Sadržaj

Predgovor ..... **xi**

Deo I Uvod u Python

1. Zašto Python za Excel? **3**
Excel je programski jezik 4
Excel u vestima 5
Najbolje prakse programiranja 6
Savremen Excel 11
Python za Excel 12
Čitljivost i održavanje 12
Standardna biblioteka i menadžer paketa 13
Naučno izračunavanje 15
Karakteristike savremenih jezika 16
Međuplatformska kompatibilnost 16
Zaključak 17

2. Razvojno okruženje **19**
Anaconda Python distribucija 19
Instalacija 20
Anaconda Prompt 21
Python REPL: Interaktivna Python sesija 24
Menadžeri paketa: Conda i pip 25
Conda okruženje 26
Jupyter beležnice 27
Pokretanje Jupyter beležnica 27
Ćelije beležnice 29
Edit vs. režim Command 30
Redosled izvršavanja je bitan 31
Isključivanje Jupyter beležnica 32
Visual Studio Code 33
Instalacija i konfiguracija 34
Pokretanje Python skripte 36
Zaključak 40

3. Početak rada sa Pythonom 41
Tipovi podataka 41
Objekti 42
Numerički tipovi 43
Logičke vrednosti 45
Znakovni nizovi (stringovi) 47
Indeksiranje i isecanje 48
Indeksiranje 48
Isecanje 49
Strukture podataka 50
Liste 50
Rečnici 52
Torke 54
Skupovi 54
Kontrola toka 56
Blokovi koda i izjava o prolasku 56
If naredba i uslovni izrazi 56
For i while petlje 57
Lista, rečnik i skup obuhvatanja 61
Organizacija koda 62
Funkcije 62
Moduli i naredba import 63
Klasa datetime 66
PEP 8: Vodič stila za Python kôd 67
PEP 8 i VS Code 69
Saveti za tip 69
Zaključak 70

Deo II Uvod u pandas

4. NumPy osnove 73
Početak rada sa NumPyem 73
NumPy niz 73
Vektorizacija i emitovanje 75
Univerzalne funkcije (ufunc) 76
Pravljenje i manipulacija nizova 77
Dobijanje i postavljanje elemenata niza 77
Korisni konstruktori niza 78
Prikaz u odnosu na kopiju 79
Zaključak 79

5. Analiza podataka pomoću pandas biblioteke 81
DataFrame i Series 81
Indeks 83
Kolone 86
Manipulacija podacima 87
Izbor podataka 87
Zadavanje podataka 93
Nedostajući podaci 96
Duplirani podaci 97
Aritmetičke operacije 98
Rad sa tekstualnim kolonama 100
Primena funkcije 101
View vs. Copy 102
Kombinovanje DataFrameova 103
Nadovezivanje 103
Join i Merge (spajanje i sjedinjavanje) 104
Opisna statistika i agregacija podataka 106
Opisna statistika 107
Grupisanje 107
Pivot i melt (zakretanje i melt) 108
Plot 110
Matplotlib 110
Plotly 112
Uvoz i izvoz tabela DataFrame 114
Izvoz CSV datoteka 115
Uvoz CSV datoteka 116
Zaključak 118

6. Analiza vremenskih nizova pomoću pandasa.....119
DatetimeIndex 120
Pravljenje DatetimeIndexa 120
Filtriranje indeksa datuma i vremena 122
Rad sa vremenskim zonama 123
Uobičajene manipulacije vremenskim nizovima 125
Pomeranja i procentualne promene 125
Promena baze i korelacija 127
Ponovno uzorkovanje 130
Pokretni prozor 131
Ograničenja pandasa 132
Zaključak 133

Deo III Čitanje i pisanje Excel datoteka bez Excela

7. Manipulacija Excelovim datotekama pomoću pandasa 137

Studija slučaja: Excel izveštavanje 137
Čitanje i pisanje Excel datoteka sa pandasom 141
Funkcija read_excel i klasa ExcelFile 141
Metoda to_excel i klasa ExcelWriter 146
Ograničenja pri korišćenju pandasa sa Excel datotekama 148
Zaključak 148

8. Manipulacija Excel datotekama pomoću paketa za čitanje i pisanje 149

Paketi čitača i pisača 149
Kada koristiti koji paket 150
Modul excel.py 151
OpenPyXL 153
XlsxWriter 157
pyxlsb 159
xlrd, xlwt i xlutils 160
Napredne teme za čitanje i pisanje 163
Rad sa velikim Excel datotekama 163
Formatiranje DataFrameova u Excelu 167
Studija slučaja (revidirana): Excel izveštavanje 172
Zaključak 173

Deo IV Programiranje Excel aplikacija sa xlwings

9. Excel automatizacija..... 177

Prvi koraci sa xlwings 178
Korišćenje programa Excel kao pregledača podataka 178
Excel objektni model 179
Pokretanje VBA koda 187
Konvertori, opcije i kolekcije 188
Rad sa DataFrameovima 188
Konvertori i opcije 189
Tabele, slike i definisana imena 191
Studija slučaja (ponovo revidirano): Excel izveštavanje 195
Napredne teme xlwingsa 197
xlwings osnove 197
Poboljšanje performansi 198
Kako zaobići nedostajuću funkcionalnost 200
Zaključak 201

10. Payton Excel alati 203
Korišćenje programa Excel kao interfejsa sa xlwings 203
Dodatak za Excel 203
Komanda za brzi početak 205
Run main 206
Funkcija RunPython 207
Implementacija 211
Python zavisnost 211
Samostalne radne sveske: oslobađanje od dodatka xlwings 212
Hijerarhija konfiguracije 213
Podešavanja 214
Zaključak 215

11. Pythonov Package Tracker 217
Šta ćemo napraviti 217
Unutrašnja funkcionalnost 219
Veb API-ji 219
Baze podataka 223
Izuzeci 231
Struktura aplikacije 234
Prednji plan 235
Pozadina 238
Otklanjanje grešaka 242
Zaključak 243

12. Korisnički definisane funkcije (UDF) 245
Početak rada sa UDF-ovima 245
UDF brzi početak 246
Studija slučaja: Google trendovi 251
Uvod u Google trendove 251
Rad sa DataFrameovima i dinamičkim nizovima 252
Preuzimanje podataka iz Google trendova 257
Iscrtavanje pomoću UDF-ova 261
Otklanjanje grešaka u UDF-ovima 262
Napredne UDF teme 264
Osnovna optimizacija performansi 264
Keširanje 267
Sub dekorator 269
Zaključak 270

A Conda okruženje 273

B Napredna funkcionalnost VS Codea 277

C Napredni koncepti Pythona 283

Indeks 291

Predgovor

Microsoft vodi forum za povratne informacije o Excelu na UserVoiceu (*https://oreil.ly/y1XwU*) gde svako može da pošalje novu ideju o kojoj će drugi glasati. Zahtev za funkcije sa najvećim brojem glasova je „Python as an Excel scripting language" (Python kao skript jezik za Excel) i ima otprilike dvostruko više glasova od drugog kandidata za najviše glasova. Iako se ništa nije dogodilo od kada je ideja dodata 2015. godine, korisnici programa Excel dobili su novu nadu krajem 2020. godine kada je Guido van Rossum, tvorac Pythona, poslao tvit (*https://oreil.ly/ N1_7N)*) da mu je „penzionisanje bilo dosadno" i da će se pridružiti Microsoftu. Ne znam da li njegov potez ima uticaja na integraciju programa Excel i Python. Znam, međutim, šta ovu kombinaciju čini tako ubedljivom i kako možete početi da koristite Excel i Python zajedno – danas. Ovo je ukratko ono o čemu ova knjiga govori.

Glavna pokretačka snaga priče o *Pythonu za Excel* je činjenica da živimo u svetu podataka. Danas su ogromni skupovi podataka dostupni svima i o svemu. Često su ti skupovi podataka toliko veliki da više ne staju u tabelu. Pre nekoliko godina ovo se možda nazivalo *big data* (veliki podaci), ali u današnje vreme skup podataka od nekoliko miliona redova zaista nije ništa posebno. Excel je evoluirao da bi se nosio sa tim trendom: uveo je Power Query za učitavanje i čišćenje skupova podataka koji se ne uklapaju u tabelu i Power Pivot, dodatak za analizu podataka na ovim skupovima podataka i predstavljanje rezultata. Power Query je zasnovan na Power Query M jeziku formule (M), dok Power Pivot definiše formule pomoću izraza za analizu podataka (Data Analysis Expressions, DAX). Ako želite da automatizujete nekoliko stvari u Excel datoteci, onda biste koristili Excelov ugrađeni jezik za automatizaciju, Visual Basic for Applications (VBA). To jest, za nešto prilično jednostavno, na kraju možete koristiti VBA, M i DAX. Jedan problem sa ovim je što svi ovi jezici rade samo u Microsoftovom svetu, najistaknutiji u Excelu i Power BI-u (Power BI ću ukratko predstaviti u poglavlju 1).

