Opdrachten week 1 les 1 – Introductie Python

In deze les maak je kennis met de programmeertaal Python en de geïntegreerde ontwikkelomgeving (in het Engels: Integrated Development Environment of kortweg IDE) Thonny. Je leert hoe je:

- de Python Shell kunt gebruiken als een geavanceerde rekenmachine;
- variabelen, expressies en statements kunt gebruiken in Python programma's;
- verschillende soorten fouten (syntax errors, runtime errors en semantic errors) kunt opsporen in een Python programma (dit wordt debuggen genoemd);
- eenvoudige Python functies kunt gebruiken.

In de volgende les leer je om zelf Python functies te schrijven.

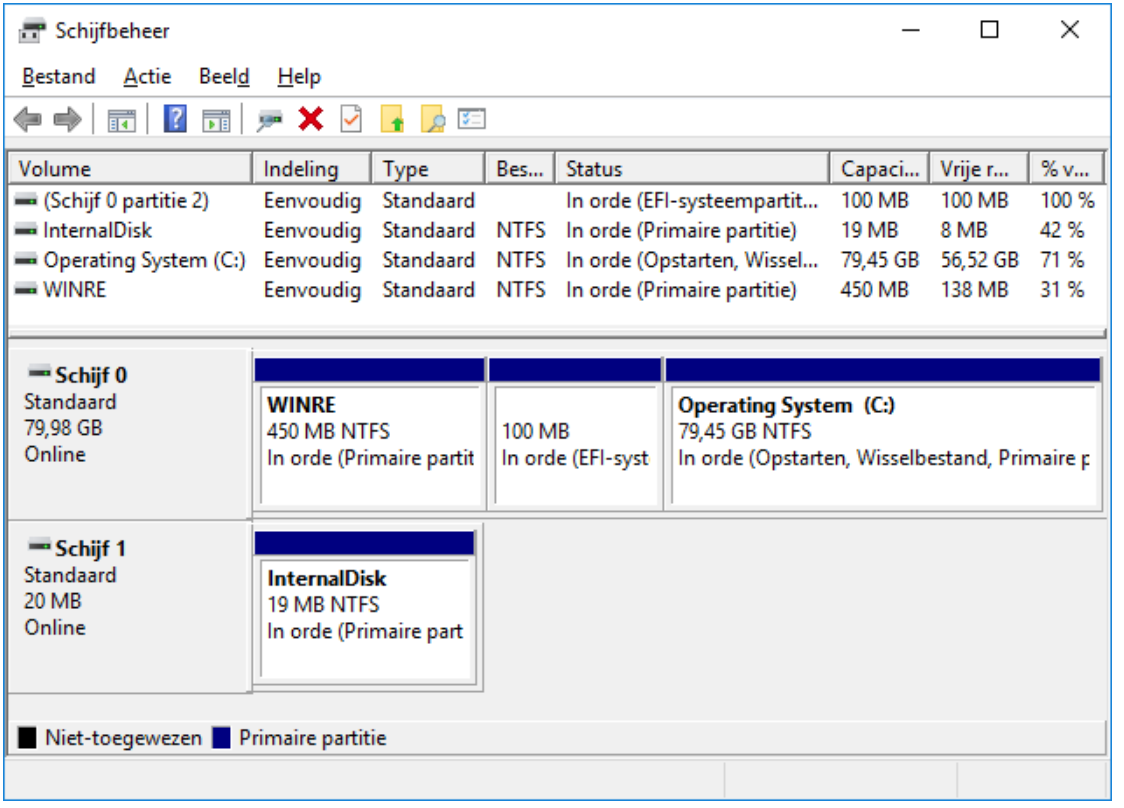
In deze les ga je programmeren in de programmeertaal Python. Je gebruikt daarbij het volgende boek: Allen B. Downey. *Think Python: How to Think Like a Computer Scientist*. 2de ed. Green Tea Press, 2016. ISBN: 978-1-4919-3936-9. URL: http://greenteapress.com/wp/think-python-2e/. Dit boek is gratis te downloaden via de bovenstaande URL. Er is tevens een Nederlands boek geschreven wat je kunt gebruiken als naslagwerk: Pieter Spronck. *De Programmeursleerling – Leren coderen met Python 3*. 2016. URL: http://www.spronck.net/pythonbook/dutchindex.xhtml.

Voorbereiding

Als beginnende programmeur is het erg belangrijk om te begrijpen hoe een besturingssysteem zoals Windows omgaat met bestanden. Wat zijn mogelijke locaties om een bestand op te slaan? Wat zijn bestandsrechten, mappen, schijven en partities? Wat is NTFS of FAT? De opdrachten in dit hoofdstuk zullen je helpen hier een beeld van te vormen.

In een computerwinkel kun je een harde schijf (hard disk) kopen. Tegenwoordig zijn Solid State Disks (SSDs) populair omdat ze erg snel zijn vergeleken met de originele magnetische harde schijven. Een schijf heeft vervolgens ruimte voor een aantal bytes (bijvoorbeeld 80 GB). Deze ruimte is softwarematig opgedeeld in delen genaamd *partities*, elk met een eigen grootte (bijvoorbeeld 450 MB).

Elke partitie is gestructureerd volgens een bestandssysteem. Een bestandssysteem is een methode om een partitie in te delen. Het bestandssysteem beschrijft en bepaalt hoe en waar bestanden worden opgeslagen. Veelgebruikte bestandssystemen zijn FAT32, NTFS (Windows), EXT4 (Linux), HFS+ (Mac).



