Leren programmeren in Python

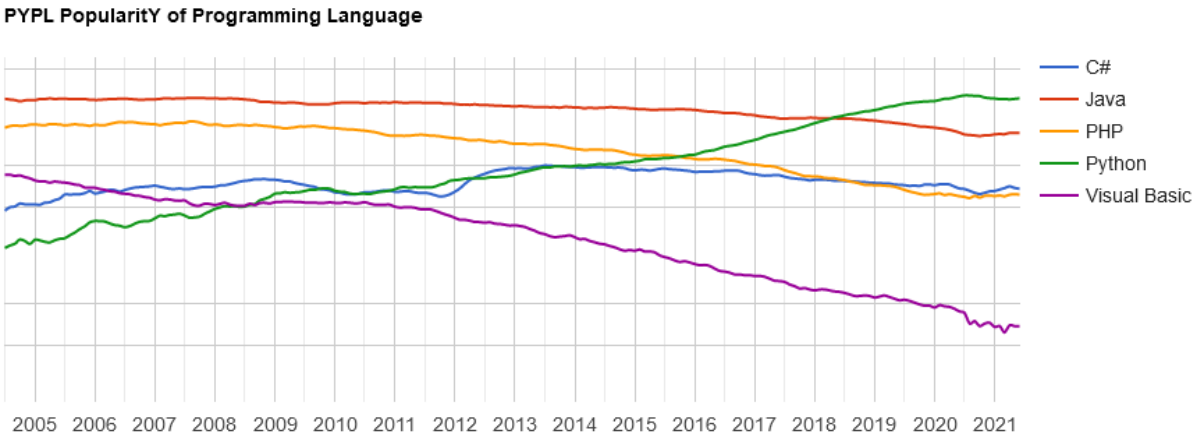
De programmeertaal Python werd in de jaren ’80 ontwikkeld door Guido Van Rossum als opvolger van de ABC-programeertaal. Hij werkte bij het Centrum Wiskunde & Informatica, waar hij theoretische computerwetenschappen en, je raadt het, wiskunde bestudeerde. Hij is deze taal blijven ontwikkelen als hoofdverantwoordelijke tot in het jaar 2018.

Daartussen is de taal erg gegroeid, te beginnen met versie 2.0 in het jaar 2000 en 3.0 in het jaar 2008. Het gekke is dat de laatste iteratie van versie 2.0, 2.7, nog tot in 2020 ondersteund werd. Dat is omdat niet alle code die geschreven is voor versie 2.7 ook werkt in versie 3.0.

Gelukkig kunnen we de problemen die daarbij hoorden ondertussen achter ons laten. We werken nu voluit in versie 3.0, en de laatste versie nu is 3.9.5. De kans is groot dat wanneer jij dit leest we aan een hogere versie zitten, maar zolang dat eerste cijfer, de 3, niet verandert, zit je goed: de voorbeelden blijven werken.

Maar dat is niet je grootste vraag op dit moment. Die is: “Waarom leer ik in hemelsnaam werken met een taal die uit de jaren ’80 stamt?”

Wel, twee redenen. Ten eerste is Python een erg populaire taal.



(PYPL PopularitY of Programming Language, 2021)

In deze grafiek staan alle talen waarover boeken zoals deze verschenen zijn, en Java. Roger Frans schreef het eerste boek over Visual Basic in de jaren ’90 en hield dat tot een stuk in de jaren ’00 vol. Daarna schakelde hij over op C#, een taal die ik in 2013 van hem overnam. Al die tijd hield Python zijn gestage opmars vol tot het ergens in 2018 de populairste programeertaal ter wereld werd.

Dat is omdat Python een taal is die zichzelf makkelijk laat uitbreiden. Je kan een Linux server opzetten, grafieken tekenen, Excel-bestanden analyseren, mechanische constructies uitrekenen, ... De mogelijkheden zijn eindeloos.