Python, je programski jezik opšte namene koji je postao jedan od najpopularnijih izbora među analitičarima i stručnjacima za podatke. Ako koristite Python sa programom Excel, moći ćete da koristite programski jezik koji je dobar u svim aspektima priče, bilo da se radi o automatizaciji programa Excel, pristupu i pripremi skupova podataka ili obavljanju zadataka analize i vizualizacije podataka. Ono što je najvažnije, možete koristiti svoje veštine rada sa Pythonom izvan programa Excel: ako treba da povećate svoju računarsku moć, lako biste mogli da premestite svoj kvantitativni model, simulaciju ili aplikaciju za mašinsko učenje u oblak, gde vas čekaju praktično neograničeni računarski resursi.

Zašto sam napisao ovu knjigu

Tokom svog rada na xlwingsu, Excelovom paketu za automatizaciju koji ćemo sresti u delu IV ove knjige, u bliskom sam kontaktu sa mnogim korisnicima koji koriste Python za Excel – bilo da je to putem programa za praćenje problema (*https://oreil.ly/ZJQkB)*) na GitHubu, pitanje na StackOverflowu (*https://stackoverflow.com)*) ili na fizičkom događaju poput sastanka ili konferencije.

Od mene se redovno traži da preporučim resurse za početak rada sa Pythonom. Iako ne nedostaje Python uvoda, oni su često ili previše opšti (ništa o analizi podataka) ili previše specifični (potpuni naučni uvod). Međutim, korisnici programa Excel imaju tendenciju da budu negde u sredini: oni rade sa podacima, ali potpuni naučni uvod može biti previše tehnički. Oni često imaju posebne zahteve i pitanja na koja nema odgovora ni u jednom postojećem materijalu. Neka od ovih pitanja su:

- Koji paket Python-Excel mi je potreban za koji zadatak?
- Kako da premestim svoju Power Query vezu sa bazom podataka na Python?
- Šta je ekvivalent Excelovog AutoFiltera ili zaokretne tabele u Pythonu?

Napisao sam ovu knjigu za vas koji nemate dovoljno znanja o Pythonu da biste mogli automatizovati svoje Excel zadatke i iskoristiti Pythonove analize podataka i alate za naučno računarstvo u Excelu i to bez stranputica.

Kome je namenjena ova knjiga

Ako ste napredni korisnik programa Excel koji želi da prevaziđe granice programa Excel savremenim programskim jezikom, ova knjiga je za vas. Najčešće to znači da svakog meseca provodite sate preuzimajući, čisteći i kopirajući/umećući velike količine podataka u kritične tabele. Iako postoje različiti načini za prevazilaženje ograničenja programa Excel, ova knjiga će se fokusirati na to kako koristiti Python za ovaj zadatak.

Trebalo bi da imate osnovno razumevanje programiranja: pomaže ako ste već napisali funkciju ili for petlju (bez obzira u kom programskom jeziku) i imate ideju o tome šta je ceo broj ili niz znakova. Možda ćete čak moći da savladate ovu knjigu ako ste navikli da pišete složene ćelijske formule ili imate iskustva sa podešavanjem snimljenih VBA makroa. Ne očekuje se da imati iskustvo vezano za Python, budući da postoje uvodi u sve alate koje ćemo koristiti, uključujući i uvod u sam Python.

Ako ste iskusni VBA programer, naći ćete redovna poređenja između Pythona i VBA koja će vam omogućiti da rešavate uobičajene probleme.

Ova knjiga takođe može biti od pomoći ako ste programer na Pythonu i morate da naučite o različitim načinima na koje Python može da se nosi sa Excel aplikacijom i Excel datotekama da bi mogao da izabere pravi paket s obzirom na zahteve vaših poslovnih korisnika.

Kako je ova knjiga organizovana

U ovoj knjizi ću vam pokazati sve aspekte priče *Python za Excel* podeljeno na četiri dela:

Deo I: Uvod u Python

Ovaj deo počinje razmatranjem razloga zašto je Python tako prijatan saputnik za Excel pre nego što predstavimo alate koje ćemo koristiti u ovoj knjizi: distribuciju Anaconda Python, Visual Studio Code i Jupyter beležnice. Ovaj deo će vas naučiti dovoljno i Pythona da savladate ostatak ove knjige.

Deo II: Uvod u pandas

pandas je Pythonova biblioteka za analizu podataka. Naučićemo kako da zamenite Excel radne sveske kombinacijom Jupyter beležnica i pandasa. Obično je pandas kôd lakši za održavanje i efikasniji od Excel radne sveske, a možete raditi i sa skupovima podataka koji ne staju u tabelu. Za razliku od Excela, pandas vam omogućava da pokrenete svoj kôd gde god želite, uključujući oblak.

Deo III: Čitanje i pisanje Excel datoteka bez Excela

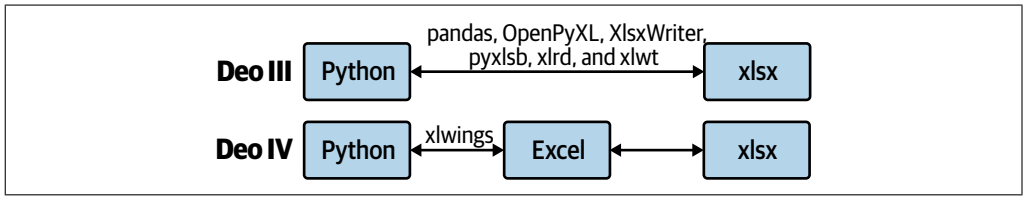
Ovaj deo se bavi manipulacijom Excel datotekama pomoću jednog od sledećih Python paketa: pandas, OpenPyXL, XlsxWriter, pyxlsb, xlrd i xlwt. Ovi paketi mogu da čitaju i pišu Excel radne sveske direktno na disku i kao takvi zamenjuju Excel aplikaciju: pošto vam nije potrebna instalacija programa Excel, oni rade na bilo kojoj platformi koju Python podržava, uključujući Windows, macOS i Linux. Tipičan slučaj upotrebe paketa za čitanje je čitanje podataka iz Excel datoteka koje svakog jutra primate od spoljne kompanije ili sistema i skladištenje njihovog sadržaja u bazu podataka. Tipičan slučaj upotrebe paketa za pisanje je obezbeđivanje funkcionalnosti koje se nalazi iza čuvenog dugmeta „Export to Excel" (Izvezi u Excel) koje možete pronaći u gotovo svakoj aplikaciji.

Deo IV: Programiranje Excel aplikacije sa xlwings

U ovom delu ćemo videti kako možemo koristiti Python sa paketom xlwings za automatizaciju Excel aplikacije umesto čitanja i pisanja Excel datoteka na disk. Zbog toga ovaj deo zahteva da imate lokalnu instalaciju programa Excel. Naučićemo kako da otvorimo Excel radne sveske i da njima rukujemo dok su ispred nas. Osim čitanja i pisanja datoteka putem programa Excel, izgradićemo i interaktivne Excel alate: oni nam omogućavaju da pritisnemo dugme da bi Python obavio nešto što ste možda prethodno radili sa VBA makroima, poput računarski zahtevnih izračunavanja. Takođe ćemo naučiti kako pisati korisnički definisane funkcije[1] (UDF) u Pythonu umesto VBA.

Važno je razumeti fundamentalnu razliku između čitanja i pisanja Excel *datoteka* (deo III) i programiranje Excel *aplikacije* (deo IV) kao što je prikazano na slici P-1.

[1] Microsoft je počeo da koristi izraz *prilagođene funkcije* (eng. *custom functions*) umesto UDF. U ovoj knjizi ću ih i dalje nazivati UDF.



Slika P-1. Čitanje i pisanje Excel datoteka (deo III) u odnosu na programiranje Excela (deo IV)

Pošto deo III ne zahteva instalaciju programa Excel, sve radi na svim platformama koje Python podržava: Windows, macOS i Linux, međutim, radiće samo na onim platformama koje podržava Microsoft Excel, tj. Windows i macOS, pošto se kôd oslanja na lokalnu instalaciju Microsoft Excela.

Python i Excel verzije

Ova knjiga zasnovana je na Pythonu 3.8, verziji Pythona koja dolazi sa najnovijom verzijom distribucije Anaconda Python u vreme pisanja ovog teksta. Ako želite da koristite noviju verziju Pythona, sledite uputstva na veb stranici ove knjige (*https://xlwings.org/book*), ali pazite da ne koristite stariju verziju. Povremeno ću ostaviti komentar ako se nešto promeni sa Pythonom 3.9.

Ova knjiga takođe očekuje da koristite savremenu verziju programa Excel, što znači najmanje Excel 2007 u sistemu Windows i Excel 2016 u sistemu macOS. Lokalno instalirana verzija programa Excel koja dolazi sa pretplatom na Microsoft 365 takođe će savršeno funkcionisati – u stvari, čak je i preporučujem, jer ima najnovije funkcije koje nećete pronaći u drugim verzijama programa Excel. To je bila i verzija koju sam koristio za pisanje ove knjige, pa ako koristite drugu verziju programa Excel, ponekad ćete možda videti malu razliku u nazivu ili lokaciji stavke menija.

