INHOUD

DANKBETUIGING **25**

INLEIDING **27**

Voor wie is dit boek bedoeld? 28
Wat kun je verwachten te gaan leren? 28
Waarom Python? 29

DEEL I: DE BASIS 31

1
AAN DE SLAG 33

Het instellen van je programmeeromgeving 33
Python 2 en Python 3 34
Fragmenten Python-code uitvoeren 34
Hello World! 34
Python op verschillende besturingssystemen 35
Python op Linux 35
Python op macOS 38
Python op Windows 41
Problemen bij installatie oplossen 45
Python-programma's vanuit een terminal uitvoeren 46
Op Linux en macOS 46
Op Windows 47
Oefening 1-1. python.org *47*
Oefening 1-2. Typefouten in Hello World *47*
Oefening 1-3. Oneindige vaardigheden *48*
Samenvatting 48

2
VARIABELEN EN EENVOUDIGE DATATYPES 49

Wat er daadwerkelijk gebeurt als je hello_world.py uitvoert 49
Variabelen 50
Variabelen benoemen en gebruiken 51
Foutmeldingen in namen voorkomen als je variabelen gebruikt 52
Oefening 2-1. Simple Message *53*
Oefening 2-2. Simple Messages *53*
Strings 53
Wisselen tussen hoofd- en kleine letters in een string met methoden 54

Combineren of samenvoegen van strings . . . 55
Witruimte toevoegen aan strings met tabs of nieuwe regels . . . 56
Witruimte weghalen . . . 56
Fouten in de syntaxis met strings voorkomen . . . 58
Afdrukken in Python 2 . . . 59
Oefening 2-3. Persoonlijk bericht . . . *59*
Oefening 2-4. Namen in hoofd- en kleine letters . . . *59*
Oefening 2-5. Beroemde quote . . . *59*
Oefening 2-6. Beroemde quote 2 . . . *59*
Oefening 2-7. Witruimte bij namen weghalen. . . . *59*
Getallen . . . 60
Gehele getallen . . . 60
Floating point-getallen . . . 61
Fouten in types vermijden met de functie str() . . . 61
Gehele getallen in Python 2 . . . 62
Oefening 2-8. Het getal acht . . . *63*
Oefening 2-9. Lievelingsgetal. . . . *63*
Opmerkingen . . . 63
Hoe schrijf je opmerkingen? . . . 64
Wat voor opmerkingen voeg je toe? . . . 64
Oefening 2-10. Opmerkingen toevoegen. . . . *65*
The Zen of Python . . . 65
Oefening 2-11. Zen of Python. . . . *67*
Samenvatting . . . 67

3
INTRODUCTIE VAN LIJSTEN **69**

Wat is een lijst? . . . 69
Elementen in een lijst benaderen . . . 70
Indexposities beginnen bij 0, niet bij 1 . . . 71
Individuele waarden uit een lijst gebruiken . . . 71
Oefening 3-1. Namen . . . *72*
Oefening 3-2. Begroeting . . . *72*
Oefening 3-3. Je eigen lijst . . . *72*
Elementen wijzigen, toevoegen en verwijderen . . . 72
Elementen in een lijst aanpassen . . . 72
Elementen aan een lijst toevoegen . . . 73
Elementen uit een lijst verwijderen . . . 74
Oefening 3-4. Gastenlijst . . . *78*
Oefening 3-5. Gastenlijst veranderen . . . *78*
Oefening 3-6. Meer gasten . . . *78*
Oefening 3-7. Gastenlijst verkleinen . . . *79*
Een lijst organiseren . . . 79
Een lijst permanent sorteren met de sort()-methode . . . 79
Een lijst tijdelijk sorteren met de sorted()-functie . . . 80
Een lijst in omgekeerde volgorde afdrukken . . . 81
De lengte van een lijst bepalen . . . 81
Oefening 3-8. De wereld zien . . . *82*
Oefening 3-9. Gasten voor een etentje . . . *82*
Oefening 3-10. Elke functie . . . *82*

Index-fouten voorkomen wanneer je met lijsten werkt 82
Oefening 3-11. Opzettelijke fout 84
Samenvatting 84

4
WERKEN MET LIJSTEN 85

Door een hele lijst lussen 85
De lus nader bekeken 86
Meer gedaan krijgen in een for-lus 87
Iets doen na een for-lus 88
Fouten met inspringen voorkomen 89
Vergeten om in te springen 90
Vergeten om extra regels in te laten springen 90
Onnodig inspringen 91
Onnodig inspringen na de lus 91
De dubbele punt vergeten 92
Oefening 4-1. Pizza's 92
Oefening 4-2. Dieren 93
Numerieke lijsten maken 93
De range()-functie gebruiken 93
De range()-functie gebruiken om een lijst met getallen te maken 94
Eenvoudige statistieken met getallenlijsten 95
Lijstcomprehensies 96
Oefening 4-3. Tot twintig tellen 96
Oefening 4-4. Eén miljoen 96
Oefening 4-5. Eén miljoen optellen 97
Oefening 4-6. Oneven getallen 97
Oefening 4-7. Veelvouden van 3 97
Oefening 4-8. Tot de derde macht 97
Oefening 4-9. Tot de derde macht met lijstcomprehensie 97
Werken met een deel van een lijst 97
Een lijst splitsen 97
Door een splitsing lussen 99
Een lijst kopiëren 99
Oefening 4-10. Splitsingen 101
Oefening 4-11. Mijn pizza's, jouw pizza's 102
Oefening 4-12. Meer lussen 102
Tupels 102
Een tupel definiëren 102
Door alle waarden in een tupel lussen 103
Een tupel overschrijven 103
Oefening 4-13. Buffet 104
Je code opmaken 104
De stijlgids 105
Inspringen 105
Lengte van regels 105
Witregels 106
Andere stijlregels 106
Oefening 4-14. Pep 8 107
Oefening 4-15. Code herzien 107
Samenvatting 107

5
IF-INSTRUCTIES **109**

Een eenvoudig voorbeeld 110
Voorwaardelijke tests 110
Controleren op gelijkheid 110
Hoofd- en kleine letters negeren als je controleert op gelijkheid 111
Controleren op ongelijkheid 112
Numerieke vergelijkingen 113
Meerdere voorwaarden controleren 113
Controleren of een waarde in een lijst staat 115
Controleren of een waarde niet in een lijst staat 115
Booleaanse uitdrukkingen 116
Oefening 5-1. Voorwaardelijke tests *116*
Oefening 5-2. Meer voorwaardelijke tests *116*
If-Instructies 117
Eenvoudige if-instructies 117
if-else-instructies 118
De if-elif-else-keten 118
Meerdere elif-blokken gebruiken 120
Het else-blok weglaten 120
Meerdere voorwaarden testen 121
Oefening 5-3. Alien Colors #1 . . . . . *123*
Oefening 5-4. Alien Colors #2 . . . . . *123*
Oefening 5-5. Alien Colors #3 . . . . . *123*
Oefening 5-6. Levensfasen *123*
Oefening 5-7. Lievelingsfruit *124*
If-instructies gebruiken bij lijsten 124
Controleren op bijzondere items 124
Controleren of een lijst niet leeg is 125
Meerdere lijsten gebruiken 126
Oefening 5-8. Hallo beheerder *127*
Oefening 5-9. Geen gebruikers *127*
Oefening 5-10. Gebruikersnamen controleren *127*
Oefening 5-11. Rangtelwoorden *128*
Je if-instructies opmaken 128
Oefening 5-12. Opmaken van if-instructies *129*
Oefening 5-13. Jouw ideeën *129*
Samenvatting 129

