Sadržaj

Predgovor ..... **ix**

1. Uvod u Python **1**
Jezik Python 1
Python standardna biblioteka i moduli proširenja 2
Python implementacije 3
Python razvoj i verzije 10
Python resursi 11
Instalacija 15
Instaliranje Pythona iz binarnih datoteka 16
Instaliranje Pythona iz izvornog koda 17

2. Python interpreter **21**
Program python 21
Python razvojna okruženja 27
Izvršavanje Python programa 29
Izvršavanje Pythona u veb čitaču 30

3. Jezik Python **33**
Leksička struktura 33
Tipovi podataka 40
Promenljive i druge reference 52
Izrazi i operatori 56
Numeričke operacije 60
Operacije nad nizovima 62
Operacije nad skupovima 68
Operacije nad rečnikom 70
Naredbe toka kontrole 72
Funkcije 91

4. Objektno orijentisan Python **113**
Klase i instance 113
Specijalne metode 139
Dekoratori 152
Metaklase 153

5. Anotacije tipa 165
Razvoj 165
Uslužni programi za proveru tipa 167
Sintaksa anotacije tipa 168
Modul typing 170
Korišćenje anotacija tipa tokom izvršavanja 184
Kako dodati anotacije tipa svom kodu 185
Rezime 187

6. Izuzeci 189
Naredba try 189
Naredba raise 194
Naredba with i menadžer konteksta 195
Generatori i izuzeci 197
Propagacija izuzetaka 198
Objekti izuzetaka 199
Prilagođene klase izuzetaka 204
ExceptionGroup i except* 206
Strategije provere grešaka 207
Naredba assert 212

7. Moduli i paketi 215
Objekti modula 216
Učitavanje modula 221
Paketi 227
Distribucioni pomoćni programi (distutils) i setuptools 228
Python okruženja 230

8. Ugrađeni moduli jezgra i moduli standardne biblioteke 239
Ugrađeni tipovi 240
Ugrađene funkcije 243
Modul sys 249
Modul copy 252
Modul collections 253
Modul functools 258
Modul heapq 260
Modul argparse 262
Modul itertools 264

9. Stringovi i ostalo 269
Metode string objekata 269
Modul string 273
Formatiranje stringa 273

Prelamanje i popunjavanje teksta 286
Modul pprint 286
Modul reprlib 287
Unicode 287

10. Regularni izrazi 289
Regularni izrazi i re modul 289
Opcione zastavice 295
Podudaranje naspram pretrage 296
Sidrenje na početku i kraju stringa 297
Objekti regularnog izraza 297
Podudaranje objekata 300
Funkcije modula re 301
RE i operator := 302
Regex modul treće strane 303

11. Operacije sa datotekama i tekstom 305
Modul io 306
Modul tempfile 312
Pomoćni moduli za ulaz/izlaz datoteka 313
Datoteke u memoriji: io.StringIO i io.BytesIO 317
Arhivirane i komprimovane datoteke 318
Modul os 325
Modul errno 336
Modul pathlib 336
Modul stat 340
Modul filecmp 341
Modul fnmatch 343
Modul glob 344
Modul shutil 344
Unos i izlaz teksta 346
Ulaz/Izlaz teksta sa bogatijim formatiranjem 348
Internacionalizacija 351

12. Postojanost i baze podataka 359
Serijalizacija 360
Moduli DBM 371
API Python baze podataka (DBAPI) 373

13. Vremenske operacije 383
Modul time 383
Modul datetime 387
Modul zoneinfo 393
Modul dateutil 395

Modul sched 395
Modul calendar 397

14. Prilagođavanje izvršavanja ... 399
Prilagođavanje po lokaciji 399
Funkcije završetka 400
Dinamičko izvršavanje i exec 400
Interni tipovi 404
Sakupljanje otpada 405

15. Uporednost: niti i procesi ... 411
Niti u Pythonu 413
Modul threading 414
Modul queue 423
Modul multiprocessing 425
Modul concurrent.futures 434
Arhitektura programske niti 436
Procesno okruženje 440
Pokretanje drugih programa 441
Modul mmap 445

16. Numerička obrada ... 451
Vrednosti sa pokretnim zarezom 451
Moduli math i cmath 454
Modul statistics 457
Modul operator 458
Nasumični i pseudonasumični brojevi 460
Modul fractions 463
Modul decimal 464
Obrada niza 466

17. Testiranje, otklanjanje grešaka i optimizacija ... 475
Testiranje 476
Otklanjanje grešaka 489
Modul warnings 498
Optimizacija 501

18. Osnove umrežavanja ... 523
Interfejs utičnice Berkeley 524
Bezbednost transportnog sloja 537
SSLContext 538

19. Moduli mrežnog protokola na strani klijenta 541
Protokoli e-pošte 541
HTTP i URL klijenti 544
Drugi mrežni protokoli 551

20. Uslužni HTTP 553
http.server 554
WSGI 554
Python veb okvir 555

21. E-pošta, MIME i druga mrežna kodiranja 567
Upravljanje MIME i formatom e-pošte 567
Kodiranje binarnih podataka kao ASCII teksta 574

22. Strukturisan tekst: HTML 579
Modul html.entities 580
Paket treće strane BeautifulSoup 580
Generisanje HTML-a 591

23. Strukturisan tekst: XML 597
ElementTree 598
Parsiranje XML-a pomoću ElementTree.parse 603
Izgradnja strukture ElementTree od nule 605
Iterativno raščlanjivanje XML-a 605

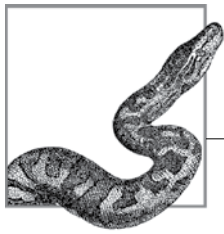
24. Programi za pakovanje i proširenja 609
Šta ne pokrivamo u ovom poglavlju 610
Kratka istorija Python pakovanja 611
Onlajn materijal 612

25. Proširivanje i ugradnja klasičnog Pythona 613
Onlajn materijal 614

26. Migracija sa v3.7 na v3.n 615
Značajne promene u Pythonu do 3.11 615
Planiranje nadogradnje Python verzije 617
Rezime 620

Dodatak Nove funkcije i promene u Pythonu od 3.7 do 3.11 621

Indeks 637



Predgovor

Programski jezik Python miri mnoge očigledne kontradikcije: elegantan ali pragmatičan, jednostavan ali moćan, veoma je visokog nivoa, ali vas ne ometa kada treba da bavite sa bitovima i bajtovima. Pogodan je za programere početnike, ali i odličan za stručnjake.

Ova knjiga je namenjena programerima koji su prethodno bili upoznati sa Pythonom, kao i iskusnim programerima koji prvi put dolaze na Python sa drugih jezika. Pruža brzu referencu na sam Python, najčešće korišćene delove njegove ogromne standardne biblioteke i nekoliko najpopularnijih i korisnih modula i paketa nezavisnih proizvođača. Python ekosistem je toliko porastao u bogatstvu, obimu i složenosti da se više ne može očekivati da će jedna knjiga biti enciklopedijska. Ipak, knjiga pokriva širok spektar oblasti primene, uključujući veb i mrežno programiranje, rukovanje XML-om, interakcije baze podataka i brzo numeričko računarstvo. Takođe istražuje Pythonove višeplatformske mogućnosti i osnove proširenja Pythona i njegovog ugrađivanja u druge aplikacije.

Kako koristiti ovu knjigu

Iako ovu knjigu možete da čitate redom od početka, želimo da bude korisna referenca za programere u svakodnevnom radu. Možete izabrati da koristite indeks za lociranje tema od interesa ili da pročitate određena poglavlja za pokrivanje njihovih posebnih tema. Kako god da je koristite, iskreno se nadamo da ćete uživati čitajući plod jednogodišnjeg rada tima.

Knjiga ima pet delova, ovim redom.

Deo I, Prvi koraci sa Pythonom

Poglavlje 1, „Uvod u Python“

Pokriva opšte karakteristike jezika Python, njegove implementacije, gde da dobijete pomoć i informacije, kako da učestvujete u Python zajednici i kako da nabavite i instalirate Python na računar(e) ili da ga pokrenete u svom veb čitaču.

Poglavlje 2, „Python interpreter“

Pokriva program Python interpreter, njegove opcije komandne linije i kako ga koristiti za pokretanje Python programa i u interaktivnim sesijama. U poglavlju se pominju editori teksta za uređivanje Python programa i pomoćni programi za proveru vaših Python izvora, zajedno sa nekim punopravnim integrisanim razvojnim okruženjima, uključujući IDLE, koji dolazi besplatno sa standardnim Pythonom. Poglavlje pokriva i pokretanje Python programa iz komandne linije.

Deo II, Osnovni jezik Pythona i ugrađene komponente

Poglavlje 3, „Jezik Python“

Obuhvata Python sintaksu, ugrađene tipove podataka, izraze, naredbe, tok kontrole i objašnjenje kako pisati i pozivati funkcije.

Poglavlje 4, „Objektno orijentisan Python“

Obuhvata objektno orijentisano programiranje u Pythonu.

Poglavlje 5, „Anotacije tipa“

Obuhvata kako da dodate informacije o tipu vašem Python kodu, da dobijete nagoveštaj tipa i pomoć za autodovršavanje od savremenih uređivača koda i da podržavate statičku proveru tipa od strane provere tipa i lintera.

Poglavlje 6, „Izuzeci“

Pokriva kako da koristite izuzetke za greške i posebne situacije, evidentiranje i kako da napišete kôd za automatsko čišćenje kada dođe do izuzetaka.

Poglavlje 7, „Moduli i paketi“

Pokriva načine na koji Python omogućava grupisanje koda u module i pakete, kako da definišete i uvezete module i kako da instalirate Python pakete treće strane. Ovo poglavlje takođe sadrži rad sa virtuelnim okruženjima za izolovanje zavisnosti projekta.

Poglavlje 8, „Ugrađeni moduli jezgra i moduli standardne biblioteke“

Pokriva ugrađene tipove podataka i funkcije i neke od najosnovnijih modula u standardnoj biblioteci Pythona (grubo govoreći, skup modula koji obezbeđuje funkcionalnost koja je, na nekim drugim jezicima, ugrađena u sam jezik).

Poglavlje 9, „Stringovi i ostalo“

Pokriva Pythonove mogućnosti za obradu stringova, uključujući Unicode stringove, bajtove i literale stringova.

Poglavlje 10, „Regularni izrazi“

Obuhvata Pythonovu podršku za regularne izraze.

Deo III, Python biblioteka i moduli proširenja

Poglavlje 11, „Operacije sa datotekama i tekstom“

Pokriva rad sa datotekama i tekstom sa mnogim modulima iz Python standardne biblioteke i ekstenzijama specifičnim za platformu za I/O obogaćenog teksta. Ovo poglavlje takođe pokriva pitanja internacionalizacije i lokalizacije.

Poglavlje 12, „Postojanost i baze podataka“

Obuhvata Pythonove mehanizme serijalizacije i postojanosti i njegove interfejse za DBM baze podataka i relacione (bazirane na SQL-u) baze podataka, posebno praktičan SQLite koji dolazi sa Pythonovom standardnom bibliotekom.

Poglavlje 13, „Vremenske operacije“

Pokriva vreme i datume u Pythonu, sa standardnom bibliotekom i ekstenzijama trećih strana.

Poglavlje 14, „Prilagođavanje izvršavanja“

Pokriva načine za postizanje napredne kontrole izvršavanja u Pythonu, uključujući izvršavanje dinamički generisanog koda i kontrolu sakupljanja otpada. Ovo poglavlje takođe pokriva neke interne tipove Pythona i specifično pitanje registrovanja funkcija „čišćenja“ da se izvrše u vreme završetka programa.

Poglavlje 15, „Uporednost: niti i procesi“

Obuhvata Python funkcionalnost za istovremeno izvršavanje, kako preko više niti koje se pokreću u okviru jednog procesa tako i preko više procesa koji se pokreću na jednoj mašini.[1] Ovo poglavlje pokriva kako pristupiti okruženju procesa i kako pristupiti datotekama putem mehanizama za mapiranje memorije.

Poglavlje 16, „Numerička obrada“

Pokriva Pythonove mogućnosti za numerička izračunavanja, kako u standardnim modulima biblioteke, tako i u paketima ekstenzija nezavisnih proizvođača; posebno, kako koristiti decimalne brojeve ili razlomke umesto podrazumevanih binarnih brojeva sa pokretnim zarezom. Ovo poglavlje pokriva kako da dobijete i koristite pseudoslučajne i zaista nasumične brojeve i kako da brzo obradite čitav niz (i matrice) brojeva.

Poglavlje 17, „Testiranje, otklanjanje grešaka i optimizacija“

Obuhvata Python alate i pristupe koji vam pomažu da se uverite da su vaši programi ispravni (tj. da rade ono za što su namenjeni), da pronađete i popravite greške u vašim programima i proverite i poboljšate performanse vaših programa. Ovo poglavlje pokriva koncept upozorenja i Python modulom biblioteke koji se njima bavi.

[1] Posebno poglavlje o asinhronom programiranju u trećem izdanju je izbačeno u ovom izdanju, a detaljnija obrada ove rastuće teme je u referencama koje se nalaze u Poglavlju 15.

Deo IV, Mrežno i veb programiranje

Poglavlje 18, „Osnove umrežavanja“

Pokriva osnove umrežavanja sa Pythonom.

Poglavlje 19, „Moduli mrežnog protokola na strani klijenta“

Pokriva module u Pythonovoj standardnoj biblioteci za pisanje mrežnih klijentskih programa, posebno za rad sa različitim mrežnim protokolima sa strane klijenta, slanje i primanje e-pošte i rukovanje URL adresama.

Poglavlje 20, „Uslužni HTTP“

Pokriva kako da služi HTTP za veb aplikacije u Pythonu, koristeći popularne lagane Python okvire nezavisnih proizvođača koji koriste Pythonov WSGI standardni interfejs za veb servere.

Poglavlje 21, „E-pošta, MIME i druga mrežna kodiranja“

Pokriva kako da obrađujete poruke e-pošte i druge mrežno strukturisane i kodirane dokumente u Pythonu.

Poglavlje 22, „Strukturisan tekst: HTML“

Pokriva popularne Python module proširenja nezavisnih proizvođača za obradu, modifikaciju i generisanje HTML dokumenata.

