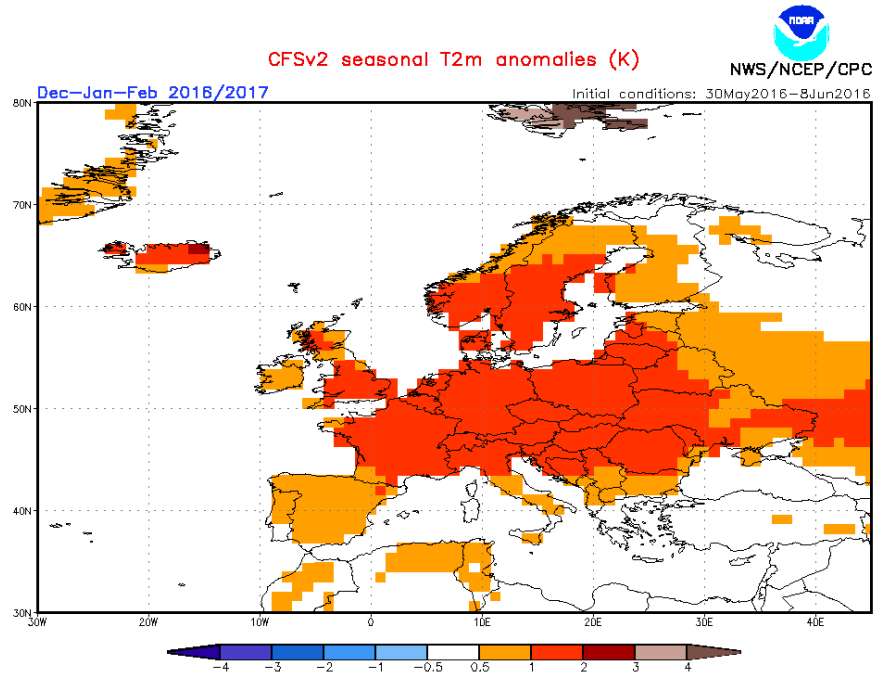
GFS model ima 4 izlaza na dan, redom u terminima: 00 UTC, 06 UTC, 12 UTC i 18 UTC (UTC, akronim od engl. *Coordinated Universal Time*). Vrijeme izlaza modela iskazano je po vremenu nultog meridijana, odnosno početne vremenske zone. Za dobivanje lokalnog vremena u Hrvatskoj dodajemo 1 sat zimi, odnosno 2 sata ljeti.

Osim GFS prognostičkog modela, NCEP razvija još veliki broj drugih modela za analizu vremena i vremensku prognozu. Tako se mogu naći specijalizirani modeli za more (valovi, temperatura mora, plima i oseka), sezonski prognostički modeli koji predviđaju vrijeme na nekoliko mjeseci pa do pola godine unaprijed (Slika 4.47), kao i regionalni modeli poboljšane rezolucije namijenjeni prvenstveno za područje SAD-a. Na temelju ulaznih podataka GFS-a izrađuju se i neki regionalni modeli visoke rezolucije poput WRF NMM-a (akronim od engl. *Weather Research and Forecasting Nonhydrostatic Mesoscale Model*) ili WRF ARW-a (akronim od engl. *Weather Research*



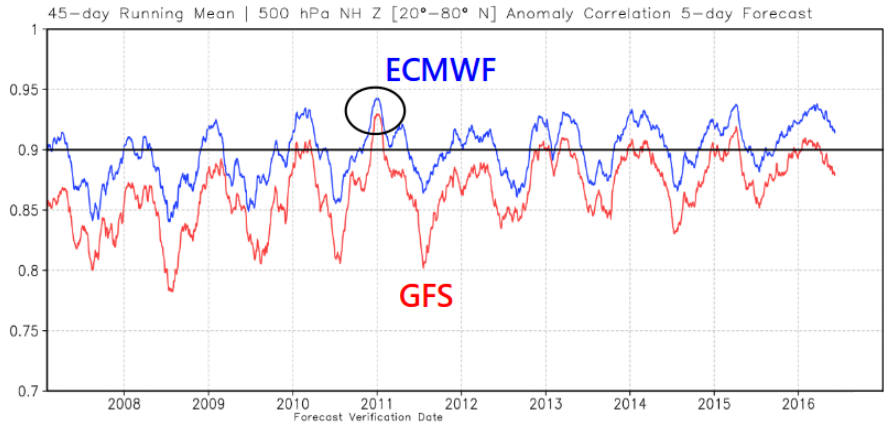
Slika 4.46: Horizontalna mreža točaka za koje GFS vrši izračune.

and Forecasting Advanced Research WRF).



Slika 4.47: Sezonska prognoza za Europu za zimu 2016./2017. koju razvija NCEP. Odstupanje temperature zraka za prosinac, siječanj i veljaču u odnosu na prosjek. Preuzeto s: `http://www.cpc.ncep.noaa.gov`

Osim američkog GFS-a, i druge meteorološke organizacije diljem svijeta razvijaju svoje prognostičke modele. Tu je najpoznatiji engleski ECMWF. Spomenimo još UKMO koji se također razvija u Engleskoj. Tu su i kanadski GEM, japanski JMA, australski BOM, itd. Prema provjerama koje se stalno izvršavaju, najbolje rezultate ostvaruju ECMWF i GFS, u čemu je ECMWF tek neznatno bolji (Slika 4.48).



Slika 4.48: Preciznost prognoze za peti prognozirani dan za GFS i ECMWF. Preuzeto s: `http://www.ecmwf.int/sites/default/files/elibrary/2016/16424-enhanced-predictability-during-extreme-winter-flow-patterns.pdf`

4.2 Učitavanje modula i podataka

Na početku Python programa moramo učitati sve potrebne module koje će nam biti potrebni u daljnjem radu. Svi Python moduli koje koristimo opisani su ranije u Poglavlju 3. Učitavanje modula prikazano je na Slici 4.49.

```
1 from mpl_toolkits.basemap import Basemap
2 import numpy as np
3 import matplotlib as mpl
4 import matplotlib.pyplot as plt
5 import scipy.ndimage as ndimage
6 import itertools
7 import pygrib
```

Slika 4.49: Učitavanje svih potrebnih Python modula.

Podatke iz GFS GRIB datoteka učitavamo u Python na način opisan u Poglavlju 3.6. Sada ćemo ponovno koristiti istu GRIB datoteku naziva *gfs.t12z.pgrb2.0p25.f006* te učitati podatke u polje. Datoteka sadržava podatke o prognozi meteoroloških elemenata prognostičkog modela GFS za 6 sati unaprijed od početnog termina, odnosno termina kada je model inicijaliziran. Učitavanje GRIB datoteke, geografskih koordinata, kreiranje karte Hrvatske i prilagođavanje koordinata za crtanje na karti prikazano je na Slici 4.50.

Valja primjetiti da smo polje s podacima geografskih dužina *lon* i geografskih širina *lat* dodatno proširili funkcijom *zoom* iz paketa *scipy.ndimage* koja je ranije također opisana u Poglavlju 3.3.

Funkcijom *zoom* proširili smo i sva polja s podacima meteoroloških elemenata. Polja su dodatno proširena zbog bolje interpolacije prilikom crtanja podataka na kartu. Slika 4.51 prikazuje učitavanje nekih osnovnih meteoroloških elemenata iz

GRIB datoteke koje ćemo kasnije ucrtati u kartu. Svi su podaci učitani u polje te su u kombinaciji s poljima geografskih dužina i geografskih širina spremni za crtanje na kartama.

```
# ucitavamo GRIB datoteku koja je vec pohranjena na lokalnom disku
gribs = pygrib.open('/putanja/gfs.t12z.pgrb2.0p25.f006')

# ucitavamo geografsku duzinu i geografsku sirinu
lat_src, lon_src = (gribs.select(name='Temperature')[0]).latlons()
lat = ndimage.zoom(lat_src, 3)
lon = ndimage.zoom(lon_src, 3)

# kreiramo kartu Hrvatske
m = Basemap(projection='merc', lat_0=45, lon_0=23,
            resolution = 'h', area_thresh = 1,
            llcrnrlon=12, llcrnrlat=42,
            urcrnrlon=20, urcrnrlat=47)

# transformiramo koordinate
x, y = m(lon, lat)
```

Slika 4.50: Učitavanje GRIB datoteke, geografskih koordinata, kreiranje karte Hrvatske i prilagođavanje koordinata za crtanje na karti.