6
WOORDENBOEKEN **131**

Een eenvoudig woordenboek 132
Werken met woordenboeken 132
Waarden in een woordenboek benaderen 133
Nieuwe sleutel-waarde-paren toevoegen 134
Met een leeg woordenboek beginnen 134
Waarden in een woordenboek aanpassen 135
Sleutel-waarde-paren verwijderen 136
Een woordenboek van soortgelijke objecten 137
Oefening 6-1. Persoon *139*
Oefening 6-2. Geluksgetallen *139*

Oefening 6-3. Woordenlijst *139*
Door een woordenboek lussen 139
Door alle sleutel-waarde-paren lussen 139
Door alle sleutels in een woordenboek lussen 141
Op volgorde door een woordenboek lussen 143
Door alle waarden in een woordenboek lussen 144
Oefening 6-4. Woordenlijst 2 . . . . . *145*
Oefening 6-5. Rivieren *146*
Oefening 6-6. Enquête *146*
Nesten 146
Een lijst met woordenboeken 146
Een lijst in een woordenboek 149
Een woordenboek in een woordenboek 151
Oefening 6-7. Personen. *152*
Oefening 6-8. Huisdieren *152*
Oefening 6-9. Favoriete plaatsen *153*
Oefening 6-10. Favoriete getallen *153*
Oefening 6-11. Steden *153*
Oefening 6-12. Uitbreidingen *153*
Samenvatting 153

7
INVOER DOOR DE GEBRUIKER EN WHILE-LUSSEN 155

Hoe de input()-functie werkt 156
Duidelijke prompts schrijven 156
int() gebruiken om numerieke invoer te accepteren 157
De modulo operator 159
Invoer accepteren in Python 2.7 159
Oefening 7-1. Huurauto. *160*
Oefening 7-2. Een tafeltje in een restaurant *160*
Oefening 7-3. Veelvoud van tien *160*
Een introductie van while-lussen 160
De while-lus in actie. 160
De gebruiker het moment van afsluiten laten kiezen. 161
Een vlag gebruiken 162
Break gebruiken om een lus af te sluiten. 164
Continue gebruiken in een lus 164
Oneindige lussen vermijden 165
Oefening 7-4. Pizza toppings *166*
Oefening 7-5. Filmkaartjes *166*
Oefening 7-6. Drie manieren om af te sluiten. *166*
Oefening 7-7. Oneindigheid *166*
Een while-lus gebruiken bij lijsten en woordenboeken 167
Items van de ene lijst naar de andere verplaatsen 167
Alle exemplaren van bepaalde waarden uit een lijst verwijderen 168
Een woordenboek vullen met invoer van een gebruiker 169
Oefening 7-8. Lunchroom *170*
Oefening 7-9. Geen pastrami *170*
Oefening 7-10. Droomvakantie *170*
Samenvatting 170

8
FUNCTIES **171**

Een functie definiëren 172
Informatie aan een functie doorgeven 172
Argumenten en parameters 173
Oefening 8-1. Bericht *173*
Oefening 8-2. Favoriete boek *174*
Argumenten doorgeven 174
Positionele argumenten 174
Trefwoordargumenten 176
Standaardwaarden 177
Gelijkwaardige functieaanroepen 178
Fouten met argumenten voorkomen 179
Oefening 8-3. T-shirt *179*
Oefening 8-4. Grote shirts *180*
Oefening 8-5. Steden *180*
Retourwaarden 180
Een eenvoudige waarde retourneren 180
Een argument optioneel maken 181
Een woordenboek retourneren 183
Een functie met een while-lus gebruiken 184
Oefening 8-6. Namen van steden *185*
Oefening 8-7. Album *185*
Oefening 8-8. Albums van gebruikers *185*
Een lijst doorgeven 186
Een lijst in een functie aanpassen 186
Voorkomen dat een functie een lijst aanpast 189
Oefening 8-9. Goochelaars *190*
Oefening 8-10. Grote goochelaars *190*
Oefening 8-11. Ongewijzigde goochelaars *190*
Een willekeurig aantal argumenten doorgeven 190
Combineren van positionele en willekeurige argumenten 191
Willekeurige trefwoordargumenten gebruiken 192
Oefening 8-12. Sandwiches *193*
Oefening 8-13. Gebruikersprofiel *193*
Oefening 8-14. Auto's *193*
Je functies in modules opslaan 194
Een hele module importeren 194
Specifieke functies importeren 195
'As' gebruiken om een functie een alias te geven 196
'As' gebruiken om een module een alias te geven 196
Alle functies in een module importeren 197
Functies opmaken 197
Oefening 8-15. Modellen afdrukken *198*
Oefening 8-16. Importeren *198*
Oefening 8-17. Functies opmaken *198*
Samenvatting 199

9
CLASSES **201**

Een class maken en gebruiken . . . 202
Het maken van de class 'Dog' . . . 202
Een exemplaar van een class maken . . . 204
Oefening 9-1. Restaurant . . . *207*
Oefening 9-2. Drie restaurants . . . *207*
Oefening 9-3. Gebruikers . . . *207*
Werken met classes en exemplaren . . . 207
De 'Car'-class . . . 207
Een standaardwaarde voor een attribuut instellen . . . 208
De waarden van attributen aanpassen . . . 209
Oefening 9-4. Aantal bediende gasten . . . *212*
Oefening 9-5. Inlogpogingen . . . *212*
Overerven . . . 213
De __init__()-methode voor een child class . . . 213
Overerven in Python 2.7 . . . 214
Attributen en methoden definiëren voor de child class . . . 215
Methoden uit de parent class overschrijven . . . 216
Exemplaren als attributen . . . 216
Objecten uit de echte wereld modelleren . . . 219
Oefening 9-6. De ijscoman . . . *219*
Oefening 9-7. Beheerder . . . *219*
Oefening 9-8. Privileges . . . *220*
Oefening 9-9. Accu-upgrade . . . *220*
Classes importeren . . . 220
Een enkele class importeren . . . 220
Meerdere classes in een module opslaan . . . 222
Meerdere classes uit een module importeren . . . 223
Een hele module importeren . . . 223
Alle classes uit een module importeren . . . 224
Een module in een module importeren . . . 224
Je eigen workflow vinden . . . 225
Oefening 9-10. Geïmporteerd restaurant . . . *226*
Oefening 9-11. Geïmporteerde beheerder . . . *226*
Oefening 9-12. Meerdere modules . . . *226*
De standaardbibliotheek van Python . . . 226
Oefening 9-13. OrderedDict herschrijven . . . *227*
Oefening 9-14. Dobbelsteen . . . *227*
Oefening 9-15. Python Module of the Week . . . *228*
Classes opmaken . . . 228
Samenvatting . . . 229

10
BESTANDEN EN UITZONDERINGEN **231**

Een bestand lezen . . . 232
Een heel bestand lezen . . . 232
Bestandspaden . . . 234
Regel voor regel lezen . . . 235
Van een bestand een lijst met regels maken . . . 236
Werken met de inhoud van een bestand . . . 237