Poglavlje 23, „Strukturisan tekst: XML“

Pokriva module Python biblioteke i popularna proširenja za obradu, modifikovanje i generisanje XML dokumenata.

Deo V, Proširivanje, distribucija i nadogradnja i migracija verzije

Poglavlja 24 i 25 uključena su u sažetom obliku u štampano izdanje ove knjige. Naći ćete pun sadržaj ovih poglavlja u pratećem onlajn skladištu, opisanom u „Kako nas kontaktirati“ na strani xv.

Poglavlje 24, „Programi za pakovanje i proširenja“

Pokriva alate i module za pakovanje i deljenje Python modula i aplikacija.

Poglavlje 25, „Proširivanje i ugradnja klasičnog Pythona“

Pokriva kako da kodirate Python module proširenja koristeći Python C API, Cython i druge alate.

Poglavlje 26, „Migracija v3.7 na v3.n“

Obuhvata teme i najbolje prakse za planiranje i primenu nadogradnje verzija za korisnike Pythona, od pojedinaca do održavalaca biblioteka do osoblja za primenu i podršku u celom preduzeću.

Dodatak, „Nove funkcije i promene u Pythonu od 3.7 do 3.11“

Pruža detaljnu listu funkcija i promena u sintaksi jezika Python i standardnoj biblioteci, po verziji.

Konvencije koje se koriste u ovoj knjizi

U ovoj knjizi se koriste sledeće konvencije.

Referentne konvencije

U referentnim unosima funkcije/metoda, kada je to izvodljivo, svaki opcioni parametar je prikazan sa podrazumevanom vrednošću koristeći Python sintaksu `ime=vrednost`. Ugrađene funkcije ne moraju da prihvataju imenovane parametre, tako da imena parametara možda neće biti značajna. Neki opcioni parametri se najbolje objašnjavaju u smislu njihovog prisustva ili odsustva, a ne kroz podrazumevane vrednosti. U takvim slučajevima, označavamo da je parametar opcioni tako što ga stavljamo u zagrade (`[]`). Kada je više od jednog argumenta opciono, zagrade mogu biti ugnežđene.

Konvencije o verzijama

Ova knjiga pokriva promene i funkcije u Python verzijama od 3.7 do 3.11.

Python 3.7 služi kao osnovna verzija za sve tabele i primere koda, osim ako nije drugačije naznačeno.[2] Videćete ove oznake koje označavaju izmene ili funkcije koje su dodate i uklonjene u nizu pokrivenih verzija:

- **3.x+** označava funkciju uvedenu u verziju 3.x, koja nije bila dostupna u prethodnim verzijama.
- **-3.x** označava funkciju koja je uklonjena u verziji 3.x, dostupna samo u prethodnim verzijama.

Tipografske konvencije

Imajte na umu da, iz razloga prikaza, naši primeri koda ponekad mogu da odstupe od PEP 8 (*https://oreil.ly/u0RLL*). Ne preporučujemo da uzimate takve slobode u svom kodu. Umesto toga, koristite uslužni program kao što je `black` (*https://oreil.ly/BM68x*) da biste usvojili kanonski stil izgleda.

U ovoj knjizi se koriste sledeće tipografske konvencije:

Italic
: Koristi se za nazive datoteka i direktorijuma, imena programa, URL-ove i za uvođenje novih termina.

`Konstantna širina`
: Koristi se za izlaz komandne linije i primere koda, kao i za elemente koda koji se pojavljuju u tekstu, uključujući metode, funkcije, klase i module.

[2] Na primer, da bi se prilagodile široko rasprostranjene promene u Pythonu 3.9 i 3.10 u napomenama tipa, većina Poglavlja 5 koristi Python 3.10 kao osnovnu verziju za funkcije i primere.

`Konstantna širina kurziv`
: Koristi se za prikaz teksta koji treba zameniti vrednostima koje je uneo korisnik u primerima koda i komandama.

`Konstantna širina podebljano`
: Koristi se za komande koje se upisuju na sistemskoj komandnoj liniji i za označavanje izlaznog koda u primerima sesije Python interpretera. Takođe se koristi za rezervisane reči za Python.



Ovaj element označava savet ili predlog.



Ovaj element označava opštu napomenu.



Ovaj element ukazuje na upozorenje ili oprez.

Upotreba primera koda

Ova knjiga je ovde da vam pomogne da završite svoj posao. Generalno, ako se uz ovu knjigu nudi primer koda, možete ga koristiti u svojim programima i dokumentaciji. Ne morate da nas kontaktirate za dozvolu osim ako ne reprodukujete značajan deo koda. Na primer, za pisanje programa koji koristi nekoliko delova koda iz ove knjige nije potrebna dozvola. Prodaja ili distribucija primera iz O'Reilly/Mikro knjiga knjiga zahteva dozvolu. Da biste odgovorili na pitanje citiranjem ove knjige i citiranjem primera koda, nije potrebna dozvola. Ugrađivanje značajne količine primera koda iz ove knjige u dokumentaciju vašeg proizvoda zahteva dozvolu.

Cenimo, ali ne zahtevamo, navođenje. Navođenje obično sadrži naslov, autora, izdavača i ISBN. Na primer: „*Python za programere*, 4. izdanje, Alex Martelli, Anna Martelli Ravenscroft, Steve Holden i Paul McGuire. Autorska prava 2023, 978-86-7555-468-4".

Ukoliko smatrate da vaše korišćenje primera koda prevazilazi fer korišćenje ili gore navedene dozvole, slobodno nas kontaktirajte na *redakcija@mikroknjiga.rs*.

Kako nas kontaktirati

Komentare i pitanja u vezi ove knjige uputite izdavaču:

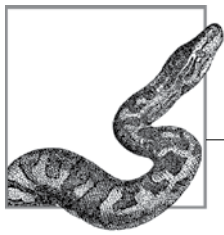
MIKRO KNJIGA, DOO
Kneza Višeslava 34
11030 Beograd, Srbija
+381 11 7702-883
redakcija@mikroknjiga.rs

Knjiga ima svoje GitHub spremište, gde navodimo greške, primere i sve dodatne informacije. Skladište takođe sadrži pun sadržaj Poglavlja 24 i 25, za koje nije bilo dovoljno prostora u štampanom izdanju. Naći ćete ga na *https://github.com/pynutshell/pynut4* (*https://github.com/pynutshell/pynut4*).

Testirali smo i verifikovali informacije u ovoj knjizi najbolje što smo mogli, ali možda ćete otkriti da su se funkcije promenile (ili čak da smo napravili greške!). Obavestite izdavača o svim greškama koje pronađete, kao i o vašim predlozima za buduća izdanja.

O'Reilly ima veb stranicu za ovu knjigu, gde su navedene greške, primeri i sve dodatne informacije. Ovoj stranici možete pristupiti na *https://oreil.ly/python-nutshell-4e* (*https://oreil.ly/python-nutshell-4e*).

Da biste komentarisali ili postavili tehnička pitanja o ovoj knjizi, pošaljite e-poštu na pinut4@gmail.com (*mailto:pinut4@gmail.com*).



1
Uvod u Python

Python je dobro uspostavljen programski jezik opšte namene, koji je prvi put objavio njegov tvorac, Guido van Rosum, 1991. Ovaj stabilan i zreo jezik je visokog nivoa, dinamičan, objektno orijentisan i višestruko platformisan – sve su ovo veoma atraktivne karakteristike. Python radi na macOS-u, najsavremenijim varijantama Unixa, uključujući Linux, Windows i, uz određena podešavanja, na mobilnim platformama.[1]

Python nudi visoku produktivnost za sve faze životnog ciklusa softvera: analizu, dizajn, izradu prototipa, kodiranje, testiranje, otklanjanje grešaka, podešavanje, dokumentaciju i, naravno, održavanje. Popularnost jezika beleži stalni rast već dugi niz godina, postajući TIOBE indeks (*https://oreil.ly/qxdeK*) lider u oktobru 2021. Danas je poznavanje Pythona plus za svakog programera: ušunjao se u većinu niša, sa korisnim ulogama u bilo kom softverskom rešenju.

Python pruža jedinstvenu mešavinu elegancije, jednostavnosti, praktičnosti i čiste snage. Brzo ćete postati produktivni sa Pythonom, zahvaljujući njegovoj doslednosti i redovnosti, njegovoj bogatoj standardnoj biblioteci i mnogim paketima i alatima nezavisnih proizvođača koji su mu lako dostupni. Python je lako naučiti, tako da je sasvim prikladan ako ste novi u programiranju, ali je i dovoljno moćan za najsofisticiranije stručnjake.

Jezik Python

Python jezik, iako nije minimalistički, štedljiv je, iz dobrih pragmatičnih razloga. Jednom kada jezik ponudi jedan dobar način izražavanja dizajna, dodavanje drugih načina ima, u najboljem slučaju, skromne prednosti; cena složenosti jezika, raste više nego linearno sa brojem karakteristika. Komplikovan jezik je teže naučiti i savladati (i implementirati efikasno i bez grešaka) nego jednostavniji. Komplikacije i nedoumice u

[1] Za Android pogledajte *https://wiki.python.org/moin/Android*, a za iPhone i iPad pogledajte Python za iOS i iPadOS (*https://oreil.ly/iYnk3*).

jeziku ometaju produktivnost u razvoju softvera, posebno u velikim projektima, gde mnogi programeri sarađuju i, često, održavaju kôd koji su prvobitno napisali drugi.

Python je prilično jednostavan, ali nije pojednostavljen. Pridržava se ideje da, ako se jezik ponaša na određen način u nekim kontekstima, idealno bi trebalo da funkcioniše slično u svim kontekstima. Python sledi princip da jezik ne bi trebalo da ima „zgodne" prečice, posebne slučajeve, ad hoc izuzetke, preterano suptilne razlike ili misteriozne i lukave optimizacije ispod haube. Dobar jezik, kao i svaki drugi dobro osmišljen artefakt, mora da uravnoteži opšte principe sa ukusom, zdravim razumom i sa puno praktičnosti.

Python je programski jezik opšte namene: njegove osobine su korisne u skoro svim oblastima razvoja softvera. Ne postoji oblast u kojoj Python ne može biti deo rešenja. „Deo" je ovde važan; dok mnogi programeri smatraju da Python ispunjava sve njihove potrebe, on ne mora da bude sam. Python programi mogu da sarađuju sa raznim drugim softverskim komponentama, što ga čini pravim jezikom za spajanje komponenti na drugim jezicima. Cilj dizajna jezika je, i dugo je bio, da se „dobro slaže sa drugima".

Python je jezik veoma visokog nivoa (very high-level language, VHLL). To znači da koristi viši nivo apstrakcije, konceptualno dalje od mašine ispod, od klasičnih kompajliranih jezika kao što su C, C++ i Rust, koji se tradicionalno nazivaju „jezici visokog nivoa". Python je jednostavniji, brži za obradu (i za ljude i za alate) i regularniji je u tome od klasičnih jezika visokog nivoa. Ovo omogućava visoku produktivnost programera, čineći Python jakim razvojnim alatom. Dobri kompajleri za klasične kompajlirane jezike mogu da generišu binarni kôd koji radi brže od Pythona. U većini slučajeva performanse aplikacija kodiranih u Pythonu su dovoljne. Kada nisu, primenite tehnike optimizacije pokrivene u „Optimizacija" da biste poboljšali performanse svog programa, a da pritom zadržite prednost visoke produktivnosti.

U pogledu nivoa jezika, Python je uporediv sa drugim moćnim VHLL-ovima kao što su JavaScript, Ruby i Perl. Prednosti jednostavnosti i pravilnosti, međutim, ostaju na strani Pythona.

Python je objektno orijentisan programski jezik, ali vam omogućava da programirate i u objektno orijentisanim i u proceduralnim stilovima, sa primesama funkcionalnog programiranja, mešanjem i usklađivanjem kako vaša aplikacija zahteva. Pythonove objektno orijentisane karakteristike su konceptualno slične onim u C++-u, ali su jednostavnije za korišćenje.

Python standardna biblioteka i moduli proširenja

Postoji više od jezika u Python programiranju: standardna biblioteka i drugi moduli proširenja su skoro jednako važni za upotrebu Pythona kao i sam jezik. Python standardna biblioteka obezbeđuje mnoge dobro dizajnirane, čvrste Python module za zgodnu ponovnu upotrebu. Uključuje module za takve zadatke kao što su predstavlja-

nje podataka, obrada teksta, interakcija sa operativnim sistemom i sistemom datoteka i veb programiranje, i radi na svim platformama koje podržava Python.

Moduli proširenja, iz standardne biblioteke ili od negde drugde, dozvoljavaju Python kodu da pristupi funkcionalnosti koju obezbeđuje osnovni operativni sistem ili druge softverske komponente, kao što su grafički korisnički interfejsi (GUI), baze podataka i mreže. Ekstenzije takođe omogućavaju veliku brzinu u računarski intenzivnim zadacima kao što su XML raščlanjivanje i izračunavanje numeričkog niza. Moduli proširenja koji nisu kodirani u Pythonu ne moraju nužno uživati istu prenosivost na više platformi kao čisti Python kôd.

Možete da pišete module proširenja na jezicima nižeg nivoa da biste optimizovali performanse za male, računarski intenzivne delove koje ste prvobitno prototipovali u Pythonu. Možete da koristite i alatke kao što su Cython, ctypes, i CFFI da omotate postojeće C/C++ biblioteke u Python module proširenja, kao što je pokriveno u „Proširivanje Pythona bez Pythonovog C API-ja“ u Poglavlju 25 (dostupno onlajn (*https://oreil.ly/python-nutshell-25*)). Možete da ugradite Python u aplikacije kodirane na drugim jezicima, izlažući funkcionalnost aplikacije Pythonu preko Python modula proširenja specifičnih za aplikaciju.

Ova knjiga dokumentuje mnoge module, iz standardne biblioteke i drugih izvora, za mrežno programiranje na strani klijenta i servera, baze podataka, obradu tekstualnih i binarnih datoteka i interakciju sa operativnim sistemima.

Python implementacije

U u vreme pisanja ovog teksta, Python ima dve implementacije potpune proizvodnje (CPython i PyPy) i nekoliko novijih, visokih performansi u nešto ranijim fazama razvoja, kao što je kao Nuitka (*https://nuitka.net*), RustPython (*https://oreil.ly/1oUWk*), GraalVM Python (*https://oreil.ly/1XRt_*), i Pyston (*https://www.pyston.org*), koje dalje ne pokrivamo. U „Drugi razvoji, implementacije i distribucije“ pominjemo neke druge, čak i ranije faze implementacije.