Osnovni meteorološki elementi koji su učitani u polje na Slici 4.51 su:

- temperatura zraka na 2 metra od tla
- tlak zraka sveden na razinu mora
- u-komponenta vjetra na 10 metara od tla
- v-komponenta vjetra na 10 metara od tla
- brzina vjetra na 10 metara od tla
- ukupna naoblaka
- količina oborine

Preostaje nam još definirati termin izlaza prognostičkog modela i termin na koji se prognoza odnosi, a te je podatke zgodno i ucrtati na karti. Unutar GRIB datoteke uz svaki meteorološki element postoje parametri koji sadržavaju podatke o terminu izlaska prognostičkog modela (u našem slučaju *20160615* i *1200*), te podatak o terminu (prognozirani sat) na koji se prognoza odnosi (u našem slučaju *6*). Prognozirani termin je broj sati unaprijed na koji se odnosi prognoza (u našem slučaju prognoza

```
# ucitavamo temperaturu zraka na 2 metra od tla
tmp2m = ndimage.zoom(((gribs.select(name='2 metre temperature')
    [0]).values) - 273.15, 3)
# ucitavamo tlak zraka sveden na razinu mora
pressure = ndimage.zoom(((gribs.select(name='Pressure reduced to
    MSL')[0]).values) * 0.01, 3)
# ucitavamo u-komponentu vjetra na 10 metara od tla
uwind = ndimage.zoom((gribs.select(name='10 metre U wind component
    ')[0]).values, 3)
# ucitavamo v-komponentu vjetra na 10 metara od tla
vwind = ndimage.zoom((gribs.select(name='10 metre V wind component
    ')[0]).values, 3)
# izracunavamo brzinu vjetra na 10 metara od tla
wind = np.sqrt(uwind*uwind + vwind*vwind)
# ucitavamo ukupnu naoblaku
cloud = ndimage.zoom((gribs.select(name='Total Cloud Cover')[0]).
    values, 3)
# ucitavamo ocekivanu kolicinu oborine zadnjih 6 sata
precip = ndimage.zoom((gribs.select(name='Total Precipitation')
    [0]).values, 3)
```

Slika 4.51: Učitavanje meteoroloških elemenata u polje.

```
# dohvacamo datum izlaza prognostickog modela
datum = gribs[1]['dataDate']
# dohvacamo sat izlaza prognostickog modela
sat = gribs[1]['dataTime']
# dohvacamo termin na koji se prognoza odnosi (prognozirani sat)
termin = gribs[1]['forecastTime']
```

Slika 4.52: Dohvaćanje podataka o vremenu izlaska modela i prognoziranom terminu.

se odnosi na 15.06.2016. u 18 UTC, pošto je izlaz modela isti dan u 12 UTC). Dohvaćanje podataka o vremenu izlaska modela i prognoziranom terminu prikazan je na slici 4.52.

Sada smo u potpunosti završili proces pripreme podataka i spremni smo za crtanje prognostičkih karata. Učitali smo sve potrebne module, kreirali kartu Hrvatske na koju ćemo ucrtavati podatke, učitali smo podatke nekih meteoroloških elemenata i pripadne koordinate te transformirali koordinate za crtanje na karti.

4.3 Temperatura zraka

Temperatura zraka jedan je od najvažnijih meteoroloških elemenata. Prema pravilima Svjetske meteorološke organizacije, temperatura zraka mjeri se u hladu mete-

orološke kućice (meteorološkog zaklona) na 2 metra visine od tla [2]. Meteorološka kućica štiti instrumente od nepoželjnog vanjskog utjecaja, ali i direktnih sunčevih zraka.

Temperatura zraka može se mjeriti i na drugim visinama, primjerice na 100 metara od tla, 1500 metara od tla, ali također i na nekoj dubini u tlu, primjerice 1 metar ispod zemlje. Isto tako i prognostički modeli prognoziraju temperaturu zraka (ali i neke druge meteorološke elemente) na više dubina i visina. Upravo iz tog razloga važno je definirati da prikazujemo temperaturu zraka na 2 metra.

Jedinica za mjerenje temperature je Kelvin, a u Kelvinima je zadana i temperatura unutar prognostičkog modela. S obzirom na to da je nama prihvatljivija Celzijeva ljestvica koja je u svakodnevnoj uporabi, temperaturu pretvaramo u Celzijeve stupnjeve pomoću formule (4.8), što smo napisali odmah prilikom učitavanja podataka u polje na Slici 4.51.

$$t(°C) = T(K) - 273.15 \tag{4.8}$$

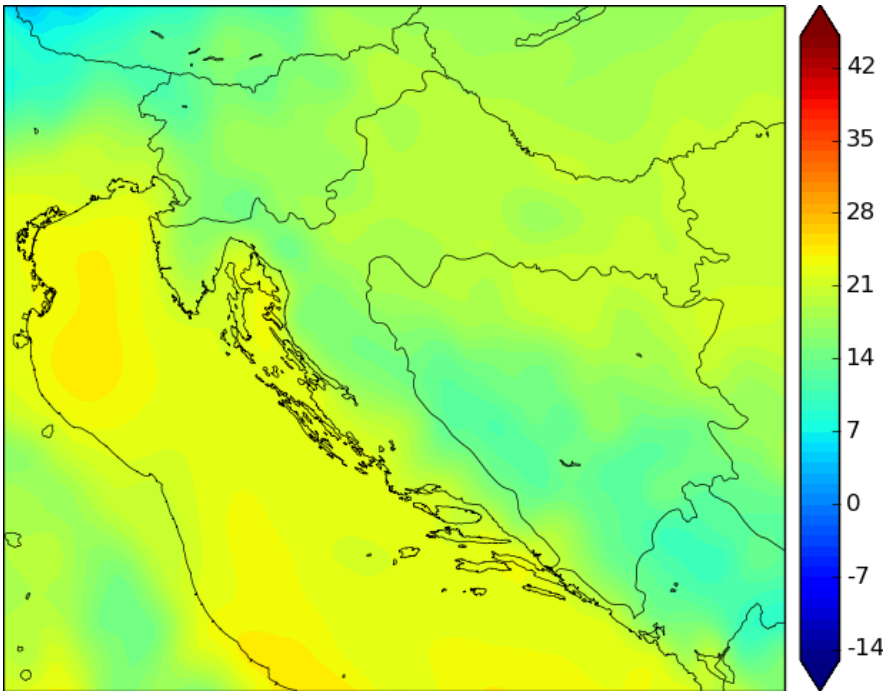
```
# ucrtavamo podatke na pozadinu karte
cs = m.contourf(x, y, tmp2m, np.linspace(-15,45,61), cmap=plt.cm.
    jet, extend='both')
# ucrtavamo legendu
cb = m.colorbar(cs, spacing='proportional', format='%1i')
# ucrtavamo obalnu liniju
m.drawcoastlines(linewidth=0.5)
# ucrtavamo politicke granice
m.drawcountries()

# spremamo sliku pod nazivom temperatura.png
plt.savefig('temperatura.png')
```

Slika 4.53: Ucrtavanje podataka o temperaturi i legende na kartu.

Crtanje podataka na kartu izvršava se pomoću programskog koda sa Slike 4.53, koji je ustvari samo nastavak na programske kodove od ranije kada smo učitavali potrebne module i podatke, crtali kartu i transformirali koordinate. Crtanje započinjemo funkcijom *contourf* koja sadržava nekoliko argumenata podrobnije opisanih u Tablici 4.4. Pomoću ove funkcije crtamo konture na karti pri čemu svaki stupanj Celzijeve skale ima drugu boju. Funkcija *colorbar* ucrtava legendu na desnom rubu karte, i ovisna je o prethodnoj funkciji *contourf* pošto se legenda formira na temelju podataka i argumenata iz funkcije *contourf*, kao što su primjerice ljestvica temperature i boje. Kao dodatne argumente funkcije *colorbar* postavili smo da razmak između pojedinih podjeljaka na skali bude proporcionalan te da vrijednosti temperature koje se ispisuju pored ljestvice budu cijeli brojevi. Potom smo ucrtavali obalnu liniju, političke granice i na kraju spremamo sliku pod nazivom *temperatura.png*.