Grote bestanden: een miljoen cijfers . . . 238
Staat jouw verjaardag in pi? . . . 238
Oefening 10-1. Python leren . . . *239*
Oefening 10-2. C leren . . . *239*
Naar een bestand schrijven . . . 240
Naar een leeg bestand schrijven . . . 240
Meerdere regels schrijven . . . 241
Iets aan een bestand toevoegen . . . 241
Oefening 10-3. Gast . . . *242*
Oefening 10-4. Gastenboek . . . *242*
Oefening 10-5. Enquête over programmeren . . . *242*
Uitzonderingen . . . 242
Afhandelen van de ZeroDivisionError-uitzondering . . . 243
Try-except-blokken gebruiken . . . 243
Uitzonderingen gebruiken om crashes te voorkomen . . . 244
Het else-blok . . . 245
Afhandelen van de FileNotFoundError-uitzondering . . . 246
Tekst analyseren . . . 247
Met meerdere bestanden werken . . . 248
Geruisloos falen . . . 249
Beslissen welke fouten gemeld worden . . . 250
Oefening 10-6. Optellen . . . *251*
Oefening 10-7. Optelcalculator . . . *251*
Oefening 10-8. Cats and Dogs . . . *251*
Oefening 10-9. Cats and Dogs (in stilte) . . . *251*
Oefening 10-10. Bekende woorden . . . *251*
Data opslaan . . . 252
json.dump() en json.load() gebruiken . . . 252
Opslaan en lezen van door de gebruiker gegenereerde data . . . 253
Opschonen . . . 255
Oefening 10-11. Favoriete getal . . . *257*
Oefening 10-12. Favoriete getal onthouden . . . *257*
Oefening 10-13. Gebruiker verifiëren . . . *258*
Samenvatting . . . 258

11
JE CODE TESTEN **259**

Een functie testen . . . 259
Unit tests en testcases . . . 261
Een geslaagde test . . . 261
Een falende test . . . 263
Reageren op een gefaalde test . . . 264
Nieuwe tests toevoegen . . . 265
Oefening 11-1. Stad, land . . . *266*
Oefening 11-2. Bevolking . . . *266*
Het testen van een class . . . 266
Een diversiteit aan assert-methoden . . . 266
Een class om te testen . . . 267
Het testen van de AnonymousSurvey-class . . . 269
De setUp()-methode . . . 271
Oefening 11-3. Medewerker . . . *272*
Samenvatting . . . 272

DEEL II: PROJECTEN 275

PROJECT 1: ALIEN INVASION 277

12
EEN SCHIP DAT KOGELS AFVUURT 279

Je project plannen 280
Pygame installeren 280
Python-pakketten installeren met pip 281
Pygame installeren op Linux 283
Pygame installeren op macOS 284
Pygame installeren op Windows 285
Beginnen met het spelproject 285
Een Pygame-venster maken en reageren op gebruikersinvoer 285
De achtergrondkleur instellen 287
Een instellingen-class maken 287
De afbeelding van het schip toevoegen 288
Het maken van de class 'Ship' 290
Het schip op het scherm tekenen 291
Opschonen: de game_functions-module 292
De check_events()-functie 292
De update_screen()-functie 293
Oefening 12-1. Blauwe lucht *294*
Oefening 12-2. Spelpersonage *294*
Het schip besturen 294
Reageren op het indrukken van een toets 294
Continue beweging mogelijk maken 295
Zowel naar links als naar rechts bewegen 297
De snelheid van het schip aanpassen 298
Het bereik van het schip beperken 300
Opschonen van check_events() 301
Kort samengevat 301
alien_invasion.py 302
settings.py 302
game_functions.py 302
ship.py 302
Oefening 12-3. Raket *302*
Oefening 12-4. Toetsen *302*
Kogels afvuren 303
De instellingen voor de kogel toevoegen 303
Het maken van de Bullet-class 303
Kogels in een groep opslaan 305
Kogels afvuren 306
Oude kogels verwijderen 307
Het aantal kogels beperken 308
De update_bullets()-functie maken 309
De fire_bullet()-functie maken 309
Oefening 12-5. Zijwaarts schieten *310*
Samenvatting 310

13
BUITENAARDSE WEZENS! 311

Je project doornemen 312
Het eerste buitenaardse wezen maken 313
Het maken van de Alien-class 313
Een exemplaar van het buitenaardse wezen maken 314
Een buitenaards wezen op het scherm laten verschijnen 315
Een hele vloot buitenaardse wezens bouwen 316
Bepalen hoeveel buitenaardse wezens er op een rij passen 316
Rijen met buitenaardse wezens maken 316
De vloot maken 317
Opschonen van create_fleet() 319
Rijen toevoegen 320
Oefening 13-1. Sterren 322
Oefening 13-2. Betere sterren 322
De vloot laten bewegen 323
De buitenaardse wezens naar rechts laten bewegen 323
Instellingen voor de richting van de vloot 324
Controleren of een buitenaards wezen de rand heeft geraakt 324
De vloot laten zakken en van richting veranderen 325
Oefening 13-3. Regendruppels 326
Oefening 13-4. Constante regen 326
Buitenaardse wezens neerschieten 326
Botsingen van kogels detecteren 327
Grotere kogels maken om mee te testen 328
De vloot aanvullen 329
De kogels sneller maken 330
Opschonen van update_bullets() 330
Oefening 13-5. Vangen 331
Het spel beëindigen 331
Detecteren van botsingen tussen een buitenaards wezen en het schip 331
Reageren op botsingen tussen een buitenaards wezen en het schip 332
Buitenaardse wezens die de onderkant van het scherm bereiken 335
Game Over! 336
Identificeren welke delen van het spel uitgevoerd moeten worden 336
Oefening 13-6. Game Over 337
Samenvatting 337

14
SCORES 339

De Play-knop toevoegen 340
Een class voor een knop maken 340
De knop op het scherm tekenen 342
Het spel starten 343
Het spel resetten 344
De Play-knop deactiveren 345
De muisaanwijzer verbergen 346
Oefening 14-1. Druk op P om te spelen 346
Oefening 14-2. Schietoefeningen 346
Een niveau hoger 347
Wijzigen van de instellingen voor snelheid 347

De snelheid resetten . . . 348
Oefening 14-3. Uitdagende schietoefeningen . . . 349
Scores . . . 349
De score weergeven . . . 350
Een scorebord maken . . . 351
De score bijwerken als buitenaardse wezens neergeschoten worden . . . 352
Ervoor zorgen dat alles wat geraakt wordt telt . . . 354
Toenemende puntenwaarden . . . 354
De score afronden . . . 355
Hoogste scores . . . 356
Het niveau weergeven . . . 359
Het aantal schepen weergeven . . . 362
Oefening 14-4. Hoogste score aller tijden . . . 365
Oefening 14-5. Opschonen . . . 365
Oefening 14-6. Alien Invasion uitbreiden . . . 365
Samenvatting . . . 366

PROJECT 2: DATAVISUALISATIE 367

15
DATA GENEREREN 369

Matplotlib installeren . . . 370
Op Linux . . . 370
Op macOS . . . 371
Op Windows . . . 371
Matplotlib testen . . . 371
De matplotlib-galerij . . . 372
Een eenvoudige lijngrafiek plotten . . . 372
Het labeltype en de dikte van een grafiek wijzigen . . . 373
De plot corrigeren . . . 374
Individuele punten plotten en vormgeven met scatter() . . . 375
Een reeks punten plotten met scatter() . . . 376
Data automatisch berekenen . . . 377
De contouren van gegevenspunten verwijderen . . . 378
Aangepaste kleuren definiëren . . . 378
Een kleurenkaart gebruiken . . . 379
Je plots automatisch opslaan . . . 380
Oefening 15-1. Tot de derde macht . . . 380
Oefening 15-2. Gekleurde machten . . . 380
Willekeurige wandelingen . . . 380
Het maken van de class 'RandomWalk()' . . . 381
De richting kiezen . . . 381
De willekeurige wandeling plotten . . . 382
Meerdere willekeurige wandelingen genereren . . . 383
De wandeling vormgeven . . . 384
De punten een kleur geven . . . 384
Het plotten van begin- en eindpunten . . . 385
De assen opruimen . . . 386
Punten in een plot toevoegen . . . 386
De grootte aanpassen om het scherm te vullen . . . 387