Ova knjiga se prvenstveno bavi CPythonom, najšire korišćenom implementacijom, koju zbog jednostavnosti često nazivamo samo „Python“. Međutim, razlika između jezika i njegovih implementacija je važna!

CPython

Klasični Python (*https://www.python.org*) – poznat kao CPython, koji se često naziva samo Python – je najažurniji, solidan, i potpuna implementacija Pythona proizvodnog kvaliteta. To je „referentna implementacija“ jezika. CPython je kompajler bajtkoda, interpreter i skup ugrađenih i opcionih modula, svi kodirani u standardnom C.

CPython se može koristiti na bilo kojoj platformi gde je C kompajler usaglašen sa standardom ISO/IEC 9899:1990[2] (tj. sve moderne, popularne platforme). U „Instalacija“ na strani 15 objašnjavamo kako da preuzmete i instalirate CPython. Cela ova knjiga, osim nekoliko delova koji su eksplicitno označeni drugačije, odnosi se na CPython. U vreme ovog pisanja, tekuća verzija CPythona, upravo objavljena, je 3.11.

PyPy

PyPy (*https://pypy.org*) je brza i fleksibilna implementacija Pythona, kodirana u podskupu samog Pythona, koja može da cilja nekoliko jezika nižeg nivoa i virtuelnih mašina koristeći napredne tehnike kao što je zaključivanje tipa. Najveća snaga PyPya je njegova sposobnost da generiše izvorni mašinski kod „baš na vreme“ dok izvršava vaš Python program; ima značajne prednosti u brzini izvršenja. PyPy trenutno implementira 3.8 (sa 3.9 u beta verziji).

Biranje između CPython, PyPy i drugih implementacija

Ako vaša platforma, kao i većina, može da pokreće CPython, PyPy i nekoliko drugih Python implementacija koje pominjemo, kako birate između njih? Pre svega, nemojte birati prerano: preuzmite i instalirajte ih sve. Oni koegzistiraju bez problema i svi su besplatni (neki od njih nude i komercijalne verzije sa dodatnom vrednošću, kao što je tehnička podrška, ali i odgovarajuće besplatne verzije su u redu). Imati ih sve na vašoj razvojnoj mašini košta samo malo vremena za preuzimanje i malo prostora na disku, i omogućava vam da ih direktno uporedite. Ipak, evo nekoliko opštih saveta.

Ako vam je potrebna prilagođena verzija Pythona ili visoke performanse za dugoizvršavajuće programe, razmislite o PyPyu (ili, ako su vam u redu verzije koje još nisu sasvim spremne za proizvodnju, onda neku od ostalih koje pominjemo).

Za rad uglavnom u tradicionalnom okruženju, CPython se odlično uklapa. Ako nemate jake alternativne preferencije, počnite sa standardnom implementacijom CPython reference, koja je najšire podržana od strane dodataka i proširenja trećih strana i nudi najnoviju verziju.

Drugim rečima, da biste eksperimentisali, učili i isprobavali stvari, koristite CPython. Za razvoj i primenu, vaš najbolji izbor zavisi od modula proširenja koje želite da koristite i načina na koji želite da distribuirate svoje programe. CPython, po definiciji, podržava sve Python ekstenzije; međutim, PyPy podržava većinu ekstenzija i često može biti brži za dugoizvršavajuće programe zahvaljujući kompilaciji u mašinskom kodu – na vreme da biste to proverili, uporedite svoj CPython kôd sa PyPyom (i, da budete sigurni, i sa drugim implementacijama).

CPython je najzreliji: postoji dugo, dok su PyPy (i ostali) noviji i manje dokazani u ovoj oblasti. Razvoj CPython verzija napreduje ispred drugih implementacija.

[2] Python verzije od 3.11 koriste „C11 bez opcionih mogućnosti“ i navode da „javni API treba da bude kompatibilan sa C++“.

PyPy, CPython i druge implementacije koje pominjemo su sve dobre, verne implementacije Pythona, relativno bliske jedna drugoj u smislu upotrebljivosti i performansi. Mudro je upoznati se sa prednostima i slabostima svakog od njih, a zatim izabrati optimalan za svaki razvojni zadatak.

Drugi razvoji, implementacije i distribucije

Python je postao toliko popularan da se nekoliko grupa i pojedinaca zainteresovalo za njegov razvoj i obezbedili su funkcije i implementacije van fokusa glavnog razvojnog tima.

Danas, većina sistema zasnovanih na Unixu uključuje Python – tipično verziju 3.*x* za neku vrednost *x*– kao „sistemski Python". Da biste dobili Python za Windows ili macOS, obično preuzimate i pokrećete instalacioni program (*https://oreil.ly/c-TxU*) (pogledajte i „macOS" na strani 16.) Ako ste ozbiljni u razvoju softvera na Pythonu, prva stvar koju treba da uradite je da *ostavite svoj sistemski instaliran Python na miru!* Sasvim drugačije od bilo čega drugog, Python sve više koriste neki delovi samog operativnog sistema, tako da bi podešavanje Python instalacije moglo dovesti do nevolje.

Stoga, čak i ako vaš sistem dolazi sa „Pythonom sistema", razmislite o instalaciji jedne ili više Python implementacija koje ćete slobodno koristiti kao vašu razvojnu pogodnost, bezbedni u znanju da ništa što uradite neće uticati na operativni sistem. Toplo preporučujemo upotrebu *virtuelnih okruženja* (pogledajte „Python okruženja" na strani 230) da izolujete projekte jedni od drugih, omogućavajući im da imaju ono što bi inače moglo biti konfliktne zavisnosti (npr. ako dva vaša projekta zahtevaju različite verzije istog modula treće strane). Alternativno, moguće je lokalno instalirati više Pythona jedan pored drugog.

Pythonova popularnost je dovela do stvaranja mnogih aktivnih zajednica, a ekosistem jezika je veoma aktivan. Sledeći odeljci navode neke od zanimljivijih dešavanja: imajte na umu da naš neuspeh da ovde uključimo projekat odražava ograničenja prostora i vremena, a ne implicira bilo kakvo neodobravanje!

Jython i IronPython

Jython (*https://www.jython.org*), podržava Python na vrhu JVM (*https://oreil.ly/Q8EQB*), i IronPython (*https://ironpython.net*), koji podržava Python na vrhu .NET-a (*https://oreil.ly/o_MTn*), su projekti otvorenog koda da, iako nudi kvalitet na nivou proizvodnje za verzije Pythona koje podržavaju, izgleda da je „zaustavljeno" u vreme pisanja ovog teksta, pošto najnovije verzije koje podržavaju znatno zaostaju za CPythonovim. Svaki „zaustavljen" projekat otvorenog koda mogao bi, potencijalno, ponovo da oživi: sve što je potrebno je jedan ili više entuzijastičnih, posvećenih programera da se posvete njegovom „oživljavanju". Kao alternativu Jythonu za JVM, možete razmotriti GraalVM Python, koji je ranije pomenut.

Numba

Numba (*https://numba.pydata.org*) je kompajler otvorenog koda tačno na vreme (just-in-time, JIT) koji prevodi podskup Pythona i NumPya. S obzirom na njegov snažan fokus na numeričku obradu, ponovo ga pominjemo u Poglavlju 16.

Pyjion

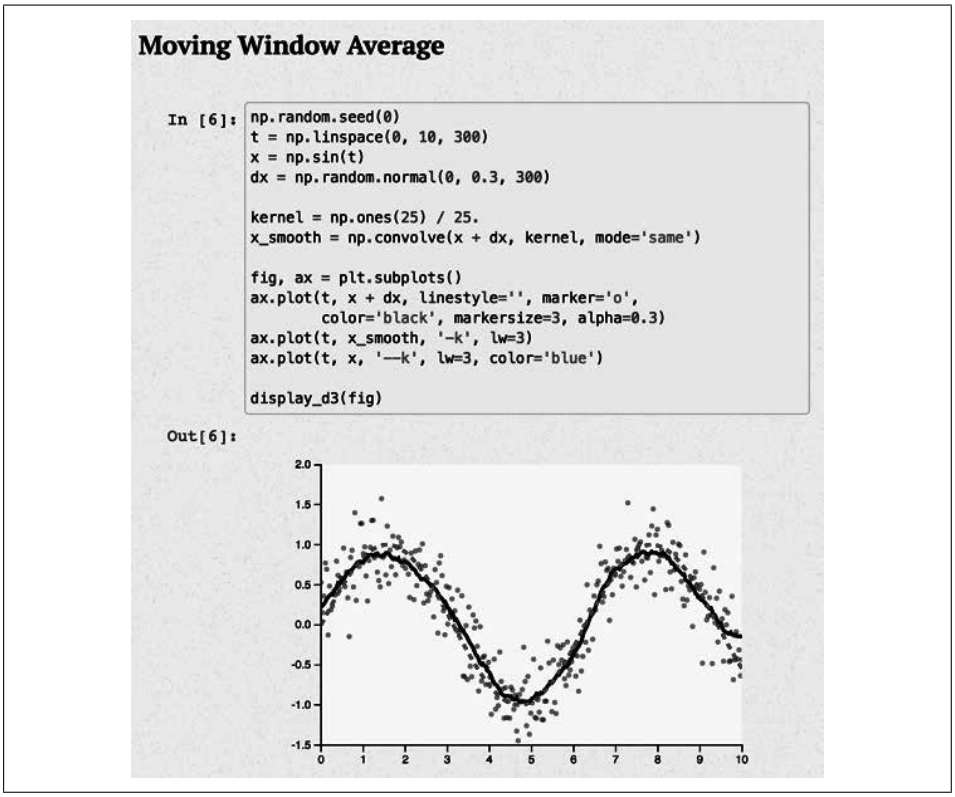
Pyjion (*https://oreil.ly/P7wKC*) je projekat otvorenog koda, koji je prvobitno pokrenuo Microsoft, sa ključnim ciljem dodavanja API-ja u CPython za upravljanje JIT kompajlerima. Sekundarni ciljevi uključuju ponudu JIT kompajlera za Microsoftovo okruženje otvorenog koda CLR (*https://oreil.ly/5zjOG*) (koje je deo .NET-a) i okvira za razvoj JIT kompajlera. Pyjion ne *zamenjuje* CPython; nego pre, to je modul koji uvozite iz CPythona (trenutno zahteva 3.10) koji vam omogućava da prevedete CPythonov bajtkod, „baš na vreme", u mašinski kod za nekoliko različitih okruženja. Integraciju Pyjiona sa CPythonom omogućava PEP 523 (*https://oreil.ly/lFDGw*); međutim, pošto pravljenje Pyjiona zahteva nekoliko alata pored C kompajlera (što je sve što je potrebno za pravljenje CPythona), Python Softvare Foundation (PSF) verovatno nikada neće uključiti Pyjion u CPython izdanja koja distribuira.

IPython

IPython (*https://ipython.org*) poboljšava CPythonov interaktivni interpreter kako bi ga učinio moćnijim i praktičnijim. Omogućava skraćenu sintaksu poziva funkcije i proširivu funkcionalnost poznatu kao *magics* uvedenu znakom procenta (`%`). Takođe obezbeđuje izlaz iz ljuske, omogućavajući Python promenljivoj da primi rezultat komande ljuske (eng. shell). Možete koristiti znak pitanja za ispitivanje dokumentacije objekta (ili dva znaka pitanja za proširenu dokumentaciju); sve standardne funkcije Python interaktivnog interpretera su takođe dostupne.

IPython je napravio posebne korake u naučnom svetu i svetu fokusiranom na podatke i polako se transformisao (kroz razvoj IPython Notebooka, sada prepravljene i preimenovane u Jupyter Notebook (beležnicu), o kojoj se govori u „Jupyter") u interaktivno programsko okruženje koje, među isečcima koda,[3] omogućava vam da umetnete komentare u litralno programiranje (*https://oreil.ly/tx5B3*) (uključujući matematičku notaciju) i prikazuju izlaz izvršavanja koda, opciono sa naprednom grafikom koju proizvode takvi podsistemi kao što su `matplotlib` i `bokeh`. Primer `matplotlib` grafike umetnute u Jupyter beležnicu prikazan je u donjoj polovini Slike 1-1. Jupyter/IPython je jedna od Pajtonovih istaknutih priča o uspehu.

[3] Što može biti na mnogim programskim jezicima, a ne samo na Pythonu.



Slika 1-1. Primer Jupyter beležnice sa ugrađenim matplotlib grafikonom

MicroPython

Kontinuirani trend minijaturizacije doveo je Python u domete hobista. Računari na ploči kao što su Raspberry Pi (*https://www.raspberrypi.org*) i Beagle ploče (*https://beagleboard.org*) omogućavaju vam da pokrenete Python u punom Linux okruženju. Ispod ovog nivoa, postoji klasa uređaja poznatih kao *mikrokontroleri* – programabilni čipovi sa konfigurabilnim hardverom – koji proširuju obim hobi i profesionalnih projekata, na primer olakšavanjem rada sa analognim i digitalnim senzorima, omogućavajući takve aplikacije kao što su merenje svetlosti i temperature sa malo dodatnog hardvera.

I hobisti i profesionalni inženjeri sve više koriste ove uređaje, koji se pojavljuju (a ponekad i nestaju) sve vreme. Zahvaljujući projektu MicroPython (*https://micropython.org*), bogata funkcionalnost mnogih takvih uređaja (*https://oreil.ly/6Ifug*) (micro:bit, Arduino, pyboard, LEGO® MINDSTORMS® EV3, HiFive, itd.) sada se mogu programirati u ograničenim dijalektima Pythona. Važno je napomenuti da je u vreme pisanja knjige uveden Raspberry Pi Pico (*https://oreil.ly/6-s7Q*). S obzirom na uspeh Raspberry Pia u svetu obrazovanja i Picoovu sposobnost da pokrene MicroPython, čini se da Python konsoliduje svoju poziciju programskog jezika sa najširim spektrom aplikacija.