Argument	Opis
x	Transformirane koordinate x-osi.
y	Transformirane koordinate y-osi.
tmp2m	Ranije učitani podaci o temperaturi zraka koji su prilagođeni za crtanje u ovisnosti o x i y koordinatama.
np.linspace(-15,45,61)	Kreiramo Numpy polje s vrijednostima od -15 do 45, pri čemu imamo 61 podjeljak. Prema tome je svaki broj od -15 do 45 jedan element tog polja.
cmap=plt.cm.jet	Odabiremo ljestvicu boja kako je ranije opisano u Poglavlju 3.4. Ljestvica boja ima 61 podjeljak kako je to definirano u retku iznad.
extend='both'	U slučaju da temperatura zraka bude niža od -15°C ili viša od 45°C pojavila bi se bijela pozadinska boja, pošto te vrijednosti izlaze izvan našeg definiranog Numpy polja. Iz tog razloga koristimo ovaj argument kako bi i te temperature zraka bile obojane zadnjom definiranom bojom u tom dijelu ljestvice.

Tablica 4.4: Opis argumenata korištenih u funkciji *contourf*.



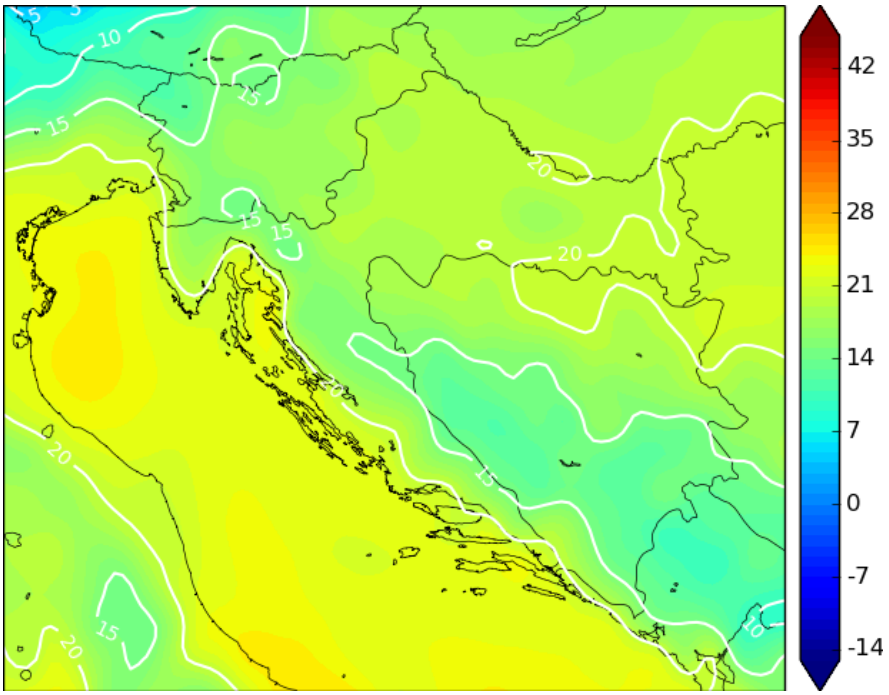
Slika 4.54: Prognostička karta s ucrtanom temperaturom zraka i legendom dobivena programskim kodom sa Slike 4.53.

Karta koju smo kreirali koristeći programski kod sa Slike 4.53 prikazana je na

Slici 4.54. Ta karta nije lako čitljiva, i pored toga što sa strane postoji jasno definirana skala boja. Kako bi olakšali čitanje karte, možemo dodatno ucrtati linije koje spajaju točke istih temperatura (izoterme). Te linije možemo ucrtati ako našem kodu sa Slike 4.53, prije spremanja slike, dodamo programske naredbe prikazane na Slici 4.55.

```
# ucrtavamo linije za svakih 5 stupnjeva
linije = m.contour(x, y, tmp2m, np.arange(-15, 45, 5), linewidths
    =1.5, colors='white')
# pridjeljujemo oznake linijama
plt.clabel(linije, inline=1, fontsize=10, fmt='%1.0f')
```

Slika 4.55: Programski kod za ucrtavanje linija koje povezuju točke istih temperatura.



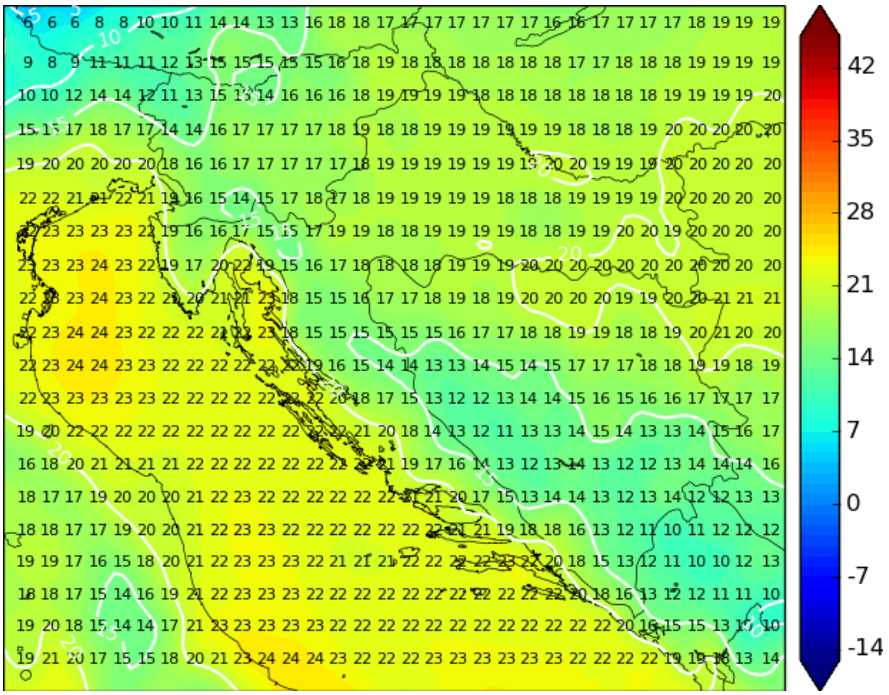
Slika 4.56: Prognostička karta temperature zraka s ucrtanim linijama koje povezuju područja istih temperatura.

Funkcija *contour* sa Slike 4.55 ucrtava linije koje povezuju područja iste temperature. Kao argumenti u funkciji nalaze se ponovno podaci transformiranih geografskih koordinata x i y, podaci o temperaturi zraka, a kreirano je i Numpy polje čiji su elementi brojevi od -15 od 45 u koraku od 5. Polje zapravo sadržava vrijednosti temperature za koju želimo ucrtati linije, a to je dakle za svakih 5 Celzijevih stupnjeva u intervalu od -15 do 45. Unutar funkcije postoje i argumenti *linewidths* za definiranje debljine linije i *colors* za definiranje boje linije. Postoji još dodatnih argumenata koji se mogu ubaciti, primjerice za animaciju linija, ali to nećemo koristiti ovom prilikom. Slika dobivena dodavanjem dodatnog programskog koda sa Slike 4.55 dostupna je na Slici 4.56.

Dodatno možemo olakšati isčitavanje prognostičke karte ucrtavanjem vrijednosti temeprature zraka direktno na kartu, što možemo ostvariti programskim naredbama prikazanim na Slici 4.57. Pritom se koristimo Pythonovim modulom Itertools koji je podrobnije opisan ranije u Poglavlju 3.7. Modul nam koristi za brzu iteraciju *i* i *j* elemenata unutar petlje kako bismo mogli efikasno čitati sve elemente sva tri dvodimenzionalna polja: x koordinate, y koordinate i vrijednosti temperature za svaki uređeni par koordinata. Elemente iteriramo u intervalu od 3 do 63 (s korakom 3) za *i* te u intervalu od 3 do 99 (s korakom 3) za *j*, što mora odgovarati broju elemenata polja geografskih dužina i širina koji u našem slučaju nije jednak.

```
# ucrtavamo vrijednosti temperature na kartu
for i, j in itertools.product(range(3, 63, 3), range(3, 99, 3)):
    plt.text(x[i][j], y[i][j], repr(int(tmp2m[i][j])), fontsize=8,
             ha='center', va='center', color='black')
```

Slika 4.57: Programski kod za ucrtavanje vrijednosti temperature na kartu.



Slika 4.58: Prognostička karta temperature zraka s ucrtanim linijama koje povezuju područja istih temperatura i vrijednostima temperature na karti.

Unutar *for* petlje koristimo se funkcijom *text* čija je zadaća da ispisuje tekst ili oznake na određene koordinate unutar karte. S obzirom da se funkcija *text* nalazi unutar petlje koja iterira, ona crta jednu po jednu vrijednost na kartu s obzirom na *i* i *j* elemente. Za svaki uređeni par koordinata ucrtavamo vrijednost temperature u određenoj točki, što smo zadali u argumentu funkcije *text*. Dodatni pridruženi

argumenti su veličina fonta, vertikalno i horizontalno poravnanje i boja fonta što je prikazano na Slici 4.57. Kreirana karta dostupna je na Slici 4.58.

