POGLAVLJE 1

Karakteristike Python jezika

Hajde da počnemo jednom malom misterijom i njenim rešenjem. Šta mislite, šta sledeća dva reda znače?

```
(Red 1): (LC) N18,N2PL,N1,obrni.
(Red 2): (NL) PD1,K,K5,K2zaj,K1,obrni.
```

Izgleda tehnički, poput nekog uputstva. Zapravo, to je *obrazac za štrikanje*; deo koji opisuje kako preokrenuti petu na čarapi. Meni ovo ima smisla koliko mojoj mački ukrštene reči u novinama, ali moja supruga ovaj jezik razume savršeno. Ukoliko štrikate, ovo ima smisla i vama.

Hajde da probamo sa još jednim primerom. Odmah ćete uvideti njegovu svrhu, mada možda ne i njegov krajnji proizvod.

```
   1/2 š. putera ili margarina
   1/2 š. šlaga
   2 1/2 š. brašna
   1 k. soli
   1 K. šećera
   4 š. kuvanog pasiranog krompira (hladnog)

Neka svi sastojci budu hladni pre dodavanja brašna.
Pomešajte sve sastojke
Dobro izgnječite.
Napravite 20 kuglica. Neka budu hladne do sledećeg koraka.
Za svaku kuglu:
   Pospite brašno na krpu.
   Svaku kuglicu spljeskajte koristeći oklagiju.
   Pržite na ploči dok se ne pojave braon tačke.
   Okrenite i pržite drugu stranu.
```

Čak iako ne kuvate, verovatno ste prepoznali da se radi o *receptu*: lista sastojaka za kojom slede uputstva za pravljenje. Ali šta se ovime dobija? To je *lefse*, norveška poslastica koja liči na tortilju. Premažite malo margarina i džema ili šta već volite, zavijte i uživajte.

Obrazac za štrikanje i recept dele neke odlike:

- Ustaljene reči, skraćenice i simboli. Neki su poznati, neki tajanstveni.
- Postoje pravila oko toga šta se može reći, i gde – njihova *sintaksa*.
- Treba poštovati redosled operacija.
- Ponekad, ima i ponavljanja neke operacije (*petlja*), kao što je metoda za prženje svakog lefsea.
- Ponekad se usmerava na drugi niz operacija (u računarskoj terminologiji, *funkcija*). U receptu, možda ćete morati da potražite kako se pasira krompir u drugom receptu.
- Pretpostavlja se da čitalac već poznaje kontekst. U receptu se pretpostavlja da znate šta je voda i kako se prokuvava. U obrascu za štrikanje se pretpostavlja da umete da štrikate i pravite porube a da se ne ubodete prečesto.
- Postoji očekivani rezultat. U našim primerima, to je nešto za vaša stopala i nešto za vaš stomak. Samo ih nemojte pomešati.

Sve ove ideje ćete videti i u računarskim programima. Koristio sam nešto što nije program kako bih demonstrirao da programiranje nije nešto misteriozno. Samo se radi o učenju pravih reči i pravila.

Hajde da ostavimo ove zamene i vidimo pravi program. Šta on radi?

```
for countdown in 5, 4, 3, 2, 1, "hey!":
  print(countdown)
```

Ukoliko ste pogodili da je to Python program koji ispisuje linije:

```
5
4
3
2
1
hey!
```

onda znate da može biti jednostavnije da naučite Python nego obrazac za štrikanje. Takođe, možete vežbati pisanje Python programa u udobnosti i bezbednosti svog radnog stola, daleko od zastrašujućih opasnosti vrele vode i šiljatih igala.

Python program ima posebne reči i simbole – `for`, `in`, `print`, zareze, dvotačke, navodnike, itd – koji su bitni delovi sintakse ovog jezika. Dobra vest je ta što Python ima lepšu sintaksu i mnogo je ima manje, nego u većini računarskih jezika. Deluje prirodnije – kao recept.