MicroPython je implementacija Pythona 3.4 („sa odabranim funkcijama iz novijih verzija“, da citiram njegove dokumente (*https://oreil.ly/Xe5YP*)), koja proizvodi bajtkod ili izvršni mašinski kod (mnogi korisnici biće srećno nesvesni ove poslednje činjenice). U potpunosti implementira Python 3.4 sintaksu, ali mu nedostaje većina standardne biblioteke. Specijalni hardverski upravljački moduli vam omogućavaju da kontrolišete različite delove ugrađenog hardvera; pristup Pythonovoj biblioteci soketa omogućava uređajima da komuniciraju sa mrežnim uslugama. Spoljni uređaji i događaji tajmera mogu pokrenuti kôd. Zahvaljujući MicroPythonu, jezik Python može u potpunosti da uđe u igru internet stvari.

Uređaj obično nudi pristup interpreteru preko USB serijskog porta ili preko pretraživača koji koristi WebREPL protokol (*https://oreil.ly/sch3F*) (ne znamo ni za jednu `ssh` implementaciju koja u potpunosti funkcioniše, još uvek, međutim, vodite računa o pravilnom zaštitnom zidu ovih uređaja: *ne bi trebalo da oni budu direktno dostupni preko interneta bez odgovarajućih, jakih mera predostrožnosti!*). Možete da programirate sekvencu podizanja uređaja po uključenju, u Pythonu tako što ćete kreirati *boot.py* datoteku u memoriji uređaja, a ova datoteka može da izvrši proizvoljan MicroPython kôd bilo koje složenosti.

Anaconda i Miniconda

Jedna od najuspešnijih Python distribucija[4] poslednjih godina je Anaconda (*https://www.anaconda.com*). Ovaj paket otvorenog koda dolazi sa ogromnim brojem[5] unapred konfigurisanih i testiranih modula proširenja kao dodatak na standardnu biblioteku. U mnogim slučajevima, možda ćete otkriti da sadrži sve neophodne zavisnosti za vaš rad. Ako vaše zavisnosti nisu podržane, takođe možete da instalirate module pomoću `pip`. Na sistemima zasnovanim na Unixu, instalira se veoma jednostavno u jedan direktorijum: da biste ga aktivirali, samo dodajte Anaconda *bin* poddirektorijum na početak `PATH` vaše školjke.

Anaconda je zasnovana na tehnologiji pakovanja koja se zove `conda`. Sestrinska implementacija, Miniconda (*https://oreil.ly/dfX4M*), daje pristup istim ekstenzijama, ali ne dolazi sa njima unapred učitanim; umesto toga preuzima ih po potrebi, što ga čini boljim izborom za kreiranje prilagođenih okruženja. `conda` ne koristi standardna virtuelna okruženja, ali sadrži ekvivalentna sredstva koja omogućavaju razdvajanje zavisnosti za više projekata.

pyenv: Jednostavna podrška za više verzija

Osnovna svrha `pyenv` (*https://oreil.ly/88o8b*) je da olakša pristup onoliko različitih verzija Pythona koliko vam je potrebno. To radi tako što instalira takozvane *shim* skripte za svaku izvršnu datoteku, koje dinamički izračunavaju potrebnu verziju gledajući različite izvore informacija sledećim redosledom:

[4] U stvari, mogućnosti `conda` se proširuju i na druge jezike, a Python je jednostavno još jedna zavisnost.

[5] 250+ koji se automatski instalira uz Anaconda, 7.500+ koji se eksplicitno instalira uz **`conda install`**.

1. Promenljiva okruženja `PYENV_VERSION` (ako je zadata).
2. Datoteku *.pyenv_version* u tekućem direktorijumu (ako postoji) – možete podesiti pomoću komande **`pyenv local`**.
3. Prva datoteka *.pyenv_version* pronađena prilikom penjanja na stablo direktorijuma (ako se pronađe).
4. Datoteku *version* u `pyenv` instalacionom osnovnom direktorijumu – možete zadati pomoću komande **`pyenv global`**.

`pyenv` instalira svoje Python interpretere ispod svog matičnog direktorijuma (obično *~/.pyenv*), i kada bude dostupan, određeni interpreter može da se instalira kao podrazumevani Python u svakom direktorijumu projekta. Alternativno (npr. kada testirate kôd pod više verzija), možete koristiti skripte da biste dinamički menjali interpreter kako skripta napreduje.

Komanda **`pyenv install -list`** prikazuje impresivnu listu od preko 500 podržanih distribucija, uključujući PyPy, Miniconda, MicroPython i nekoliko drugih, plus svaku zvaničnu implementaciju CPythona od 2.1.3 do (u vreme pisanja) 3.11.0rc1.

Transkriptovanje: Konvertujte svoj Python u JavaScript

Učinjeno je mnogo pokušaja da se Python pretvori u jezik baziran za veb čitače, ali JavaScript je uporno drži to mesto. Transcript (*https://www.transcrypt.org*) sistem je Python paket koji se može instalirati `pip`-om za pretvaranje Python koda (trenutno do verzije 3.9) u JavaScript koji se može izvršiti u veb čitaču. Imate pun pristup DOM-u veb čitača, omogućavajući vašem kodu da dinamički manipuliše sadržajem prozora i da koristi JavaScript biblioteke.

Iako kreira minimiziran kôd, Transcript pruža pune izvorne mape (*https://oreil.ly/WjVAa*) koje vam omogućavaju da otklanjate greške u odnosu na Python izvorni kôd, a ne na generisan JavaScript. Možete pisati rukovaoce događajima u veb čitaču u Pythonu, slobodno ih mešajući sa HTML-om i JavaScriptom. Python možda nikada neće zameniti JavaScript kao ugrađen jezik veb čitača, ali Transcript znači da možda više ne morate da brinete o tome.

Još jedan veoma aktivan projekat koji vam omogućava da programirate svoje veb stranice pomoću Pythona (do 3.10) je Brython (*https://brython.info*), a postoje i drugi: Skulpt (*https://skulpt.org*), još uvek nije dorastao Pythonu 3, ali se kreće u tom pravcu; PyPy.js (*https://pypyjs.org*), isto; Pyodide (*https://oreil.ly/jb_US*), trenutno podržava Python 3.10 i mnoga naučna proširenja, a usredsređen je na Wasm (*https://webassembly.org*); i, najnovije, Anacondin PyScript (*https://pyscript.net*), izgrađen na vrhu Pyodidea. Nekoliko od ovih projekata opisujemo detaljnije u „Izvršavanje Pythona u veb čitaču“ na strani 30.

Licenciranje i cene

CPython je pokriven sa Python Softvare Foundation License verzija 2 (*https://oreil.ly/NjjDu*), što je kompatibilno sa GNU javnom licencom (GPL), i omogućava vam da koristite Python za bilo koji vlasnički, besplatni ili drugi razvoj softvera otvorenog koda, slično BSD/Apache/MIT licencama. Licence za PyPy i druge implementacije su slično slobodne. Sve što preuzmete sa glavnih Python i PyPy sajtova neće vas koštati ni dinar. Dalje, ove licence ne ograničavaju uslove licenciranja i cene koje možete da koristite za softver koji razvijate koristeći alate, biblioteke i dokumentaciju koju pokrivaju.

Međutim, nije sve što se odnosi na Python oslobođeno troškova licenciranja ili problema. Mnogi Python izvorni kodovi, alati i moduli proširenja drugih proizvođača koje možete besplatno preuzeti imaju slobodne licence, slične onima za sam Python. Druge su pokrivene GPL ili sa Lesser GPL (LGPL), ograničavajući uslove licenciranja koje možete postaviti na izvedena dela. Neki komercijalno razvijeni moduli i alati mogu zahtevati da platite naknadu, bilo bezuslovno ili ako ih koristite za profit.[6]

Ne postoji zamena za pažljivo ispitivanje uslova i cena licenciranja. Pre nego što uložite vreme i energiju u bilo koji softverski alat ili komponentu, proverite da li možete da podnesete njegovu licencu. Često, posebno u korporativnom okruženju, takva pravna pitanja mogu uključivati konsultacije advokata. Moduli i alati obuhvaćeni u ovoj knjizi, osim ako izričito ne kažemo drugačije, mogu se smatrati, u vreme pisanja ovog teksta, slobodnim za preuzimanje, otvorenim kodom i pokriveni slobodnom licencom sličnom Pythonu. Međutim, izjavljujemo da nemamo pravnu ekspertizu, a licence se mogu menjati tokom vremena, tako da je dvostruka provera uvek potrebna.

Python razvoj i verzije

Python razvija, održava i izdaje tim osnovnih programera predvođenih Guidom van Rossumom, Pythonovim izumiteljem, arhitektom, a sada i „ex“ doživotni dobrovoljni vladar (Benevolent Dictator for Life, BDFL). Ova titula je značila da je Gvido imao poslednju reč o tome šta postaje deo jezika i standardne biblioteke Python. Kada je Gvido odlučio da se povuče kao BDFL, njegovu ulogu u donošenju odluka preuzeo je mali „Upravni savet“, koji su na godišnjem nivou birali članovi PSF-a.

Pythonova intelektualna svojina pripada PSF-u, neprofitnoj korporaciji posvećenoj promociji Pythona, opisanoj u „Python zadužbina“ (Python Softvare Foundation, PSF) na strani 13. Mnogi PSF Sledbenici (Fellows) i članovi imaju privilegije za Python izmene u izvornim spremištima (*https://github.com/python*), kao što je dokumentovano u „Python Developer's Guide“ (*https://oreil.ly/WKjXc*), a većina Python committera su Sledbenici ili članovi PSF-a.

[6] Popularni poslovni model je *freemium*: izdavanje i besplatne verzije i komercijalna „premium“ verzija sa tehničkom podrškom i, možda, dodatnim funkcijama.

Predložene promene u Pythonu su detaljno opisane u javnim dokumentima pod nazivom Python Enhancement Proposals (PEPs) (*https://oreil.ly/HxHfs*). O PEP-ovima raspravljaju programeri Pythona i šira Python zajednica, a na kraju ih Upravni savet (Steering Council) odobrava ili odbija. (Upravni savet može uzeti u obzir debate i preliminarne glasove, ali ih ne obavezuje.) Stotine ljudi doprinose razvoju Pythona putem PEP-a, diskusija, izveštaja o greškama i zakrpa za Python izvorni kôd, biblioteke i dokumente.

Python tim jezgra objavljuje manje verzije Pythona (3.*x* za rastuće vrednosti *x*), takođe poznate kao „izdanja mogućnosti" (feature releases), trenutno brzinom od jednom godišnje (*https://oreil.ly/VYX-k*).

Svako manje izdanje (za razliku od mikroizdanja za ispravku grešaka) dodaje mogućnosti koje Python čine moćnijim, ali takođe vodi računa o održavanju kompatibilnosti unazad. Python 3.0, kome je bilo dozvoljeno da prekine kompatibilnost unazad kako bi se uklonile suvišne „nasleđene" mogućnosti i pojednostavio jezik, prvi put je objavljen u decembru 2008. Python 3.11 (najnovija stabilna verzija u vreme objavljivanja) je prvi put objavljen u oktobru 2022.

Svako manje izdanje 3.*x* prvo je dostupno u alfa izdanjima, označeno kao 3.*x*a0, 3.*x*a1, itd. Posle alfa dolazi najmanje jedno beta izdanje, 3.*x*b1, a posle beta, najmanje jedan kandidat za izdanje, 3.*x*rc1. Do trenutka kada konačno izdanje 3.*x* (3.*x*.0) izađe, on je solidan, pouzdan i testiran na svim glavnim platformama. Svaki Python programer može pomoći da se to obezbedi preuzimanjem alfa, beta verzija i kandidata za izdavanje, isprobavanjem i podnošenjem izveštaja o greškama za sve probleme koji se pojave.

Jednom kada izađe manje izdanje, deo pažnje glavnog tima prelazi na sledeće manje izdanje. Međutim, manje izdanje obično ima uzastopna objavljivanja (tj. 3.*x*.1, 3.*x*.2, itd.), jedno svaka dva meseca, koja ne dodaju funkcionalnost, ali mogu da poprave greške, reše bezbednosne probleme, prenese Python na nove platforme, poboljšavaju dokumentaciju i dodaju alate i (100% kompatibilne unazad!) optimizacije.

Pythonova kompatibilnost unazad je prilično dobra u velikim izdanjima. Možete pronaći kôd i dokumentaciju na mreži (*https://oreil.ly/JbCv3*) za sva stara izdanja Pythona, a Dodatak sadrži sažetu listu promena u svakom od izdanja obuhvaćenih u ovoj knjizi.

Python resursi

Najbogatiji Python resurs je veb: počnite na Python početnoj stranici (*https://www.python.org*), koja je puna veza za istraživanje.

Dokumentacija

ICPython i PyPy dolaze sa dobrom dokumentacijom. Možete čitati CPythonove priručnike na mreži (*https://docs.python.org/3*) (mi ih često nazivamo „dokumentima na mreži"), i razne formate za preuzimanje pogodne za gledanje van mreže, pretraživanje, i štampanje su takođe dostupni. Stranica Python dokumentacije (*https://www.*

python.org/doc) sadrži dodatne pokazivače na veliki broj drugih dokumenata. Postoji i stranica sa dokumentacijom (*http://doc.pypy.org*) za PyPy, a možete pronaći i česta pitanja na mreži za oba Pythona (*https://oreil.ly/-NU8p*) i PyPy (*https://oreil.ly/ajNWC*).

Python dokumentacija za neprogramere

Većina Python dokumentacije (uključujući ovu knjigu) pretpostavlja određeno znanje o razvoju softvera. Međutim, Python je prilično pogodan za ljude koji prvi put programiraju, tako da postoje izuzeci od ovog pravila. Dobri uvodni, besplatni onlajn tekstovi za neprogramere su:

- Uputstvo Josha Cogliatia „Non-Programmers Tutorial for Python 3" (*https://oreil.ly/HnXMA*) (trenutno usredsređeno na Python 3.9)
- „Learning to Program" Alana Gaulda (*https://oreil.ly/FQExV*) (trenutno usredsređen na Python 3.6) Allen Downeyjev *Think Python*, 2. izdanje (*https://oreil.ly/kg6Yd*) (centrirano na neodređenoj verziji Pythona 3.*x*)

Odličan resurs za učenje Pythona (za neprogramere, ali i za manje iskusne programere) je wiki „Beginners' Guide to Python" (*https://oreil.ly/Yf5cK*), koji sadrži obilje linkova i saveta. Održava ga zajednica, tako da će ostati ažuran dok se dostupne knjige, kursevi, alati i ostalo razvijaju i poboljšavaju.

