Inhoud

1 Kennismaken met Prakisch Python 1

Een (extreem) korte introductie in Python **2**
Guido van Rossum 2
Versies van Python 2
Kenmerken van Python **4**
Eenvoud 4
Ecosysteem 4
Standaardbibliotheek 5
Leesbare code 6
Duck-typing 7
Interpreter 7
Samenvatting 8
Voor wie is dit boek bedoeld? **9**
Benodigde voorkennis 10
Ik heb nog nooit Python gebruikt 10
Ik kan al programmeren in taal XYZ! 11
Wat hoef ik niet te weten? 12
Waarom een boek? **12**
Oplossingen 13
Hoe zijn de projecten vormgegeven? **14**
Hints voor verdere uitwerking 15
Een praktijkvoorbeeld – eenvoud 15
Projecten als voorbeelden 17
Python downloaden en installeren **17**
Installatie op Windows 18
Installatie op Mac/Linux 20
Problemen met installeren? 22
Een editor kiezen **23**
Visual Studio Code 23
PyCharm 26
Overige editors 26

Aanvullende Python-modules installeren 27
Python package manager 28
Welke packages zijn al geïnstalleerd? 29
Python op je mobiele telefoon of tablet 30
Voorbeeldcode downloaden en gebruiken 31
Boek registreren 31
De voorbeeldcode 31
Engels en Nederlands 31
Programma's debuggen 32
De debugger 32
Informatievensters 34
Oefening 35
Conclusie – aan de slag 37

2 Project Raden Maar 39

Het project 40
De uitvoer 40
De code 41
Analyse van Raden Maar 42
Modules 42
Commentaar 43
Constanten en variabelen 43
Regels voor naamgeving van Python-variabelen 44
'Nadenken' 44
Willekeurig getal genereren 45
Strings formatteren 45
Hoofdlus 46
If-statements 46
Break 47
Uitbreiding – een oneindige lus 47
Functie om willekeurig getal te berekenen 48
While True 48
Ideeën voor uitbreiding en verbetering 49
Constanten aanpassen 49
Geen te grote getallen toestaan 49
Enkelvoud en meervoud in de boodschappen 50
Nadenken op één regel 50
Meer lezen 50

3 Project Geboortedatum 51

Het project **52**
De uitvoer **53**
De code **53**
Analyse van Geboortedatum **55**
Modules 55
De functie age() 55
Docstrings 56
Docstrings en de helpfunctie 57
Inhoud van age() 58
Ternary statements in Python 59
De functie main() 61
String converteren naar datumobject 61
if __name__ == '__main__': 62
Uitleg 63
importlib 64
Conclusie 64
Uitbreiding/verbetering **65**
Vervolgprojecten **65**
Meer lezen **66**

4 Project Mastermind 67

Het project **68**
Logisch nadenken 68
De uitvoer **68**
Code **70**
Analyse **72**
Docstrings afdrukken 72
Geheime code bepalen 72
Controleren op geldige invoer 73
De hints samenstellen 74
De hoofdlus van het programma 74
Ideeën voor uitbreidingen of aanpassingen **75**
Vervolgprojecten **76**
Meer lezen **76**

5 Project Persoonsgegevens via API 77

Het project **78**
De requirements 79
De uitvoer **80**
Foutafhandeling 81
Code – stap 1 **82**
Package requests 82
Analyse **83**
Foutafhandeling met try-except 83
Leeg except-blok aan het einde 84
Http-foutafhandeling met raise_for_status() 85
Verwerk HTTPError 85
Code – stap 2 **85**
De functie json.dumps() 86
Code – stap 3 **87**
Ideeën voor uitbreidingen of aanpassingen **88**
Algemene structuur 89
Verder uitbreiden 91
Vervolgprojecten **91**
Movie Finder 91
Country Finder 94
Meer open API's 95
Kenmerken? 96
Meer lezen **96**

6 Project API maken met Flask 97

Het project **98**
Werken met virtuele omgevingen **99**
Een virtuele omgeving maken 99
Stappenplan 100
Activeren 100
Packages installeren 101
Pip list 101
Deactiveren 102
Flask versus Django **102**
Wat is Flask? 102
Wat is Django? 103
Volgende stappen 104
De uitvoer **104**
Personen teruggeven 106
Gegevens POSTen 106

Code – stap 1 **107**
Analyse 108
Code – stap 2 en 3 **109**
Oefeningen 109
Code – stap 4 **110**
Analyse 110
Code – stap 5 **111**
Analyse 111
Code – stap 6 **112**
Analyse 112
POST testen 112
Flask-applicatie automatisch opnieuw starten **114**
Debugmodus 114
Breekpunten 114
Ideeën voor uitbreidingen **115**
Vervolgprojecten **116**
Meer lezen **116**

7 Project Website maken met Flask 119

Het project **120**
Webpagina's meegeleverd 120
Virtuele omgeving 121
Mappen en bestanden voor de website 122
De uitvoer **122**
Code – stap 1 **125**
Analyse 125
Code – stap 2 **126**
Analyse 127
Code – stap 3 **128**
Python 128
layout.html 129
Analyse 130
index.html 131
Analyse 131
Code – stap 4 **132**
Python 132
Detail.html 133
Analyse 133
Code – stap 5 **134**
Python 134
Analyse 134
add.html 135
Analyse 136

Je webapp publiceren **137**
PythonAnywhere 137
Bestanden uploaden 139
Opstartbestand aanpassen 140
Website beoordelen 141
Ideeën voor uitbreidingen **142**
Vervolgprojecten **142**
Meer lezen **142**

8 Project Website met database 143

Het project **144**
Structuur 145
Virtuele omgeving 145
De uitvoer **146**
Code **147**
Databaseschema 147
De database genereren 147
Alle gebruikers tonen **149**
Helperfuncties 150
Analyse 151
De route schrijven 151
Details voor een enkele gebruiker tonen **152**
Analyse 153
Gebruiker verwijderen **153**
Analyse 153
Gebruikers wijzigen **154**
Analyse 155
HTML voor wijzigen **155**
Conclusie **157**
Ideeën voor uitbreidingen **157**
Meer lezen **158**

9 Project Exif-gegevens verwijderen 159

Het project **160**
Werken met eenvoudige bestanden 160
Virtuele omgeving? 161
De uitvoer **161**
Code – stap 1 **162**
Analyse 163
Bestanden openen, lezen, schrijven en toevoegen 163

Code – stap 2 **164**
Analyse 165
Controleren of bestand bestaat 165
De methode with open(…) as … 166
Uitgebreidere foutcontrole met bestanden 167
Try-except 168
Code – stap 3 **168**
Exif-gegevens tonen (nog niet verwijderen) 169
Analyse 169
Exif-gegevens verwijderen 170
De package exif **171**
Ideeën voor uitbreidingen **172**
Vervolgprojecten **172**
Meer lezen **174**

10 Project Afbeeldingen bewerken **175**

Het project **176**
Upgraden 176
Mogelijkheden van Pillow 177
Top-5 178
De uitvoer **179**
Thumbnails 179
Datum en tijd als overlay 179
Code – thumbnails **180**
Analyse 181
Alternatieven 182
Code – overlays **183**
Analyse 184
Ideeën voor uitbreidingen **185**
Meer lezen **186**

11 Project Webscraping **189**

Het project **190**
Modules 190
Virtuele omgeving 190
De uitvoer **191**
Code – stap 1 **193**
Analyse 194
Code – stap 2 **195**
Analyse 196
HTML-code 196

Code – stap 3 **197**
Analyse 198
Meer over Beautiful Soup 200
Code – stap 4 **202**
Analyse 203
Code – stap 5 **204**
Analyse 206
Ideeën voor uitbreidingen **207**
Vervolgprojecten **208**
Meer over webscraping **210**
Meer lezen **210**

12 Project PythonExcel 211

Het project **212**
Modules 212
De uitvoer **213**
Code – stap 1 **215**
Analyse 215
Code – stap 2 **216**
Analyse 216
Code – stap 3 **217**
Analyse 217
Code – stap 4 **218**
Analyse 219
Meer over werkbladen **221**
Hardware store 221
Mogelijkheden van openpyxl 221
Praktisch gebruik 222
Werkbladen en cellen **223**
Werkboek openen en werkbladen afdrukken 223
Cellen inlezen 224
Rijen en kolommen afdrukken 225
Rechthoekige gebieden 225
Nieuw werkblad maken en vullen 226
Case – prijzen in een werkblad aanpassen **227**
De uitvoer 228
Code 228
Analyse 229
Ideeën voor uitbreidingen **230**
Vervolgprojecten **230**
Meer lezen **232**

13 Project PythonPdf 235

Het project **236**
Pdf's lezen en schrijven 236
ReportLab 237
De uitvoer **238**
Code **240**
Analyse 241
Metagegevens toevoegen **242**
Conclusie **243**
Ideeën voor uitbreidingen **243**
Meer lezen **244**

14 Project Audioboeken 245

Het project **246**
De modules 246
hulpbestanden 247
De uitvoer **249**
Code – stap 1 **250**
Analyse 250
Code – stap 2 **252**
Analyse 252
Code – stap 3 **253**
Analyse 254
Ideeën voor uitbreidingen **255**
Vervolgprojecten **255**
Meer lezen **256**
Conclusie **256**
Meer mogelijkheden 257
Bonusfunctie – mail verzenden 258
App password 259

Index 261

Hoofdstuk 1

Kennismaken met Prakisch Python

Je kent Python al, maar je bent op zoek naar handvatten om je theoretische kennis van lists, dicts, tuples en functies om te zetten in praktische projecten? Welkom bij Praktisch Python! In dit boek ga je concreet aan de slag met Python. Er zijn geen hoofdstukken waarin wordt uitgelegd wat variabelen of functies zijn, of waarin wordt uitgelegd hoe een while-lus werkt. In plaats daarvan worden deze Python-kenmerken gebruikt om direct applicaties te maken die je rechtstreeks kunt inzetten, of kunt gebruiken als basis of idee voor eigen toepassingen. Dat laatste is natuurlijk nog beter! Maar ook wanneer je nog geen Python-voorkennis hebt en benieuwd bent wat deze programmeertaal in petto heeft, kun je met dit boek aan de slag. Gaandeweg maak je kennis met de belangrijkste kenmerken van Python. Dit eerste hoofdstuk geeft een korte inleiding en daarna gaan we direct aan de slag.

