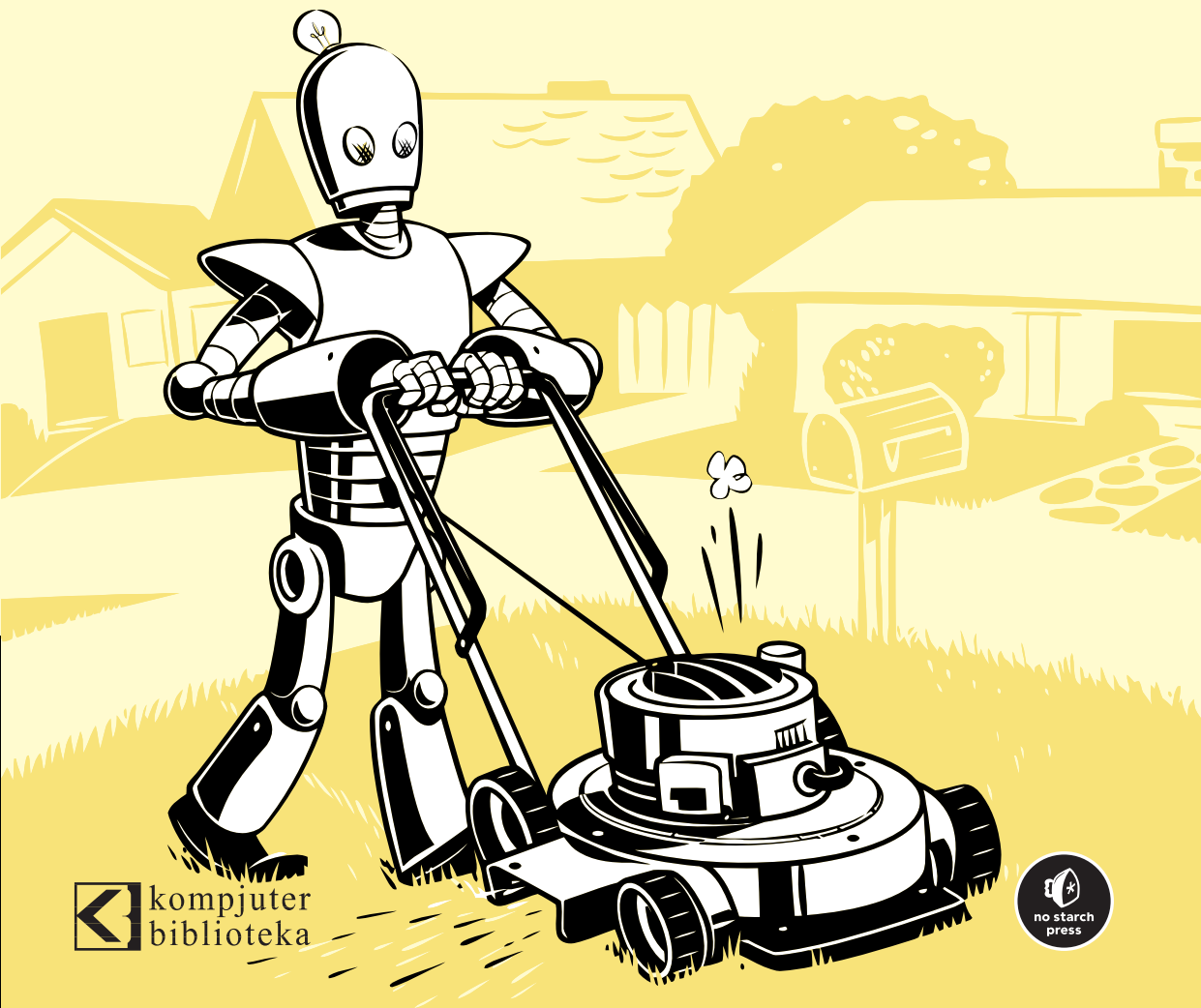
UVOD U PYTHON

Automatizovanje dosadnih poslova

PRAKTIČNO PROGRAMIRANJE ZA POČETNIKE

AL SWEIGART



kompjuter biblioteka

no starch press

UVOD U PYTHON

Automatizovanje dosadnih poslova

PRAKTIČNO PROGRAMIRANJE ZA POČETNIKE

Al Sweigart

kompjuter biblioteka

no starch press

Izdavač:
kompjuter biblioteka

Obalskih radnika 15, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autor: Al Sweigart

Prevod: Slavica Prudkov

Lektura: Miloš Jevtović

Slog : Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Pekograf“, Zemun

Tiraž: 500

Godina izdanja: 2016.

Broj knjige: 482

Izdanje: Prvo

ISBN: 978-86-7310-505-5

Automate the Boring Stuff with Python

by Al Sweigart,

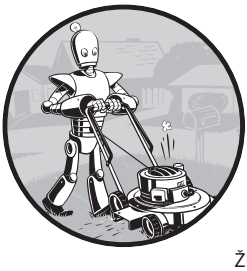
ISBN: 978-1-59327-599-0
2015

CIP - Каталогизација у публикацији
Народна библиотека Србије, Београд

Kratak sadržaj

Sadržaj

UVOD



Moj cimer je radio u prodavnici elektronike. Povremeno, prodavnica je primala tabele sa cenama hiljada proizvoda po kojima su konkurenti prodavali proizvode. Tim od tri radnika bi štampao tabelu i dobili bi veliku gomilu papira. Za svaku cenu proizvoda u tabeli oni bi potražili cenu po kojoj njihova prodavnica prodaje proizvod i beležili sve proizvode koje su konkurenti prodavali jeftinije. Ovaj posao je, obično, trajao nekoliko dana.

„Mogao bih da napišem program koji će to uraditi ako imate originalni fajl za štampu“, rekao im je moj cimer kada ih je video da sede na podu sa razbacanim i nagomilanim papirima oko sebe.

Nakon nekoliko sati, on je imao kratak program koji je čitao cene konkurenata iz fajla, pronalazio proizvod u bazi podataka prodavnice i označavao da li je cena konkurenta niža. Ovaj program je zahtevao svega nekoliko sekundi za pokretanje.

To je moć računarskog programiranja. Računar je kao švajcarski vojni nož koji možete da konfigurišete za bezbroj zadataka. Mnogi ljudi provode sate klikćući i kucujući za

izvršavanje ponavljajućih zadataka i nisu svesni da mašina koju koriste može da obavi taj posao za nekoliko sekundi ako joj se daju odgovarajuće instrukcije.

Za koga je ova knjiga?

Softver je, u osnovi, skup raznih alata koje danas koristimo. Skoro svako koristi društvene mreže za komunikaciju, mnogi ljudi imaju računare povezane na Internet u svojim telefonima i većina kancelarijskih poslova uključuje interakciju sa računarom da bi posao bio izvršen. Kao rezultat svega toga, potražnja za ljudima koji znaju da programiraju je u ogromnom porastu. Reklame za nebrojene knjige, interaktivne web tutorijale i kampove za programere obećavaju da će ambiciozne početnike pretvoriti u softverske inženjere sa šestocifrenim zaradama.

Ova knjiga nije za ljude koji u to veruju. Ona je za sve ostale.

Ova knjiga vas neće pretvoriti u profesionalnog programera više nego što će nekoliko časova sviranja na gitari da vas pretvori u rok zvezdu. Međutim, ako obavljate kancelarijske poslove, poslove administratora, predavača ili neki drugi posao za koji je potreban računar za rad ili zabavu, naučićete osnove programiranja da biste mogli da automatizujete jednostavne zadatke, kao što su sledeći:

- pomeranje i promena naziva hiljada fajlova i sortiranje tih fajlova u direktorijume
- popunjavanje online formulara, bez kucanja
- preuzimanje fajlova ili kopija teksta sa web sajta kad god je ažuriran
- podešavanje računara da šalje određena obaveštenja
- ažuriranje i formatiranje Excel tabela
- provera e-maila i slanje unapred napisanih odgovora

Ovi zadaci su jednostavni, ali korisnicima oduzimaju vreme i često su veoma trivijalni ili specifični da ne postoji već napisan softver za njihovo izvršenje. „Naoružani“ sa malo programerskog znanja, možete da obezbedite da računar izvrši ove zadatke umesto vas.