Moduli proširenja i Python izvorni kôd

Dobra početna tačka za istraživanje binarnih datoteka i izvornog koda Python ekstenzija je indeks Python paketa (*https://oreil.ly/PGIim*) (od nas oldtajmera još uvek rado poznat kao „Prodavnica sira", ali se danas generalno naziva PyPI), koji u vreme pisanja ovog teksta nudi više od 400.000 paketa, svaki sa opisima i uputstvima.

Standardna Python izvorna distribucija sadrži odličan Python izvorni kôd u standardnoj biblioteci i u direktorijumu *Tools*, kao i C izvorni kôd za mnoge ugrađene module proširenja. Čak i ako niste zainteresovani za pravljenje Pythona od izvornog koda, predlažemo da preuzmete i raspakujete Python izvornu distribuciju (npr. najnovije stabilno izdanje Pythona 3.11 (*https://oreil.ly/rqYZ9*)) samo u svrhu proučavanja; ili, ako tako odlučite, pregledajte tekuću verziju Python standardne biblioteke na mreži (*https://oreil.ly/zDQ1Z*).

Mnogi Python moduli i alati obrađeni u ovoj knjizi imaju namenske lokacije. Uključujemo reference ka tim sajtovima u odgovarajućim poglavljima.

Knjige

Iako je veb bogat izvor informacija, knjige i dalje imaju svoje mesto (da se ne slažete sa nama u vezi ovoga, mi ne bismo napisali ovu knjigu, a ni vi je ne biste čitali). Knjige o Paythonu su brojne. Evo nekoliko koje preporučujemo (neke pokrivaju starije verzije Python 3, a ne aktuelne):

- Ako znate nešto o programiranju, ali tek počinjete da učite Python, i volite grafičke pristupe instrukciji, *Head First Python*, 2. izdanje, Paul Barry (O'Reilly) može vam dobro poslužiti. Kao i sve knjige iz serije Head First (Um caruje), ona koristi grafiku i humor da vas uči.
- *Dive Into Python 3* (*https://diveintopython3.net*), Marka Pilgrima (Apress), podučava primerom na brz i temeljan način koji je prilično pogodan za one koji su već stručni programeri na drugim jezicima.
- *Beginning Python: From Novice to Professional* (*https://oreil.ly/YtWRs*), Magnusa Lie Hetlanda (Apress), podučava kako kroz detaljna objašnjenja tako i kroz potpun razvoj kompletnih programa u različitim oblastima primene.
- *Fluent Python*, od Luciana Ramalhoa (O'Reilly), je odlična knjiga za iskusnije programere koji žele da koriste više Python idioma i funkcija.

Zajednica

Jedna od najvećih prednosti Pythona je njegova robusna, prijateljska i gostoljubiva zajednica. Python programeri i saradnici se sastaju na konferencijama, „hakatonima" (često poznatim kao *sprintovi* (*https://oreil.ly/oQceG*) u Python zajednici) i u lokalnim grupama korisnika; aktivno razgovaraju o zajedničkim interesima; i pomažu jedni drugima na mejling listama i društvenim mrežama. Za kompletnu listu načina za povezivanje, posetite *https://www.python.org/community*.

Python zadužbina

Osim što poseduje prava intelektualne svojine za programski jezik Python, zadužbina PSF (Python Software Foundation) promoviše Python zajednicu. Zadužbina sponzoriše korisničke grupe, konferencije i sprintove i obezbeđuje grantove za razvoj, širenje i obrazovanje, između ostalih aktivnosti. PSF ima na desetine sledbenika (Fellows) (*https://oreil.ly/maILY*) (nominovanih za doprinos Pythonu, uključujući ceo glavni tim Pythona, kao i tri autora ove knjige); stotine članova koji doprinose vremenom, radom i novcem (uključujući mnoge koji su zaradili nagrade za društvene usluge (*https://oreil.ly/MiQRf*)); i desetine korporativnih sponzora (*https://oreil.ly/FFOZ7*). Svako ko koristi i podržava Python može postati član PSF-a.[7] Pogledajte stranicu za članstvo (*https://oreil.ly/MzdRK*) za informacije o različitim nivoima članstva i o tome kako postati član PSF-a. Ako ste zainteresovani da doprinesete samom Pythonu, pogledajte „Python Developer's Guide" (*https://oreil.ly/1Jwwb*).

Radne grupe

Radne grupe (*https://oreil.ly/0GmfI*) su komiteti koje je osnovao PSF da rade specifične, važne projekte za Python. Evo nekoliko primera aktivnih radnih grupa u vreme pisanja:

[7] Pythononva zadžbina odžava značajnu infrastrukturu za podršku Python ekosistemu. Donacije za PSF su uvek dobrodošle.

- Python Packaging Authority (PyPA) (*https://oreil.ly/0Zxm7*) poboljšava i održava Python ekosistem pakovanja i objavljuje „Python Packaging User Guide“ (*https://packaging.python.org*).
- Radna grupa Python Education (*https://oreil.ly/ZljIc*) promoviše obrazovanje i učenje sa Pythonom.
- Radna grupa za raznovrsnost i inkluziju (*https://oreil.ly/koEo4*) podržava i olakšava rast raznovrsne i međunarodne zajednice Python programera.

Python konferencije

Postoji mnogo Python konferencija širom sveta. Opšte Python konferencije uključuju međunarodne i regionalne, kao što su PyCon (*https://us.pycon.org*) i EuroPython (*https://oreil.ly/nF74d*), i druge lokalnije kao što su PyOhio (*http://www.pyohio.org*) i PyCon Italia (*https://www.pycon.it/en*). Tematske konferencije uključuju SciPy (*https://www.scipy2022.scipy.org*) i PyData (*http://pydata.org/events.html*). Konferencije su često praćene sprintovima kodiranja, gde se Python saradnici okupljaju na nekoliko dana kodiranja fokusiranih na određene projekte otvorenog koda i obilje druženja. Spisak konferencija možete pronaći na stranici Community Conferences and Workshops (*https://oreil.ly/asosj*). Više od 17.000 video snimaka govora o Pythonu, sa više od 450 konferencija, dostupno je na PyVideo sajtu (*https://pyvideo.org*).

Korisničke grupe i organizacije

Python zajednica ima lokalne korisničke grupe na svim kontinentima osim Antarktika[8] – više od 1600 prema listi na LocalUserGroups (*https://oreil.ly/cY6Mk*). Postoje Python susreti (*https://oreil.ly/h6oEs*) širom sveta. PyLadies (*http://www.pyladies.com*) je međunarodna mentorska grupa, sa lokalnim ograncima, za promociju žena u Pythonu; svako ko se interesuje za Python je dobrodošao. NumFOCUS (*https://numfocus.org*), neprofitna dobrotvorna organizacija koja promoviše otvorene prakse u istraživanju, podacima i naučnom računarstvu, sponzoriše PyData konferenciju i druge projekte.

Liste slanja

Stranica Community Mailing Lists (*https://www.python.org/community/lists*) ima veze do nekoliko spiskova slanja u vezi sa Pythonom (i nekih Usenet grupa, za one dovoljno stare koji se sećaju Useneta (*https://oreil.ly/5qYdq*)!). Alternativno, pretražite Mailman (*https://mail.python.org/archives*) da biste pronašli aktivne mejling liste koje pokrivaju širok spektar interesovanja. Zvanična saopštenja u vezi sa Pythonom se postavljaju na listu python-announce (*https://oreil.ly/eg9Ft*). Da biste zatražili pomoć u vezi sa određenim problemima, pišite na *help@python.org*. Za pomoć u učenju ili podučavanju Pythona, pišite na *tutor@python.org*, ili, još bolje, pridružite se listi (*https://oreil.ly/iEQJF*). Za koristan nedeljni pregled vesti i članaka u vezi sa Pythonom, pretplatite

[8] Moramo da se mobilišemo da zainteresujemo više *pingvina* za naš jezik!

se na Python Weekly (*http://www.pythonweekly.com*). Takođe možete pratiti Python Weekly na *@python_discussions@mastodon.social*.

Društveni mediji

Za RSS izvor (*https://oreil.ly/pf4AS*) blogova vezanih za Python, pogledajte Planet Python (*http://planetpython.org*). Ako ste zainteresovani za praćenje razvoja jezika, pogledajte *discuss.python.org* – šalje korisne sažetke ako ga ne posećujete redovno. Na Twiteru pratite @ThePSF. Libera.Chat (*https://libera.chat*) na IRC-u (*https://oreil.ly/AXMAf*) hostuje nekoliko kanala vezanih za Python: glavni je `#python`. LinkedIn (*https://www.linkedin.com*) ima mnogo Python grupa, uključujući Python veb programere (*https://oreil.ly/-LKFZ*). Na Slacku, pridružite se zajednici PySlackers (*https://pyslackers.com*). Na Discordu pogledajte Python Discord (*https://pythondiscord.com*). Tehnička pitanja i odgovori o Python programiranju se mogu naći i pratiti na Stack Overflowu (*http://stackoverflow.com*) pod raznim oznakama, uključujući [python] (*https://oreil.ly/GHoVY*). Python je trenutno najaktivniji (*https://oreil.ly/K3oK3*) programski jezik na Stack Overflowu, i tamo se mogu naći mnogi korisni odgovori sa razjašnjavajućim diskusijama. Ako volite podkaste, pogledajte Python podkaste, kao što su Python Bytes (*https://pythonbytes.fm*).

Instalacija

Možete da instalirate klasične (CPython) i PyPy verzije Pythona na većini platformi. Sa odgovarajućim razvojnim sistemom (C za CPython; PyPy, kodiran u samom Pythonu, potreban je samo prvo instaliran CPython), možete instalirati verzije Pythona iz odgovarajućih distribucija izvornog koda. Na popularnim platformama, imate preporučenu alternativu za instaliranje unapred izgrađenih binarnih distribucija.



Instaliranje Pythona ako je unapred instaliran

Ako vaša platforma dolazi sa unapred instaliranom verzijom Pythona, i dalje vam je najbolje savetovati da instalirate zasebnu ažuriranu verziju za sopstveni razvoj koda. Kada to učinite, *nemojte* da uklanjate ili pišete preko originalne verzije vaše platforme: umesto toga, instalirajte novu verziju pored prve. Na ovaj način nećete ometati nijedan drugi softver koji je deo vaše platforme: takav softver se može oslanjati na specifičnu verziju Pythona koja je isporučena sa samom platformom.

Instaliranje CPythona iz binarne distribucije je brže, štedi vam značajan rad na nekim platformama i jedina je mogućnost ako nemate odgovarajući C kompajler. Instalacija iz izvornog koda vam daje veću kontrolu i fleksibilnost i neophodna je ako ne možete da pronađete odgovarajuću unapred izgrađenu binarnu distribuciju za svoju platformu. Čak i ako instalirate iz binarnih datoteka, najbolje je da preuzmete i izvornu distribuciju, jer ona može sadržati primere, demonstracije i alate koji obično nedostaju u unapred napravljenim binarnim datotekama. Sledeće ćemo pogledati kako da uradimo oboje.

Instaliranje Pythona iz binarnih datoteka

Ako je vaša platforma popularna i aktuelna, lako ćete pronaći unapred napravljene, upakovane binarne verzije Pythona spremne za instalaciju. Binarni paketi se obično sami instaliraju, bilo direktno kao izvršni programi ili putem odgovarajućih sistemskih alata, kao što je Red Hat Package Manager (RPM) na nekim verzijama Linuxa i Microsoft Installer (MSI) na Windowsu. Nakon preuzimanja paketa, instalirajte ga tako što ćete pokrenuti program i izabrati instalacione parametre, kao što je direktorijum u koji će se instalirati Python. U Windowsu izaberite opciju sa oznakom „Add Python 3.10 to PATH“ da bi instalater dodao lokaciju za instalaciju u PATH kako bi lako koristio Python u komandnoj liniji (pogledajte „Program python“ na strani 21).

Možete da preuzmete „zvanične“ binarne datoteke sa stranice za preuzimanje (*https://oreil.ly/b3AP7*) na Python veb lokaciji: pritisnite dugme sa natpisom „Download Python 3.11.x“ da biste preuzeli najnoviji binarni fajl pogodan za platformu vašeg veb čitača.

Mnoge treće strane obezbeđuju besplatne binarne instalacione programe za Python za druge platforme. Postoje instaleri za Linux distribucije, bilo da je vaša distribucija zasnovana na RPM-u (*http://rpmfind.net*) (Red Hat, Fedora, Mandriva, SUSE, itd.) ili zasnovana na Debianu (*http://www.debian.org*) (uključujući Ubuntu, verovatno najpopularniju distribuciju Linuxa u vreme pisanja ovog teksta). Stranica drugih platforma (Other Platforms) (*https://oreil.ly/xvFYV*) pruža veze do binarnih distribucija za sada pomalo egzotične platforme kao što su AIX, OS/2, RISC OS, IBM AS/400, Solaris, HP-UX, i tako dalje (često ne najnovije verzije Pythona, s obzirom na sada „neobičnu“ prirodu takvih platformi), kao i na veoma aktuelnu iOS platformu (*https://oreil.ly/gnJND*), operativni sistem popularnih iPhone (*https://oreil.ly/RelC0*) i iPad (*https://oreil.ly/Sb7_n*) uređaja.

Anaconda (*https://oreil.ly/DxmAG*), pomenuta ranije u ovom poglavlju, je binarna distribucija uključujući Python, plus `conda` (*http://conda.pydata.org/docs*) menadžer paketa, plus stotine ekstenzija nezavisnih proizvođača, posebno za nauku, matematiku, inženjering i analizu podataka. Dostupan je za Linux, Windows i macOS. Miniconda (*https://oreil.ly/RrY5_*), pomenuta ranije u ovom poglavlju, je isti paket, ali bez svih ovih proširenja; možete selektivno instalirati njihove podskupove pomoću `conda`.



macOS

Popularni macOS menadžer paketa otvorenog koda Homebrew (*http://brew.sh*) nudi, između mnogih drugih paketa otvorenog koda, odlične verzije Pythona (*https://oreil.ly/rnK6U*). `conda`, pomenuta u „Anaconda i Miniconda“ na strani 8, takođe dobro funkcioniše na macOS-u.

