



**UCLL**  
HOGESCHOOL



# **HANDLEIDING**

## **Kraak jij de wiskundige en biologische code?**

STEM-PROJECT

### **Studenten:**

Caroline Geerts

Shelsey Geypen

Fleur Schuurmans

Sarah Van Genechten

### **Promotoren :**

Anke Buckinx

Mieke Schuermans



# Inhoud

<b>1. Inleiding .....</b>	<b>3</b>
<b>2. Introductie.....</b>	<b>6</b>
<b>3. Inleidende opdracht .....</b>	<b>7</b>
3.1 Opdracht 1: Set .....	8
3.2 Opdracht 2: driehoeken .....	13
3.3 Opdracht 3: smartgames .....	16
<b>4. Het stappenplan .....</b>	<b>17</b>
4.1 Decompositie .....	18
4.2 Patroonherkenning .....	20
4.3 Abstractie.....	25
4.4 Algoritme .....	29
4.5 Foutenanalyse.....	34
4.6 Overzicht van het stappenplan.....	39
4.7 Algemene oefeningen .....	40
<b>5. Programmeren in Python.....</b>	<b>42</b>
5.1 De basis in de programmeerwereld.....	43
5.1.1 Startscherm.....	44
5.1.2 Bewerkingen in Python .....	45
5.2 Basisfuncties in Python .....	48
5.2.1 print().....	49
5.2.2 Variabelen .....	49
5.2.3 input() .....	50
5.2.4 int() en float() .....	51
5.2.5 str() .....	51
5.2.6 type() .....	52

5.3	ALS/DAN-functies .....	57
5.3.1	Booleaanse expressie .....	57
5.3.2	if-statement of enkelvoudige keuzestructuur .....	59
5.3.3	if/else-statement of tweevoudige keuzestructuur .....	60
5.3.4	if/elif/else-statement of meervoudige keuzestructuur.....	61
5.3.5	Booleaanse operatoren .....	68
5.3.6	Combinaties met backslash .....	70
5.3.7	Built-in-functies .....	71
5.4	Lussen in Python .....	75
5.4.1	While-lus.....	75
5.4.2	Keyword 'in'.....	77
5.4.3	For-lus en de functie range() .....	78
5.5	Overkoepelende oefeningen .....	85
<b>6.</b>	<b>Bijlagen .....</b>	<b>101</b>
6.1	Leerplannen .....	101
6.1.1	Wiskunde .....	101
6.1.2	Biologie.....	102
6.1.3	ICT.....	102
6.1.4	STEM .....	103
6.2	Verbetersleutel set.....	104
6.3	Verbetersleutel driehoeken.....	108
6.4	Infofiches smartgames .....	<b>Fout! Bladwijzer niet gedefinieerd.</b>
6.5	Exit-card .....	<b>Fout! Bladwijzer niet gedefinieerd.</b>
6.6	Intro-card .....	<b>Fout! Bladwijzer niet gedefinieerd.</b>
6.7	Oefeningenblad .....	<b>Fout! Bladwijzer niet gedefinieerd.</b>
6.8	Oplossingenbundel.....	<b>Fout! Bladwijzer niet gedefinieerd.</b>
6.9	PowerPoint quiz.....	<b>Fout! Bladwijzer niet gedefinieerd.</b>
6.10	Quizformulier .....	<b>Fout! Bladwijzer niet gedefinieerd.</b>
6.11	Algemene PowerPoint .....	<b>Fout! Bladwijzer niet gedefinieerd.</b>
<b>7.</b>	<b>Geraadpleegde bronnen .....</b>	<b>Fout! Bladwijzer niet gedefinieerd.</b>

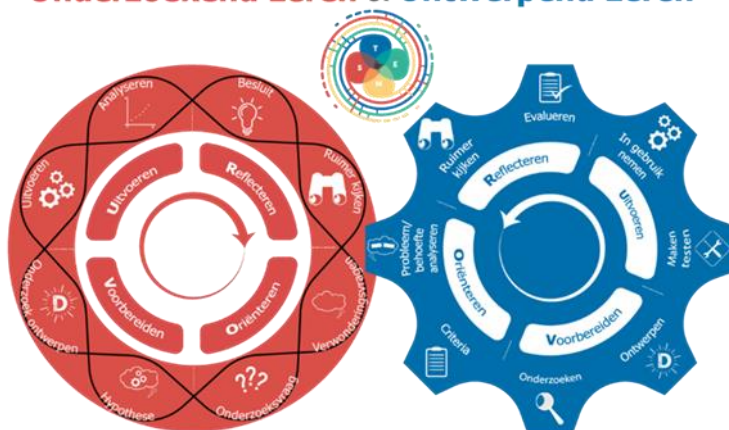
# 1. Inleiding

*Computationeel denken is een onderwerp dat niet bij iedereen voor de hand ligt. Nochtans maakt dit onderwerp, sinds de vernieuwing van het secundair onderwijs, integraal deel uit van het vak wiskunde. Om leerkrachten hierin te ondersteunen, hebben wij, vier studenten van de lerarenopleiding secundair onderwijs van de hogeschool UCLL te Diepenbeek, een STEM-project uitgewerkt rond computationeel denken. Dit project wordt uitgewerkt als opdracht voor het opleidingsonderdeel GO<sup>3</sup>. GO<sup>3</sup> staat voor geïntegreerde ontwikkeling, onderzoek en onderwijs. Het is een opleidingsonderdeel dat kadert binnen de bachelorproef ter afronding van de opleiding.*

*GO<sup>3</sup> is een project georganiseerd door de hogeschool UCLL. Tijdens dit project worden studenten uit verschillende onderwijsvakken samengebracht om zo een interdisciplinair STEM-project te ontwikkelen. Wij hebben dit project gevormd vanuit de eigen onderwijsvakken, namelijk wiskunde en biologie.*

*Doorheen de handleiding zal u symbolen zien verschijnen van onderstaand STEMOVUR-model. Dit model vormt een leidraad en zorgt voor enige ondersteuning van dit project. Het baseert zich op het onderzoekend en ontwerpend leren. Het doel van dit model is een zekere groei van deze aspecten te creëren bij de leerlingen. In dit project zal er hoofdzakelijk rond onderzoekend leren gewerkt worden. Dit doordat er geen materieel eindresultaat is.*

## STEMOVUR model voor Onderzoekend Leren & Ontwerpend Leren



*Om het computationeel denken minder abstract te maken, is er zo veel mogelijk gebruik gemaakt van voorbeelden uit de biologie.*

*In deze bundel vindt u een volledig uitgewerkte en gebruiksvriendelijke lerarenversie van het STEM-project. Bijkomend staan er in de bijlagen de leerlingenbundel en andere nodige documenten om dit project optimaal te kunnen uitrollen. De stukjes handleiding, die voor leerkrachten van belang zijn, staan steeds schuin gedrukt met daarnaast een verticale lijn.*

Daarnaast bevat deze bundel ook een begeleid zelfstandig werk over programmeren in Python. Deze kan gebruikt worden voor zowel leerlingen zonder voorkennis rond programmeren als leerlingen met een basis aan programmeren. Binnen dit deel kunnen leerlingen op eigen tempo aan de slag met de nodige klassikale reflectiemomenten ter ondersteuning van het geleerde.

Indeling lessenreeks:

Het ontwikkeld STEM-project wordt uitgerold in een lessenreeks van 15 à 20 lesuren afhankelijk van de voorkennis. Hieronder staat kort weergegeven hoeveel tijd er aan ieder deel wordt besteed.

Introductie + inleidende opdrachten: 1 lesuur

Stappenplan: 2 à 3 lesuren

Programmeren met Python: 10 à 15 lesuren

Leerplan:

Binnen dit STEM-project worden uiteraard ook tal van leerplandoelstellingen behandeld. Hieronder een weergave waarin ze per onderwijsvak beknopt staan beschreven. In bijlage staan deze volledig uitgeschreven.

De leerplandoelstellingen van biologie en STEM komen aan bod, maar worden niet tot in detail behandeld.

**Wiskunde:** D/2021/13.758/045

LPD 2 → Onderzoeken van interacties tussen wiskunde en andere domeinen

LPD 60 → Ontwerpen van algoritmes om problemen op te lossen

**ICT:** D/2021/13.758/001

LPD 3 → Gebruiken van toepassingen om digitale inhoud te beheren

GLI-ddaa9 → Oefenen van de ALS-functie

**Biologie:**

1<sup>ste</sup> graad: D/2019/13.758/011

LPD 9 → Onderzoeken van onderlinge afhankelijkheid van verschillende organismen en biotische- en abiotische factoren

3<sup>de</sup> graad: D/2014/7841/011

LPD B23 → Structuur van DNA en RNA

LPD B38 → Oplossen van vraagstukken die gaan over overerving

**STEM:** D/2021/13.758/034

LPD S2 → Analyseren van systemen a.d.h.v. verschillende STEM-concepten


LPD S8 → Ontwikkelen van modellen voor visualisatie, onderzoek en verklaringen

### Symbolen:


*Doorheen de bundel wordt gebruik gemaakt van enkele symbolen. Deze staan hieronder opgesomd met wat extra uitleg.*

- *Tijd voor oefeningen:* 

*Dit symbool geeft aan dat er oefeningen gemaakt moeten worden. Afhankelijk van de oefening mogen ze samenwerken of individueel maken. Dit staat steeds vermeldt in de handleiding.*

- *Even over het muurtje kijken:* 

*Dit symbool geeft aan dat we over het muurtje gaan kijken. Dit houdt in dat er op dat moment de link gelegd zal worden met een ander vak. In de titel zal steeds worden aangegeven binnen welk domein er verder gewerkt zal worden.*

- *Even herhalen:* 

*Dit symbool geeft aan dat er een korte herhaling gedaan wordt van eerder geziene leerstof. De leerlingen weten hierdoor dat ze gebruik moeten maken van hun voorkennis en indien nodig even moeten terugbladeren in de bundel.*

## 2. Introductie



*Om te starten met dit project willen we aftoetsen wat de leerlingen al weten van computationeel denken. Dit doen we aan de hand van een woordwolk in mentimeter. Op het bord zal het woord "computationeel denken" verschijnen. De leerlingen kunnen dan individueel via hun smartphone aangeven waar ze aan denken wanneer ze dit horen. Op deze manier krijgt de leerkracht een idee van de voorkennis van de leerlingen.*

Link naar mentimeter voor de leerkracht:

<https://www.mentimeter.com/app/presentation/alcbfrsnjpp7ee4vqhu64qnkzkqk9sk/do42uox57vqc/edit>

In dit STEM-project zal je doorheen de lessen meer te weten komen over het concept computationeel denken. Je zal zelf aan de slag gaan, uitgedaagd worden,... Om te beginnen mag je zelf via de QR-code of de link aangeven waar jij aan denkt bij de woorden "computationeel denken".



Link: <https://www.menti.com/al1rpc2zsmwc>

De bundel is verder opgebouwd met een vaste structuur. In de werkbundel wordt gebruik gemaakt van enkele symbolen. Hieronder een korte opsomming van deze symbolen.

- Tijd voor oefeningen:



- Even over het muurtje kijken:



- Even herhalen:



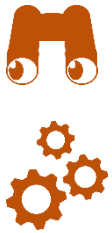
### Doelstellingen:

Na dit STEM-project kan je...

1. Een beeld vormen van wat computationeel denken is.
2. In eigen woorden uitleggen hoe computationeel denken in elkaar zit.
3. De verschillende stappen van computationeel denken in eigen woorden uitleggen.
4. De verschillende stappen van computationeel denken toepassen in oefeningen.
5. In eigen woorden uitleggen op welke manieren computationeel denken aan bod kan komen binnen het vak biologie/natuurwetenschappen.
6. Programmeren in Python.



### 3. Inleidende opdracht



*Om het project computationeel denken in te leiden wordt er gestart met een inleidende opdracht. Op deze manier maken leerlingen kennis met dit begrip. Ze zullen merken dat het een heel complex gegeven is, maar vooral dat ze hier dagelijks gebruik van maken zonder dat ze dit beseffen. Voor deze opdracht zullen de leerlingen in groepjes van 3 ingedeeld worden. Deze groepjes worden vooraf bepaald zodanig dat alles vlot kan verlopen.*

*De inleidende opdracht bestaat uit 3 deelopdrachten:*

- 1. Set*
- 2. Driehoeken*
- 3. Smartgames*

*De leerlingen krijgen hiervoor ongeveer 30-40 min. Het is de bedoeling dat ze deelopdracht 1 en 2 finaliseren vooraleer ze aan deelopdracht 3 beginnen. Wanneer leerlingen aan de smartgames beginnen komen ze deze halen bij de leerkracht. Hij/zij controleert eerst of de vorige deelopdrachten volledig zijn afgerond. Deze controle maakt de leerkracht zichtbaar door een handtekening in het voorziene (zie deelopdrachten) kadertje te zetten. Op deze manier wordt voorkomen dat leerlingen meteen smartgames beginnen spelen. De leerlingen ontvangen vervolgens een verbetersleutel van deze opdrachten. Iedere opdracht staat steeds uitvoerig beschreven. Er wordt ook voldoende materiaal voorzien voor alle groepen zodat de leerlingen dus in groep zelfstandig aan het werk kunnen. Om te voorkomen dat elke smartgame door de leerkracht volledig moet worden uitgelegd, staan in bijlage infofiches. Deze fiches zullen in praktijk bij iedere smartgame worden meegegeven met de leerlingen.*

*Na deze opdrachten wordt er gepolst naar de verwachtingen van de leerlingen. Dit doen we aan de hand van een onderwijsleergesprek (OLG). Samen met de leerlingen worden hun verwachtingen afgetoetst.*

Wat is computationeel denken eigenlijk? Om hier een antwoord op te kunnen bieden, volgt een inleidende opdracht. Deze opdracht bestaat uit 3 deelopdrachten, die verder in de bundel volledig staan uitgelegd.

Enkele **afspraken** om deze opdrachten vlot te laten verlopen:

- ✓ Je werkt voor deze opdrachten in je toegewezen groep van 3. Je blijft voor deze opdrachten steeds in die groep!
- ✓ Je werkt de opdrachten op de aangegeven volgorde af.
- ✓ Bij aanvang van opdracht 3 vraag je de leerkracht om de eerder gemaakte opdracht af te tekenen. De leerkracht zal dan de eerder gemaakte opdracht controleren en je toestemming geven om verder te werken.
- ✓ Je krijgt voor de volledige opdrachten ongeveer 30-40 min.  
Bij iedere deelopdracht staat aangegeven wat de benodigdheden zijn en hoelang je mag werken aan deze opdracht.

## 3.1 Opdracht 1: Set







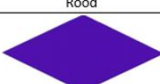
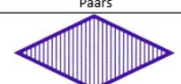
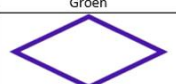

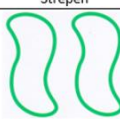
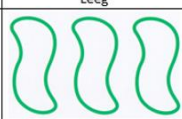
**Benodigdheden:** kaarten

**Tijd:** maximum 15 min.

**Gecontroleerd:** ☐

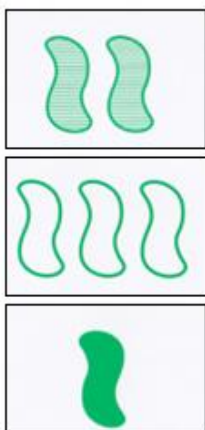
### Uitleg spel:

Dit spel speel je met kaarten waarop symbolen staan afgebeeld. Deze symbolen hebben allerlei eigenschappen. Ze kunnen onderscheiden worden op basis van vorm, kleur, opvulling en aantal. (Zie afbeelding hiernaast.)

Vorm			
	Ruit	Ovaal	Golf
Kleur			
	Rood	Paars	Groen
Opvulling			
	Vol	Strepen	Leeg
Aantal			
	1	2	3

De bedoeling is dat je op zoek gaat naar een 'set'. Een set is een combinatie van drie kaarten die bij elkaar horen. Dit is het geval wanneer voor elke eigenschap geldt dat ze allemaal gelijk óf allemaal verschillend zijn.

Hieronder een voorbeeld:



Dit is een set. De kaarten hebben allemaal dezelfde vorm, dezelfde kleur, een verschillende opvulling en een verschillend aantal. Iedere eigenschap moet dus in de 3 gevallen gelijk of verschillend zijn.



## Tijd voor oefeningen

### Oefening 1:

Vormen onderstaande kaarten een set?

- Bespreek iedere eigenschap.
- Besluit vervolgens of er al dan niet een set gevormd wordt. Duid dit aan.
- Geef tot slot een verantwoording waarom het wel/niet een set is.

1.



Vorm: De vorm is steeds hetzelfde.

Kleur: De kleuren zijn steeds verschillend.

Opvulling: De opvulling is steeds verschillend.



Aantal: Het aantal is steeds verschillend.



Dit is **wel**/niet een set.

Verantwoording: Alle eigenschappen zijn voor de 3 kaarten ofwel gelijkend ofwel verschillend.

2.



Vorm: De vorm is niet overal verschillend maar ook niet overal gelijk.

Kleuren: De kleuren zijn steeds verschillend.



Opvulling: De opvulling is niet overal verschillend maar ook niet overal gelijk.

Aantal: Het aantal is steeds verschillend.



Dit is wel/**niet** een set. → Deze conclusie kon al genomen worden bij het bestuderen van de vorm. De andere eigenschappen bekijken was dus eigenlijk overbodig.

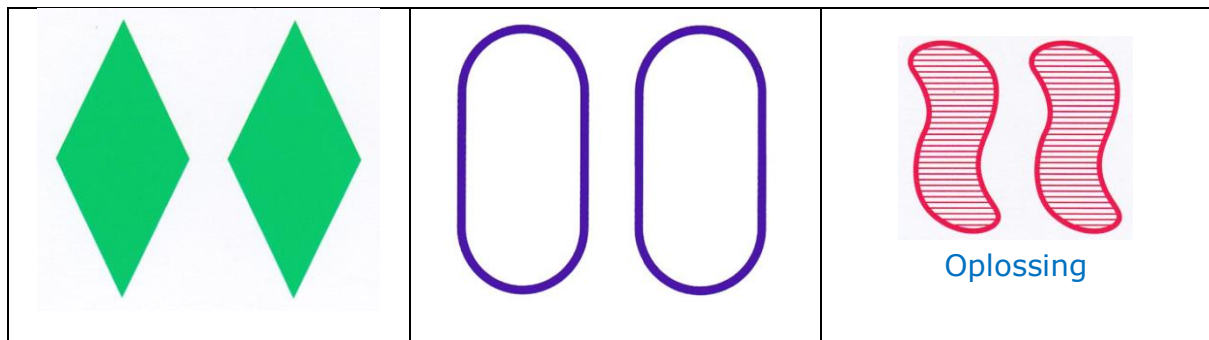
Verantwoording: Niet alle eigenschappen zijn voor de 3 kaarten gelijkend of verschillend.