Oefening 15-3. Moleculaire beweging *388*
Oefening 15-4. Aangepaste willekeurige wandelingen *388*
Oefening 15-5. Opschonen *388*
Rollende dobbelstenen met Pygal 388
Pygal installeren 389
De Pygal-galerij 389
Het maken van de class 'Die' 389
De dobbelsteen gooien 390
De resultaten analyseren 390
Een histogram maken 391
Twee dobbelstenen gooien 393
Het gooien van dobbelstenen met een verschillend aantal zijden 394
Oefening 15-6. Automatische labels *396*
Oefening 15-7. Twee keer een D8 . . . . . *396*
Oefening 15-8. Drie dobbelstenen *396*
Oefening 15-9. Vermenigvuldiging *396*
Oefening 15-10. Oefenen met beide bibliotheken *396*
Samenvatting 396

16
DATA DOWNLOADEN **399**

De CSV-bestandsindeling 400
De headers van het CSV-bestand verwerken 400
De headers en hun posities afdrukken 401
Dataextractie en het inlezen van data 402
Data plotten in een temperatuurgrafiek 403
De datetime-module 404
Datums plotten 405
Een langer tijdsbestek plotten 406
Een tweede reeks data plotten 407
Een gebied in de grafiek inkleuren 408
Foutcontrole 409
Oefening 16-1. San Francisco *412*
Oefening 16-2. Sitka-Death Valley-vergelijking *412*
Oefening 16-3. Neerslag *412*
Oefening 16-4. Verkennen *412*
Globale datasets in kaart brengen 412
Data over de wereldbevolking downloaden 412
Relevante data uitpakken 413
Strings omzetten in numerieke waarden 414
Het verkrijgen van tweecijferige landencodes 415
Een wereldkaart bouwen 417
Numerieke data op een wereldkaart plotten 418
Een complete kaart met inwoneraantallen plotten 419
Landen groeperen op inwonertal 421
Wereldkaarten in Pygal opmaken 422
Het kleurenthema lichter maken 424
Oefening 16-5. Alle landen *425*
Oefening 16-6. Bruto binnenlands product *425*
Oefening 16-7. Kies je eigen data *425*
Oefening 16-8. Testen van de `country_codes`-module *425*
Samenvatting 425

17
WERKEN MET API'S **427**

Een web-API gebruiken 428
Git en GitHub 428
Data opvragen met een API-aanroep 428
Het installeren van verzoeken 429
Een API-antwoord verwerken 429
Met het antwoordwoordenboek werken 430
De top-archieven samenvatten 433
Beperkingen van gelimiteerde API's in de gaten houden 434
Archieven visualiseren met Pygal 435
Pygal-grafieken verfijnen 436
Aangepaste tekstballonnen toevoegen 438
De data plotten 439
Klikbare links aan onze grafiek toevoegen 440
De Hacker News API 441
Oefening 17-1. Andere talen *444*
Oefening 17-2. Actieve discussies *444*
Oefening 17-3. Testen van python_repos.py *444*
Samenvatting 444

PROJECT 3: WEBTOEPASSINGEN **445**

18
AAN DE SLAG MET DJANGO **447**

Een project opzetten 448
Een spec schrijven 448
Een virtuele omgeving maken 448
Virtualenv installeren 449
De virtuele omgeving activeren 449
Django installeren 450
Een project maken in Django 450
De database maken 451
Het project bekijken 452
Oefening 18-1. Nieuwe projecten *453*
Een toepassing starten 453
Modellen definiëren 453
Modellen activeren 455
De Django admin-site 456
Het model voor vermeldingen definiëren 458
Het model voor vermeldingen migreren 459
Entry registreren in de admin-site 460
De Django-shell 461
Oefening 18-2. Korte vermeldingen *463*
Oefening 18-3. De Django API *463*
Oefening 18-4. Pizzeria *463*
Pagina's maken: de homepage van Learning Log 463
Een URL laten verwijzen 464
Een weergave schrijven 465
Een sjabloon schrijven 466

Oefening 18-5. Maaltijdplanner *467*
Oefening 18-6. Homepage van een pizzeria *467*
Extra pagina's bouwen 468
Overerven van een sjabloon 468
De pagina met onderwerpen 470
Pagina's met individuele onderwerpen 473
Oefening 18-7. Documentatie over sjablonen *476*
Oefening 18-8. Pizzeria-pagina's *476*
Samenvatting *477*

19
GEBRUIKERSACCOUNTS **479**

Gebruikers toestaan om data in te voeren 480
Nieuwe onderwerpen toevoegen 480
Nieuwe vermeldingen toevoegen 484
Vermeldingen bewerken 488
Oefening 19-1. Blog *491*
Gebruikersaccounts instellen 491
De toepassing voor gebruikers 492
De pagina voor het inloggen 493
Uitloggen 495
De registratiepagina 497
Oefening 19-2. Blog-accounts *500*
Gebruikers toestaan om hun eigen data te beheren 500
Toegang beperken met @login_required 500
Data aan bepaalde gebruikers verbinden 502
Toegang tot onderwerpen beperken tot de juiste gebruikers 505
De onderwerpen van een gebruiker beschermen 506
Het beschermen van de edit_entry-pagina 506
Nieuwe onderwerpen koppelen aan de huidige gebruiker 507
Oefening 19-3. Opschonen *508*
Oefening 19-4. Het beschermen van new_entry *508*
Oefening 19-5. Beveiligd blog *508*
Samenvatting 508

20
EEN TOEPASSING VORMGEVEN EN IMPLEMENTEREN **511**

Learning Log vormgeven 512
De django-bootstrap3-toepassing 512
Bootstrap gebruiken om Learning Log vorm te geven 513
Het aanpassen van base.html 514
De homepage vormgeven met een Jumbotron 517
De pagina voor het inloggen vormgeven 518
De new_topic-pagina vormgeven 519
De pagina met onderwerpen vormgeven 520
De vermeldingen op de pagina van een onderwerp vormgeven 520
Oefening 20-1. Andere formulieren *522*
Oefening 20-2. Stijlvol blog *522*
Learning Log implementeren 523
Een account maken bij Heroku 523
De Heroku CLI installeren 523

De vereiste pakketten installeren . . . 523
Een lijst van pakketten maken met een bestand requirements.txt . . . 524
De Python-runtime specificeren . . . 525
Het aanpassen van settings.py voor Heroku . . . 525
Een Procfile maken om processen te starten . . . 526
Het aanpassen van wsgi.py voor Heroku . . . 527
Een map voor statische bestanden maken . . . 527
De gunicorn-server lokaal gebruiken . . . 527
Git gebruiken om de bestanden in het project bij te houden . . . 528
Implementeren op Heroku . . . 530
De database op Heroku instellen . . . 531
De Heroku-implementatie verfijnen . . . 532
Het live-project beveiligen . . . 533
Wijzigingen vastleggen en doorsturen . . . 534
Aangepaste foutpagina's maken . . . 535
Doorgaan met ontwikkelen . . . 538
De instellingen van SECRET_KEY . . . 539
Een project verwijderen van Heroku . . . 539
Oefening 20-3. Live Blog . . . *540*
Oefening 20-4. Meer van 404 . . . *540*
Oefening 20-5. Learning Log uitbreiden . . . *540*
Samenvatting . . . 540