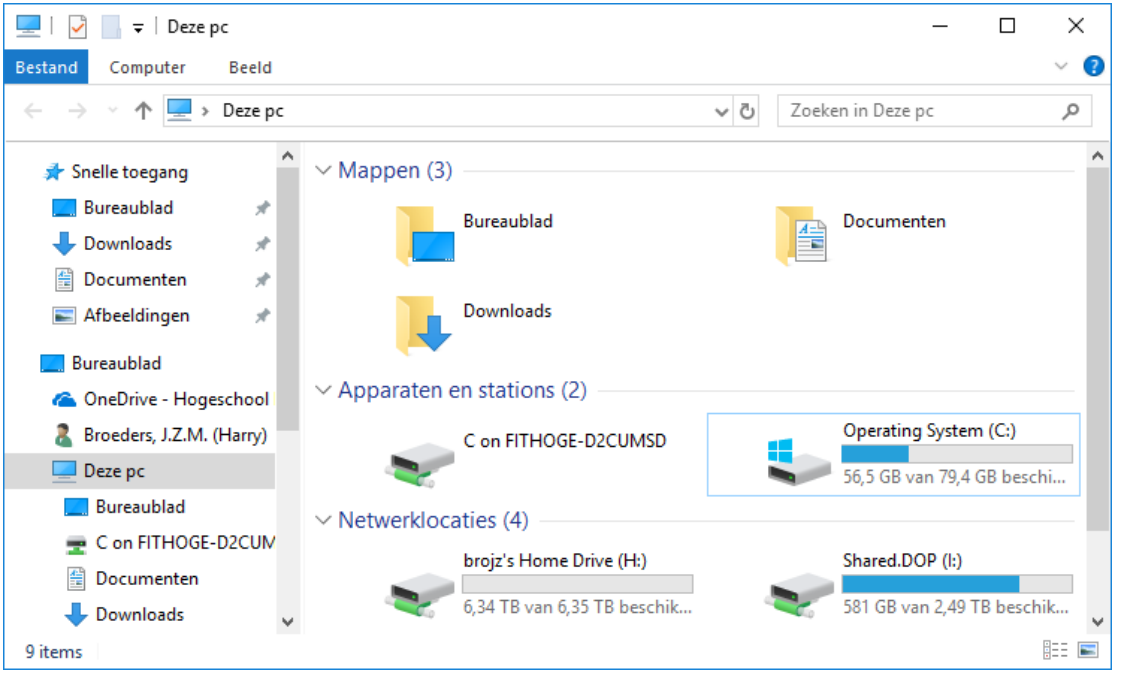
Figuur 1: De opdeling in partities op een willekeurig systeem.

Het systeem van figuur 1 heeft 3 partities op een schijf van 79,98 GB. Een van de partities heeft het label *Operating System* en is gekoppeld aan *C:*. De term *C-schijf* verwijst dus eigenlijk naar een deel op de harde schijf waarop Windows is geïnstalleerd. Op een Mac en op Linux heeft dit de naam *root* en het pad `/`.

Laten we kijken naar het systeem waarop je dit leest.

1.1.1 Open Windows verkenner (file explorer) door met je rechtermuisknop op het Windows-logo te klikken, en te kiezen voor Verkenner (File Explorer). Een alternatieve manier is om op het bureaublad op *Deze pc* te dubbelklikken.

Als het goed is zie je aan de rechterkant van het venster wat opent drie categorieën: 'Mappen', 'Apparaten en stations' en 'Netwerklocaties'. Aan de linkerkant zie je onder *Deze Computer* dezelfde opties zonder categorisatie. Zie figuur 2 voor een voorbeeld.



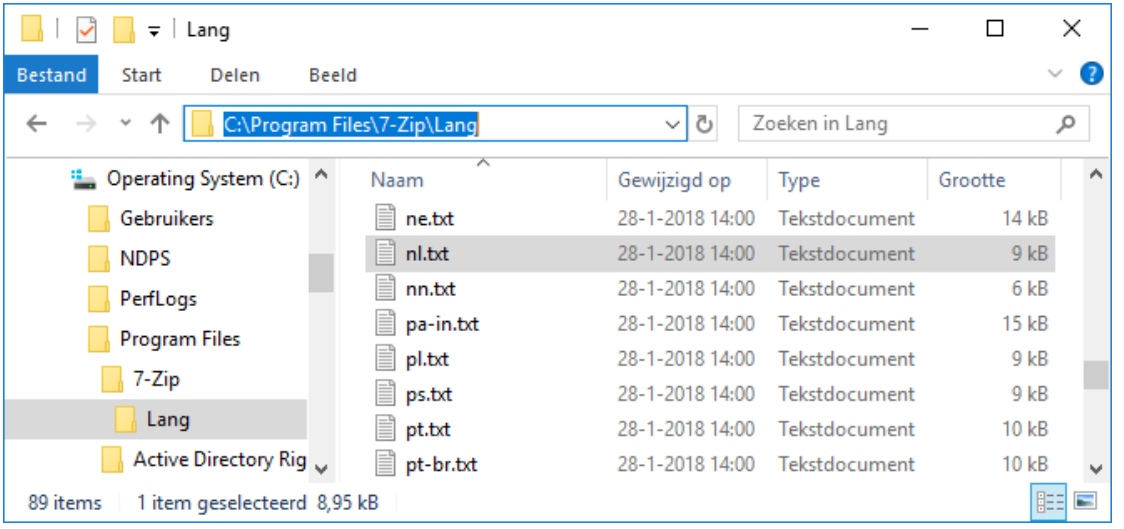
Figuur 2: Overzicht bij het openen van Deze pc op een Nederlandstalige computer.

Onder *Apparaten en stations* zie je onder andere de schijf *Operating System (C:)*. Bekijk de eigenschappen van deze schijf door met je rechtermuisknop naar Eigenschappen te gaan en schrijf op hoe groot deze partitie is en welk bestandssysteem ervoor wordt gebruikt. Optioneel: Als je op je eigen laptop werkt, kun je ook *Disk Management* openen om je eigen partities te zien; druk op Windows + R en type in `diskmgmt.msc`.

1.1.2 Het benaderen van bestanden gebeurt volgens een *pad*. Dit pad volgt een boomstructuur van mappen (Engels: directories) om bij een bestand te komen. Figuur 3 heeft bijvoorbeeld bestand `nl` geselecteerd met als pad `c:\Program Files\7-Zip\Lang\nl.txt`. De gevolgde *mappenboom* is in het linkerdeel van figuur 3 te zien.

Zoek nu zelf uit aan welk pad de map *Mijn Documenten* is gekoppeld.

1.1.3 Bestanden en mappen hebben ook zogenaamde rechten; bepaalde gebruikers van het systeem hebben (wel of niet) toegang tot bepaalde mappen en bestanden. Probeer bijvoorbeeld naar `C:\Windows\System` te bladeren m.b.v. Windows Verkenner en hier een nieuwe map aan te maken; er wordt gevraagd naar administratie rechten! De



Figuur 3: Bestand `nl` heeft de extensie `txt` en een pad.

administrator (Windows) of *root* (Linux/MacOS) heeft altijd de hoogste rechten van een systeem.

Standaard heeft een gebruiker wel schrijfrechten op zijn eigen mappen onder `C:\Users\gebruikersnaam` (Windows) of `/home/gebruikersnaam` (Linux/MacOS). Zoek uit wat het pad is van jouw persoonlijke map op deze computer. Maak hierin een nieuwe map aan met de naam `EMS10`. Maak in de map EMS10 nu een nieuw tekstbestand aan met de naam `EMS10`. Welke extensie heeft dit bestand? Wat gebeurd er als je de naam wijzigt naar `EMS10.pdf`? Kun je dit PDF-bestand openen?

