

Les 10 - Pakketten

[« Vorige les](#)[» Volgende les](#)

Huiswerk

Programmeren voor Beginners - Les 10

Je hebt je huiswerk nog niet ingediend.

[Huiswerk inzenden](#)

Je hebt 1 vraagmoment. 1 moment over.

[Ga naar stel een vraag](#)[!\[\]\(d3102649f02e825ddb76dc3de0190154_img.jpg\) Markeer als deelgenomen aan deze les](#)

Inhoudsopgave

- 10.1** Inleiding
- 10.2** Pakketten van derden installeren
- 10.3** Een pakket installeren
- 10.4** Een pakket importeren
- 10.5** Zelf een pakket maken en importeren
- 10.6** Veelgebruikte pakketten
- 10.7** Project 'IJssalon'
- 10.8** Samenvatting
- 10.9** Vraagmoment
- 10.10** Oefenopgaven
- 10.11** Huiswerkopgaven

10.1 Inleiding

In een eerdere les noemden we al een van de grote voordelen van Python: u kunt relatief gemakkelijk bestaande codes in uw eigen programma verwerken. Dat zorgt ervoor dat u tamelijk snel van idee tot een werkend programma kunt komen.



Dankzij hulp van andere programmeurs kunt u sneller uw doel bereiken:
een werkend programma.

Dergelijke bruikbare codes zijn verpakt in pakketten. Enkele daarvan zitten al in uw computer, andere zult u nog moeten installeren. We laten u zien hoe dat werkt, evenals het importeren van pakketten.

Ook laten we u zien hoe u uw eigen pakket kunt samenstellen en het kunt gebruiken in uw 'hoofdcode'.

Leerdoelen

Aan het einde van deze les weet u:

- wat een pakket is;
- het grote voordeel van werken met pakketten;
- wat de *Python Package Index* (PyPi) is;
- hoe u een pakket installeert met pakketmanager PIP;
- hoe u een pakket importeert;
- hoe u zelf een pakket maakt en het gebruikt in uw 'hoofdcode';
- welke pakketten veel gebruikt worden door andere programmeurs.



Enkele afbeeldingen in deze les zijn moeilijk leesbaar in de printversie van deze les. Wilt u een afbeelding uitvergroten? Bekijk dan de digitale versie van de les op Plaza. Klik op de afbeeldingen om ze te vergroten.

10.2 Pakketten van derden installeren

We werken in deze cursus om meerdere redenen met Python. Twee van die redenen zijn:

1. De taal is relatief gemakkelijk te leren.
2. De taal is populair.

Vooral op die tweede reden willen we dieper ingaan. Doordat de taal populair is, heeft de taal veel gebruikers. Anders gezegd: er is sprake van een grote **community**. Dat heeft grote voordelen:

- Er is veel hulp beschikbaar voor iemand die vragen of problemen heeft.
- De community zorgt ervoor dat de taal continu in ontwikkeling is.

Gebruikers dragen op verschillende manieren bij aan deze ontwikkeling. Dat doen ze onder meer door het maken van **pakketten**, oftewel kant-en-klare uitbreidingen van uw projecten. U kunt deze pakketten vrij gebruiken in uw Python-project. U hoeft dan niet een deel van

uw programma zelf te schrijven.

We onderscheiden twee soorten pakketten:

1. Pakketten die al geïnstalleerd zijn
2. Pakketten die u nog moet installeren

Informatie over beide soorten pakketten vindt u op www.pypi.org. PyPi staat voor *Python Package Index*. Het is een centrale opslagplaats waar u alle pakketten vindt die beschikbaar zijn voor installatie.

The screenshot shows the homepage of PyPi. At the top, there's a navigation bar with links for Help, Sponsors, Log in, and Register. Below the header, a large blue banner features the text "Find, install and publish Python packages with the Python Package Index". A search bar is positioned below the banner, with the placeholder "Search projects" and a magnifying glass icon. Underneath the search bar, there's a link "Or browse projects". In the middle section, there are statistics: 384,124 projects, 3,572,130 releases, 6,282,569 files, and 603,693 users. The main content area contains the PyPi logo and a brief description: "The Python Package Index (PyPi) is a repository of software for the Python programming language. PyPi helps you find and install software developed and shared by the Python community. Learn about installing packages." It also mentions that "Package authors use PyPi to distribute their software. Learn how to package your Python code for PyPi." At the bottom of the page, a box contains the text: "De Python Package Index bevat alle pakketten die beschikbaar zijn voor installatie."