Evo još jednog Python programa koji bira stereotipne izraze iz vesti (engl. *cliche*) iz Python *liste* i ispisuje ih:

```
cliches = [
   "At the end of the day",
   "Having said that",
   "The fact of the matter is",
   "Be that as it may",
   "The bottom line is",
   "If you will",
   ]
print(cliches[3])
```

Program ispisuje četvrtu frazu:

```
Be that as it may
```

Python lista kao `cliches` je niz vrednosti kojima se pristupa prema njihovoj udaljenosti od početka liste. Prva vrednost je `0`, a četvrta vrednost `3`.

Ljudi broje od 1, te može delovati čudno brojati od 0. Može biti lakše ukoliko razmišljate po principu udaljenosti umesto pozicija.

Liste su u Python-u vrlo česte, a u poglavlju 3 je prikazano kako se koriste.

Sledi drugi program koji ispisuje citat, s tim da se ovaj put na njega upućuje prema osobi koja ga je rekla, umesto njegove pozicije u listi:

```
quotes = {
   "Moe": "A wise guy, huh?",
   "Larry": "Ow!",
   "Curly": "Nyuk nyuk!",
   }
stooge = "Curly"
print(stooge, "says:", quotes[stooge])
```

Ukoliko biste pokrenuli ovaj mali program, on bi ispisao sledeće:

```
Curly says: Nyuk nyuk!
```

`quotes` je Python *rečnik* – komplet jedinstvenih *ključeva* (u ovom primeru, ime osobe koja služi za imenovanje – *stooge*) i pripisanih *vrednosti* (ovde, bitan citat te osobe). Korišćenjem rečnika, možete skladištiti i pretraživati stvari po imenu, što je neretko korisna alternativa za listu. Možete pročitati više o rečnicima u poglavlju 3.

U primeru sa stereotipnim izrazima koriste se uglaste zagrade (`[` i `]`) kako bi se napravila Python lista, a u primeru sa citatom se koriste vitičaste zagrade (`{` i `}`), kako bi se napravio Python rečnik. Ovo su primeri Python sintakse, a videćete još više u sledećim poglavljima.

A sada nešto sasvim drugačije: u Primeru 1-1 predstavljen je Python program koji izvodi kompleksnije nizove zadataka. Nemojte još uvek očekivati da razumete kako funkcioniše program; tome služi ova knjiga. Namera je da vas uvedem u to kako izgleda i kakav je osećaj raditi sa netrivijalnim Python programom. Ukoliko ste upoznati sa ostalim računarskim jezicima, procenite i uporedite Python.

Primer 1-1 vas povezuje sa YouTube veb-stranicom i uzima informacije o najviše ocenjenim video snimcima u datom trenutku. Ukoliko bi vratio normalnu veb-stranu prepunu teksta u HTML formatu, bilo bi teško pronaći informacije koje želite (govorim o *izdvajanju informacija sa interneta (*engl. *web scraping)* u odeljku „Pretraga i izdvajanje" na strani 237). Umesto toga, on vraća podatke u JSON formatu, koji je pogodniji za računare. JSON, ili JavaScript Object Notation je tekstualni format zgodan ljudima za čitanje, a opisuje tipove podataka, vrednosti i redosled vrednosti koje sadrži. On je kao mali programski jezik i postao je popularan način da se podaci razmenjuju među različitim računarskim jezicima i sistemima. Više ćete čitati o JSON formatu u „JSON" odeljku na strani 185.

Python programi mogu prevoditi JSON tekst u Python *strukture podataka* – kakve ćete videti u sledeća dva poglavlja – kao da ste napisali program da ih sam napravi. U ovom odgovoru od YouTube-a postoji mnogo podataka, tako da ću za ovaj primer samo ispisati naslove prvih šest video snimaka. Ponovo, ovo je jedan kompletan Python program koji možete pokrenuti i sami.

