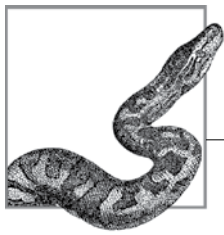

2
Python interpreter

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.