Konačno, našoj karti možemo pridijeliti naslov te vrijeme izlaska modela i prognozirani sat, kako bismo znali koji se meteorološki element prikazuje na karti i termin na koji se odnosi. Također potrebno je označiti mjernu jedinicu meteorološkog elementa kojeg prikazujemo, a u slučaju temperature to je °C. Ukoliko je na karti ucrtana ljestvica boja prognoziranog elementa, najbolje je mjernu jedinicu ucrtati pored same ljestvice (legende).

```
# ucrtavamo naslov
plt.title('Temperatura zraka na 2 metra\n +' + str(termin) + 'h')
# uctavamo mjernu jedinicu za temperaturu zraka
cb.set_label('°C', labelpad=-10, y=1.05, rotation=0)
# ucrtavamo naziv modela i vrijeme izlaza modela
plt.figtext(.15, .07, 'Izlaz: GFS 0.25°, ' + str(datum) + ' ' +
    str(sat) + 'UTC')
```

Slika 4.59: Programski kod za ucrtavanje naslova i termina, mjerne jedinice, naziva prognostičkog modela i vremena izlaza modela.

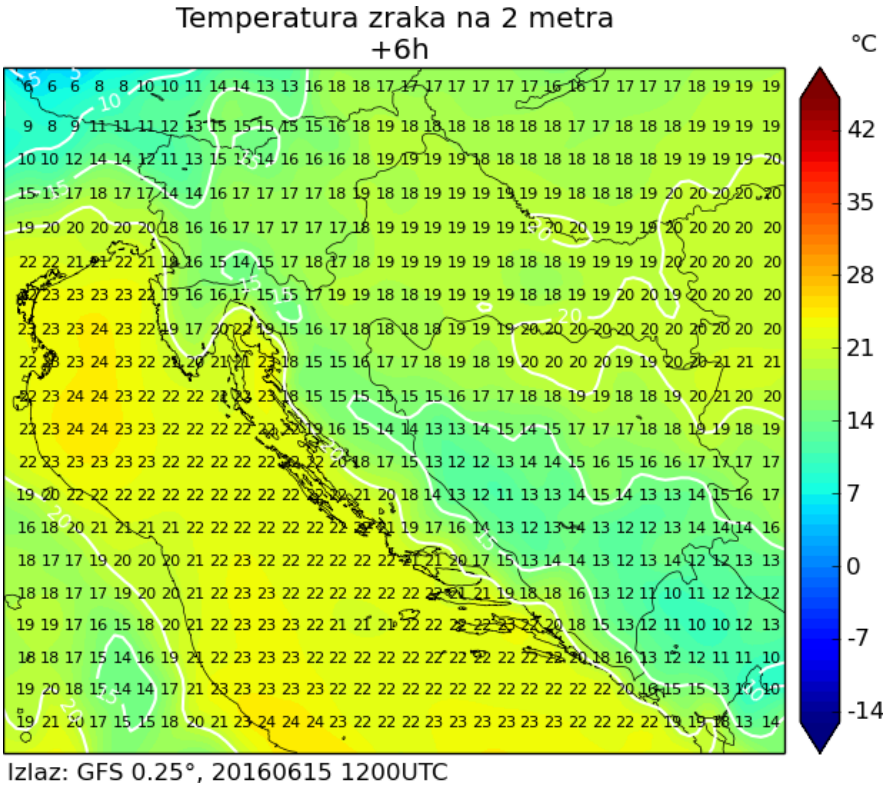
```
cs = m.contourf(x, y, tmp2m, np.linspace(-15,45,61), cmap=plt.cm.
    jet, extend='both')
cb = m.colorbar(cs, spacing='proportional', format='%1i')
m.drawcoastlines(linewidth=0.5)
m.drawcountries()

linije = m.contour(x, y, tmp2m, np.arange(-15, 45, 5), linewidths
    =1.5, colors='white')
plt.clabel(linije, inline=1, fontsize=10, fmt='%1.0f')

for i, j in itertools.product(range(3, 63, 3), range(3, 99, 3)):
    plt.text(x[i][j], y[i][j], repr(int(tmp2m[i][j])), fontsize=8,
            ha='center', va='center', color='black')

cb.set_label('°C', labelpad=-10, y=1.05, rotation=0)
plt.title('Temperatura zraka\n +' + str(termin) + 'h')
plt.figtext(.15, .07, 'Izlaz: GFS 0.25°, ' + str(datum) + ' ' +
    str(sat) + 'UTC')

plt.savefig('temperatura.png')
```

Slika 4.60: Program za crtanje svih elemenata prognostičke karte temperature zraka na 2 metra.

Za ucrtavanje naslova prognostičke karte i termina na koji se prognostička karta

odnosi koristimo se naredbom *title* kako je to prikazano na Slici 4.59. Ta naredba uobičajeno, ako to drugačije ne definiramo, ucrtava naslov iznad karte. U naslovu ispisujemo naziv i termin karte, odnosno meteorološki element i broj sati na koji se karta odnosi. Naredba *set_label* ucrtava mjernu jedinicu pored skale boja (legende), dok smo s argumentima *labelpad*, *y* i *rotation* precizirali točno mjesto ucrtavanja mjerne jedinice. Naredba *figtext* služi nam za ucrtavanje teksta na području karte čiju poziciju možemo točno definirati s prva dva argumenta naredbe. Tom naredbom ispisujemo na kartu model i njegovu rezoluciju (GFS 0.25°), kao i vrijeme izlaza prognostičkog modela. Spomenimo još da u Pythonu postoje dodatni moduli uz pomoć kojih možemo bolje oblikovati i prikazati termine izlaska modela i termine na koji se prognostička karta odnosi.

Potpuni kod za crtanje svih elemenata prognostičke karte temperature zraka prikazan je na Slici 4.60. Kod je funcionalan ako se prethodno učitaju svi potrebni Python moduli, kreira karta i učitaju svi potrebni podaci u polja (geografske širine, temperature zraka, termin izlaza i termin prognoze). Dobivena karta dostupna je na Slici 4.61.



Slika 4.61: Prognostička karta temperature zraka na 2 metra dobivena pomoću programa sa Slike 4.60.

4.4 Tlak zraka

Tlak se mjeri kao težina stupca zraka, tj. dovodi se u ravnotežu s težinom stupca neke tekućine, npr. žive. Jedinica za tlak je Paskal (Pa), no u meteorologiji se najčešće

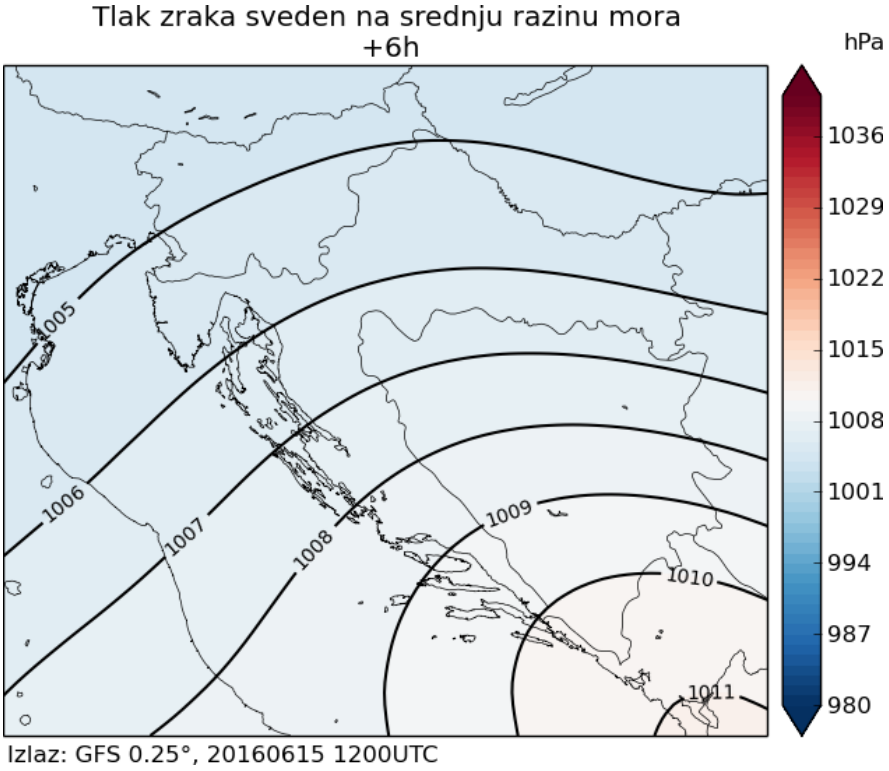
uzima sto puta veća jedinica (hPa) [2]. S obzirom da tlak zraka pada s porastom visine, ukoliko želimo uspoređivati vrijednosti tlakova iznad nekog područja (gdje se meteorološke postaje općenito nalaze na različitim visinama), moramo obaviti svođenje tlaka na referentnu razinu. Najčešće se kao referentna razina uzima srednja morska razina.