Primer 1-1. intro/youtube.py

```
import json
from urllib.request import urlopen
url = "https://gdata.youtube.com/feeds/api/standardfeeds/top_rated?alt=json"
response = urlopen(url)
contents = response.read()
text = contents.decode('utf8')
data = json.loads(text)
for video in data['feed']['entry'][0:6]:
   print(video['title']['$t'])
```

Poslednji put kada sam pokrenuo ovaj program, dobio sam ovakav izlaz:

```
Evolution of Dance – By Judson Laipply
Linkin Park – Numb
Potter Puppet Pals: The Mysterious Ticking Noise
"Chocolate Rain" Original Song by Tay Zonday
Charlie bit my finger – again !
The Mean Kitty Song
```

Ovaj mali Python program je uradio dosta toga u devet poprilično čitljivih linija koda. Ukoliko ne znate sve te termine, ne brinite; saznaćete ih u nekoliko sledećih poglavlja:

- Linija koda 1: uvozi sav kod iz Python *standardne biblioteke* pod imenom `json`
- Linija koda 2: uvozi samo `urlopen` funkciju iz standardne `urllib` biblioteke

- Linija koda 3: dodeljuje YouTube URL promenljivoj `url`
- Linija koda 4: povezuje se na veb server na tom URL i zahteva određeni *veb servis*
- Linija koda 5: uzima podatke iz odgovora i dodeljuje ih promenljivoj `contents`
- Linija koda 6: *dekodira* `contents` u tekstualni string u JSON formatu i dodeljuje ih promenljivoj `text`
- Linija koda 7: konvertuje `text` u `data` – Python strukture podataka o video snimcima
- Linija koda 8: stavlja informacije o jednom po jednom video snimku u promenljivu `video`
- Linija koda 8: koristi Python rečnik sa dve dimenzije (`data['feed']['entry']`) i *odsečak* (`[0:6]`)
- Linija koda 9: koristi `print` funkciju kako bi ispisao samo naslov video snimka.

Ove informacije o video snimcima su kombinacije raznih Python struktura podataka koje ste nedavno videli, a o kojima će biti reči u poglavlju 3.

U prethodnom poglavlju, koristili smo neke od modula Python standardne biblioteke (programi koji su uključeni u Python kad ga instalirate), ali u njima nema ničeg posebnog. Kod koji sledi prikazuje prepis za koje se koristi eksterni Python softverski paket pod imenom `requests`:

```
import requests
url = "https://gdata.youtube.com/feeds/api/standardfeeds/top_rated?alt=json"
response = requests.get(url)
data = response.json()
for video in data['feed']['entry'][0:6]:
   print(video['title']['$t'])
```

Ova nova verzija ima samo šest linija koda i pretpostavljam da je čitljivija većini ljudi. Imaću još mnogo toga da kažem o `requests` i ostalim eksterno autorizovanim Python softverskim paketima u poglavlju 5.

Python u praksi

Dakle, da li je učenje Python jezika vredno vremena i truda? Da li je on samo privremena moda ili igračka? Zapravo, on je tu još od 1991. (duže nego Java) i konstantno se nalazi u 10 najpopularnijih računarskih jezika. Ljude plaćaju da pišu Python programe – za ozbiljne stvari koje koristite svakodnevno, kao što su Google, YouTube, Dropbox, Netflix i Hulu. Lično sam ga koristio za pravljenje aplikacija koje su raznovrsne koliko i alatke za pretraživanje elektronske pošte ili veb-strana za elektronsku trgovinu. Python važi za produktivan jezik koji odgovara brzim organizacijama.

Python ćete naći u mnogim računarskim okruženjima, uključujući i sledeće:

- Komandna linija na monitoru ili terminal
- Grafički korisnički interfejs, uključujući i internet.
- Internet i kod klijenata i kod servera
- Serveri sadržaja koji podržavaju popularne sajtove
- *oblak* (serveri kojima upravlja treće lice)
- Mobilni uređaji
- Ugrađeni uređaji

Python programi su raznovrsni i kreću se od onih sa pojedinačnim skriptovima – poput onih koje ste videli do sada u ovom poglavlju – pa do sistema sa milion linija koda. Videćemo i njegovu upotrebu na veb-stranama, administraciji sistema i u upravljanju podacima. Takođe ćemo baciti pogled i na posebnu upotrebu Python jezika u umetnostima, nauci i poslovanju.

