DEO I

Uvod u Pyton

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:

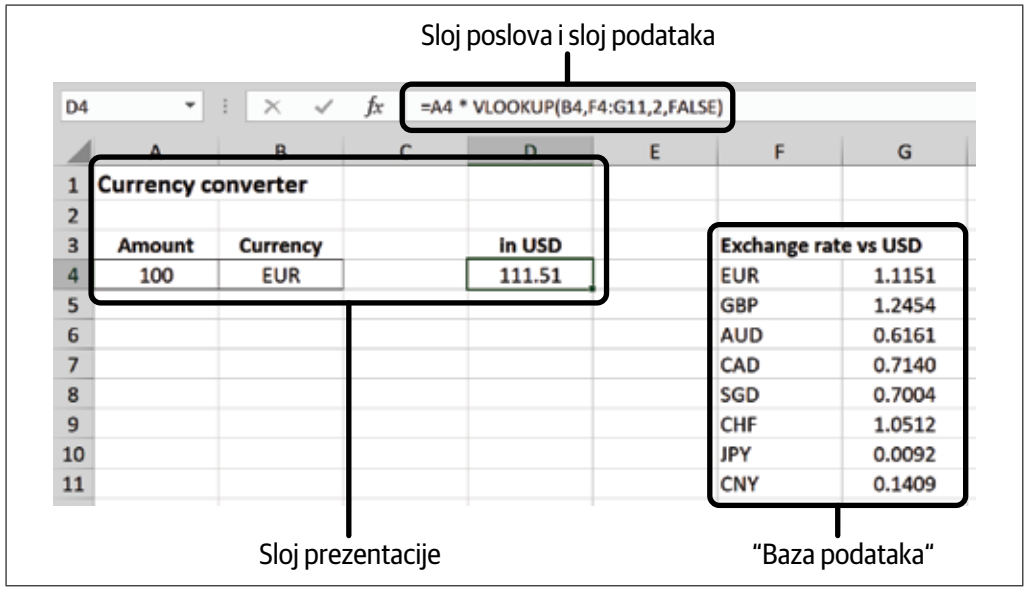
Sloj prezentacije
: Ovo je ono što vidite i sa njim komunicirate, tj. korisnički interfejs: vrednosti ćelija A4, B4 i D4 zajedno sa njihovim oznakama grade sloj prezentacije konvertora.

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!