Instaliranje Pythona iz izvornog koda

Da biste instalirali CPython iz izvornog koda, potrebna vam je platforma sa C kompajlerom kompatibilnim sa ISO-om i alatima kao što je `make`. U Windowsu, normalan način za pravljenje Pythona je pomoću Visual Studia (idealno VS 2022 (*https://oreil.ly/eblTI*), koji je trenutno dostupan programerima besplatno (*https://oreil.ly/2j1dK*)).

Da biste preuzeli Python izvorni kôd, posetite stranicu Python Source Releases (*https://oreil.ly/HeGVY*) (na veb lokaciji Python, pređite kursorom preko Download na traci menija i izaberite „Source code“) i izaberite svoju verziju.

Datoteka ispod veze sa oznakom „Gzipped source tarball“ ima ekstenziju datoteke *.tgz*; ovo je ekvivalentno *.tar.gz* (tj. *tar* arhiva datoteka, komprimovana popularnim `gzip` kompresorom). Alternativno, možete koristiti vezu sa oznakom „XZ compressed source tarball“ da biste dobili verziju sa ekstenzijom *.tar.xz* umesto *.tgz*, komprimovanu sa još moćnijim `xz` kompresorom, ako imate sve potrebne alate za rad sa XZ kompresijom.

Microsoft Windows

Na Windowsu, instaliranje Pythona iz izvornog koda može biti naporan posao osim ako niste upoznati sa Visual Studiom i navikli ste da radite u tekstualnom prozoru poznatom kao *komandna linija*[9] – većina korisnika Windowsa radije jednostavno preuzima unapred izgrađen Python sa sajta Microsoft Store (*https://oreil.ly/wNIMo*).

Ako vam sledeća uputstva zadaju probleme, držite se instaliranja Pythona iz binarnih datoteka, kao što je opisano u prethodnom odeljku. Ionako je najbolje da uradite odvojenu instalaciju iz binarnih datoteka, čak i ako instalirate i iz izvora. Ako primetite nešto čudno dok koristite verziju koju ste instalirali iz izvora, proverite još jednom instalaciju iz binarnih datoteka. Ako neobičnost nestane, to mora da je posledica neke hirovitosti u vašoj instalaciji iz izvora, tako da morate još jednom da proverite detalje o tome kako ste izabrali da izgradite ovo drugo.

U sledećim odeljcima, radi jasnoće, pretpostavljamo da ste napravili novi folder pod nazivom *%USERPROFILE%\py* (npr. *c:\users\tim\py*), što možete da uradite, na primer, upisivanjme komande `mkdir` u bilo kom komandnom prozoru. Preuzmite izvornu *.tgz* datoteku – na primer, *Python3.11.0.tgz*– u taj folder. Naravno, možete imenovati i postaviti folder kako vam najviše odgovara: naš izbor imena je samo u demonstrativne svrhe.

Dekomprimovanje i raspakivanje Python izvornog koda

Možete dekomprimovati i raspakovati datoteku *.tgz* ili *.tar.xz* pomoću, na primer, besplatnog programa 7-Zip (*http://www.7-zip.org*). Preuzmite odgovarajuću verziju sa stranice za preuzimanje (*https://oreil.ly/Fwv5d*), instalirajte je i pokrenite je u datoteci *.tgz* (npr. *c:\users\alek\pi\Python3.11.0.tgz*) koje ste preuzeli sa veb lokacije Pythona.

[9] Ili, u modernim verzijama Windowsa, mnogo poželjniji Windows Terminal (*https://oreil.ly/_Cu97*).

Pod pretpostavkom da ste preuzeli ovu datoteku u folder *%USERPROFILE%\py* (ili je premestili tamo sa *%USERPROFILE%\downloads*, ako je potrebno), sada ćete imati folder pod nazivom *%USERPROFILE%\py\Python3.11.0* ili slično, u zavisnosti od verzije koju ste preuzeli. Ovo je koren stabla koje sadrži celu standardnu Python distribuciju u izvornom obliku.

Izgradnja Python izvornog koda

Otvorite datoteku *readme.txt* koja se nalazi u poddirektorijumu *PCBuild* ovog osnovnog direktorijuma pomoću bilo kog editora teksta i pratite detaljna uputstva koja se tamo nalaze.

Platforme slične Unixu

Na platformama sličnim Unixu, instaliranje Pythona iz izvornog koda je generalno jednostavno.[10] U sledećim odeljcima, radi jasnoće, pretpostavljamo da ste kreirali novi direktorijum pod imnom *~/py* i preuzeli izvornu datoteku *.tgz* – na primer, *Python3.11.0.tgz* – u taj direktorijum. Naravno, možete imenovati i postaviti direktorijum kako vam najviše odgovara: naš izbor imena je samo u svrsi prikaza.

Dekomprimovanje i raspakivanje Python izvornog koda

Možete dekomprimovati i raspakovati datoteku *.tgz* ili *.tar.xz* pomoću popularne GNU verzije `tar`. Samo ukucajte sledeće u odzivnik ljuske:

```
$ cd ~/py && tar xzf Python-3.11.0.tgz
```

Sada imate direktorijum koji se zove *~/py/Python3.11.0* ili slično, u zavisnosti od verzije koju ste preuzeli. Ovo je koren stabla koje sadrži celu standardnu Python distribuciju u izvornom obliku.

Konfigurisanje, pravljenje i testiranje

Naći ćete detaljne beleške u datoteci *README* unutar ovog direktorijuma, pod naslovom „Build instructions“ (Uputstva za izgradnju), i preporučujemo da proučite te beleške. U najjednostavnijem slučaju, međutim, sve što vam treba može biti da date sledeće komande u odzivniku ljuske:

```
$ cd ~/py/Python-3.11/0
$ ./configure
    [configure writes much information, snipped here]
$ make
    [make takes quite a while and emits much information, snipped here]
```

[10] Većina problema sa instalacijom izvornog koda se odnosi na odsustvo različitih biblioteka za podršku, što može dovesti do toga da neke mogućnosti nedostaju u izgrađenom interpreteru. „Python Developers' Guide“ objašnjava kako da rukujete zavisnostima na različitim platformama (*https://oreil.ly/j3XJs*). *build-python-from-source.com* je korisna lokacija koja vam pokazuje sve komande neophodne za preuzimanje, izgradnju i instaliranje određene verzije Pythona, kao i većinu potrebnih podržavajućih biblioteke na nekoliko Linux platformi.

Ako pokrenete **make** bez prethodnog pokretanja **./configure**, **make** implicitno pokreće **./configure**. Kada se **make** završi, proverite da li Python koji ste upravo napravili radi kako se očekuje:

```
$ make test
    [takes quite a while, emits much information, snipped here]
```

Obično, **make test** potvrđuje da vaša verzija radi, ali vas obaveštava da su neki testovi preskočeni jer su nedostajali opcioni moduli.

Neki moduli su specifični za platformu (npr. neki mogu raditi samo na mašinama koje koriste SGI-jev drevni IRIX (*https://oreil.ly/SsGHY*) operativni sistem); ne morate da brinete o njima. Međutim, drugi moduli mogu biti preskočeni jer zavise od drugih paketa otvorenog koda koji trenutno nisu instalirani na vašoj mašini. Na primer, na Unixu, modul `_tkinter` – potreban za pokretanje Tkinter GUI paketa i IDLE integrisanog razvojnog okruženja, koje dolazi sa Pythonom – može da se napravi samo ako **./configure** može pronaći na vašoj mašini instalaciju Tcl/Tk 8.0 ili novije verzije. Pogledajte datoteku *README* za više detalja i posebna upozorenja o različitim Unix i nalik Unix platformama.

Izgradnja iz izvornog koda vam omogućava da podesite konfiguraciju na nekoliko načina. Na primer, možete da napravite Python na poseban način koji vam pomaže da otklonite greške iz memorije kada razvijate Python ekstenzije kodirane u C-u, što je pokriveno u „Izgradnja i instaliranje C-kodiranih Python ekstenzija“ u poglavlju 25 (*https://oreil.ly/python-nutshell-25*). **./configure --help** je dobar izvor informacija o opcijama konfiguracije koje možete da koristite.

Instaliranje nakon izgradnje

Podrazumevano, **./configure** priprema Python za instalaciju u */usr/local/bin* i */usr/local/lib*. Ova podešavanja možete da promenite tako što ćete pokrenuti **./configure** sa opcijom **--prefix** pre izvršavanja **make**. Na primer, ako želite privatnu instalaciju Pythona u poddirektorijum *pi311* vašeg home direktorijuma, pokrenite:

```
$ cd ~/py/Python-3.11.0
$ ./configure --prefix=~/py311
```

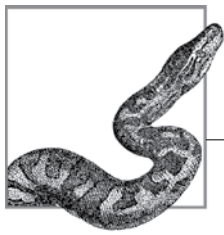
i nastavite sa **make** kao u prethodnom odeljku. Kada završite sa pravljenjem i testiranjem Pythona, da biste izvršili stvarnu instalaciju svih datoteka, pokrenite sledeću komandu:[11]

```
$ make install
```

Korisnik koji izvršava **make install** mora da ima dozvole za pisanje u ciljnim direktorijumima. U zavisnosti od vašeg izbora ciljnih direktorijuma i dozvola za te direktorijume, možda ćete morati da izvršite **su** na *root, bin* ili nekog drugog korisnika kada

[11] Ili **make altinstall**, ako želite da izbegnete pravljenje veza na Python izvršnu stranicu i stranicu dokumentacije.

pokrenete **`make install`**. Uobičajen idiom za ovu svrhu je **`sudo make install`**: ako **`sudo`** zatraži lozinku, unesite lozinku korisnika, a ne *root* lozinku. Alternativni i preporučeni pristup je instaliranje u virtuelno okruženje, kao što je pokriveno u „Python okruženja“ na strani 230.



2
Python interpreter

Da biste razvili softverske sisteme u Pythonu, obično pišete tekstualne datoteke koje sadrže Python izvorni kôd. To možete da uradite koristeći bilo koji editor teksta, uključujući i one koje navodimo u „Python razvojna okruženja“ na strani 27. Zatim obrađujete izvorne datoteke pomoću Python kompajlera i interpretera. To možete uraditi direktno, u okviru integrisanog razvojnog okruženja (IDE) ili preko drugog programa koji sadrži Python. Python interpreter vam omogućava da interaktivno izvršavate Python kôd, kao i IDE.

Program python

Python interpreter program se pokreće kao **`python`** (naziva se *python.exe* na Windowsu). Program uključuje i sam interpreter i Python kompajler, koji se implicitno poziva po potrebi na uvezenim modulima. U zavisnosti od vašeg sistema, program će možda morati da bude u direktorijumu navedenom u vašoj promenljivoj okruženja `PATH`. Alternativno, kao i kod bilo kog drugog programa, možete da navedete njegovu potpunu putanju u komandnoj (shell) liniji ili u skripti ljuske (ili ciljnoj prečici, itd.) koja ga pokreće.[1]

Na Windowsu pritisnite taster Windows i počnite da pišete **`python`**. Pojavljuje se „Python 3.x“ (verzija komandne linije), zajedno sa drugim izborima, kao što je „IDLE“ (Python GUI).

Promenljive okruženja

Pored `PATH`, druge promenljive okruženja utiču na **`python`** program. Neke od njih imaju iste efekte kao opcije prosleđene u **`python`** na komandnoj liniji, kao što ćemo pokazati u sledećem odeljku, ali nekoliko promenljivih okruženja obezbeđuje podeša-

[1] Ovo može uključivati korišćenje navodnika ako ime putanje sadrži razmake – ovo zavisi od vašeg operativnog sistema.

vanja koja nisu dostupna preko opcija komandne linije. Sledeća lista predstavlja neke od često korišćenih; za potpune detalje pogledajte onlajn dokumente (*https://oreil.ly/sYdEK*):

`PYTHONHOME`
: Python instalacioni direktorijum. Poddirektorijum *lib*, koji sadrži standardnu biblioteku Pythona, mora biti u ovom direktorijumu. Na sistemima sličnim Unixu, moduli standardne biblioteke treba da budu u *lib/python-3.x* za Python 3.*x*, gde je *x* manja verzija Pythona. Ako `PYTHONHOME` nije podešen, Python pravi nagađanje o instalacionom direktorijumu.

`PYTHONPATH`
: Lista direktorijuma, odvojenih znakom dve tačke na sistemima sličnim Unixu i znakom tačka-zarez na Windowsu, iz kojih Python može da uveze module. Ova lista proširuje početnu vrednost za Pythonovu `sys.path` promenljivu. Pokrivamo module, uvoz i `sys.path` u Poglavlju 7.

`PYTHONSTARTUP`
: Ime Python izvorne datoteke za izvršenje svaki put kada se pokrene interaktivna sesija interpretera. Takva datoteka se ne pokreće ako ne podesite ovu promenljivu ili je postavite na putanju datoteke koja nije pronađena. `PYTHONSTARTUP` datoteka se ne pokreće kada pokrenete Python skriptu; pokreće se samo kada započnete interaktivnu sesiju.

Kako da podesite i ispitate promenljive okruženja zavisi od vašeg operativnog sistema. Na Unixu koristite komande ljuske, često unutar skripti ljuske za pokretanje. Na Windowsu, pritisnite taster Windows i počnite da kucate **`environment var`** i pojaviće se nekoliko prečica: jedne za promenljive korisničkog okruženja, druge za sistemske. Na Macu možete da radite kao i na drugim sistemima sličnim Unixu, ali imate više opcija, uključujući IDE specifičan za MacPython. Za više informacija o Pythonu na Macu pogledajte „Korišćenje Pythona na Macu“ u dokumentima na mreži (*https://oreil.ly/Co1au*).

Sintaksa i opcije komandne linije

Pythonova sintaksa komandne linije može se sažeti na sledeći način:

```
[path]python {options} [-c command | -m module | file | -] {args}
```

Uglaste zagrade (`[]`) obuhvataju ono što je opciono, vitičaste zagrade (`{}`) obuhvataju stavke od kojih može biti nula ili više prisutnih, a vertikalne crte (`|`) znače izbor između alternativa. Python koristi kosu crtu (`/`) za putanje datoteka, kao u Unixu.