Python u poređenju sa jezikom X

Kakav je Python u poređenju sa ostalim jezicima? Kada i gde bi trebalo da ga izaberete umesto nekog drugog? U ovom odeljku, prikazaću primere koda iz ostalih jezika, čisto da biste mogli da vidite kako izgleda konkurencija. Od vas se *ne očekuje* da ih razumete ukoliko sa njima niste radili. (Dok dođete do primera Python jezika, možda će vam laknuti što niste morali da radite sa drugima.) Ukoliko vas zanima samo Python, nećete propustiti ništa ukoliko samo preskočite i odete na sledeći odeljak.

Svaki program treba da ispiše broj i kaže nešto o tom jeziku.

Ukoliko koristite komandnu liniju, ili prozor komandne linije, program koji čita ono što ste napisali, to pokreće i prikazuje rezultate, naziva se *shell* program. Windows komandna linija se zove `cmd`; ona pokreće *batch* fajlove sa sufiksom `.bat`. Linux i ostali sistemi slični Unix-u (uključujući i Mac OS X) imaju mnogo shell programa, a najpopularniji se zove bash ili `sh`. Mogućnosti shell programa su jednostavne, poput jednostavnih logičkih i džoker simbola kao na primer `*` u imenima fajlova. Komande možete sačuvati u fajlovima koji se zovu *shell skriptovi* i kasnije ih pokrenuti. Ovo su možda prvi programi na koje sta naišli kao programer. Problem je u tome što shell skriptovi nisu pogodni za programe koji imaju više od nekoliko stotina linija koda i znatno su sporiji od alternativnih jezika. Sledeći kod predstavlja mali shell program:

```
#!/bin/sh
language=0
echo "Language $language: I am the shell. So there."
```

Ukoliko ste ovo sačuvali u fajlu kao `meh.sh` i pokrenuli ga sa `sh meh.sh`, videli biste sledeće na svom ekranu:

```
Language 0: I am the shell. So there.
```

Stariji C i C++ su jezici prilično nižeg nivoa, koji se koriste kada je bitna brzina. Teže ih je naučiti i zahtevaju da vodite računa o mnogim detaljima, koji mogu dovesti do prekida rada programa, a i probleme je teško dijagnostifikovati. Evo ga osvrt na to kako izgleda jedan mali C program:

```
#include <stdio.h>
int main(int argc, char *argv[]) {
   int language = 1;
   printf("Language %d: I am C! Behold me and tremble!\n", language);
   return 0;
}
```

C++ poseduje porodične sličnosti sa C jezikom, ali sa prepoznatljivim karakteristikama:

```
#include <iostream>
using namespace std;
   int main() {
   int language = 2;
   cout <<"Language " <<language << \
      ": I am C++! Pay no attention to that C behind the curtain!" << \
      endl;
   return(0);
}
```

Java i C# su naslednici C i C++ koji imaju neke od problema ova dva, ali su donekle preterano opširni i restriktivni. Sledi primer u kojem je prikazano kako izgleda Java:

```
public class Overlord {
   public static void main (String[] args) {
      int language = 3;
      System.out.format("Language %d: I am Java! Scarier than C!\n", language);
   }
}
```

Ukoliko niste pisali programe ni na jednom od ovih jezika, možda ćete se zapitati: *šta* je sve *ovo*? Neki jezici su poprilično teški po pitanju sintakse. Ponekad se nazivaju i *statičkim jezicima* jer zahtevaju da za računar odredite neke detalje nižeg nivoa. Dajte da objasnim.

Jezici imaju *promenljive* – imena za vrednosti koje želite da koristite u programu. Statički jezici traže od vas da deklarišete tip svake promenljive: koliko prostora će zauzimati u memoriji, i šta ćete raditi sa njim. Računar ove informacije koristi kako bi kompajlirao program u *mašinski jezik* vrlo nižeg nivoa (napravljen specifično za računarski hardver kako bi ga lakše razumeo, ali je teže ljudima da ga razumeju). Proizvođači računarskih jezika često moraju odlučiti između toga da stvari olakšaju ili ljudima ili računarima. Deklarisanje tipova promenljivih pomaže računaru da pronađe greške i radi brže, ali mu je neophodno napredno ljudsko razmišljanje i pisanje. U velikom delu koda koji ste videli u C, C++, C# i Java primerima, bilo je neophodno navoditi i tipove promenljivih. Na primer, u svima je bila neophodna deklaracija `int` kako bi se promenljiva `language` gledala kao ceo broj. (Ostali tipovi podrazumevaju i brojeve u pokretnom zarezu kao što je 3.14159 i karaktere ili tekstualne podatke, koji se skladište drugačije.)