Iz GFS GRIB datoteke u polje sa Slike 4.51 učitali smo tlak zraka sveden na srednju razinu mora, te ga odmah pretvorili u hPa. Za crtanje na karti koristimo se sličnim postupkom kao i za ranije opisanu temperaturu zraka, ali bez ucrtavanja podataka direktno na kartu, pošto je u slučaju tlaka zraka karta preglednija samo s linijama koje povezuju točke istih tlakova i skalom boja. Programski kod za crtanje karte tlaka zraka svedenog na razinu mora prikazan je na Slici 4.62, dok je odgovarajuća karta dostupna na Slici 4.63. Programski kod je funkcionalan ako su ranije već učitani potrebni moduli i podaci.

```
cs = m.contourf(x, y, pressure, np.linspace(980, 1040, 61), cmap=
    plt.cm.RdBu_r, extend='both')
cb = m.colorbar(cs, spacing='proportional', format='%1i')
m.drawcoastlines(linewidth=0.5)
m.drawcountries()

linije = m.contour(x, y, pressure, np.arange(980, 1040, 1),
    linewidths=1.5, colors='black')
plt.clabel(linije, inline=1, fontsize=10, fmt='%1.0f')

cb.set_label('hPa', labelpad=-10, y=1.05, rotation=0)
plt.title('Tlak zraka sveden na srednju razinu mora\n +' + str(
    termin) + 'h')
plt.figtext(.15, .07, 'Izlaz: GFS 0.25°, ' + str(datum) + ' ' +
    str(sat) + 'UTC')

plt.savefig('tlak.png')
```

Slika 4.62: Program za crtanje svih elemenata prognostičke karte tlaka zraka svedenog na razinu mora.

Tlak zraka na Zemljinoj površini nije svuda jednak, ne samo zbog različite nadmorske visine, nego i zato što se u atmosferi neprekidno zbivaju procesi hlađenja i grijanja zraka te priljeva mase zraka na jedno mjesto (tlak raste) ili razilaženja (tlak pada). Kada su izobare (linije koje povezuju područja istog tlaka) na nekom području zatvorene krivulje, one tvore područja visokog ili niskog tlaka (anticiklone i ciklone) [2]. Na karti tlaka zraka mogu se dodatno ucrtati oznake visokog i niskog tlaka u području gdje je tlak najviši, odnosno najniži. To najčešće označavamo slovima V za visoki tlak i N za niski tlak. Također, karta prizemnog tlaka zraka ponekad se može povezati s ostalim meteorološkim elementima koji se prikazuju na istoj karti,



Slika 4.63: Prognostička karta tlaka zraka svedenog na razinu mora dobivena pomoću programa sa Slike 4.62.

iz koje se onda može iščitati više podataka u nekim specifičnim uvjetima.

4.5 Smjer i brzina vjetra

Vjetar kao vektorska veličina određen je smjerom i brzinom. Pod smjerom vjetra podrazumijeva se strana svijeta otkuda vjetar puše. Pod brzinom vjetra podrazumijeva se put što ga prevali čestica zraka u jedinici vremena. Prizemni vjetar obično se mjeri na visini 10 metara od tla. Najčešće se određuje srednji vjetar u nekom intervalu (primjerice 10-minutnom) [2]. Brzina vjetra najčešće se izražava u m/s ili km/h. U pomorstvu i zrakoplovstvu još se koriste nautička milja na sat ili čvor.

Prilikom učitavanja podataka iz GRIB datoteke na Slici 4.51, učitali smo posebno u i posebno v komponentu vjetra, koje ustvari predstavljaju x i y komponentu rezultantnog vektora. Za izračunavanje rezultantnog vektora vjetra koristimo se Pitagorovim poučkom (4.9) koji je primijenjen odmah nakon učitavanja podataka u polje na Slici 4.51.

$$wind = \sqrt{uwind^2 + vwind^2} \tag{4.9}$$

Za razliku od prognostičke karte temperature ili tlaka, na kartama prizemnog vjetra osim brzine vjetra, moramo ucrtati i njegov smjer. Smjer se najčešće crta pomoću posebnih oznaka ili vektora. Crtanje kontura, linija i dodatnog teksta provode se na sličan način kao i ranije.

```
# kreiramo vektorsko polje
uproj, vproj, xx, yy = m.transform_vector(uwind, vwind, lon[0],
    lat[:,0], 20, 20, returnxy=True)
# ucrtavamo vektorsko polje na kartu
Q = m.quiver(xx, yy, uproj, vproj)
```

Slika 4.64: Programske naredbe za crtanje vektora na kartu.

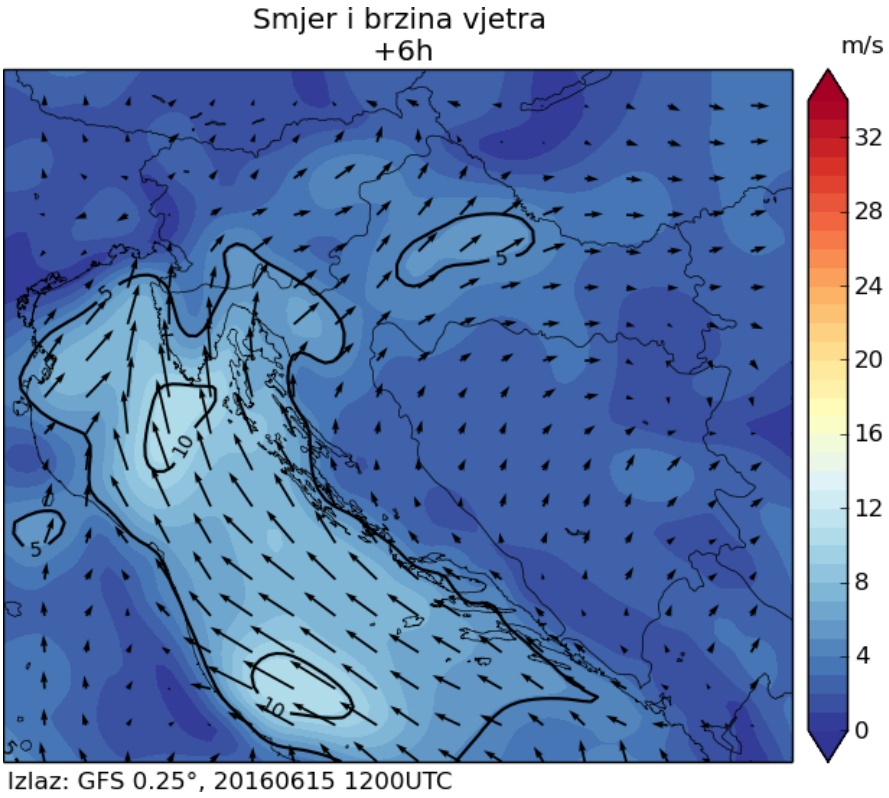
Unutar paketa Matplotlib postoje funkcije koje kreiraju vektor i ucrtavaju ga, a prikazane su na Slici 4.64. Funkcija *transform_vector* rotira i interpolira vektorsko polje za danu mrežu geografskih koordinata. Argumenti koji su potrebni funkciji za rad s vektorima su *u* i *v* komponente vjetra koje smo ranije učitali te geografske dužine i geografske širine polja. Možemo još definirati dimenzije vektorskog polja kojeg želimo kreirati, u našem slučaju je to 20x20. Dodatno, postavljajući argument *returnxy* želimo da nam se vektorsko polje prilagodi dimenzijama za crtanje na karti. Funkcija nam vraća 4 polja s podacima za crtanje na karti.

```
cs = m.contourf(x, y, wind, np.linspace(0, 35, 35), cmap=plt.cm.
    RdYlBu_r, extend='both')
cb = m.colorbar(cs, spacing='proportional', boundaries='bounds',
    format='%1i')
m.drawcoastlines(linewidth=0.5)
m.drawcountries()

linije = m.contour(x, y, wind, np.arange(0, 35, 5), linewidths
    =1.5, colors='black')
plt.clabel(linije, inline=1, fontsize=10, fmt='%1.0f')

uproj, vproj, xx, yy = m.transform_vector(uwind, vwind, lon[0],
    lat[:,0], 20, 20, returnxy=True, masked=True)
Q = m.quiver(xx, yy, uproj, vproj)

plt.title('Smjer i brzina vjetra\n +' + str(termin) + 'h')
cb.set_label('m/s', labelpad=-10, y=1.05, rotation=0)
plt.figtext(.15, .07, 'Izlaz: GFS 0.25°, ' + str(datum) + ' ' +
    str(sat) + 'UTC')

plt.savefig('vjetar.png')
```

Slika 4.65: Program za crtanje svih elemenata prognostičke karte prizemnog vjetra.