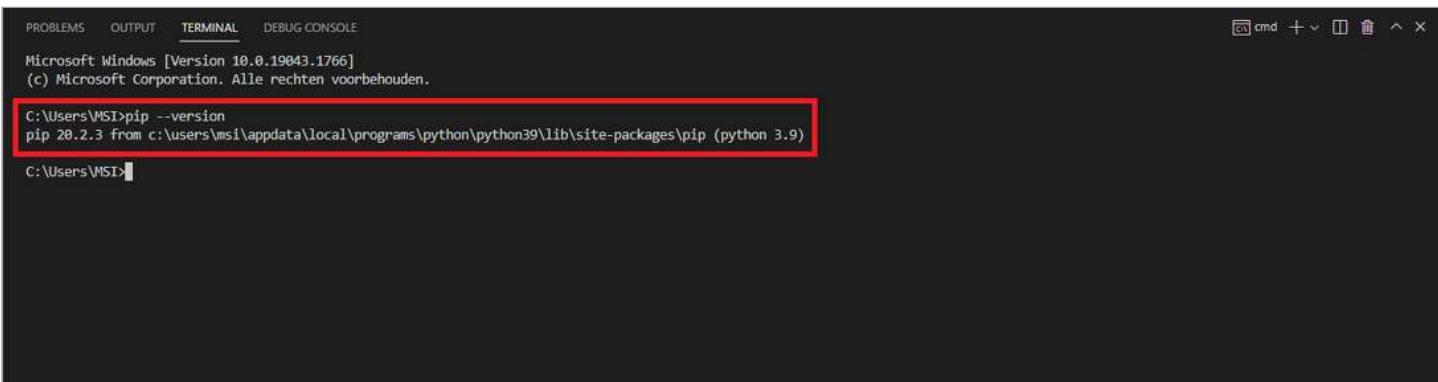
10.3 Een pakket installeren

Als u een pakket wil gebruiken dat niet standaard geïnstalleerd is, moet u het zelf installeren. Dat kan op verschillende manieren. De eenvoudigste manier is het gebruiken van een **pakketmanager** (Engels: *package manager*). De meest gebruikte pakketmanager is **PIP**. PIP staat voor *Pip Installs Packages* of *Pip Installs Python*.

Sinds versie 3.4 wordt deze pakketmanager standaard meegeleverd. De kans is dus groot dat u het niet meer apart hoeft te installeren. U kunt op de volgende manier controleren of dat inderdaad het geval is:

- Open VS Code.
- Open de Terminal.
- Typ pip --version en druk op Enter.

Als PIP inderdaad geïnstalleerd is, krijgt u te zien welke versie er geïnstalleerd is. Bijvoorbeeld:



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Microsoft Windows [Version 10.0.19043.1766]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI>pip --version
pip 20.2.3 from c:\users\msi\appdata\local\programs\python\python39\lib\site-packages\pip (python 3.9)

C:\Users\MSI>
```

Mocht PIP niet geïnstalleerd zijn, dan dient u Python te de-installeren en daarna opnieuw te installeren.



Zodra u weet dat PIP voor u klaarstaat, kunt u deze pakketmanager gebruiken om pakketten te installeren.

Er zijn veel pakketten voor u beschikbaar: op het moment van schrijven van deze cursus waren dat er meer dan 350.000. En dan hebben we het alleen nog maar over de pakketten die officieel geregistreerd zijn in de PyPi. Die pakketten heeft u lang niet allemaal nodig. Maar welke wel? Dat zal afhangen van het project waaraan u werkt.

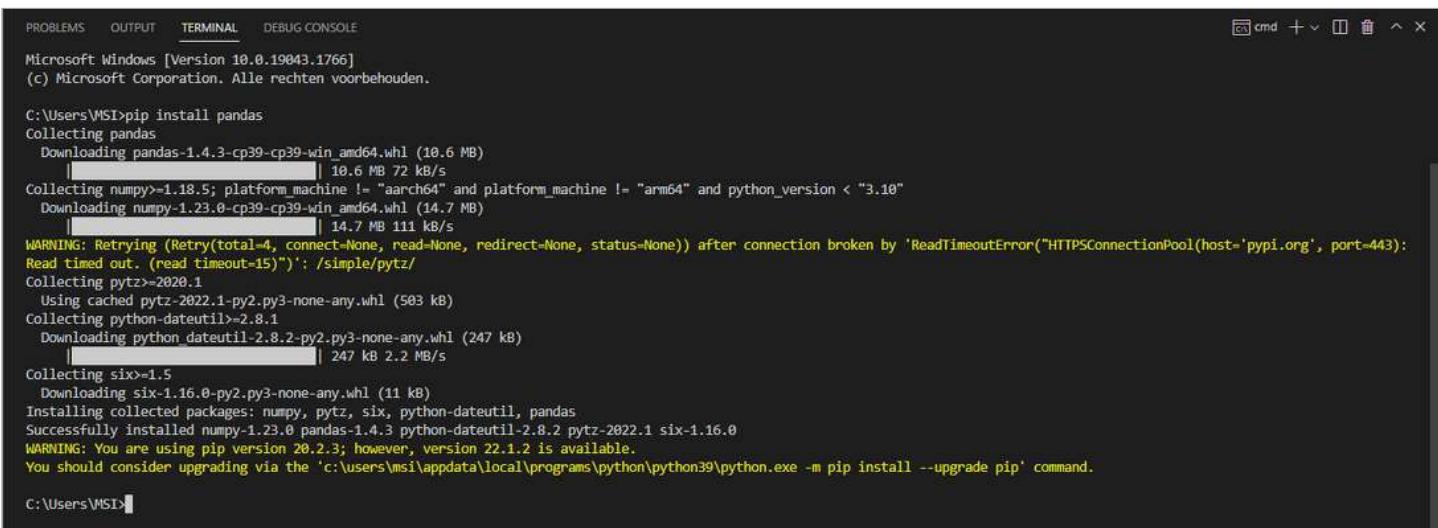
In deze paragraaf installeren we het pakket *Pandas*. Dit pakket is speciaal ontworpen voor het analyseren van middelgrote datasets. Het wordt veelgebruikt door programmeurs die gereeld met data werken.

Meer informatie over dit pakket vindt u op www.w3schools.com/python/pandas/pandas_intro.asp.



- Open de Terminal.
- Typ `pip install pandas` en druk op Enter.

Het duurt even voordat er output in de *Terminal* verschijnt, maar uiteindelijk zal duidelijk worden dat de installatie is gestart. Zodra de installatie is afgerond, ziet u ongeveer het volgende:



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Microsoft Windows [Version 10.0.19043.1766]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI>pip install pandas
Collecting pandas
  Downloading pandas-1.4.3-cp39-win_amd64.whl (10.6 MB)
    |████████| 10.6 MB 72 kB/s
Collecting numpy>=1.18.5; platform_machine != "aarch64" and platform_machine != "arm64" and python_version < "3.10"
  Downloading numpy-1.23.0-cp39-cp39-win_amd64.whl (14.7 MB)
    |████████| 14.7 MB 111 kB/s
WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='pypi.org', port=443): Read timed out. (read timeout=15)": /simple/pytz/'): /simple/pytz/
Collecting pytz>=2020.1
  Using cached pytz-2022.1-py2.py3-none-any.whl (503 kB)
Collecting python-dateutil<=2.8.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    |████████| 247 kB 2.2 MB/s
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: numpy, pytz, six, python-dateutil, pandas
Successfully installed numpy-1.23.0 pandas-1.4.3 python-dateutil-2.8.2 pytz-2022.1 six-1.16.0
WARNING: You are using pip version 20.2.3; however, version 22.1.2 is available.
You should consider upgrading via the 'c:\users\msi\appdata\local\programs\python\python39\python.exe -m pip install --upgrade pip' command.

C:\Users\MSI>
```