Zašto se onda nazivaju *statičkim* jezicima? Zato što promenljive u tim jezicima nikada ne mogu menjati svoj tip; oni su statički. Ceo broj je uvek i zauvek ceo broj.

Nasuprot tome, *dinamički jezici* (takođe poznati i kao *skript jezici*), ne teraju vas da deklarišete promenljive pre nego što ih unesete. Ukoliko ukucate nešto poput `x = 5`, dinamički jezik zna da je `5` ceo broj, samim tim i promenljiva `x`. Ovi jezici vam omogućavaju da postignete više sa manje linija koda. Umesto da budu kompajlirani, njih *prevodi* program koji se zove – iznenađenje! – *prevodilac*. Dinamički jezici su često sporiji od kompajliranih statičkih jezika, ali im se brzina unapređuje zajedno sa optimizacijom njihovih prevodilaca. Dugo vremena su se dinamički jezici koristili samo za kratke programe (skriptove) i to najčešće da bi duži programi pisani na statičkim jezicima pripremili podatke za obradu. Takvi programi se zovu *povezujući programi* (engl. *glue code*). Premda su dinamički jezici dobro skrojeni za tu svrhu, danas se i sa njima može odraditi većina ozbiljnijih zadataka obrade podataka.

Dinamički jezik za sveopštu upotrebu je mnogo godina bio **Perl**. Perl je vrlo moćan i ima velike biblioteke. Ipak, njegova sintaksa može biti čudnovata, a i čini se da je izgubio svoj zamah poslednjih nekoliko godina u odnosu na Python i Ruby. U sledećem primeru ćete videti jednu dosetku na Perl jeziku:

```
my $language = 4;
print "Language $language: I am Perl, the camel of languages.\n";
```

Ruby je noviji jezik. Nešto je preuzeto iz Perl jezika, a najpopularniji je zbog veb razvojnog alata, *Ruby on Rails*. Koristi se u mnogim istim oblastima kao i Python, dok se izbor između ova dva uglavnom svodi na stvar ukusa ili dostupne biblioteke za vašu pojedinačnu aplikaciju. Ovaj primer koda opisuje Ruby:

```
language = 5
puts "Language #{language}: I am Ruby, ready and aglow."
```

PHP, koji možete videti u primerima koji slede, veoma je popularan za veb programiranje zato što olakšava kombinovanu upotrebu HTML i koda. Međutim, PHP jezik sam po sebi ima dosta začkoljica i nije u tolikoj meri postao opšti standard kao jezik van veb programiranja.

```
<?PHP
$language = 6;
echo "Language $language: I am PHP. The web is <i>mine</i>, I say.\n";
```

Primer koji sledi predstavlja trijumf Python jezika:

```
language = 7
print("Language %s: I am Python. What's for supper?" % language)
```

Dakle, zašto Python?

Python je dobar sveopšti jezik visokog nivoa. Njegov dizajn ga čini vrlo *čitljivim*, što je bitnije nego kako zvuči. Svaki računarski jezik se piše samo jednom, ali se čita i ispravlja mnogo puta jer je mnogo ljudi uključeno. Čitljivost takođe olakšava i učenje i pamćenje, dakle samim tim je i *lakši za pisanje*. U poređenju sa ostalim popularnim jezicima, Python je jednostavniji za početnike i pomaže vam da ranije postanete produktivni, a opet ima i veliki broj mogućnosti koje možete istraživati kako budete sticali iskustvo.