Pokretanje Python skripte na komandnoj liniji može biti jednostavno kao:

```
$ python hello.py
Hello World
```

Možete eksplicitno da navedete putanju do skripte:

```
$ python ./hello/hello.py
 Hello World
```

Ime datoteke skripte može biti apsolutna ili relativna putanja datoteke i ne mora imati nikakvu specifičnu ekstenziju (iako je uobičajeno koristiti ekstenziju *.py*).

opcije su kratki stringovi koji razlikuju velika i mala slova, koji počinju crticom, koji traže od **python**-a nepodrazumevano ponašanje. **python** prihvata samo opcije koje počinju crticom (-). Najčešće korišćene opcije su navedene u Tabeli 2-1. Opis svake opcije daje zahtevano ponašanje ako je promenljiva okruženja (ako postoji) postavljena. Mnoge opcije imaju duže verzije, počevši od dve crtice, kao što pokazuje **python -h**. Potpune detalje pogledajte u dokumentaciji na mreži (*https://oreil.ly/1ZcA9*).

Tabela 2-1. Često korišćene python opcije komandne linije

Opcija	Značenje (i odgovarajuća promenljiva okruženja, ako postoji)
-B	Ne čuvaj datoteke bajtkoda na disk (PYTHONDONTWRITEBYTECODE)
-c	Daje Python naredbe unutar komandne linije
-E	Ignoriše sve promenljive okruženja
-h	Pokazuje punu listu opcija, a zatim završava
-i	Pokreće interaktivnu sesiju nakon pokretanja datoteke ili komande (PYTHONINSPECT)
-m	Zadaje Python modul za pokretanje kao glavnu skriptu
-O	Optimizuje bajtkod (PYTHONOPTIMIZE) – imajte na umu da je ovo veliko slovo O, a ne cifra 0
-OO	Kao **-O**, ali uklanja docstrings iz bajtkoda
-S	Izostavlja implicitni **import** site pri pokretanju (pokriveno u „Prilagođavanje po lokaciji“ na strani 399)
-t, **-tt**	Izdaje upozorenja o nedoslednoj upotrebi tabova (**-tt** izdaje greške, a ne samo upozorenja, za iste probleme)
-u	Koristi binarne datoteke bez baferovanja za standardni izlaz i standardnu grešku (PYTHONUNBUFFERED)
-v	Opširno prati radnje uvoza i čišćenja modula (PYTHONVERBOSE)
-V	Ispisuje broj verzije Pythona i završava
-W arg	Dodaje unos u filter upozorenja (pogledajte „Modul warnings“ na strani 498)
-x	Isključuje (preskače) prvi red izvora skripte

Koristite **-i** kada želite da dobijete interaktivnu sesiju odmah nakon pokretanja neke skripte, sa promenljivim najvišeg nivoa i dalje netaknutim i dostupnim za pregled. Ne treba vam **-i** za normalne interaktivne sesije, iako ne šteti.

-O i **-OO** donose malu uštedu vremena i prostora u bajtkodu generisanom za module koje uvozite, pretvarajući **assert** naredbe u ne-operacije, kao što je pokriveno u „Naredba assert“ na strani 212. **-OO** odbacuje stringove dokumentacije.[2]

[2] Ovo može uticati na kôd koji analizira docstringove u smislene svrhe; predlažemo da izbegavate pisanje takvog koda.

Nakon opcija, ako ih ima, recite Pythonu koju skriptu da pokrene dodavanjem putanje datoteke toj skripti. Umesto putanje datoteke, možete koristiti **`-c`** *`command`* da biste izvršili string komandu Python koda. *`command`* obično sadrži razmake, tako da ćete morati da dodate navodnike oko nje da biste zadovoljili ljusku ili procesor komandne linije vašeg operativnog sistema. Neke ljuske (npr. **`bash`** (*https://oreil.ly/seIne*)) vam omogućavaju da unesete više redova kao jedan argument, tako da `command` može biti niz Python naredbi. Druge ljuske (npr. Windows ljuske) ograničavaju vas na jednu liniju; `command` tada može biti jedna ili više jednostavnih naredbi razdvojenih tačkazarezom (`;`), kao što ćemo raspravljati u „Naredbama“ na strani 39.

Drugi način da zadate koju Python skriptu treba pokrenuti je pomoću **`-m`** *`module`*. Ova opcija govori Pythonu da učita i pokrene modul pod nazivom ***`module`*** (ili član *__main__.py* paketa ili ZIP datoteke pod nazivom *`module`*) iz nekog direktorijuma koji je deo Pythonovog `sys.path`; ovo je korisno sa nekoliko modula iz Python standardne biblioteke. Na primer, kao što je pokriveno u „Modul timeit“ na strani 511, **`-m timeit`** je često najbolji način da se izvrši mikro-benchmarking Python naredbi.

Crtica (**`-`**), ili nedostatak bilo kakvog tokena na ovoj poziciji, govori interpreteru da pročita izvor programa sa standardnog unosa – obično, interaktivna sesija. Crtica vam je potrebna samo ako slede dalji argumenti. *args* su proizvoljni stringovi; Python koji pokrećete može pristupiti ovim stringovima kao stavkama liste `sys.argv`.

Na primer, unesite sledeće u komandnu liniju da bi Python pokazao tekući datum i vreme:

```
$ python -c "import time; print(time.asctime())"
```

Možete da pokrenete komandu samo sa **`python`** (ne morate da navedete punu putanju do Pythona) ako se direktorijum Python izvršnog fajla nalazi u vašoj `PATH` promenljivoj okruženja. (Ako imate instalirano više verzija Pythona, možete zadati verziju sa, na primer, **`python3`** ili **`python3.10`**, po potrebi; tada, verzija koja se koristi ako samo zadate **`python`** je ona koju ste poslednju instalirali.)

Windows py pokretač

Na Windowsu, Python obezbeđuje pokretač **`py`** za instaliranje i pokretanje više verzija Pythona na mašini. Na dnu programa za instalaciju, naći ćete opciju za instaliranje pokretača za sve korisnike (podrazumevano je označeno). Kada imate više verzija, možete da izaberete određenu verziju koristeći **`py`** praćenu opcijom verzije umesto obične komande **`python`**. Uobičajene komandne opcije **`py`** su navedene u Tabeli 2-2 (koristite **`py -h`** da biste videli sve opcije).

Tabela 2-2. Često korišćene `py` opcije komandne linije

Opcija	Značenje
`-2`	Pokreni poslednje instaliranu verziju Python 2.
`-3`	Pokreni poslednje instaliranu verziju Python 3.
`-3.`*`x`* ili **`-3.`*`x-nn`***	Pokreni određenu verziju Pythona 3. Kada se navede samo kao **`-3.10`**, koristi 64-bitnu verziju ili 32-bitnu verziju ako 64-bitna verzija nije dostupna. **`-3.10-32`** ili **`-3.10-64`** bira određenu verziju kada su oba instalirana.
`-0` ili **`--list`**	Lista sve instalirane verzije Pythona, uključujući naznaku da li je verzija 32- ili 64-bitna, kao što je **`3.10-64`**.
`-h`	Lista sve **`py`** opcije komande, praćene standardnom pomoći za Python.

Ako nije data opcija verzije, **`py`** pokreće poslednje instalirani Python.

Na primer, da biste prikazali lokalno vreme koristeći instaliranu Python 3.9 64-bitnu verziju, možete da izvršite ovu komandu:

```
C:\> py -3.9 -c "import time; print(time.asctime())"
```

(Uobičajeno, nema potrebe da date putanju do **`py`**, pošto instaliranje Pythona dodaje **`py`** sistemskom PATH.)

PyPy interpreter

PyPy, napisan u Pythonu, implementira sopstveni kompajler za generisanje LLVM međukoda za izvršavanju na LLVM u pozadini. PyPy projekat nudi neka poboljšanja u odnosu na standardni CPython, posebno u oblastima performansi i višenitnosti. (U trenutku pisanja, PyPy je ažuriran sa Pythonom 3.9.)

`pypy` se može pokrenuti slično kao **`python`**:

```
[path]pypy {options} [-c command | file | - ] {args}
```

Pogledajte PyPy početnu stranicu (*http://pypy.org*) za uputstva za instalaciju i najnovije informacije.

Interaktivne sesije

Kada pokrenete **`python`** bez argumenta skripte, Python pokreće interaktivnu sesiju i traži od vas da unesete Python naredbe ili izraze. Interaktivne sesije su korisne za istraživanje, proveru stvari i korišćenje Pythona kao moćnog, proširivog interaktivnog kalkulatora. (Jupyter beležnica, o kojoj se ukratko govori na kraju ovog poglavlja, je poput „Pythona na steroidima“ posebno za korišćenje interaktivnih sesija.) Ovaj režim se često naziva *REPL*, ili čitanje–procenjivanje–ispis petlja, pošto je to uglavnom ono što interpreter onda radi.

Kada unesete kompletan izraz, Python ga izvršava. Kada unesete kompletan izraz, Python ga procenjuje. Ako izraz ima rezultat, Python ispisuje string koji predstavlja rezultat i dodeljuje rezultat promenljivoj pod nazivom _ (jedna donja crta) tako da možete odmah da koristite taj rezultat u drugom izrazu. String upita je >>> kada

Python očekuje naredbu ili izraz, i `...` kada su naredba ili izraz započeti, ali nisu završeni. Konkretno, Python ima odzivnik `...` kada ste otvorili zagradu, uglastu zagradu ili vitičastu zagradu u prethodnom redu, a niste je zatvorili.

Dok radite u interaktivnom Python okruženju, možete da koristite ugrađenu **`help()`** funkciju da uđete u pomoćni program koji nudi korisne informacije o Python rezervisanim rečima i operatorima, instaliranim modulima i o opštim temama. Kada listate kroz dug opis pomoći, pritisnite **`q`** da biste se vratili na `help>` odzivnik. Da biste izašli iz uslužnog programa i vratili se na Python `>>>` prompt, upišite **`quit`**. Možete dobiti pomoć za određene objekte na Python odzivniku bez ulaska u pomoćni program tako što ćete upisati **`help(`**`obj`**`)`**, gde je `obj` programski objekat za koji želite dodatnu pomoć.

Postoji nekoliko načina na koje možete da završite interaktivnu sesiju. Najčešći su:

- Unesite tastere za kraj datoteke za vaš OS (Ctrl-Z na Windowsu, Ctrl-D na sistemima sličnim Unixu).
- Izvršite bilo koju od ugrađenih funkcija `quit` ili `exit`, koristeći formu `quit()` ili `exit()`. (Izostavljanje `()` na kraju će prikazati poruku poput „Koristite quit() ili Ctrl-D (tj. EOF) za izlaz“, ali će vas i dalje ostaviti u interpreteru.)
- Izvršite naredbu **`raise`**`SystemExit`, ili pozovite `sys.exit()` (mi pokrivamo `SystemExit` i **`raise`** u Poglavlju 6, a `sys` modul u Poglavlju 8).



Koristite Pythonov interktivni interpreter za eksperimentisanje

Isprobavanje Python naredbi u interaktivnom interpreteru je brz način da eksperimentišete sa Pythonom i odmah vidite rezultate. Na primer, evo jednostavne upotrebe ugrađene funkcije `enumerate`:

```
>>> print(list(enumerate("abc")))
[(0, 'a'), (1, 'b'), (2, 'c')]
```

Interaktivni interpreter je dobra uvodna platforma za učenje osnovne Python sintakse i funkcija. (Iskusni Python programeri često otvaraju Python interpreter kako bi brzo proverili retko korišćenu komandu ili funkciju.)

Mogućnosti za editovanje programske linije i istorije delimično zavise od toga kako je Python izgrađen: ako je uključen modul `readline`, dostupne su sve funkcije GNU `readline` biblioteke. Windows ima jednostavnu, ali upotrebljivu istoriju za programe interaktivnog režima teksta kao što je **`python`**.

Pored ugrađenog Python interaktivnog okruženja i onih koji se nude kao deo bogatijih razvojnih okruženja opisanih u sledećem odeljku, možete slobodno da preuzmete druga moćna interaktivna okruženja. Najpopularniji je *IPython* (*http://ipython.org*), pokriven na strani 6, koji nudi zadivljujuće bogatstvo funkcija. Jednostavniji, lakši, ali i dalje prilično zgodan alternativni interpreter za čitanje je *bpython* (*https://oreil.ly/UBZVL*).

Python razvojna okruženja

Interaktivni režim Python interpretera je najjednostavnije razvojno okruženje za Python. Primitivan je, ali je lagan, ima mali otisak i brzo se pokreće. Zajedno sa dobrim editorom teksta (kao što je objašnjeno u „Besplatni editori teksta sa podrškom za Python“ na strani 28) i mogućnostima za editovanje linija i istorije, interaktivni interpreter (ili, alternativno, mnogo moćniji IPython/Jupyter interpreter komandne linije) je upotrebljivo razvojno okruženje. Postoji i nekoliko drugih razvojnih okruženja koje možete koristiti.

IDLE

Pythonovo Integrisano okruženje za razvoj i učenje (Integrated Development and Learning Environment, IDLE) (*https://oreil.ly/1vXr6*) dolazi sa standardnim Python distribucijama na većini platformi. IDLE je višeplatformska, 100% čista Python aplikacija zasnovana na Tkinter GUI. Nudi Python školjku sličnu interaktivnom Python interpreteru, ali bogatiju. Takođe uključuje editor teksta optimizovan za editovanje Python izvornog koda, integrisani interaktivni program za otklanjanje grešaka i nekoliko specijalizovanih veb čitača i prikazivača.

Za više funkcionalnosti u IDLE-u, instalirajte IdleX (*https://oreil.ly/cU_aD*), značajnu kolekciju besplatnih ekstenzija trećih strana.

Da biste instalirali i koristili IDLE na macOS-u, pratite posebna uputstva (*https://oreil.ly/wHA6I*) na veb lokaciji Python.

Drugi Python IDE

IDLE je razvijen, stabilan, lak, prilično bogat i proširiv. Postoji i mnogo drugih IDE-a: namenjenih za određenu platformu ili za više platformi, besplatni ili komercijalni (uključujući komercijalne IDE sa besplatnim ponudama, posebno ako razvijate softver otvorenog koda), samostalni ili dodaci drugim IDE-ovima.

Neki od ovih IDE-a imaju funkcije kao što su statička analiza, GUI graditelji, programi za otklanjanje grešaka i tako dalje. Pythonova IDE wiki stranica (*https://oreil.ly/EMpSD*) navodi preko 30 i ukazuje na mnoge druge URL adrese sa recenzijama i poređenjima. Ako ste IDE kolekcionar, srećan vam lov!