## Oefening 2:

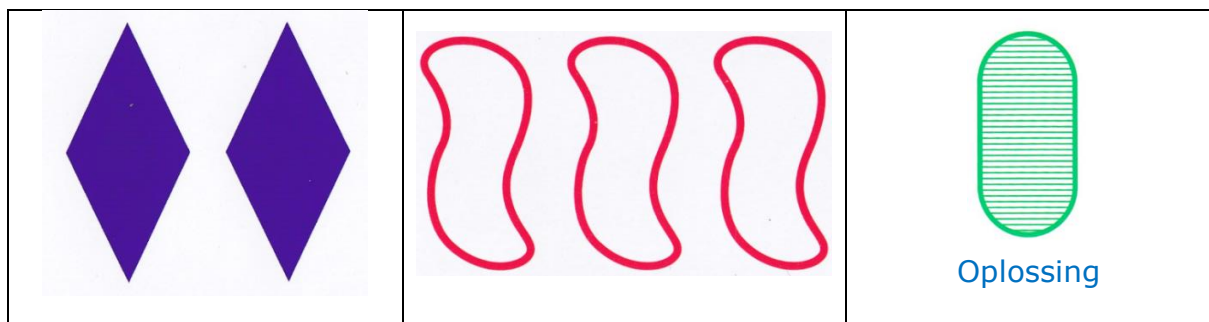
Er zijn steeds twee kaarten gegeven. Ga zelf op zoek naar de derde kaart om de set te vervolledigen. Je kent de eigenschappen waar een set aan moet voldoen.

- Teken in het laatste vakje de ontbrekende kaart van de set. Let op elke eigenschap!

1.

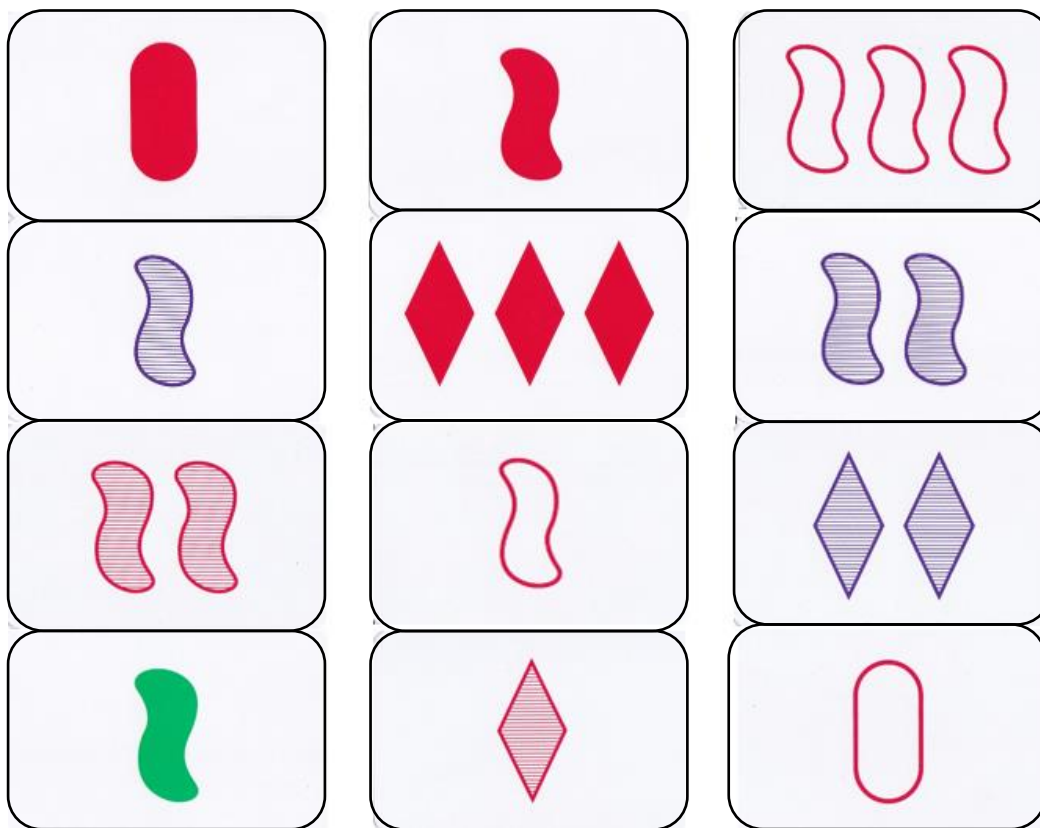


2.



### Oefening 3:

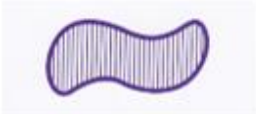



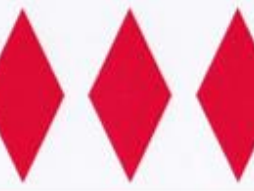






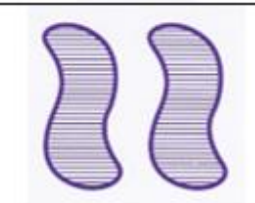






Hieronder zijn 12 kaarten gegeven. De bedoeling is dat jullie hier zo veel mogelijk sets in terug vinden. In totaal zijn er 6 sets te vinden. Hieronder staat een tabel waar je de figuren kan in overtekenen en zo de set zichtbaar kan maken.



Teken hier 3 sets die je terug kan vinden.

Set 1			
Set 2			
Set 3			

## Oplossing

Oplossing 1			
Oplossing 2			
Oplossing 3			
Oplossing 4			
Oplossing 5			
Oplossing 6			

## 3.2 Opdracht 2: driehoeken

**Benodigdheden:** driehoeken

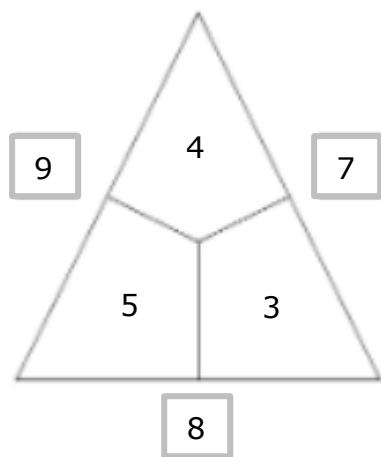
**Tijd:** maximum 15 min.

**Gecontroleerd:** ☐

**Uitleg opdracht:**

Voor deze opdracht zijn er 4 driehoeken gegeven. Iedere driehoek is opgesplitst in 3 gebieden. In elk gebied van de driehoek moet één getal staan. Aan iedere zijde van de driehoek staan vierkantjes waarin ook getallen horen te staan. Een driehoek is correct ingevuld als elk getal aan een zijde de som is van de getallen in de aanpalende gebieden.

Voorbeeld van een correct ingevulde driehoek:



**Beantwoord volgende vragen:**

Som getallen in de vierkanten = 24

Som getallen in de driehoek = 12

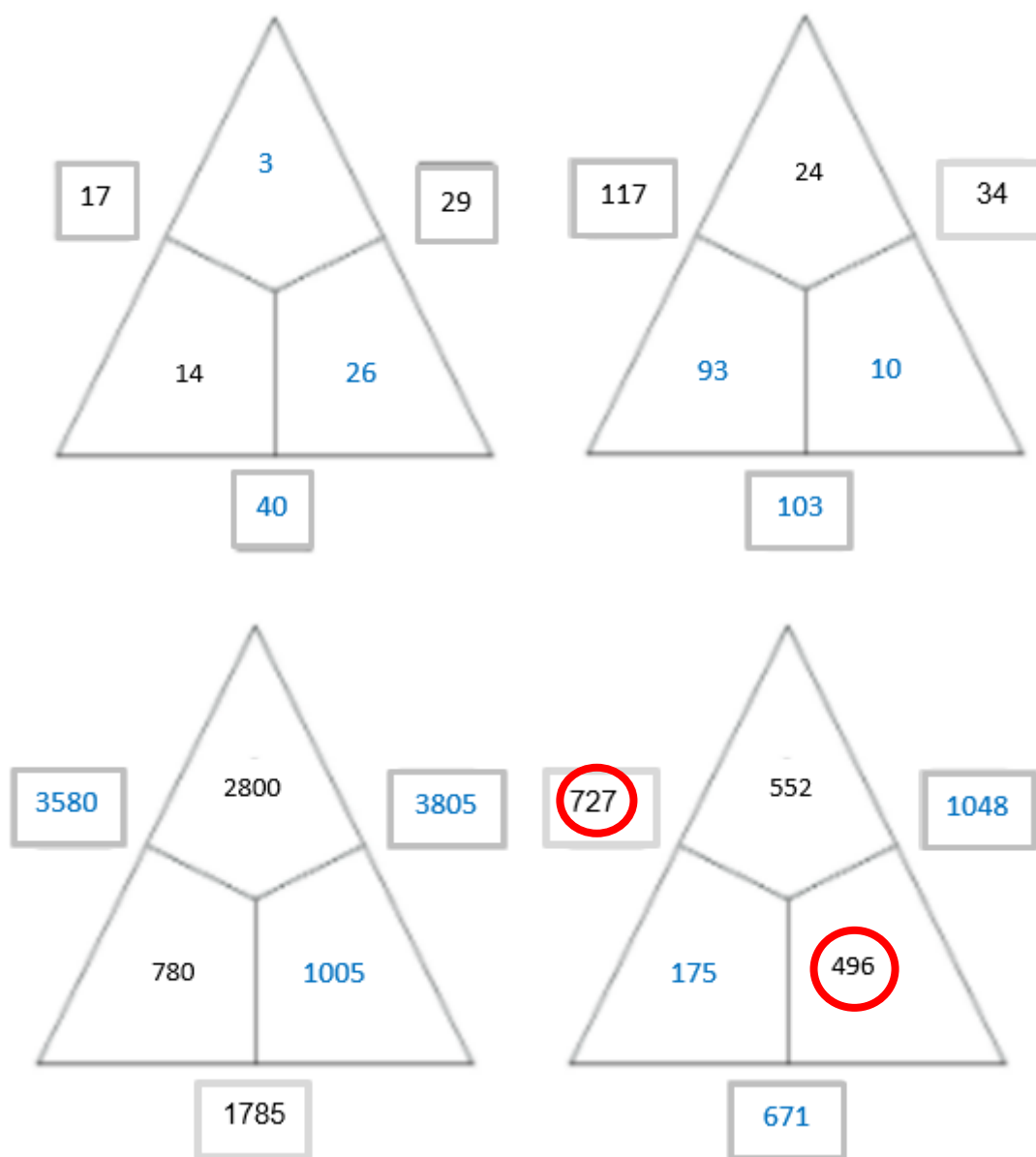
Welk verband vind je tussen de som van de getallen in de vierkantjes en de som van de getallen binnen de driehoek?

De som van de getallen binnen de driehoek is de helft van de som van de getallen in de vierkantjes.



## Tijd voor oefeningen

Vervolledig volgende driehoeken. In te vullen



### Verbanden:

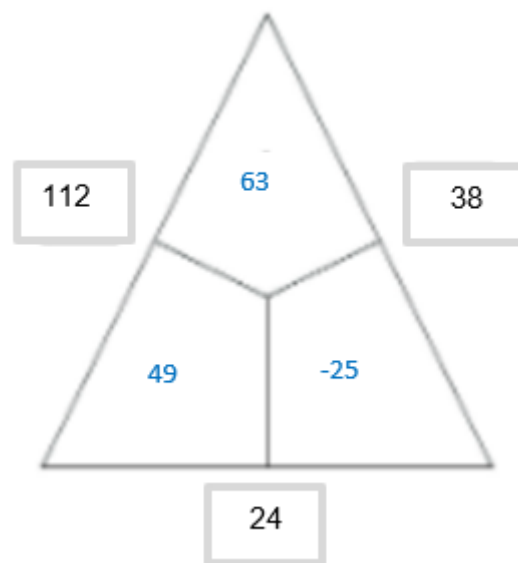
Om dergelijke driehoeken vlot op te lossen wordt gebruik gemaakt van 2 verbanden. Probeer beide verbanden aan te vullen/op te sporen.

1. De binnensom is de helft van de buitensom.
2. Zoek een verband tussen de binnensom en de omcirkelende getallen (zie voorgaande opdracht). Welk verband bestaat er tussen deze componenten?   
Neem de binnensom en trek daar een getal dat buiten de driehoek staat vanaf. Dit is het getal dat tegenover dit getal in de driehoek moet komen.



**Extra:**

Indien je tijd over hebt, probeer je onderstaande driehoek te vervullen.



### 3.3 Opdracht 3: smartgames



*Indien leerlingen klaar zijn met de vorige opdracht mogen ze, na controle van de leerkracht, van start gaan met opdracht 3. Het is dan de bedoeling dat 1 leerling een smartgame achteraan het lokaal bij de leerkracht komt ophalen. De leerkracht zal zelf een spel toewijzen.*

*Zoals eerder aangehaald, hoort bij iedere smartgame een fiche (zie bijlage). Op deze fiche staan, naast de inhoud en spelregels, ook de verschillende opgaven die de leerlingen bij deze smartgame moeten maken. Het is dus van groot belang dat de fiches bij iedere smartgame liggen.*

**Benodigdheden:** een smartgame + infofiche

**Tijd:** maximum 15 min.

Laat de vorige opdrachten controleren door de leerkracht voor je begint met een smartgame. Indien deze zijn gecontroleerd, kan je met je groep aan de slag met een smartgame.

De smartgame komt 1 leerling van de groep halen achteraan in de klas. Bij ieder spel hoort een fiche waarop staat uitgelegd hoe de smartgame gespeeld wordt. Op deze fiche staat ook aangegeven welke opgaven je speelt van het spel. Volg dus de fiche!

Enkele **afspraken**:

- ✓ Draag zorg voor het materiaal.
- ✓ Zorg dat het spel volledig blijft.
- ✓ Speel en overleg op een rustige manier.

**Opmerking:** Bij het overtreden van deze afspraken wordt er direct overgegaan naar het vervolg van de les!

## 4. Het stappenplan

*In dit lesonderdeel zullen leerlingen leren hoe ze kunnen denken als een computer. Het is zo dat computationeel denken steeds volgens een vast stappenplan verloopt. Ook digitaal worden deze stappen, weliswaar onzichtbaar, gevolgd. Iedere stap wordt uitgelegd a.d.h.v. concrete voorbeelden. Leerlingen worden zelf ook handiger in het begrijpen en toepassen van deze stappen doordat ieder stukje wordt ondersteund met oefeningen.*

*De inhoud zal klassikaal gegeven worden. Oefeningen worden afhankelijk per stap klassikaal, per 2, individueel ... gemaakt. Dit wordt in iedere handleiding vermeld.*

*Op het einde van dit stappenplan zal de leerkracht een exit card afnemen. Hierop vraagt de leerkracht naar de tot nu toe behandelde kennis. Bovendien zal de leerkracht toetsen in hoeverre de leerlingen de link tussen wiskunde en biologie begrepen hebben.*

*Aan het begin van het volgende lesuur/lesblok kan er een introcard afgenomen worden. Dit om te achterhalen in hoeverre de leerlingen de geziene leerstof nog onthouden hebben.*

Computationeel denken is een manier van probleemoplossend denken. Het is een aanpak om problemen op te lossen door het gebruik van talloze digitale methodes. Maar voordat we een computer kunnen gebruiken als hulpmiddel, moeten we goed kunnen begrijpen hoe de computer werkt.

Vriendelijk vragen aan een computer om een probleem op te lossen zal je niet ver brengen. Dat zal een computer niet begrijpen. Maar je probleem opdelen in kortere acties en deze vertalen naar concrete stappen zal je al veel verder brengen.

In dit deeltje leer je, met behulp van een stappenplan, denken als een echte computer.



## 4.1 Decompositie



*Binnen deze stap zal de leerkracht het inleidend probleem klassikaal met de leerlingen overlopen. De leerlingen proberen op basis hiervan het kenniskader aan te vullen. De leerlingen maken de oefening individueel. Het is een oefening die focust op het ontleden van een groter probleem. Leerlingen zullen hierbij terug moeten grijpen naar voorkennis van de eerste graad. Automatisch zullen ze het gegeven probleem dus moeten ontleden.*

De **eerste stap** van het stappenplan van het computationeel denken is decompositie.

### Inleidend probleem

Stel je hebt volgend probleem gegeven: 'Maak een uienpreparaat.'

Normaal gezien komen er dan een aantal vragen in je op. Noteer hieronder een aantal van de vragen die je jezelf stelt.

- Wat heb ik allemaal nodig om een preparaat te maken?
- Hoe moet ik dit preparaat maken?
- Is er iets waar ik op moet letten bij het maken van dit preparaat?
- Wat moet ik achteraf doen met dit preparaat?
- ...

Waarom kan je niet meteen aan de slag met het gegeven probleem?

Het gegeven probleem is niet gedetailleerd genoeg. Om het op te kunnen lossen, moeten de deelproblemen eerst beantwoord worden. De deelproblemen zijn elk op zich makkelijker te begrijpen en op te lossen.

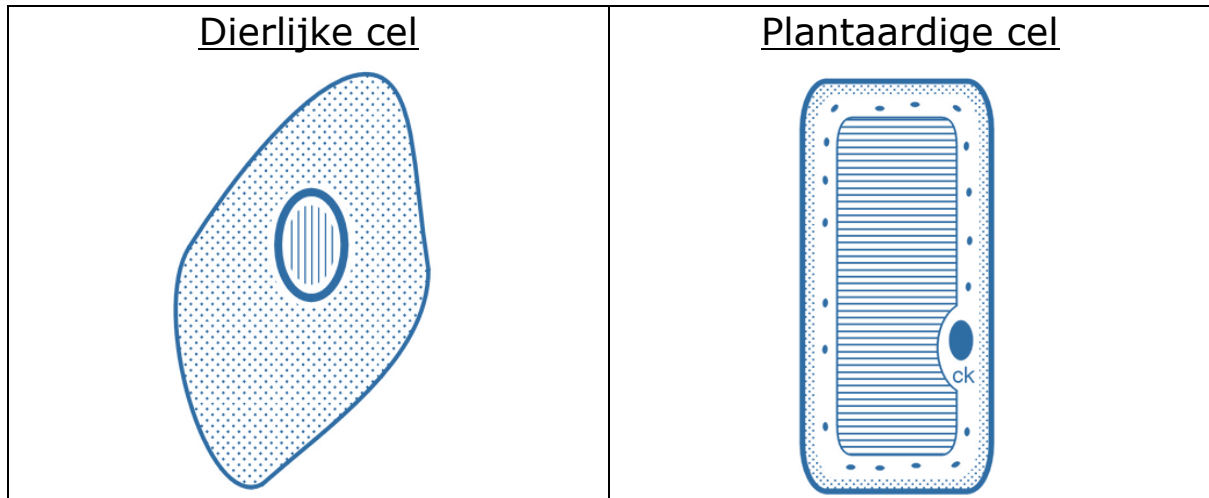
Vertrekkend vanuit 1 probleem ontstaan er dus tal van andere problemen. Ook voor een computer zijn deze afzonderlijke problemen beter te begrijpen.

**Decompositie is** het ontleden van een complex probleem in deelproblemen. Deze zijn afzonderlijk beter te begrijpen en op te lossen. Het oplossen van deze deelproblemen zorgt er echter voor dat het grote probleem opgelost geraakt.



### Tijd voor oefeningen

Probleem: Teken een dierlijke- en plantaardige cel in onderstaande kaders.



Wat heb je jezelf afgevraagd vooraleer je aan deze tekeningen kon beginnen?

- Hoe ziet een dierlijke- en plantaardige cel eruit?
- Welke celorganellen bevatten een dierlijke- en plantaardige cel?
- In wat verschilt de dierlijke cel met de plantaardige cel?
- ...

Je hebt je deze vragen dus misschien onbewust gesteld vooraleer je kon beginnen tekenen. Eens deze vragen werden beantwoord, kon je dus beginnen tekenen.

## 4.2 Patroonherkenning



Bij deze stap zullen leerlingen verwonderd zijn wanneer ze zien dat enorm veel natuurelementen een bepaald patroon vertonen. Ze zullen op deze manier mogelijk een fascinerendere blik op de natuur creëren.