1.1.4 Bestandsextensies zijn de letters achter een punt (`.`). Deze letters geven aan wat voor soort informatie in het bestand is opgeslagen. Windows maakt gebruik van de bestandsextensie om het juiste programma te starten om het bestand mee te openen. Elk bestand is gestructureerd volgens een bepaald formaat; een foto (`.jpg`) is bijvoorbeeld anders gecodeerd dan een Word-bestand (`.docx`). Zoek uit welke extensies worden gebruikt voor een Python-bestand.

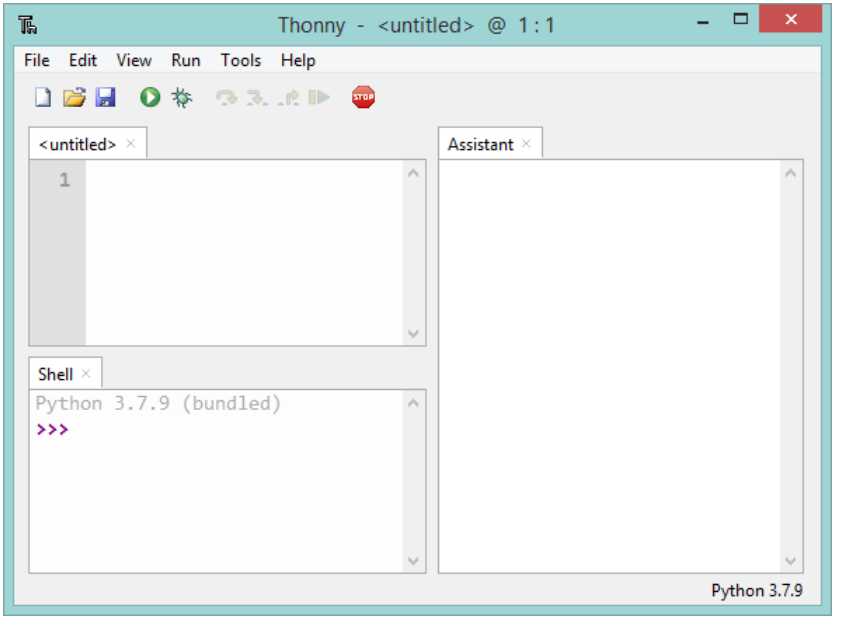
Je bent nu in staat een bestand te vinden op basis van een pad en zelf een bestand aan te maken in een opgegeven pad, hiermee ben je klaar om te beginnen met programmeren!

Omdat je zowel thuis als op school aan EMS10 zult werken is het handig om alle bestanden op je OneDrive te zetten zodat je er altijd bij kunt.

Opdrachten

Je gaat het programma Thonny (zie http://thonny.org/) gebruiken om te programmeren in Python. Dit programma is gratis te gebruiken en is speciaal gemaakt voor beginnende programmeurs.

1.1.5 Download het boek *Think Python: How to Think Like a Computer Scientist* van http://greenteapress.com/thinkpython2/thinkpython2.pdf en open het programma Thonny in de Liquit Workspace of installeer het programma Thonny op je eigen pc.



Figuur 4: Het programma Thonny.

Als je het programma Thonny uitvoert, zie je drie deelwindows zoals in figuur 4. Beide deelwindows aan de linkerkant zijn invoervelden. Het bovenste invoerveld kun je gebruiken om Python scripts (programma's) in te schrijven. In het onderste invoerveld, de zogenoemde Shell, kun je Python commando's invoeren en deze commando's worden meteen uitge-

voerd door de Python interpreter[1]. Het is erg handig om eerst met Python commando's te experimenteren in de Shell voordat je ze in een script (programma) gaat gebruiken.

Je kunt de Python Shell ook gebruiken als een geavanceerde rekenmachine.

1.1.6 Gebruik de Python Shell om uit te rekenen hoeveel seconden er in een etmaal zitten ($24 \times 60 \times 60$).

1.1.7 Gebruik de Python Shell om 2^{10} uit te rekenen (het juiste antwoord is 1024). Zoek zelf op (in het boek of op het internet) hoe je machten kunt berekenen in de Python Shell.

1.1.8 Gebruik je eigen rekenmachine om 2^{3000} uit te rekenen. Wat gebeurt er? Gebruik de Python Shell om 2^{3000} uit te rekenen.

1.1.9 Gebruik de Python Shell om `4e6 + 2e-5` uit te rekenen? Wat betekent `4e6` en `2e-5`?[2]

1.1.10 Kan Python ook rekenen met complexe getallen? Gebruik de Python Shell om $(3+2j)^2$ uit te rekenen.

1.1.11 Probeer sin(1) uit te rekenen door `sin(1)` in te typen in de Python Shell. Wat gebeurt er? In het window genaamd 'Assistant' geeft Thonny je enkele tips. Zijn die zinvol?

Om bepaalde commando's te kunnen gebruiken moet je in veel gevallen een zogenoemde module[3] importeren. Als je wiskundige functies en constanten wilt gebruiken moet je de module `math` importeren. Dit doe je door `import math` in te typen in de Python Shell. Vervolgens kun je de functies en constanten uit de module `math` bijvoorbeeld als volgt gebruiken: `math.sin(1)`.

---

[1] Een interpreter is een computerprogramma dat de broncode van computerprogramma's vertaalt naar een voor de processor begrijpelijke vorm, en die ook meteen uitvoert. Dit in tegenstelling tot een compiler, die programma's opslaat in dergelijke vorm zodat ze later uitgevoerd kunnen worden. Bron: https://nl.wikipedia.org/wiki/Interpreter.

[2] De notatie `4e6` betekent 4×10^6 en `2e-5` betekent 2×10^{-5}. Dit wordt de wetenschappelijke notatie genoemd, zie https://nl.wikipedia.org/wiki/Wetenschappelijke_notatie#Programmeertalen.

[3] Een module is een bestand dat Python definities en commando's bevat.

1.1.12 Gebruik de Python Shell om $\sin(\frac{1}{4}\pi)$ uit te rekenen.

1.1.13 Gebruik de Python Shell om $e^{j\frac{1}{4}\pi}$ uit te rekenen.

1.1.14 De identiteit van Euler luidt: $e^{j\pi} + 1 = 0$. Gebruik de Python Shell om de identiteit van Euler te controleren door $e^{j\pi} + 1$ te berekenen. Je zult zien dat het antwoord niet 0 is zoals je zou verwachten. Wie heeft er gelijk, Euler of Python? Wat concludeer je hieruit?[4]

Lees nu hoofdstuk 1 tot en met paragraaf 1.1 van het boek. Paragraaf 1.2 van het boek kun je overslaan, door Thonny te gebruiken kun je eenvoudig Python code uitvoeren.