Konvencije korišćene u ovoj knjizi

U ovoj knjizi se koriste sledeće tipografske konvencije:

Italic
: Označava nove termine, URL-ove, adrese e-pošte, nazive datoteka i oznake tipa datoteka.

`Konstantna širina`
: Koristi se za listing programa, kao i u pasusima za upućivanje na programske elemente kao što su imena promenljivih ili funkcija, baze podataka, tipovi podataka, promenljive okruženja, izrazi i ključne reči.

`Konstantna širina podebljano`
: Prikazuje komande ili drugi tekst koji korisnik treba doslovno da otkuca.

`Kurziv konstantne širine`
: Prikazuje tekst koji treba zameniti vrednostima koje je uneo korisnik ili vrednostima određenim kontekstom.



Ovaj element označava savet ili sugestiju.



Ovaj element označava opštu napomenu.



Ovaj element ukazuje na upozorenje ili oprez.

Korišćenje primera koda

Održavam veb stranicu (*https://xlwings.org/book*) sa dodatnim informacijama koje će vam pomoći oko ove knjige. Posetite je, posebno ako naiđete na problem.

Dodatni materijal (primeri koda, vežbe itd.) dostupan je za preuzimanje na *https://github.com/fzumstein/python-for-excel*. Da biste preuzeli ovo prateće spremište, kliknite na zeleno dugme Code, a zatim odaberite Download ZIP. Kada je preuzeto, kliknite desnim tasterom miša na datoteku u Windowsu i izaberite Extract All da biste raspakovali sadržaj datoteke u folder. Na macOS-u jednostavno dvaput kliknite na datoteku da biste je raspakovali. Ako znate kako da radite sa Gitom, takođe biste mogli da koristite Git za kloniranje spremišta na vaš lokalni čvrsti disk. Folder možete staviti gde želite, ali ja ću ga povremeno pominjati na sledeći način u ovoj knjizi:

```
C:\Users\imekorisnika\python-for-excel
```

Jednostavnim preuzimanjem i raspakovanjem ZIP datoteke u Windowsu dobićete strukturu foldera sličnu ovoj (obratite pažnju na ponavljajuća imena foldera):

```
C:\...\Downloads\python-for-excel-1st-edition\python-for-excel-1st-edition
```

Kopiranje sadržaja ovog foldera u jedan koju pravite pod *C:\Users\<imekorisnika>\python-for-excel* moglo bi vam olakšati praćenje. Iste napomene važe i za macOS, tj. kopirajte datoteke u */Users/<imekorisnika>/python-for-excel*.

Ako imate tehničko pitanje ili problem sa primerima koda, pošaljite e-poruku na *bookkuestions@oreilli.com*.

Ova knjiga je tu da vam pomogne da obavite svoj posao. Uopšteno, ako se uz ovu knjigu nudi primer koda, možete ga koristiti u svojim programima i dokumentaciji. Ne morate da nas kontaktirate za dozvolu ako ne reprodukujete značajan deo koda. Na primer, za pisanje programa koji koristi nekoliko delova koda iz ove knjige nije potrebna dozvola. Za prodaju ili distribuciju primera iz ove knjiga potrebna je dozvola. Za odgovaranje na pitanje citiranjem ove knjige i citiranjem primera koda nije potrebna dozvola. Za uključivanje značajne količine primera koda iz ove knjige u dokumentaciju vašeg proizvoda potrebna je dozvola.

Cenimo, ali generalno ne zahtevamo, navođenje. Navođenje obično uključuje naslov, autora, izdavača i ISBN broj. Na primer: „*Python za Excel* Felix Zumstein (O'Reilly). Autorska prava 2021 Zoomer Analytics LLC, 978-1-492-08100-5. "

Ako smatrate da upotreba primera koda ne spada u poštenu upotrebu ili gore navedenu dozvolu, slobodno nas kontaktirajte na permissions@oreilli.com.

Mikro knjiga

Preko 37 godina Mikro knjiga donosi odabrane IT naslove i naslove opšteobrazovnih tema u ediciji *Za neupućene*... Sa preko 460 objavljenih naslova pomogla je u obrazovanju desetina hiljada IT stručnjaka. Za više informacija o našim izdanjima pogledaje na *www.mikroknjiga.rs*

Kako da nas kontaktirate

Komentare i pitanja u vezi ove knjige uputite izdavaču:

MIKRO KNJIGA, DOO
Kneza Višeslava 34
11030 Beograd, Srbija
+381 11 3058-044

e-mail: *redakcija@mikroknjiga.rs*

Facebook: *http://facebook.com/mikroknjiga*

Twitter: *http://twitter.com/mikroknjiga*

Na ovoj strnaici (*https://oreil.ly/py4excel*) naći ćete greške, primere i sve dodatne informacije o ovoj knjizi..

Veb prikaz ove knjige nalazi se na adresi
https://www.mikroknjiga.rs/store/prikaz.php?ref=PYEX

Sve vaše sugestije, primedbe i prijave eventualnih grešaka u knjizi su dobrodošle.

Zahvalnice

Kao autor po prvi put, veoma sam zahvalan na pomoći koju sam dobio od toliko ljudi na ovom putu – koji su mi znatno olakšali ovo putovanje!

U O'Reillyju želim da se zahvalim svojoj urednici Melissi Potter, koja je uradila sjajan posao održavajući me motivisanim i u tempu i koja mi je pomogla da ovu knjigu dovedem u čitljiv oblik. Takođe bih želeo da se zahvalim Michelle Smith, koja je sa mnom radila na početnoj verziji knjige, i Danielu Elfanbaumu, koji se nije umorio odgovorajući na moja tehnička pitanja.

Veliko hvala svim mojim kolegama, prijateljima i klijentima koji su uložili mnogo sati u čitanje najranijih oblika mojih nacrta. Njihove povratne informacije bile su ključne za lakše razumevanje knjige, a neke studije slučaja inspirisane su stvarnim problemima u Excelu koje su podelili sa mnom. Zahvaljujem se Adamu Rodrigezu, Manu Beeslaru, Simonu Schieggu, Rui Da Costi, Jurgu Nageru i Christopheu de Montrichardu.

Dobio sam korisne povratne informacije i od čitalaca rane verzije knjige koja je objavljena na O'Reilly platformi za učenje na mreži. Hvala vam Felipe Maion, Ray Doue, Kolyu Minevski, Scott Drummond, Volker Roth i David Ruggles!

Imao sam veliku sreću što su knjigu recenzirali visoko kvalifikovani recenzenti i zaista cenim njihov trud koji su uložili pod velikim vremenskim pritiskom. Hvala vam na vašoj pomoći, Jordan Goldmeier, George Mount, Andreas Clenow, Werner Brönnimann i Eric Moreira!

Posebna zahvalnost ide Björnu Stielu, koji nije bio samo tehnički recenzent, već je neko od koga sam naučio i mnoge stvari o kojima pišem u ovoj knjizi. Uživao sam radeći sa vama ovih nekoliko godina!

Poslednje, ali ne i najmanje važno, želeo bih da se zahvalim Ericu Reinoldsu, koji je 2016. spojio svoj ExcelPython projekat u bazu koda xlwings. Redizajnirao ceo paket od nule, čineći moj užasan API iz ranih dana stvar prošlosti. Mnogo vam hvala!

POGLAVLJE 1

Zašto Python za Excel?

Korisnici programa Excel obično počinju da preispituju svoje alate za obradu tabela kada naiđu na ograničenje. Klasičan primer je kada Excel radne sveske (eng. *workbooks*) sadrže toliko podataka i formula da postaju spore ili u najgorem slučaju, ruše se. Ipak, ima smisla preispitati svoje radno okruženje pre nego što stvari krenu naopako: ako radite sa radnim sveskama kritičnim za poslove u kojima greške mogu da rezultuju finansijskom ili reputacionom štetom ili ako provodite sate svaki dan ručno ažurirajući Excelove radne sveske, trebalo bi da naučite kako da automatizujte svoje procese pomoću programskog jezika. Automatizacija uklanja rizik od ljudske greške i omogućava vam da provedete vreme na produktivnijim zadacima od kopiranja/umetanja (eng. *copy/paste*) podataka u Excel tabelu.

U ovom poglavlju daću vam nekoliko razloga zašto je Python odličan izbor u kombinaciji sa Excelom i koje su njegove prednosti u poređenju sa Excelovim ugrađenim jezikom za automatizaciju, VBA. Nakon što predstavim Excel kao programski jezik i razumete njegove posebnosti, ukazaću na specifične karakteristike koje Python čine toliko jačim u poređenju sa VBA. Za početak, međutim, hajde da pogledamo poreklo naša dva programa!