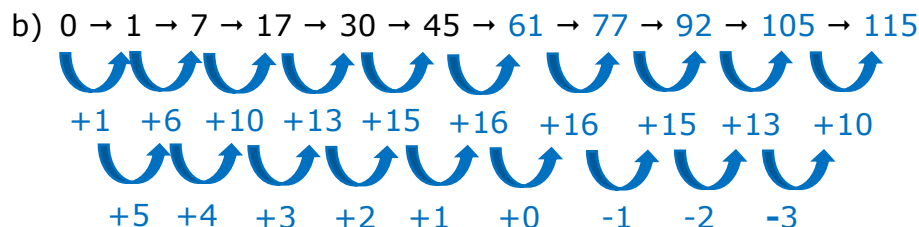
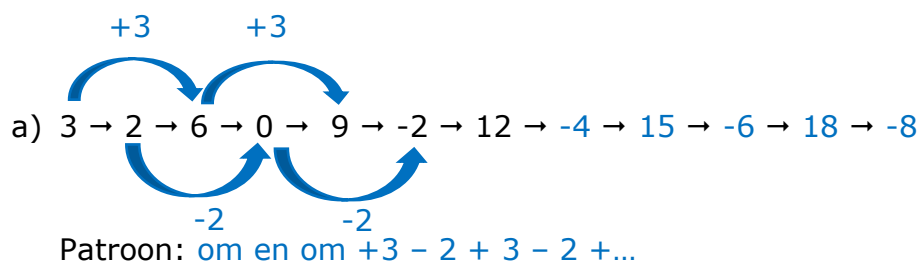
Leerlingen zullen eerst oefening a voorbereiden. Hierna volgt een klassikale verbetering. Vervolgens bespreekt de leerkracht het weetje i.v.m. de rij van Fibonacci in Python. Ten slotte gaan ze individueel aan de slag gaan met het herkennen en aanvullen van patronen. Hierna volgt een OLG.

De **tweede stap** van het stappenplan van het computationeel denken is patroonherkenning.



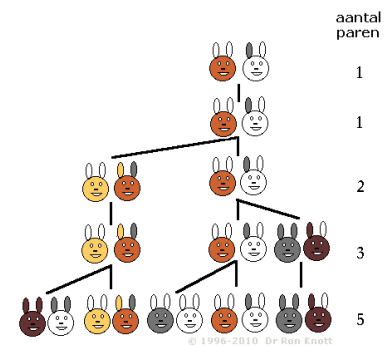
Tijd voor oefeningen: patronen herkennen en aanvullen

Vul volgende patronen verder aan (minstens 5 getallen toevoegen bij getalpatronen). Geef ook steeds het patroon weer.



c) Rij van Fibonacci:

Fibonacci is vooral bekend voor de konijnenreeks of de rij van Fibonacci. Op een bepaald moment begon hij met konijnen te fokken. Hij begon (uiteindelijk) met twee, maar al snel had hij er drie. Na het bestuderen van vermenigvuldigen van deze dieren kwam hij tot de reeks:



*Figuur 1: rij van Fibonacci*

0 → 1 → 1 → 2 → 3 → 5 → 8 → 13 → 21 → 34 → 55 → 89 → 144

Patroon: som van 2 voorgaande getallen

### Verwonderingsweetje:

De rij van Fibonacci kan je ook programmeren in Python. Heel wat sneller om op deze manier tot je uitkomst te komen.

Het 11de getal:

```
main.py  [ ] [ ] [Run] [Shell]
1 ▾ def fib(N): 144
2 ▾     if N < 2: > |
3     |         return 1
4     |         return (fib(N-1) + fib(N-2))
5
6 resultaat = fib(11)
7 print(resultaat)
```

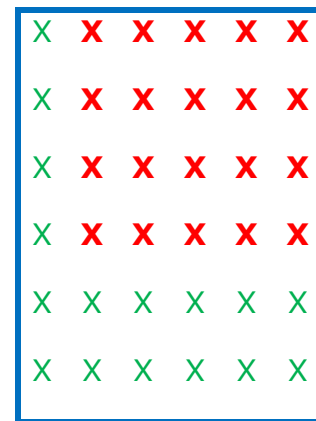
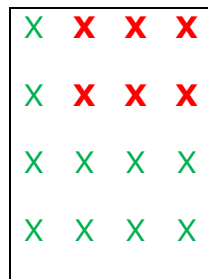
*Figuur 2: 11de getal van de rij van Fibonacci*

Stel je maar eens voor dat je het 44<sup>ste</sup> getal moet berekenen. Dit zal je dagen kosten. Kijk maar eens hoe snel Python dit voor jou kan berekenen.

```
main.py  [ ] [ ] Run Shell
1 ▸ def fib(N):
2 ▸     if N < 2:
3 ▸         return 1
4 ▸     return (fib(N-1) + fib(N-2))
5
6 resultaat = fib(44)
7 print(resultaat)
```

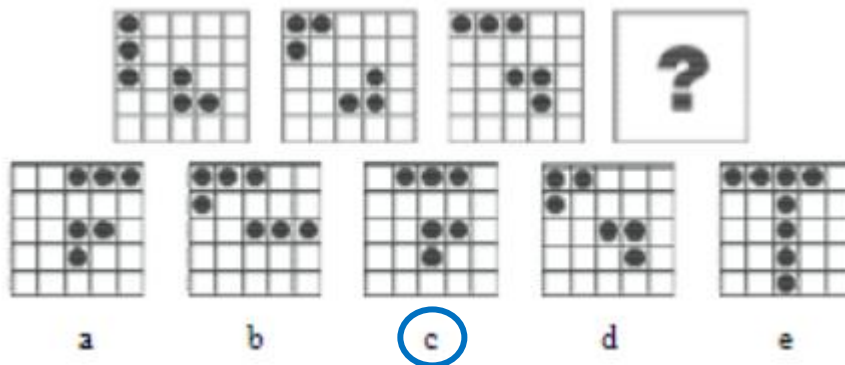
*Figuur 3: 44ste getal van de rij van Fibonacci*

d) Teken het eerst volgende patroon:



Oplossing

e) Welke figuur moet logischerwijs op de plaats van het vraagteken staan? Omcirkel het juiste antwoord.



**Patronen zijn** steeds terugkerende kenmerken. Eens ergens een patroon in teruggevonden kan worden, zal het probleem makkelijker op te lossen zijn.





### Even over het muurtje kijken in de biologie:

Patronen komen veelvuldig voor in het dagelijks leven. Hieronder enkele voorbeelden:



*Figuur 4: Patronen in de biologie*

Waar in het dagelijks leven kom je nog overal patronen tegen?

Eigen voorbeelden; Zonnebloem, bloemkool, dennenappel,...

#### Weetjes:

##### **De honinggraat: zie figuur 3**

We zien steeds dezelfde figuur terugkomen. Dit zijn de zeshoekige cellen waaruit een honinggraat bestaat. De bij gebruikt deze cellen als kraamkamer en opslagplaats voor stuifmeel en honing. Ze kunnen deze cellen ook afsluiten met een dekseltje. De bijen maken gebruik van de zeshoekige vorm omdat dit de efficiëntste stabiele indeling is. Deze vorm zorgt ervoor dat ze een grote opslagruimte hebben om honing op te slaan.

##### **De nautilus: zie figuur 4 en 5**

Als we kijken naar de nautilus, zien we dat deze schelp bestaat uit verschillende compartimenten. In het laatste compartiment leeft het dier, dit is zijn woonruimte. De rest van de compartimenten zijn luchtkamers die een rol spelen bij het drijfvermogen. Dit doordat de kamers zichzelf kunnen vullen met gas en water.

Wanneer de woonruimte te klein wordt voor de nautilus zal deze dus uitbreiden. Het organisme zal een nieuw compartiment aanmaken, dat dan zijn nieuwe woonruimte wordt. De schelp zal steeds groter worden om het lijf van de nautilus te laten passen.

We kunnen hierin een wiskundig patroon herkennen. Zo bestaat de schelp uit drie windingen, die steeds met factor drie breder worden. De binnenste winding is ongeveer één centimeter breed, de tweede drie centimeter en de laatste negen centimeter.



*Figuur 5: Honinggraat bij*

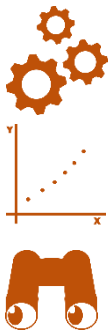


*Figuur 7: Nautilus (schelp)*



*Figuur 6: Nautilus (organisme)*

## 4.3 Abstractie



*Binnen deze stap leren de leerlingen wat abstractie is en waarom het belangrijk is om bij het oplossen het probleem abstracter voor te stellen. Er is namelijk vaak veel overbodige informatie die het probleem ingewikkelder kunnen laten lijken dan het in werkelijkheid is.*

*Om deze stap in te leiden maken we gebruik van verwondering. Hierbij krijgen de leerlingen een scène vanuit de film 'Inside out' te zien. In deze scène gaan 3 van de personages door een tunnel genaamd "abstract denken". Terwijl ze hier doorheen wandelen, veranderen ze zelf in steeds simpelere en dus abstractere vormen. Aan de hand van een OLG rond deze scène, komt de leerkracht samen met de leerlingen tot het begrip abstractie.*

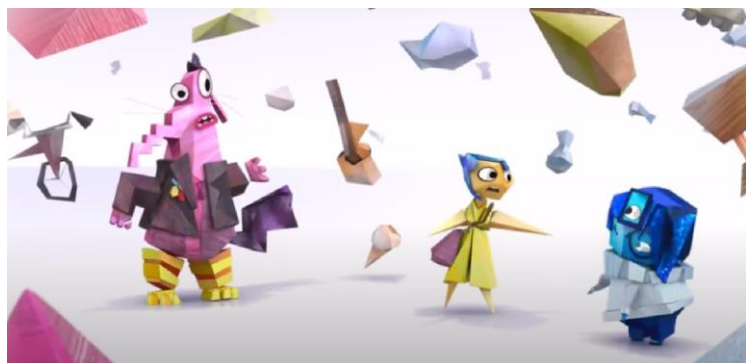
<https://www.youtube.com/watch?v=606396EJcJo> (1:00 – 2:45)

*Hierna werken de leerlingen zelfstandig aan de oefeningen. Het doel van de eerste oefening is vooral om de leerlingen te laten zien dat éénzelfde item op verschillende manieren abstract kan voorstellen. Binnen de tweede oefening is het doel dat de leerlingen een beter zicht krijgen van abstractie in het dagelijkse leven. Bij de derde oefening worden de leerlingen uitgedaagd om na te denken over hoe ze zelf iets abstracter kunnen voorstellen. Hierna is het aangewezen om een klein OLG te houden over hoe de leerlingen kijken naar abstractie. In dit OLG moet aan bod komen dat er bij abstractie getracht wordt om alle overbodige informatie weg te laten zonder hierbij essentiële informatie kwijt te spelen.*

De **derde stap** van het stappenplan van het computationeel denken is abstractie.

Wat gebeurt er met de personages wanneer ze door de abstractietunnel lopen?

*Ze doorlopen verschillende fases. In elke fase worden ze steeds simpeler en dus abstracter voorgesteld.*



Figuur 8: Fragment 'Inside out'

**Abstractie is** het weglaten van overbodige informatie. Door een probleem abstracter voor te stellen wordt het minder ingewikkeld en is het dus makkelijker op te lossen.

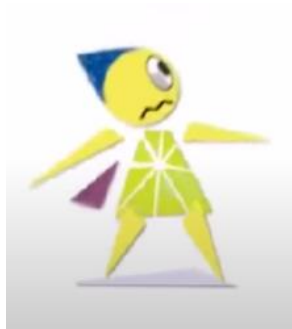


## Tijd voor oefeningen

### Oefening 1:

Plaats in onderstaande tabel de nummers van onderstaande afbeeldingen in de juiste volgorde: van minst abstract naar meest abstract.

#### Reeks 1:



1



2



3



4

#### Reeks 2:



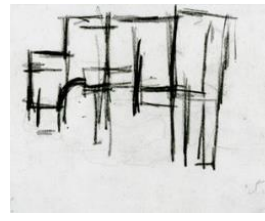
1



2



3



4

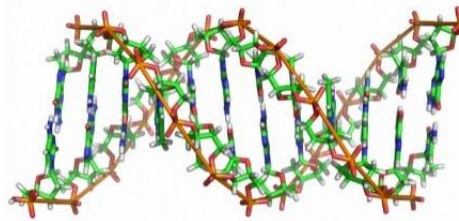
	<div>Minst abstract</div> <div>↔</div> <div>Meest abstract</div>			
Reeks 1	2	4	1	3
Reeks 2	3	1	4	2



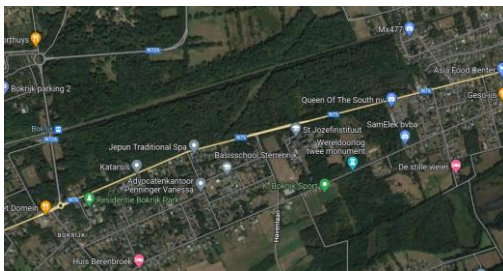
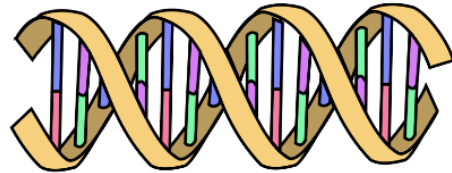
## Oefening 2:

Abstractie is iets dat we vaak in ons leven gebruiken zonder er bij stil te staan. Kan je zelf een aantal voorbeelden geven wanneer we het gebruiken en waarom?

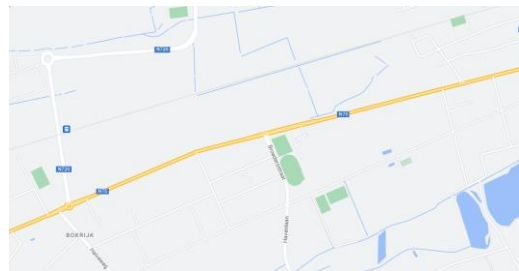
Hieronder staan een aantal afbeeldingen die ter inspiratie kunnen dienen.



⇒



⇒



Wanneer we ergens de weg moeten zoeken is dit makkelijker met een kaart waar niet te veel gegevens op staan.

Door de details weg te laten bij een tekening, is het makkelijker om bepaalde dingen (zoals bv. DNA) voor te stellen.

### Oefening 3:

Louis gaat met vrienden naar de Efteling. Zijn vrienden gaan naar de Python, maar hij blijft liever nog wat langer in het kasteel van Aladin. Hij besluit om een kwartiertje later ook naar de Python te gaan, maar zijn vrienden hebben het plannetje meegenomen. Hij vindt langs het kasteel van Aladin een grondplan. Om niet verkeerd te lopen, noteert hij de route die hij moet wandelen op een Post-it.

Hieronder zie je het bord met de plattegrond.



*Figuur 9: Plattegrond Efteling*

Kan jij de weg die Louis moet volgen om tot bij de Python te geraken, tekenen?



## 4.4 Algoritme



*Binnen deze stap zullen de leerlingen kennismaken met het begrip algoritme. De leerkracht zal dit aanbrengen aan de hand van het maken van een preparaat. Het is de bedoeling dat de leerlingen, door het geven van instructies, de leerkracht een preparaat van een ajuinvlies laten maken. Als leerkracht heeft u de taak om de instructies van de leerlingen op te volgen. Als u dit heel letterlijk neemt, begrijpen ze dat kleine woorden een grote invloed kunnen hebben. Op deze manier leren ze dat bij het schrijven van een algoritme het belangrijk is om zeer precies te zijn. De leerkracht kan hier al kort een koppeling maken naar het deel foutenanalyse, debuggen. Hierna gaan ze individueel aan de slag met de oefeningen. Deze oefeningen hebben betrekking op de verschillende stappen, die aanwezig zijn binnen een instructie, en het correct kunnen verwoorden van deze stappen. We gaan hier ook al een eerste keer komen tot het programmeren. Bij oefening 1 zullen de leerlingen een schema schrijven. Dit zal naar Python vertaald worden. Dit is het programma waar wij in deze bundel mee aan de slag gaan.*

De **vierde stap** van het stappenplan van het computationeel denken is algoritme.

### Inleidend voorbeeld

We hebben allemaal al wel eens een preparaat gemaakt. Het is hierbij ook heel belangrijk om dit rustig en stap voor stap te doen. Schrijf hieronder het stappenplan voor het maken van een preparaat van een ajuinvlies uit.

1. Neem een ajuin
2. Snij een ajuin overlangs door
3. Neem een pincet en haal een rok uit de ajuin
4. Neem een vliesje met een pincet
5. Leg het vliesje op een preparaatglaasje
6. Druppel op het vliesje een druppel water
7. Leg één dekglasje op het preparaat
8. Het preparaat is klaar

De bedoeling is dat je de leerkracht een ajuinpreparaat laat maken door te vertellen welke stappen er doorlopen moeten worden. Succes!

Ging dit vlot bij de leerkracht? Hoe kwam het dat het niet zo ging als verwacht?

Nee, mijn instructies waren niet duidelijk genoeg.

**Een algoritme is een eindige reeks instructies waarmee een taak automatisch wordt uitgevoerd.**

Een algoritme kan je dus vergelijken met een stappenplan waarbij je stap voor stap een probleem snel en handig kan oplossen.



Welke algoritmes ken je vanuit het dagelijks leven?

Persoonlijke antwoorden, enkele voorbeelden:

- Ochtend
- Bouwplan van Lego
- Recept in kookboek
- De weg naar het school fietsen en terug naar huis
- Avondeten
- ...



Even over het muurtje kijken in de biologie

Er zijn natuurlijk ook een aantal algoritmes in de biologie, kan je er zelf enkele opsommen? Onderstaande afbeeldingen kunnen je een duwtje in de rug geven.



*Figuur 10: Algoritmes in de biologie*

Bouwen van een bloemkool, steeds terugkomende kopieën, net zoals de romanescos en de zonnebloem. De nautilus die zijn schelp zal hervormen als het te klein wordt, dit steeds op dezelfde manier. Mieren die hun weg zoeken naar het nest door een geurspoor te maken en zo steeds via dezelfde weg naar het voedsel te gaan.





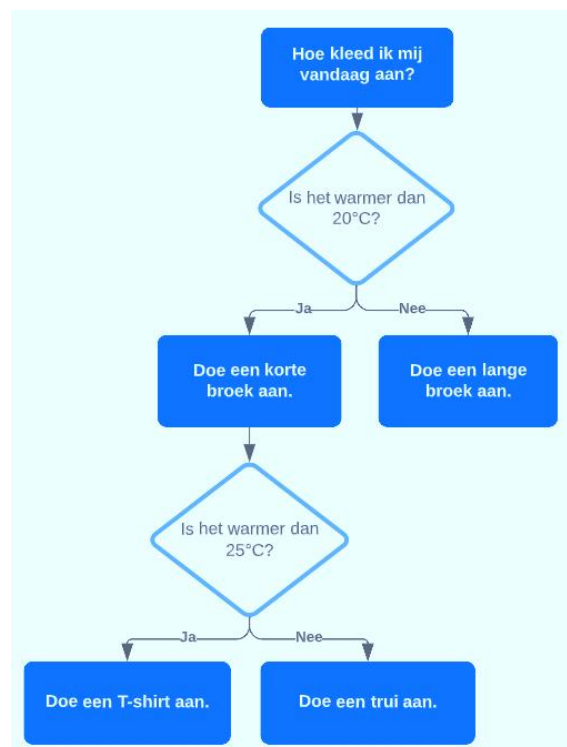
## Tijd voor oefeningen

### Oefening 1:

Als het warmer is dan 20° graden, dan draag ik een korte broek. Wanneer het kouder is dan 20° draag ik een lange broek. Voor mijn outfit heb ik ook nog een trui of T-shirt nodig. Als het warmer is dan 25° doe ik een T-shirt aan. Anders draag ik een trui.