Konvencije

Ova knjiga nije dizajnirana kao referentni priručnik; ovo je vodič za početnike. Stil programiranja je ponekad suprotan najboljoj praksi (na primer, neki programi koriste globalne varijable), ali to je kompromis da bi kod bio jednostavniji za učenje. Ova knjiga je napisana za ljude koji žele da napišu jednostavan kod, pa se ne troši vreme na stil i eleganciju programa. Sofisticirani koncepti programiranja, kao što su objektno-orijentisano programiranje, sagledavanje liste i generatori, nisu opisani u ovoj knjizi zbog složenosti koju oni dodaju programu. Iskusni programeri mogu da ukažu na načine na koji kod u ovoj knjizi

može da se promeni i poboljša efikasnost, ali ova knjiga isključivo je fokusirana na pokretanje programa sa najmanje moguće uloženog truda.

Šta je programiranje?

U TV emisijama i filmovima često se prikazuju programeri koji furiozno kucaju zagonetne nizove jedinica i nula na blještavim ekranima, ali moderno programiranje nije toliko misteriozno. *Programiranje* je jednostavan čin unošenja instrukcija koje računar treba da izvrši. Ove instrukcije mogu da uključuju neke brojeve, modifikovanje teksta ili potragu za informacijama u fajlovima ili da komuniciraju sa drugim računarima posredstvom Interneta.

Svi programi koriste osnovne instrukcije kao gradivne blokove. Evo nekoliko najčešće upotrebljavanih (na engleskom jeziku):

Do this; then do that. (Uradi ovo; zatim uradi ono.)

If this condition is true, perform this action; otherwise, do that action. (Ako je ovaj uslov tačan, izvrši ovu akciju; u suprotnom, izvrši onu akciju.)

Do this action that number of times. (Izvrši ovu akciju toliko puta.)

Keep doing that until this condition is true. (Nastavi izvršenje dok je ovaj uslov tačan.)

Takođe možete da kombinujete ove gradivne blokove da biste implementirali složenije odluke. Na primer, ovo su programerske instrukcije, pod nazivom izvorni kod, za jednostavan program koji je napisan Python programskim jezikom. Počevši od vrha, Python softver pokreće svaku liniju koda (neke linije se pokreću samo ako je određeni uslov tačan; u suprotnom, Python pokreće neku drugu liniju), dok ne dođe do kraja.

```
❶ passwordFile = open('SecretPasswordFile.txt')
❷ secretPassword = passwordFile.read()
❸ print('Enter your password.')
  typedPassword = input()
❹ if typedPassword == secretPassword:
❺    print('Access granted')
❻    if typedPassword == '12345':
❼       print('That password is one that an idiot puts on their luggage.')
  else:
❽    print('Access denied')
```

Možda ne znate ništa o programiranju, ali verovatno, ako pročitate prethodni kod, možete razumno da pretpostavite šta izvršava. Prvo, otvoren je fajl *SecretPasswordFile.txt* ❶ i pročitana je tajna lozinka u njemu ❷. Zatim se od korisnika traži da unese lozinku (pomoću tastature) ❸. Ove dve lozinke se upoređuju ❹, a ako su iste, program prikazuje na ekranu *Access granted* ❺. Zatim, program proverava da li je lozinka *12345* ❻ i prikazaće da ovaj izbor možda i nije najbolji za lozinku ❼. Ako lozinke nisu iste, program prikazuje na ekranu *Access denied* ❽.

Šta je Python?

Python se odnosi na Python programski jezik (sa pravilima za pisanje sintakse koja se smatraju validnim Python kodom) i Python softver za prevođenje koji čita izvorni kod (napisan u Python programskom jeziku) i izvršava njegove instrukcije. Python softver za prevođenje može se preuzeti besplatno na adresi http://python.org/; postoje verzije za Linux, OS X i Windows.

Naziv Python potiče od naziva britanske komedijaške grupe Monty Python, a ne od zmije. Python programeri se od milja zovu Pythonistas, a reference Monty Pythona i zmije obično su šaljivi deo Python uputstava i dokumentacije.

Programerima nije potrebno veliko poznavanje matematike

Mnogi ljudi misle da je za učenje programiranja neophodno veliko poznavanje matematike. Međutim, programiranje uglavnom ne zahteva matematiku, osim osnovne aritmetike. U stvari, biti dobar u programiranju je slično kao i biti dobar u rešavanju sudoku slagalica.

Da biste rešili sudoku slagalicu, brojevi od 1 do 9 treba da se popune u svakom redu, svakoj koloni i svakom unutrašnjem kvadratu veličine 3x3, na tabli koja je veličine 9x9. Rešenje pronalazite primenom oduzimanja i logike, koristeći početne brojeve. Na primer, pošto se broj 5 prikazuje u gornjem levom uglu sudoku slagalice koja je prikazana na slici 0-1, ne može da se prikaže nigde u gornjem redu, u koloni sa leve strane table ili u gornjem levom 3x3 kvadratu. Rešavanje jednog reda, kolone ili kvadrata će pružiti više nagoveštaja za ostatak slagalice.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Slika 0-1 Nova sudoku slagalica (levo) i njeno rešenje (desno) - bez obzira na upotrebu brojeva, sudoku ne uključuje matematiku (Images © Wikimedia Commons).

Samo zato što sudoku uključuje brojeve ne znači da treba da budete dobar matematičar da biste otkrili rešenje. Isto važi i za programiranje. Kao i rešavanje sudoku slagalice, pisanje programa uključuje razbijanje problema na pojedinačne, detaljne korake. Slično tome, kada *ispravljate grešku* u programima (odnosno, pronalazite i ispravljate greške), pažljivo ćete posmatrati šta program radi i pronaći uzrok greške. Kao i za sve druge veštine, što više programirate, postaćete sve bolji.

Programiranje je kreativna aktivnost

Programiranje je kreativan zadatak, sličan konstruisanju zamka od LEGO kockica. Započinjete sa osnovnom idejom kako želite da zamak izgleda i pripremite dostupne blokove. Zatim, počinjete gradnju. Kada završite gradnju programa, možete da ulepšate kod isto kao što biste ulepšali i zamak.

Razlika između programiranja i drugih kreativnih aktivnosti je što, kada programirate, u računaru imate sav „sirovi" materijal koji vam je neophodan; ne treba da kupite dodatno platno, boju, film, predivo, LEGO kocke ili elektronske komponente. Kada je program napisan, može lako da se deli online sa celim svetom. I, mada ćete praviti greške prilikom programiranja, ta aktivnost je i dalje veoma zabavna.

O ovoj knjizi

Prvi deo ove knjige obuhvata osnovne koncepte Python programiranja, a drugi deo različite zadatke koje možete da automatizujete na računaru. Svako poglavlje u drugom delu ima programe projekta koje možete da prostudirate. Evo kratkog pregleda onoga što ćete pronaći u svakom poglavlju:

Deo I: Osnove Python programiranja

Poglavlje 1: **Osnove Pythona** – Obuhvaćeni su izrazi, najosnovnije vrste Python instrukcija i načini na koje možete da upotrebite Python interaktivnu konzolu programa za eksperimentisanje sa kodom.

Poglavlje 2: Kontrola procesa – Opisano je kako program odlučuje koju instrukciju da izvrši tako da kod može inteligentno da odgovara na različite uslove.

Poglavlje 3: Funkcije – Opisano je kako se definišu funkcije tako da možete da se organizuje kod u više blokova kojima se lako rukuje.

Poglavlje 4: Liste – Predstavljena je vrsta podataka liste i opisano je kako se organizuju podaci.

Poglavlje 5: Rečnici i strukturiranje podataka – Predstavljena je vrsta podataka i prikazani su napredniji načini za organizovanje podataka.

Poglavlje 6: Manipulisanje nizovima – Opisana je upotreba tekstualnih podaka (u Pythonu se nazivaju nizovi).

Deo II: Automatizacija zadataka

Poglavlje 7: Poklapanje šablona regularnim izrazima – Opisano je kako Python može da manipuliše nizovima i pretražuje tekstualne šablone koji sadrže regularne izraze.

Poglavlje 8: Čitanje i pisanje fajlova – Opisano je kako programi mogu da čitaju sadržaje tekstualnih fajlova i snimaju informacije u fajlove na hard drajvu.

Poglavlje 9: Organizovanje fajlova – Prikazano je kako Python može da kopira, pomera, menja naziv i briše veliki broj fajlova mnogo brže nego korisnik. Takođe su opisani kompresovanje i dekompresovanje fajlova.