In dit hoofdstuk:

Een korte introductie in Python (echt kort!)

Voor wie is dit boek geschikt en voor wie niet?

Wat zijn de projecten die in dit boek worden beschreven?

Python downloaden en installeren.

Gereedschappen: een editor kiezen en Python-scripts debuggen.

Aanvullende Python-modules installeren met pip.

De voorbeeldcode downloaden en gebruiken.

Aan de slag.

Een (extreem) korte introductie in Python

Guido van Rossum

Python is oorspronkelijk een Nederlandse uitvinding. De programmeertaal is in de jaren tachtig en negentig van de vorige eeuw ontwikkeld door Guido van Rossum. Van Rossum was destijds werkzaam bij het Centrum voor Wiskunde en Informatica in Amsterdam (CWI). Daarna heeft hij onder meer nog gewerkt bij Google, Dropbox en Microsoft. Hoewel Guido van Rossum inmiddels gepensioneerd is, is hij nog steeds nauw betrokken bij de ontwikkelingen rondom Python.

De taal is open source, hetgeen betekent dat iedereen kan beschikken over de broncode en hiermee toepassingen kan schrijven. Je hoeft niet te betalen voor het gebruik ervan. De naam is overigens niet afkomstig van de slang python, maar van het Engelse comedygezelschap *Monty Python*.

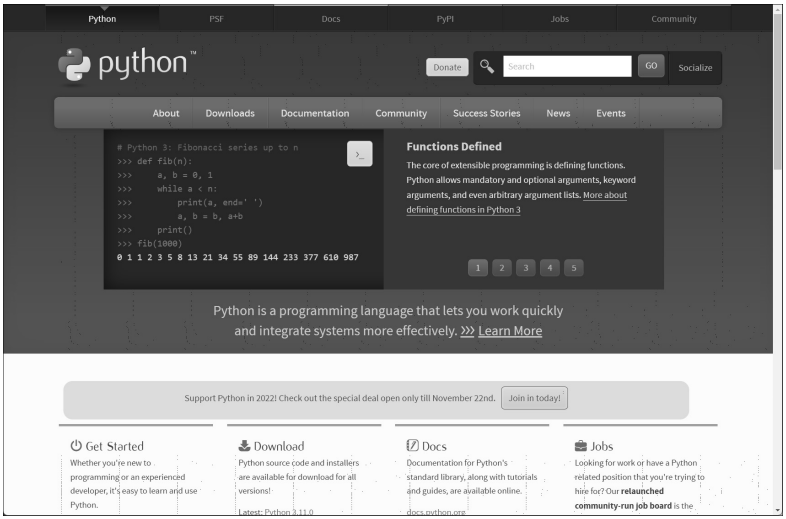
Afbeelding 1.1 *Guido van Rossum is de maker van Python (bron afbeelding: CWI).*

Versies van Python

Er zijn verschillende versies van Python in omloop. Dat kan ook bijna niet anders bij een programmeertaal die al ruim dertig jaar oud is. Een – zeer beknopte – tijdlijn van de geschiedenis van Python ziet er als volgt uit:

- 1989 – Eerste implementatie van Python, voor persoonlijk gebruik en op het CWI in Amsterdam.
- 1994 – Python 1.0
- 2000 – Python 2.0
- 2008 – Python 3.0

Binnen de hoofdversies zijn weer allerlei subversies verschenen, zoals Python 2.1, 2.2 enzovoort. Op het moment van schrijven van dit boek was Python 3.11 de meest recente versie. Op de homepage van Python (**python.org**) vind je altijd de meest recente versies en kun je een exemplaar downloaden voor jouw besturingssysteem. Python is beschikbaar voor Windows, macOS en Linux.



Afbeelding 1.2 *Ga naar python.org om de nieuwste versie te downloaden, documentatie en nieuwsberichten te lezen, en meer.*

Python 2 en Python 3

Python 2 was lange tijd de standaardversie van Python en er zijn talloze programma's in geschreven. Ook op dit moment is Python 2 nog erg populair (hoewel het dus al in 2008 werd opgevolgd door Python 3). Vooral bij bestaande applicaties moet je goed kijken in welke versie van Python ze zijn geschreven. Python 2 en 3 zijn weliswaar grotendeels hetzelfde, maar kennen ook belangrijke onderlinge verschillen. In programmeertermen: de talen zijn niet honderd procent compatible. Code die is geschreven voor Python 2 kan in Python 3 problemen opleveren en omgekeerd. Nieuwe applicaties worden vrijwel zonder uitzondering geschreven in Python 3. Dat is ook de versie die we in dit boek gebruiken.

Kenmerken van Python

De populariteit van Python vertoont een beetje de kenmerken van een golfbeweging. In het begin, met name ten tijde van Python 2, was de taal zeer populair. Daarna zakte de belangstelling een beetje in. Maar de laatste jaren mag Python zich weer op toenemende populariteit verheugen. Dat heeft zonder twijfel te maken met het feit dat softwareontwikkeling steeds complexer wordt. Java en C#/.NET-toepassingen zijn krachtig, maar ook erg ingewikkeld. Zelfs voor een simpele app op Android of IOS moet je al tientallen gigabytes aan schijfruimte opofferen en uren durende installaties doorlopen van tools als Android Studio of XCode.

Eenvoud

Python is daar een aangename uitzondering op. De taal zelf is relatief eenvoudig. Er zijn niet al te veel typeringen en er is een beperkte instructieset. Dit is de kern van Python. Het bevat onderdelen als variabelen, lists, dicts, tuples en functies. Voor het programmaverloop worden voornamelijk `if`- en `while`-statements gebruikt, terwijl de data met rechttoe-rechtaan operatoren voor optellen, aftrekken, delen en vermenigvuldigen worden bewerkt. Dat is heel overzichtelijk en in een dag te leren.

Ecosysteem

Dat betekent – gelukkig! – echter niet dat met Python alleen maar eenvoudige programma's gemaakt kunnen worden. Rondom de kern van Python is een enorm *ecosysteem* van aanvullende modules en toepassingen beschikbaar. Als je al enige Python-kennis hebt (en dat heb je waarschijnlijk, anders zou je dit boek niet lezen), dan weet je dat dit de *Python Package Index* is, of kortweg pypi, beschikbaar op **pypi.org**.

Afbeelding 1.3 *Op pypi.org zijn duizenden aanvullende packages aanwezig waarmee de kernmogelijkheden van Python worden uitgebreid.*

Met een eenvoudige opdracht als `pip install <naam_van_package>` voeg je een onvoorstelbare hoeveelheid mogelijkheden toe aan je Python-programma. Zo hoef je het wiel niet opnieuw uit te vinden. Er zijn packages voor rekenkundige bewerkingen, grafische user interfaces, frameworks, API's, hardware- en softwareprotocollen, werken met natuurlijke taal en kunstmatige intelligentie (AI), modules voor specifieke hardware zoals telefoons, calculators en geïntegreerde systemen, en nog veel meer.

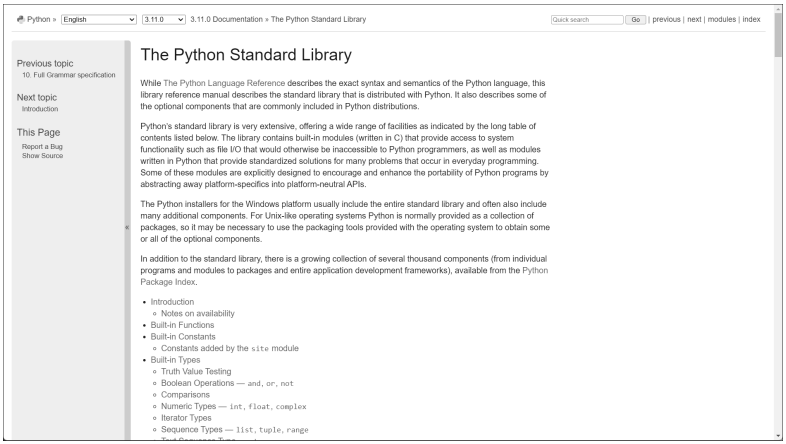
Pypi en andere package managers

De Python Package Index laat zich wat dat betreft bijvoorbeeld vergelijken met NPM en Nodejs. Dit is het packagesysteem voor JavaScript- en webdevelopment. C# en .NET-ontwikkeling kennen Nuget als package manager. Java heeft Maven en Gradle als package managers. Hoewel ze allemaal in functionaliteit verschillen, hebben ze in de kern hetzelfde doel: de basismogelijkheden van de desbetreffende programmeertaal uitbreiden.

Standaardbibliotheek

Maar om goede programma's te schrijven, is het beslist niet nodig altijd Pypi in te zetten. Integendeel. Een kenmerk van Python is dat het met een zeer uitgebreide standaardbibliotheek wordt geleverd. Hiermee kun je rechtstreeks standaardmodules en -packages importeren en belangrijke handelingen uitvoeren. Denk bijvoorbeeld aan een opdracht in je code als:

```
import math
```



Afbeelding 1.4 *De Python Standard Library bevat tientallen modules met honderden functies om veelvoorkomende programmeerproblemen op gestandaardiseerde wijze op te lossen.*

Hiermee wordt de standaardbibliotheek met allerlei wiskundige functies geïmporteerd (sinus, cosinus, pi, afronden enzovoort). Je hebt `math` niet eerst apart hoeven installeren. Het wordt meegeleverd met Python. De standaardbibliotheek biedt – zoals de naam al aangeeft – gestandaardiseerde oplossingen, functies en code voor alledaagse programmeerproblemen.

Denk aan het werken met wiskundige functies (zoals hiervoor beschreven), het werken met het besturingssysteem en bestanden, met datums en tijden, met bestandsformaten zoals `csv` en `json`, e-mail, HTML en nog veel, veel meer. Een compleet overzicht van de Python-standaardbibliotheek is te vinden op **docs.python.org/3/library/**.