Stel hiervoor een algoritme op.

Oplossing:



### Verwonderingsweetje:

Het algoritme dat je hiervoor opstelt, kan je ook in Python programmeren. Wanneer je dit doet, zal Python aan jou de vraag 'Wat is de temperatuur?' stellen. Aan de hand van het antwoord zal het programma je zeggen welke kleding je het beste aandoet.

Hieronder het geprogrammeerde voorbeeld:



The screenshot shows a code editor with a file named 'main.py'. The code is a Python script that asks the user for their temperature and provides clothing advice based on the input. The code is as follows:

```
1  temperatuur= float(input("Wat is de temperatuur?"))
2  if temperatuur>20:
3      print( "Doe een korte broek aan.")
4      if temperatuur>25:
5          print ("Draag een T-shirt.")
6      else:
7          print ("Draag een trui.")
8  else:
9      print( "Doe een lange broek aan.")
10 print( "Draag een trui.")
```

To the right of the code editor is a 'Shell' window showing the output of the program. The user has entered '26' for the temperature. The program's output is:

```
Wat is de temperatuur?26
Doe een korte broek aan.
Draag een T-shirt.
>
```

*Figuur 11: Algoritme omtrent temperatuur*

Op de site <https://www.kanikeenkortebroekaan.nl/> is bovenstaand programma geprogrammeerd, rekening houdend met de temperatuur en de regenkans. Deze site zal je dus elke dag vertellen of je een korte broek aan kan doen.

## Oefening 2:

Als ik naar school ga, kan ik me op verschillende wijzen verplaatsen. Ik heb de keuze uit te voet, met de fiets of met de auto. Wanneer het in de winter heeft gevoren en de wegen glad zijn, kies ik er voor om te voet naar school te gaan. Als het niet heeft gevoren, ga ik met de fiets of met de auto. Dit hangt natuurlijk af van het weer. Regent het, dan ga ik met de auto. Is het droog, dan neem ik de fiets.

Stel hiervoor een algoritme op.

Oplossing:



## 4.5 Foutenanalyse



*Binnen deze stap zullen leerlingen in aanraking komen met foutenanalyse. Hierbij zullen leerlingen zelf opzoek gaan naar gemaakte fouten.*



*Wanneer leerlingen later aan de slag zullen gaan met het begeleid zelfstandig werk zullen ze hier voortdurend mee te maken krijgen. Het is bij het programmeren dan ook een veelvoorkomend probleem.*



*De oefeningen maken de leerlingen per 2.*

*Indien leerlingen al gezien hebben wat DNA, een chromosoom en een allel is, kan hier snel overeen gegaan worden door middel van een klasgesprek. Indien leerlingen nog niet geleerd hebben over deze termen, is dit een korte inleiding tot de uitleg van DNA. Het is een zeer eenvoudige voorstelling. De leerlingen zullen dit nog veel uitgebreider leren. De leerkracht kan gebruik maken van de volgende video. Hierbij kan de leerkracht aanhalen dat er bij de analyse van het menselijk lichaam vaak fouten opduiken. Ook onderzoekers kunnen in het laboratorium fouten analyseren.*

[https://www.youtube.com/watch?v=UK7kT\\_22QfI&t=2s](https://www.youtube.com/watch?v=UK7kT_22QfI&t=2s)

De **laatste stap** van het stappenplan van het computationeel denken is foutenanalyse.

### Inleidend voorbeeld

Stel je komt na berekeningen uit dat een persoon 625 kg weegt. Hoe kan je achterhalen waar de fout ligt?

Verschillende mogelijke antwoorden zijn mogelijk vanuit de leerlingen. Hieronder zijn enkele correcte voorbeelden gegeven.

- Door opnieuw de opgave te lezen enkele overschrijffouten ontdekken.
- Door tussenuitkomsten opnieuw uit te rekenen een telfout ontdekken.
- Door het rekenmachine grondig na te kijken op het overschrijven van fouten.

Als we een vraagstuk hebben opgelost, gaan we altijd kijken voor mogelijke fouten. Dit is van groot belang als we later in deze lessenreeks gaan programmeren. Stel je hebt een fout ingegeven, dan is het aan jou om deze fout eruit te kunnen halen door een foutenanalyse te doen. Dit proces noemen we 'debuggen'.

**Foutenanalyse is het nagaan van een oefening om een fout eruit te halen, dit door middel van tussenstappen te controleren.**



### Even over het muurtje kijken in de biologie

In het filmpje wordt er uitleg gegeven over de opbouw van DNA. Indien je na dit filmpje nog extra uitleg wil, kan je deze altijd lezen onder de link.

[https://www.youtube.com/watch?v=UK7kT\\_22QfI&t=2s](https://www.youtube.com/watch?v=UK7kT_22QfI&t=2s)

Wanneer er aan voortplanting gedaan zal worden, zullen we altijd moeten spreken over het stukje erfelijkheid dat hiermee verbonden is. Het kind zal zowel van de moeder als van de vader erfelijke factoren meekrijgen, maar hoe zit dat nu juist?

Als we spreken van erfelijkheid, hebben we het over overerven. We krijgen een deeltje mee dat al van iemand anders was.

Iedere cel bevat een celkern waarin het erfelijk materiaal of DNA gelegen is. Dat wordt gedragen door verschillende chromosomen die elk opgebouwd zijn uit 2 chromatiden. Het DNA bevat de erfelijke code die ervoor zorgt dat ieder individu uniek is. Deze code wordt gevormd door combinaties van 4 verschillende basen: A (adenine), C (cytosine), T (thymine) en G (guanine).

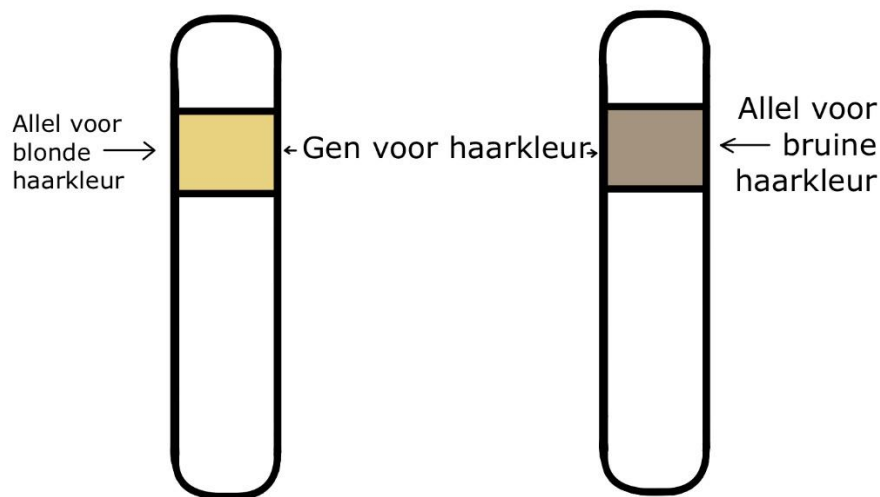


*Figuur 12: Chromosomen waarbij chromosoom 1 bestaat uit allel A en allel B en waarbij chromosoom 2 bestaat uit allel C en allel D*

Wanneer een nakomeling gevormd wordt, zullen de chromosomen van deze nakomeling ontstaan uit de combinatie van 1 chromatide van de vader en 1 chromatide van de moeder.

De totale erfelijke code wordt opgedeeld in kleine codedeeltjes die elk verantwoordelijk zijn voor een specifiek kenmerk zoals bijvoorbeeld de oogkleur, de haarkleur, bloedgroep ... Deze codedeeltjes worden genen genoemd.

Wanneer er een nakomeling gevormd wordt, zal deze zowel van de moeder als de vader hetzelfde basisgen doorkrijgen maar het resultaat zal anders zijn. Dit komt doordat de code in het gen dat bestaat uit A, C, T en G een andere volgorde heeft bij de moeder en de vader. Deze twee varianten in een gen noemen we een allel.

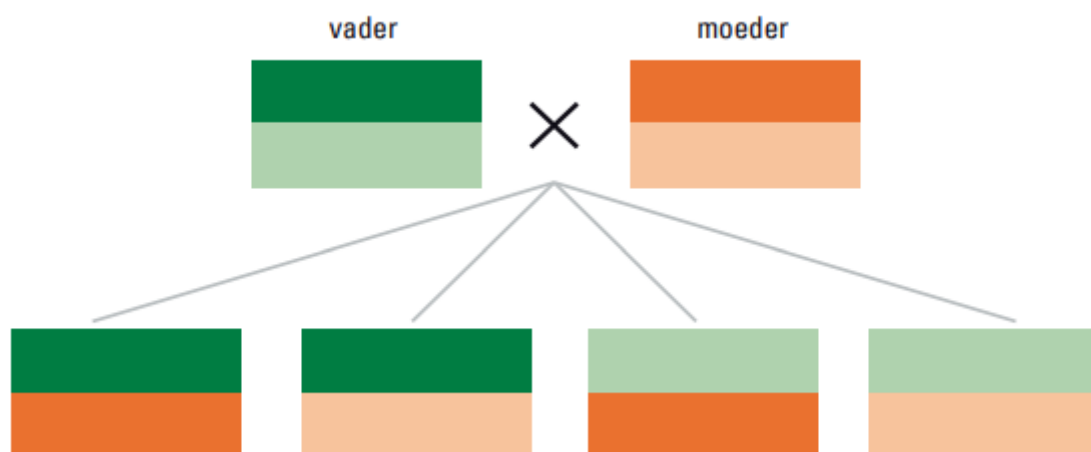


Figuur 13: Genen met 2 allelen voor haarkleur

#### Weetje:

Van slechts 5% van het DNA kennen we de functie. Of de andere 95% een functie heeft, weet nog niemand.

Op de figuur 10 zie je een voorbeeld van hoe de overerving of erfelijkheid gebeurt.



Figuur 14: Iedere nakomeling krijgt één van de twee allelen van elke ouder; maar welke wordt door toeval bepaald

#### Verwonderingsweetje: Dawkins genetic algoritme

Richard Dawkins is één van de bekendste schrijvers van de evolutietheorie. Hij beschrijft in zijn werken dat mutaties (fouten) in de genetische code zorgen voor grotere overlevingskansen. Dit hebben jullie ook al benoemd onder de term 'survival of the fittest'. Hiervoor ontwikkelde hij een computersimulatie die ervoor zorgde dat zijn onderzoek sneller verliep.

Indien je hier nog interesse voor hebt, bekijk dan zeker volgende site: <https://sargasso.nl/de-blinde-horlogemaker/>

## Tijd voor oefeningen

### Oefening 1:

In onderstaande oefening staat een bewijs. Dit bewijs zegt dat  $2=1$ . Uiteraard weten jullie dat dit niet klopt. Schrijf bij elke stap uit wat je doet en zoek op deze manier de fout in het bewijs.

#### Het bewijs van $2 = 1$

$a = b$	
$a^2 = ab$	Vermenigvuldigen met $a$
$2a^2 = a^2 + ab$	Aan beide kanten van het gelijkheidsteken $\rightarrow +a^2$
$2a^2 - 2ab = a^2 - ab$	Aan beide kanten van het gelijkheidsteken $\rightarrow -2ab$
$2(a^2 - ab) = a^2 - ab$	2 afzonderen uit linkerlid
$2 = \frac{a^2 - ab}{a^2 - ab}$	$a^2 - ab$ naar de andere kant overbrengen
$2 = 1$	$\frac{a^2 - ab}{a^2 - ab}$ is 1, uitwerken breuk

Welke fout zit er in dit bewijs?

In de 2<sup>de</sup> regel zeggen we dat  $a^2=ab$ , als we  $a^2-ab$  doen, is dit gelijk aan 0. We kunnen dus in de 6<sup>de</sup> regel niet delen door  $a^2-ab$  want delen door 0 is niet toegelaten.

## Oefening 2:

In onderstaande tabel staan alle mogelijke bloedgroepen die de mens kan hebben. In de rechter kolom staan de mogelijke combinaties waaruit de desbetreffende bloedgroep gevormd kan worden. Dit is de combinatie van de bloedgroepen, die je ouders kunnen hebben. Deze tabel dient ter ondersteuning van de volgende oefening.

Bloedgroepen	Combinaties die behoren tot deze bloedgroep
Bloedgroep A	AA / AO / OA
Bloedgroep B	BB / BO / OB
Bloedgroep O	OO
Bloedgroep AB	AB

Lola wil onderzoeken of haar mama en papa wel echt haar ouders zijn. Ze doet dit door de bloedgroepen van haarzelf en beide ouders te vergelijken. Zijzelf heeft bloedgroep OO, haar mama heeft bloedgroep AO en haar papa heeft bloedgroep AB. Ga na of de ouders van Lola wel echt haar mama en papa kunnen zijn. Stel hiervoor een schema op.

### Oplossing:

		Mama	
		A	O
Papa	A	AA	AO
	B	AB	BO
		Kinderen	

De mama van Lola kan haar echte mama zijn maar de papa kan niet de echte vader zijn. Dan zou er ergens in de bloedgroep van hem een O moeten voorkomen.





## 4.6 Overzicht van het stappenplan

*In dit deel staat een overzicht van de delen van het stappenplan, leerlingen kunnen hier de definitie nog eens nalezen. Dit is voor hun dus een handig overzichtsblad. Onder het kader is er ook nog een QR-code, deze leidt je verder naar een Prezi. Hierin zal op een leuke manier het stappenplan worden herhaald. Leerlingen kunnen dit dan thuis bekijken, als voorbereiding op wat nog zal volgen.*

- 1. Decompositie is** het ontleden van een complex probleem in deelproblemen. Deze zijn afzonderlijk beter te begrijpen en op te lossen. Het oplossen van deze deelproblemen zorgt er echter voor dat het grote probleem opgelost geraakt.
- 2. Patronen zijn** steeds terugkerende kenmerken. Eens ergens een patroon in teruggevonden kan worden, zal het probleem makkelijker op te lossen zijn.
- 3. Abstractie is** het weglaten van overbodige informatie. Door een probleem abstracter voor te stellen wordt het minder ingewikkeld en is het dus makkelijker op te lossen.
- 4. Een algoritme is** een eindige reeks instructies waarmee een taak automatisch wordt uitgevoerd.
- 5. Foutenanalyse is** het nagaan van een oefening om een fout eruit te halen. Dit doormiddel van tussenstappen te controleren.

Scan volgende QR-code om een samenvatting te krijgen van het stappenplan.





## 4.7 Algemene oefeningen



*In dit onderdeel krijgen de leerlingen oefeningen waar de verschillende stappen van het stappenplan in voorkomen. De leerlingen gaan de verschillende stappen dus moeten herkennen en kunnen toepassen.*

*Oefening 2 is een inleidende oefening op het programmeren. De leerlingen volgen onbewust het algoritme dat ze eerder leerde. Hierna wordt de overschakeling gemaakt naar het programmeren, want ook vele apps en computers kunnen planten determineren a.d.h.v. dit algoritme.*

Oefening 1:

De bedoeling van deze oefening is dat jullie de getallen van 1 en met 200 bij elkaar optellen.

a) Hoe zou je dit doen?

$$1+2+3+4+5+6+\dots+196+197+198+199+200=\dots$$

b) We zien dat als we dit als een som willen opschrijven het heel veel werk gaat zijn. Kunnen we dit in kleiner stukken opdelen? Kunnen we delen samen nemen zodat we altijd op hetzelfde getal terecht komen?

$$1+200 = 201$$

$$2+199 = 201$$

$$3+198 = 201$$

...

$$100+101=201$$

c) Hoeveel koppels van dit getal zullen we dan krijgen?

We zullen 100 koppels kunnen vormen van elke 201.

d) Welke stappen van het stappenplan hebben we nu gebruikt? En waarom?

Decompositie → We hebben van iets uitgebreid een kleiner idee gemaakt.

Abstractie → We hebben het probleem minder ingewikkeld gemaakt.

e) Welke formules kunnen we opstellen voor het oplossen van deze som?

$$201 \cdot 100 = 20100$$

f) Kunnen we de formules ook veralgemenen om zo een som te maken voor getallen van 1 tot n?

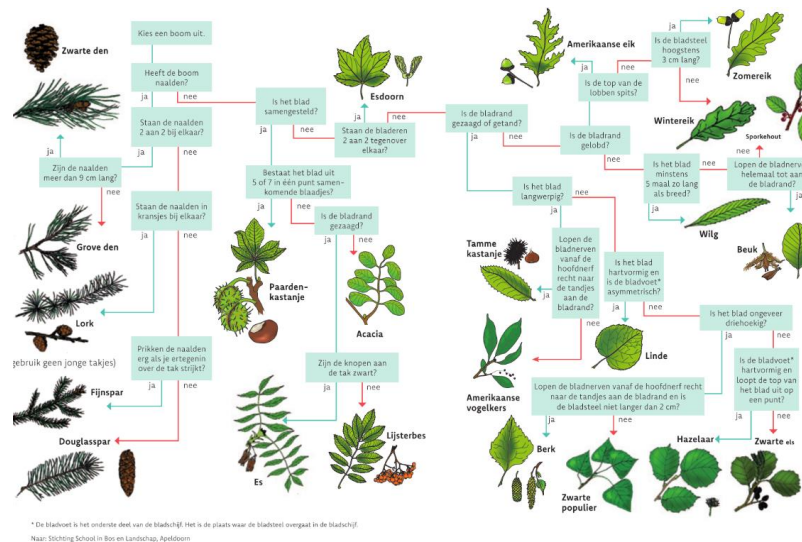
$$(n+1) \cdot \frac{n}{2} = \text{Som van de getallen}$$

## Oefening 2:

Je krijgt van de leerkracht per 2 een blad. Determineer het blad met behulp van een determinatiekaart.

Omcirkel op de determinatiekaart de verschillende tussenstappen.

### Determinatiekaart:



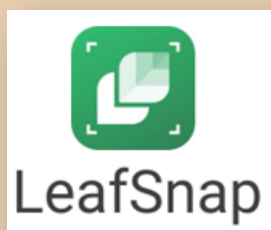
Figuur 15: Determinatietabel bladeren

Mijn blad is afkomstig van .....

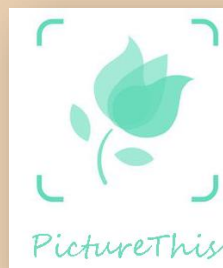
### Weetje:

Wist je dat je een veel sneller resultaat krijgt met behulp van een goede determinatie-app. Met een simpele scan weet je binnen enkele seconden welk blad je voor je hebt. Houd ook rekening met het feit dat deze niet altijd even correct zijn.

Hieronder enkele voorbeelden van apps die in de toekomst misschien handig kunnen zijn.



Figuur 17: logo LeafSnap



Figuur 16: logo PictureThis



Figuur 18: logo ObsIdentify

De apps zijn dus op een bepaalde manier geprogrammeerd om steeds hetzelfde algoritme uit te voeren. Dit komt overeen met het algoritme dat jij zonet op papier hebt moeten doorlopen.