NAWOORD **541**

A
PYTHON INSTALLEREN **543**

Python op Linux . . . 543
Uitvinden welke versie er geïnstalleerd is . . . 543
Python 3 installeren op Linux . . . 544
Python op macOS . . . 544
Uitvinden welke versie er geïnstalleerd is . . . 544
Homebrew gebruiken om Python 3 te installeren . . . 545
Python op Windows . . . 546
Python 3 installeren op Windows . . . 546
De Python-interpreter vinden . . . 546
Python toevoegen aan de variabele voor het pad . . . 547
Python trefwoorden en ingebouwde functies . . . 547
Trefwoorden van Python . . . 548
Ingebouwde functies van Python . . . 548

B
TEKSTEDITORS **549**

Geany . . . 550
Geany installeren op Linux . . . 550
Geany installeren op Windows . . . 551
Python-programma's uitvoeren in Geany . . . 551
Instellingen van Geany aanpassen . . . 552
Sublime Text . . . 552
Sublime Text installeren op MacOS . . . 553
Sublime Text installeren op Linux . . . 553

Sublime Text installeren op Windows . . . 553
Python-programma's uitvoeren in Sublime Text . . . 553
Configureren van Sublime Text . . . 553
De instellingen van Sublime Text aanpassen . . . 554
IDLE . . . 555
IDLE installeren op Linux . . . 555
IDLE installeren op macOS . . . 555
IDLE installeren op Windows . . . 555
Instellingen van IDLE aanpassen . . . 555
Emacs en vim . . . 556

C
HULP KRIJGEN **557**

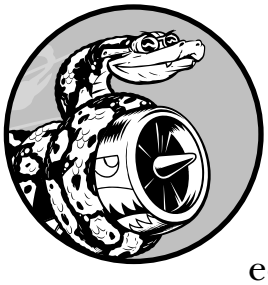
Eerste stappen . . . 557
Probeer het nog eens . . . 558
Neem een pauze . . . 558
Raadpleeg de oefenbestanden bij dit boek . . . 559
Online opzoeken . . . 559
Stack Overflow . . . 559
De officiële Python-documentatie . . . 560
Officiële documentatie van de bibliotheek . . . 560
r/learnpython . . . 560
Blog-berichten . . . 560
IRC (Internet Relay Chat) . . . 560
Een IRC-account maken . . . 560
Kanalen om aan deel te nemen . . . 561
IRC-cultuur . . . 561

D
GIT GEBRUIKEN VOOR VERSIECONTROLE **563**

Git installeren . . . 564
Git installeren op Linux . . . 564
Git installeren op macOS . . . 564
Git installeren op Windows . . . 564
Git configureren . . . 564
Een project maken . . . 565
Bestanden negeren . . . 565
Een archief initialiseren . . . 565
De toestand controleren . . . 566
Bestanden aan het archief toevoegen . . . 566
Een commit vastleggen . . . 567
Het logboek controleren . . . 567
De tweede commit . . . 568
Een verandering terugdraaien . . . 569
Vorige commits bekijken . . . 570
Het archief verwijderen . . . 571

INDEX **575**

INLEIDING



Alle programmeurs hebben wel een verhaal over hoe ze geleerd hebben hun eerste programma te schrijven. Ik begon er als kind mee toen mijn vader voor Digital Equipment Corporation werkte, een van de pioniers van het moderne computertijdperk. Ik schreefW mijn eerste programma op een zelfgebouwde computer die mijn vader in onze kelder in elkaar had gezet. De computer bestond uit niets meer dan een kaal moederbord, aangesloten op een toetsenbord zonder een behuizing, met een kale kathodestraalbuis als beeldscherm. Mijn eerste programma was een simpel spel om een getal te raden, en zag er ongeveer zo uit:

```
Ik denk aan een getal! Probeer te raden aan welk getal ik denk: 25
Te laag! Raad opnieuw: 50
Te hoog! Raad opnieuw: 42
```

```
Goed! Wil je nog een keer spelen? (ja/nee) nee
Bedankt voor het spelen!
```

Ik zal me altijd blijven herinneren hoe tevreden ik me voelde als ik mijn familie een spel zag spelen dat ik gemaakt had en dat werkte zoals ik het bedoeld had.

Die vroege ervaring heeft een blijvende impact gehad. Je kunt echt voldoening halen uit iets met een doel bouwen, iets dat een probleem oplost. De software die ik nu schrijf, komt tegemoet aan een belangrijkere behoefte dan mijn pogingen toen ik kind was, maar het gevoel van voldoening dat ik haal uit het maken van een programma dat werkt is nog grotendeels hetzelfde.

Voor wie is dit boek bedoeld?

Het doel van dit boek is om je zo snel mogelijk op de hoogte te brengen van Python zodat je programma's kunt bouwen die werken (spellen, datavisualisatie en webtoepassingen), waarbij je een basis in programmeren ontwikkelt waar je de rest van je leven nog veel aan zult hebben. *Crash course programmeren in Python* is geschreven voor mensen van alle leeftijden die nog nooit in Python geprogrammeerd hebben of überhaupt nog nooit geprogrammeerd hebben. Als je snel de basisprincipes van programmeren wilt leren, zodat je je op interessante projecten kunt richten en je jouw inzicht in nieuwe concepten wilt testen door zinvolle problemen op te lossen, dan is dit boek echt iets voor jou. *Crash course programmeren in Python* is ook geschikt voor docenten op middelbare scholen die hun studenten een projectmatige inleiding in programmeren willen aanbieden.

Wat kun je verwachten te gaan leren?

Het doel van dit boek is om van jou een goede programmeur in het algemeen te maken en een goede Python-programmeur in het bijzonder. Je leert op een efficiënte manier en maakt je goede gewoonten eigen doordat ik je een stevige basis bied in de algemene concepten van het programmeren. Nadat je *Crash course programmeren in Python* doorgelezen hebt, zou je klaar moeten zijn voor meer geavanceerde Python-technieken en begrijp je een volgende programmeertaal nog makkelijker.

In het eerste deel van dit boek leer je de basisconcepten van programmeren die je nodig hebt om Python-programma's te schrijven. Deze concepten zijn dezelfde als die je leert wanneer je start met vrijwel elke andere programmeertaal. Je leert over verschillende soorten gegevens en de manieren waarop je gegevens opslaat in lijsten en woordenboeken in je programma's. Je leert om verzamelingen van gegevens te bouwen en op efficiënte manieren met die verzamelingen te werken. Je komt te weten hoe je `while`- en `if`-loops gebruikt om te testen op bepaalde voorwaarden zodat je specifieke delen van je code kunt uitvoeren als aan deze voorwaarden voldaan wordt, en andere delen als dat niet het geval is; een techniek die

enorm helpt om processen te automatiseren.

Hierbij zul je zien dat je de code programmeert in het Engels. De gegevens die je invoert zijn eveneens in het Engels. Hier is voor gekozen omdat in de meeste programmeertalen in het Engels wordt geprogrammeerd en je hier zo direct aan gewend raakt. Dit is tevens handig als je jouw code door anderen wilt laten nakijken of problemen hebt. Zo kan iedereen ter wereld jou helpen.