1.1.15 Welke vijf basisinstructies komen in vrijwel elke programmeertaal voor?

Er zijn twee versies van Python in gebruik: Python 2 en Python 3. Wij gebruiken Python 3. Let er bij het zoeken naar informatie op internet goed op dat je informatie over de juiste versie van Python gebruikt.

Lees nu paragraaf 1.3 en 1.4 van het boek.

1.1.16 Type het commando om de tekst `Hallo wereld!` te printen in de Python Shell. Tip: zet de toetsenbordinstellingen in Windows op ENG (English (United States) US keyboard) zodat je karakters zoals `'` en `"` eenvoudig kunt invoeren.

1.1.17 Bereken `2.5 * 2` in de Python Shell. Let op: er staat 2 *punt* 5.

1.1.18 Bereken `2,5 * 2` in de Python Shell. Let op: er staat 2 *komma* 5. Wat concludeer je hieruit?

Lees nu paragraaf 1.5 van het boek.

[4] Het echte antwoord is nul, maar Python geeft *bijna* nul. Dat komt omdat de computer getallen moet afronden op een aantal cijfers na de komma. De getallen e en π zijn irrationale getallen (zie https://nl.wikipedia.org/wiki/Irrationaal_getal) die niet exact in een computer kunnen worden opgeslagen. Daardoor ontstaat er dus een afrondingsfout. Python 'denkt' dus dat het bijna nul is, maar het is dus echt nul.

1.1.19 Gebruik de Python Shell om uit te zoeken wat het type van `3+2j` is.

Paragraaf 1.6 van het boek mag je overslaan. Deze paragraaf beschrijft het verschil tussen natuurlijke talen zoals Nederlands en programmeertalen zoals Python. Als je dat interessant vind, mag je deze paragraaf in je eigen tijd lezen.

Paragraaf 1.7 van het boek mag je ook overslaan. Deze paragraaf definieert de term *debuggen*: het opsporen van fouten, ook wel *bugs* genoemd, in programma's. Het opsporen van fouten in programma's is vaak lastig en soms frustrerend, maar je zult zien dat het programma Thonny je hier goed bij kan helpen.

Paragraaf 1.8 van het boek definieert de verschillende (vak)termen die in hoofdstuk 1 gebruikt worden. Het kan handig zijn om deze paragraaf te raadplegen als je niet meer weet wat met een bepaalde (vak)term bedoeld wordt.

Paragraaf 1.9 van het boek bevat enkele oefeningen. De oefeningen die de moeite waard zijn, zijn hieronder in het Nederlands vertaald.

1.1.20 Wat is de uitkomst van de Python expressies `-+-+-+-+3` en `-+-+-+-+-+3`? Verklaar de gevonden antwoorden.

1.1.21 Wat gebeurt er als je in de Python Shell twee getallen intypt zonder operator ertussen? Begrijp je de foutmelding[5]?

Lees nu hoofdstuk 2 tot en met paragraaf 2.3 van het boek. In het boek wordt de term 'state diagram' gebruikt voor een overzicht van een aantal variabelen met hun waarde en hun naam. In de elektrotechniek heeft de term 'state diagram' (toestandsdiagram), zoals je weet, een heel andere betekenis[6].

Door variabelen te gebruiken kun je berekeningen stap voor stap uitvoeren. In opdracht 1.1.6 heb je berekend hoeveel seconden er in een etmaal zitten. Dit kan je ook stap voor stap uitrekenen door achtereenvolgens in te voeren:

```
>>> seconden_in_minuut = 60
```

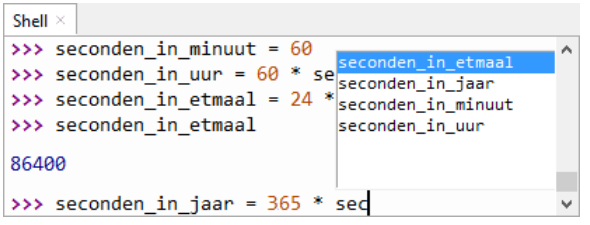
[5] De foutmelding is een zogenoemde 'syntax error'. Dit is een veel voorkomende foutmelding, die aangeeft dat de Python interpreter de code van een commando niet begrijpt.

[6] In de digitale techniek gebruiken we een toestandsdiagram om de werking van een state machine weer te geven.

```
>>> seconden_in_uur = 60 * seconden_in_minuut
>>> seconden_in_etmaal = 24 * seconden_in_uur
>>> seconden_in_etmaal
```

1.1.22 Bereken het aantal seconden in een jaar[7] en ken deze waarde toe aan de variabele `seconden_in_jaar`. Maak daarbij gebruik van de al eerder gedefinieerde variabele `seconden_in_etmaal`. Druk de waarde van de variabele `seconden_in_jaar` ook af in de Python Shell.

Als je in de Python Shell op de tab-toets drukt, dan wordt het woord dat je aan het typen bent vanzelf afgemaakt. Dit wordt ook wel auto-aanvullen (of in het Engels: autocomplete) genoemd. Als er meerdere mogelijkheden zijn, dan verschijnt er een keuzemenu zodat je kan kiezen, zie figuur 5.



Figuur 5: Als je op de tab-toets drukt, wordt het woord dat je aan het typen bent automatisch aangevuld.

1.1.23 Welk van de volgende namen kun je niet als variabelenaam gebruiken in Python en waarom niet?

```
4all
seconden/jaar
seconden_in_jaar
seconden-in-jaar
euros2dollars
euros2$$$$
return
```

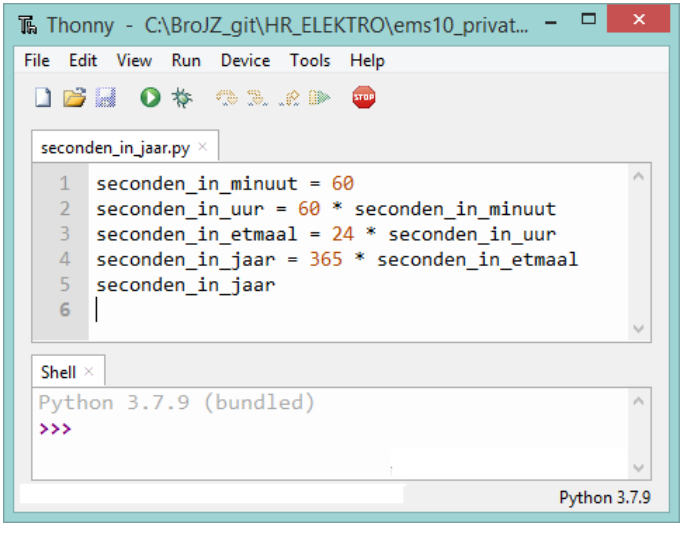
Lees nu paragraaf 2.4 van het boek.