Relativna sažetost Python jezika vam omogućava da pišete program koji je mnogo manji nego njegov ekvivalent u statičkom jeziku. Studije su pokazale da programeri imaju tendenciju da dnevno napišu otprilike isti broj linija koda – bez obzira na jezik – te bi pisanje upola manjeg broja linija koda udvostručilo vašu produktivnost, samo tako. Python je ne-toliko-tajno oružje mnogih kompanija koje ovo smatraju bitnim.

Python je najpopularniji jezik za uvodne kurseve iz računarskih nauka u vrhunskim američkim koledžima. Takođe je i najpopularniji jezik i za procenjivanje programerskih veština kod više od dve hiljade poslodavaca.

Naravno, uz to je i besplatan. Napišite šta god hoćete na Python-u i koristite ga svugde, slobodno. Niko ne može da pročita Python program i kaže: „Hm, vrlo ti je dobar taj programčić tu. Bila bi prava šteta kad bi mu se nešto desilo...”

Python funkcioniše skoro svugde i dolazi „sa baterijama” – u svojoj standardnoj biblioteci ima tonu korisnog softvera.

Međutim, možda je najbitniji razlog da koristite Python vrlo neočekivan: ljudima se on generalno *sviđa*. Oni zapravo uživaju da programiraju u njemu, umesto da ga tretiraju kao samo još jedan od alata kojim mogu da završe stvari. Često će reći kada treba da rade sa nekim drugim jezikom, kako im nedostaje neka od mogućnosti Python-a. To je ono što Python izdvaja od njegove konkurencije.

Kada ne koristiti Python

Python nije najadekvatniji jezik za svaku situaciju.

Nije podrazumevano da je svugde instaliran. U dodatku D je prikazano kako se Python instalira ukoliko ga već nemate na svom računaru.

Dovoljno je brz za većinu aplikacija, ali možda neće biti za neke od zahtevnijih. Ukoliko vaš program najveći deo vremena troši na računanje stvari (tehnički termin je *vezan za procesor*, *CPU-bound*), program napisan na C, C++ ili na Javi će u principu raditi brže nego njegov Python ekvivalent. Ali ne uvek!

- Ponekad bolji *algoritam* (postupno rešenje) u Pythonu pobeđuje naspram neefikasnog algoritma u C jeziku. Veća brzina programiranja u Python-u vam daje više vremena da eksperimentišete sa alternativama.
- U mnogim aplikacijama, program će biti besposlen dok čeka na odgovor sa nekog servera sa mreže. Procesor (centralna jedinica za obrađivanje, *čip* u računaru koji obavlja sve računanje) skoro da nije umešan; samim tim, ukupno vreme izvršavanja će biti približno između statičkih i dinamičkih programa.
- Standardni Python prevodilac napisan je u C jeziku i može se nadograditi korišćenjem C koda. O ovome ću govoriti u odeljku „Optimizujte svoj kod" na strani 327.
- Python prevodioci postaju sve brži. Java je na svom početku bila užasno spora i uloženo je dosta vremena i novca u njeno ubrzavanje. Python ne poseduje korporacija te su njegova poboljšanja bila postepenija. U odeljku „PyPy" na strani 330, govoriću o PyPy projektu i njegovim implikacijama.
- Moguće je da imate vrlo zahtevnu aplikaciju, i da šta god da uradite Python ne zadovoljava vaše potrebe. Onda, kao što je rekao Ijan Holm u filmu „Osmi putnik", onda vam upućujem svoje žaljenje. Najčešće alternative su C, C++, i Java, mada bi odgovor na vaš problem mogao biti i noviji jezik *Go* (koji izgleda kao Python a radi kao C).

Python 2 naspram Python-a 3

Najveći problem sa kojim ćete se susresti u ovom trenutku jeste to što postoje dve Python verzije. Python 2 je tu odavno i on je preinstaliran na Linux i Apple računarima. On je bio izvrstan jezik, ali ništa nije savršeno. U računarskim jezicima, kao i u mnogim drugim oblastima, neke greške su estetske i lake za popraviti, dok su druge teške. Teže ispravke (engl. hard fixes) su *nekompatibilne*: novi programi napisani sa njima neće raditi na starim Python sistemima dok stari programi napisani pre popravke neće raditi na novom sistemu.