Niet overal een standaardbibliotheek

Niet alle programmeertalen en -platforms kennen zo'n standaardbibliotheek. In Node.js zijn bijvoorbeeld wél standaardmodules aanwezig voor het werken met bestanden, protocollen, streams enzovoort. Maar een taal als JavaScript heeft geen standaardbibliotheek. Daarom hebben talloze programmeurs allemaal zelf hun functies en reguliere expressies geschreven om bijvoorbeeld een e-mailadres te valideren. Hetzelfde geldt voor het werken met datums en tijden, formuliervalidaties en meer. Een willekeurig bezoekje aan Stack Overflow bevestigt dit JavaScript-kenmerk. Je bent helaas vaak bezig het wiel opnieuw uit te vinden. In Python hoeft dat niet – al moet je natuurlijk wel weten dat er standaard een module aanwezig is die het desbetreffende klusje voor je klaart.

Leesbare code

In Python ligt de nadruk zwaar op leesbare code. Guido van Rossum vond het belangrijk dat programma's bijna als een gewoon boek te lezen waren. Daarom wordt bijvoorbeeld structuur in een programma aangebracht door inspringing. Er zijn geen accolades zoals in JavaScript en PHP/C/C++/C#/Java. Je hoeft een statement ook niet af te sluiten met een puntkomma. Het einde van een regel betekent simpelweg het einde van een statement. De volgende codeblokken laten dit verschil duidelijk zien.

```
# Python-code
print('Voer tekst in, of typ "stop" om af te sluiten:')
while True:
    response = input()
    print('Je typte: ' + response)
    if (response.lower() == 'stop'):
        break
```

Hoewel de codevoorbeelden niet hetzelfde doen, is het verschil duidelijk. In Python (hiervoor) worden codeblokken gedefinieerd door inspringing. De standaardinspringing is vier spaties. In JavaScript-code (hierna) maar ook in C of Java worden codeblokken gedefinieerd door `{ … }`:

```
// Node.js/JavaScript-code
const prompt = require('prompt');
prompt.start();
prompt.get(['username', 'email'], function (err, result) {
    console.log('Invoer via toetsenbord:');
    console.log('  Username: ' + result.username);
    console.log('  Email: ' + result.email);
});
```

Weer andere talen werken met sleutelwoorden zoals `BEGIN` en `END` of `{% … %}` om codeblokken te omsluiten. Maar in Python wordt dus uitsluitend inspringing gebruikt.

Na een opdracht die een codeblok begint (zoals het `while`- en `if`- statement in de Python-code hiervoor), wordt een dubbele punt geplaatst. De volgende regel begint dan vier spaties ingesprongen (of acht, twaalf enzovoort) ten opzichte van de regel daarvoor.

Duck-typing

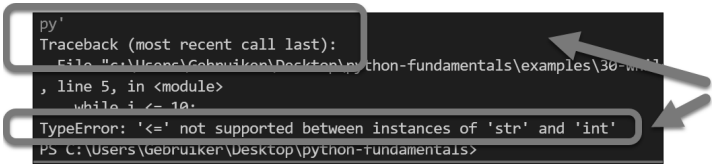
Als je meer leest over Python zul je ongetwijfeld het begrip *duck-typing* tegenkomen. Dit betekent dat het type van een variabele en bijvoorbeeld van een argument/parameter van een functie niet vooraf gedefinieerd hoeft te worden (het mág vaak overigens wel, maar is dus niet verplicht). Bij compilatie vindt dan ook geen typecontrole vooraf plaats. Als het niet klopt, wordt er runtime een foutmelding gegenereerd. Dit is de verantwoordelijkheid van jou als programmeur. Hierin lijkt Python wel een beetje op JavaScript en PHP. Ook dit zijn dynamisch getypeerde programmeertalen (Engels: *loosely* of *dynamically typed*). In andere talen (ook weer: C#, Java) moet je vaak vooraf het type van een variabele of argument opgeven. In Python is dat niet verplicht.

Interpreter

Een ander kenmerk is dat een Python-programma niet vooraf hoeft te worden gecompileerd tot een uitvoerbaar bestand zoals in Java of C. De Python-interpreter compileert het programma zodra het wordt gestart en voert het daarna regel voor regel uit. Als er iets niet klopt, volgt een foutmelding en wordt het programma afgebroken. Je bent ongetwijfeld wel eens een scherm tegengekomen als in afbeelding 1.5.

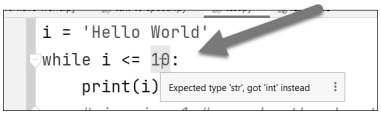
- Achter `Traceback` staat vaak een bestandsnaam en regelnummer waar de fout is opgetreden.

- Bij `xxxError` staat welk type fout is opgetreden. Dit kan bijvoorbeeld zijn een `TypeError` (zoals in afbeelding 1.5), maar ook een `ValueError`, `OSError` of nog anders, afhankelijk van de bewerking die op dat moment gaande was.



Afbeelding 1.5 *Fouten worden altijd runtime gegenereerd. Probeer te ontdekken waar de fout is opgetreden en wat het type van de fout is. Dan kun je op basis daarvan verder zoeken.*

Goede editors wachten overigens niet af totdat de Python-runtime een fout geeft, maar tonen in de user interface alvast dat code logische fouten bevat. Een reden te meer om te investeren in een goede editor en je code niet in Kladblok te schrijven. Meer over editors lees je verderop in dit hoofdstuk.



Afbeelding 1.6 *In de editor (hier: PyCharm) wordt aangegeven dat het programma een logische fout bevat.*

Dat Python-code door een interpreter wordt verwerkt, betekent ook dat je altijd met leesbare code werkt en eenvoudig breekpunten kunt zetten in de code. Dat is handig bij het debuggen. Python wordt niet eerst gecompileerd naar (onleesbare) bytecode en vervolgens uitgevoerd.

Samenvatting

Uiteraard zijn er nog veel meer kenmerken van Python op te noemen, maar de belangrijkste zijn nu genoemd. Samengevat zou je het volgende kunnen stellen:

"Python is een multifunctionele programmeertaal die niet wordt vertaald naar bytecode. Met eenvoudig Engels geef je aan wat je bedoelt. Python wordt rechtstreeks uitgevoerd door een interpreter, heeft een dynamisch typesysteem en focust zich op leesbaarheid."

Voor wie is dit boek bedoeld?

Praktisch Python is bedoeld voor programmeurs. Dat klinkt eenvoudig, maar het betekent onder meer dat je je graag bezig houdt met code. Je geeft de computer opdrachten die worden uitgevoerd. Als je opdrachten een fout bevatten (een logische fout of een structurele fout), dan wordt het simpelweg niet uitgevoerd, of de resultaten zijn niet wat je verwacht. Dat is heel duidelijk. Een computer kent alleen maar enen en nullen, het is goed of het is fout.

Dit boek is daarmee geschikt voor degenen die zelf graag met code aan de slag gaan, willen experimenteren en het resultaat daarvan snel op het scherm willen zien. Het is niet geschikt voor computergebruikers die meer visueel zijn ingesteld. Denk bijvoorbeeld aan Photoshop, Illustrator, PowerPoint of Figma. Ook is dit geen geschikt boek als je via een Windows-wizard, of graag in een visuele low-code of no-code omgeving programma's maakt. Daar is niks mis mee, maar daar is dit boek niet voor bedoeld. Kijk in dat geval nog even verder.

```
print('opening workbook')
fileName = 'multiplication-table.xlsx'
try:
    wb = openpyxl.Workbook()
    sheet = wb.active

    print('calculating...')
    number = 6 # TODO: get this from command line
    for i in range(1, number + 1):
        #  First row, print number 1...x
        sheet.cell(row=1, column=i+1).value= i
        # second row, print number again, the i * numbers
        newRow = [i,]
        # calculate the multiplication number
        for j in range(1, number+ 1):
            newRow.append(j * i)
        # print to the screen, for reference, and add to worksheet
        print('adding row: ', newRow)
        sheet.append(newRow)

except Exception as ex:
    print('ERROR! {0}'.format(ex))

print('closing workbook: ', fileName)
wb.save(filename=fileName)
print('Done.')
```

Afbeelding 1.7 *We gaan ervan uit dat je graag met code werkt, bereid bent fouten te maken en te onderzoeken hoe je deze fouten kunt oplossen.*

Projecten

Dit boek is vooral gebaseerd op praktische projecten. We beginnen in het volgende hoofdstuk eenvoudig, maar gaandeweg wordt het telkens een stapje complexer. Je hoeft dit boek niet van begin tot einde door te werken in de aangewezen volgorde. Pik er een project uit dat jouw interesse heeft en zie hoe Python je hierbij kan helpen.

Benodigde voorkennis

Hoewel we eenvoudig beginnen, gaan we er in dit boek van uit dat je al over enige Python-voorkennis beschikt. Je weet wat variabelen zijn, je kunt het verschil tussen een `list` en een `set` uitleggen, je weet wat er met een `dict` wordt bedoeld, je weet wat een `tuple` is enzovoort.

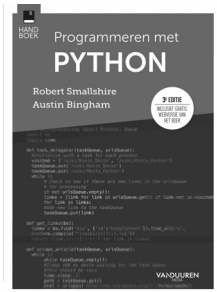
Dit is geen klassiek leerboek, in de zin van 'we beginnen met variabelen en beschrijven daarna alle onderdelen van de programmeertaal van A tot Z'. Het is ook minder geschikt als naslagwerk. Je vindt niet op alfabetische volgorde een overzicht van alles wat we bespreken.

Het is evenwel ook geen boek voor de hardcore Python-gebruiker. Die vindt waarschijnlijk meer van zijn gading op Stack Overflow, Reddit, Medium of andere gespecialiseerde Python-fora met oplossingen voor heel specifieke problemen.

Dit boek is vooral geschikt voor beginnende en intermediate Python-programmeurs.