[7] Ga uit van een 'normaal' jaar met 365 dagen. Een schrikkeljaar heeft 366 dagen.

Als je een Python script (programma) wilt schrijven dan kun je dit doen in het bovenste invoerveld van Thonny, zie figuur 4.

1.1.24 Maak een variabele aan in de Python Shell en geef deze variabele een waarde. Druk de waarde van deze variabele af in de Python Shell. Sluit Thonny af en start het programma opnieuw op. Bestaat de variabele nu nog?

1.1.25 Type een Python script in het bovenste invoerveld van Thonny om stap voor stap het aantal seconden in een jaar te berekenen en sla dit script op in het bestand `seconden_in_jaar.py`, zie figuur 6. Als je deze commando's een voor een had ingetypt in de Python Shell, dan was de waarde van de variabele `seconden_in_jaar` in de Shell afgedrukt. Voer nu het script uit door op de afspeelknop in Thonny te klikken, of door op F5 te drukken. Er verschijnt geen uitvoer in de Python Shell, maar het programma is wel uitgevoerd. Dit kun je zien door een variabelenaam uit het programma in te typen in de Python Shell.



Figuur 6: Een Python script (programma) om het aantal seconden in een jaar te berekenen: `seconden_in_jaar.py`.

1.1.26 Pas het programma `seconden_in_jaar.py` nu zo aan dat het aantal seconden in een jaar aan het einde van het programma automatisch afgedrukt wordt in de Python Shell.

Lees nu paragraaf 2.5, 2.6 en 2.7 van het boek.

1.1.27 Wat wordt er geprint door het volgende Python commando?

```
print('Hiep '*2 + 'Hoera') # Gefeliciteerd!
```

De volgorde van het uitvoeren van een Python programma is goed te volgen door in Thommy het script stap voor stap uit te voeren.

1.1.28 Voer het programma `seconden_in_jaar.py` stap voor stap uit door eerst op het de debug-knop te klikken of door op Ctrl + F5 te drukken en daarna steeds op de step-into-knop te klikken of op F7 te drukken. Je kunt nu stap voor stap zien hoe de Python commando's uitgevoerd worden.

1.1.29 Herhaal opdracht 1.1.28 maar gebruik nu de step-over-knop of F6 om door het programma te stappen. Beschrijf het verschil tussen step-into en step-over in je eigen woorden.

Je kunt ook de variabelen en hun waarden zien tijdens het stap voor stap uitvoeren van een programma.

1.1.30 Sluit Thonny af en start het programma opnieuw op. Open het variabelen window via de menuoptie View » Variables. Herhaal opdracht opdracht 1.1.28. Als het goed is zie je zoiets als figuur 7.

Lees nu paragraaf 2.8 van het boek.

1.1.31 Gegeven is het foutieve programma uit listing 1. Download dit programma door op de programmanaam (link) in de titel van de listing te klikken met je rechtermuisknop. Kies vervolgens voor "Save link as..." en sla het programma op[8]. Laad dit programma vervolgens in Thonny en speur de fout op. Hoe heb je de fout gevonden? Is er sprake van een syntax error, runtime error of semantic error?

[8] Normaal gesproken is het beter om voorbeeldcode over te typen (want dan leer je de syntax), maar in dit geval moet je de code kopiëren om te zorgen dat je de fout ook overneemt.

Figuur 7: In het Variables window kun je de waarden van de verschillende variabelen in je programma zien tijdens het stap voor stap uitvoeren van je programma.

1.1.32 Gegeven is het foutieve programma uit listing 2. Download en laad dit programma in Thonny en speur de fout op. Hoe heb je de fout gevonden? Is er sprake van een syntax error, runtime error of semantic error?

1.1.33 Gegeven is het foutieve programma uit listing 3. Download en laad dit programma in Thonny en speur de fout op. Hoe heb je de fout gevonden? Is er sprake van een syntax error, runtime error of semantic error?

```
seconden_in_minuut = 60
seconden_in_uur = 60 * seconden_in_minuut
seconden_in_etmaal = 24 * seconden_in_uur
seconden_in_jaar = 365 * seconden_in_etmaal
print(seconden_in_jaar)
```

Listing 1: Foutief Python programma. Zie `error1.py`.

Paragraaf 2.9 van het boek definieert de verschillende (vak)termen die in hoofdstuk 2 gebruikt worden. Het kan handig zijn om deze paragraaf te raadplegen als je niet meer weet wat met een bepaalde (vak)term bedoeld wordt.

```
r1 = 0
r2 = 0
r1_parallel_r2 = (r1 * r2) / (r1 + r2)
print(r1_parallel_r2)
```

Listing 2: Foutief Python programma. Zie `error2.py`.

```
seconden_in_minuut = 60
seconden_in_uur = 60 * seconden_in_minuut
seconden_in_etmaal = 12 * seconden_in_uur
seconden_in_jaar = 365 * seconden_in_etmaal
print(seconden_in_jaar)
```

Listing 3: Foutief Python programma. Zie `error3.py`.

Paragraaf 2.10 van het boek bevat enkele oefeningen. De oefeningen die de moeite waard zijn, zijn hieronder in het Nederlands vertaald.

1.1.34 Het is goed om te experimenteren met de Python commando's die je tot nu toe hebt geleerd. Beantwoord de volgende vragen:

A Je hebt gezien dat `n = 42` correcte code is. Geldt dat ook voor `42 = n`?

B Is de code `x = y = 1` correct? Wat gebeurt er als deze code uitgevoerd wordt?

C In sommige programmeertalen moet elk commando afgesloten worden met een puntkomma. Wat gebeurt er als je een Python commando afsluit met een puntkomma? Waarom zou je dit willen doen?[9]

D Wat gebeurt er als je een Python commando afsluit met een punt? [10]

E In wiskundige notatie kun je `x` en `y` vermenigvuldigen door te schrijven: `xy`. Wat gebeurt er als je dat in Python probeert?

Lees nu hoofdstuk 3 tot en met paragraaf 3.3 van het boek.