Je leert om invoer van gebruikers te accepteren om je programma's interactief te maken en om je programma's te laten draaien zolang de gebruiker actief is. Je verkent hoe je functies schrijft om delen van je programma herbruikbaar te maken, zodat je blokken code die bepaalde handelingen uitvoeren maar één keer hoeft te schrijven en je die daarna zo vaak als je wilt kunt gebruiken. Daarna breid je dit concept uit naar meer complex gedrag met classes, waardoor relatief eenvoudig programma's reageren op een diversiteit aan situaties. Je leert om programma's te schrijven die netjes omgaan met veelvoorkomende fouten. Nadat je elk van deze basisbegrippen hebt doorgenomen, ga je enkele korte programma's schrijven die enkele goed gedefinieerde problemen oplossen. Ten slotte zet je je eerste stap richting iets verder gevorderd programmeren door te leren hoe je tests voor je code schrijft, zodat je jouw programma's verder kunt ontwikkelen zonder bang te hoeven zijn dat er bugs in komen. Alle informatie in deel I bereidt je voor op grotere, meer complexe projecten.

In deel II pas je wat je geleerd hebt in deel I toe op drie projecten. Je kunt een of al deze projecten doen in de volgorde die voor jou het beste werkt. In het eerste project (hoofdstukken 12-14) maak je een shooting game in de stijl van Space Invaders, genaamd Alien Invasion, dat bestaat uit niveaus van toenemende moeilijkheid. Nadat je dit project hebt afgerond, ben je goed op weg om je eigen 2D-spellen te kunnen ontwikkelen.

Het tweede project (hoofdstukken 15-17) introduceert datavisualisatie. Datawetenschappers hebben als doel om iets zinnigs te zeggen over de enorme hoeveelheid informatie die hen ter beschikking staat door middel van verschillende visualisatietechnieken. Je gaat aan de slag met datasets die je genereert via code, datasets gedownload van onlinebronnen en datasets die jouw programma's automatisch downloaden. Nadat je klaar bent met dit project, kun je programma's schrijven die grote datasets doornemen en van die opgeslagen informatie visuele weergaven maken.

In het derde project (hoofdstukken 18-20) bouw je een kleine webtoepassing met de naam Learning Log. Met dit project kun je een dagboek bijhouden van ideeën en concepten die je hebt geleerd over een specifiek onderwerp. Je kunt aparte dagboeken bijhouden voor verschillende onderwerpen en anderen toestaan om een account te maken en hun eigen dagboeken te beginnen. Je leert ook hoe je jouw project uitrolt, zodat iedereen er online van waar dan ook toegang toe heeft.

Waarom Python?

Elk jaar overweeg ik door te gaan met Python of over te stappen op een andere taal, misschien een die nieuwer is in de wereld van het

programmeren. Maar om veel redenen blijf ik me richten op Python. Python is een ongelooflijk efficiënte taal: je programma's doen meer in minder regels code dan in vele andere talen nodig zouden zijn. De syntax van Python helpt je ook om 'schone' code te schrijven. Je code is gemakkelijk te lezen, gemakkelijk te debuggen en gemakkelijk om uit te breiden en op verder te bouwen in vergelijking met andere talen.

Mensen gebruiken Python voor vele doeleinden: om spellen te maken, webtoepassingen te bouwen, zakelijke problemen op te lossen en om interne tools te ontwikkelen bij allerlei interessante bedrijven. Python wordt ook veel gebruikt in wetenschappelijke vakgebieden voor academisch en toegepast onderzoek.

Een van de belangrijkste redenen waarom ik Python blijf gebruiken is vanwege de Python-gemeenschap, die bestaat uit een onvoorstelbaar gevarieerde en hartelijke groep mensen. Een gemeenschap is essentieel voor programmeurs, omdat programmeren geen solitair streven is. De meesten van ons, zelfs de meest ervaren programmeurs, moeten soms advies vragen aan anderen die al vergelijkbare problemen hebben opgelost. Het hebben van een betrokken en ondersteunende gemeenschap is essentieel in het helpen oplossen van problemen en de Python-gemeenschap is heel solidair met mensen zoals jij die Python leren als hun eerste programmeertaal.

Python is een geweldige taal om te leren, dus ga aan de slag!

DEEL I

DE BASIS

In deel I van dit boek leer je de basisconcepten die je nodig hebt om Python-programma's te schrijven. Veel van deze begrippen zijn gemeenschappelijk voor alle programmeertalen, dus zijn ze tijdens je hele leven als programmeur nuttig.

In **hoofdstuk 1** installeer je Python op je computer en voer je je eerste programma uit dat het bericht *Hello world!* in je venster zal afdrukken.

In **hoofdstuk 2** leer je om informatie in variabelen op te slaan en om met tekst en numerieke waarden te werken.

In de **hoofdstukken 3** en **4** introduceren we lijsten. Lijsten kunnen zoveel informatie als je wilt in één variabele opslaan, zodat je efficiënt met die gegevens kunt werken. In slechts een paar regels code kun je werken met honderden, duizenden of zelfs miljoenen waarden.

In **hoofdstuk 5** gebruik je `if`-instructies om code te schrijven die op de ene manier reageert als aan bepaalde voorwaarden wordt voldaan, en op een andere manier als niet aan die voorwaarden wordt voldaan.

In **hoofdstuk 6** lees je hoe je de woordenboeken van Python kunt gebruiken, waarmee je verbindingen kunt maken tussen verschillende stukjes informatie. Net zoals lijsten kunnen woordenboeken net zoveel informatie bevatten als je moet opslaan.

In **hoofdstuk 7** leer je hoe je invoer van gebruikers accepteert om je programma's interactief te maken. Je leert dan ook meer over `while`-lussen, waarmee blokken code herhaaldelijk uitgevoerd worden zolang aan bepaalde voorwaarden voldaan blijft worden.

In **hoofdstuk 8** schrijf je functies. Dat zijn blokken code met een naam, die een specifieke taak uitvoeren en uitgevoerd kunnen worden wanneer je ze nodig hebt.

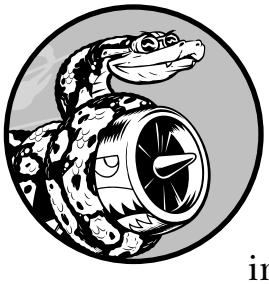
In **hoofdstuk 9** worden classes geïntroduceerd, waarmee je objecten uit de echte wereld kunt modelleren, zoals honden, katten, mensen, auto's, raketten en nog veel meer, zodat je code alles wat echt of abstract is voor kan stellen.

In **hoofdstuk 10** zie je hoe je met bestanden kunt werken en hoe je omgaat met fouten, zodat je programma's niet onverwacht zullen crashen. Je slaat gegevens op voordat je programma afgesloten wordt en je leest de gegevens opnieuw in als het programma weer draait. Je komt alles te weten over de uitzonderingen in Python, waarmee je kunt anticiperen op fouten en je zorgt dat jouw programma's op een elegante manier met die fouten omgaan.

In **hoofdstuk 11** leer je om tests voor je code te schrijven, om te controleren of je programma's werken zoals je ze bedoeld hebt. Zo kun je je programma's uitbreiden zonder je zorgen te maken dat er nieuwe bugs in terechtkomen. Het testen van je code is een van de eerste vaardigheden die je helpt de overgang van beginnende naar gemiddelde programmeur te maken.