U pogledu računarske tehnologije, Excel i Python su prisutni već duže vreme: Excel je prvi put lansirao Microsoft 1985. Godine – i to bi moglo biti iznenađenje za mnoge – bio je dostupan samo za Apple Macintosh. Tek 1987. Microsoft Windows dobija svoju prvu verziju u obliku Excel 2.0. Microsoft ipak nije bio prvi igrač na tržištu proračunskih tabela: VisiCorp je 1979. izašao sa VisiCalcom, a nakon njega Lotus Software 1983. sa Lotus 1-2-3. I Microsoft nije vodio sa Excelom: tri godine ranije objavili su Multiplan, program za tabelarne proračune koji se mogao koristiti na MS-DOS-u i nekoliko drugih operativnih sistema, ali ne i na Windowsu.

Python rođen je 1991. godine, samo šest godina nakon Excela. Iako je Excel rano postao popularan, Pythonu je trebalo malo duže dok nije usvojen u određenim oblastima kao što su veb razvoj ili administracija sistema. Godine 2005. Python je počeo da postaje ozbiljna alternativa u naučnom računarstvu kada je prvi put objavljen *NumPy*, paket za računanje i linearnu algebru zasnovano na nizovima (eng. *array-based*). NumPy je kombinovao dva prethodna paketa i stoga uskladio sve razvojne napore za naučno računanje

u jedan projekat. Danas čini osnovu bezbroj naučnih paketa, uključujući *pandas*, koji je izašao 2008. godine i koji je u velikoj meri zaslužan za široko usvajanje Pythona u svetu nauke o podacima i finansijama koje su počele da se dešavaju posle 2010. Zahvaljujući pandasu, Python je, zajedno sa R, postao jedan od najčešće korišćenih jezika za zadatke nauke o podacima, poput analize podataka, statistike i mašinskog učenja.

Činjenica da su i Python i Excel davno izmišljeni nije jedino što im je zajedničko: Excel i Python su takođe programski jezici. Iako verovatno niste iznenađeni što to čujete o Pythonu, možda će biti potrebno objašnjenje za Excel, koje ću vam dati u nastavku.

Excel je programski jezik

Ovaj odeljak počinje uvođenjem programa Excel kao programskog jezika, koji će vam pomoći da razumete zašto se problemi sa tabelama redovno pojavljuju u vestima. Zatim ćemo pogledati nekoliko najboljih postupaka koji su se pojavili u zajednici za razvoj softvera i koje vas mogu spasiti od mnogih tipičnih grešaka programa Excel. Završićemo kratkim uvodom u Power Query i Power Pivot, dva savremena Excel alata koji pokrivaju vrstu funkcionalnosti za koju ćemo umesto toga koristiti pandas.

Ako koristite Excel za više od svoje liste namirnica, definitivno koristite funkcije poput `=SUM(A1:A4)` da biste sumirali niz ćelija. Ako na trenutak razmislite kako ovo funkcioniše, primetićete da vrednost ćelije obično zavisi od jedne ili više drugih ćelija, koje mogu ponovo koristiti funkcije koje zavise od jedne ili više drugih ćelija itd. Upućivanje takvih poziva ugnežđenih funkcija se ne razlikuje od načina na koji funkcionišu drugi programski jezici, samo što kôd upisujete u ćelije umesto u tekstualne datoteke. A ako vas to još nije ubedilo: krajem 2020. Microsoft je najavio uvođenje *lambda funkcija*, koje omogućavaju da funkcije za višekratnu upotrebu pišete u Excelovom jeziku formule, tj. oslanjajući se na drugačiji jezik poput VBA. Prema rečima Brajana Jonesa, šefa Excelovog odeljenja, ovo je deo koji nedostaje i koji Excel konačno čini „pravim" programskim jezikom.[1] Ovo takođe znači da korisnici programa Excel zaista treba da se zovu Excel programeri!

Postoji, međutim, posebna stvar u vezi sa programerima programa Excel: većina njih su poslovni korisnici ili stručnjaci za domene bez formalnog obrazovanja iz računarstva. Oni su trgovci, računovođe ili inženjeri, da spomenemo samo nekoliko primera. Njihovi alati za tabele osmišljeni su da reše poslovni problem i često zanemaruju najbolje postupke u razvoju softvera. Kao posledica toga, ovi alati za tabelarne proračune često mešaju ulaze, proračune i izlaze na istim listovima, možda će biti potrebno da se izvrše neočigledni koraci da bi oni ispravno radili, a kritične promene se vrše bez ikakve sigurnosne mreže. Drugim rečima, alatkama za tabelarne proračune nedostaje čvrsta arhitektura aplikacija i često su nedokumentovane i neproverene. Ponekad sve ovo može imati

[1] Najavu lambda funkcija možete pročitati na Excel blogu (*https://oreil.ly/4-0y2*).

razorne posledice: ako zaboravite da osvežite proračune u radnoj svesci pre trgovine akcijama, možete kupiti ili prodati pogrešan broj akcija, što može dovesti do gubitka novca. A ako ne trgujete samo svojim novcem, o takvim slučajevima možemo čitati u vestima, kao što ćemo videti dalje.

Excel u vestima

Excel je redovan gost u vestima, a tokom ovog pisanja dve nove priče dospele su na naslovne strane. Prvi se odnosio na Odbor za nomenklaturu ljudskog genoma (HUGO Gene Nomenclature Committee), koji je preimenovao nekoliko ljudskih gena kako ih Excel više ne bi tumačio kao datume. Na primer, da bi se sprečilo da se gen `MARCH1` pretvori u `1-Mar`, preimenovan je u `MARCHF1`.[2] U drugoj priči, Excel je okrivljen za odloženo izveštavanje o 16.000 rezultata testa na COVID-19 u Engleskoj. Do problema je došlo zato što su rezultati testa upisani u stariji format Excelove datoteke (*.xls*) koji je bio ograničen na otprilike 65.000 redova. To je značilo da su veći skupovi podataka jednostavno odsečeni preko tog ograničenja.[3] Iako ove dve priče pokazuju stalnu važnost i dominaciju Excela u današnjem svetu, verovatno ne postoji drugi „Excel incident" koji je poznatiji od Londonskog kita.

London Whale je nadimak trgovca čije su greške u trgovanju primorale kompaniju JP Morgan da objavi zapanjujući gubitak od 6 milijardi dolara u 2012. Izvor eksplozije bio je model rizika zasnovan na Excelu koji je značajno potcenjivao pravi rizik gubitka novca u jednom od svojih portofolija. Izveštaj radne grupe za upravljanje JPMorgan Chase & Co. u vezi sa gubicima CIO za 2012. Godinu (*Report of JPMorgan Chase & Co. Management Task Force Regarding 2012 CIO Losses*)[4] (2013) pominje da je „model funkcionisao kroz niz Excel tabela, koje je trebalo popuniti ručno, procesom copy/paste podataka iz jedne tabele u drugu". Povrh ovih operativnih pitanja, postojala je i logička greška: u jednom proračunu delili su sa sumom umesto sa srednjom vrednošću.

Ako želite da vidite više ovih priča, pogledajte Horror Stories (Horor priče) veb stranicu *https://oreil.ly/WLO-I* koju održava Evropska interesna grupa za rizike u tabelama (EuSpRIG).

Da biste sprečili da vaša kompanija završi u vestima sa sličnom pričom, pogledajmo nekoliko najboljih postupaka koje vaš rad sa Excelom čine znatno sigurnijim.

---

[2] James Vincent, „Scientists rename human genes to stop Microsoft Excel from misreading them as dates," *The Verge*, 6. avgusta 2020. (*https://oreil.ly/0qo-n*).

[3] Leo Kelion, „Excel: Why using Microsoft's tool caused COVID-19 results to be lost", *BBC Nevs*, 5. oktobra 2020. (*https://oreil.ly/vvB6o*).

[4] Wikipedija se povezuje na dokument u jednoj od fusonata u svom članku o slučaju (*https://oreil.ly/0uUj9*).

Najbolje prakse programiranja

Ovaj odeljak će vas upoznati sa najvažnijim i najboljim praksama programiranja, uključujući razdvajanje nadležnosti, princip DRY, testiranje i kontrolu verzija. Kao što ćemo videti, njihovo praćenje će biti lakše kada zajedno počnete da koristite Python sa Excelom.

Odvajanje nadležnosti

Jedan od najvažnijih principa projektovanja u programiranju je *razdvajanje nadležnosti*, koje se ponekad naziva i *modularnost*. To znači da bi nezavisni deo programa trebalo da brine o povezanom skupu funkcionalnosti tako da se može lako zameniti bez uticaja na ostatak aplikacije. Na najvišem nivou, aplikacija se često deli na sledeće slojeve:[5]

- Sloj prezentacije
- Sloj poslova
- Sloj podataka

Da biste objasnili ove slojeve, razmislite o jednostavnom konvertoru valuta poput onog prikazanog na slici 1-1. Naći ćete Excel datoteku *currency_converter.xlsx* u folderu *xl*.