Da bi vektori konačno bili i ucrtani na karti, koristimo se funkcijom *quiver*. Funkcija uzima 4 polja kao argumente, koje je prethodno kreirala funkcija *transform_vector*. Nakon izvršenja funkcije *quiver*, na karti se ucrtaju vektori. Za dobivanje prognostičke karte prizemnog vjetra koristimo program sa Slike 4.65, dok je rezultat

dostupan na slici 4.66. Programski kod je funkcionalan ako su ranije već učitani potrebni moduli i podaci.



Slika 4.66: Prognostička karta prizemnog vjetra dobivena pomoću programa sa Slike 4.65.

4.6 *Ukupna naoblaka*

Naoblaka, kao količina oblaka koja pokriva nebo, procjenjuje se u osminama ili desetinama neba. Potpuno vedro nebo prikazano je s 0, a potpuno oblačno s 8/8 ili 10/10. Kod opažanja, tj. određivanja naoblake, sve oblake koji se u trenutku motrenja nalaze na nebu treba u mislima skupiti na jedno mjesto i procijeniti koliki dio neba oni pokrivaju [2].

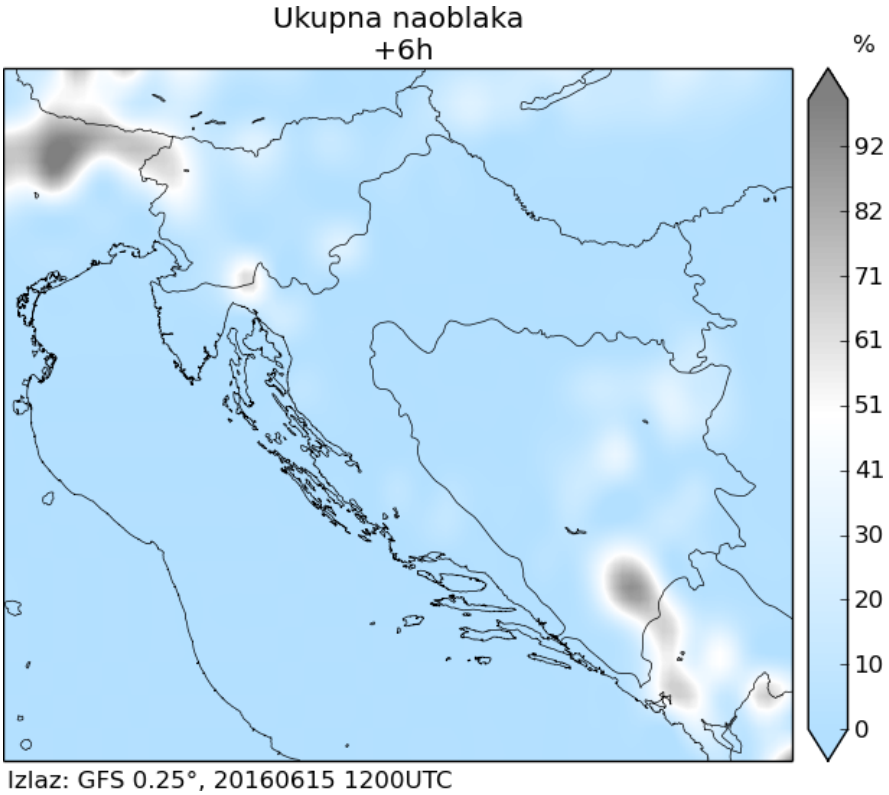
```
# kreiranje vlastite skale boja
cloud_color = mpl.colors.LinearSegmentedColormap.from_list('cloud'
    ,['#b3e0ff','white','gray'])
```

Slika 4.67: Naredba za kreiranje vlastite prilagođene skale boja.

Prognostički model, odnosno GRIB datoteka, sadržava više podataka o naoblaci koju možemo podijeliti na visoku, srednju, nisku, konvektivnu i ukupnu naoblaku. Prilikom crtanja prognostičke karte naoblake, korišteni su podaci o ukupnoj naoblaci koja se izražava u postocima, pri čemu 0% označava da nema naoblake, dok 100%

```
cloud_color = mpl.colors.LinearSegmentedColormap.from_list('cloud'
    ,['#b3e0ff','white','gray'])
cs = m.contourf(x, y, cloud, np.linspace(0,100,100), cmap=
    cloud_color, extend='both')
cb = m.colorbar(cs, spacing='proportional', boundaries='bounds',
    format='%1i')
m.drawcoastlines(linewidth=0.5)
m.drawcountries()

plt.title('Ukupna naoblaka\n +' + str(termin) + 'h')
cb.set_label('%', labelpad=-10, y=1.05, rotation=0)
plt.figtext(.15, .07, 'Izlaz: GFS 0.25°, ' + str(datum) + ' ' +
    str(sat) + 'UTC')

plt.savefig('naoblaka.png')
```

Slika 4.68: Program za crtanje svih elemenata prognostičke karte ukupne naoblake.



Slika 4.69: Prognostička karta ukupne naoblake dobivena pomoću programa sa Slike 4.68.

označava da je nebo u potpunosti pokriveno oblacima. Za potrebe ucrtavanja ukupne naoblake na kartu, kreirali smo vlastitu skalu boja pomoću naredbe na Slici 4.67. Vlastita skala boja ima definirane tri boje; početnu svijetlo plavu (označenu s HTML kodom boje), bijelu, i konačnu sivu. Bijela i siva boja definirane su engleskim nazivima tih boja, kako je to ranije opisano u Poglavlju 3.4.

Konačno, koristeći programski kod sa Slike 4.68 kreirali smo prognostičku kartu ukupne naoblake koja je dostupna na Slici 4.69. Programski kod je funkcionalan ako su ranije već učitani potrebni moduli i podaci.