Ne možemo navesti čak ni mali podskup svih dostupnih IDE-a. Besplatni dodatak PyDev (*http://www.pydev.org*) za popularan modularan IDE Eclipse na više platformi na više jezika (*http://www.eclipse.org*) ima odličnu podršku za Python. Steve je dugogodišnji korisnik Winga (*https://wingware.com*) kompanije Archaeopteryx, najcenjenijeg IDE-a specifičnog za Python. Paulov IDE je dobar izbor, a možda jedini najpopularniji Python IDE treće strane danas je PyCharm (*https://oreil.ly/uQWxm*) od JetBrainsa. Thonny (*https://thonny.org*) je popularan početnički IDE, lagan, ali potpuno funkcionalan i lak za instalaciju na Raspberry Pi (ili na bilo koju drugu popularnu platformu). I ne treba zanemariti Microsoftov Visual Studio Code (*https://code.visualstudio.com*), odličan, veoma popularan IDE za više platformi sa podrškom (pre-

ko dodataka) za brojne jezike, uključujući Python. Ako koristite Visual Studio, pogledajte PTVS (*https://oreil.ly/VZ7Dl*), dodatak otvorenog koda koji je posebno dobar u omogućavanju otklanjanja grešaka na mešovitoj upotrebi jezika Python i C.

Besplatni editori teksta sa podrškom za Python

Možete da uređujete Python izvorni kôd sa bilo kojim editorom teksta, čak i jednostavnim kao što je Notepad na Windowsu ili *ed* na Linuxu. Mnogi moćni besplatni editori podržavaju Python sa dodatnim funkcijama kao što su kolorizacija zasnovana na sintaksi i automatsko uvlačenje. Uređivači na više platformi vam omogućavaju da radite na uniformne načine na različitim platformama. Dobri editori teksta vam omogućavaju da iz editora pokrećete alate po vašem izboru na izvornom kodu koji uređujete. Ažurna lista editora za Python može se naći na wikiju PythonEditors (*https://oreil.ly/HGAzB*), na kojoj su navedene desetine njih.

Najbolji za čistu moć uređivanja može biti klasičan Emacs (*https://oreil.ly/MnEBy*) (pogledajte Python wiki stranicu (*https://oreil.ly/AIocZ*) za dodatke specifične za Python). Emacs nije lako naučiti, niti je lagan.[3] Aleksov lični favorit[4] je još jedan klasik: Vim (*http:/www.vim.org*), poboljšana verzija Bram Moolenaarovog tradicionalnog Unix editora *vi*. Verovatno nije *prilično* moćan kao Emacs, ali ipak vredi razmotriti – brz je, lagan, programabilan za Pythona i radi svuda u tekstualnom režimu i GUI verzijama. Za odličnu pokrivenost Vimom, pogledajte *Learning the vi and Vim Editors*, 8. izdanje, Arnolda Robbinsa i Elberta Hannah (O'Reilly); pogledajte Pyhton wiki stranicu (*https://oreil.ly/6pQ6t*) za savete i dodatke specifične za Python. Steve i Ana koriste Vim, a tamo gde je dostupan, Steve koristi komercijalni editor Sublime Text (*https://www.sublimetext.com*), sa dobrim bojama sintakse i dovoljno integracije za pokretanje vaših programa iz editora. Za brzo uređivanje i izvršavanje kratkih Python skripta (i kao brz i lagan opšti editor teksta, čak i za tekstualne datoteke od više megabajta), SciTE (*https://scintilla.org/SciTE.html*) je Paulov glavni editor.

Alatke za proveru Python programa

Python kompajler proverava sintaksu programa dovoljno da može da pokrene program ili da prijavi sintaksičku grešku. Ako želite detaljnije provere svog Python koda, možete preuzeti i instalirati jedan ili više alata nezavisnih proizvođača za tu svrhu. `pyflakes` (*https://oreil.ly/RPeeJ*) je veoma brz, lagan proveravač: nije temeljan, ali ne uvozi module koje proverava, što ga čini brzim i bezbednim. Na drugom kraju spektra, `pylint` (*https://www.pylint.org*) je veoma moćan i veoma podesiv; nije lagan, ali uzvraća time što može da proveri mnoge detalje stila na veoma prilagodljive načine zasnovane na konfiguracionim datotekama koje se mogu uređivati.[5] `flake8` (*https://*

[3] Odlično mesto za početak je *Learning GNU Emacs*, 3. izdanje (O'Reilly).

[4] Ne samo kao „editor", već i kao Alekov omiljen alat „što bliže IDE-u koliko god Alex ide"!

[5] `pylint` sadrži koristan `pyreverse` (*https://oreil.ly/vSs_v*) uslužan program za automatsko generisanje *UML* dijagrama klasa i paketa direktno iz vašeg Python koda.

pypi.org/project/flake8) povezuje `pyflakes` sa drugim formaterima i prilagođenim dodacima i može da rukuje velikim bazama koda tako što širi posao na više procesa. `black` (*https://pypi.org/project/black*) i njena varijanta `blue` (*https://pypi.org/project/blue*) namerno se manje mogu konfigurisati; ovo ih čini popularnim među široko rasprostranjenim projektnim timovima i projektima otvorenog koda kako bi se primenio zajednički Python stil. Da biste bili sigurni da ne zaboravite da ih pokrenete, možete da ugradite jedan ili više ovih čekera/formatera u svoj tok rada koristeći paket `pre-commit` (*https://pypi.org/project/pre-commit*).

Za detaljniju proveru Python koda za ispravnu upotrebu tipa, koristite alatke kao što su `mypy` (*http://mypy-lang.org*); pogledajte Poglavlje 5 za više o ovoj temi.

Izvršavanje Python programa

Koje god alate koristite za izradu vaše Python aplikacije, možete da vidite svoju aplikaciju kao skup Python izvornih datoteka, koje su normalne tekstualne datoteke koje obično imaju ekstenziju (oznaku tipa) *.py*. *script* je datoteka koju možete direktno pokrenuti. *modul* je datoteka koju možete da uvezete (kao što je pokriveno u Poglavlju 7) da biste obezbedili neke funkcionalnosti drugim datotekama ili interaktivnim sesijama. Python datoteka može biti *i* modul (omogućava funkcionalnost kada se uvozi) *i* skripta (OK za direktno pokretanje). Korisna i široko rasprostranjena konvencija je da Python datoteke koje su prvenstveno namenjene za uvoz kao moduli, kada se pokreću direktno, treba da izvrše neke operacije samotestiranja, kao što je pokriveno u „Testiranje“ na strani 476.

Python interpreter automatski kompajlira Python izvorne datoteke po potrebi. Python čuva kompajliran bajtkod u poddirektorijumu koji se zove *__pycache__* unutar direktorijuma sa izvorom modula, sa ekstenzijom specifičnom za verziju označenom da označi nivo optimizacije.

Da biste izbegli čuvanje prevedenog bajtkoda na disk, možete pokrenuti Python sa opcijom **`-B`**, što može biti praktično kada uvozite module sa diska koji je samo za čitanje. Python ne čuva prevedenu formu bajtkoda skripta kada direktno pokrenete skriptu; umesto toga, Python ponovo kompajlira skriptu svaki put kada je pokrenete. Python čuva datoteke bajtkoda samo za module koje uvozite. Automatski ponovo izgrađuje bajtkod fajl svakog modula kad god je to potrebno – na primer, kada editujete izvor modula. Na kraju, za primenu, možete da pakujete Python module koristeći alate pokrivene u Poglavlju 24 (dostupno na mreži (*https://oreil.ly/python-nutshell-24*)).

Možete da pokrenete Python kôd pomoću Python interpretera ili IDE.[6] Obično počinjete izvršavanje pokretanjem skripte najvišeg nivoa. Da biste pokrenuli skriptu, dajte njenu putanju kao argument **`python`**, kao što je pokriveno u „Program python“ na stani 21. U zavisnosti od vašeg operativnog sistema, možete pozvati **`python`** direktno iz shell skripte ili komandne datoteke. Na sistemima sličnim Unixu, Python skriptu

[6] Ili na mreži: Paul, na primer, održava listu (*https://oreil.ly/GVT93*) onlajn Python interpretera.

možete učiniti direktno izvršnom tako što ćete postaviti bitove dozvole datoteke `x` i `r` i započeti skriptu sa *shebang* redom, red kao što je:

```
#!/usr/bin/env python
```

ili neki drugi red koji počinje sa `#!` praćen putanjom do `python` interpreter programa, u kom slučaju opciono možete dodati jednu reč opcija – na primer:

```
#!/usr/bin/python -OB
```

Na Windowsu možete koristiti isti stil `#!` reda, u skladu sa PEP 397 (*https://oreil.ly/lmMal*), da biste naveli određenu verziju Pythona, tako da vaše skripte mogu biti razmenljive između Unixovih i Windows sistema. Možete pokrenuti Python skripte sa uobičajenim Windows mehanizmima, kao što je dvostruki pritisak na njihove ikone. Kada pokrenete Python skriptu dvostrukim pritiskom na ikonu skripte, Windows automatski zatvara konzolu tekstualnog režima koja je povezana sa skriptom čim se skripta završi. Ako želite da se konzola zadrži (da omogući korisniku da pročita izlaz skripte na ekranu), obezbedite da se skripta ne prekine prerano. Na primer, koristite, kao poslednju naredbu skripte:

```
input('Prtisni Enter za završetak')
```

Ovo nije neophodno kada pokrenete skriptu iz komandne linije.

Na Windowsu možete da koristite ekstenziju *.pyw* i program interpretera *pythonw.exe* umesto *.py* i *python.exe*. *w* varijante izvršava Python bez konzole u tekstualnom režimu, a samim tim i bez standardnog unosa i izlaza. Ovo je dobro za skripte koje se oslanjaju na GUI ili se pokreću nevidljivo u pozadini. Koristite ih samo kada je program potpuno očišćen od grešaka, da bi standardni izlaz i greška bili dostupni za informacije, upozorenja i poruke o greškama tokom razvoja.

Aplikacije kodirane na drugim jezicima mogu da ugrade Python, kontrolišući izvršavanje Pythona za svoje potrebe. Ovo ukratko ispitujemo u „Ugrađivanje Pythona“ u Poglavlju 25 (dostupno na mreži (*https://oreil.ly/python-nutshell-25*)).

Izvršavanje Pythona u veb čitaču

Postoje opcije za pokretanje Python koda u okviru sesije čitača, koje se izvršava ili u procesu čitača ili u nekoj zasebnoj komponenti zasnovanoj na serveru. PyScript je primer prvog pristupa, a Jupyter drugog.

PyScript

Nedavni razvoj u nastojanju Python-u-čitaču je objavljivanje PyScripta (*https://pyscript.net*) od strane Anaconde. PyScript je izgrađen na vrhu Pyodidea,[7] koji koristi WebAssembly za podizanje potpunog Python motora u veb čitaču. PyScript uvodi

[7] Odličan primer sinergije otvorenog koda dobijaju projekti koji „stoje na ramenima divova“ kao obična, svakodnevna stvar!

prilagođene HTML oznake tako da možete pisati Python kôd bez potrebe da znate ili koristite JavaScript. Koristeći ove oznake, možete kreirati statičnu HTML datoteku koja sadrži Python kôd koji će se izvršavati u udaljenom veb čitaču, bez potrebe za dodatnim instaliranim softverom.

Jednostavan PyScript „Zdravo svima!“ HTML datoteka može izgledati ovako:


```
<html>
<head>
    <link rel='stylesheet'
 href='https://pyscript.net/releases/2022.06.1/pyscript.css' />
    <script defer
 src='https://pyscript.net/releases/2022.06.1/pyscript.js'></script>
</head>
<body>
<py-script>
import time
print('Zdravo svima!')
print(f'The current local time is {time.asctime()}')
print(f'The current UTC time is {time.asctime(time.gmtime())}')
</py-script>
</body>
</html>
```

Možete da sačuvate ovaj kôd kao statičnu HTML datoteku i uspešno ga pokrenete u klijentskom čitaču, čak i ako Python nije instaliran na vašem računaru.



Stižu promene u PyScript

PyScript je još uvek u ranom razvoju u vreme objavljivanja, tako da će se specifične oznake i API-ji prikazani ovde verovatno promeniti kako se paket dalje razvija.

Za potpunije i ažurnije informacije pogledajte veb lokaciju PyScript (*https://pyscript.net*).

Jupyter

Proširenja za interaktivni interpreter u IPythonu (pokrivena u „IPython“, na strani 6) dodatno su proširena Jupyter projektom (*https://jupyter.org*), najpoznatijim po Jupyter Notebooku (Jupyter beležnice), koji Python programerima nudi alatku za „pismeno programiranje“ (*https://oreil.ly/yvn4z*). Server za beležnice, kome se obično pristupa preko veb lokacije, čuva i učitava svaku beležnicu, kreirajući proces Python kernela za interaktivno izvršavanje njegovih Python komandi.

Beležnice su bogato okruženje. Svaka od njih je niz ćelija čiji sadržaj može biti ili kôd ili obogaćen tekst formatiran jezikom Markdown proširenim sa LaTeX-om, omogućavajući uključivanje složene matematike. Ćelije koda mogu da proizvedu bogate rezultate, uključujući najpopularnije formate slika, kao i skriptovani HTML. Posebne

integracije prilagođavaju biblioteku `matplotlib` vebu, a postoji i sve veći broj mehanizama za interakciju sa kodom beležnice.

Dalje integracije omogućavaju da se beležnice pojavljuju na druge načine. Na primer, sa pravim proširenjem, možete lako formatirati Jupyter beležnicu kao reveal.js (*https://revealjs.com*) projekciju slajdova za prezentacije u kojima se ćelije koda mogu interaktivno izvršavati. Jupyter Book (*https://jupyterbook.org*) vam omogućava da skupite beležnice zajedno kao poglavlja i objavite kolekciju kao knjigu. GitHub omogućava pregledavanje (ali ne i izvršavanje) beležnica (poseban renderer obezbeđuje ispravno formatiranje beležnice).

Postoji mnogo primera Jupyter beležnica dostupnih na internetu. Za dobru demonstraciju njegovih karakteristika, pogledajte veb lokaciju Executable Books (*https://oreil.ly/Y2WS0*); beležnice podupiru njegov format za objavljivanje.