Ik heb nog nooit Python gebruikt

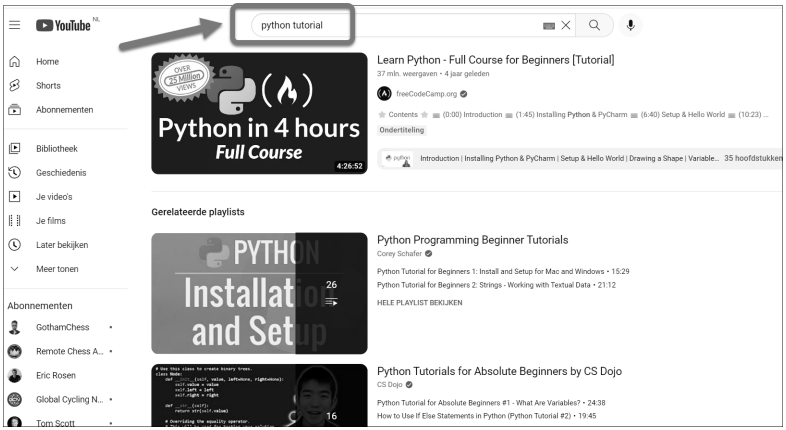
Mocht je vinden dat je Python-voorkennis nog niet op orde is, dan zijn er talloze andere bronnen waar je je dit stap voor stap kunt leren. Kijk in dat geval bijvoorbeeld naar het uitstekende *Handboek Programmeren met Python, 3e editie* van Robert Smallshire en Austin Bingham (ISBN 978-94-6356-227-0). Dit boek is ook uitgegeven door Van Duuren Media; zie **mijn.cc/python**. Ook op YouTube zijn talloze tutorialkanalen beschikbaar. Udemy beschikt over een schat aan Pythoncursussen en zo is er enorm veel te vinden.



Afbeelding 1.8 *Het Handboek Programmeren met Python is geschikt als je Python onder de knie wilt krijgen voordat je aan de projecten in dit boek begint.*

Verdere voorkennis die handig is:

- Je moet overweg kunnen met mappen en bestanden op het besturingssysteem waarop jij werkt (Windows, macOS of Linux). Het wisselen tussen mappen/directories/folders kent geen geheimen voor jou.
- Je kunt omgaan met een opdrachtregelomgeving zoals Windows CMD of Terminal, of Mac Terminal. Je kent hierin de belangrijkste opdrachten als `cd`, `mkdir`, `ls` enzovoort.
- Je bent bekend met het installeren van toepassingen vanaf internet. Indien noodzakelijk op jouw computer kun je het pad (`path`) aanpassen zodat het uitvoerbare programma `py` of `python` (op Windows) of `python3` (op Mac/Linux) gevonden kan worden.
- Enige kennis van de opbouw van relationele databases met tabellen en records en van de taal SQL is handig. Dit geldt natuurlijk voornamelijk voor het project Websites en databases, dat je kunt vinden in hoofdstuk 8.
- Als je meer wilt doen met Python en websites (zoals de projecten in hoofdstuk 6 en hoofdstuk 7) komt het ook van pas dat je HTML en CSS kent. We stellen de voorbeelden beschikbaar via de downloads bij dit boek, maar we gaan er in de tekst niet op in.



Afbeelding 1.9 *Ook op YouTube zijn talloze introducties in Python te vinden.*

Ik kan al programmeren in taal XYZ!

Maar misschien heb je ervaring in een andere programmeertaal dan Python. Je hebt geen tijd of zin om langdurig andere boeken of lessen door te werken, waarbij je zestig of zeventig procent al kent. Je kunt bijvoorbeeld goed overweg met C#, Java, PHP of JavaScript. Je wilt alleen maar weten 'Hoe doe ik dit in Python?'.

In dat geval is dit boek *heel erg geschikt* voor jou. Omdat we in concrete projecten telkens een bepaald Python-onderwerp in het zonnetje zetten, leer je snel

hoe deze programmeertaal in de praktijk werkt. De eerste hoofdstukken kun je in dat geval gebruiken als Python-opstapje, terwijl je gaandeweg vanzelf wel ziet hoe het werkt met variabelen, functiedefinities, inspringen en dergelijke. En hopelijk zul je zien dat Python specifieke toepassingsgebieden heeft waarin het echt uitblinkt.

Daarom hebben we met opzet zaken gekozen die niet altijd makkelijk zijn na te bootsen in andere programmeertalen, maar waar Python juist erg geschikt voor is.

Wat hoef ik niet te weten?

In dit boek gaan we niet in op webdevelopment. Je hoeft zelf geen HTML, CSS of JavaScript te schrijven, of frameworks zoals Vue, React of Angular te kennen. Enige kennis van objectgeoriënteerd programmeren met klassen en overerving is handig, maar niet beslist noodzakelijk.

OOP in Python

Python klopt zichzelf ook op de borst omdat het een *multi paradigm programming language* is. Je kunt in Python dus bijvoorbeeld werken met objectgeoriënteerde (OOP) kenmerken zoals klassen, overerving, polymorfisme en meer. Tegelijkertijd kom je veel kenmerken van functioneel programmeren tegen. Denk aan het doorgeven van functies als parameters voor andere functies, werken met functies als `map()` en `filter()` en meer. In dit boek zullen we de OOP-kenmerken van Python en het sleutelwoord `class` niet gebruiken. Maar het *kan* wel.

Python wordt veel ingezet in wetenschappelijke omgevingen, of voor Big Data-projecten. Toch hoef je geen wiskundige wizard te zijn om dit boek te kunnen gebruiken. Het is handig als je iets weet van logica (zoals `True`, `False` en `And` en `Or`), maar verder is geen wiskundeknobbel vereist. Wel een flinke dosis nieuwsgierigheid.

Waarom een boek?

'Waarom zou ik nog een boek kopen?', wordt ons regelmatig gevraagd. 'Alle informatie is toch al op internet beschikbaar?' Jazeker, dat is beslist waar. Het probleem is echter dat op internet vaak geen *ordening* in die informatie is aangebracht. Als googelt op `python`, vind je honderdduizenden hits. Zelfs als je het trefwoord `tutorial` en eventueel nog een ander trefwoord er aan toevoegt, blijven er talloze pagina's over.

Vaak wordt dan de eerste link aangeklikt en kom je ergens midden in een probleem met één specifiek antwoord (hopelijk) op Stack Overflow terecht. Je hebt dan een oplossing (opnieuw: hopelijk) voor één probleem, maar weet nog niet hoe je dit in je eigen programma's kunt gebruiken.

De laatste tijd zijn de robots die op basis van kunstmatige intelligentie (*Artificial Intelligence*, *AI*) antwoorden genereren erg populair. Denk aan toepassingen als ChatGPT, Dall-E en Copy.ai. Deze zijn zonder meer erg waardevol (en je kunt ze beslist gebruiken in combinatie met dit boek). Je moet echter wel precies weten wát je ze wilt vragen, wil je er goede antwoorden uit krijgen. Dat ontbreekt vaak nog als je een programmeertaal aan het leren bent.

Oplossingen

In dit boek willen we je *complete oplossingen* aanbieden, zodat je niet één specifieke techniek leert, maar direct ziet hoe die techniek nuttig ingezet kan worden in een compleet programma. Hopelijk zie je hoe technieken in combinatie met elkaar worden gebruikt om tot een resultaat te komen. Je ziet *wanneer* je een dictionary gebruikt, en wanneer liever een list van tuples (om maar een voorbeeld te geven). Zo ga je denken in complete toepassingen, in plaats van in losse technieken.

Balans

Zoals we hiervoor beschrijven willen we graag complete oplossingen aanbieden. Tegelijkertijd willen we de projecten niet helemaal dichtmetselen met het afvangen van alle mogelijke uitzonderingen en fouten die kunnen optreden. Dat zou het zicht op de eigenlijke uitwerking van een probleem vertroebelen. Dat is een lastige balans. De code in dit boek is dan vaak ook *geen productiecode*. In echte toepassingen, die ook door anderen worden gebruikt, wil je vaak juist wél veel foutafhandeling, betere meldingen op het scherm enzovoort. Voor het overzicht en de beknoptheid blijft die in dit boek echter vaak achterwege. Maar zorg er voor dat het in je eigen programma's wel goed geregeld is! Want je wilt niet dat de gebruikers van jouw programma met een cryptische melding als in afbeelding 1.5 worden geconfronteerd.

Tot slot weten we dat veel lezers het gewoon prettig vinden om met een boek op de bank te zitten. In een boek blader je makkelijker vooruit en achteruit. Je kunt aantekeningen maken met potlood en markeerstift en je ziet de samenhang van code en uitleg in een programma in een oogopslag. Je hoeft niet heen en terug te scrollen door een lange pagina. Een boek kun je naast het toetsenbord leggen terwijl je aan het werk bent.

Tegelijkertijd is dit natuurlijk vooral een praktijkboek. Door de projecten in dit boek ook als download beschikbaar te maken nodigen we je zeker uit ook achter de pc plaats te nemen en zelf aan de slag te gaan

Programmeren == doen

Het schrijven van eigen programma's – ongeacht in welke taal – leer je niet door een boek te lezen. Dit is misschien een teleurstelling, maar als je dit boek van begin tot einde doorleest en daarna dichtslaat, ben je nog geen programmeur. Je moet het letterlijk *doen*. Programmeren is een ambacht. Je moet de code typen, fouten maken, zoeken naar oplossingen voor die fouten en flexibel omgaan met je programma. Dit boek, in combinatie met de downloads en jouw eigen tijdsinvestering maken dit een totaaloplossing voor het schrijven van praktische Python-programma's.

Hoe zijn de projecten vormgegeven?

Zoals je al hebt gezien is dit boek opgebouwd rondom *projecten*. Elk project heeft een bepaald kerndoel: een programma uitvoeren vanaf de opdrachtregel, een Excel-bestand lezen en schrijven, een API aanroepen en de data verwerken enzovoort. En zoals hiervoor werd uitgelegd, is niet elk project volledig *foolproof*. Dan zou de techniek waar het eigenlijk om gaat te veel ondersneeuwen.