Hoewel de gele regels (die beginnen met `WARNING`) anders doen vermoeden, was de installatie van het pakket in ons geval wel gewoon succesvol. Dat is te zien aan de regel die begint met `Succesfully installed`. De eerste geelgekleurde regel duidt op een verbroken verbinding (die uiteindelijk werd hersteld), de tweede geelgekleurde regel geeft aan dat we een verouderde versie van PIP gebruiken. Er wordt aangegeven dat u PIP kunt upgraden met het commando `-- upgrade pip`.



Daarmee bent u er nog niet: u moet dit pakket nog importeren.

10.4 Een pakket importeren

We gaven eerder in deze les al aan dat we twee soorten pakketten onderscheiden:

1. Pakketten die al geïnstalleerd zijn
2. Pakketten die u nog moet installeren

In beide gevallen is het - na de eventuele installatie – ook nog nodig om het pakket te importeren. Pas na het importeren kunt u het pakket gebruiken in uw eigen code.

10.4.1 Importeren van een geïnstalleerd pakket

In deze subparagraaf laten we u zien hoe u het zojuist geïnstalleerde pakket kunt importeren:

- Open VS Code.
- Open een map die u eerder aanmaakte, bijvoorbeeld NHA-prg).
- Klik op de knop “New Folder”.
- Geef de map een naam (bijvoorbeeld Les 10).
- Klik in de Menu Bar op “File”.
- Klik op “Open Folder”.
- Selecteer de map die u zojuist als laatste aanmaakte (Les 10).
- Klik in de Side Bar op de knop “New File”.
- Geef het bestand een naam. Eindig met de extensie .py om aan te geven dat het om een Python-bestand gaat (bijvoorbeeld test.py).
- Typ in het bestand `import pandas`.
- Sla de wijzigingen op.

U kunt nu gebruikmaken van het pakket. Voor het importeren van pakketten is er namelijk niet meer nodig dan het statement `import`, gevolgd door de naam van het pakket (in ons geval `pandas`). Dat moet u echter wel doen in ieder bestand waarin u het pakket wil gebruiken.

We gaan het pakket gebruiken:

- Breid de code in het bestand als volgt uit:

```
import pandas

datums = pandas.date_range("20220101", periods=6)

for datum in datums:
    print(datum.date())
```

- Sla de wijzigingen op.
- Voer het programma uit in de Terminal.

U ziet dan de volgende uitvoer:

```
PROBLEMS 1 OUTPUT TERMINAL JUPYTER DEBUG CONSOLE cmd + ×
```

```
Microsoft Windows [Version 10.0.19043.1766]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\Les 10>python test.py
2022-01-01
2022-01-02
2022-01-03
2022-01-04
2022-01-05
2022-01-06
```

```
C:\Users\MSI\Documents\NHA-prg\Les 10>
```

Door het toevoegen van deze regels hebben we op een relatief eenvoudige manier een aantal data gegenereerd, gebaseerd op een bepaalde startdatum. Daarvoor hadden we voorheen tamelijk wat functies en variabelen moeten definiëren.

10.4.2 Importeren van een standaardpakket

Zoals gezegd is het ook nodig om standaardpakketten te importeren. Dat doet u op precies dezelfde manier: via het statement `import`, gevolgd door de naam van het pakket.

In deze subparagraph importeren we het standaardpakket `os`, dat alles te maken heeft met het *operating system* waarmee u werkt (Windows of Mac).

- Vervang de code in het bestand door de volgende code:

```
import os
```



Gebruik kleine letters: `os` is in dit geval iets anders als OS!

- Sla de wijzigingen op.

Ook het gebruik van dit pakket zullen we demonstreren:

- Breid de code in het bestand als volgt uit:

```
import os
```

```
mijn_huidige_map = os.getcwd()  
print(mijn_huidige_map)
```

- Sla de wijzigingen op.
- Voer het programma uit in de *Terminal*.