## 5. Programmeren in Python



*In dit gedeelte zullen de leerlingen werken aan het programmeren in Python. Dit zal aangereikt worden door middel van een begeleid zelfstandig werk. Hiermee hopen we het onderzoekend en ontwerpend leren van de leerlingen te bevorderen. Dit door klassikaal te beginnen en ze daarna zelfstandig aan de slag te laten gaan.*

*Vanaf de oefeningenreeks binnen het onderdeel 'basisfuncties in Python' zullen de leerlingen volledig op eigen tempo verder werken. Echter zullen ze gedurende het zelfstandig werk wel op vaste momenten starten met nieuwe inhoud. Hierdoor lopen alle leerlingen ongeveer gelijk met de theorie. Sommige inhoud wordt voorzien van een begeleidende instructievideo. Deze vormen naast de uitgeschreven uitleg een extra ondersteuning.*

*Link naar de afspeellijst met alle video's :*

*<https://www.youtube.com/watch?v=7f90IFmsj6o&list=PLb7wduIUUvjWIRhtPIFC6S8lrm9kw8D-p>*

*Zelfstandige oefeningen dienen de leerlingen in een word-document te plakken. Dit doen ze door hun geprogrammeerde code te kopiëren en te plakken in het daarvoor voorziene document. Het sjabloon is terug te vinden in de bijlage. Dit zal ervoor zorgen dat de begeleidende leerkracht het traject van de leerlingen nauwgezet kan opvolgen.*

*Ter afsluiting van het project kan een quiz gespeeld worden. Deze bevindt zich in de bijlage.*

We nemen nu een duikje in de programmeerwereld. In dit hoofdstuk zullen jullie aan de slag gaan met een begeleid zelfstandig werk.

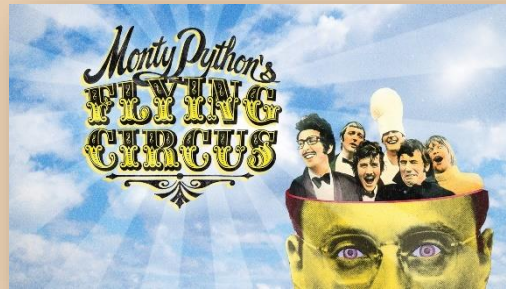
In dit deel is het belangrijk dat je steeds voorzien bent van het juiste materiaal: laptop, oortjes, kladpapier en je werkbundel. De werkomgeving die gebruikt wordt is: <https://www.programiz.com/python-programming/online-compiler/>

Je zal leren programmeren in de programmeertaal Python. Een uitstekende keuze om op ieder niveau aan de slag te gaan.

Weetje:

### Ontstaan van Python

Wist je dat de eerste versie van Python in het jaar 1991 is gelanceerd door de Nederlandse programmeur Guido van Rossem? Hij heeft de naam 'Python' bovendien niet lukraak gekozen. De naam stamt af van zijn grote interesse voor de komedieserie genaamd "Monty Python's Flying Circus". Hij zocht een naam die kort, uniek en een beetje mysterieus was.



*Figuur 19: Komedieserie "Monty Python's Flying Circus"*

Ondertussen is de programmeertaal al zo geëvolueerd met als gevolg dat het een van de populairste talen is, die bovendien wereldwijd wordt gebruikt. Nog maar een greep van grote bedrijven die Python gebruiken zijn: Instagram, Uber, Spotify, YouTube, ... Vanwege de eenvoud, het praktische karakter en de gunstige ontwikkelingssnelheid kozen ze voor deze taal. Dit voor bijvoorbeeld data science, oplossen van complexe algoritmes, webapplicaties, ... Ook wordt Python bij NASA gebruikt om apparatuur en ruimtevoertuigen te programmeren.

Behoorlijk indrukwekkend toch!

Waarom Python?

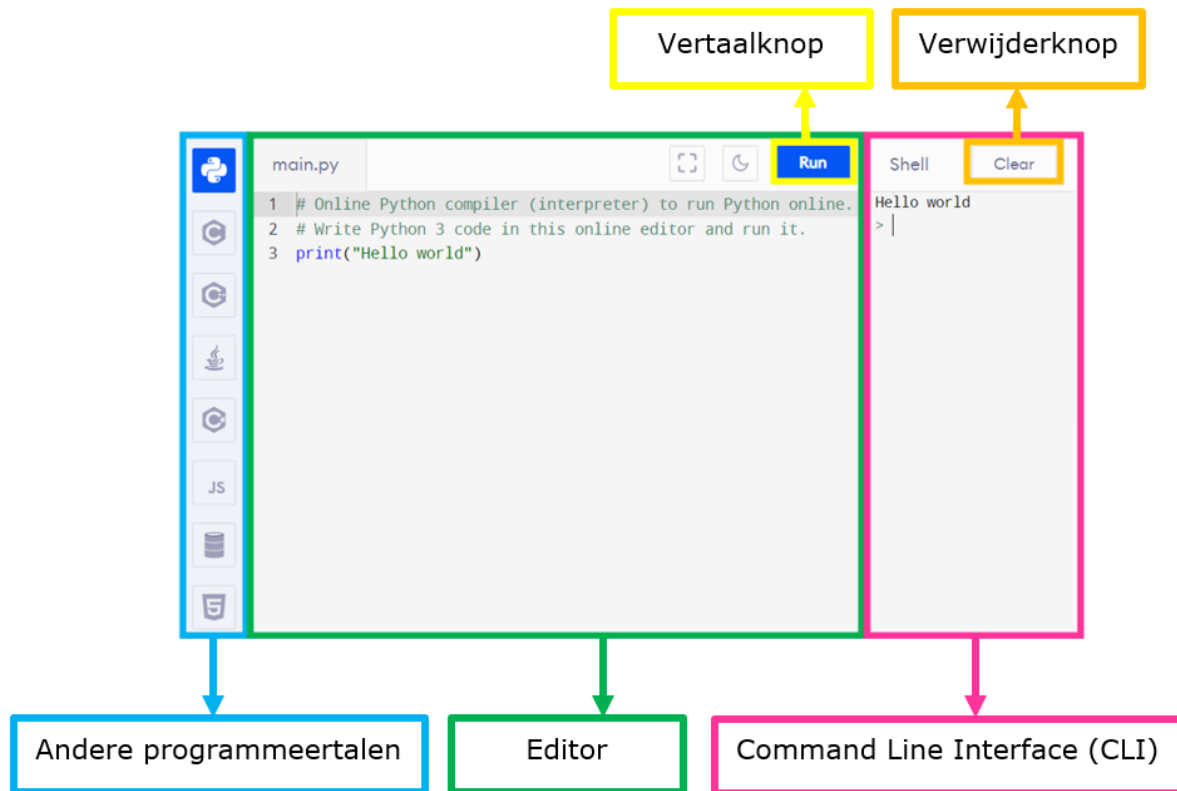
- Het is toegankelijk, laagdrempelig en toch relevant.
- Eenmaal je Python onder de knie hebt, kan je gemakkelijk overstappen naar een andere programmeertaal.
- Python kan door veel mensen worden gebruikt, denk maar aan ingenieurs, wiskundigen, economen, informatici, ...

## 5.1 De basis in de programmeerwereld

Je zal eerst de basis onder de knie moeten krijgen vooraleer aan het echte werk te kunnen beginnen. We bekijken nu enkele basisprincipes van Python.

## 5.1.1 Startscreen

Ga naar de programmeeromgeving (zie voorgaande link) en bekijk het startscreen. Volgende items kan je nu onderscheiden in het scherm. De bijhorende functies worden hieronder besproken.



Onderdelen startscreen	Functies
CLI	Door communicatie met Python wordt de programmacode regel per regel verwerkt.
Editor (bewerkingsomgeving)	Laat toe om in één keer verschillende regels code te schrijven of aan te passen.
Run	Deze knop wordt gebruikt wanneer je een programma wilt laten lopen.
Clear	Deze knop gebruik je om het scherm van de Command Line Interface leeg te maken.

**Opmerking:** Een regel die begint met een # is een commentaarregel. Deze regel wordt genegeerd bij het laten lopen van een opgesteld programma. Het is een regel die extra uitleg geeft, maar zal nooit verschijnen in de CLI.

## 5.1.2 Bewerkingen in Python

Bewerkingen in Python zien er gelijkaardig uit, maar vertonen toch enkele eigenaardigheden.



Oefening:

Ontdek de bewerkingen in Python door onderstaande oefeningen te ontrafelen. Noteer steeds het bewerkingsteken en de bijhorende bewerking. Werk met de gegeven waarden.

Stel  $a = 5$  en  $b = 3$  en  $c = 2$ , bepaal dan:

**Opmerking:** Bij de euclidische deling noteer je de benaming van de uitkomsten.

	Oefening	Bewerkingsteken	Bewerking
a)	$a + b$	+	Som
b)	$a - b$	-	Verschil
c)	$a * b$	*	Vermenigvuldiging
d)	$a / b$	/	Deling
e)	$a ** b$	**	Macht
f)	$a ** (1/b)$	$** (1/x)$	Wortel (derdegraadswortel)
g)	$a ** (1/c)$	$** (1/x)$	Vierkantswortel
Euclidische deling			Uitkomst
h)	$a // b$	//	Quotiënt
i)	$a \% b$	%	Rest

Een euclidische deling is een deling met al dan niet een restwaarde. Het quotiënt wordt in Python ook wel de **Floor division** genoemd. De rest wordt door Python de **Modulo** genoemd.

**Opmerking:** Vergeet niet om rekening te houden met de volgorde van bewerkingen. Python gebruikt de volgorde die jullie altijd geleerd hebben. Plaats de nodige haakjes zodat Python volgens de gewenste volgorde je bewerking(en) kan doorlopen.

- 1) Haakjes
- 2) Machten en wortels
- 3) Delen en vermenigvuldigen van links naar rechts
- 4) Som en verschil van links naar rechts

### Relationele operatoren

Zoals je in de wiskunde gebruik maakt van symbolen om getallen te vergelijken met elkaar, is dit in Python ook mogelijk. De symbolen in Python zijn echter aangepast aan de programmeertaal. Hieronder een opsomming van de symbolen die jullie in de oefeningen van toepassing kunnen zijn.

Symbool in de wiskunde	Symbool in Python
<	<
>	>
≤	<=
≥	>=
=	==
≠	!=





## Tijd voor oefeningen

### Oefening 1:

Geef onderstaande bewerkingen zoals in de opgave staat weergegeven. Beantwoord onderstaande vragen.

	In de CLI!	Python's reactie
a)	12 + 13	25
b)	1 2 + 1 3	Error

Wat valt je op? Hoe komt dit?

Omdat de getallen opgesplitst worden door een spatie geeft dit een error. Python kan deze coderegels niet interpreteren.

### Oefening 2:

Geef onderstaande bewerkingen in en noteer de uitkomsten in de rechterkolom.

	In de CLI!	Python's reactie
a)	1325/36	36.8055555555556
b)	3612 + 2789	6401
c)	58**9	7427658739644928
d)	$\sqrt[5]{1048576}$	16.000000000000004
e)	11 != 11	False
f)	2 <= 100	True
g)	'appendix' > 'beenmerg'	False Verklaring: Aan de letters van het alfabet worden de cijfers 1,2,...,26 toegekend. Zo krijgt a het cijfer 1 en b het cijfer 2. Hierdoor bekom je bij deze vergelijking de reactie 'False'.

**Opmerking:** Bij de 5<sup>de</sup>-machtswortel geeft Python een reactie die een lichte afwijking heeft. Noteer in het kader de volledige reactie van Python, maar weet dat de uitkomst in realiteit het afgeronde resultaat hiervan is.

### Oefening 3:

456 bomen worden verdeeld over verschillende percelen. Op ieder perceel kunnen precies 16 bomen geplant worden.

Hoeveel percelen kunnen volledig aangeplant worden en hoeveel bomen blijven er over? Bereken je resultaat in Python. Noteer ook je tussenstappen en antwoordzin hieronder, .

$456 // 16 = 28$

$456 \% 16 = 8$

Er kunnen 28 percelen volledig aangeplant worden. Er blijven 8 bomen over.

In de verzamelingenleer spreken we over gehele en decimale getallen. Dit is niet het geval in Python. Daar wordt er gesproken over verschillende datatypen. Zo wordt een geheel getal **een integer** genoemd. Daarnaast wordt een decimaal getal **een float** genoemd. Belangrijk hierbij is dat de floats geschreven worden met een punt/~~komma~~ (doorstreep het foute antwoord).

## 5.2 Basisfuncties in Python

*Binnen dit leerstofonderdeel zullen we basisfuncties in Python onderzoeken. Onderstaande functies worden aangereikt:*

- `print()`
- `input()`
- `int()`
- `float()`
- `str()`
- `type()`

*Om deze functies te kunnen gebruiken, wordt er extra uitleg gegeven over een string, een variabele en datatypes. Deze inhouden worden klassikaal aangebracht. Iedere functie zal voorzien worden van een bijhorend oefenvoorbeeld. Dit maken leerlingen ter inoefening van de nieuwe inhouden. Oefeningen maken de leerlingen individueel.*


In wat volgt zal je enkele basisfuncties ontdekken. Deze zullen je rekenwerk vergemakkelijken in grotere opgaven dan  $1 + 1 = 2$ .

Iedere functie is voorzien van een bijhorende video die je later nog kan raadplegen indien nodig.

## 5.2.1 print()

Met de functie `print()`, die je ingeeft in de editor kan je een bepaalde tekst in de CLI laten lopen. Binnen de haakjes noteer je altijd tussen aanhalingstekens de te projecteren tekst. De tekst tussen de aanhalingstekens noemen we een string. Deze kan naast letters ook cijfers, rekenkundige operatoren of spaties bevatten. Het maakt geen verschil of je enkele, dubbele of twee enkele aanhalingstekens gebruikt, je resultaat blijft gelijk. Belangrijk hierbij is dat je begint en afsluit met hetzelfde aanhalingsteken.

Oefenvoorbeeld:



```
main.py
1 print("Wist je dat: Python is naast een programmeertaal ook een soort slang.")
2

Shell
Wist je dat: Python is naast een programmeertaal ook een soort slang.
>
```

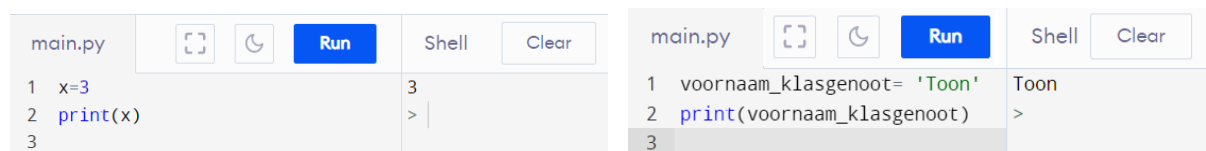
De functie `print()`: <https://youtu.be/7f90IFmsj6o>

## 5.2.2 Variabelen

Variabelen worden binnen Python gebruikt om waarden op te slaan tot ze weer nodig zijn. Een variabele in Python bevat een naam en een waarde. Dit is zeer gelijkend aan de variabele  $x = 3$  in de geschreven wiskunde. In dit geval is 'x' de naam van de variabele en '3' de waarde. Er bestaan geen 2 of meerdere variabelen met dezelfde naam en een verschillende waarde.

De naam van een variabele, het linkergedeelte, kan nooit spaties bevatten. Daarom worden spaties in de naam weergegeven met behulp van een underscore '\_'. (zie voorbeeld)

Oefenvoorbeeld:



```
main.py
1 x=3
2 print(x)
3

3

main.py
1 voornaam_klasgenoot= 'Toon'
2 print(voornaam_klasgenoot)
3

Toon
>
```

### Opmerkingen:

- Let op! We gebruiken '=' bij het opstellen van variabelen. Wanneer we getallen met elkaar willen vergelijken, maken we gebruik van '=='. Dit doen we om ervoor te zorgen dat Python ons toch begrijpt.
- Merk op: In het oefenvoorbeeld zie je dat wanneer je een variabele print, je deze niet tussen aanhalingstekens mag plaatsen!

### Verwonderingsweetje:

Zonder dat we het zelf beseffen, maken we in het dagelijkse leven ook al veel gebruik van variabelen. Denk hierbij maar aan wanneer je een bericht naar iemand verstuurt. Je gaat hierbij niet steeds opnieuw het volledige gsm-nummer ingeven, maar verstuurt dit bericht naar een naam. Deze naam is dan gekoppeld aan zijn waarde: het telefoonnummer.

Wat is een variabele?: <https://youtu.be/gkCpMyIOYm4>

De toekenning van waarden aan variabelen: <https://youtu.be/QsJ3t9gXZIE>


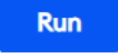

## 5.2.3 input()

Als we in Python een vraag willen stellen waarop we kunnen antwoorden, moeten we gebruik maken van de functie `input()`. Deze functie wordt gebruikt om steeds met veranderlijke waarden verder te werken. Binnen de haakjes wordt een vraag gesteld waarop geantwoord kan worden in de CLI. De functie `input()` gaat er automatisch vanuit dat je antwoord een string is, ook al typ je zelf geen aanhalingstekens.

Je zou misschien denken aan de functie `print()`, maar deze is te beperkt. Deze kan je enkel gebruiken om een waarde op het scherm te tonen. Je kan hiermee wel een vraag stellen, maar antwoorden kan niet.

Om niet elke keer opnieuw de vraag te moeten stellen, koppelen we ons antwoord aan een variabele. In onderstaand voorbeeld wordt de vraag "Wat is je voornaam?" gekoppeld aan de naam "voornaam". De waarde kan nu dus steeds aangepast worden door een ander antwoord op de vraag te geven. Analooog geldt dit ook voor de achternaam. Om uiteindelijk de volledige naam te bekomen, kunnen we beiden strings aan elkaar plakken. Om te voorkomen dat de voor- en achternaam tegen elkaar plakken, voeren we een spatie in. Dit doen we door een string, bestaande uit enkel een spatie, toe te voegen. Vervolgens gebruiken we de functie `print()` om de volledige naam te bekomen.



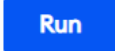
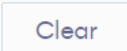
Oefenvoorbeeld:

<code>main.py</code>	  	Shell	
<pre>1 voornaam = input("Wat is je voornaam?") 2 achternaam = input("Wat is je achternaam?") 3 volledige_naam = voornaam + " " + achternaam 4 print(volledige_naam) 5</pre>		<pre>Wat is je voornaam?Alexander Wat is je achternaam?Broux Alexander Broux &gt;</pre>	

De functie `input()`: <https://youtu.be/ss0nw0VByPQ>

## 5.2.4 int() en float()




Bij input() hebben we gezien hoe we meerdere strings aan elkaar kunnen plakken. Omdat Python een string als een reeks symbolen ziet, zullen we vastlopen wanneer we getallen willen optellen. Dit is duidelijk zichtbaar in het voorbeeld hieronder.

main.py	  	Shell	
<pre>1 getal_1=input("Wat is je eerste getal?") 2 getal_2= input("Wat is je tweede getal?") 3 som =getal_1+getal_2 4 print(som) 5</pre>		<pre>Wat is je eerste getal?5 Wat is je tweede getal?8 58 &gt;</pre>	

Om dit probleem op te lossen, gaan we aan Python aangeven dat hij de string moet interpreteren als een geheel getal, ook een integer genoemd. Dit doen we aan de hand van de functie int(). Door de string tussen de haakjes van de functie te plaatsen, werken we niet langer met een reeks symbolen, maar met gehele getallen.

Analoog hieraan gaan we ook werken met de functie float(), waarmee we een string als decimaal getal kunnen voorstellen.

Oefenvoorbeelden:

main.py	  	Shell	
<pre>1 getal_1=input("Wat is je eerste getal?") 2 getal_2 = input("Wat is je tweede getal?") 3 som = int(getal_1)+int(getal_2) 4 print(som) 5</pre>		<pre>Wat is je eerste getal?5 Wat is je tweede getal?8 13 &gt;</pre>	



De functie int(): <https://youtu.be/1sn58AmshII>

De functie float(): <https://youtu.be/wgCC9nfV320>

## 5.2.5 str()

We hebben al gezien hoe we een string kunnen omzetten naar een integer of een float. Soms zal het toch handig zijn om te werken met string-waarden. Dit zullen we doen aan de hand van de functie str(). Hierbij zullen we het integer of de float ingeven tussen de haakjes.