Pythonov autor (Gvido van Rosum) i ostali, odlučili su da spoje teže ispravke u jedno i nazvali ga Pythonom 3. Python 2 je prošlost, Python 3 budućnost. Poslednja verzija Python-a 2 je 2.7 i ona će biti podržana dugo vremena, ali je to kraj reda; neće biti Python-a 2.8. Nove izmene biće na Pythonu 3.

U ovoj knjizi predstavljen je Python 3. Ukoliko ste koristili Python 2, skoro je identičan. Najočiglednija promena jeste način na koji se poziva **`print`**. Najbitnija promena je tretiranje *Unikod* karaktera, što će biti pokriveno u poglavlju 2 i poglavlju 7. Konverzija popularnog Python softvera bila je postepena, sa klasičnim kokoška-jaje analogijama. Ali, izgleda da smo došli do prekretnice.

Instaliranje Python-a

Kako ne bih prepunio ovo poglavlje, detalji o instaliranju Python-a 3 su u dodatku D. Ukoliko nemate Python 3, ili nista sigurni, idite na taj odeljak i pogledajte šta da učinite.

Pokretanje Python-a

Nakon što ste instalirali Python 3, možete ga koristiti da pokrećete programe iz ove knjige kao i vaš sopstveni Python kod. Kako se konkretno pokreće Python program? Postoje dva glavna načina:

- *Interaktivni prevodilac* koji dolazi uz Python vam daje mogućnost da eksperimentišete sa malim programima. Unosite komande liniju po liniju i odmah vidite rezultat. Usled malog vremenskog razmaka između pisanja i toga da vidite rezultat, možete eksperimentisati brže. Koristeći interaktivnog prevodioca kako bih vam demonstrirao mogućnosti jezika, a te iste komande možete unositi i u vašem sopstvenom Python okruženju.
- Za sve ostalo, skladištite vaše Python programe u tekstualnim fajlovima, uglavnom sa ekstenzijom `.py`, i pokrenite ih tako što ćete ukucati `python` i zatim imena tih fajlova.

Hajde da sada isprobamo obe metode.

Korišćenje interaktivnog prevodioca

Veći deo primera koda u ovoj knjizi koristi interaktivnog prevodioca. Kada unesete iste komande koje vidite u primerima i dobijete iste rezultate, znaćete da ste na pravom putu.

Prevodilac se pokreće unošenjem samo imena glavnog Python programa na vašem računaru: trebalo bi da to bude `python`, `python3`, ili nešto slično. Za ostatak knjige, pretpostavićemo da je to `python`; ukoliko vaš ima drugačije ime, unesite njega gde god vidite u primeru koda da piše `python`.

Interaktivni prevodilac radi skoro isto kao što Python radi sa fajlovima, sa jednim izuzetkom: kada unesete nešto što ima vrednost, interaktivni prevodilac će vam njegovu vrednost ispisati automatski. Na primer, ukoliko pokrenete Python i unesete broj `61` u prevodilac, on će se kao eho pojaviti u vašoj komandnoj liniji.

U primeru koji sledi, `$` vam je primer *poruke* sistema da unesete komandu kao što je `python` u komandnu liniju. Koristićemo ovaj znak za primere koda u knjizi, iako će možda vaša poruka biti drugačija.

```
$ python
Python 3.3.0 (v3.3.0:bd8afb90ebf2, Sep 29 2012, 01:25:11)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 61
61
>>>
```

Ovakvo automatsko ispisivanje vrednosti je mogućnost koja prevodiocu štedi na vremenu, što nije deo Python jezika.

Usput, `print()` takođe radi u prevodiocu kada želite da nešto ispišete:

```
>>> print(61)
61
>>>
```

Ukoliko ste ove primere isprobali sa interaktivnim prevodiocem i dobili iste rezultate, onda ste pokrenuli jedan pravi (ali mali) Python kod. U sledećih nekoliko poglavlja, postepeno ćete napredovati od programa od jedne linije do dužih Python programa.