Wat u als uitvoer ziet, zal afhankelijk zijn van de mapnaam die u eerder in deze les en cursus koos. In ons geval werken we met de hoofdmap NHA-prg en de submap Les 10:

A screenshot of a terminal window. At the top, tabs are labeled PROBLEMS, OUTPUT, TERMINAL, JUPYTER, and DEBUG CONSOLE. The TERMINAL tab is selected. The window title is "cmd". Below the tabs, the text "Microsoft Windows [Version 10.0.19043.1766]" and "(c) Microsoft Corporation. Alle rechten voorbehouden." is displayed. The command line shows the path "C:\Users\MSI\Documents\NHA-prg\Les 10>python test.py" and the output "C:\Users\MSI\Documents\NHA-prg\Les 10". A red box highlights the command and its output. The bottom of the terminal shows the prompt "C:\Users\MSI\Documents\NHA-prg\Les 10>".

In dit voorbeeld gebruiken we de functie `os.getcwd()`. Deze functie zoekt uit in welke map u zich op dat moment bevindt. Dit geeft hij terug als *string*.



Meer informatie over dit pakket vindt u op <https://docs.python.org/3/library/os.html>.

10.5 Zelf een pakket maken en importeren

In plaats van te werken met standaardpakketten of pakketten die zijn gemaakt door derden, is het ook mogelijk om zelf een pakket te maken. Dat is minder ingewikkeld dan het misschien klinkt.

- Voeg twee nieuwe bestanden toe aan de map waarin u momenteel werkt (bijvoorbeeld Les 10).
- Geef een bestand de naam 'hoofd_programma.py' en het andere bestand de naam 'mijn_pakket.py'.



U mag ook voor andere namen kiezen; wij kozen deze namen puur voor de duidelijkheid van dit voorbeeld.

- Voeg aan het bestand 'mijn_pakket.py' de volgende code toe:

```
speelgoed = ["bal", "x-box", "puzzel"]

def voeg_namen_samen(voornaam, achternaam)
    return voornaam + " " + achternaam
```

Zoals u ziet, zal dit bestand geen uitvoer hebben: er wordt niets geprint. We hebben slechts een *list* en een functie. Wanneer u het bestand 'mijn_pakket.py' (lees: het pakket *mijn_pakket*) nu importeert in het bestand 'hoofd_programma.py', kunt u zowel de variabele *speelgoed* als de functie *voeg_namen_samen* gebruiken.

- Voeg de volgende code toe aan het bestand 'hoofd_programma.py':

```
import mijn_pakket

print(speelgoed[0])
print(voeg_namen_samen("Jan", "Janssen"))
```



Merk op dat we de extensie .py weglaten bij het importeren van het pakket. Python gaat ervan uit dat, wanneer u iets importeert, dat altijd een Python-bestand zal zijn.

- Sla de wijzigingen op.
 - Voer het programma uit in de *Terminal*.

U krijgt dan een foutmelding te zien:

```
Microsoft Windows [Version 10.0.19043.1766]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\Les 10>python hoofd_programma.py
Traceback (most recent call last):
  File "C:\Users\MSI\Documents\NHA-prg\Les 10\hoofd_programma.py", line 3, in <module>
    print(speelgoed[0])
NameError: name 'speelgoed' is not defined

C:\Users\MSI\Documents\NHA-prg\Les 10>
```

Dat komt doordat het importeren van een eigen pakket iets anders werkt dan het importeren van een standaardpakket of een pakket van een ander.

- Pas de code in het bestand ‘hoofd_programma.py’ als volgt aan:

```
from mijn_pakket import speelgoed, voeg_namen_samen

print(speelgoed[0])
print(voeg_namen_samen("Jan", "Janssen"))
```

- Sla de wijzigingen op.
 - Voer het programma uit in de Terminal.

U krijgt dan de volgende uitvoer te zien:

A screenshot of a terminal window titled 'TERMINAL'. The window shows the following text:

```
Microsoft Windows [Version 10.0.19043.1766]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\Les 10>python hoofd_programma.py
bal
Jan Janssen

C:\Users\MSI\Documents\NHA-prg\Les 10>
```

Wanneer u zelf een pakket aanmaakt, moet u expliciet aangeven wat u uit dit pakket wil gebruiken in het hoofdprogramma. In dit geval is dat prima te doen: het gaat slechts om één variabele en één functie. Maar wat als uw pakket omvangrijker is?

In dat geval kunt u gebruikmaken van dit handigheidje:

```
from mijn_pakket import *
```

Door het gebruik van de asterisk (*) worden alle functies en variabelen in één keer geïmporteerd vanuit het pakket.

10.6 Veelgebruikte pakketten

Zoals we al eerder vermeldden, zijn er talloze pakketten voor u beschikbaar. Om daarin een beetje overzicht te bieden, behandelen we in de volgende subparagrafen enkele veelgebruikte pakketten.

10.6.1 Standaard aanwezige pakketten

We beginnen met modules die al in de *Python Standard Library* zitten (en die u dus niet hoeft te installeren voordat u ze importeert).

10.6.1.1 csv

Instanties die overzichten van transacties aan klanten beschikbaar stellen, bijvoorbeeld banken en openbaarvervoerbedrijven, bieden vaak de keus tussen een pdf- en een CSV-bestand. Terwijl een pdf-bestand een vast overzicht geeft, kan een CSV-bestand met behulp van de genoemde programma's ook gebruikt worden voor onder meer berekeningen.

Omdat de instanties niet weten met welk programma u die gegevens wilt verwerken, zijn afspraken gemaakt over hoe de gegevens moeten worden opgeslagen. CSV, dat staat voor *Comma Separated Values* (ofwel 'door komma's gescheiden waarden'), is een veelgebruikte indeling. Het databestand wordt dan gestructureerd door onder meer komma's.