Oefenvoorbeeld:

main.py	  	Shell	
<pre>1 rekeningsaldo = 123 2 zin = "Het bedrag op je rekening bedraagt " + str(rekeningsaldo) + ' euro.' 3 print(zin) 4</pre>		<pre>Het bedrag op je rekening bedraagt 123 euro. &gt;</pre>	

De functie str(): <https://youtu.be/wqvSZ20Pbug>

## 5.2.6 type()

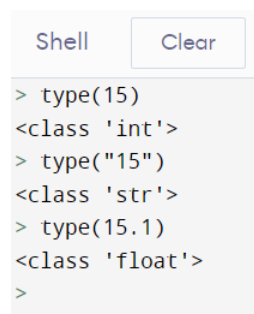
Datatypes brengen structuur en efficiëntie in de programmeercode. Ook Python gebruikt een aantal standaard datatypes zoals integer, float, string,... Het is in eerdere voorbeelden al duidelijk geworden dat "7" + "93" niet gelijk is aan 7 + 93.

Om het datatype van een waarde te kunnen achterhalen maken we gebruik van de functie type(). Zetten we de waarde tussen de haken dan geeft Python aan of het een integer, float, string, ... is.

Verder gaan we ook bekijken hoe een waarde aan zijn datatype komt. Python baseert zich hierbij op een aantal basisafspraken. Wat een string, een integer en een float zijn hebben we eerder al gezien. Verder zijn onderstaande regels nog belangrijk:

- De som, het verschil en het product van 2 integers is een integer.
- De deling van 2 integers zal altijd een float zijn.

Oefenvoorbeeld:



```
Shell Clear
> type(15)
<class 'int'>
> type("15")
<class 'str'>
> type(15.1)
<class 'float'>
>
```

Datatypes en de functie type(): <https://youtu.be/J2YsWdYuoLU>



## Samenvatting

### Functies

`print("tekst")` → Een tekst in de CLI laten lopen.

`input("vraag")` → Een vraag in de CLI laten lopen, waarbij met veranderlijke antwoorden verder gewerkt kan worden.

`int("getal")` → Een getal als geheel getal laten interpreteren i.p.v. als string.

`float("getal")` → Een getal als kommagetal laten interpreteren i.p.v. als string.

`str("getal")` → Een getal als string laten interpreteren i.p.v. als geheel getal of kommagetal.

`type("waarde")` → Het datatype van een waarde achterhalen.

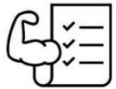
### Begrippen

String = Een reeks van symbolen bestaande uit letters, cijfers, rekenkundige operatoren en/of spaties.

Variabelen = Een naam waar een bepaalde waarde onder kan worden opgeslagen, totdat ze later weer nodig is.

Integer = Een geheel getal

Float = Een kommagetal



### Tijd voor oefeningen

#### Oefening 1:

Je maakt de som van drie getallen. De getallen liggen nog niet vast en kunnen nog gekozen worden. Schrijf een programma, waarmee je Python deze getallen kan laten optellen. Maak gebruik van de functie `input()`. Test het programma ook uit op negatieve getallen.

#### Mogelijke oplossing:

main.py	Shell
<pre>1 getal1 = int(input("Geef een eerste getal op: ")) 2 getal2 = int(input("Geef een tweede getal op: ")) 3 getal3 = int(input("Geef een derde getal op: ")) 4 som = getal1 + getal2 + getal3 5 print(som)</pre>	<pre>Geef een eerste getal op: 5 Geef een tweede getal op: 621 Geef een derde getal op: 24 650 &gt;</pre>





## Oefening 2:

Maak een programma waarmee Python de volgende 3 vragen aan je stelt:

- In welk jaar zitten we?
- Welke maand zijn we?
- Welke dag van de maand zijn we?

Zorg dat Python vervolgens de volledige datum in getallen geeft. Bv.  
18/03/2002




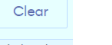
### Mogelijke oplossing:

main.py	  	Shell	
<pre>1 jaar=input("In welk jaar zitten we?") 2 maand=input("Welke maand zijn we?") 3 dag=input("Welke dag van de maand zijn we?") 4 datum=dag+"/"+maand+"/"+jaar 5 print(datum)</pre>		<pre>In welk jaar zitten we?2002 Welke maand zijn we?03 Welke dag van de maand zijn we?18 18/03/2002 &gt;</pre>	

## Oefening 3:

Stel je hebt een willekeurige rechthoekige driehoek. Je kan hierop de lengte van rechthoekszijde 1 en 2 aflezen. Schrijf een programma dat met deze lengtes de schuine zijde kan berekenen.

### Mogelijke oplossing:

main.py	  	Shell	
<pre>1 rechthoekzijde_1=int(input("Wat is de lengte van de eerste rechthoekzijde van je driehoek?")) 2 rechthoekzijde_2=int(input("Wat is de lengte van de tweede rechthoekzijde van je driehoek?")) 3 schuine_zijde=((rechthoekzijde_1**2)+((rechthoekzijde_2**2))**(1/2) 4 print(schuine_zijde) 5</pre>		<pre>Wat is de lengte van de eerste rechthoekzijde van je driehoek?4 Wat is de lengte van de tweede rechthoekzijde van je driehoek?3 5.0 &gt;</pre>	

## Oefening 4:

De programma's hieronder werken niet zoals verwacht. Zoek de fouten, duid ze aan en verbeter ze.

### Programma 1:

main.py	  
<pre>1 print('Baby's hebben geen knieschijven')</pre>	

### Oplossing:

"Baby's hebben geen knieschijven"

→Indien er in de tekst een apostrof gebruikt wordt, moeten we het begin en het einde van de string aangeven met dubbele of twee enkele aanhalingstekens.



### Programma 2:

```
main.py  [ ] [ ] Run
1  soortnaam=input("Wat is de soortnaam van de plant?")
2  geslacht=input("Wat is het geslacht van de plant?")
3  Latijnse naam=geslacht+" "+soortnaam
4  print(Latijnse naam)
```

Oplossing:

Latijnse\_naam

→Er mogen geen spaties staan bij de naam van een variabele.

### Programma 3:

```
main.py  [ ] [ ] Run
1  zijde=input("Wat is de lengte van de zijde van het vierkant?")
2  oppervlakte=zijde*zijde
3  print(oppervlakte)
```

Oplossing:

zijde=int(input("Wat is de lengte van de zijde van het vierkant?"))

→Indien we willen dat Python een string als een geheel getal interpreteert, moeten we gebruik maken van de functie int()

### Programma 4:

```
main.py  [ ] [ ] Run
1  voornaam_klasgenoot= Toon
2  print(voornaam_klasgenoot)
3
```

Oplossing:





'Toon'

→Toon moet tussen aanhalingstekens aangezien dit een string is.

## Oefening 5:

Schrijf een programma dat de oppervlakte van een driehoek berekent. Zorg ervoor dat het programma vraagt naar de basis en de hoogte. Print vervolgens de oppervlakte.

Mogelijke oplossing:

main.py	  	Shell	
<pre>1 Basis = input("Wat is de basis van de driehoek?") 2 Hoogte = input("Wat is de hoogte van de driehoek?") 3 Oppervlakte = 1/2*int(Basis)*int(Hoogte) 4 print(Oppervlakte)</pre>		<pre>Wat is de basis van de driehoek?9 Wat is de hoogte van de driehoek?6 27.0 &gt;</pre>	

## Oefening 6:

Je hebt bij de appelboer een grote doos met appels gekregen en deelt deze vervolgens uit. Alle leerlingen krijgen evenveel appels. Schrijf een programma waarmee je kan bepalen hoeveel appels elke leerling krijgt en hoeveel appels er in de doos blijven liggen. Zorg dat je programma werkt voor eender aantal appels en studenten.




Mogelijke oplossing:

main.py	  	Shell	
<pre>1 aantal_appels=int(input('Hoeveel appels zitten er in totaal in de doos?')) 2 aantal_studenten=int(input('Hoeveel studenten zijn er in totaal?')) 3 appels_per_student= aantal_appels // aantal_studenten 4 appels='Iedere student krijgt '+str(appels_per_student)+' appels.' 5 print(appels) 6 overblijvende_appels=aantal_appels%aantal_studenten 7 appels_in_de_dooos='In totaal blijven er '+str(overblijvende_appels)+' over.' 8 print(appels_in_de_dooos) 9</pre>		<pre>Hoeveel appels zitten er in totaal in de doos?150 Hoeveel studenten zijn er in totaal?20 Iedere student krijgt 7 appels. In totaal blijven er 10 over. &gt;</pre>	

## Oefening 7:

Hieronder zijn een aantal programma's geschreven. Kan jij achterhalen wat de programma's doen?

Programma 1:

main.py	  
<pre>1 aantal_kinderen=int(input("Hoeveel kinderen gaan er naar het pretpark?")) 2 aantal_volwassenen=int(input("Hoeveel volwassenen gaan er naar het pretpark?")) 3 4 prijs=aantal_kinderen*30+aantal_volwassenen*45 5 6 print("Je moet "+str(prijs)+" euro betalen.")</pre>	

Oplossing:

Het programma vraagt aan de gebruiker met hoeveel kinderen en volwassenen men naar het pretpark gaat. Als reactie laat Python zien hoeveel men uiteindelijk moet betalen. Voor elk kind wordt er 30 euro betaald en voor elke volwassene 45 euro.

## Programma 2:

```
main.py     
1 aantal_boeken=int(input("Hoeveel boeken koopt de winkel aan?"))  
2  
3 prijs_boeken=aantal_boeken*15  
4 prijs_vervoer=1*3+(aantal_boeken-1)*0.75  
5  
6 totaal_prijs=prijs_boeken+prijs_vervoer  
7  
8 print("De winkel moet in totaal "+str(totaal_prijs)+" betalen.")  
9
```

### Oplossing:

Het programma vraagt aan de gebruiker hoeveel boeken een winkel aankoopt. Als reactie krijgt men te zien hoeveel de winkel moet betalen. Hierbij wordt rekening gehouden met de prijs van de boeken (15 euro/boek), maar ook met de prijs van het vervoeren van de boeken. Dit is 3 euro voor het eerste boek en 75 cent voor elk extra boek.

## 5.3 ALS/DAN-functies

*Binnen dit leerstofonderdeel zullen we de ALS/DAN-functies in Python onderzoeken. Onderstaande statements worden aangereikt:*

- *if-statement of enkelvoudige keuzestructuur*
- *if/else-statement of tweevoudige keuzestructuur*
- *if/elif/else-statement of meervoudige keuzestructuur*

*Om deze te kunnen gebruiken wordt het begrip 'statement' uitgelegd. Verder wordt er ook ingegaan op de 3 booleaanse operatoren: 'or', 'and' en 'not'. Ten slotte wordt er kort ingegaan op de backslash en een aantal built-in-functies. Deze inhouden worden volledig zelfstandig doorlopen.*

### 5.3.1 Booleaanse expressie

Een booleaanse expressie is een expressie waarbij de waarde evolueert naar 'waar' (True) of 'onwaar/vals' (False). In het Nederlands wordt deze uitspraak ook een bewering genoemd.

True en False hebben het datatype boolean. Dit is een nieuw datatype naast de string, integer en float. True en False dienen altijd met hoofdletter genoteerd te worden.

Beweringen laten toe een specifiek gevolg te laten afhangen van het al dan niet waar zijn van een bepaalde bewering. ALS de bewering waar is DAN een specifiek gevolg.

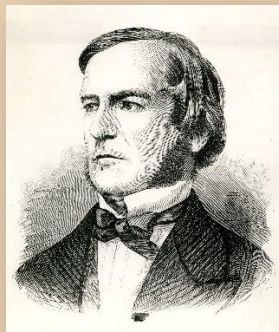
Voorbeeld: ALS mijn lichaamstemperatuur stijgt DAN begin ik te zweten.

Booleaanse expressie

#### Weetje:

De naam van de booleaanse expressie is afgeleid van de Engelse wiskundige en logicus George Boole. Hij is de grondlegger van de computerwetenschappen. Daar zijn werk zo belangrijk en invloedrijk was, is de naam van dit datatype een eerbetoon aan hem.

George Boole, geboren in 1815 te Lincoln, heeft zijn interesse voor de wiskunde meegekregen vanuit zijn vader. Vanaf zijn 16<sup>de</sup> levensjaar begon George in zijn thuisstad te werken als leerkracht om zijn familie te kunnen onderhouden. 3 jaar later, op 19 jarige leeftijd richtte hij er een school op. Boole's uitzonderlijke wiskundige talent bleef niet onopgemerkt en in 1849 werd hij aangesteld als hoogleraar wiskunde aan het Queen's College in Cork. George was iemand die vooral op zoek was naar de waarheid en helemaal niet naar erkenning. Hierdoor heeft hij niet steeds de waardering gekregen die hij verdiende voor zijn ontdekkingen binnen de computerwetenschappen en de logica. Wel kreeg hij een onderscheiding van de Royal society en een eredoctoraat van de universiteit van Dublin. In 1855 trouwde hij met Mary Everest. Samen kregen ze 5 dochters. Op 49 jarige leeftijd werd een koortsaanval, die eindigde in longfalen, hem fataal.



*Figuur 20: George Boole (1814-1864)*

Booleaanse expressie: [https://youtu.be/lohZ5T\\_xo7E](https://youtu.be/lohZ5T_xo7E)

## 5.3.2 if-statement of enkelvoudige keuzestructuur




Het if-statement komt overeen met de ALS/DAN-bewering. Een statement is een stukje programmeercode dat een, door de computer uit te voeren, taak beschrijft. Bij het if-statement beschrijft men welke taken er uitgevoerd moeten worden, wanneer de booleaanse expressie evolueert naar de waarde **True**.

Vaak wordt er gebruik gemaakt van geneste if-statements. Deze benaming wordt gebruikt voor een if-statement die ingebed is in een andere if-statement. Hieronder een voorbeeld van geneste if-statements.

```
4 ▾ if temperatuur >=20:  
5 ▾     if bewolking == 'nee':
```

Oefenvoorbeeld:

Zoals je in het oefenvoorbeeld ziet, vervangen we ALS door 'if' en DAN door ':'. Verder worden de specifieke taken weergegeven met een aantal inspringingen. Dit laat zien dat ze bij de booleaanse expressie horen. Deze inspringingen zal Python automatisch invoeren als.

main.py	  	Shell	
<pre>1 lichaamstemperatuur = float(input("Wat is je lichaamstemperatuur?")) 2 ▾ if lichaamstemperatuur &gt; 38: 3     print("Je hebt koorts! Probeer je lichaam af te koelen.") 4     print("Blijf je temperatuur regelmatig controleren!") 5</pre>		<pre>Wat is je lichaamstemperatuur?39.1 Je hebt koorts! Probeer je lichaam af te koelen. Blijf je temperatuur regelmatig controleren! &gt;</pre>	

In codetaal:

if << booleaanse expressie >> :

<< taken >>



**Opmerking:** Let op! De functie if wordt net zoals andere functies genoteerd met een kleine letter.

Het if-statement: [https://youtu.be/ID\\_cYtxyyZA](https://youtu.be/ID_cYtxyyZA)

### 5.3.3 if/else-statement of tweevoudige keuzestructuur

Met het if-statement doet Python niets wanneer de booleaanse expressie evolueert naar de waarde False. Toch zullen we dit vaak wel nodig hebben. Hiervoor voegen we het if/else-statement in. De 'else' zullen we dus gebruiken om aan te geven welke actie Python moet uitvoeren wanneer de bewering niet klopt.

Oefenvoorbeeld:

main.py		Shell	
<pre>1 lichaamstemperatuur = float(input("Wat is je lichaamstemperatuur?")) 2 if lichaamstemperatuur &gt; 38: 3     print("Je hebt koorts! Probeer je lichaam af te koelen.") 4 else: 5     print("Je temperatuur is normaal. Maak er een leuke dag van!") 6</pre>		<pre>Wat is je lichaamstemperatuur?36.7 Je temperatuur is normaal. Maak er een leuke dag van! &gt;</pre>	

In codetaal:

if << booleaanse expressie >> :

<< taken >> } True

else:

<< alternatieve taken >> } False

Het if/else-statement: <https://youtu.be/35ff-d8HtZE>

## 5.3.4 if/elif/else-statement of meervoudige keuzestructuur

Indien er meerdere if/else-statements staan geprogrammeerd, wordt de code in Python zeer lang. Om dit korter en overzichtelijker te maken, wordt het if/elif/else-statement ingevoerd. Wanneer er een 'if' volgt op een 'else', kunnen we deze samen nemen in een 'elif'. In het oefenvoorbeeld zie je duidelijk hoe je dit kan doen.

Oefenvoorbeeld:

```
main.py [Run] Shell [Clear]

1 lichaamstemperatuur = float(input("Wat is je lichaamstemperatuur?"))
2 if lichaamstemperatuur > 38:
3     print("Je hebt koorts! Probeer je lichaam af te koelen.")
4 else:
5     if lichaamstemperatuur < 35:
6         print("Je bent onderkoeld! Warm je lichaam op.")
7     else:
8         print("Je temperatuur is normaal. Maak er een leuke dag van!")
9
10 lichaamstemperatuur = float(input("Wat is je lichaamstemperatuur?"))
11 if lichaamstemperatuur > 38:
12     print("Je hebt koorts! Probeer je lichaam af te koelen.")
13 elif lichaamstemperatuur < 35:
14     print("Je bent onderkoeld! Warm je lichaam op.")
15 else:
16     print("Je temperatuur is normaal. Maak er een leuke dag van!")
17
```

Wat is je lichaamstemperatuur?37.5  
Je temperatuur is normaal. Maak er een leuke dag van!

Wat is je lichaamstemperatuur?37.5  
Je temperatuur is normaal. Maak er een leuke dag van!  
>

Het if/elif/else-statement: <https://youtu.be/qb7P3EWcbW4>



## Samenvatting

### Begrippen

Booleaanse expressie = Een expressie waarbij de waarde evolueert naar 'waar' (True) of 'onwaar/vals' (False).

Statement = Een stukje programmeercode dat een, door de computer uit te voeren, taak beschrijft.

### Statements

if-statement → Een statement dat beschrijft welke taken er uitgevoerd moeten worden, wanneer een specifieke booleaanse expressie evolueert naar de waarde 'True'.

if/else-statement → Een statement dat beschrijft welke taken er uitgevoerd moeten worden, wanneer een specifieke booleaanse expressie evolueert naar de waarde 'True' en welke taken er uitgevoerd moeten worden, wanneer de expressie evolueert naar de waarde 'False'.

if/elif/else-statement → Een statement dat bij meerdere if/else-statements gebruikt wordt. Wanneer een 'if' volgt op een 'else' wordt dit samengenomen in een 'elif'.



## Tijd voor oefeningen

### Oefening 1:

Schrijf een programma dat laat weten of het ingevoerde geheel getal een positief getal, negatief getal of het getal nul is. Hierbij moet er steeds gewerkt kunnen worden met een veranderlijk geheel getal.

Test het programma uit op een positief getal, negatief getal en nul.

### Mogelijke oplossing:

```
main.py  [Icons] Run Shell Clear
1 g = int(input('Wat is het geheel getal?'))
2 if g > 0:
3     print('Het getal is positief.')
4 if g < 0:
5     print('Het getal is negatief.')
6 if g == 0:
7     print('Het getal is gelijk aan 0.')
8
```

Wat is het geheel getal?-9  
Het getal is negatief.  
>



## Oefening 2:

Schrijf een programma dat vraagt naar een getal en vervolgens de persoon laat weten of het opgegeven getal even of oneven is. Laat hierbij ook weten dat het programma volledig doorlopen is.