[9] Door het gebruik van puntkomma's kun je meerdere Python commando's op één regel plaatsen. Bijvoorbeeld: `x = 3; y = 5; z = -2`

[10] De meeste Python commando's kun je niet afsluiten met een punt, maar sommige wel. Zoek uit wat het verschil is tussen `x = 10.` en `x = 10`

1.1.35 De stelling van De Moivre luidt:

$$(\cos(\alpha) + j\sin(\alpha))^n = \cos(n\alpha) + j\sin(n\alpha)$$

Schrijf een Python programma dat de linker en rechterkant van de bovenstaande vergelijking berekent voor $\alpha = \frac{1}{4}\pi$ en $n = 5$. Hoe verklaar je het verschil tussen de linker en de rechterkant?[11]

Hier eindigen de verplichte opdrachten van week 1 les 1. Er volgen nu nog twee gedeelten:

- oefening, in dit gedeelte vind je extra oefeningen die je helpen om de leerstof van deze les beter te onthouden;
- verdieping, in dit gedeelte vind je uitdagende, verdiepende opdrachten.

Oefening

Het is handig om dingen uit je hoofd te leren als je begint met programmeren. Dit is om twee redenen belangrijk:

- Als je kennis in je langetermijngeheugen hebt opgeslagen, dan wordt je werkgeheugen minder belast tijdens het programmeren.
- Als je ontbrekende kennis moet opzoeken tijdens het programmeren, dan werkt dat meer verstorend dan je wellicht denkt. Zeker als je daarvoor het internet gebruikt.

Er zijn twee manieren waarop je er voor kunt zorgen dat kennis in je langetermijngeheugen wordt opgeslagen:

- Door jezelf regelmatig te overhoren (b.v. met flashcards[12]).
- Door te denken aan de kennis die je wilt opslaan. Dit kun je doen door jezelf vragen te stellen zoals:
  - Waar doet me deze nieuwe kennis aan denken en waarom is dat zo?
  - Waar zou ik deze nieuwe kennis voor kunnen gebruiken?
  - Voegt deze nieuwe kennis echt iets toe of zou ik met mijn al aanwezige kennis hetzelfde kunnen bereiken?

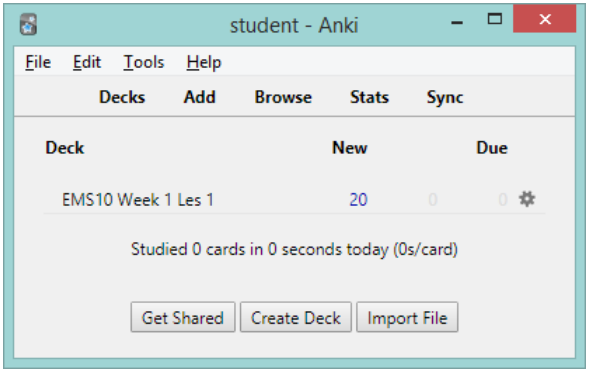
[11] Kijk nog eens naar opdracht 1.1.14 als je het verschil niet kunt verklaren.

[12] Zie eventueel Wikipedia: https://en.wikipedia.org/wiki/Flashcard.

Er zijn diverse programma's beschikbaar waarin je flashcards kunt aanmaken en die je helpen om regelmatig te oefenen. Wij raden je aan om het programma Anki te gebruiken. Dit programma kun je gratis downloaden van https://apps.ankiweb.net/.

Tip: installeer het ook op je telefoon, zodat je gemakkelijk tussendoor ook even kan oefenen.

Voor deze les is een stok (Engels: deck) flashcards beschikbaar waarmee je kunt oefenen. Deze stok kun je hier downloaden: EMS10 Week 1 Les 1.apkg. Als je op de bovenstaande link klikt en je hebt Anki geïnstalleerd, dan kun je het stok meteen in Anki openen. Je kunt het ook eerst opslaan en vervolgens in Anki importeren door op Import File te klikken, zie figuur 8.



Figuur 8: Het stok genaamd EMS10 Week 1 Les 1 is geïmporteerd in Anki.

Je kunt vervolgens beginnen met oefenen door op "EMS10 Week 1 Les 1" te klikken. Een handleiding kun je, indien nodig, vinden op https://docs.ankiweb.net/studying.html.

Je kunt ook zelf flashcards toevoegen!

Als je Anki ook op je telefoon hebt geïnstalleerd, dan kun je jouw werk op je pc en op je telefoon synchroniseren door een account aan te maken op Ankiweb, zie https://docs.ankiweb.net/syncing.html.

Doe dit want oefenen op je telefoon werkt veel gemakkelijker dan op de pc!

Verdieping

Hieronder vind je nog een aantal uitdagende, verdiepende opdrachten. Het is niet noodzakelijk om deze opdrachten te maken, maar wel leuk en handig voor bij de wiskunde en ELE20 lessen!

Werken met een Jupyter notebook

Het is mogelijk om Python code te gebruiken in een zogenoemd Jupyter notebook vanuit je internet browser. De onderstaande opdrachten voer je uit in deze browser omgeving en dus niet in Thonny.

Je gaat nu eerst de Jupyter omgeving en de benodigde Python packages installeren. Hier gebruiken we Thonny wel voor.

1.1.36

A Open Thonny.

B Kies Tools > Manage Packages....

C In het zoekveld type je *jupyter*, druk op Search en als het is gevonden, klik je op de link en kies je vervolgens Install. Na de installatie zou je een vergelijkbaar beeld als figuur 9 moeten zien[13].

D Installeer nu ook de packages:

- *sympy*[14], waarmee symbolisch gerekend kan worden;
- *scipy*[15], waarmee wetenschappelijk (Engels: scientific) gerekend kan worden;
- *numpy*[16], waarmee met vectoren (Engels: scientific) gerekend kan worden.