De naam *Comma Separated Values* is voor Nederland en België niet helemaal juist. Hoewel wij - in tegenstelling tot bijvoorbeeld Amerikanen - wél komma's gebruiken in ons getalsysteem, gebruiken wij ze niet als scheidingstekens. In plaats daarvan gebruiken wij de puntkomma (;).

Zo'n bestand ziet er dan bijvoorbeeld zo uit:

```
Datum;Naam / Omschrijving;Rekening;Tegenrekening;Code;Af
Bij;Bedrag (EUR);Mutatiesoort;Mededelingen;Saldo na
mutatie;Tag
20220129;Supermarkt;NL01BANK0001234577;NL02BANK0001234579;GT;A
f;28,76;Online bankieren;Supermarkt;971,24;
20220129;Supermarkt;NL01BANK0001234577;NL02BANK0001234579;GT;A
f;10;Online bankieren;Supermarkt;961,24;
20220127;BELASTINGDIENST;NL01BANK0001234577;NL02BANK0001234587
;IC;Af;34;Incasso;Belasting Auto;927,24;
20220127;BELASTINGDIENST;NL01BANK0001234577;NL02BANK0001234587
;IC;Af;11;Incasso;Belasting Auto;916,24;
20220131;BUDGETENERGIE;NL01BANK0001234577;NL02BANK0001234586;I
C;Af;325;Incasso;Energie;591,24;
```

Met het pakket csv kunt u gegevens omzetten naar een CSV-bestand (exporteren) of CSV-bestanden uitlezen (importeren).



Meer informatie over dit pakket vindt u op <https://docs.python.org/3/library/csv.html>.

10.6.1.2 datetime

Dit pakket bevat klassen om te werken met datums en tijd. Deze klassen bieden een aantal functies om met datums, tijden en tijdsintervallen te bewerken, zoals:

Input	Output
<pre>from datetime import date d = date(2002, 12, 31) d.replace(day=26) print(d)</pre>	2002-12-31

Het werken met datums en tijd is niet gemakkelijk. Bekijk de volgende (Engelstalige) video voor meer informatie over het werken met dit pakket:



Het werken met datums en tijd is niet gemakkelijk. In de digitale les op Plaza vindt u op deze plek een (Engelstalige) video over het werken met dit pakket.



Meer informatie over dit pakket vindt u op: <https://docs.python.org/3/library/datetime.html>.

10.6.1.3 random

Dankzij dit pakket kunt u gebruikmaken van verschillende *pseudo-random number generators*. Dit betekent dat de getallen dus niet willekeurig worden gegenereerd, maar volgens een bepaalde procedure. Voor de meeste toepassingen is dat echter goed genoeg.

Enkele voorbeelden van *generators* zijn:

Input	Output
<pre>from random import randint i = randint(0, 100) print(i) i = randint(0, 100) print(i) i = randint(0, 100) print(i)</pre>	22 77 37

Input	Output
<pre>import random smaken = ['citroen', 'peer', 'appel'] print(random.choice(smaken)) print(random.choice(smaken)) print(random.choice(smaken))</pre>	<pre>peer peer citroen</pre>



Meer informatie over dit pakket vindt u op <https://docs.python.org/3/library/random.html>.

10.6.2 Nog te installeren pakketten

In deze subparagrafen richten we ons op enkele populaire pakketten die u nog wél moet installeren voordat u ze kunt importeren.

10.6.2.1 Pillow

Dit pakket is een bijgewerkte versie van de *Python Image Library (PIL)*. Het ondersteunt een reeks eenvoudige en geavanceerde functies voor beeldmanipulatie. Met andere woorden: het stelt u in staat om aan de slag te gaan met grafische bestanden (zoals JPG-, PNG- en BMP-bestanden).

The screenshot shows the PyCharm IDE interface. On the left, the code editor displays a Python script named `image1.py` with the following content:

```
phytutorials - image1.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
phytutorials > images > image1.py
Project: phytutorials > image1.py < Run
1 # Import image module from PIL
2 from PIL import Image
3
4 # Open the image file
5 img = Image.open('bird1.png')
6 # Display the image
7 img.show()
```

On the right, the run output window shows the output of running the script, which includes the path to the image file and the command to exit. Below the run output, there is a preview of the image itself, which is a vibrant orange and black bird perched on a branch.

Dankzij het pakket Pillow kunt u afbeeldingen laden en manipuleren.



Meer informatie over dit pakket vindt u op <https://pypi.org/project/Pillow/>.

10.6.2.2 NumPy

Als u actief bent in de *data science* (of dat gaat zijn), kunt u eigenlijk niet zonder dit pakket.

NumPy is een Python-bibliotheek die wordt gebruikt voor het werken met *arrays*. Het heeft ook functies voor het werken in het domein van lineaire algebra, fourier-transformatie en matrices. Simpeler gezegd: dit pakket stelt u in staat om extra wiskundige bewerkingen uit te voeren.



Meer informatie over dit pakket vindt u op www.numpy.org.