Korišćenje Python fajlova

Ukoliko ste u fajl uneli samo `61` preko Python-a, ono će raditi, ali neće ništa ispisivati. U normalnim neinteraktivnim Python programima, moraćete da pozovete `print` funkciju kako biste ispisali stvari, kao što je prikazano u sledećem kodu:

```
print(61)
```

Hajde da napravimo Python programski fajl i da ga pokrenemo:

1. Otvorite svoj editor teksta.
2. Ukucajte liniju `print (61)`, kao što je prikazano gore.
3. Sačuvajte ovo u fajl pod imenom `61.py`. obavezno ga sačuvajte kao prosti tekst, umesto u nekom „obogaćenom" formatu kao što je RTF ili Word. Ne morate koristiti sufiks `.py` za svoje Python programske fajlove, ali vam to pomaže da zapamtite gde se oni nalaze.
4. Ukoliko koristite grafički korisnički interfejs – oni su skoro svugde – otvorite komandnu liniju[1].
5. Pokrenite program unošenjem sledeće komande:

   ```
   $ python 61.py
   ```

Trebalo bi da vidite jednu liniju kao izlaz:

```
61
```

[1] Ukoliko nista sigurni šta ovo znači, pogledajte Dodadak D za detalje o različitim operativnim sistemima.

Da li vam je uspelo? Ukoliko jeste, čestitam vam na pokretanju vašeg prvog samostalnog Python programa.

Šta je sledeće?

Unosićete komande u konkretni Python sistem, i sve što treba da radite jeste da pratite ispravnu Python sintaksu. Umesto da vas preplavim svim sintaksičkim pravilima odjednom, prolazićemo lagano kroz njih u toku sledećih nekoliko poglavlja.

Osnovni način da napravite Python programe jeste da koristite najobičniji editor teksta i komandnu liniju. U ovoj knjizi prikazivaću ponekad sesije sa interaktivnim prevodiocem, a ponekad i delove Python fajlova. Trebalo bi da znate da postoji mnogo dobrih *integrisanih razvojnih okruženja* (engl. *integrated development environment, IDE*) za Python. U njima će možda biti istaknut grafički korisnički interfejs sa naprednijim editorom za tekst i pomoć. Više detalja o nekim od njih možete naučiti u poglavlju 12.

Vaš trenutak Zena

Svaki računarski jezik ima sopstveni stil. U predgovoru, pomenuo sam da često postoji *pajtonski* način da se izrazite. U Python je ugrađena i nekolicina slobodnih stihova koji jezgrovito izražavaju Python filozofiju (koliko ja znam, Python je jedini jezik koji podrazumeva i takav ugrađen dodatak). Samo ukucajte `import this` u svom interaktivnom prevodiocu i pritisnite taster Enter kad god vam je neophodan trenutak Zena:

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one--and preferably only one--obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea--let's do more of those!
```

Izražavaću primere ovakvih emocija kroz knjigu.

Stvari koje treba učiniti

Ovo poglavlje je bilo uvod u Python jezik – šta on radi, kako izgleda, i gde se on uklapa u računarski svet. Na kraju svakog poglavlja ću preporučiti neke mini-projekte koji će vam pomoći da zapamtite ono što ste upravo pročitali i pripremiti vas za ono što sledi.

1.1 Ukoliko već nemate Python 3 instaliran na računaru, instalirajte ga sada. Pročitajte dodatak D za više detalja o vašem sistemu računara.

1.2 Pokrenite Python 3 interaktivnog prevodioca. Ponovo, detalji se nalaze u dodatku D. Trebalo bi da ispiše nekoliko redova o samom sebi a zatim i jedan red koji počinje sa `>>>`. Ovo je poruka za vas da ukucate Python komande.

1.3 Igrajte se malo sa prevodiocem. Upotrebite ga kao kalkulator i ukucajte ovo: `8 * 9`. pritisnite taster Enter i vidite rezultat. Python bi trebalo da ispiše `72`.

1.4 Ukucajte broj `47` i pritisnite taster Enter. Da li vam je u sledećem redu ispisao `47`?

1.5 Sada ukucajte `print(47)` i pritisnite Enter. Da li je vam je opet u sledećem redu ispisao `47`?