Test het programma uit op zowel een even als een oneven getal.

Mogelijke oplossing:

main.py	Run	Shell
<pre>1 Getal = int(input("Wat is je getal? ")) 2 3 if Getal % 2 == 0: 4     print("Het getal is EVEN.") 5 else: 6     print("Het is ONEVEN.") 7 print ("Het programma is doorlopen.")</pre>		<pre>Wat is je getal? 36 Het getal is EVEN. Het programma is doorlopen. &gt;</pre>

## Oefening 3:

Schrijf een programma voor het berekenen van de discriminant alsook het aantal nulpunten van een tweedegraadsvergelijking. Het if/elif/else-statement moet zeker in de programmeercode aanwezig zijn.

Mogelijke oplossing:

main.py	Run	Shell
<pre>1 a = int(input("Wat is de waarde die bij x² staat?")) 2 b = int(input("Wat is de waarde die bij x staat?")) 3 c = int(input("Wat is de constante waarde?")) 4 Discriminant = int(b**2 - 4*a*c) 5 print (Discriminant) 6 if Discriminant &lt; 0: 7     print ("Er zijn geen nulpunten.") 8 elif Discriminant == 0: 9     print ("Er is een dubbel nulpunt.") 10 else: 11     print ("Er zijn 2 nulpunten.")</pre>		<pre>Wat is de waarde die bij x² staat?2 Wat is de waarde die bij x staat?2 Wat is de constante waarde?3 -20 Er zijn geen nulpunten. &gt;</pre>

Wanneer de determinatietabel eindigt, laat je programma dan printen dat de boom niet gekend is.



main.py	Run	Shell	Clear
<pre>1 naalden=input("Heeft de boom naalden?") 2- if naalden=="ja": 3     per_twee=input("Staan de naalden 2 aan 2 bij elkaar?") 4-     if per_twee=="ja": 5         aantal_cm=input("Zijn de naalden meer dan 9 cm lang?") 6         if aantal_cm=="ja": 7             print("Zwarte den") 8         else: 9             print("Grove den") 10-     else: 11         print("De boom is niet genkend.") 12- else: 13     samengesteld=input("Is het blad samengesteld?") 14-     if samengesteld=="ja": 15         aantal_bladeren=input("Bestaat het blad uit 5 of 7 in één punt             samenkomen de blaadjes?") 16-         if aantal_bladeren=="ja": 17             print("Paardenkastanje") 18-         else: 19             print("De boom is niet gekend.") 20-     else: 21         stand=input("Staan de bladeren 2 aan 2 tegenover elkaar?") 22-         if stand=="ja": 23             print("Esdoorn") 24-         else: 25             print("De boom is niet gekend. ") 26</pre>		<pre>Heeft de boom naalden?nee Is het blad samengesteld?nee Staan de bladeren 2 aan 2 tegenover elkaar?nee De boom is niet gekend. &gt;</pre>	

## Oefening 5:

Je leert zweefvliegen. De instructeur acht je klaar voor je eerste solovlucht. Je moet aan een aantal voorwaarden voldoen om alleen te mogen vertrekken met een zweefvliegtuig.

- Je moet minstens 14 jaar oud zijn.
- Je moet minstens 1,5 m groot zijn.
- Je moet minstens 60 kg wegen.

Schrijf een programma dat 'Fijne vlucht!' op het scherm toont indien aan alle voorwaarden tegelijk voldaan is.

Mogelijke oplossing:

main.py	Run	Shell	Clear
<pre>1 leeftijd = int(input("Wat is je leeftijd?")) 2 lengte = float(input("Wat is je lengte?")) 3 gewicht = float(input("Wat is je gewicht?")) 4 5 if leeftijd &gt;= 14: 6     if lengte &gt;= 150: 7         if gewicht &gt;= 60: 8             print("Fijne vlucht gewenst!") 9</pre>		<pre>Wat is je leeftijd?15 Wat is je lengte?162 Wat is je gewicht?70 Fijne vlucht gewenst! &gt;</pre>	





## Oefening 6:

Schrijf een if/elif/else-keten die na het vragen van iemands geboortjaar aangeeft in welke levensfase de persoon zich bevindt.

- Baby: 0-2jaar
- Peuter: 3-4 jaar
- Kind:5-13 jaar
- Tiener:14-19 jaar
- Volwassene: 20-64
- Oudere: 65+

In deze oefening moet je geen rekening houden met de maand waarin een persoon geboren is. Zo zal je bijvoorbeeld al een volwassene zijn in het jaar dat je 20 wordt. Zorg er bovendien voor dat je programma werkt in eerder welk jaar. Test je programma op minstens 3 verschillende leeftijden.

### Mogelijke oplossing:

main.py	  	Shell	
<pre>1 geboortjaar=int(input("In welk jaar ben jij geboren?")) 2 3 leeftijd=2023-geboortjaar 4 if leeftijd&lt;2: 5     print("Je bent een baby.") 6 elif leeftijd&lt;5: 7     print("Je bent een peuter.") 8 elif leeftijd&lt;14: 9     print("Je bent een kind.") 10 elif leeftijd&lt;20: 11     print("Je bent een tiener.") 12 elif leeftijd&lt;65: 13     print("Je bent een volwassene.") 14 else: 15     print("Je bent een oudere.")</pre>		<pre>In welk jaar ben jij geboren?2007 Je bent een tiener. &gt;</pre>	

## Oefening 7:

Hieronder is een programma geschreven. Kan jij achterhalen wat het programma doet?

main.py	  
<pre>1 getal=int(input("Met hoeveel personen kom je dineren?")) 2 3 if getal&gt;8: 4     print("Onze excuses, u moet nog even wachten op een tafel.") 5 else: 6     print("Uw tafel is gereed, volgt u maar.")</pre>	

### Oplossing:

Het programma vraagt de gebruiker met hoeveel personen ze komen dineren. Als het antwoord meer dan 8 is, zegt het programma dat ze moeten wachten op een tafel. Anders wordt er gezegd dat hun tafel gereed is.



## Oefening 8:

Een Belgisch bankrekeningnummer bv. BE05 2100 2335 9975 eindigt op 2 controlecijfers. Die twee laatste cijfers gaan na of de voorgaande 10 cijfers correct zijn. Het controlegetal is namelijk de rest van de deling van 2100 2335 99 door 97.

2100 2335 99/97= 21 651 892 met rest 75

Schrijf een programma waarbij je in 2 aparte variabelen, het deeltal en het controlegetal, opvraagt. Laat je programma controleren of het opgegeven bankrekeningnummer echt bestaat.

### Mogelijke oplossing:

main.py			Run	Shell	
<pre>1 deeltal=int(input("Wat is het deeltal van je bankrekeningnummer?")) 2 controlegetal= int(input("Wat is het controlegetal van je bankrekeningnummer?")) 3 4- if deeltal%97==controlegetal: 5     print("Je bankrekeningnummer bestaat.") 6- else: 7     print("Je bankrekeningnummer bestaat niet, controleer je gegevens opnieuw.")</pre>				<pre>Wat is het deeltal van je bankrekeningnummer?2100233599 Wat is het controlegetal van je bankrekeningnummer?75 Je bankrekeningnummer bestaat. &gt;</pre>	

## Oefening 9:

Overgewicht kan enkele gevolgen met zich meebrengen zoals: verhoogde kans op diabetes type 2, ... De BMI (= Body Mass Index) is een methode om te controleren hoe het gesteld is met ons lichaam. Voor jongeren is een BMI tussen 16 en 24 ideaal. Als iemand zijn BMI lager is, spreekt men van ondergewicht. Is hij hoger? Dan spreekt men van overgewicht.

Ontwerp een programma dat iemand zijn BMI berekent en meld of die persoon ondergewicht, een gezond gewicht of overgewicht heeft. Controleer je programma ten slotte voor Piet. Hij weegt 55 kg en is 1,60 m groot.

Formule:  $BMI = \frac{\text{massa}}{\text{lengte}^2}$

### Mogelijke oplossing:

main.py			Run	Shell	
<pre>1 massa= float(input("Wat is het gewicht van de persoon in kg?")) 2 lengte= float(input("Wat is de lengte van de persoon in meter?")) 3 4 BMI=massa/(lengte**2) 5 print ("De persoon zijn BMI is "+str(BMI)) 6- if BMI&lt;16: 7     print("Hij/zij heeft ondergewicht.") 8- elif BMI&gt;24: 9     print("Hij/zij heeft overgewicht.") 10- else: 11     print ("Het gewicht van de persoon is goed.")</pre>				<pre>Wat is het gewicht van de persoon in kg?55 Wat is de lengte van de persoon in meter?1.60 De persoon zijn BMI is 21.484374999999996 Het gewicht van de persoon is goed. &gt;</pre>	

## Oefening 10:

Vind jij de fout in deze programma's? Duid ze aan en verbeter indien mogelijk.

### Programma 1:

```
main.py     
1 leeftijd = int(input("Hoe oud ben je?"))  
2 if leeftijd < 30:  
3     print("Je bent in de jonge jaren. ")  
4 elif leeftijd < 20:  
5     print("Je bent een tiener!")  
6 elif leeftijd < 10:  
7     print("Je bent een kind!")  
8 else:  
9     print("Je jonge jaren zijn voorbij.")  
10
```

### Oplossing:

Wanneer je een leeftijd hebt die kleiner is dan 30 jaar, dan wordt de tekst "Je bent in je jonge jaren." geprint. Dit resultaat bekom je ook bij een leeftijd van bijvoorbeeld 16. Je wilt bij deze leeftijd echter een ander resultaat krijgen. Dit doordat de volgorde van de booleaanse expressies foutief is gekozen. Mogelijke oplossingen voor het programma zouden zijn om het domein beter af te bakenen of om de volgorde van het programma aan te passen.

### Programma 2:

```
main.py     
1 aantal_vissen=int(input("Hoeveel vissen wil je kopen?"))  
2  
3 if aantal_vissen<5:  
4     prijs=aantal_vissen*15  
5 else:  
6     prijs=aantal_vissen*13  
7 elif aantal_vissen>15:  
8     prijs=aantal_vissen*10  
9  
10 print("Je betaalt "+prijs+" euro voor "+aantal_vissen+" vissen.")
```

### Oplossing:

De 'elif' en de 'else' moeten van plaats gewisseld worden.

## 5.3.5 Booleaanse operatoren

In wat volgt leggen we de verschillende booleaanse operatoren uit. Wanneer er verschillende if-statements zijn, kunnen we deze op meerdere manieren samen nemen in één if-statement. We bespreken 'or', 'not' en 'and'. We zien hierin ook tal van linken met de logica in de wiskunde.

## **Operator 'or'**

Met de operator 'or' kunnen we twee booleaanse expressies samen nemen onder 1 if-statement. Hierbij moet minstens 1 van de twee booleaanse expressies het resultaat True opleveren om als uiteindelijke waarde True te bekomen.

Opdracht:

Vervolledig onderstaande tabel met 'True' of 'False'.

	True	False
True	True	True
False	True	False

Oefenvoorbeeld:

```
main.py  Run  Shell  Clear
1 neus = input("Door welk neusgat adem jij? LINKS of RECHTS? ")
2* if neus == 'LINKS' or neus == 'RECHTS':
3     print('Je neus zit niet verstopt.')
4
```

```
Door welk neusgat adem jij? LINKS of RECHTS? LINKS
Je neus zit niet verstopt.
>
```

## **Operator 'and'**

Bij de operator 'and' moeten beide booleaanse expressies het resultaat True opleveren om als eindresultaat True te bekomen.

Opdracht:

Vervolledig onderstaande tabel met 'True' of 'False'.

	True	False
True	True	False
False	False	False

Oefenvoorbeeld:

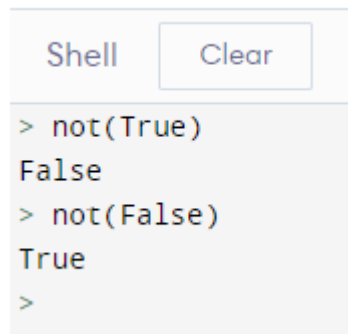
```
main.py  Run  Shell  Clear
1 celkern = input("Heeft de cel een celkern?")
2 bladgroenkorrels = input("Bevat de cel bladgroenkorrels?")
3* if celkern == "ja" and bladgroenkorrels == "ja":
4     print("Het is een plantaardige cel.")
5
```

```
Heeft de cel een celkern?ja
Bevat de cel bladgroenkorrels?ja
Het is een plantaardige cel.
>
```

## Operator 'not'

De operator 'not' komt in de logica overeen met het symbool '¬' ook winkelhaak genoemd. Beide hebben als betekenis 'niet'. Wanneer we deze operator voor een booleaanse expressie plaatsen, verkrijgen we het tegengestelde resultaat.

Oefenvoorbeeld:



```
> not(True)
False
> not(False)
True
>
```

Booleaanse operatoren: <https://youtu.be/sV3JWgWhCM8>

'or': 0:00 – 7:33

'and': 7:33-15:19

'not': 15:19-16:25

## 5.3.6 Combinaties met backslash

Binnen het programmeren lopen we soms tegen het probleem aan, dat Python symbolen op een andere manier interpreteert dan oorspronkelijk bedoeld is. Bij een aanhalingsteken gaat Python steeds op zoek naar een string. Dit zal niet altijd nodig zijn. Om dit aan te geven wordt er in de code gebruik gemaakt van een backslash, die op dat moment de betekenis van het teken erachter wijzigt. Echter zullen we een combinatie met een backslash zelden zelf gebruiken. Dit zal Python soms in zijn reactie weergeven zolang je de functie print() niet gebruikt.

In de editor	Geprinte waarde
\'	'
\"	"
\\	\
\n	Verspringen naar nieuwe regel

De backslash: [https://youtu.be/emkJb3\\_cE7k](https://youtu.be/emkJb3_cE7k)



## 5.3.7 Built-in-functies

Er zijn enkele functies die standaard ingebouwd zijn in Python. Deze worden built-in-functies genoemd. Voorbeelden hiervan zijn:

- `min()` → minimum bv. `min(1,2,3) = 1`
- `max()` → maximum bv. `max("aap", "dolfijn", "zebra") = zebra`
- `abs()` → absolute waarde
- `len()` → lengte woorden en getallenreeksen  
bv. `len("elektronenmicroscoop") = 20`

Deze functies kunnen zowel in de editor als in de CLI ingegeven worden. Wanneer deze in de editor ingegeven worden, moet de functie echter wel voorzien worden van de functie `print()`. In de CLI is dit niet nodig.

Built-in functies: <https://youtu.be/nMh86o-hXmw>



### Samenvatting

#### Booleaanse operatoren

Operator 'or' → Twee booleaanse expressies samennemen onder één if-statement waarbij minstens één van de twee als resultaat True oplevert om als eindresultaat True te bekomen.

Operator 'and' → Twee booleaanse expressies samennemen onder één if-statement waarbij beiden als resultaat True opleveren om als eindresultaat True te bekomen.

Operator 'not' → Het tegengestelde van het resultaat van een booleaanse expressie bekomen.

#### Extra

Backslash → De betekenis van het teken erachter wijzigen.

Built-in-functies → Functies die standaard ingebouwd zijn in Python. Alle functies die tot nu toe al besproken zijn, zijn built-in-functies.



## Tijd voor oefeningen

### Oefening 1:

Net zoals de bewerkingen een bepaalde volgorde hebben, kent Python ook aan de booleaanse operatoren een bepaalde volgorde toe. Onderzoek in onderstaande tabel welke operatoren voorrang krijgen op elkaar.

	Python's reactie
True or True and False	True
not(True) or False	False
True and False or True	True

Geef hieronder de hiërarchie van de operatoren.

1. not
2. and
3. or

**Opmerking:** De haakjes hebben nog steeds een absolute voorrang op de operatoren.

### Oefening 2:

Schrijf een programma waarin Python je kan vertellen of een gegeven jaar een schrikkeljaar is of niet.

Gegeven: Je hebt een schrikkeljaar als het jaartal deelbaar is door 4 en niet deelbaar is door 100 of als je jaartal deelbaar is door 4, 100 en 400.

Test je programma uit op de volgende jaartallen:

1300	Schrikkeljaar
2016	Schrikkeljaar
1800	Geen schrikkeljaar

### Mogelijke oplossing:

```
main.py  Run  Shell  Clear
1 Jaar = int(input("Welk jaartal wil je nakijken op basis van schrikkeljaar of niet?"))
2 if Jaar%4 ==0 and (Jaar%100!=0 or (Jaar%100==0 and Jaar%400==0)) :
3     print("SCHRIKKELJAAR")
4 else:
5     print("GEEN SCHRIKKELJAAR")
6
```

```
Welk jaartal wil je nakijken op basis van schrikkeljaar of niet?1300
GEEN SCHRIKKELJAAR
>
```

### Oefening 3:

We spelen het spel 'Bingo'. We gaan ervan uit dat het trekken van het nummer 12, 53 of 72 'Bingo!' oplevert. Schrijf een programma dat 'Bingo!' print als één van de bovenstaande voorwaarden voldaan is. Bij een ander gegeven geheel getal levert het programma 'Jammer, geen bingo.'

#### Mogelijke oplossing:

```
main.py  [ ] [ ] Run Shell Clear
1  getal = int(input("Wat is je getrokken getal?"))
2
3  if getal == 12 or getal == 53 or getal == 72:
4      print("Bingo!")
5  else:
6      print("Jammer, geen bingo")
7
```

Wat is je getrokken getal?53  
Bingo!  
>

### Oefening 4:

In onderstaande code is heel wat misgelopen. Zoek de fouten en duid ze aan.  
Tip: in totaal staan er 8 fouten in.

```
main.py  [ ] [ ] Run
1  x=str(input("Wat is de waarde van x?"))
2  y=str(input("Wat is de waarde van y?"))
3
4  if x==3 and y==4
5      print("x is 3")
6      print("y is 4")
7  if x>2 end y<5:
8  print ("x>2")
9  print ("y<5")
10 if x<4 and y>3:
11     print ("x<4")
12     print("y>3")
```

## Oplossing:

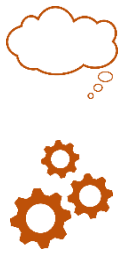
```
main.py
1 x=str(input("Wat is de waarde van x?"))
2 y=str(input("Wat is de waarde van y?"))
3
4 if x==3 and y==4:
5     print("x is 3")
6     print("y is 4")
7 * if x>2 end y<5:
8 print ("x>2")
9 print ("y<5")
10 * if x<4 and y>3:
11     print ("x<4")
12 |     print("y>3")
```

- In regel 1 en 2 moet `str()` aangepast worden naar `int()` of `float()`
- Op het einde van regel 4 moet een `:`
- De `print()` in regel 5,8,9 en 12 zijn verkeerd uitgelijnd. Ze moeten op hetzelfde niveau komen als regel 6 en 11.
- In regel 7 moet `'end'` vervangen worden door `'and'`.

## Oefening 5:

Geef het einderesultaat van onderstaande opgaven. Je mag hiervoor **geen** computer gebruiken!