10.6.2.3 Matplotlib

Met het pakket Matplotlib kunt u verschillende soorten 2D-diagrammen en -grafieken maken. Bijvoorbeeld:

The screenshot shows a Jupyter Notebook cell with Python code for creating a survey results visualization. The code defines a function `survey` that takes `results` (a dictionary) and `category_names` (a list of categories). The visualization is a stacked horizontal bar chart titled "Figure 1". The legend indicates five categories: "Strongly disagree" (red), "Disagree" (orange), "Neither agree nor disagree" (yellow), "Agree" (green), and "Strongly agree" (dark green). The data for six questions is as follows:

Question	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Question 1	10	15	17	32	26
Question 2	26	22	29	10	13
Question 3	35	37	7	2	19
Question 4	32	11	9	15	33
Question 5	21	29	5	5	40
Question 6	8	19	5	30	38

The screenshot shows a Jupyter Notebook cell with Python code for creating a gradient plot of an electrical dipole potential. The code imports `Triangulation`, `UniformTriRefiner`, and `CubicTriInterpolator` from `matplotlib.tri`, and `numpy`. It defines a function `dipole_potential` that calculates the potential V at position (x, y) based on the formula $V = \frac{(np.max(z) - z)}{(np.max(z) - np.min(z))}$. The visualization is a vector field plot titled "Figure 1" with the subtitle "Gradient plot: an electrical dipole". The plot shows concentric contour lines around a central point, with arrows indicating the direction of the gradient.

U moet even in de documentatie van dit pakket duiken om precies uit te zoeken hoe alle functies werken, maar neemt u die moeite, dan heeft u vele mogelijkheden om op basis van uw data mooie grafieken aan te maken.



Meer informatie over dit pakket vindt u op www.matplotlib.org.

10.6.2.4 Requests

Met dit pakket kunt u met Python HTTP-verzoeken verzenden. Daardoor zou u na installatie van dit pakket vanuit uw Python-programma onder andere op het internet kunnen surfen.

U vraagt zich wellicht af wanneer dat van pas kan komen. Twee voorbeelden:

- Als u uw browser gebruikt om naar een webpagina te surfen, krijgt u die webpagina te zien zoals de eigenaar van die pagina wil. Er is echter allerlei verborgen informatie die u normaliter niet te zien krijgt.

U kunt zich wellicht voorstellen dat het in sommige toepassingen handig kan zijn om die verborgen informatie toch boven water te krijgen. Deze module kan dat voor u regelen.

- U kunt vanuit uw Python-programma ook bijvoorbeeld snel even controleren of een bepaalde webpagina eigenlijk wel bestaat:

```
>>> requests.get('https://api.github.com') <Response [200]>
```

Als u 200 terugkrijgt, weet u dat de website bestaat.



Meer informatie over dit pakket vindt u op <https://pypi.org/project/requests> en op <https://requests.readthedocs.io/en/latest>.

10.7 Project 'IJssalon'

Ook in deze les werken we weer even aan ons overkoepelende project. We gaan verder waar we gebleven zijn. Als het goed is, bevat het bestand 'helper.py' op dit moment onder meer de volgende code:

```
def decoreer(tekst=""):
    tekst="header"
    lengte = len(tekst) + 4
    print()
    print(lengte * "*")
    print(f"* {tekst} *")
    print(lengte * "*")
    print()

def fooi_pp(bedrag, personen):
    bedrag_pp = bedrag/personen
    return f"Het bedrag per persoon is {bedrag_pp} euro"
```

Door wat u deed in de vorige les, bevat het bestand waarschijnlijk nog meer code. **Die code dient u te verwijderen.**

U weet inmiddels dat het mogelijk is om een functie te importeren in een ander Python-bestand. Dat doen we in ons project.

- Ga naar VS Code.
- Open de map die u eerder aanmaakte voor de uitwerking van deze casus (bijvoorbeeld IJssalon-start).
- Open het bestand 'helper.py'.

We gaan aan dit bestand een nieuwe functie toevoegen. Net als in de vorige les zullen we u niet letterlijk aangeven welke code u moet toevoegen, maar plaatsen we slechts een beschrijving (op basis waarvan u de code uitbreidt). Verderop in de les tonen we u wat de juiste code had moeten zijn.

De beschrijving van de toe te voegen functie is als volgt:

- De naam van de functie is `onderstreep()`.
- De functie heeft één parameter, genaamd `tekst`. Deze parameter heeft de standaardwaarde "".
- Op de eerste regel van deze functie zet u een variabele met de naam `uit`. Dit is een lege `list` (dus `uit = []`).
- U gebruikt de `method .append()` om de waarde van de parameter `tekst` als element toe te voegen aan de nu nog lege `list` `uit`.
- U bedenkt zelf een manier om als tweede element in de `list` `uit` een `string` toe te voegen die bestaat uit een aantal `=`-tekens. Het aantal `=`-tekens moet gelijk zijn aan het aantal karakters in de parameter `tekst`.
- De functie geeft de waarde van de variabele `uit` als uitvoer.

We gaan deze functie importeren in een nieuw bestand. Ook nu houden we het slechts bij een beschrijving:

- Het nieuwe bestand heeft de naam 'hoofd_programma.py'.
- In dit nieuwe bestand importeert u de functie onderstreep() uit het bestand 'helper.py'.
- Definieer in het nieuwe bestand een variabele met de naam uitvoer.
- Geef deze variabele de waarde onderstreep("AANBIEDING").
- Gebruik de method .append() om de volgende string toe te voegen aan de list uitvoer:

Aardbeienijs, emmertje van 5 liter: 5 euro

- Gebruik de method .append() om de volgende string toe te voegen aan de list uitvoer:

Slagroom, spuitbus van 1 liter: 2 euro

- Print een lege regel (met behulp van print()).
- Creëer een for-loop om door de waarden van uitvoer te loopen en de elementen van uitvoer één voor één uit te printen.