Poglavlje 10: Ispravljanje grešaka – Prikazano je kako se upotrebljavaju različite alatke Pythona za pronalaženje i ispravljanje grešaka.

Poglavlje 11: Web isecanje – Prikazano je kako se pišu programi koji mogu automatski da preuzmu web stranice i raščlane ih za informacije. To se naziva web isecanje.

Poglavlje 12: Upotreba Excel tabela – Opisano je programsko manipulisanje Excel tabelama da ne bi trebalo da ih čitate. Ovo je korisno kada imate stotine i hiljade dokumenata koje treba da analizirate.

Poglavlje 13: Upotreba PDF i Word dokumenata – Opisano je programsko čitanje Word i PDF dokumenata.

Poglavlje 14: Upotreba CSV fajlova i JSON podataka – Nastavljeno je objašnjenje kako se programski manipuliše dokumentima za CSV i JSON fajlove.

Poglavlje 15: Zadržavanje vremena, raspoređivanje zadataka i pokretanje programa – Opisano je kako Python programi rukuju vremenom i datumom i kako se zadaje računaru da izvršava zadatke u određeno vreme. U ovom poglavlju je opisano i kako Python programi mogu da pokreću druge programe.

Poglavlje 16: Slanje e-maila i tekstualnih poruka – Opisano je kako da pišete programe koji mogu da šalju e-mailove i tekstualne poruke u svoje ime.

Poglavlje 17: Manipulisanje slikama – Opisano je kako da programski manipulišete slikama kao što su JPEG ili PNG fajlovi.

Poglavlje 18: Kontrolisanje tastature i miša pomoću GUI automatizacije – Opisano je kako da programski kontrolišete miš i tastaturu za automatizaciju klikova i pritiska tastera.

Preuzimanje i instaliranje Pythona

Možete da preuzmete Python za Windows, OS X ili Ubuntu besplatno sa adrese http://python.org/downloads/. Ako preuzmete najnoviju verziju sa stranice za preuzimanje na web sajtu, svi programi u ovoj knjizi bi trebalo da funkcionišu.

UPOZORENJE

Obavezno preuzmite verziju Python 3 (kao što je 3.4.0). Programi u ovoj knjizi su pisani za pokretanje na verziji Python 3 i možda se neće pravilno pokretati na verziji Python 2.

Na stranici za preuzimanje pronaći ćete Python instalacioni program za 64-bitne i 32-bitne računare za svaki operativni sistem, pa prvo proverite koji instalacioni program je vama potreban. Ako ste računar kupili 2007. godine ili kasnije, verovatno imate 64-bitni sistem. U suprotnom, imate 32-bitnu verziju. Evo kako se vrši provera:

- Na Windows sistemu selektujte Start ▸ Control Panel ▸ System i proverite da li za opciju System Type piše 64-bit ili 32-bit.
- Na OS X sistemu kliknite na Apple eni, selektujte **About This Mac ▸ More Info ▸ System Report ▸ Hardware**, a zatim potražite polje **Processor Name**. Ako je napisano bilo šta drugo (uključujući Intel Core 2 Duo), imate 64-bitnu mašinu.
- Na Ubuntu Linux sistemu otvorite Terminal i pokrenite komandu `uname -m`. Odgovor i686 znači da je sistem 32-bitni, a odgovor x86_64 znači da je sistem 64-bitni.

Na Windows sistemu preuzmite Python instalacioni program (naziv fajla će se završavati ekstenzijom .msi) i dvostruko kliknite na njega. Pratite instrukcije koje instalacioni program prikazuje na ekranu da biste instalirali Python, kao što je ovde izlistano:

1. Selektujte **Install for All Users**, a zatim kliknite na **Next**.
2. Instalirajte program u direktorijum C:\Python34 klikom na dugme **Next**.
3. Ponovo kliknite na **Next** da biste preskočili odeljak Customize Python.

Na Mac OS X sistemu preuzmite .dmg fajl koji odgovara vašoj verziji OS X-a i dvostruko kliknite na njega. Pratite instrukcije koje instalacioni program prikazuje na ekranu da biste instalirali Python, kao što je ovde izlistano:

1. Kada se DMG paket otvori u novom prozoru, dvostruko kliknite na fajl Python.mpkg. Možda će biti potrebno da unesete lozinku administratora.
2. Kliknite na dugme **Continue** u odeljku Welcome i kliknite na **Agree** da biste prihvatili licencu.
3. Selektujte HD Macintosh (ili koji god naziv ima vaš hard drajv) i kliknite na **Install**.

Ako koristite Ubuntu sistem, možete da instalirate Python iz Terminala prateći sledeće korake:

1. Otvorite prozor Terminal.
2. Ukucajte sudo apt-get install python3.
3. Ukucajte sudo apt-get install idle3.
4. Ukucajte sudo apt-get install python3-pip.

Pokretanje IDLE-a

Iako je *Python prevodilac* program koji pokreće Python programe, *interactive development environment (IDLE)* softver je mesto na koje ćete uneti programe, kao u procesor reči. Pokrenite sada IDLE.

- Na Windows 7 sistemu ili novijem kliknite na ikonicu Start u donjem levom uglu ekrana, u polje za pretragu ukucajte IDLE i selektujte IDLE (Python GUI).

- Na Windows XP sistemu kliknite na dugme **Start**, a zatim selektujte **Programs ▸ Python 3.4 ▸ IDLE** (Python GUI).
- Na Mac OS X sistemu otvorite prozor **Finder**, kliknite na **Applications**, pa na **Python 3.4** a zatim na ikonicu **IDLE**.
- Na Ubuntu sistemu selektujte **Applications ▸ Accessories ▸ Terminal**, a zatim unesite **idle3** (možda ćete moći da kliknete na **Applications** na vrhu ekrana, da selektujete **Programming** i da kliknite na **IDLE 3**).

Interaktivna konzola

Bez obzira koji operativni sistem koristite, IDLE prozor koji će biti prikazan treba da bude uglavnom prazan, osim teksta koji izgleda otprilike ovako:

```
Python 3.4.0 (v3.4.0:04f714765c13, Mar 16 2014, 19:25:23) [MSC v.1600 64 bit
(AMD64)] on win32Type "copyright", "credits" or "license()" for more information.
>>>
```

Ovaj prozor se naziva *interaktivna konzola*. Konzola je program koji omogućava da unosite instrukcije u računar, kao što su Terminal ili Command Prompt na OS X i Windows sistemima. Pythonova interaktivna konzola omogućava unos instrukcija za pokretanje softvera Python prevodioca. Računar čita instrukcije koje unesete i odmah ih pokreće.

Na primer, u interaktivnu konzolu pored stavke >>> prompt unesite sledeće:

```
>>> print('Hello world!')
```

Nakon što ukucate ovu liniju i pritisnete ENTER, interaktivna konzola treba da prikaže sledeći odgovor:

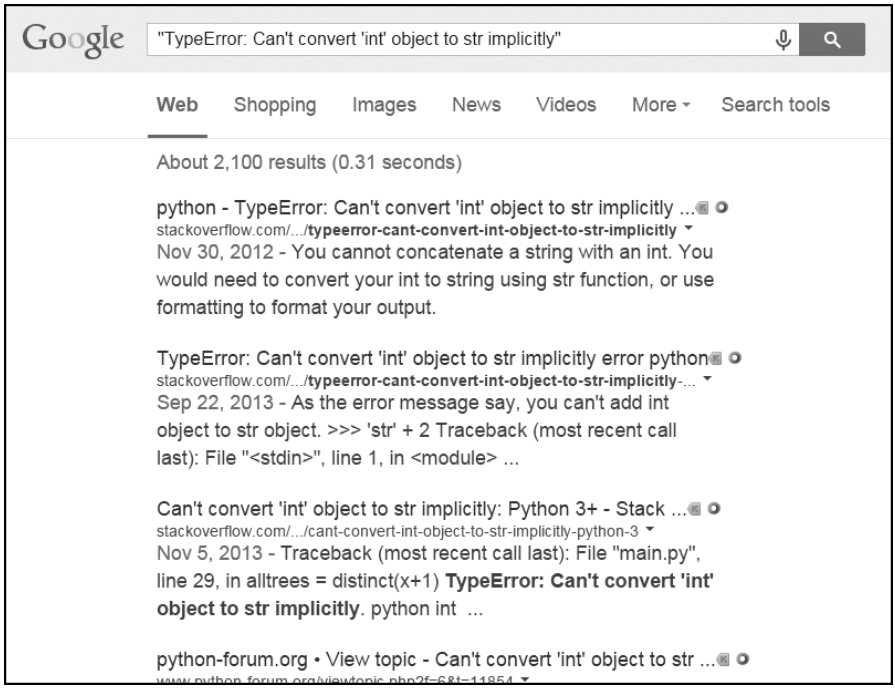
```
>>> print('Hello world!')
Hello world!
```