In plaats daarvan hebben we de volgende focuspunten in de projecten opgenomen:

- **Kort** Projecten zijn zo kort mogelijk gehouden. Niemand heeft er baat bij om honderden regels code over te typen. Maar van kopiëren/plakken word je helemaal niks wijzer! We zouden dus graag zien dat je de programmacode in dit boek zelf typt – en de daarbij behorende fouten maakt. Om dat te bereiken zijn de programma's zo kort mogelijk gehouden en komen we snel tot de kern van de zaak. In je eigen programma's geef je natuurlijk meer uitleg, heb je uitgebreidere foutafhandeling enzovoort.
- **Tekstgebaseerd** Met uitzondering van projecten waarbij we het web en de browser gebruiken, zijn de projecten in dit boek tekstgebaseerd. Ook dit is weer gedaan met het oog op eenvoud. Tekst is eenvoudiger dan grafische code. Een programmaregel als `input("Welk bestand wil je openen?")` waarna de gebruiker een bestandsnaam moet typen, is eenvoudiger dan een venster tekenen met een invoerveld of een uitklapmenu voor het gewenste bestand en vervolgens dit bestand uitlezen en verder gebruiken. Nogmaals, in je eigen programma's wil je dat misschien wel, maar voor het doel van dit boek hebben we gekozen voor een tekstgebaseerde benadering.
- **Geen installatie nodig** De programma's in dit boek kun je rechtstreeks uitvoeren. Alle code staat in één bestand met de Python `.py`-extensie, bijvoorbeeld `hello_world.py`. Zo behoud je het overzicht en kun je je code makkelijk online met iemand delen. De enige voorwaarde is dat degene die het programma gebruikt of test, ook Python heeft geïnstalleerd. Meer hierover lees je in de volgende paragraaf. Maar er zijn geen installatiepakketten, wij-

zigingen in het Windows-register of Mac Sleutelhangertoegang (*keychain access*) nodig. Je hoeft er niet aan te denken meerdere losse bestanden te bundelen en te verspreiden en installatie-instructies te schrijven.

- **Eenvoud** *Eenvoud is de sleutel tot begrip*. De programma's zijn bewust eenvoudig gehouden zodat je goed begrijpt waar het om gaat. Om die reden hebben we vaak ook – expres – statements over meerdere regels verspreid om duidelijk te maken waar het om gaat. Niet iedereen is een expert in Python *comprehension*. Het levert wel kortere code op, maar dat is niet per se leesbaarder. Soms zullen we er gebruik van maken (als het ondubbelzinnige, begrijpelijke code oplevert), maar vaak zullen we de code ook gewoon uitschrijven. Kun jij zelf een statement van vier regels samenvatten in één opdracht? Gefeliciteerd. Doe dat vooral in je eigen toepassingen, maar voor de eenvoud en leesbaarheid van de code is daar in dit boek niet altijd voor gekozen.

Analyse van het project

Behalve de code voor het project vind je in de volgende hoofdstukken altijd ook analyse van die code. Anders zouden we net zo goed één lange listing in dit boek kunnen opnemen. De analyses zijn bedoeld om je inzicht te geven in de werking van het programma en geven achtergronden bij modules, programmastatements en de manier waarop Python werkt.

Hints voor verdere uitwerking

Tot slot geven we aan het eind van elk project enkele hints over hoe je het project verder zou kunnen uitbreiden. We vermijden daarbij dooddoeners als 'breng foutcontrole aan in het programma' (behalve natuurlijk als het project ging over foutcontrole), maar proberen dezelfde techniek vanuit een ander standpunt te bekijken. We dagen je kennis uit. Hopelijk ga je zo ook een programmeerprobleem vanuit meerdere oogpunten bezien. Je ziet dan dat er met soms kleine wijzigingen in de code grote veranderingen bereikt kunnen worden.

Een praktijkvoorbeeld – eenvoud

Bekijk bijvoorbeeld de volgende codevoorbeelden. Stel, je wilt een programma schrijven waarin je de gemiddelde lengte van een woord berekent. Want, hoe hoger de gemiddelde woordlengte, hoe ingewikkelder de tekst is waar het om gaat (juridische documenten zijn vooral berucht). Aan de andere kant: hoe korter de gemiddelde woordlengte, hoe beter de tekst door de gemiddelde lezer begrepen zal worden.

Je kunt dan bijvoorbeeld deze code schrijven (opmerking: het teken ➥ aan het begin van een regel betekent dat de code van die en de volgende regel één coderegel vormen – druk dus niet op Enter of Return tijdens het typen):

```
➥ from statistics import mean # mean is een functie uit de
      standaardbibliotheek om het gemiddelde te berekenen.
➥ woorden = "Het was een koude, donkere winteravond en Jochem vroeg
      zich af of hij naar huis zou gaan".split()
  lengtes = [len(woord) for woord in woorden ]
  print("Jouw zin: ", woorden)
  print("Gemiddelde woordlengte: ", round(mean(lengtes), 1))
```

En de uitvoer is zoals in de afbeelding is te zien.

Afbeelding 1.10 *Een kort fragment om de gemiddelde woordlengte van een tekst te berekenen.*

Met de regel

```
lengtes = [len(woord) for woord in woorden ]
```

wordt gebruikgemaakt van *list comprehension*, een typische Python-constructie. Maar met onderstaand codefragment waarin wordt gewerkt met een klassieke `for`-lus en een tussenvariabele (`gemiddelde`) wordt exact hetzelfde bereikt:

```
from statistics import mean
➥ woorden = "Het was een koude, donkere winteravond en Jochem vroeg zich
      af of hij naar huis zou gaan".split()
lengtes = []
for word in woorden:
    lengtes.append(len(word))
gemiddelde = mean(lengtes)
print("Jouw zin: ", woorden)
print("Gemiddelde woordlengte: ", round(gemiddelde, 1))
```

De werking van deze code is exact gelijk maar is voor velen wel beter leesbaar. Sommigen vinden het een nadeel dat je code langer wordt (vier regels versus één regel), maar in veel voorbeelden laten wij de duidelijkheid voorop staan.

Corrigeren voor leestekens

In feite zouden we in deze code nog moeten corrigeren voor komma's en andere leestekens, omdat die formeel gesproken natuurlijk geen deel uitmaken van het woord, terwijl de lengte van het leesteken wel wordt meegeteld. Opnieuw omwille van de eenvoud doen we dat hier niet. Wel nodigen we je van harte uit om de code zodanig uit te breiden dat hiervoor wordt gecorrigeerd.

Oefening: ingewikkelde zinnen

Voortbordurend op de vorige opmerking: een andere leuke oefening zou zijn om te tellen hoeveel komma's of puntkomma's in een zin worden gebruikt. Immers, hoe meer komma's in een zin staan, hoe ingewikkelder de tekst is. Probeer eens om zelf een dergelijk script te schrijven, waarbij je het aantal komma's tussen twee punten telt. Een hoge score (bijvoorbeeld `> 3`) geeft in dat geval aan dat de tekst behoorlijk ingewikkeld is. Een score van `0 – 1` geeft eenvoudiger teksten aan.

Projecten als voorbeelden

Je ziet dat de projecten in dit boek relatief kort en eenvoudig zijn. Je zult ze niet rechtstreeks willen gebruiken in productiecode. Probeer de projecten vooral te zien als *blauwdruk* voor je eigen programma's. Elk project heeft een bepaald leerdoel (aangegeven op de eerste pagina van het project) en beschrijft best practices. Probeer deze te leren en na te streven in je eigen code.

Om een voorbeeld te geven: het project Geboortedatum (zie hoofdstuk 3) berekend een aantal dagen tussen twee datums. Dat is natuurlijk leuk, maar deze vraag zou je ook eenvoudig kunnen googelen. Probeer vooral het complete plaatje te zien. In het project wordt bijvoorbeeld ook uitgelegd hoe je docstrings gebruikt en hoe je argumenten vanaf de opdrachtregel kunt inlezen. Dit zijn aanvullende leerdoelen die je een compleet beeld van een programma geven. Het werken met de (geboorte)datums is in dat geval meer een kapstok om de overige technieken mee te demonstreren.

We gaan die aanvullende code niet in elk project herhalen, maar zelf doe je dit in je code natuurlijk wel.

Python downloaden en installeren

Misschien ben je afkomstig uit een andere programmeertaal en is Python nog niet op jouw computer geïnstalleerd. In dat geval kun je de aanwijzingen in deze paragraaf volgen. Staat Python al wel op jouw systeem, dan kun je deze hele paragraaf overslaan.

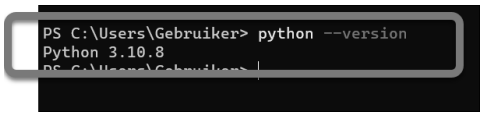
Wel is het een goed idee om even te checken welke versie van Python je gebruikt:

```
# Windows:
C:\Users\Peter> python --version
# Mac/Linux
$ python3 --version
```

Als het goed is krijg je iets te zien als

```
Python 3.10.8
```

Het versienummer op jouw computer kan natuurlijk anders zijn. Wel adviseren we om in ieder geval Python 3 te installeren. De projecten in dit boek zijn niet getest met Python 2.

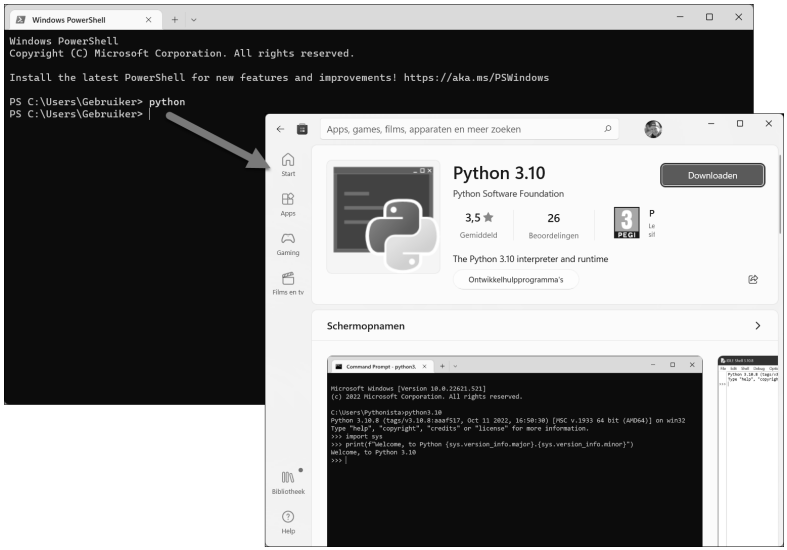


Afbeelding 1.11 *Python 3 is correct geïnstalleerd (in dit geval op een Windows-pc).*

Krijg je een foutmelding, of wordt de Windows Store geopend, installeer dan Python op je computer.