4.7 Količina oborine

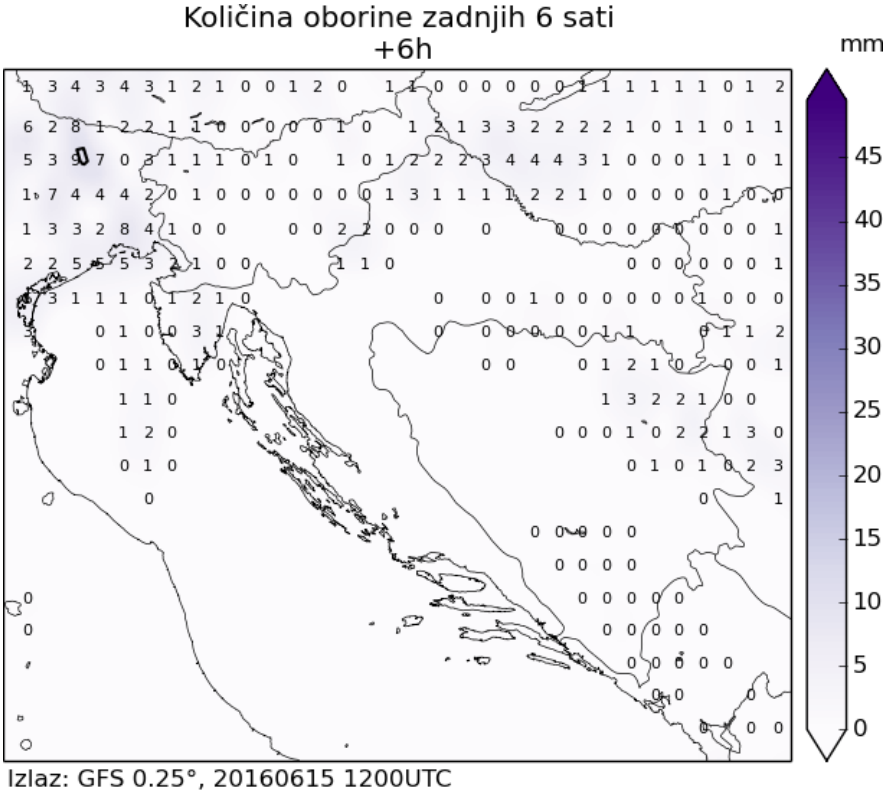
Oborina (hidrometeor) se u pravilu mjeri količinom vodenog taloga koji padne na ravnu podlogu pri Zemljinoj površini u nekom razdoblju [2]. Pri tome se misli i na tekuće i na krute oborine. Meteorološke postaje najčešće mjere količinu oborine u zadnja 24 sata ili zadnjih 6 sati. Jednica oborine je njena visina i obično se izražava u milimetrima, što odgovara broju litara vode na jediničnoj površini (1 metar kvadratni).

```
cs = m.contourf(x, y, precip, np.linspace(0,50,100), cmap=plt.cm.
    Purples, extend='both')
cb = m.colorbar(cs, spacing='proportional', boundaries='bounds',
    format='%1i')
m.drawcoastlines(linewidth=0.5)
m.drawcountries()

linije = m.contour(x, y, precip, np.arange(10, 100, 10),
    linewidths=1.5, colors='black')
plt.clabel(linije, inline=1, fontsize=10, fmt='%1.0f')

for i, j in itertools.product(range(3, 63, 3), range(3, 99, 3)):
    if precip[i][j] > 0.2:
        plt.text(x[i][j], y[i][j], repr(int(precip[i][j])),
    fontsize=8,
                ha='center', va='center', color='black')


plt.title('Kolicina oborine zadnjih 6 sati\n +' + str(termin) + 'h
    ')
cb.set_label('mm', labelpad=-10, y=1.05, rotation=0)
plt.figtext(.15, .07, 'Izlaz: GFS 0.25°, ' + str(datum) + ' ' +
    str(sat) + 'UTC')

plt.savefig('oborine.png')
```

Slika 4.70: Program za crtanje svih elemenata prognostičke karte količine oborine.

GFS prognostički model, a time i njegove GRIB datoteke, sadržavaju podatke o oborinama svakih 6 sati, 3 sata ili 1 sat, ovisno o terminu oborina kojeg želimo promatrati. Naša prognostička karta količine oborina prikazuje očekivanu količinu oborina u 6-satnom terminu i dostupna je na Slici 4.71. Program za kreiranje prognostičke karte količine oborina prikazan je na Slici 4.70 i funkcionalan je ukoliko su

ranije već učitani svi moduli i potrebni podaci. Količina oborina koja je prikazana na karti podrazumijeva količinu oborina u periodu od 6 sati do termina prikazanog na karti. U našem slučaju to je količina oborina koja je pala od početnog termina (inicijalizacije modela) do 6 sati unaprijed.



Slika 4.71: Prognostička karta očekivane količine oborina u 6-satnom terminu dobivena pomoću programa sa Slike 4.70.

5 Zaključak

Diplomski rad započet je kratkim uvodom u meteorologiju i povijesnim pregledom meteorologije. Osim što vrijeme možemo na oko opažati i osjetiti, meteorološke elemente možemo bilježiti i numerički. Za to su nam potrebni meteorološki instrumenti. Daljnjim razvojem meteorologije kao znanosti, znanstvenici su opisali atmosferu sustavom nelinearnih parcijalnih diferencijalnih jednadžbi pomoću kojih danas možemo (koristeći računala) relativno precizno predvidjeti vrijeme. Predviđanje vremena iz godine u godinu postaje sve preciznije zahvaljujući sve jačim računalima i sve boljim algoritmima, ali i zbog boljeg poznavanja početnog stanja. Prije više desetaka godina nismo mogli ni razmišljati o satelitima koji će kružiti oko Zemlje i u svakom trenutku slati snimke naoblake, ali i mnogo konkretnije podatke: primjerice podatke o temperaturi mora, temperaturi zraka i sl. S obzirom da je mreža meteoroloških postaja na Zemlji relativno rijetka (treba uzeti u obzir velike površine oceana bez meteoroloških mjerenja ili nenaseljene i teško pristupačne dijelove Zemlje), sateliti imaju veliku važnost u praćenju i prognoziranju vremena. U budućnosti valja očekivati da će sustav praćenja i bilježenja vremena dalje napredovati, a sve u cilju što boljeg poznavanja planeta Zemlje na kojem živimo. Treba spomenuti i razvoj računala koji i dalje napreduje, razvoj novih programskih sustava i boljih algoritama koji na kraju rezultiraju i sve preciznijom vremenskom prognozom.

Osim računala, značajno se razvijaju i programski jezici. Dostupno je sve više programskih paketa, a zbog interneta svatko može besplatno naučiti i dalje razvijati programske kodove. Upravo je to jedan od glavnih razloga popularnosti programskog jezika Python koji je korišten kao temelj u ovom diplomskom radu prilikom izrade meteoroloških prognostičkih karata. Korištenje dodatnih Pythonovih biblioteka (modula) i velika dostupnost uputa za korištenje na internetu značajno su olakšali proces izrade prognostičkih karata.

Treba spomenuti i američku agenciju NOAA-u, koja veliki broj informacija i podataka pruža besplatno. Ti podaci su korišteni u ovom diplomskom radu. Besplatno su dostupni i produkti izračuna globalnog prognostičkog modela GFS na kojem se prognostičke karte temelje. GFS je jedan od najpoznatijih prognostičkih modela u svijetu kojeg koriste i brojni državni meteoroloških zavodi. Besplatnim davanjem informacija NOAA potiče razvoj meteorologije i srodnih znanosti te osvještava čovječanstvo o važnosti meteorologije, posebice klimatskih promjena.

Klimatske promjene sve su češća tema razgovora, pošto nas dovode do pitanja opstanka čovječanstva, ili do izraženih poremećaja na Zemlji. Pritom se naglasak najviše stavlja na problem nedostatka hrane, koji bi mogao postati sve veći ukoliko čovječanstvo odgovorno ne prihvati činjenicu da se klima mijenja, i da se tome treba prilagoditi.

Prognostičke karte temeljene na GFS modelu javno su dostupne na web stranicama Istrameta, `http://www.istramet.hr/modeli/gfs-hrvatska/`, udruge koja

se bavi popularizacijom meteorologije i približavanjem meteorologije kao znanosti pučanstvu. Meteorološke informacije danas su vrlo tražene i lako dostupne, posebice preko mobilnih aplikacija i brojnih internet stranica. U tom moru informacija treba znati koji su podaci relevantni kako bi se spriječile neželjene posljedice i moguća zloupotreba. GFS prognostičke karte treba koristiti odgovorno, u kombinaciji s prognostičkim modelima Državnog hidrometeorološkim zavodima, kao i njihovom službenom prognozom.

Na kraju možemo zaključiti da se programski jezik Python može koristiti za vizualizaciju podataka na geografskim kartama, a time i za prikaz meteoroloških elemenata za dobivanje prognostičkih karata. Dodatno možemo interpolirati podatke radi dobivanja boljeg prikaza na kartama, a lako se provodi i čitanje podataka iz gotovo svih tipova datoteka, u našem slučaju iz GRIB datoteka.

U daljnjem radu s Pythonom planirana je vizualizacija, odnosno izrada prognostičkih karata koje će uzimati podatke ostalih prognostičkih modela, primjerice ECMWF-a ili UKMO-a. Također će se izraditi prognoza po lokacijama kada će se na jednoj slici moći pregledati svi meteorološki elementi za svih 10 dana unaprijed za neku točku na Zemlji zadanu preko geografske dužine i geografske širine. Kasnije se očekuje i izrada lokalnog modela za područje Hrvatske koji će uzimati ulazne podatke iz globalnog modela te će se provoditi vlastiti izračuni prognoze vremena.

Dodatak

Dodatak A Meteorologija u školama

Učenici se prvi put u školama susreću s meteorologijom već u nižim razredima u sklopu prirode i društva, kada se spominju pojmovi vrijeme i klima te osnovni procesi u atomosferi poput vjetra. Tu spominju najpoznatije vjetrove na Jadranu, poput bure, juga ili maestrala. Iz dodatne nastave u sklopu projekta moguće je pratiti vrijeme u svom mjestu i svakodnevno bilježiti temperaturu zraka i vremenske pojave (naoblaku, kišu, vjetar, ...) u 13 sati i tako kroz mjesec dana, čime se kod djece razvija sposobnost opažanja i bilježenja pojava oko njih. Time učenici imaju aktivnost koju moraju provoditi i tijekom vikenda ili praznika, kada inače nema nastave, čime se kod učenika postiže odgovornost i spremnost za rad i izvan nastave.