Ovako aplikacija radi: unesite iznos i valutu u ćelije A4 i B4, a Excel će to pretvoriti u američke dolare u ćeliji D4. Mnoge aplikacije za tabele slede takav dizajn i preduzeća ih svakodnevno koriste. Dozvolite mi da podelim aplikaciju na njene slojeve:

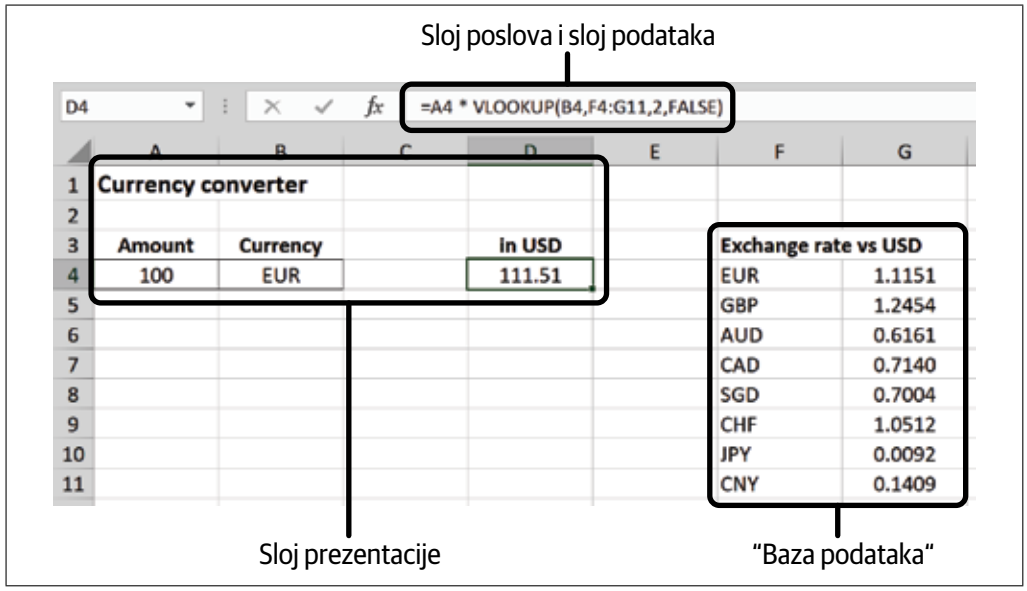
Sloj prezentacije
: Ovo je ono što vidite i sa njim komunicirate, tj. korisnički interfejs: vrednosti ćelija A4, B4 i D4 zajedno sa njihovim oznakama grade sloj prezentacije konvertora.

Sloj poslova
: Ovaj sloj vodi računa o logici specifičnoj za aplikaciju: ćelija D4 definiše kako se iznos pretvara u USD. Formula `= A4 * VLOOKUP (B4, F4:G11, 2, FALSE)` prevodi se u iznos puta kurs.

Sloj podataka
: Kao što naziv govori, ovaj sloj vodi računa o pristupu podacima: deo `VLOOKUP` ćelije D4 obavlja ovaj posao.

Sloj podataka pristupa podacima iz tabele deviznih kurseva koja počinje u ćeliji F3 i koja deluje kao baza podataka ove male aplikacije. Ako ste obratili pažnju, verovatno ste primetili da se ćelija D4 pojavljuje u sva tri sloja: ova jednostavna aplikacija meša u jednoj ćeliji sloj prezentacije, poslova i podataka.

[5] Terminologija je preuzeta iz *Microsoft Application Architecture Guide, 2nd Edition*, koji je dostupan na mreži: *https://oreil.ly/8V-GS*.



Slika 1-1. currency_converter.xlsx

Ovo nije nužno problem za ovaj jednostavan konvertor valuta, ali često se ono što počinje kao mala Excel datoteka ubrzo pretvara u mnogo veću aplikaciju. Kako se ova situacija može poboljšati? Većina profesionalnih resursa za programere Excela preporučuje vam da koristite zaseban list za svaki sloj, u Excelovoj terminologiji koja se obično naziva *ulazi*, *proračuni* i *rezultati*. Često se ovo kombinuje sa definisanjem određenog koda boje za svaki sloj, na primer, plava pozadina za sve ćelije za unos. U poglavlju 11 izgradićemo pravu aplikaciju zasnovanu na ovim slojevima: Excel će biti sloj prezentacije, dok će se sloj poslova i sloj podataka premestiti u Python, gde je mnogo lakše strukturisati kôd ispravno.

Sada kada znate šta znači razdvajanje nadležnosti, hajde da saznamo šta je DRY princip!

DRY princip

Knjiga *Pragmatični programer* od autora Hunta i Thomasa (Pearson Education, prevod Kompjuter bibiloteka, Beograd, 2019.) popularizovala je DRY princip: *ne ponavljajte se*. Bez dupliranog koda znači manje redova koda i manje grešaka, što olakšava održavanje koda. Ako se vaša poslovna logika nalazi u formulama ćelije, praktično je nemoguće primeniti DRY princip, jer ne postoji mehanizam koji vam omogućava da je ponovo koristite u drugoj radnoj svesci. Ovo, nažalost, znači da je uobičajen način za pokretanje novog Excel projekta kopiranje radne sveske iz prethodnog projekta ili iz šablona (eng. *tamplate*).

Ako pišete VBA, najčešći deo koda za višekratnu upotrebu je funkcija. Na primer, funkcija vam omogućava pristup istom bloku koda iz više makroa. Ako imate više funkcija koje stalno koristite, možda ćete želeti da ih delite između radnih svezaka. Standardni instrument za deljenje VBA koda kroz radne sveske su programski dodaci, ali dodacima VBA nedostaje robustan način za njihovu distribuciju i ažuriranje. Iako je Microsoft predstavio Excelovu internu prodavnicu dodatnih programa za Excel koji rešavaju taj problem, ovo funkcioniše samo sa dodacima zasnovanim na JavaScriptu, tako da to nije opcija za VBA programere. To znači da je i dalje veoma uobičajeno da se koristi pristup copy/paste sa VBA: pretpostavimo da vam je potrebna funkcija *kubni splajn*, složena kriva (eng. *spline*) u Excelu. Kubna splajn funkcija je način interpolacije krive zasnovane na nekoliko datih tačaka u koordinatnom sistemu i često je koriste trgovci sa fiksnim prihodom za izvođenje krive kamatne stope za sva dospeća na osnovu nekoliko poznatih kombinacija dospeća/kamatne stope. Ako tražite „Cubic Spline Excel" na internetu, ubrzo ćete dobiti stranicu VBA koda koja radi ono što želite. Problem sa ovim je što je najčešće ove funkcije pisala jedna osoba sa verovatno dobrim namerama, ali bez formalne dokumentacije ili testiranja. Možda radi za većinu ulaznih podataka, ali šta je sa nekim neuobičajenim graničnim slučajevima? Ako trgujete višemilionskim portofoliom sa fiksnim prihodom, želite da imate nešto u šta možete verovati. Tako ćete bar čuti od svojih internih revizora kada saznaju odakle potiče kôd.

Python olakšava distribuciju koda pomoću menadžera paketa, kao što ćemo videti u poslednjem odeljku ovog poglavlja. Pre nego što stignemo tamo, nastavimo sa testiranjem, jednim od kamena temeljaca razvoja solidnog softvera.

Testiranje

Kada kažete programeru Excela da testira njihove radne sveske, oni će najverovatnije izvršiti nekoliko slučajnih provera: pritisnuće na dugme i videće da li makro i dalje radi ono što bi trebalo da uradite ili će promeniti nekoliko ulaza i proverite da li rezultat izgleda razumno. Ovo je, međutim, rizična strategija: Excel olakšava uvođenje grešaka koje je teško uočiti. Na primer, možete prepisati formulu sa fiksnom kodiranom vrednošću. Ili ste zaboravili da prilagodite formulu u skrivenoj koloni.

Kada kažete profesionalnom programeru softvera da testira svoj kôd, on će napisati *jedinične testove*. Kao što ime govori, to je mehanizam za testiranje pojedinačnih komponenti vašeg programa. Na primer, jedinični testovi obezbeđuju da jedna funkcija programa radi ispravno. Većina programskih jezika nudi način za automatsko pokretanje jediničnih testova. Pokretanje automatizovanih testova dramatično će povećati pouzdanost vaše baze kodova i čini razumno sigirnim da nećete pokvarite ništa što trenutno funkcioniše kada uređujete kôd.

Ako pogledate alatku za konverziju valuta na slici 1-1, mogli biste napisati test koji proverava da li formula u ćeliji D4 ispravno vraća 105 USD sa sledećim ulazima: 100 EUR kao iznos i 1,05 kao kurs EURUSD. Zašto ovo pomaže? Pretpostavimo da ste slučajno

izbrisali ćeliju D4 sa formulom za konverziju i morate da je prepišete: umesto da iznos pomnožite sa kursom, delite ga sa njim – uostalom, rad sa valutama može biti zbunjujući. Kada pokrenete gornji test, dobićete neuspeh testa jer 100 EUR / 1,05 više neće rezultirati sa 105 USD kako test očekuje. Ovako, možete otkriti formulu i popraviti pre nego što tabelu predate korisnicima.