De tweede reden: Python is erg toegankelijke taal die je toch dwingt correct te programmeren. Je moet jezelf geen zorgen maken over declaratie van variabelen, maar je moet er wel voor zorgen dat je altijd net genoeg tabs en spaties zet. Dat laatste is erg belangrijk. In de vijftien jaar dat ik lesgeef in programmeren zag ik keer op keer hoe snel je

de mist ingaat wanneer je niet genoeg aandacht hebt voor spaties en tabs. Python helpt je hierbij.

Alhoewel de keuze voor Python als taal voor dit boek dus erg onderbouwd is, is "Python leren" maar een deel van het opzet. Dit boek is een hulpmiddel om je te "leren programmeren". Programmeren is de kunst van een vraagstuk om te zetten naar code, waarna een computer (in eender welke vorm) het vraagstuk voor je gaat oplossen. Een programmeur bij Microsoft heeft bijvoorbeeld de rekenmachine in Windows gemaakt.

Dat is een heel werkje geweest. Je moet knoppen maken om getalletjes in te voeren, die moet je dan kunnen vermenigvuldigen of delen. Het antwoord van zo'n som wordt opgeslagen zodat je het opnieuw kan gebruiken... Je zal een aantal hoofdstukken onder de knie moeten hebben vooraleer je zelf een rekenmachine kan bouwen.

Maar dit boek neemt je bij de hand bij de eerste stappen die je zet in de wondere wereld van het programmeren. Elk hoofdstuk volgt logisch op het volgende en aan de hand van voorbeelden wordt uitgelegd hoe een bepaald deel van Python werkt. Dit kan je dan bij het maken van de oefeningen instuderen. Het is, vooral in het begin, sterk aan te raden alle oefeningen te maken.

Op het einde van dit boek heb je een goede werkende kennis van Python. Hier bovenop zul je kunnen programmeren, waardoor je de technieken die je hebt geleerd ook kan toepassen in Javascript (voor een website), C# (voor een Windows-forms applicatie) of in C (op je Arduino).

Rest me nog je veel succes te wensen, en vooral veel programmeerplezier.

Jochen Mariën
Geel, juni 2021

PS: De voorbeelden uit dit boek zijn te downloaden op http://www.campiniamedia.be/, bij de infofiche van dit boek. Eventuele errata zullen ook daar verschijnen.

Inhoudsopgave

Richtlijnen bij het boek

In dit boek komt erg dikwijls programmacode voor. Die ziet er telkens als volgt uit:

```
# Versie 1
mondeling = 73
schriftelijk_opstel = 65
schriftelijk_woordenschat = 81
```

Code springt altijd een half centimetertje in en staat in het lettertype "Consolas". Dat is een lettertype waarbij elke letter even breed is, wat standaard is bij programmeren.

Alhoewel het erg te vermijden is, kan het dat een regel code langer is dan de bladspiegel. In dat geval zie je dit:

```
mondeling = 73
schriftelijk_opstel = 65
schriftelijk_woordenschat = 81
een_andere_veel_te_lange_variabelenaam = 30

totaal = (mondeling + schriftelijk_opstel + schriftelijk_woordenschat) /
                      een_andere_veel_te_lange_variabelenaam
```

Je ziet dat de regel raar eindigt, en dat de volgende regel ver is ingesprongen. Zo'n code moet je nooit overnemen (de voorbeelden trouwens sowieso niet, die kan je downloaden op de website van de uitgever), maar mag je op één regel overtypen.

Wanneer een regel te lang wordt doordat er een lange, statische tekst in nodig is korten we die als volgt af:

```
schriftelijk_woordenschat = int(input("Geef ... in:"))
```

Je kan in de uitvoer dan zien wat de tekst tussen de quotes moet zijn. Of je kan door de naam van de variabele vooraan ook al een goede vraag bedenken.