Schrijf beide codes en probeer deze uit door het bestand 'hoofd_programma.py' uit te voeren in de Terminal. Voer verschillende waarden in om te kijken of uw code goed werkt. Zo niet: probeer de debug-strategieën uit voordat u spekt wat de juiste code had moeten zijn.

De juiste extra code in het bestand 'helper.py' had moeten zijn:

```
def onderstreep(tekst=""):
    uit = []
    uit.append(tekst)
    uit.append(len(tekst) * "=")
    return uit
```

De juiste extra code in het bestand 'hoofd_bestand.py' had moeten zijn:

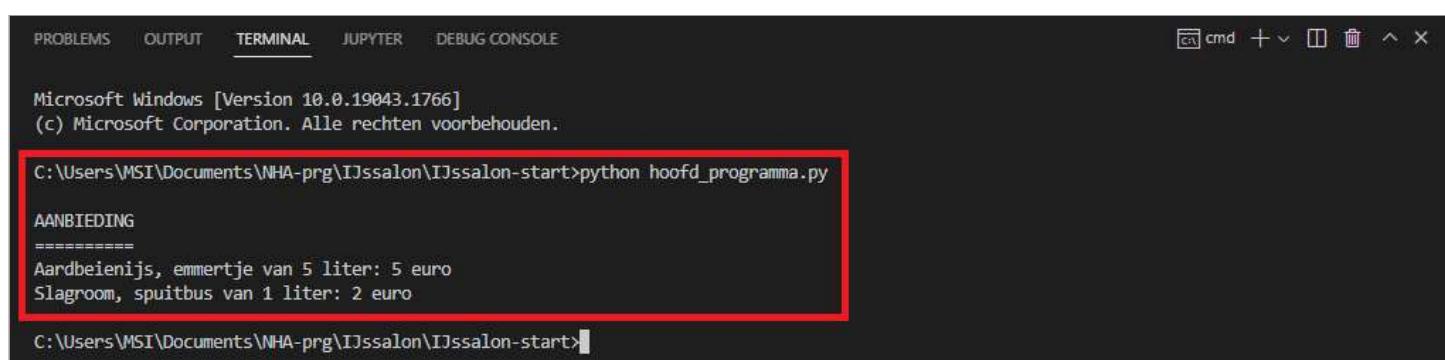
```
from helper import onderstreep

uitvoer = onderstreep("AANBIEDING")
uitvoer.append("Aardbeienijs, emmertje van 5 liter: 5 euro")
uitvoer.append("Slagroom, spuitbus van 1 liter: 2 euro")

print()

for el in uitvoer:
    print(el)
```

Wanneer u deze code uitvoert in de Terminal, krijgt u de volgende uitvoer:



```
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE cmd + ×

Microsoft Windows [Version 10.0.19043.1766]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\IJssalon\IJssalon-start>python hoofd_programma.py

AANBIEDING
=====
Aardbeienijs, emmertje van 5 liter: 5 euro
Slagroom, spuitbus van 1 liter: 2 euro

C:\Users\MSI\Documents\NHA-prg\IJssalon\IJssalon-start>
```

Misschien denkt u:

Is dit niet erg veel werk voor slechts vier regeltjes uitvoer?

Ja, dat klopt. Maar u zorgt er op deze manier wel voor dat u delen van uw code kunt hergebruiken, ook in andere bestanden.

10.8 Samenvatting

In deze les heeft u geleerd hoe u Python-functies en -variabelen kunt importeren van pakketten. Dit zijn kant-en-klare uitbreidings van uw projecten, die u vrij kunt gebruiken in uw Python-project. U hoeft dan niet een deel van uw programma zelf te schrijven.

U weet nu dat er in feite twee soorten pakketten zijn:

1. Pakketten die al geïnstalleerd zijn (en onderdeel zijn van de *Python Standard Library*)
2. Pakketten die u nog moet installeren

Pakketten installeren kan op verschillende manieren. Wij lieten u zien hoe u dit doet met pakketmanager PIP.

Na de eventuele installatie is het nog nodig om het pakket te importeren. Dat doet u met behulp van het statement `import`, gevolgd door de naam van het pakket.

Het is ook mogelijk om zelf een pakket te maken en functies en/of variabelen hieruit te gebruiken in uw ‘hoofdcode’. In dat geval werkt importeren iets anders: u moet dan expliciet aangeven wat u wil gebruiken. Met behulp van een asterisk (*) kunt u alle functies en variabelen in één keer importeren. Na het importeren kunt u de zelfgeschreven functies en variabelen gebruiken alsof ze zich gewoon in uw ‘hoofdcode’ bevinden.

10.9 Vraagmoment

Heeft u vragen over deze les? Dan kunt u die stellen via een vraagmoment.

U herkent een vraagmoment aan het spreekballonnetje dat op Plaza in het overzicht “Lessen” bij een les staat. Door op deze spreekballoon te klikken, komt u op een nieuwe pagina, waar u uw vraag in het tekstvak kunt typen.



Het tekstvak mag maximaal vijfhonderd tekens bevatten. Zorg er dus voor, dat u uw vraag/vragen bondig formuleert.

Nadat u op “Verstuur vraag” heeft geklikt, wordt uw vraag automatisch naar uw docent verstuurd. Zodra uw docent de vraag heeft beantwoord, verschijnt er op uw dashboard bij de opleiding een spreekballoon.

U kunt per les één keer vragen stellen. Verzamel dus uw vragen over de les, voordat u deze instuurt. Als u gebruik heeft gemaakt van het vraagmoment, verdwijnt het spreekballonnetje bij de les.