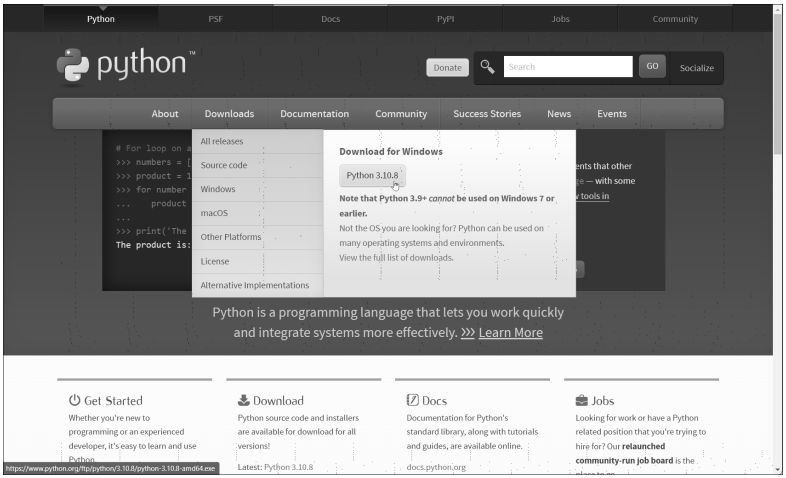
Installatie op Windows

Python wordt niet standaard meegeleverd met Windows. Je moet het apart installeren. Dat kán via de Windows Store, maar dit download- en installatieproces verloopt tergend traag. We adviseren je om via de Python-website te werken.

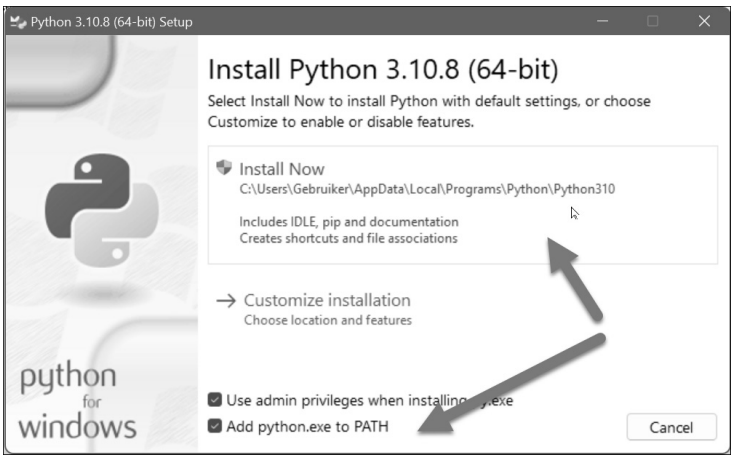


Afbeelding 1.12 *Als Python nog niet is geïnstalleerd op een Windows-pc, probeert Windows je te helpen door de Windows Store te openen. We adviseren echter om te werken via de Python-website. Dit werkt sneller.*

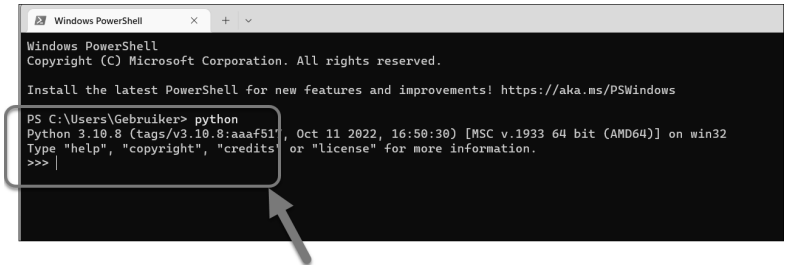
1. Ga naar **python.org** en kies **Downloads** uit het hoofdmenu. De website geeft direct aan of Python voor Windows of voor een ander besturingssysteem de beste keuze is. Klik op de knop **Python 3.x.x** en sla het programma op.
2. Voer de installatiewizard uit. Je kunt alle basisinstellingen overnemen.
3. Denk er wel aan Python toe te voegen het Windows `PATH` door het selectievakje te activeren.
4. In de laatste stap sluit je de wizard met **Close**. Probeer nu nogmaals de opdracht `py` of `python` uit te voeren in een opdrachtregelomgeving (**cmd.exe** of Windows Terminal). Als het goed is ziet het venster er dan uit zoals in afbeelding 1.15.



Afbeelding 1.13 *Download Python via de officiële website. Het versienummer zal in jouw geval natuurlijk anders zijn.*



Afbeelding 1.14 *Doorloop alle stappen van de installatiewizard. Denk er aan Python aan het PATH toe te voegen.*

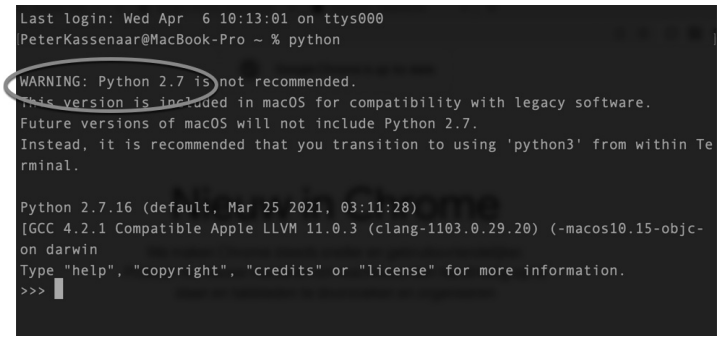


Afbeelding 1.15 *Als je nu de opdracht python typt, verschijnt de Python-shell of REPL. Hierin kun je rechtstreeks opdrachten typen. Verlaat de shell met exit() of quit().*

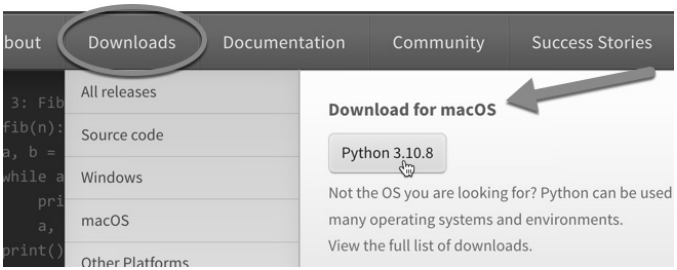
Installatie op Mac/Linux

Op Apple-computers wordt wel standaard Python geïnstalleerd. Maar let op: vaak is dit een oudere versie. Je kunt dan zelf de nieuwste Python 3.x-versie installeren. Check dit als volgt:

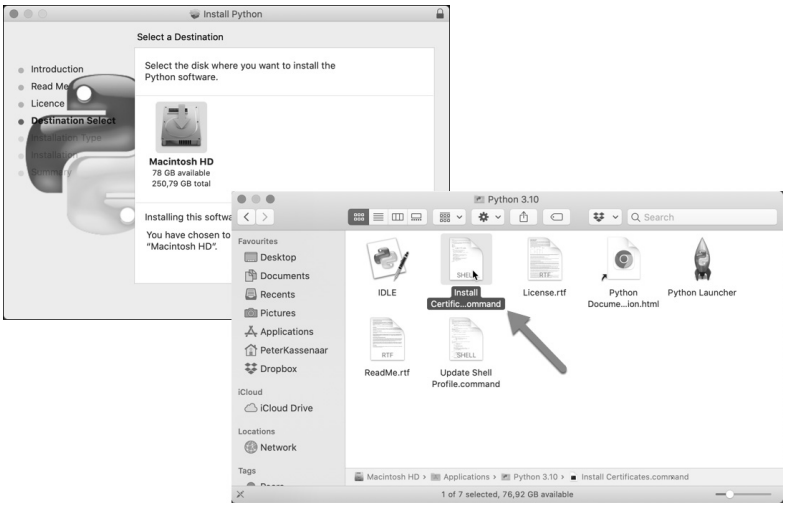
1. Typ `python` in een Terminal-venster. Als je een venster ziet als in afbeelding 1.16, moet je Python upgraden.
2. Python 3 kan gewoon naast Python 2 worden geïnstalleerd. Ga naar de Python-website op **python.org** en kies de koppeling **Downloads**.
3. Ook nu verloopt de installatie via een wizard en kun je de standaardinstellingen overnemen.
4. Let op: op macOS moet je na afloop van de installatie de juiste certificaten installeren. Dit is een eenmalig klusje, maar moet wel even gebeuren. Dubbelklik op het bestand **Install Certificates Command** in de Python-map.
5. Het downloaden en installeren van de certificaten verloopt vanzelf, maar moet je zoals gezegd niet vergeten. Het venster kan er zo uitzien als in afbeelding 1.19.



Afbeelding 1.16 *Let goed op: op Apple-computers wordt vaak een oudere versie van Python meegeleverd. Installeer in dat geval zelf de nieuwste versie.*



Afbeelding 1.17 *Download de versie voor Mac of Linux, afhankelijk van wat op de website wordt aangegeven.*



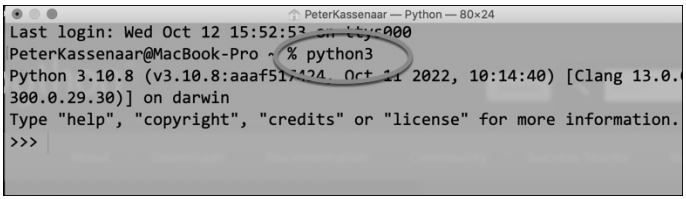
Afbeelding 1.18 *Doorloop de installatiewizard, denk er aan na afloop de juiste certificaten te installeren. Dit is een eenmalig klusje.*

```
Last login: Wed Oct 12 15:40:15 on ttys000
/Applications/Python\ 3.10/Install\ Certificates.command ; exit;
PeterKassenaar@MacBook-Pro ~ % /Applications/Python\ 3.10/Install\ Certificates
command ; exit;
 -- pip install --upgrade certifi
Collecting certifi
  Downloading certifi-2022.9.24-py3-none-any.whl (161 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 161.1/161.1 kB 5.2 MB/s eta 0:00:0
Installing collected packages: certifi
Successfully installed certifi-2022.9.24
 -- removing any existing file or link
 -- creating symlink to certifi certificate bundle
 -- setting permissions
 -- update complete

[Process completed]
```

Afbeelding 1.19 *De Mac-certificaten worden geïnstalleerd.*

Na afloop van de installatie op Mac of Linux kun je in een terminalvenster de opdracht `python3` typen. Als het goed is wordt dan de prompt >>> (drie groter-dantekens) getoond en kun je eventueel rechtstreeks Python-opdrachten typen. Deze shell kun je beëindigen met `exit()` of `quit()`.