Uitvoer van een programma ziet er als volgt uit:

```
Geef de punten voor mondeling in: 73
Geef de punten voor schriftelijk opstel in: 65
```

```
Geef de punten voor schriftelijk woordenschat in: 81
In totaal haalde je 7.3 op tien.
```

Hierbij staat hetgene je als gebruiker van het programma ingeeft in italics of cursief en in het vet. In dit geval zijn dat de getallen 73, 65 en 81.

Dit boek is een Nederlandstalig boek, maar soms wordt de Engelse term voor een bepaald concept ook gebruikt. Zo is "tekst" in een programma altijd een "string". Het vertalen zorgt in zo'n geval alleen voor toegevoegde verwarring. Zulke termen worden ook altijd eerst uitgelegd vooraleer ze veelvuldig voorkomen.

Ook namen van variabelen en functies zijn in dit boek doorgaans in het Nederlands gekozen. Dat is niet conform de industrie-standaard, maar het verhoogd wel erg de leesbaarheid, net hetgeen we in een boek waarin je leert programmeren belangrijk vinden.

Niets houd je echter tegen om in de oplossingen van de oefeningen wel Engelstalige namen te kiezen, of Franstalige, of Finse, ... Zorg er echter wel voor dat je in elk programma één taal kiest en je daar consequent aan houdt.

Besluit

Je kan nu met behulp van Python een programma schrijven om eenvoudige problemen op te lossen. Als de Python-microbe je gebeten heeft, merk je dat je na een tijdje bij bijvoorbeeld een youtube-filmpje over het 3x+1-probleem (ofwel het Collatz-vermoeden) denkt "ik ga dat eens programmeren en zien waar ik uitkom."

Spijtig genoeg ben je er nog niet. Functies zijn tot nu toe de enige manier die we hebben bestudeerd om code te scheiden. Code (kunnen) scheiden is erg belangrijk, want programma's worden snel onoverzichtelijk als ze meer en meer kunnen.

De volgende stap is om klasses te maken. Dan kan je niet langer een programma maken door te beginnen programmeren en bekijken waar je uitkomt. Je moet een analyse op voorhand maken om duidelijk te weten wat je nodig hebt en hoe alles in elkaar past. Vanaf dan kan je ook met meerdere programmeurs tegelijk aan één programma werken ("Werk jij aan de klasse "Boek", dan zorg ik dat de klasse "Uitlener" werkt.")

Wanneer je klasses onder de knie hebt, kijk je vervolgens naar design patterns. Design patterns zijn standaard oplossingen voor problemen die altijd maar terugkomen. Bijvoorbeeld de Model-View-Controller zorgt ervoor dat op een website "de databank" en "de website zelf" niet in elkaars vaarwater komen. En dat is moeilijker dan het lijkt.

Je hebt ook nog maar één taal geleerd, Python. Python is een handige en gebruiksvriendelijke taal, maar elke taal heeft zijn eigen sterke punten. Er zijn toepassingen waarvoor C# of Java een veel betere keuze zijn. Het goede nieuws is dat de principes uit dit boek grotendeels toepasbaar zijn op elke nieuwe programmertaal die je wil leren.

Je moet trouwens niet te snel aan extra talen beginnen, met Python ga je al erg ver komen. Het is een breed toepasbare taal. Je kan:

- Een foto of livestream van een camera analyseren (OpenCV).
- Een Excel-bestand inladen en analyseren (Pandas).
- Grafieken maken van gegevens (plotly).
- Een website bouwen (Flask).
- Een Linux-server beheren (sys).
- ...

Je kan dus kiezen hoe je verder aan de slag gaat met Python. Het belangrijkste is wel dat je de taal blijft gebruiken. Programmeren is een vaardigheid die je moet blijven bijschaven om ze scherp te houden. Door veel te programmeren ga je ook merken dat je er beter in wordt, wat het nog fijner maakt. Dit was een beginners-boek alhoewel zeker de laatste hoofdstukken al niet meer voor prille beginners zijn. Het is normaal dat je vloekte of het boek eens met passie door de kamer slingerde.

Onderschat niet de weg die je hebt afgelegd. Veel succes met de volgende uitdagingen op je pad!