Skoro svi tradicionalni programski jezici nude jedan ili više radnih okvira za pisanje jediničnih testova bez mnogo napora – ali ne i Excel. Srećom, koncept jediničnih testova je dovoljno jednostavan i povezivanjem Excela sa Pythonom dobijate pristup moćnim Pythonovim okvirima za testiranje jedinica. Iako detaljnija prezentacija jediničnih testova izlazi iz okvira ove knjige, pozivam vas da pogledate moj blog post *https://oreil.ly/crwTm*, u kome ću vas provesti kroz praktične primere.

Jedinični testovi se često podešavaju tako da se automatski izvršavaju kada prosledite kôd sistemu kontrole verzija. Sledeći odeljak objašnjava šta su sistemi za kontrolu verzija i zašto ih je teško koristiti sa Excel datotekama.

Kontrola verzija

Još jedna karakteristika profesionalnih programera je da koriste sistem za *kontrolu verzija* ili *kontrolu izvora*. *sistem za kontrolu verzija* (VCS) prati promene u vašem izvornom kodu tokom vremena, omogućavajući vam da vidite ko je šta promenio, kada i zašto, i omogućava vam da se vratite na stare verzije u bilo kom trenutku. Najpopularniji sistem kontrole verzija u današnje vreme je Git: *https://git-scm.com*. Prvobitno je stvoren za upravljanje izvornim kodom Linuxa i od tada je osvojio svet programiranja – čak je i Microsoft usvojio Git 2017. za upravljanje izvornim kodom operativnog sistema Windows. U Excelovom svetu, naprotiv, daleko najpopularniji sistem kontrole verzija dolazi u obliku foldera u kom se datoteke arhiviraju na ovaj način:

```
currency_converter_v1.xlsx
currency_converter_v2_2020_04_21.xlsx
currency_converter_final_edits_Bob.xlsx
currency_converter_final_final.xlsx
```

Ako se, za razliku od ovog primera, programer Excela pridržava određene konvencije u imenovanju datoteka, u tome nema ništa loše. Međutim, čuvanje istorije verzija vaših datoteka lokalno vas isključuje iz važnih aspekata kontrole izvora u obliku lakše saradnje, pregleda saradnika, procesa odjave i evidencija revizije. A ako želite da vaše radne sveske budu sigurnije i stabilnije, ne želite da propustite ove stvari. Najčešće, profesionalni programeri koriste Git u vezi sa veb platformom kao što su GitHub, GitLab, Bitbucket ili Azure DevOps. Ove platforme vam omogućavaju da radite sa takozvanim *zahtevima za povlačenje* ili *zahtevima za spajanje*. Oni dozvoljavaju programerima da formalno zatraže da se njihove izmene spoje u glavnu bazu kodova. Zahtev za povlačenje nudi sledeće informacije:

- Ko je autor promena
- Kada su izvršene promene
- Koja je svrha promena opisanih u *poruci izvršenja*
- Koji su detalji promena prikazani u *diferencijalnom prikazu*, tj. prikazu koji ističe promene zelene boje za novi kod i crvene za izbrisan kod

Ovo omogućava saradniku ili šefu tima da pregledaju promene i uoče nepravilnosti. Često će dodatni par očiju moći da uoči grešku ili dve ili da na neki drugi način da programeru vrednu povratnu informaciju. Uz sve ove prednosti, zašto programeri Excela radije koriste lokalni sistem datoteka i sopstvenu konvenciju imenovanja umesto profesionalnog sistema poput Gita?

- Mnogi korisnici programa Excel jednostavno ne znaju za Git ili odustaju rano, jer Git ima relativno strmu krivu učenja.
- Git omogućava da više korisnika paralelno rade na lokalnim kopijama iste datoteke. Nakon što svi obave svoj posao, Git obično može spojiti sve promene bez ikakve ručne intervencije. Ovo ne funkcioniše za Excel datoteke: ako se paralelno menjaju na zasebnim kopijama, Git ne zna kako da spoji ove promene nazad u jednu datoteku.
- Čak i ako uspete da se nosite sa prethodnim problemima, Git jednostavno ne daje toliko vrednosti Excel datotekama kao što daje tekstualnim datotekama: Git ne može prikazati promene između Excel datoteka, sprečavajući pravilan proces pregledanja.

Zbog svih ovih problema, moja kompanija je osmislila xltrail *https://xltrail.com*, Git sistem kontrole verzija koji zna kako da se nosi sa Excel datotekama. Ovaj sistem sakriva kompleksnost Gita tako da se poslovni korisnici osećaju ugodno koristeći ga, a takođe vam omogućava i povezivanje sa spoljnim Git sistemima, u slučaju da već pratite svoje datoteke pomoću GitHuba, na primer. xltrail prati različite komponente radne sveske, uključujući formule ćelija, imenovane opsege, Power Queries i VBA kôd, omogućavajući vam da iskoristite prednosti klasičnih prednosti kontrole verzija, uključujući recenzije kolega.

Druga mogućnost da olakšate kontrolu verzija pomoću programa Excel je premeštanje poslovne logike iz programa Excel u datoteke Python, što ćemo učiniti u poglavlju 10. Kako se Python datoteke lako prate pomoću Gita, najvažniji deo vašeg alata za tabelarne proračune imaćete pod kontrolom.

Dok se ovaj odeljak naziva Najbolje prakse programiranja, uglavnom ukazuje na to zašto ih je teže slediti u Excelu nego u tradicionalnom programskom jeziku poput Pythona. Pre nego što skrenemo pažnju na Python, želeo bih ukratko predstaviti Power Query i Power Pivot, Microsoftov pokušaj modernizacije programa Excel.

Savremen Excel

Savremena era programa Excel započela je sa programom Excel 2007 kada su uvedeni trakasti meni i novi formati datoteka (npr. *xlsx* umesto *xls*). Međutim, Excel zajednica koristi *savremen Excel* za upućivanje na alate koji su dodati u programu Excel 2010: najvažnije Power Query i Power Pivot. Omogućavaju da se povežete sa spoljnim izvorima podataka i da analizirate podatke koji su preveliki da bi se ulili u tabelu. Kako se njihova funkcionalnost preklapa sa onim što ćemo raditi sa pandasom u poglavlju 5, ukratko ću ih predstaviti u prvom delu ovog odeljka. Drugi deo govori o Power BI-u, koji bi se mogao opisati kao samostalna aplikacija poslovne inteligencije koja kombinuje funkcionalnost Power Querya i Power Pivota sa mogućnostima vizualizacije – i ima ugrađenu podršku za Python!

Power Query i Power Pivot

Sa Excelom 2010, Microsoft je uveo dodatak pod nazivom *Power Query*. Power Query se povezuje sa raznim izvorima podataka, uključujući Excel radne sveske, CSV datoteke i SQL baze podataka. Takođe nudi veze sa platformama poput Salesforca i čak se može proširiti za povezivanje sa sistemima koji nisu podržani. Osnovna funkcionalnost programa Power Query bavi se skupovima podataka koji su preveliki da bi stali u tabelu. Nakon učitavanja podataka, možete izvršiti dodatne korake da biste ih očistili i manipulisali tako da stignu u upotrebljivom obliku u Excelu. Možete, na primer, podeliti kolonu na dve, spojiti dve tabele ili filtrirati i grupisati podatke. Od programa Excel 2016, Power Query više nije dodatak, ali mu se može pristupiti direktno na kartici Data na trakastom meniju pomoću dugmeta Get Data. Power Query je samo delimično dostupan na macOS-u – međutim, aktivno se razvija, pa bi trebalo da bude podržan u budućem izdanju programa Excel.

Power Pivot ide ruku pod ruku sa Power Queryjem: konceptualno, to je drugi korak nakon prikupljanja i čišćenja vaših podataka pomoću Power Queryja. Power Pivot vam pomaže da analizirate i predstavite svoje podatke na privlačan način direktno u programu Excel. Zamislite to kao tradicionalnu pivot tabelu koja se, poput Power Queryjaa, može nositi sa velikim skupovima podataka. Power Pivot vam omogućava da definišete formalne modele podataka sa odnosima i hijerarhijama, a možete da dodate izračunate kolone preko jezika DAX formula. Power Pivot je takođe uveden sa programom Excel 2010, ali ostaje i dalje kao dodatak i za sada nije dostupan na macOS-u.

Ako volite da radite sa Power Queryjem i Power Pivotom i želite da napravite kontrolne table (eng. *dashboards*) iznad njih, Power BI bi vredelo pogledati – hajde da vidimo zašto!