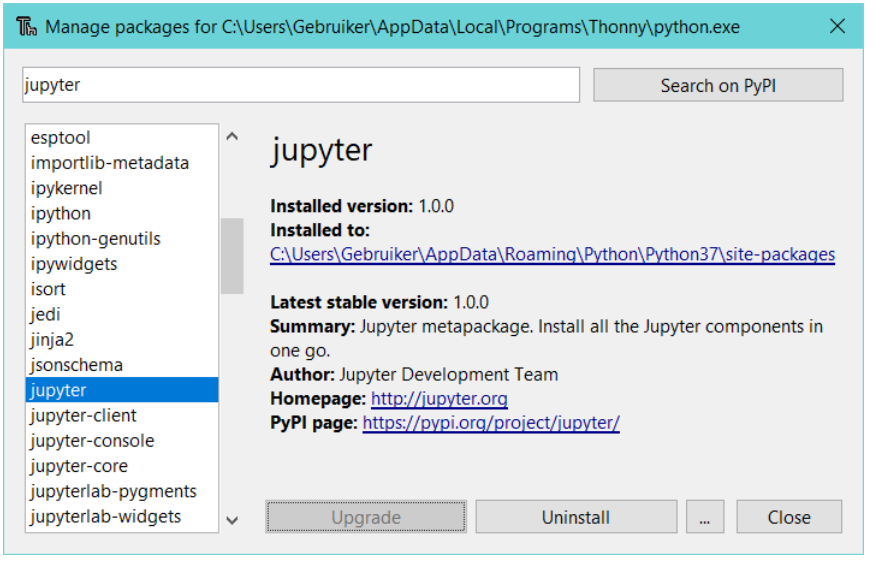
1.1.37 Sluit Thonny af en start het opnieuw op zodat de zojuist geïnstalleerde packages gebruikt kunnen worden.

[13] Als je een foutmelding krijgt bij het installeren van deze packages, dan kun je deze (64-bits) versie van Thonny installeren: https://github.com/thonny/thonny/releases/download/v4.0.1/thonny-4.0.1.exe.

[14] Zie eventueel: https://www.sympy.org.

[15] Zie eventueel: https://www.scipy.org/.

[16] Zie eventueel: https://numpy.org/.



Figuur 9: Thonny manage packages

Nu alle benodigdheden voor Python zijn geïnstalleerd, moet alleen nog het Jupyter notebook worden gestart. Dit notebook is de plek waarin je Python-statements kunt uitvoeren en aantekeningen kunt maken.

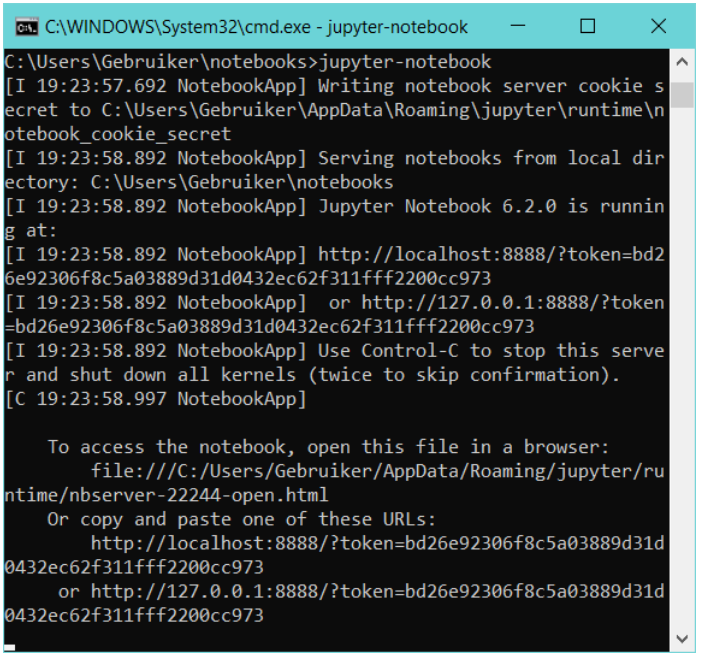
1.1.38

A In Thonny, ga naar Tools > Open System Shell... . Dit opent een nieuw command venster.

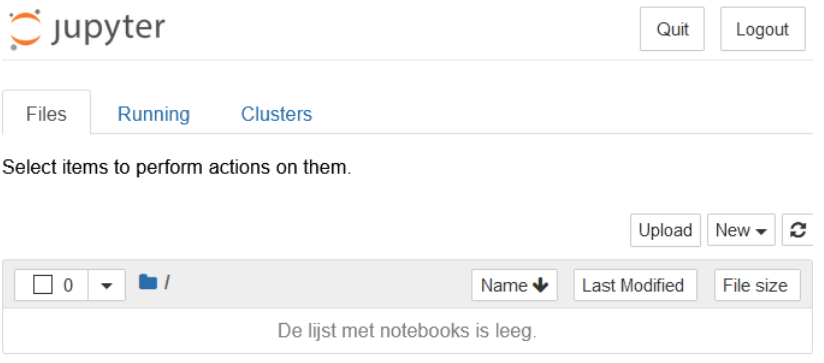
B Type in het venster `jupyter-lab` en druk op enter (figuur 10). Als er een foutmelding wordt gegeven, dan kun je het commando `jupyter-notebook` gebruiken[17]. Een nieuw venster zou moeten openen in je browser (figuur 11). Zo niet, dan kun je naar de link gaan die in het commandvenster is gegeven na uitvoer van het commando.

C In het browser venster uit figuur 11 kies je New > Python 3 . Dit opent jouw eerste notebook (figuur 12).

[17] Bij Thonny versie 3.3.7 moet het commando `jupyter-lab` gebruikt worden, maar bij Thonny versie 4.0.1 moet het commando `jupyter-notebook` gebruikt worden.



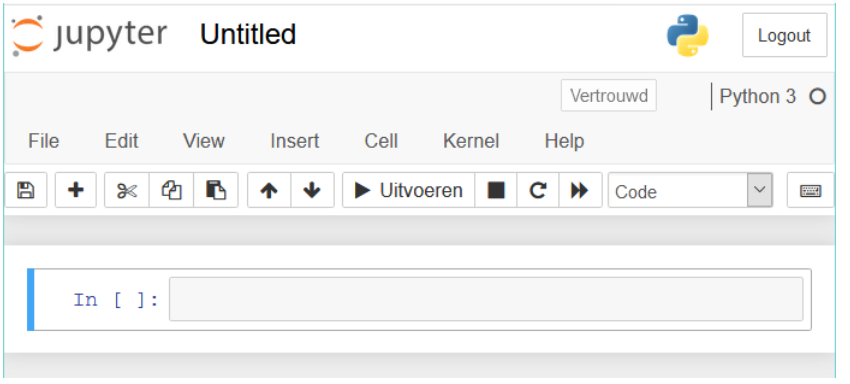
Figuur 10: Invoeren commando.



Figuur 11: Jupyter vóór het starten van een notebook.

Binnen het notebook heb je `In []` velden. Binnen zo'n veld heb je de mogelijkheid om Code, Markdown[18] of een Heading (koptekst) te typen. Met de dropdown box in de menubalk kun je kiezen wat je wilt invoeren. Shift + Enter zal de code binnen het veld uitvoeren.