Pronalaženje pomoći

Samostalno rešavanje programerskih problema je lakše nego što možda mislite. Ako ne verujete, hajde namerno da izazovemo grešku: unesite `'42' + 3` u interaktivnu konzolu. Ne treba da znate šta znači ova instrukcija, ali bi rezultat trebalo da izgleda ovako:

```
>>> '42' + 3
❶ Traceback (most recent call last):
    File "<pyshell#0>", line 1, in <module>
      '42' + 3
❷ TypeError: Can't convert 'int' object to str implicitly
>>>
```

Poruka o grešci ❷ prikazana je ovde zato što Python nije mogao da razume instrukciju. U delu istorije steka ❶ poruke o grešci prikazana je specifična instrukcija i broj linije sa kojom Python ima problem. Ako niste sigurni šta da uradite sa određenom porukom o grešci, potražite na Internetu tačnu poruku. Unesite „**TypeError: Can‘t convert ‚int‘ object to str implicitly**“ (uključujući i navodnike) u omiljeni pretraživač i trebalo bi da vidite veliki broj linkova koji objašnjavaju šta znači ova poruka i šta je izaziva, kao što je prikazano na slici 0-2.



Slika 0-2 Google rezultati za poruku o grešci mogu da budu veoma korisni.

Često ćete pronaći da je neko drugi već postavio isto pitanje i da su neki drugi korisnici već odgovorili na njega. Ni jedna osoba ne može znati sve o programiranju, pa je, stoga, potrebno da programer svakodnevno traži odgovore na tehnička pitanja.

„Pametno“ postavljanje programerskih pitanja

Ako ne možete da pronađete odgovor pretraživanjem Interneta, pokušajte da pitate ljude na web forumima, kao što su Stack Overlow (http://stackoverflow.com/) ili “learn programming” subreddit na adresi http://reddit.com/r/learnprogramming/. Ali imajte na umu da postoje „pametni“ načini za postavljanje programerskih pitanja koji će pomoći i drugima i vama. Obavezno pročitajte odeljak Frequently Asked Questions koje ovi web sajtovi imaju o pravilnom načinu postavljanja pitanja.

Kada postavljate programersko pitanje, ne zaboravite da uradite sledeće:

- Objasnite šta pokušavate da uradite, a ne samo šta ste uradili. To će onima kojima ste se obratili za pomoć pomoći da odrede da li ste na pogrešnom putu.
- Specifikujte tačku u kojoj se greška desila. Da li se desila na samom početku programa ili samo nakon određene akcije koju ste izvršili?
- Kopirajte i pejstujte celu poruku o grešci i kod na stranicu http://pastebin.com/ ili http://gist.github.com/.

  Ovi web sajtovi olakšavaju deljenje velike količine koda sa ljudima preko Weba, bez rizika od gubitka formatiranja teksta. Možete, nakon toga, da postavite URL postavljenog koda u e-mail ili post na forumu. Na primer, ovo su delovi koda koje sam ja postavio: http://pastebin.com/SzP2DbFx/ i https://gist.github.com/asweigart/6912168/.
- Objasnite šta ste već pokušali da uradite da biste rešili problem. To će onima od kojih tražite pomoć dati do znanja da ste već uložili trud da rešite sami problem.
- Napišite koju verziju Pythona koristite (postoje neke ključne razlike između verzije 2 Python prevodioca i verzije 3 Python prevodioca). Navedite i koji operativni sistem i verziju koristite.
- Ako se greška javi nakon što izvršite promene u kodu, objasnite tačno šta ste promenili.
- Napomenite da li se greška javlja svaki put kada pokrenete program ili samo nakon što izvršite određene akcije (opišite te akcije).

Uvek pratite online bonton. Na primer, nemojte da postavljate pitanje napisano velikim slovima ili da postavljate nerazumne zahteve ljudima koji pokušaju da vam pomognu.

Rezime

Za većinu ljudi računar je samo uređaj, ali ne i alat. Međutim, učenjem programiranja dobićete pristup jednoj od najmoćnijih alatki modernog sveta, a uz to ćete se i zabaviti. Programiranje nije operacija na mozgu – ono je dobro i za amatere za eksperimentisanje i pravljenje grešaka.

Ja volim da pomažem ljudima da otkriju Python. Pišem tutorijale za programiranje na mom blogu na adresi http://inventwithpython.com/blog/, a pitanja mi možete postaviti na adresi al@inventwithpython.com.

U toku upoznavanja programiranja ćete možda imati neka pitanja koja su van delokruga ove knjige. Ne zaboravite da su postavljanje efektivnog pitanja i poznavanje načina kako da se pronađu odgovori veoma vredni na ovom programerskom putovanju.

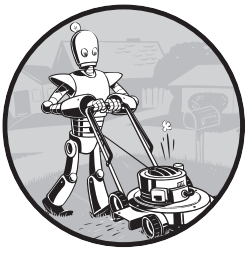
Započnimo to putovanje!

DEO I

OSNOVE PYTHON PROGRAMIRANJA

1

OSNOVE PYTHONA



PROGRAMSKI JEZIK PYTHON IMA ŠIROK SPEKTAR SINTAKSNE KONSTRUKCIJE, STANDARDNE FUNKCIJE BIBLIOTEKE I INTERAKTIVNE FUNKCIJE RAZVOJNOG OKRUŽENJA. SREĆOM, MOŽETE DA IGNORIŠETE VEĆINU OVIH STAVKI; TREBA SAMO DA NAUČITE ONO ŠTO JE DOVOLJNO ZA PISANJE KORISNIH MALIH PROGRAMA.

Međutim, neophodno je najpre da naučite osnovne koncepte programiranja. Možda ćete pomisliti da su ovi koncepti tajnoviti i dosadni, ali kada budete stekli neko znanje i praksu, moći ćete da zapovedate vašem računaru da izvršava neverovatne podvige.

U ovom poglavlju dato je nekoliko primera koji će vas ohrabriti da kucate u interaktivnu konzolu, što će omogućiti da izvršavate Python instrukcije jednu po jednu, a rezultat će vam biti odmah prikazan. Upotreba interaktivne konzole je odlična da biste naučili šta rade osnovne Python instrukcije, pa, stoga, isprobajte neke od njih dok čitate ovo poglavlje. Mnogo bolje ćete zapamtiti sve ako i uradite zadatak nego ako ga samo pročitate.

Unošenje izraza u interaktivnu konzolu

Interaktivnu konzolu pokrenite tako što ćete pokrenuti IDLE, koji je instaliran u uvodu ove knjige zajedno sa Pythonom. Na Windows sistemu otvorite meni **Start**, pa selektujte **All Programs ▸ Python 3.3** i IDLE (Python GUI). Na OS X sistemu selektujte **Applications ▸ MacPython 3.3 4 IDLE**. Na Ubuntu sistemu otvorite novi prozor Terminala i unesite idle3.Trebalo bi da se otvori prozor koji sadrži >>> prompt; to je interaktivna konzola. Unesite 2 + 2 u prompt da bi Python izvršio jednostavno izračunavanje.

```
>>> 2 + 2
4
```

IDLE prozor bi sada trebalo da prikazuje tekst sličan ovome:

```
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:06:53) [MSC v.1600 64 bit (AMD64)]
on win32
Type  copyright“,  credits“ or  license()“ for more information.
>>> 2 + 2
4
>>>
```

U Pythonu 2 + 2 se naziva *izraz*, što je i najosnovnija vrsta programerske instrukcije u jeziku. Izraz se sastoji od *vrednosti* (kao što je 2) i *operatora* (kao što je +); oni uvek mogu da se *smanje* na jednu vrednost. To znači da možete da upotrebite izraze bilo gde u Python kodu gde možete da upotrebite vrednost.