Power BI

Power BI je samostalna aplikacija koja je objavljena 2015. To je odgovor Microsofta na alate poslovne inteligencije poput Tableau i Qlik. Power BI Desktop je besplatan, pa ako želite da se poigrate s njim, idite na Power BI početnu stranicu *https://oreil.ly/I1kGj*

i preuzmite ga – imajte na umu da je Power BI Desktop dostupan samo za Windows. Power BI želi da da smisao velikim skupovima podataka vizualizujući ih na interaktivnim kontrolnim tablama. U osnovi se oslanja na istu Power Query i Power Pivot funkcionalnost kao Excel. Komercijalni planovi korišćenja vam omogućavaju da sarađujete i delite kontrolne table preko mreži, i odvojene su od verzije za računare. Glavni razlog zašto je Power BI uzbudljiv u kontekstu ove knjige je taj što podržava Python skripte od 2018. Python se može koristiti za deo upita kao i deo za vizualizaciju korišćenjem Python biblioteka za crtanje. Meni se čini da je korišćenje Pythona u Power BI-u pomalo nezgrapno, ali ovde je važan deo to što je Microsoft prepoznao značaj Pythona u pogledu analize podataka. Shodno tome, velike su nade da će jednog dana i Python pronaći službeni put u Excel.

Šta je toliko sjajno u Pythonu da je uspelo u Microsoftovom Power BI-u? Sledeći odeljak ima nekoliko odgovora!

Python za Excel

Excel služi za skladištenje, analizu i vizualizaciju podataka. A pošto je Python posebno jak u oblasti naučnog izračunavanja, prirodno se uklapa u kombinaciju sa Excelom. Python je takođe jedan od retkih jezika koji je privlačan i profesionalnim programerima i početnicima koji svakih nekoliko nedelja pišu nekoliko redova koda. S jedne strane, profesionalni programeri vole da rade sa Pythonom jer je to programski jezik opšte namene i stoga vam omogućava da postignete bilo šta bez vratolomija. Početnici, s druge strane, vole Python jer se lakše uči od drugih jezika. Kao posledica toga, Python se koristi i za ad hoc analizu podataka i manje zadatke automatizacije, kao i u ogromnim proizvodnim bazama kodova, poput pozadine Instagrama.[6] To takođe znači da kada vaš Excel alat sa Pythonom postane zaista popularan, lako je u projekat dodati veb programera koji će vaš protokol Excel-Python pretvoriti u punopravnu veb aplikaciju. Jedinstvena prednost Pythona je u tome što deo sa poslovnom logikom najverovatnije ne mora da se prepisuje, već se može premestiti u postojećem stanju iz protokola Excel u proizvodno veb okruženje.

U ovom odeljku predstaviću osnovne koncepte Pythona i uporediću ih sa Excelom i VBAom. Dotaknuću se čitljivosti koda, standardne Python biblioteke i menadžera paketa, naučnog računarskog steka, savremenih jezičkih funkcija i kompatibilnosti na više platformi. Hajdemo prvo u čitljivost!

Čitljivost i održavanje

Ako je vaš kôd čitljiv, to znači da ga je lako pratiti i razumeti – posebno za pridošlice koji sami nisu napisali kôd. Ovo olakšava uočavanje grešaka i održavanje koda u budućnosti.

[6] Možete saznati više o tome kako Instagram koristi Python na njihovom inženjerskom blogu: *https://oreil.ly/SSnQG*.

Zato je jedan red u *Zen of Python* „čitljivost se računa". Zen of Python je rezime Pythonovih osnovnih principa projektovanja, a kako ćemo ga odštampati naučićemo u sledećem poglavlju. Hajde da pogledamo sledeći isečak koda u VBA:

```
If i < 5 Then
    Debug.Print "i je manje od 5"
ElseIf i <= 10 Then
    Debug.Print "i je između 5 i 10"
Else
    Debug.Print "i je veće od 10"
End If
```

U VBA možete da preformatirate isečak u sledeće, što je potpuno ekvivalentno:

```
If i < 5 Then
    Debug.Print "i je manje od 5"
    ElseIf i <= 10 Then
    Debug.Print "i je između 5 i 10"
    Else
    Debug.Print "i je veće od 10"
End If
```

U prvoj verziji, vizuelno uvlačenje usklađuje se sa logikom koda. Ovo olakšava čitanje i razumevanje koda, što opet olakšava uočavanje grešaka. U drugoj verziji, programer koji je nov u kodu možda neće videti stanje `ElseIf` i `Else` pri prvom pregledu – ovo je očigledno još tačnije ako je kôd deo veće baze kodova.

Python ne prihvata kod koji je formatiran kao drugi primer: primorava vas da vizuelno uvlačenje poravnate sa logikom koda, sprečavajući probleme čitljivosti. Python to može učiniti jer se oslanja na uvlačenje za definisanje blokova koda dok ih koristite u `if` izrazima ili `for` petljama. Umesto uvlačenja, većina drugih jezika koristi vitičaste zagrade, a VBA koristi rezervisane reči kao što je `End If`, kao što smo upravo videli u isečcima koda. Razlog za korišćenje uvlačenja za blokove koda je taj što se u programiranju većina vremena troši na održavanje koda umesto na njegovo pisanje. Čitanje koda pomaže novim programerima (ili vama nekoliko meseci nakon pisanja koda) da se vrate i shvate šta se dešava.

Naučićemo sve o pravilima uvlačenja Pythona u poglavlju 3, ali za sada pređimo na standardnu biblioteku: funkcionalnost koja dolazi sa Pythonom.

Standardna biblioteka i menadžer paketa

Python dolazi sa bogatim skupom ugrađenih funkcija koje pruža njegova *standardna biblioteka*. Python zajednica voli da se poziva na to govoreći da Python dolazi sa „baterijama". Bez obzira da li treba da dekomprimujete ZIP datoteku, pročitate vrednosti CSV datoteke ili želite da preuzmete podatke sa interneta, standardna Python biblioteka vam to omogućava, a sve to možete postići obično samo u nekoliko redova koda. Ista

funkcionalnost u VBA bi zahtevala da napišete znatnu količinu koda ili instalirate dodatak. Često rešenja koja pronađete na internetu funkcionišu samo u operativnom sistemu Windows, ali ne i u sistemu macOS.

Dok standardna Python biblioteka pokriva impresivnu količinu funkcionalnosti, još uvek postoje zadaci koji su nezgrapni za programiranje ili spori kada se oslanjate samo na standardnu biblioteku. Ovde dolazi PyPI *https://pypi.org*. PyPI označava *Indeks Pythonovih paketa* (Python Package Index) i predstavlja ogromno spremište u koje svi (uključujući i vas!) mogu pridodati Python pakete otvorenog koda koji pružaju dodatnu funkcionalnost Pythonu.



PyPI vs. PyPy

PyPI se izgovara „paj pi aj". Ovo služi za razlikovanje PyPI od PyPy koji se izgovara „paj paj" i koji je brza alternativna implementacija Pythona.

Na primer, da biste olakšali preuzimanje podataka iz izvora na Internetu, mogli biste da instalirate paket Requests (Zahtevi) da biste dobili pristup skupu naredbi koje su moćne, ali jednostavne za upotrebu. Da biste ga instalirali, koristili biste Pythonov menadžer paketa *pip*, koji pokrećete na komandnoj liniji ili terminalu. pip je rekurzivna skraćenica za *pip installs packages* (pip instalira pakete). Ne brinite ako ovo sada zvuči pomalo apstraktno. Objasniću kako ovo funkcioniše detaljno u sledećem poglavlju. Za sada je važnije shvatiti zašto su menadžeri paketa toliko važni. Jedan od glavnih razloga je to što svaki razumni paket neće zavisiti samo od standardne biblioteke Pythona, već opet od drugih paketa otvorenog koda koji se takođe nalaze na PyPI-u. Ove zavisnosti mogu ponovo zavisiti od podzavisnosti i tako dalje. pip rekurzivno proverava zavisnosti i podzavisnosti paketa i preuzima ih i instalira. pip takođe olakšava ažuriranje vaših paketa kako biste mogli da ažurirate svoje zavisnosti. Ovo uveliko olakšava pridržavanje principa DRY jer ne morate ponovo da izmišljate ili radite copy/paste za ono što je već dostupno na PyPI-u. Sa pipom i PyPI-om takođe imate solidan mehanizam za distribuciju i instaliranje ovih zavisnosti, što nedostaje Excelu sa svojim tradicionalnim dodacima.

Softver otvorenog koda (OSS)

Na ovom mestu želim da kažem nekoliko reči o *otvorenom kodu*, jer sam tu reč koristio nekoliko puta u ovom odeljku. Ako se softver distribuira pod licencom otvorenog koda, to znači da je njegov izvorni kod besplatno dostupan bez ikakvih troškova, omogućavajući svima da doprinesu novim funkcijama, ispravkama grešaka ili dokumentacijom. Sam Python i skoro svi Python paketi nezavisnih proizvođača otvorenog su koda i programeri ih najčešće održavaju u slobodno vreme. Ovo nije uvek idealno stanje: ako se vaša kompanija oslanja na određene pakete, imate interes za stalni razvoj i održavanje ovih paketa od strane profesionalnih programera. Srećom, naučna Python zajednica je prepoznala da su neki paketi previše važni da bi sudbinu prepustili rukama nekoliko volontera koji rade uveče i vikendom.