10.10 Oefenopgaven

De volgende opgaven werkt u als oefenopgaven voor uzelf uit. De uitwerkingen van deze opgaven kunt u meteen zelf controleren.

Wat is een pakket?

[Bekijk antwoord](#)

Welke twee soorten pakketten kennen we?

[Bekijk antwoord](#)

Wat is PyPi?

[Bekijk antwoord](#)

Waarvoor staat de afkorting PIP en wat is het?

→ Bekijk antwoord

Welk commando gebruikt u om een pakket uit de *Python Standard Library* te installeren?

→ Bekijk antwoord

Welk commando gebruikt u om een pakket uit de *Python Standard Library* te importeren?

→ Bekijk antwoord

U heeft een Python-bestand aangemaakt met de naam 'wiskunde.py'. Dit pakket bevat onder meer de functie `maal_twee()`. Welk commando gebruikt u wanneer u deze functie `wil` importeren in uw 'hoofdcode'?

→ Bekijk antwoord

U heeft een Python-bestand aangemaakt met de naam 'helper.py'. Welk commando gebruikt u als u alle functies en variabelen in dit bestand in één keer `wil` importeren in uw 'hoofdcode'?

→ Bekijk antwoord

U gebruikt het volgende commando om een functie uit een zelfgemaakt pakket te importeren in uw 'hoofdcode':

```
from mijn_bestand import vermenigvuldigen(a,b)
```

Waarom zult u een foutmelding krijgen?

→ Bekijk antwoord

U gebruikt het volgende commando om een functie uit een zelfgemaakt pakket te importeren in uw 'hoofdcode':

```
import willekeurige_cijfers.py
```

Waarom zult u een foutmelding krijgen?

→ Bekijk antwoord

10.11 Huiswerkopgaven

Maak deze huiswerkopgaven en stuur ze via Plaza naar uw docent. Deze opgaven worden nagekeken en voorzien van een cijfer. Mocht u een onvoldoende cijfer behalen, dan krijgt u een herkansing.

De volgende stappen vormen samen de huiswerkopgaven:

1. 1. Voeg een nieuw bestand toe aan de map 'IJssalon-start'. Geef dit bestand de naam 'boekhouding.py'.

2. Plaats in het bestand 'boekhouding.py' een *dictionary* genaamd `inkomsten`. Zorg ervoor dat deze *dictionary* de volgende *key-value pairs* bevat:

Key	Value
Aardbeien-ijs-totaal	1000
Vanille-ijs-totaal	2000
Chocolade-ijs-totaal	1500
Waterijsjes-totaal	750

3. Plaats in het bestand 'helper.py' een functie genaamd `som()`. Zorg ervoor dat deze functie een enkele *dictionary* als invoer accepteert, vervolgens alle waarden (*values*) optelt en de som als resultaat teruggeeft.

4. Importeer aan het begin van het bestand 'boekhouding.py' alle functies en variabelen uit het bestand 'helper.py'.

5. Plaats in het bestand 'boekhouding.py' een variabele genaamd `totaal_inkomsten`. Gebruik de functie `som()` (uit 'helper.py') om de totale som van inkomsten toe te kennen aan deze variabele.

6. Voeg een nieuw bestand toe aan de map 'IJssalon-start'. Geef dit bestand de naam 'presentatie.py'.

7. Plaats in het bestand 'presentatie.py' een functie genaamd `presenteer()`. Zorg ervoor dat deze functie een *dictionary* en de parameter `totaal` als invoer accepteert. Als voorlopige code in de functie gebruikt u `pass`.

8. Vervang de voorlopige code `pass` door een code die ervoor zorgt dat de functie `presenteer()` informatie op een bepaalde manier uitvoert. Volg daarbij het volgende voorbeeld:

Invoer:

```
mijn_dict = {'vis' : 10, 'vlees': 25, 'overig' : 15}

totaal = 50
```

Uitvoer:

```
vis : 10 euro
vlees : 25 euro
overig : 15 euro
=====
totaal : 50 euro
```

9. Importeer aan het begin van het bestand 'boekhouding.py' alle functies en variabelen uit het bestand 'presentatie.py'.

10. Gebruik in het bestand 'boekhouding.py' de functie `presenteer()` met de argumenten `inkomsten` en `totaal_inkomsten` (uit 'boekhouding.py') om de volgende uitvoer te bewerkstelligen:

```
Aardbeien-ijs-totaal : 1000 euro
Vanille-ijs-totaal : 2000 euro
Chocolade-ijs-totaal : 1500 euro
Waterijsjes-totaal : 750 euro
=====
Totaal : 5250 euro
```

11. Importeer aan het begin van het bestand 'boekhouding.py' het pakket `csv` (uit de *Python Standard Library*).

12. Voeg de volgende code toe aan het bestand 'boekhouding.py':

```
with open('boekhouding.csv', 'w', newline='') as csvfile:
    for key, value in inkomsten.items():
```

```
writer = csv.writer(csvfile, delimiter=';')  
writer.writerow([key,value])
```

13. Maak een nieuwe lokale versie van het project aan. Gebruik daarvoor twee commando's. Plaats bij het tweede commando een passende tekst tussen aanhalingstekens.

14. Update uw *remote repo* op *GitHub*.

Stuur de URL van uw *remote repo* op *GitHub* via Plaza naar uw docent. Uw docent kan dan niet alleen het eindresultaat zien, maar ook alle tussentijdse wijzigingen.