Konkretniji doticaj s meteorologijom učenici imaju tijekom 5. razreda osnovne škole iz geografije. Osim detaljnijih definicija vremena i klime, u tom razredu se spominju i pojmovi poput ciklona i anticiklona, koje učenici moraju povezati s područjem niskog, odnosno visokog tlaka zraka. Detaljnije se uči o svim vremenskim pojavama te se dublje ulazi u razmatranje klime. Očekuje se od učenika da povezuju promjenu godišnjih doba s nagibom Zemlje i njezinom revolucijom oko Sunca. Također učenici moraju prepoznati i objasniti zbog čega dolazi do izmjena dana i noći, i kako sve to utječe na vrijeme i klimu. Detaljnije se spominju meteorološki instrumenti za mjerenje, poput termometra, higrometra, barometra i kišomjera. U daljnjem školovanju do kraja osnovnog obrazovanja, prvenstveno iz geografije, učenici spominju klimatske karateristike svih dijelova Zemlje, te moraju objasniti što je uzrok različitih tipova klime na Zemlji. Povezuje se klima s vegetacijom, poljoprivredom i čovjekovim životnim navikama. Pritom se često vremenske prilike povezuju i s ostalim predmetima u školi, kao što su npr. priroda i povijest. Znamo da su nevremena često remetila povijesni tijek ratova i bila uzrok seoba naroda.

U višim razredima osnovnih škola učenici se mogu uključiti u projekt GLOBE. Zamisao o znanstveno obrazovnom programu GLOBE (Globalno učenje i opažanje za dobrobit okoliša) obznanio je na Dan planeta Zemlje 1994. godine tadašnji američki potpredsjednik Al Gore. Na isti je dan (22. travnja) 1995. godine program pokrenut u SAD-u, a vlade mnogih zemalja diljem svijeta tih su dana s američkim predstavnicima potpisale sporazume o provođenju programa GLOBE. Hrvatska je bila među prvim zemljama koje su pristupile ostvarivanju tog svjetskog programa (sporazum je potpisan 13. travnja 1995.) [12].

Danas je u projekt GLOBE uključeno 226 škola u Hrvatskoj i ukupno 309 profesora. Od početka provođenja projekta u Hrvatskoj, u projektu GLOBE sudjelovalo je više od 5100 učenika, a ukupno je upisano više od 7 milijuna podataka. Škole u Hrvatskoj koje su uključene u provođenje projekta GLOBE prikazane su na

Slici A.1.

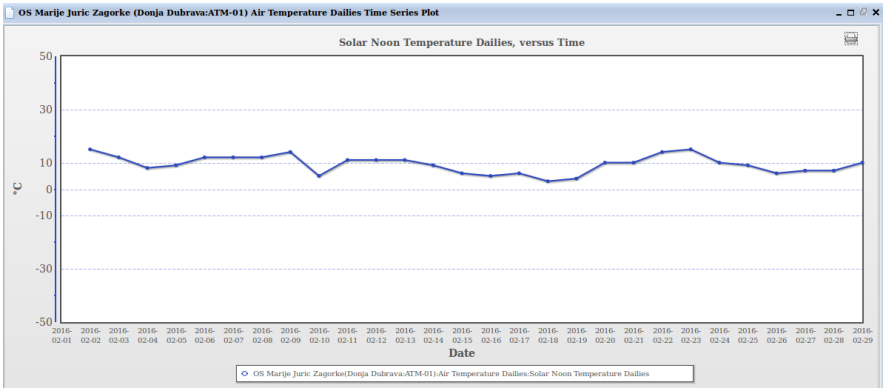
Slika A.1: Škole u Hrvatskoj koje su uključene u projekt GLOBE. Preuzeto s: `http://www.globe.gov/globe-community/community-map`

U sklopu GLOBE projekta učenici u dogovoru s profesorima nekoliko puta unutar tjedna očitavaju neke meteorološke elemente, bilježe ih i potom unose u bazu podataka. Najčešće se provodi očitanje temperature zraka u 13 sati (Slika A.2), minimalne temperature, maksimalne temperature te količinu oborina koja je pala od prošlog očitanja. Dodatno, ukoliko škole posjeduju odgovarajuću opremu, učenici mogu mjeriti i podatke poput tlaka zraka, temperaturu vode, temperaturu tla, pH vrijednost tla, itd. Učenici pritom razvijaju sposobnost očitavanja podataka s mjernih instrumenata, analiziranja podataka te unošenja u bazu podataka čime je cijeli projekt dodatno povezan s informatikom. Sudjelujući u programu učenici na konkretnim primjerima primjenjuju školska teorijska znanja te iskustvenim učenjem stječu nove spoznaje o cjelovitosti okoliša, razvijajući pritom pozitivne stavove, ali i samosvijest temeljenu na svom aktivnom sudjelovanju [12].

Povodom 20. godišnjice programa GLOBE u Hrvatskoj je 13. studenoga 2015. održana konferencija u Zagrebu. Sudjelovalo je 70 GLOBE učitelja i nastavnika te 15 uzvanika. Svoja iskustva o utjecaju programa GLOBE na kvalitetu obrazovanja u našim školama prikazalo je 16 GLOBE voditelja [13].

U srednjoškolskom obrazovanju učenici uče još detaljnije o meteorologiji, posebice iz geografije u općim gimnazijama. Učenici strukovih škola više nemaju doticaj s meteorologijom, ali se i dalje mogu uključiti u GLOBE program koji djeluje i u srednjim školama.

Smatram da bi trebalo meteorološka mjerenja više povezati s tehničkim predme-



Slika A.2: Grafički prikaz temperature zraka u 13 sati za veljaču 2016. godine koju su mjerili učenici uključeni u GLOBE projekt u OŠ Marije Jurić Zagorke u Zagrebu. Preuzeto s: `http://vis.globe.gov/GLOBE/?site_id=5502`

tima, posebice prilikom izrade vlastitih mjernih instrumenata (termometra, barometra, higrometra). Iako bi izrada profesionalnih instrumenata bila prilično složena, postoje brži i jeftiniji načini izrade. Takvi instrumenti su još uvijek dovoljno praktični da se mogu koristiti u nastavi, pa čak i u GLOBE programu. Učenike bi trebalo dodatno potaknuti na uključenje u GLOBE projekt i proširiti mrežu meteoroloških postaja, dok bi škole trebale osigurati svu potrebnu opremu kako se mjerenja ne bi prekidala u slučaju kvara ili oštećenja mjernog instrumenta.

Literatura

[1] Sijerković, M. Istarska meteorološka škrinjica. Buzet: Reprezent, 2008.

[2] Gelo, B. Opća i pomorska meteorologija. Zadar: Sveučilište u Zadru - Odjel za promet i pomorstvo, 2010.

[3] Budin, L.; Brođanac, P.; Markučič, Z.; Perić, S. Rješavanje problema programiranjem u Pythonu. 2. izdanje. Zagreb: Element, 2013.

[4] Groš S.; Kalafatić Z.; Šegvić S. Uvod u programski jezik Python, https://www.fer.unizg.hr/_download/repository/Skriptni_3_Python[5].pdf, 07.06.2016.

[5] Scott Shell, M. An introduction to Numpy and Scipy, http://www.engr.ucsb.edu/~shell/che210d/numpy.pdf, 09.06.2016.

[6] Više autora. Multi-dimensional image processing, http://docs.scipy.org/doc/scipy/reference/ndimage.html#module-scipy.ndimage, 11.06.2016.

[7] Hunter, J.; Dale, D.; Firing, E.; Droettboom, M.; Matplotlib, http://matplotlib.org/, 12.06.2016.

[8] Whitaker, J. Matplotlib Basemap Toolkit, http://matplotlib.org/basemap/, 13.06.2016.

[9] Whitaker, J. Module Pygrib, https://github.com/jswhit/pygrib, 14.06.2016.

[10] Više autora. Itertools — Functions creating iterators for efficient looping, https://docs.python.org/3.1/library/itertools.html, 15.06.2016.

[11] Više autora. Global Forecast System - GFS, https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs, 25.06.2016.

[12] Više autora. GLOBE Hrvatska: O programu GLOBE, http://globe.pomsk.hr/GLOBEinfo.htm, 03.07.2016.

[13] Više autora. The GLOBE program, http://www.globe.gov/, 03.07.2016.