U prethodnom primeru izraz 2 + 2 je vrednovan kao jedna vrednost - 4. Jedna vrednost bez operatora se takođe smatra izrazom, mada se on svodi samo na sebe samog, kao što je ovde prikazano:

```
>>> 2
2
```

GREŠKE SU DOBRE!

Program će pasti ako sadrži kod koji računar ne može da „razume“, što će izazvati da Python prikaže poruku o grešci. Ta poruka neće pokvariti vaš računar, pa ne treba da se plašite ako napravite grešku. Pad sistema samo znači da je program neočekivano prestao da radi.

Ako želite da saznate više o poruci o grešci, možete da potražite tu tekstualnu poruku online i otkrijete više o specifičnoj grešci. Takođe možete da pogledate izvore na adresi http://nostarch.com/automatestuff/ i vidite listu uobičajenih Python poruka o greškama i njihova značenja.

Postoji mnogo drugih operatora koje možete da upotrebite u Python izrazima. Na primer, u tabeli 1-1 izlistani su svi matematički operatori u Pythonu.

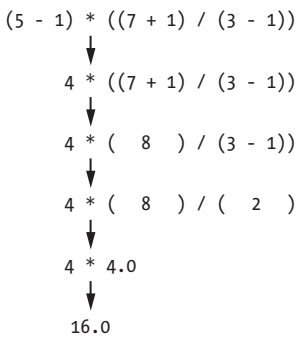
Tabela 1-1 Matematički operatori od najvišeg do najnižeg prioriteta

Operator	Operacija	Primer	Procenjuje se…
**	Eksponent	2 ** 3	8
%	Modul/ostatak	22 % 8	6
//	Deljenje celog broja/količnik	22 // 8	2
/	Deljenje	22 / 8	2.75
*	Množenje	3 * 5	15
-	Oduzimanje	5 - 2	3
+	Sabiranje	2 + 2	4

Redosled operatora (takođe se naziva *prioritet*) za Pythonove matematičke operatore je sličan kao u matematici. Prvo se izračunava operator **, zatim operatori *, /, //, i %, (sleva nadesno), a na kraju operatori + i - (takođe sleva nadesno). Možete da upotrebite zagrade da biste prepisali uobičajeni prioritet ako je to potrebno. U interaktivnu konzolu unesite sledeći izraz:

```
>>> 2 + 3 * 6
20
>>> (2 + 3) * 6
30
>>> 48565878 * 578453
28093077826734
>>> 2 ** 8
256
>>> 23 / 7
3.2857142857142856
>>> 23 // 7
3
>>> 23 % 7
2
>>> 2      +           2
4
>>> (5 - 1) * ((7 + 1) / (3 - 1))
16.0
```

U svakom slučaju, vi, kao programer, treba da unesete izraz, ali Python obavlja teži deo izračunavanja i svođenja na jednu vrednost. On će nastaviti da izračunava delove izraza, dok ne dobije jednu vrednost, kao što je prikazano na slici 1-1.



Slika 1-1 Izračunavanjem izraz se svodi na jednu vrednost.

Ova pravila za spajanje operatora i vrednosti za formiranje izraza su osnovni deo Pythona kao programskog jezika, isto kao i gramatička pravila, koja nam pomažu u komunikaciji. Evo primera:

This is a grammatically correct English sentence.

This grammatically is sentence not English correct a.

Drugu liniju je teško raščlaniti, zato što ne prati pravila engleskog jezika. Slično tome, ako ukucate pogrešnu Python instrukciju, Python neće moći da je „razume“ i prikazaće poruku o grešci SyntaxError, kao što je prikazano ovde:

```
>>> 5 +
  File  <stdin>", line 1
    5 +
      ^
SyntaxError: invalid syntax
>>> 42 + 5 + * 2
  File  <stdin>", line 1
    42 + 5 + * 2
             ^
SyntaxError: invalid syntax
```

Uvek možete da testirate i vidite da li instrukcija funkcioniše, tako što ćete je ukucati u interaktivnu konzolu. Ne brinite da ćete pokvariti računar: najgore što može da se desi je da Python odgovori porukom o grešci. Profesionalni programeri stalno dobijaju poruke o grešci dok pišu kodove.

Vrste podataka celog broja, pokretne tačke i niza

Zapamtite da su izrazi samo vrednosti kombinovane sa operatorima i uvek će se izračunavati do jedne vrednosti. *Vrsta podataka* je kategorija za vrednosti, a svaka vrednost pripada samo jednoj vrsti podataka. Najčešće vrste podataka u Pythonu su izlistane u tabeli 1-2. Vrednosti -2 i 30, na primer, nazivaju se vrednosti celog broja. Podatak celog broja (ili

int) ukazuje na vrednosti koje su *ceo broj*. Brojevi sa decimalnim tačkama, kao što je 3.14, nazivaju se brojevi sa *pokretnom tačkom* (ili *plutajućom tačkom*). Čak i ako je vrednost 42 ceo broj, vrednost 42.0 je broj sa pokretnom tačkom.

Tabela 1-2 Uobičajene vrste podataka

Vrsta podatka	Primeri
Ceo broj	-2, -1, 0, 1, 2, 3, 4, 5
Brojevi sa pokretnom tačkom	-1.25, -1.0, --0.5, 0.0, 0.5, 1.0, 1.25
Nizovi	‚a', ‚aa', ‚aaa', ‚Hello!', ‚11 cats'

Python programi takođe mogu da imaju tekstualne vrednosti koje se nazivaju *nizovi* ili *strs* (izgovara se „*stirs*"). Niz uvek obuhvatite karakterima navodnika (‘), kao što je ‚Hello' ili ‚Goodbye cruel world!', da bi Python znao gde niz počinje i gde se završava. Možete čak da imate niz bez ijednog karaktera unutar njega ‚', koji se naziva *prazan niz*. Nizovi su detaljnije opisani u Poglavlju 4.

Ako ikada vidite poruku o grešci SyntaxError: EOL while scanning string literal, verovatno nije ukucan krajnji karakter navodnika na kraju niza, kao u ovom primeru:

```
>>> 'Hello world!
SyntaxError: EOL while scanning string literal
```

Spajanje i ponavljanje niza

Značenje operatora može da se promeni na osnovu vrste podatka vrednosti pored njega. Na primer, + je operator sabiranja kada vrši operaciju za dva cela broja ili broja sa pokretnom tačkom. Međutim, kada je + upotrebljen za dve vrednosti niza, on *spaja nizove* kao operator spajanja niza. U interaktivnu konzolu unesite sledeće:

```
>>> 'Alice' + 'Bob'
'AliceBob'
```

Izraz se svodi na jednu vrednost - novu vrednost niza koja kombinuje tekst dva niza. Međutim, ako pokušate da upotrebite operator + u nizu i celom broju, Python neće „znati" kako da time rukuje i prikazaće poruku o grešci.

```
>>> 'Alice' + 42
Traceback (most recent call last):
  File  <pyshell#26>", line 1, in <module>
    'Alice' + 42
TypeError: Can't convert 'int' object to str implicitly
```

Poruka o grešci `Can't convert 'int' object to str implicitly` znači da Python „pretpostavlja“ da ste pokušali da spojite ceo broj sa nizom ‚Alice‘. Kod treba da eksplicitno konvertuje ceo broj u niz, zato što Python to ne može da uradi automatski (konvertovanje vrsta podataka će biti opisano u odeljku „Razlaganje programa“ kada budemo govorili o `str()`, `int()` i `float()` funkcijama).Operator * se koristi za množenje kada vrši operaciju za dva cela broja ili broja sa pokretnom tačkom. Međutim, kada se koristi u jednoj vrednosti niza i jednoj vrednosti celog broja, on postaje operator ponavljanja niza. Unesite niz pomnožen sa brojem u interaktivnu konzolu da biste videli ovu operaciju u akciji.

```
>>> 'Alice' * 5
'AliceAliceAliceAliceAlice'
```

Izraz se svodi na jednu vrednost niza koja replicira originalni broj jednak sa vrednošću celog broja. Ponavljanje niza je koristan trik, ali se ne upotrebljava toliko često kao spajanje niza.