2

VARIABELEN EN EENVOUDIGE DATATYPES



In dit hoofdstuk leer je over de verschillende soorten data waarmee je in je Python-programma's kunt werken. Je leert ook hoe je je data in variabelen opslaat en hoe je die variabelen in je programma's gebruikt.

Wat er daadwerkelijk gebeurt als je hello_world.py uitvoert

Laten we eens beter kijken naar wat Python doet wanneer je *hello_world.py* uitvoert. Wat blijkt? Python verzet een behoorlijke hoeveelheid werk, zelfs wanneer het een eenvoudig programma uitvoert:

hello_world.py

```
print("Hello Python world!")
```

Wanneer je deze code uitvoert, zie je deze uitvoer:

```
Hello Python world!
```

Als je het bestand *hello_world.py* uitvoert, geeft de extensie *.py* aan dat het bestand een Python-programma is. Je editor haalt het bestand dan door de *Python-interpreter*, die het programma leest en bepaalt wat ieder woord in het programma betekent. Als de interpreter bijvoorbeeld het woord `print` ziet, drukt hij naar het scherm af wat er tussen de haakjes staat.

Terwijl je je programma's schrijft, markeert je editor de verschillende onderdelen van je programma op verschillende manieren. Hij herkent bijvoorbeeld dat `print` de naam van een functie is en geeft dat woord in het blauw weer. Hij herkent dat "Hello Python world!" geen Python-code is en geeft die zin in het oranje weer. Deze functie heet *syntaxis markeren* en is heel handig als je begint met het schrijven van je eigen programma's.

Variabelen

We zullen eens proberen om een variabele te gebruiken in *hello_world.py*. Voeg een nieuwe regel toe aan het begin van het bestand en wijzig de tweede regel:

```
message = "Hello Python world!"
print(message)
```

Voer dit programma uit om te zien wat er gebeurt. Je zou dezelfde uitvoer moeten krijgen als voorheen:

```
Hello Python world!
```

We hebben een *variabele* toegevoegd, die `message` heet. Elke variabele heeft een *value*, een waarde, namelijk de informatie die bij die variabele hoort. In dit geval is de waarde de tekst "Hello Python world!"

Het toevoegen van een variabele zorgt voor een klein beetje meer werk voor de Python-interpreter. Wanneer die de eerste regel verwerkt, associeert hij de tekst "Hello Python world!" met de variabele `message`. Wanneer hij de tweede regel bereikt, drukt hij de waarde op het scherm af die met `message` is geassocieerd.

We breiden dit programma wat uit door *hello_world.py* aan te passen, zodat een tweede bericht afgedrukt wordt. Voeg een lege regel toe aan *hello_world.py* en voeg daarna twee nieuwe regels code toe:

```
message = "Hello Python world!"
print(message)

message = "Hello Python Crash Course world!"
print(message)
```

Als je nu *hello_world.py* uitvoert, zie je als het goed is twee regels uitvoer:

```
Hello Python world!
Hello Python Crash Course world!
```

Je kunt de waarde van een variabele in je programma altijd aanpassen. Python houdt altijd de huidige waarde bij.

Variabelen benoemen en gebruiken

Wanneer je in Python variabelen gebruikt, moet je je aan een aantal regels en richtlijnen houden. Als je sommige regels overtreedt, leidt dat tot foutmeldingen; andere richtlijnen zijn bedoeld om je te helpen de code zo te schrijven dat die makkelijker te lezen en te begrijpen is. Zorg dat je je steeds aan de volgende regels voor variabelen houdt:

- Namen van variabelen mogen alleen letters, getallen en liggende streepjes bevatten. Ze kunnen beginnen met een letter of liggend streepje, maar niet met een getal. Zo kun je een variabele wel *message_1* noemen, maar niet *1_message*.
- Spaties mogen niet gebruikt worden in namen van variabelen, maar je kunt wel liggende streepjes gebruiken om woorden in namen van variabelen te scheiden. Zo werkt *greeting_message* wel, maar zal *greeting message* foutmeldingen opleveren.
- Vermijd het gebruik van Python-trefwoorden en -functienamen in namen van variabelen. Gebruik dus geen woorden die Python heeft gereserveerd voor een bepaald programmeerdoel, zoals het woord `print`. (Zie “Python-trefwoorden en ingebouwde functies” op bladzijde 547.)
- Namen van variabelen moeten kort maar duidelijk zijn. *name* is beter bijvoorbeeld dan *n*, *student_name* is beter dan *s_n*, en *name_length* is beter dan *length_of_persons_name*.
- Pas op wanneer je de kleine letter *l* en de hoofdletter *O* gebruikt, omdat ze verward kunnen worden met de getallen *1* en *0*.
- In dit boek worden Engelstalige namen gebruikt, maar je kunt in je eigen programma’s ook Nederlandstalige namen gebruiken.

Het vergt enige oefening om te leren hoe je variabelen goede namen geeft, vooral wanneer je programma’s interessanter en ingewikkelder worden. Als je meer programma’s schrijft en begint met het bekijken van code van anderen, word je beter in het bedenken van zinvolle namen.

LET OP! *De Python-variabelen die je nu gebruikt moeten in kleine letters geschreven zijn. Je zult geen foutmeldingen krijgen als je hoofdletters gebruikt, maar het is goed om ze op dit moment niet te gebruiken.*

Foutmeldingen in namen voorkomen als je variabelen gebruikt

Elke programmeur maakt fouten, de meesten iedere dag. Hoewel goede programmeurs foutmeldingen kunnen veroorzaken, weten ze ook hoe ze efficiënt op die fouten kunnen reageren. We zullen eens kijken naar een fout die je in het begin waarschijnlijk zult maken en leren hoe je die oplost.

Je gaat wat code schrijven die met opzet een foutmelding oplevert. Voer de volgende code in, inclusief het verkeerd gespelde, vetgedrukte woord *mesage*:

```
message = "Hello Python Crash Course reader!"
print(mesage)
```

Wanneer er een fout zit in je programma, helpt de Python-interpreter je te ontdekken waar het probleem zich voordoet. De interpreter geeft een traceback wanneer een programma niet succesvol kan worden uitgevoerd. Een *traceback* is een registratie van waar de interpreter op problemen stuitte bij de poging om je code uit te voeren. Hier volgt een voorbeeld van de traceback die Python geeft nadat je per ongeluk een naam van een variabele verkeerd gespeld hebt:

```
Traceback (most recent call last):
❶  File "hello_world.py", line 2, in <module>
❷    print(mesage)
❸ NameError: name 'mesage' is not defined
```

De uitvoer bij ❶ geeft aan dat er een fout zit in regel 2 van het bestand *hello_world.py*. De interpreter toont deze regel om je te helpen de fout snel te ontdekken ❷ en vertelt ons wat voor soort fout hij gevonden heeft ❸. In dit geval is er een *name error* (naamfout) gevonden en geeft de interpreter aan dat de variabele die wordt afgedrukt, `mesage`, niet gedefinieerd is. Python kan de naam van de aangeboden variabele niet identificeren. Een naamfout betekent meestal dat je vergeten bent om de waarde van een variabele in te stellen voordat je die gebruikt, of dat je een spelfout gemaakt hebt bij het invoeren van de naam van de variabele.