Zato je 2012. godine NumFOCUS *https://numfocus.org*, neprofitna organizacija, stvorena za sponzorisanje različitih Python paketa i projekata u oblasti naučnih izračunavanja. Najpopularniji Python paketi koje sponzoriše NumFOCUS su pandas, NumPy, SciPy, Matplotlib i Project Jupyter, ali danas podržavaju i pakete sa drugih jezika, uključujući R, Julia i JavaScript. Postoji nekoliko velikih korporativnih sponzora, ali svi se mogu pridružiti NumFOCUS-u kao besplatni član zajednice – donacije se ne odbijaju od poreza.

Sa pipom možete instalirati pakete za gotovo sve, ali za korisnike programa Excel neki od najzanimljivijih su paketi za naučna izračunavanja. Naučimo nešto više o naučnom izračunavanju sa Pythonom u sledećem odeljku!

Naučno izračunavanje

Važan razlog za uspeh Pythona je činjenica da je stvoren kao programski jezik opšte namene. Mogućnosti naučnog računarstva su kasnije dodate u obliku paketa trećih strana. Ovo ima jedinstvenu prednost u tome što istraživač podataka može koristiti isti jezik za eksperimente i istraživanja kao veb programer, koji bi na kraju mogao da izgradi aplikaciju spremnu za upotrebu oko računarskog jezgra. Sposobnost pravljenja naučnih aplikacija na jednom jeziku smanjuje trenje, vreme implementacije i troškove. Naučni paketi poput NumPy, SciPy i pandasa daju nam pristup vrlo sažetom načinu formulisanja matematičkih problema. Kao primer, pogledajmo jednu od poznatijih finansijskih formula koja se koristi za izračunavanje varijanse portfolija prema Savremenoj teoriji portfolija:

$$\sigma^2 = w^T C w$$

Varijansa portfolija označava se σ^2, dok je w vektor težine pojedinačnih dobara i Cje matrica kovarijanse portfolija. Ako su `w` i `C` opsezi programa Excel, možete izračunati varijansu portfolija u VBA na sledeći način:

```
variance = Application.MMult(Application.MMult(Application.Transpose(w), C), w)
```

Uporedite ovo sa gotovo matematičkim zapisom u Pythonu, pretpostavljajući da su `w` i `C` pandas DataFrameovi ili NumPy nizovi (formalno ću ih uvesti u delu II):

```
variance = w.T @ C @ w
```

Ali ne radi se samo o estetici i čitljivosti: NumPy i pandas koriste kompajlirani Fortran i C kod u pozadini, što vam daje povećanje performansi pri radu sa velikim matricama u poređenju sa VBA.

Nedostatak podrške za naučna izračunavanja je očigledno ograničenje u VBA. Ali čak i ako pogledate osnovne jezičke karakteristike, VBA je zaostao, kao što ću istaći u sledećem odeljku.

Karakteristike savremenih jezika

Od Excela 97, jezik VBA nije imao većih promena u pogledu jezičkih karakteristika. To, međutim, ne znači da VBA više nije podržan: Microsoft isporučuje ažuriranja sa svakim novim izdanjem programa Excel kako bi mogao automatizovati nove funkcije programa Excel uvedene sa novim izdanjem. Na primer, Excel 2016 je dodao podršku za automatizaciju programa Power Query. Jeziku koji je prestao da se razvija pre više od dvadeset godina nedostaju savremeni jezički koncepti koji su tokom godina uvedeni u sve glavne programske jezike. Na primer, rukovanje greškama u VBA zaista pokazuje svoju starost. Ako želite elegantno da rešite grešku u VBA, to ide otprilike ovako:

```
Sub PrintReciprocal(number As Variant)
    ' Pojaviće se greška ako je rezultat 0 ili znakovni niz
    On Error GoTo ErrorHandler
        result = 1 / broj
    On Error GoTo 0
    Debug.Print "Nema greške!"
Finally:
    ' Izvršava se bez obzira na grešku
    If result = "" Then
        result = "N/A"
    End If
    Debug.Print "Recipročno je: " & result
    Exit Sub
ErrorHandler:
    ' Izvršava se samo u slučaju greške
    Debug.Print "Imamo grešku: " & Err.Description
    Resume Finally
End Sub
```

VBA rukovanje greškama uključuje upotrebu *oznaka* (eng. *labels*) poput `Finally` i `ErrorHandler` u primeru. Upućujete kôd da pređe na ove oznake pomoću naredbi `GoTo` ili `Resume`. Od ranije oznake su prepoznate kao odgovorne za ono što bi mnogi programeri nazvali *špageta kôd*: lep način da se kaže da je tok koda teško pratiti i stoga teško održavati. Zato su skoro svi aktivno razvijeni jezici uveli `try/catch` mehanizam – u Pythonu pod nazivom `try/except` koji ću objasniti u poglavlju 11. Ako ste iskusni VBA programer, možda ćete uživati i u činjenici da Python podržava nasleđivanje klasa, što je funkcija objektno orijentisanog programiranja koja nedostaje u VBA.

Osim savremenih jezičkih funkcija, postoji još jedan zahtev za savremeni programski jezik: kompatibilnost na više platformi. Hajde da vidimo zašto je ovo važno!

Međuplatformska kompatibilnost

Čak i ako razvijete kôd na lokalnom računaru koji radi na Windowsu ili macOS-u, velika je verovatnoća da ćete u nekom trenutku želeti da pokrenete program na serveru ili u oblaku. Serveri omogućavaju izvršavanje vašeg koda prema rasporedu i čine vašu

aplikaciju dostupnom sa bilo kog mesta, sa računarskom snagom koja vam je potrebna. Zapravo, pokazaću vam kako da pokrenete Python kôd na serveru u sledećem poglavlju upoznajući vas sa hostovanim Jupyter beležnicama. Velika većina servera radi na Linuxu, jer je to stabilan, siguran i isplativ operativni sistem. A budući da programi Python rade nepromenjeni na svim glavnim operativnim sistemima, ovo će vam biti veliko olakšanje pri prelasku sa lokalne mašine na proizvodno okruženje.

Nasuprot tome, iako Excel VBA radi na Windowsu i macOS-u, lako je uvesti funkcionalnost koja radi samo na Windowsu. U službenoj VBA dokumentaciji ili na forumima često ćete videti ovakav kod:

```
Set fso = CreateObject("Scripting.FileSystemObject")
```

Kad god imate poziv `CreateObject` ili vam se kaže da odete u Tools > References u VBA editoru da biste dodali referencu, skoro uvek imate posla sa kodom koji će se izvoditi samo u operativnom sistemu Windows. Još jedno istaknuto područje na koje morate paziti ako želite da vaše Excel datoteke rade u sistemu Windows i macOS su *ActiveX kontrole*. ActiveX kontrole su elementi poput dugmadi i padajućih menija koje možete postaviti na svoje listove, ali rade samo u operativnom sistemu Windows. Izbegavajte ih ako želite da i vaša radna sveska radi na macOS-u!

Zaključak

U ovom poglavlju smo upoznali Python i Excel, dve veoma popularne tehnologije koje postoje već više decenija – dugo u poređenju sa mnogim drugim tehnologijama koje danas koristimo. Londonski kit poslužio je kao primer koliko može poći po zlu (u dolarima) ako Excel ne koristite pravilno sa radnim sveskama kritičnim za misiju. Ovo nam je bio motiv da razmotrimo minimalan skup najboljih praksi programiranja: primenu razdvajanja nadležnosti, praćenje DRY principa i korišćenje automatizovanog testiranja i kontrole verzija. Zatim smo pogledali Power Query i Power Pivot, Microsoftov pristup bavljenju podacima, većim od vaše tabele. Čini mi se da oni često nisu pravo rešenje jer vas zaključavaju u Microsoft svet i sprečavaju vas da iskoristite fleksibilnost i moć savremenih rešenja zasnovanih na oblaku.

Python dolazi sa ubedljivim funkcijama koje nedostaju Excelu: standardna biblioteka, menadžer paketa, biblioteke za naučno računarstvo i kompatibilnost na više platformi. Naučivši kako da kombinujete Excel sa Pythonom, možete imati najbolje iz oba sveta i uštedećete vreme automatizacijom, napraviti manje grešaka jer je lakše slediti najbolje prakse programiranja, a moći ćete da preuzmete svoju aplikaciju i da je skalirate prema potrebama, van Excela, ako vam ikada zatreba.

Sada kada znate zašto je Python tako moćan pratilac za Excel, vreme je da podesite svoje razvojno okruženje da biste mogli da pišete prve redove Python koda!