Operator * može da se koristi samo sa dve numeričke vrednosti (za množenje) ili za jednu vrednost niza i jednu vrednost celog broja (za ponavljanje niza). U suprotnom, Python će samo prikazati poruku o grešci.

```
>>> 'Alice' * 'Bob'
Traceback (most recent call last):
  File  <pyshell#32>", line 1, in <module>
    'Alice' * 'Bob'
TypeError: can't multiply sequence by non-int of type 'str'
>>> 'Alice' * 5.0
Traceback (most recent call last):
  File  <pyshell#33>", line 1, in <module>
    'Alice' * 5.0
TypeError: can't multiply sequence by non-int of type 'float'
```

Jasno je zašto Python ne „razume “ ove izraze: ne možete da pomnožite dve reči, a teško je ponoviti proizvoljni niz za frakcioni broj puta.

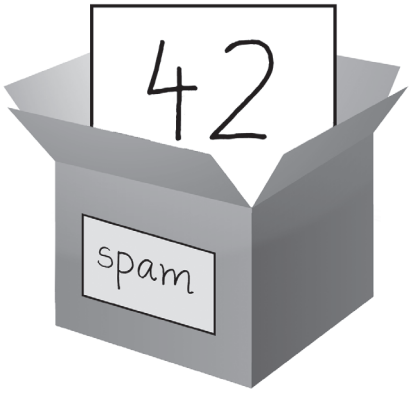
Čuvanje vrednosti u promenljivim

Promenljiva je kao kutija u memoriji računara, gde možete da čuvate pojedinačne vrednosti. Ako želite da kasnije upotrebite rezultat izračunatog izraza u programu, možete da ga sačuvate unutar promenljive.

Iskazi dodele

Vrednosti ćete čuvati u promenljivim sa *iskazom dodele*. Iskaz dodele sadrži naziv promenljive, znak jednakosti (koji se naziva *operator dodele*) i vrednost koja će biti sačuvana. Ako unesete iskaz dodele `spam = 42`, onda će promenljiva pod nazivom `spam` imati u sebi sačuvanu vrednost celog broja 42.

Zamislite promenljivu kao kutiju sa nazivom u koju je postavljena vrednost, kao na slici 1-2.



Slika 1-2 Izraz spam = 42 znači da kažete programu: „Promenljiva spam sada ima u sebi vrednost celog broja 42".

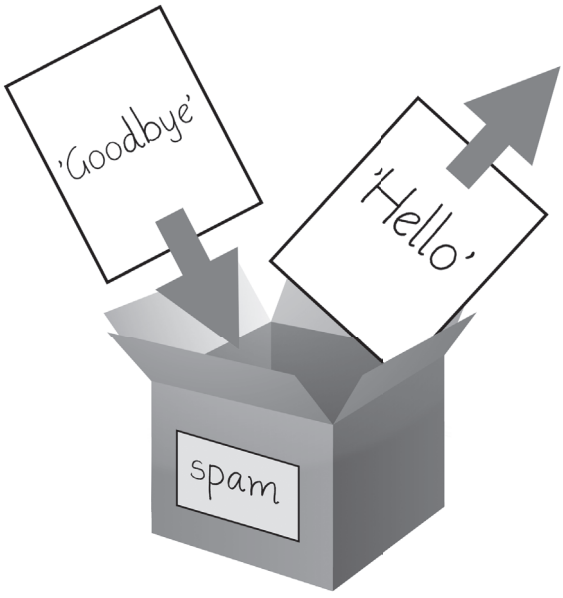
Na primer, u interaktivnu konzolu unesite sledeće:

```
❶ >>> spam = 40
  >>> spam
  40
  >>> eggs = 2
❷ >>> spam + eggs
  42
  >>> spam + eggs + spam
  82
❸ >>> spam = spam + 2
  >>> spam
  42
```

Promenljiva je *započeta* (ili kreirana) kada je prvi put vrednost sačuvana u njoj ❶. Nakon toga, možete da je upotrebite u izrazu sa drugim promenljivim i vrednostima ❷. Kada je promenljivoj dodeljena nova vrednost ❸, stara vrednost je zaboravljena i zbog toga je `spam` promenljiva procenjena na 42, umesto na 40 na kraju primera. To se naziva *prepisivanje* promenljive. U interaktivnu konzolu unesite sledeći kod da biste pokušali da prepišete niz:

```
>>> spam = 'Hello'
>>> spam
'Hello'
>>> spam = 'Goodbye'
>>> spam
'Goodbye'
```

Kao i kutija na slici 1-3, spam promenljiva u ovom primeru čuva ‚`Hello`', dok ovaj niz ne zamenite sa ‚`Goodbye`'.



Slika 1-3 Kada je nova vrednost dodeljena u promenljivu, stara vrednost je zaboravljena.

Nazivi promenljivih

Tabela 1-3 sadrži primere pravilnih naziva promenljivih. Možete da imenujete promenljivu kako god želite dok poštujete sledeća tri pravila:

1. Može biti samo jedna reč.
2. Mogu da se koriste samo karakteri slova, brojevi i donje crtice.
3. Ne može počinjati brojem.

Tabela 1-3 Pravilni i nepravilni nazivi promenljivih

Pravilni nazivi promenljivih	Nepravilni nazivi promenljivih
`balance`	`current-balance` (crtice nisu dozvoljene)
`currentBalance`	`current balance` (razmaci nisu dozvoljeni)
`current_balance`	`4account` (ne može se na početku nalaziti broj)
`_spam`	`42` (ne može se na početku nalaziti broj)
`SPAM`	`total_$um` (specijalni karakteri, kao što je $, nisu dozvoljeni)
`account4`	`'hello'` (specijalni karakteri, kao što je ,, nisu dozvoljeni)

Nazivi promenljivih su zavisni od veličine slova, što znači da su `spam`, `SPAM`, `Spam` i `sPaM` četiri različite promenljive. Pravilo u Pythonu je da se nazivi promenljivih pišu malim slovima.

U ovoj knjizi upotrebljava se camelcase za nazive promenljivih, umesto donjih crtica; odnosno, promenljive izgledaju ovako – `lookLikeThis`, umesto da izgledaju ovako - `loo-`

king_like_this. Neki iskusni programeri možda će istaći da zvanični stil Python koda PEP 8 navodi da bi trebalo da se upotrebe donje crtice. Ja i dalje preferiram camelcase i ukazujem na to da je „besmislena doslednost izmišljotina malih umova“ u samom PEP-u 8:

> „Doslednost praćenja vodiča za stil je važna. Ali je najvažnije znati kada biti nedosledan – ponekad vodič za stil jednostavno nije primenljiv. Kada ste u nedoumici, upotrebite svoju najbolju procenu.“

Dobar naziv za promenljivu opisuje podatke koje ona sadrži. Zamislite da se selite u novu kuću i sve kutije za selidbu označite sa „*stvari*“. Kako da pronađete u toj hrpi kutija ako vam je nešto hitno potrebno? Nazivi promenljivih spam, eggs i bacon se koriste kao generički nazivi za primere u ovoj knjizi i u većem delu Python dokumentacije (nazivi su inspirisani Monty Pythonovim skečom „Spam“), ali u vašim programima opisni naziv će vam pomoći da kod bude mnogo čitkiji.

Vaš prvi program

Iako je interaktivna konzola dobra za pokretanje Python instrukcija, jedne po jedne, da biste napisali ceo Python program, instrukcije ćete kucati u *fajl editor*. Fajl editor je sličan editorima teksta, kao što su Notepad ili TextMate, ali postoje neke specifične funkcije za pisanje u izvornom kodu. Da biste otvorili fajl editor u IDLE-u, selektujte **File ➡ New Window**.

Prozor koji će se otvoriti trebalo bi da sadrži kursor koji čeka vaš unos, ali se razlikuje od interaktivne konzole, koja pokreće Python instrukcije čim pritisnete Enter. Fajl editor omogućava da kucate više instrukcija, snimite fajl i pokrenete program. Evo kako možete da razlikujete fajl editor i interaktivnu konzolu:

- Prozor interaktivne konzole će uvek biti onaj u kojem se nalazi >>> prompt.
- Prozor fajl editora neće imati >>> prompt.