Natuurlijk hebben we in dit voorbeeld in de tweede regel de letter *s* weggelaten in de naam van de variabele `message`. De Python-interpreter voert geen spellingscontrole uit op je code, maar zorgt er wel voor dat de namen van variabelen consistent gespeld worden. Kijk bijvoorbeeld maar wat er gebeurt wanneer we *message* ook op een andere plaats in de code onjuist spellen:

```
mesage = "Hello Python Crash Course reader!"
print(mesage)
```

In dit geval wordt het programma succesvol uitgevoerd!

```
Hello Python Crash Course reader!
```

Computers zijn precies, maar kijken niet of iets goed gespeld is. Je hoeft dus geen rekening te houden met de Engelse spellings- en grammaticaregels wanneer je variabelen een naam geeft en code schrijft.

Veel programmeerfouten zijn simpele typefouten van één teken in één programmaregel. Als je heel lang naar zo'n fout aan het zoeken bent, weet dan dat je in goed gezelschap bent. Veel ervaren en getalenteerde programmeurs zijn uren bezig dit soort kleine fouten op te sporen. Probeer er maar om te lachen en gewoon weer door te gaan, in de wetenschap dat het nog wel vaker voor zal komen in je programmeercarrière.

LET OP! *De beste manier om nieuwe programmeerconcepten te leren begrijpen is ze te gebruiken in je programma's. Als je tijdens het werken aan een oefening in dit boek vastloopt, ga dan even iets anders doen. Als je dan nog steeds vastzit, lees dan het betreffende deel van dat hoofdstuk nog eens door. Als je nog steeds hulp nodig hebt, raadpleeg dan de suggesties in Bijlage C.*

PROBEER HET ZELF

Schrijf aparte programma's voor elk van deze oefeningen. Sla elk programma op met een bestandsnaam volgens de standaard Python-conventies en gebruik kleine letters en liggende streepjes, zoals *simple_message.py* en *simple_messages.py*.

2-1. Simple Message: Sla een bericht op in een variabele en druk vervolgens dat bericht af.

2-2. Simple Messages: Sla een bericht op in een variabele en druk dat bericht af. Verander vervolgens de waarde van je variabele in een nieuw bericht en druk dat nieuwe bericht af.

Strings

Omdat de meeste programma's verschillende soorten data definiëren en verzamelen en daar dan iets nuttigs mee doen, is het handig om verschillende datatypes te classificeren. Het eerste datatype waar we naar gaan kijken is de string. Strings lijken op het eerste gezicht vrij eenvoudig, maar je kunt ze op veel verschillende manieren gebruiken.

Een *string* is niets anders dan een reeks tekens. In Python wordt alles tussen aanhalingstekens beschouwd als een string. Je kunt enkele of dubbele aanhalingstekens rond je strings gebruiken, zoals hieronder:

```
"This is a string."
'This is also a string.'
```

Dankzij deze flexibiliteit kun je aanhalingstekens en apostroffen in je strings gebruiken:

```
'I told my friend, "Python is my favorite language!"'
"The language 'Python' is named after Monty Python, not the snake."
"One of Python's strengths is its diverse and supportive community."
```

We zullen eens kijken op welke manieren je strings kunt gebruiken.

Wisselen tussen hoofd- en kleine letters in een string met methoden

Een van de eenvoudigste taken die je met strings kunt uitvoeren is het veranderen van de hoofd- of kleine letters van de woorden in een string. Bekijk de onderstaande code en probeer te bedenken wat er gebeurt:

name.py

```
name = "ada lovelace"
print(name.title())
```

Sla dit bestand op als *name.py* en voer het dan uit. Als het goed is, krijg je deze uitvoer:

```
Ada Lovelace
```

In dit voorbeeld is de string `"ada lovelace"` met kleine letters opgeslagen in variabele `name`. De methode `title()` staat achter de variabele in de `print()`-uitdrukking. Een *methode* is een actie die Python kan uitvoeren op een stukje informatie. De punt (`.`) achter `name` in `name.title()` vertelt Python dat het de `title()`-methode moet uitvoeren op de variabele `name`. Elke methode wordt gevolgd door een set haakjes, omdat methodes vaak aanvullende informatie nodig hebben om hun werk te doen. Die informatie wordt verstrekt tussen de haakjes. De `title()`-functie heeft echter geen aanvullende informatie nodig, dus er staat niets tussen de haakjes.

`title()` zorgt ervoor dat elk woord begint met een hoofdletter. Dit is handig omdat je een naam vaak beschouwt als een stukje informatie. Je wilt bijvoorbeeld dat je programma de invoerwaarden `Ada`, `ADA`, en `ada` als dezelfde naam herkent en die allemaal weergeeft als `Ada`.

Er zijn nog meer handige methoden beschikbaar voor het werken met hoofdletters. Je kunt een string bijvoorbeeld als volgt veranderen in allemaal hoofdletters of allemaal kleine letters:

```
name = "Ada Lovelace"
print(name.upper())
print(name.lower())
```

Dan krijg je het volgende:

```
ADA LOVELACE
ada lovelace
```

De `lower()`-methode is vooral handig voor het opslaan van data. Vaak is het beter om niet te vertrouwen op het gebruik van hoofdletters zoals je gebruikers die aan jou doorgeven, dus zet je de strings om naar kleine

DEEL II

PROJECTEN

Gefeliciteerd! Je weet nu genoeg over Python om te beginnen met het bouwen van interactieve en zinvolle projecten. Door het maken van je eigen projecten leer je nieuwe vaardigheden en verstevig je jouw begrip van de concepten die in Deel I geïntroduceerd zijn.

Deel II bevat drie soorten projecten en je kunt kiezen of je er een doet of allemaal, in de volgorde die jij wilt. Hier volgt een korte beschrijving van elk project om je te helpen beslissen met welke je als eerste aan de slag wilt gaan.

Alien Invasion: een spel maken met Python

In het Alien Invasion-project (hoofdstukken 12, 13 en 14) gebruik je het Python-pakket om een 2D-spel te ontwikkelen waarin het de bedoeling is om een vloot van buitenaardse wezens neer te schieten die op het scherm omlaag vallen, in niveaus die toenemen qua snelheid en moeilijkheid. Aan het einde van het project heb je vaardigheden geleerd waarmee je jouw eigen 2D-spellen in Pygame kunt ontwikkelen.

Datavisualisatie

Het Datavisualisatie-project begint in hoofdstuk 15, waarin je leert om data te genereren en een reeks functionele en mooie visualisaties van die data te maken met matplotlib en Pygal. In hoofdstuk 16 leer je data uit onlinebronnen te halen en deze door te sturen naar een visualisatiepakket

om een grafische weergave van weergegevens en een kaart van de wereldbevolking te maken. Tot slot zie je in hoofdstuk 17 hoe je een programma schrijft om automatisch data te downloaden en te visualiseren. Door te leren hoe je visualisaties maakt, kun je het vakgebied van datamining verkennen, wat in de hedendaagse wereld een uiterst gewilde vaardigheid is.

Webtoepassingen

In het Webtoepassingen-project (hoofdstukken 18, 19 en 20) gebruik je het Django-pakket om een eenvoudige webtoepassing te maken waarmee gebruikers een dagboek kunnen bijhouden over een onbeperkt aantal onderwerpen waarvan ze kennisgenomen hebben. Gebruikers maken een account aan met een gebruikersnaam en wachtwoord, voeren een onderwerp in en maken dan aantekeningen over wat ze leren. Je leert ook hoe je jouw toepassing uitrolt, zodat iedereen in de wereld er toegang toe heeft.

Na het afronden van dit project ben je in staat je eigen eenvoudige webtoepassingen te bouwen en ben je klaar om je te verdiepen in diepgaandere hulpmiddelen over het bouwen van toepassingen met Django.