Afbeelding 1.20 *De installatie van Python 3 op Mac is geslaagd.*

Let er op dat je op Mac de opdracht `python3` gebruikt om een programma te starten. Als je 'gewoon' de opdracht `python` gebruikt, zoals op Windows, dan wordt de oude versie gebruikt (tenminste, als die op jouw systeem aanwezig was. Het *hoeft* niet. Check dit zelf op je eigen computer).

Problemen met installeren?

Mocht de installatie niet lukken, kijk dan bijvoorbeeld op:

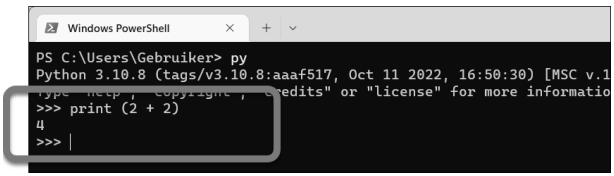
- **wiki.python.org/moin/BeginnersGuide/Download**
- **realpython.com/installing-python/**

Afbeelding 1.21 *De installatie van Python is in het algemeen eenvoudig, maar als je er niet uitkomt zijn er websites die je verder op weg helpen.*

Een editor kiezen

Zoals je al zag in afbeelding 1.15 (Windows) en 1.20 (Mac) kun je eventueel de Python-shell gebruiken om rechtstreeks opdrachten te typen of zelfs complete modules te importeren en programma's te schrijven. Wij doen dan hier echter niet.

Zodra je immers de shell afsluit met `exit()` of `quit()` is je programma weer verdwenen. Er zijn overigens ook sneltoetsen om de Python-shell af te sluiten: Ctrl+Z, Enter op Windows of Ctrl+D op Mac/Linux.



Afbeelding 1.22 *Eenvoudige opdrachten rechtstreeks in de Python-shell.*

Andere naam

In feite kun je in de Python-shell veel meer dan alleen eenvoudige opdrachten geven. Als je wit kun je packages importeren, help opvragen, met de pijltoetsen vorige opdrachten nogmaals uitvoeren en nog veel meer. Andere namen voor de Python-shell zijn ook wel REPL (van *Read*, *Evaluate*, *Print*, *Loop*) of IDLE (van *Integrated Development and Learning Environment*). Het is zeker nuttig, maar in dit boek maken we er geen gebruik van.

In plaats daarvan adviseren wij om een editor te gebruiken waarmee je bestanden kunt maken en uitvoeren. Je slaat ze op met de `.py`-extensie, zodat je ze later nogmaals kunt uitvoeren, kunt aanpassen en uitbreiden, naar een vriend of docent kunt mailen enzovoort.

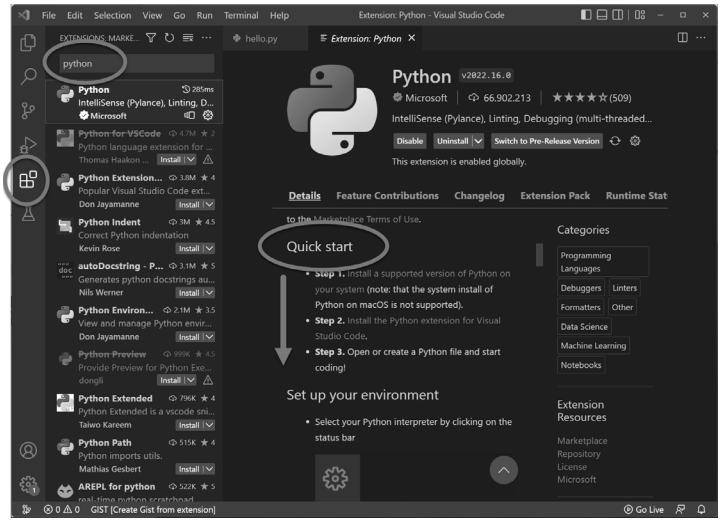
Er zijn tal van editors beschikbaar die geschikt zijn voor het programmeren in Python. Veel ervan zijn gratis. Handige mogelijkheden van editors zijn bijvoorbeeld automatisch aanvullen (*autocomplete*), kleurcodering van opdrachten en functies en automatische matching van haakjes en aanhalingstekens.

Visual Studio Code

Erg populair is de opensource-editor Visual Studio Code, van Microsoft. Deze is gratis te gebruiken en beschikbaar voor Windows, Mac en Linux. In combinatie met de extensie **Python** is dit een uitstekende keuze.

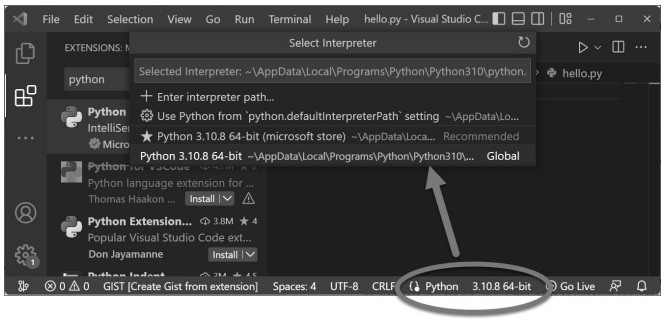
- Je vindt Visual Studio Code op **code.visualstudio.com**.

- Gebruik binnen VS Code het menu **Extensions** en zoek vervolgens op `Python`. Dit toont de extensie met aanvullingen waardoor je perfecte Python-scripts kunt schrijven in VS Code.
- Denk er aan om de Quick start te lezen. Veel programmeurs slaan dit over ('direct aan het werk!'), maar er staan nuttige tips en sneltoetsen in die het werken met Python in VS Code een stuk versnellen en vergemakkelijken.



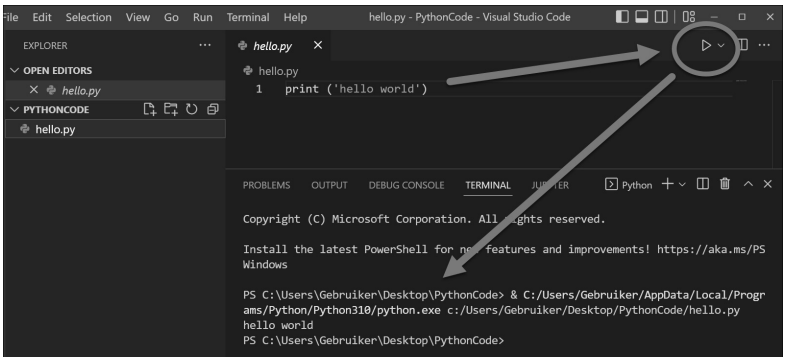
Afbeelding 1.23 *Visual Studio Code in combinatie met de Python-extensie is een uitstekende keuze als editor.*

Na installatie van de Python-extensie moet je VS Code laten weten welke Python-interpreter gebruikt wordt. Dit doe je door in het palet (Ctrl+Shift+P) de opdracht **Select Interpreter** te kiezen. Je opent deze opdracht snel door (als een `.py`-bestand geopend is) op het versienummer of de naam Python in de statusbalk te klikken.



Afbeelding 1.24 *Laat VS Code weten welke versie van Python je gebruikt en waar deze geïnstalleerd is. Dan kun je snel Python-programma's starten vanuit de editor.*

Als de extensie en Python-versie in VS Code goed zijn ingesteld, kun je testen of alles goed werkt door een eenvoudig Python-bestand te maken met één of twee regels code en deze uitvoeren vanuit de editor. Afbeelding 1.25 laat dit zien.



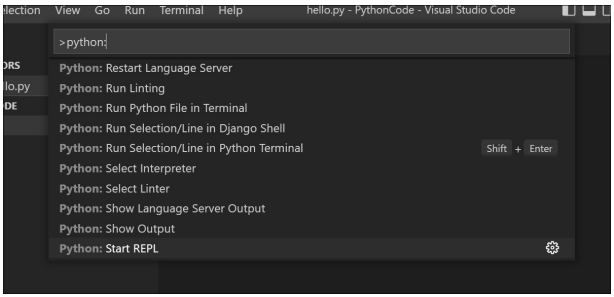
Afbeelding 1.25 *Maak een eenvoudig Python-bestand en voer dit uit. De knop wordt toegevoegd door de extensie.*

Uiteraard kun je ook met sneltoetsen je Python-programma starten:

- **Ctrl+F5** Draai het Python-programma rechtstreeks in de terminal onder in het venster van VS Code.
- **F5** Start een debugsessie. Je kunt breekpunten plaatsen in de code en VS Code stopt wanneer de uitvoering van het programma bij een breekpunt is aangekomen. Later in dit hoofdstuk bespreken we het debuggen van programma's.

Met de extensie zijn allerlei extra Python-opdrachten aan VS Code toegevoegd. Je ziet ze eventueel door weer het palet te openen (Ctrl+Shift+P) en de eerste letters van het woord **Python** te typen. In een uitklapmenu worden alle mogelijkheden getoond.

VS Code is daarmee een zeer complete Python-ontwikkelomgeving geworden.