Sada je vreme da kreirate svoj prvi program! Kada se otvori prozor fajl editora, ukucajte u njega sledeće:

```
❶ # This program says hello and asks for my name.

❷ print('Hello world!')
  print('What is your name?')    # ask for their name
❸ myName = input()
❹ print('It is good to meet you, ' + myName)
❺ print('The length of your name is:')
  print(len(myName))
  ❻ print('What is your age?')    # ask for their age
  myAge = input()
  print('You will be ' + str(int(myAge) + 1) + ' in a year.')
```

Kada unesete izvorni kod, snimite ga da ne biste morali da ga kucate svaki put kada pokrećete IDLE. Iz menija na vrhu prozora fajl editora selektujte opciju **File ▸ Save As**. U prozoru Save As unesite `hello.py` u polje File Name, a zatim kliknite na **Save**.

Dok kucate programe, s vremena na vreme bi trebalo da ih snimite. Ako to radite, nećete izgubiti kod kad sistem padne ili slučajno zatvorite IDLE. Kao prečicu za snimanje možete da upotrebite tastere Ctrl-S na Windows i Linux sistemu ili Command-⌘ na OS X sistemu da biste snimili fajl.

Kada je fajl snimljen, pokrenite program. Selektujte komandu **Run ▸ Run Module** ili samo pritisnite taster **F5**. Program bi trebalo da se pokrene u prozoru interaktivne konzole, koji je otvoren kada ste prvi put pokrenuli IDLE. Ne zaboravite da treba da pritisnete F5 u prozoru fajl editora, a ne u prozoru interaktivne konzole. Unesite naziv programa kada se to od vas zatraži. Izvod programa u interaktivnoj konzoli bi trebalo da izgleda, otprilike, ovako:

```
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:06:53) [MSC v.1600 64 bit (AMD64)]
on win32
Type  copyright“,  credits“ or  license()“ for more information.
>>> ================================ RESTART ================================
>>>
Hello world!
What is your name?
Al
It is good to meet you, Al
The length of your name is:
2
What is your age?
4
You will be 5 in a year.
>>>
```

Kada nema više linija koda za izvršavanje, Python program se *isključuje*; odnosno, prestaje da se pokreće (takođe može da se kaže da se Python program zatvara).

Možete da zatvorite fajl editor, tako što ćete kliknuti na X na vrhu prozora. Da biste ponovo učitali snimljeni program, selektujte komandu **File ▸ Open** iz menija. Kada kliknete na komandu, u prozoru koji će se otvoriti izaberite stavku **`hello.py`** i kliknite na dugme **Open**. Program `hello.py`, koji ste prethodno snimili, sada bi trebalo da se otvori u prozoru fajl editora.

Analiza programa

Dok je novi program otvoren u fajl editoru, pogledajte na brzinu Python instrukcije koje koristi, tako što ćete analizirati šta radi svaka linija koda.

Komentari

Sledeća linija naziva se *komentar*.

```
❶ # This program says hello and asks for my name.
```

Python ignoriše komentare, koje možete da upotrebite za pisanje beležaka ili da biste se podsetili šta određena linija koda pokušava da izvrši. Svaki tekst u ostatku linije iza znaka tarabe (#) je deo komentara.

Ponekad će programeri postaviti # ispred linije koda da bi je privremeno uklonili dok testiraju program. To se naziva *obeležavanje komentara*, koje može biti veoma korisno kada pokušavate da shvatite zašto program ne funkcioniše. Možete kasnije da uklonite # oznaku kada budete spremni da vratite liniju koda u program.

Python takođe ignoriše praznu liniju iza komentara. Možete da dodate onoliko praznih linija u program koliko želite. To će olakšati čitanje koda, kao pasusi u knjizi.

Funkcija print()

Funkcija `print()` prikazuje vrednost niza unutar zagrada na ekranu.

```
❷ print('Hello world!')
  print('What is your name?') # ask for their name
```

Linija `print('Hello world!')` znači „Odštampaj tekst u nizu `'Hello world!'`“. Kada Python izvrši ovu liniju, možete da kažete da je *pozvao* funkciju `print()` i da je vrednost niza prosleđena funkciji. Vrednost koja je prosleđena funkciji naziva se *argument*. Videćete da navodnici nisu odštampani na ekranu. Oni samo označavaju gde je početak i kraj niza; nisu deo vrednosti niza.

NAPOMENA

Takođe možete da upotrebite ovu funkciju za postavljanje prazne linije na ekranu; samo pozovite funkciju `print()`, bez ikakvog teksta između zagrada.

Kada pišete naziv funkcije, otvorena i zatvorena zagrada na kraju identifikuju funkciju. Zbog toga ćete u ovoj knjizi videti `print()` umesto `print`. U Poglavlju 2 detaljnije su opisane funkcije.

Funkcija input()

Funkcija `input()` čeka da korisnik ukuca tekst na tastaturi i pritisne **ENTER**.

```
❸ myName = input()
```

Pozivanje ove funkcije svodi vrednost na niz jednak sa tekstom korisnika, a prethodna linija koda dodeljuje promenljivu `myName` u ovu vrednost niza.

O pozivanju funkcije `input()` možete da razmišljate kao o izrazu koji izračunava bilo koji niz koji korisnik unese. Ako je korisnik uneo `'Al'`, onda bi izraz trebalo da se svede kao `myName = 'Al'`.

Štampanje imena korisnika

Poziv sledeće funkcije `print()` sadrži, u stvari, izraz ‚`It is good to meet you, ' + myName` između zagrada.

```
❹ print('It is good to meet you, ' + myName)
```

Ne zaboravite da izraz uvek može da se svede na jednu vrednost. Ako je `'Al'` vrednost koja je sačuvana u izrazu `myName` u prethodnoj liniji, onda se ovaj izraz svodi na `'It is good to meet you, Al'`. Ova vrednost jednog niza se prosleđuje funkciji `print()`, koja će je odštampati na ekranu.

Funkcija len()

Funkciji `len()` možete da prosledite vrednost niza (ili promenljivu koja sadrži niz), a funkcija je izračunava do vrednosti celog broja za broj karaktera u određenom nizu.

```
❺ print('The length of your name is:')
  print(len(myName))
```

Da biste to isprobali, u interaktivnu konzolu unesite sledeće:

```
>>> len('hello')
5
>>> len('My very energetic monster just scarfed nachos.')
46
>>> len('')
0
```

Kao i ovi primeri, funkcija len(myName) svodi vrednost na ceo broj. Zatim se ta vrednost prosleđuje funkciji `print()` da bi bila odštampana na ekranu. Vidite da funkcija `print()` omogućava da prosledite vrednosti celog broja ili vrednosti niza. Međutim, vidite i grešku koja se prikazuje kada u interaktivnu konzolu ukucate sledeće:

```
 >>> print('I am ' + 29 + ' years old.')
Traceback (most recent call last):
  File  <pyshell#6>", line 1, in <module>
    print('I am ' + 29 + ' years old.')
TypeError: Can't convert 'int' object to str implicitly
```

Funkcija `print()` ne izaziva ovu grešku, već je izaziva izraz koji ste pokušali da prosledite toj funkciji. Istu poruku o grešci dobićete ako u interaktivnu konzolu ukucate izraz kao samostalan.

```
>>> 'I am ' + 29 + ' years old.'
Traceback (most recent call last):
  File  <pyshell#7>", line 1, in <module>
    'I am ' + 29 + ' years old.'
TypeError: Can't convert 'int' object to str implicitly
```

Python prikazuje grešku, zato što operator + možete da upotrebite samo za sabiranje dva cela broja ili za spajanje dva niza. Ne možete da dodate ceo broj nizu, jer to u Pythonu nije gramatički tačno. To možete da ispravite upotrebom verzije niza celog broja, kao što je opisano u sledećem odeljku.

Funkcije str(), int() i float()

Ako želite da spojite ceo broj, kao što je 29, sa nizom da biste ih prosledili funkciji `print()`, treba da upišete vrednost ‚29', što je, u stvari, niz od `29`. Funkcija `str()` može da se prosledi celom broju i svešće ga na verziju vrednosti niza na sledeći način:

```
>>> str(29)
'29'
>>> print('I am ' + str(29) + ' years old.')
I am 29 years old.
```

Pošto funkcija (`29`) svodi vrednost na ‚`29`', izraz ‚`I am ' + str(29) + ' years old.`' svodi se na `'I am ' + '29' + ' years old.'`, a ta vrednost se ponovo svodi na `'I am 29 years old.'`. Ova vrednost se prosleđuje funkciji `print()`.Funkcije `str()`, `int()` i `float()` će svesti niz, ceo broj i oblike pokretne tačke vrednosti koje prosleđujete. Pokušajte da konvertujete neke vrednosti u interaktivnoj konzoli pomoću ovih funkcija i pogledajte šta će se desiti.

```
>>> str(0)
'0'
>>> str(-3.14)
'-3.14'
>>> int('42')
42
>>> int('-99')
-99
>>> int(1.25)
1
>>> int(1.99)
1
>>> float('3.14')
3.14
>>> float(10)
10.0
```

Prethodni primeri pozivaju funkcije `str()`, `int()` i `float()` i prosleđuju im vrednosti drugih vrsta podataka za dobijanje niza, celog broja ili oblika pokretne tačke tih vrednosti.