[18] Markdown is een eenvoudige opmaaktaal voor tekst (vergelijkbaar met HTML maar veel eenvoudiger), zie eventueel: https://www.markdownguide.org/basic-syntax/.



Figuur 12: Een nieuw Jupyter notebook.

1.1.39 Kies File > Save as... en geef het notebook een naam, bijvoorbeeld `Week1`. Type nu de Python code om 2^{150} uit te rekenen in de geopende cel. Druk op Alt + Enter om de code in een cel uit te voeren en een nieuwe cel te openen. Als het goed is verschijnt de waarde van 2^{150} zoals weergegeven in figuur 13.

```
In [1]: 2**150
Out[1]: 1427247692705959881058285969449495136382746624

In [ ]:
```

Figuur 13: 2^{150} berekent in een Jupyter notebook.

Zoals je in figuur 14 kunt zien, kun je in Python symbolische vergelijkingen oplossen.

```
In [2]: import sympy
        x = sympy.Symbol('x')
        sympy.solve(5*x - 15, x)
Out[2]: [3]
```

Figuur 14: De vergelijking $5x - 15 = 0$ opgelost in een Jupyter notebook.

1.1.40 Bepaal de punten waarin de functies $f(x) = x^2 + 3x + 6$ en $g(x) = 4x + 9$ elkaar snijden[19] in een Jupyter notebook.

[19] Bron: WIS10 Oefenopdrachten week 1.1

1.1.41 Los de volgende vergelijkingen[20] op in een Jupyter notebook:

A $\sqrt{x-2} = 2 + x$

B $2^{3x+1} = 4^{x+1}$

C ${}^{2}\log(x-3) = {}^{4}\log(x+3)$

Zoals je in figuur 15 kunt zien, kun je in Python limieten symbolisch oplossen.

```
In [3]: import sympy
        x = sympy.Symbol('x')
        sympy.limit(sympy.sin(x) / x, x, 0)

Out[3]: 1
```

Figuur 15: De $\lim_{x \to 0} \frac{\sin(x)}{x}$ opgelost in een Jupyter notebook.

1.1.42 Bereken de volgende limiet[21] in een Jupyter notebook:

A $\lim_{x \to 0} \frac{e^x - \sqrt[3]{1+3x}}{1-\cos(2x)}$

Zoals je in figuur 16 kunt zien, kun je in Python symbolisch differentiëren.

```
In [4]: import sympy
        x = sympy.Symbol('x')
        sympy.diff(12*x**4 - 8*x**2 -4, x)

Out[4]: 48*x**3 - 16*x
```

Figuur 16: De functie $f(x) = 12x^4 - 8x^2 - 4$ gedifferentieerd in een Jupyter notebook.

1.1.43 Differentieer de volgende functies[22] in een Jupyter notebook:

A $f(x) = 5\sqrt{x} + 3x$

B $f(x) = \ln(\sqrt[3]{x^7 - 4x})$

[20] Bron: WIS10 Oefenopdrachten weken 1.1 en 1.2

[21] Bron: WIS10 Oefenopdrachten week 2.6

[22] Bron: WIS10 Oefenopdrachten week 2.4

Zoals je in figuur 17 kunt zien, kun je in Python onbepaald integreren.

```
In [5]: import sympy
        x = sympy.Symbol('x')
        sympy.integrate(11*x**5, x)

Out[5]: 11*x**6/6
```

Figuur 17: De functie $\int 11x^5 \mathrm{d}x$ geïntegreerd in een Jupyter notebook.

1.1.44 Bepaal de volgende integralen[23] in een Jupyter notebook:

A $\int x\sqrt{x}(1-x)\mathrm{d}x$

B $\int \frac{5}{\sqrt{1-x^2}}\mathrm{d}x$

In Python kun je een stelsel van vergelijkingen oplossen. In figuur 18 kunt zien hoe je het hierna gegeven stelsel[24] kunt oplossen.

$$\begin{cases} 8x_1 - 2x_2 + 4x_3 = -11 \\ -4x_1 - 6x_2 \quad\quad = -27 \\ 10x_1 - 4x_2 - 2x_3 = 9 \end{cases}$$

```
In [6]: import numpy
        import scipy.linalg as lin
        A = [[8,-2,4],[-4,-6,0],[10,-4,-2]]
        b = [-11,-27,9]
        lin.solve(A, b)

Out[6]: array([ 1.5,  3.5, -4. ])
```

Figuur 18: Een stelsel van vergelijkingen opgelost in een Jupyter notebook.

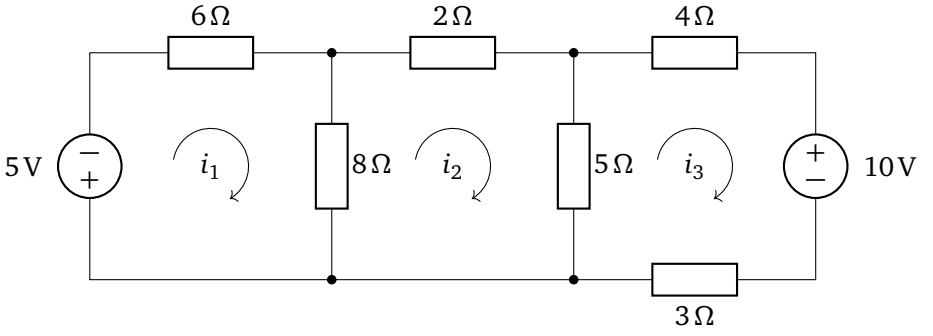
[23] Bron: WIS10 Oefenopdrachten week 2.5

[24] Bron: WIS10 Oefenopdrachten week 2.1

1.1.45 Los, indien mogelijk, het volgende stelsel[25] op in een Jupyter notebook:

$$\mathbf{A}\begin{cases} -10x_1 - 3x_2 - 15x_3 = 11 \\ -2x_1 + x_2 - 3x_3 = 3 \\ 6x_1 + 5x_2 + 9x_3 = 2 \end{cases}$$

1.1.46 Bereken bij schakeling 1[26] i_1, i_2 en i_3 in een Jupyter notebook:



Schakeling 1: Een weerstandsnetwerk met meerdere spanningsbronnen.

Bij ELE20 kun je een Jupyter notebook gebruiken om een stelsel van vergelijkingen symbolisch op te lossen: Handleiding_-Symbolische_vergelijkingen_oplossen_met_python.pdf.

[25] Bron: WIS10 Oefenopdrachten week 2.1

[26] Bron: WIS10 Oefenopdrachten week 2.1