Afbeelding 1.26 *Met de extensie zijn allerlei Python-mogelijkheden aan VS Code toegevoegd. In dit boek gebruiken we ze niet, maar het is goed om te weten wat er mogelijk is.*

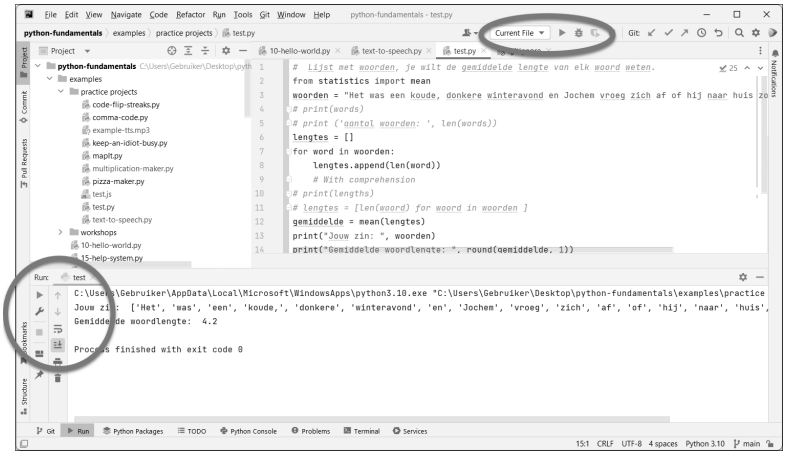
PyCharm

PyCharm is een editor van JetBrains (**www.jetbrains.com**). Als je een Java- of webdevelopmentachtergrond hebt, ben je misschien bekend met IntelliJ of WebStorm. PyCharm is de versie van IntelliJ die is geoptimaliseerd voor Python-ontwikkeling.

PyCharm is eveneens beschikbaar voor Windows, MacOS en Linux en is verkrijgbaar in twee versies:

- PyCharm Community, deze is gratis.
- PyCharm Professional, dit is een betaald product.

Het voordeel/kenmerk van JetBrains-editors is dat ze na installatie direct alles bevatten wat je nodig hebt. Je hoeft niet te zoeken naar aanvullende plug-ins of extensies, alles is aanwezig. Het nadeel is natuurlijk dat het niet gratis is. Voor professioneel gebruik is PyCharm echter beslist een goede keuze. De aanschaf- en abonnementskosten heb je er snel uit. Voor hobbymatig gebruik is PyCharm Community een betere keuze.



Afbeelding 1.27 *PyCharm biedt out-of-the-box alles wat de professionele Python-ontwikkelaar nodig heeft.*

Bekijk zelf de mogelijkheden op **www.jetbrains.com/pycharm/download/**.

Overige editors

Behalve Visual Studio Code en PyCharm zijn er nog tal van andere Python-editors beschikbaar. Als de hiervoor genoemde editors je niet bevallen, kijk dan bijvoorbeeld eens naar:

- **Mu Editor** Beschikbaar op **codewith.mu**. Dit is een eenvoudige Python-editor waarin je niet wordt afgeleid door talloze extensies of werkbalken die je misschien toch nooit zult gebruiken.

- **Notepad++** Beschikbaar op **notepad-plus-plus.org**. Met de aanvullende Python-plug-in `NppExec` kun je ook direct vanuit Notepad++ je Python-programma's starten.
- **Atom** Beschikbaar op **atom.io**. Ook dit is een uitgebreide editor waarvoor een speciale Python-plug-in beschikbaar is. Kijk eventueel op **atom.io/packages/ide-python** als Atom jouw keuze is.

Het maakt ons totaal niet uit welke editor jij gebruikt. Als je er maar code mee kunt typen en bestanden kunt opslaan als `.py`-bestanden. Misschien werk jij graag in Vi, Vim of Kladblok. Wij vinden het prima. Uiteindelijk is een Python-programma gewoon uitvoerbare code geschreven als platte tekst. In een terminalvenster kun je altijd 'los' je programma uitvoeren met de opdracht `py`, `python` of `python3` (afhankelijk van je besturingssysteem. Dit herhalen we vanaf nu niet telkens).

Afbeelding 1.28 *Een Python-programma 'los' uitgevoerd, buiten een editor. Dit doe je gewoon in een opdrachtregelomgeving zoals cmd.exe, PowerShell of Terminal.*

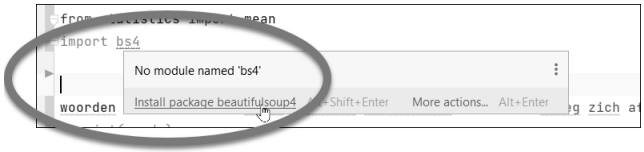
Let er alleen op dat je je code niet opslaat in een binair bestandsformaat, zoals `.docx` van Word. Dat gaat niet werken.

Aanvullende Python-modules installeren

In het korte programmavoorbeeld dat we hiervoor gebruikten, importeerden we de module `mean`. Hiermee konden we snel het gemiddelde uitrekenen van een lijst met getallen. De module `mean` is aanwezig in de standaardbibliotheek `statistics`. We hoeven die niet apart te installeren.

Installeren versus importeren

Als een module is geïnstalleerd (via pip), wil dat nog niet zeggen dat hij ook automatisch wordt geïmporteerd in een programma. Dit moet je altijd zelf doen met de opdracht `import`. Mocht een module niet aanwezig zijn, dan geeft je editor daar hopelijk een foutmelding over. En anders merk je het vanzelf bij het uitvoeren van het programma. Dan wordt namelijk een foutmelding getoond.



Afbeelding 1.29 *PyCharm heeft gezien dat het pakket bs4 dat we willen importeren in ons programma nog niet is geïnstalleerd en biedt aan dit te installeren. Op de achtergrond wordt hiervoor pip gebruikt.*

Python package manager

Voor het installeren van externe modules (ook wel *packages* of pakketten genoemd), wordt pip gebruikt. Pip is de package manager voor Python. Het wordt automatisch geïnstalleerd als je Python op je systeem zet.

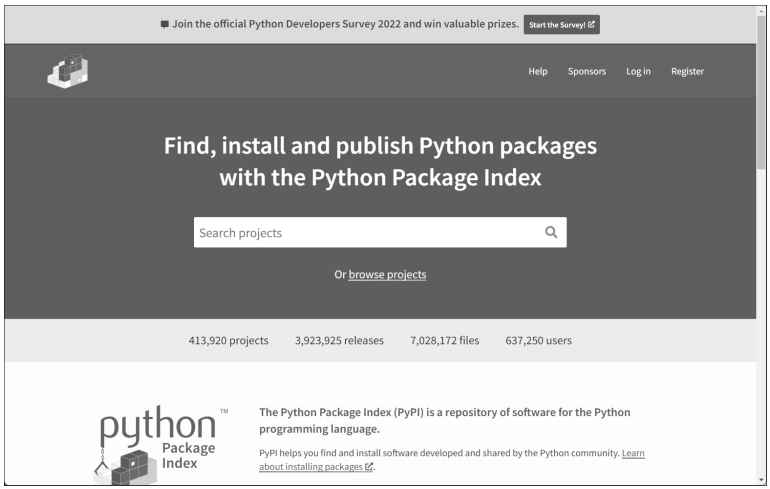
Om bijvoorbeeld `bs4` te installeren (een package die we gaan gebruiken om HTML te parsen in het project met webscraping, zie hoofdstuk 11) geef je de volgende opdracht:

```
pip install bs4
```

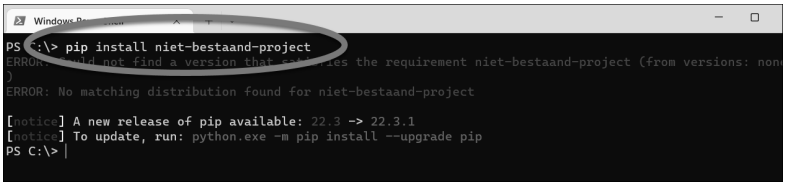
Maar in plaats van `bs4` kun je natuurlijk elke geldige packagenaam gebruiken. Als dit niet werkt (pip kan bijvoorbeeld niet gevonden worden), probeer dan of je Python kunt starten met de aanvullende opdracht `-m` (dit staat voor module) en geef daarna pip op. Bijvoorbeeld:

```
python -m pip install <package-naam>
```

Pip kijkt naar de *Python Package Index*, pypi op **pypi.org**. De desbetreffende module wordt daar opgehaald en op je systeem geïnstalleerd. Als je een spelfout maakt in de packagenaam krijg je hiervan vanzelf een melding.



Afbeelding 1.30 *Pip installeert packages vanaf de Python Package Index…*



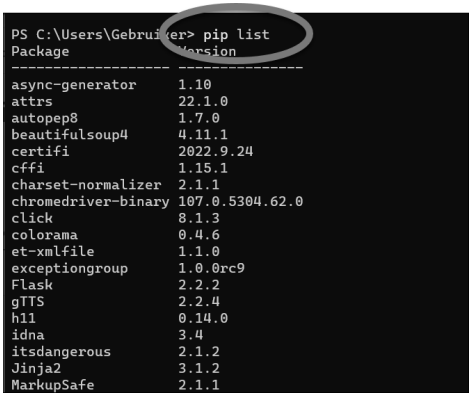
Afbeelding 1.31 *…maar je moet natuurlijk wel een geldige naam opgeven (en geen typefouten maken). Anders zie je een foutmelding zoals deze.*

Welke packages zijn al geïnstalleerd?

Pip kan ook een totaaloverzicht geven van alle packages die al geïnstalleerd zijn. Gebruik hiervoor de opdracht

```
pip list
```

Je ziet dan een alfabetisch overzicht van alle geïnstalleerde packages en hun versienummer. Deze hoeven maar eenmalig, centraal op het systeem te worden geïnstalleerd. Er is dus niet zoals bij Node.js een map /node_modules waar de afhankelijkheden per project in worden geïnstalleerd. Python lijkt in dit geval meer op Java en .NET.



Afbeelding 1.32 *De opdracht pip list geeft een overzicht van geïnstalleerde packages en hun versienummers. Pip zelf staat hier ook tussen!*

In de projecten van de volgende hoofdstukken geven we het altijd aan als we externe packages gebruiken. We geven ook aan welke opdracht nodig is om de package te installeren.

Voor meer achtergronden bij het installeren van packages en de talloze mogelijkheden hierbij verwijzen we naar de Pypi-documentatie, beschikbaar op **packaging.python.org/en/latest/tutorials/installing-packages**. Je vindt deze ook via een koppeling op de homepage van Pypi. Klik hiervoor op **Learn about installing packages**.