Funkcija `str()` je korisna kada imate ceo broj ili pokretnu tačku koju želite da dodate nizu. Funkcija `int()` je takođe korisna ako imate broj kao vrednost niza koji želite da upotrebite u nekim drugim izračunavanjima. Na primer, funkcija `input()` uvek vraća niz, čak i ako korisnik unese broj. Unesite `spam = input()` u interaktivnu konzolu i unesite `101` kada program zatraži tekst.

```
>>> spam = input()
101
>>> spam
'101'
```

Vrednost koja je sačuvana unutar spama nije ceo broj `101`, već je to niz `'101'`. Ako želite da izvršite izračunavanje koristeći vrednost u spamu, upotrebite funkciju `int()` da biste dobili oblik celog broja spama, a zatim sačuvajte ceo broj kao novu vrednost u spamu.

```
>>> spam = int(spam)
>>> spam
101
```

Sada bi trebalo da možete da tretirate promenljivu `spam` kao ceo broj, umesto kao niz.

```
>>> spam * 10 / 5
202.0
```

Zapamtite: ako prosledite vrednost funkciji `int()` koju ona ne može da svede na ceo broj, Python će prikazati poruku o grešci.

```
>>> int('99.99')
Traceback (most recent call last):
  File  <pyshell#18>", line 1, in <module>
    int('99.99')
ValueError: invalid literal for int() with base 10: '99.99'
>>> int('twelve')
Traceback (most recent call last):
  File  <pyshell#19>", line 1, in <module>
    int('twelve')
ValueError: invalid literal for int() with base 10: 'twelve'
```

Funkcija `int()` je takođe korisna ako treba da zaokružite broj sa pokretnom tačkom.

```
>>> int(7.7)
7
>>> int(7.7) + 1
8
```

U programu ste upotrebili funkcije `int()` i `str()` u najmanje tri linije da biste dobili vrednost odgovarajuće vrste podataka za kod.

```
❻ print('What is your age?') # ask for their age
  myAge = input()
  print('You will be ' + str(int(myAge) + 1) + ' in a year.')
```

Promenljiva `myAge` sadrži vrednost koju je vratila funkcija `input()`. Pošto funkcija `input()` uvek vraća niz (čak i ako korisnik ukuca broj), možete da upotrebite kod `int(myAge)` da biste vratili vrednost celog broja niza u promenljivoj `myAge`. Ova vrednost celog broja se, zatim, dodaje jedinici (1) i izrazu `int(myAge)` + 1.

Rezultat ovog dodavanja je prosleđen funkciji `str()` function: `str(int(myAge) + 1)`. Vrednost niza koja je vraćena je, zatim, spojena sa nizovima `'You will be ' i ' in a year.'` da bi se svela na jednu vrednost velikog niza. Ovaj veliki niz je na kraju prosleđen funkciji `print()` da bi bio prikazan na ekranu.

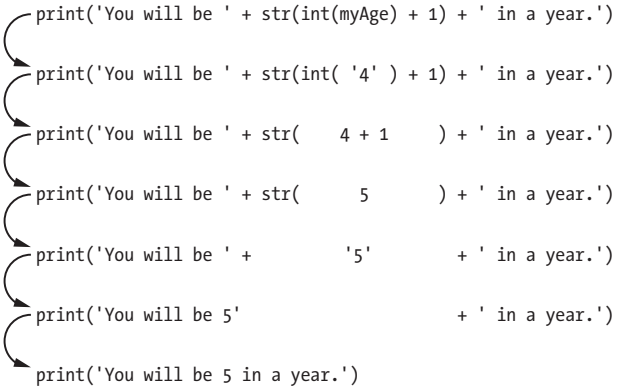
Recimo da korisnik unese niz `'4'` za promenljivu `myAge`. Niz `'4'` je konvertovan u ceo broj da biste mogli da mu dodate 1. Rezultat je 5. Funkcija `str()` konvertuje rezultat nazad na niz, pa možete da ga spojite sa drugim nizom `'in a year.'`, da biste kreirali finalnu poruku. Ovi koraci procene bi izgledali slično onima na slici 1-4.

EKVIVALENTNOST TEKSTA I BROJA

Iako se vrednost niza broja smatra potpuno drugačijom vrednošću od verzije celog broja ili broja sa pokretnom tačkom, ceo broj može da bude jednak sa brojem pokretne tačke.

```
>>> 42 == '42'
False
>>> 42 == 42.0
True
>>> 42.0 == 0042.000
True
```

Python razlikuje ovo zato što su nizovi tekstualni, dok su celi brojevi i brojevi pokretnog zareza brojevi.



Slika 1-4 Koraci svođenja ako je broj 4 sačuvan u promenljivoj myAge

Rezime

Možete da izračunate izraze pomoću kalkulatora ili da ukucate nadovezane nizove u procesoru reči. Možete čak da ponovite niz jednostavnim kopiranjem i pejstovanjem teksta. Međutim, izrazi i vrednosti njihovih komponenata (operatori, promenljive i pozivanje funkcija) su osnovni gradivni blokovi koji čine programe. Kada znate kako da rukujete ovim elementima, moći ćete da date instrukcije Pythonu da izračuna velike količine podataka umesto vas.

Dobro je da zapamtite različite vrste operatora (+, -, *, /, //, % i ** za matematičke operacije i + i * za operacije sa nizovima) i tri vrste podataka (cele brojeve, brojeve plutajuće tačke i nizove) koji su predstavljeni u ovom poglavlju.

Takođe su predstavljene neke različite funkcije. Funkcije `print()` i `input()` rukuju jednostavnim ispisom teksta (na ekranu) i unosom (pomoću tastature). Funkcija `len()` uzima niz i svodi ga na veliki broj karaktera u nizu. Funkcije `str()`, `int()` i `float()` će svesti vrednost na niz, ceo broj ili broj plutajuće tačke za vrednost koja im je prosleđena.

U sledećem poglavlju naučićete kako da ukažete Pythonu da inteligentno odluči koji kod će pokrenuti, koji kod će preskočiti i koji kod će ponoviti na osnovu vrednosti koje sadrži. To je poznato kao *kontrola toka*, koja omogućava da pišete programe koji će donositi inteligentne odluke.

Praktična pitanja

1. Koji od prikazanih su operatori, a koje su vrednosti?

```
*
'hello'
-88.8
-
/
+
5
```

2. Koji od prikazanih je promenljiva, a koje je niz?

```
spam
'spam'
```

3. Navedite tri vrste podataka.
4. Od čega je sastavljen izraz? Šta rade svi izrazi?
5. U ovom poglavlju predstavljeni su iskazi dodele, kao što je `spam = 10`. U čemu je razlika između izraza i iskaza?
6. Šta sadrži promenljiva bacon nakon pokretanja sledećeg koda?

```
bacon = 20
bacon + 1
```

7. Na šta bi trebalo da se svede sledeći izraz?

```
'spam' + 'spamspam'
'spam' * 3
```

8. Zašto je egg pravilan naziv promenljive, a 100 nepravilan?
9. Koje tri funkcije mogu da se upotrebe za dobijanje vrednosti celog broja, broja pokretne tačke ili niza za vrednost?
10. Zašto ovaj izraz izaziva grešku? Kako možete da ga ispravite?

```
'I have eaten ' + 99 + ' burritos.'
```

Dodatni kredit: Potražite online Python dokumentaciju za funkciju `len()`. Pronaći ćete je na web stranici pod naslovom „Built-in Functions“. Preuzmite listu funkcija koje Python sadrži, potražite šta radi funkcija `round()` i eksperimentišite njome u interaktivnoj konzoli.