	Resultaat (True/False)
<code>not(True and not(False))</code>	False
<code>not(True or False)</code>	False
<code>not(True) or not(False)</code>	True
<code>False or False and True or not(False) and not(True)</code>	False
<code>True or not(True) and False or not(False)</code>	True



## 5.4 Lussen in Python

*In wat volgt zal het programmeren van lussen aan bod komen. Hierin wordt de while-lus als eerst besproken door middel van uitgeschreven theorie. Het is aan te raden om deze theorie klassikaal aan te brengen. Bijkomend wordt de functie range() en het keyword 'in' aangehaald.*

*De for-lus wordt aangeleerd aan de hand van een verplichte video.*

Om in Python herhalingen of iteraties uit te voeren, zijn vaste functies ontwikkeld. Deze worden ook lussen of loops genoemd. Door het gebruik van lussen kan je repetitief werk laten uitvoeren.

Zo kan je verschillende if-statements met bijhorende takenlijsten in 1 lus bundelen. Hierdoor zal Python het if-statement een bepaald of onbepaald aantal keer overlopen afhankelijk van de aanwezigheid van een eindwaarde.


We gaan verder in deze bundel gebruik maken van lussen om ervoor te zorgen dat we bij bijvoorbeeld een onderzoek van 600 proefpersonen niet 600 if-statements voor iedere persoon moeten invullen. We zullen dus een lus gebruiken die Python 600 keer kan doorlopen.

### 5.4.1 While-lus

De while-lus is één van de twee manieren om herhaling toe te passen. Elke herhaling kan geprogrammeerd worden met behulp van een while-lus. Bij deze herhaling kan op voorhand al vastliggen hoe vaak de takenlijst opnieuw uitgevoerd moet worden of wordt het eindpunt vastgesteld met een voorwaarde (zie voorbeeld). Dit doen we zodat het programma niet oneindig blijft doorlopen.

In codetaal:

```
while << booleaanse expressie >>:
    << taak 1 >>
    << taak 2 >>
    << ... >>
```




Hierin herken je zeker en vast de codetaal van het if-statement.

Python controleert eerst de booleaanse expressie en voert de taken enkel uit als deze evolueert naar de waarde True. In tegenstelling tot het if-statement keert Python na het uitvoeren van de takenlijst terug naar de expressie en voert hij opnieuw de controle uit. Als deze nog steeds evolueert naar de waarde True, wordt de takenlijst opnieuw doorlopen. De herhalingen stoppen vanaf dat de expressie naar de waarde False evolueert. Als dit het geval is, negeert Python de takenlijst en gaat hij verder naar de regels die onder de while-lus staan. Om te voorkomen dat het programma oneindig blijft doorlopen, is het belangrijk dat er een taak aanwezig is die ervoor zorgt dat de variabele(n) niet voldoet/voldoen aan de voorwaarde. Dit zodat de booleaanse expressie evolueert naar de waarde False.





Voorbeeld:

Wanneer je de getallen van 1 tot 6 wil afdrukken, kan je dit uiteraard doen met de functie print(). We kunnen dit echter veel korter noteren. Stel dat je de getallen van 1 tot 6000 wil noteren, dan moet je deze code 6000 keer uittypen en ben je uren bezig. Dat zullen we vermijden door het invoeren van een while-lus.

main.py	Run	Shell	Clear
1 # De manier waarop we herhalingen schrijven zonder lussen.		1	
2 getal_1=1		2	
3 if getal_1<=6:		3	
4     print(getal_1)		4	
5 getal_2=2		5	
6 if getal_2 <=6:		6	
7     print(getal_2)			
8 getal_3 = 3			
9 if getal_3<=6:			
10     print(getal_3)			
11 getal_4=4			
12 if getal_4<=6:			
13     print(getal_4)			
14 getal_5=5			
15 if getal_5<=6:			
16     print(getal_5)			
17 getal_6=6			
18 if getal_6<=6:			
19     print(getal_6)			
20 getal_7=7			
21 if getal_7<=6:			
22     print(getal_7)		1	
23 # De manier waarbij er wel gebruik wordt gemaakt van de while-lus		2	
24 getal = 1		3	
25 while getal <=6:		4	
26     print(getal)		5	
27     getal = getal +1		6	

voorwaarde

Oefenvoorbeeld:

main.py	  	Shell	
<pre>1 # Met dit programma kan je een aantal positieve gehele getallen optellen. 2 # Vanaf dat er een negatief geheel getal wordt gegeven, zal de lus eindigen. 3 getal = 0 4 som = 0 5 while getal &gt;=0: 6     som = som + getal 7     getal = int(input("Geef een een geheel getal: ")) 8 print(som) 9</pre>		<pre>Geef een een geheel getal: 5 Geef een een geheel getal: 6 Geef een een geheel getal: -2 11 &gt;</pre>	





De while-lus: <https://youtu.be/o7iJ6asIY1Q>

## 5.4.2 Keyword 'in'

Keywords in Python zijn woorden met een bijzondere betekenis. Zo zal het woord 'in' ons toelaten om te controleren of een specifiek element voorkomt in een bepaalde groep elementen. Deze groep van elementen wordt ook een collectie of verzameling genoemd. Wanneer de volgorde van de elementen in een verzameling van belang is, spreken we over een geordende collectie. Iedere string is hier een voorbeeld van, daar de tekens in de string een bepaalde volgorde aannemen.

Oefenvoorbeeld:

Volgend voorbeeld wordt ingegeven in de editor. Je kan dit ook ingeven in de CLI maar hier is de functie print() niet noodzakelijk. Met dit voorbeeld wordt duidelijk dat de string 'x' deel uitmaakt van de string 'desoxyribonucleïnezuur'. We noemen dit dan ook een substring.

main.py	  	Shell	
<pre>1 print('x' in 'desoxyribonucleïnezuur') 2 print('a' in 'desoxyribonucleïnezuur') 3 print('xy' in 'desoxyribonucleïnezuur') 4 print('yx' in 'desoxyribonucleïnezuur') 5</pre>		<pre>True False True False &gt;</pre>	

In onderstaande tabel zie je een overzicht van alle keywords in Python. Zo zie je dat we al enkele keywords hebben besproken.

Zo zullen de gearceerde keywords alvast een belletje doen rinkelen.

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Keyword 'in': <https://youtu.be/yzGZkiY0g6M>

### 5.4.3 For-lus en de functie range()

De for-lus en de functie range() zal je leren begrijpen met behulp van een video. Bekijk dus onderstaande video. Hierin gebruiken ze echter een andere programmeeromgeving. Geen paniek, deze is zeer gelijkend op de omgeving waar wij gebruik van maken.



<https://youtu.be/pzjs6XWvrP0>

Voorbeeld:

```
main.py  [Icons]  Run  Shell  Clear
1 woord = 'mutualisme'
2 print("De for-lus start:")
3- for letter in woord:
4     print(letter)
5 print("Klaar met de for-lus!")
6

De for-lus start:
m
u
t
u
a
l
i
s
m
e
Klaar met de for-lus!
>
```



Opdracht: kijk de video en maak onderstaande opgave

In de video zag je het voorbeeld waarbij het grootste getal gezocht werd in een collectie van 6 positieve gehele getallen.

Schrijf nu zelf een programma waarin je het kleinste negatieve gehele getal zoekt in een collectie van 6 getallen.

Mogelijke oplossing:

main.py	Shell
<pre>1 kleinste = 0 2 3- for beurt in range(6): 4     getal = int(input("Geef een negatief, geheel getal op? ")) 5-     if getal &lt; kleinste: 6         kleinste = getal 7 8 print("Het kleinste getal uit deze 6 getallen is:", kleinste) 9</pre>	<pre>Geef een negatief, geheel getal op? -9 Geef een negatief, geheel getal op? -5 Geef een negatief, geheel getal op? -8 Geef een negatief, geheel getal op? -6 Geef een negatief, geheel getal op? -10 Geef een negatief, geheel getal op? -17 Het kleinste getal uit deze 6 getallen is: -17 &gt;</pre>



## Samenvatting

### Lussen

De while-lus → Lus die het mogelijk maakt om een takenlijst te herhalen voor een op voorhand bepaald aantal herhalingen of voor een aantal herhalingen dat afhankelijk is van een voorwaarde.

De for-lus → Lus die het mogelijk maakt om een takenlijst te herhalen voor een op voorhand bepaald aantal herhalingen.

### Functie

range("getal") → Gebruik maken van een bepaald bereik.

### Extra

Keyword 'in' → Controleren of een specifiek element voorkomt in een bepaalde groep elementen.



## Tijd voor oefeningen

### Oefening 1:

Schrijf een programma dat aftelt. Er wordt begonnen met een specifiek nummer, bijvoorbeeld 10. De code telt dan af van tien naar nul, waarbij ieder nummer geprint wordt (10, 9, 8, ...). Zorg dat het programma werkt voor eender welke beginwaarde.

Nul wordt niet geprint, maar in plaats daarvan drukt het programma "Start!" af.

Test het programma voor twee verschillende getallen.

#### Mogelijke oplossing:

main.py	Shell
<pre>1 getal = int(input("Van welk getal wil je beginnen af te tellen? ")) 2 while getal &gt;= 1: 3     print(getal) 4     getal = getal - 1 5 print("Start!")</pre>	<pre>Van welk getal wil je beginnen af te tellen? 5 5 4 3 2 1 Start! &gt;</pre>

### Oefening 2:

Schrijf een programma dat de gebruiker een getal laat ingeven. Het programma geeft vervolgens de tafel van vermenigvuldiging van het getal voor de factoren 1 tot en met 10. Bijvoorbeeld, als de gebruiker 12 ingeeft, dan is de eerste regel die afgedrukt wordt "1 \* 12 = 12" en de laatste regel "10 \* 12 = 120".

#### Mogelijke oplossing:

main.py	Shell
<pre>1 getal = int(input("Geef een nummer: ")) 2 factor = 1 3 while factor &lt;= 10: 4     print(factor*getal) 5     factor = factor+1 6 print("Klaar")</pre>	<pre>Geef een nummer: 12 12 24 36 48 60 72 84 96 108 120 Klaar &gt;</pre>

### Oefening 3:

Schrijf een programma waarin je de faculteit van een positief natuurlijk getal kan berekenen.

Herhaling: de faculteit van het getal  $n$  wordt weergegeven door

$$n! = 1 \cdot 2 \cdot \dots \cdot n$$

#### Mogelijke oplossing:

main.py	Shell
<pre>1 n = int(input("Geef een positief natuurlijk getal waarvan je de faculteit wil berekenen:")) 2 getal = 1 3 product = 1 4 for faculteit in range(n+1): 5     if getal &lt;= n: 6         product = product * getal 7         getal = getal + 1 8 print(product) 9</pre>	<pre>Geef een positief natuurlijk getal waarvan je de faculteit wil berekenen:5 120 &gt;</pre>

### Oefening 4:

Schrijf een programma dat vraagt om twee woorden. Druk vervolgens alle letters af die de woorden gemeen hebben. Je mag hoofletters beschouwen als verschillend van kleine letters, maar iedere letter die je rapporteert, mag slechts één keer gerapporteerd worden (bijvoorbeeld: de strings "een" en "peer" hebben slechts één letter gemeen, namelijk de letter "e").

Tip: Sla de letters die de woorden gemeen hebben op in een derde variabelen, en als je een letter vindt die beide woorden gemeen hebben, test je of de letter al in de derde string staat alvorens je hem rapporteert.

#### Mogelijke oplossing:

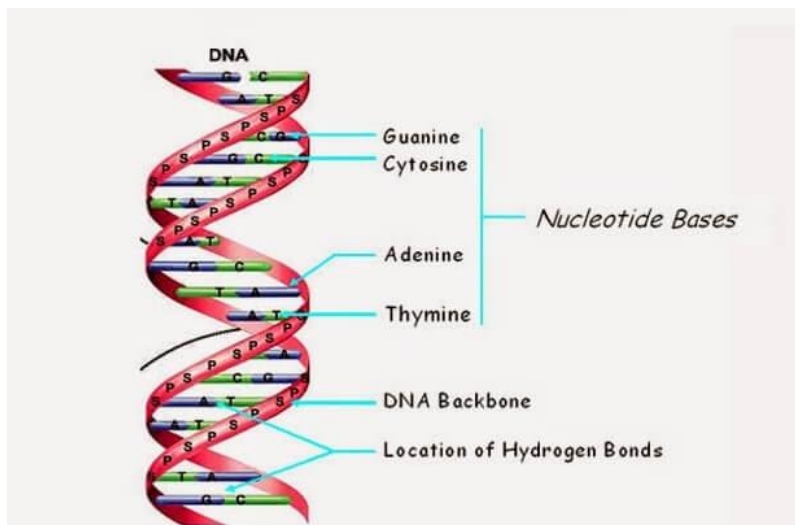
main.py	Shell
<pre>1 woord1 = str(input( "Geef woord 1: " )) 2 woord2 = str(input( "Geef woord 2: " )) 3 gemeen = "" 4 for letter in woord1: 5     if (letter in woord2) and (letter not in gemeen): 6         gemeen = gemeen + letter 7 if gemeen == "": 8     print( "De woorden hebben geen tekens gemeen." ) 9 else: 10    print( "De woorden delen de volgende tekens:", gemeen ) 11</pre>	<pre>Geef woord 1: Bo Geef woord 2: Juul De woorden hebben geen tekens gemeen. &gt; Geef woord 1: Toon Geef woord 2: Jonah De woorden delen de volgende tekens: on</pre>

## Oefening 5:

DNA bestaat uit een helixstructuur die opgebouwd is uit twee strengen. Deze zijn complementair aan elkaar. Schrijf een programma waarbij de gebruiker steeds 1 nucleotide per keer ingeeft. De complementaire streng wordt steeds groter. De omzetting is als volgt:

- G → C
- C → G
- T → A
- A → T

Indien andere letters of meer dan één letter worden ingegeven, stopt de gebruiker met letters ingeven en wordt het resultaat getoond.



## Mogelijke oplossing:

```
main.py  [Icons] [Run] Shell [Clear]

1 streng_1=input("Wat is de eerste nucleotide uit jouw DNA-streng?")
2 streng_2=""
3 while streng_1=="G" or streng_1=="U" or streng_1=="T" or streng_1=="A":
4     if streng_1=="G":
5         streng_2=streng_2+"U"
6     elif streng_1=="U":
7         streng_2=streng_2+"G"
8     elif streng_1=="T":
9         streng_2=streng_2+"A"
10    elif streng_1=="A":
11        streng_2=streng_2+"T"
12    streng_1=input("Wat is de volgende nucleotide uit jouw DNA-streng?")
13
14 print("De complementaire DNA-streng is " + streng_2 + ".")
15
```

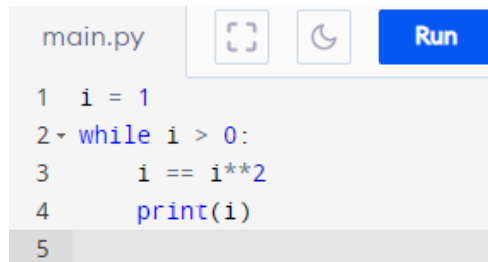
Wat is de eerste nucleotide uit jouw DNA-streng?A  
Wat is de volgende nucleotide uit jouw DNA-streng?T  
Wat is de volgende nucleotide uit jouw DNA-streng?A  
Wat is de volgende nucleotide uit jouw DNA-streng?G  
Wat is de volgende nucleotide uit jouw DNA-streng?U  
Wat is de volgende nucleotide uit jouw DNA-streng?r  
De complementaire DNA-streng is TATUG.  
> |

## Oefening 6:

Vind jij de fout in deze programma's. Duid ze aan. Je mag hiervoor **geen** gebruik maken van een computer.

### Programma 1:

Ik wil alle natuurlijke getallen tot de 2<sup>de</sup> macht verheffen. Als ik onderstaande code gebruik zal ik echter niet tot mijn gewenste resultaat komen. Wat is er mis met onderstaande code?



```
main.py
1 i = 1
2 while i > 0:
3     i == i**2
4     print(i)
5
```

### Oplossing:

Als bovenstaande code wordt doorlopen zullen niet alle natuurlijke getallen tot de tweede macht verheven worden. Dit doordat  $i$  de waarde 1 zal blijven aanhouden aangezien  $1^2$  altijd 1 blijft. De lus zal dus oneindig blijven doorlopen met steeds als resultaat 1.

## Programma 2:

Zoek de 5 fouten in dit voorbeeld.

```
main.py  [Icons] [Run]

1  # de tussentijdse som
2  Som = 0
3  # hoeveel getallen er al ingegeven zijn
4  n = 0
5  while n < 5:
6      x = int(input("Welk getal wil je ingeven?"))
7      som = som
8      Print('De tussentijdse som is , som)
9      n = n + 1
10  print('De totaalsom is', som)
11
```

## Oplossing:

```
main.py  [Icons] [Run]

1  # de tussentijdse som
2  Som = 0
3  # hoeveel getallen er al ingegeven zijn
4  n = 0
5  while n < 5:
6      x = int(input("Welk getal wil je ingeven?"))
7      som = som + x
8      Print('De tussentijdse som is , som)
9      n = n + 1
10  print('De totaalsom is', som)
11
```

- 'Som' staat met hoofdletter geschreven. In de lus wordt deze met een kleine letter genoteerd.
- De haakjes op regel 6 worden niet gesloten.
- 'print' wordt op regel 8 met hoofdletter geschreven. Een functie moet steeds met kleine letter genoteerd worden!
- Op regel 8 wordt een aanhalingsteken vergeten.
- Op regel 7 mist + x

## Oefening 7:

Schrijf een programma dat de gebruiker om een getal vraagt en zegt of het een veelvoud van 7 is of niet. Indien het getal geen veelvoud van 7 is, geeft het programma het eerstvolgende getal dat wel een veelvoud van 7 is.

Test het programma op zowel een veelvoud van 7 als geen veelvoud van 7.

### Mogelijke oplossing:

```
main.py  Run Shell Clear
1 getal=int(input("Geef een willekeurig getal."))
2
3 if getal%7==0:
4     print("Het getal is een veelvoud van 7.")
5 else:
6     print("Het getal is geen veelvoud van 7")
7 while getal%7!=0:
8     getal=getal+1
9     print("Het eerstvolgende veelvoud van 7 is" + " " + str(getal))
```

Geef een willekeurig getal.18  
Het getal is geen veelvoud van 7  
Het eerstvolgende veelvoud van 7 is 21  
>



## 5.5 Overkoepelende oefeningen

### Oefening 1:

Programmeer de rij van Fibonacci in Python met de functies die wij gezien hebben. Het is de bedoeling dat er gevraagd wordt naar welk getal in de rij men wil weten en dat Python hier op reageert met het juiste getal.

### Mogelijke oplossing:

```
main.py  Run Shell Clear
1 n=int(input("Het hoeveelste getal uit de rij wil je weten?"))
2
3 n0=0
4 n1=1
5
6 if n==0:
7     getal=0
8 elif n==1:
9     getal=1
10 else:
11     generatie=2
12 while generatie<=n:
13     generatie=generatie+1
14     getal=n0+n1
15     n0=n1
16     n1=getal
17 print(getal)
```

Het hoeveelste getal uit de rij wil je weten?13  
233  
>

### Extra uitleg bij de oplossing:

We kijken even terug naar de rij van Fibonacci om de oplossing beter te begrijpen.

n	0	1	2	3	4	5	6	7	8
getal	0	1	1	2	3	5	8	13	21

Om te beginnen moet er eerst gevraagd worden naar het hoeveelste getal uit de rij van Fibonacci ze willen weten. We zullen hier de variabelen  $n$  aan koppelen.

Een getal in de rij wordt steeds gevormd door de twee voorgaande getallen bij elkaar op te tellen. Hierbij moeten de eerste twee getallen gegeven worden. Dit doen we in ons programma door te zeggen dat  $n_0$  gelijk is aan 0 en  $n_1$  aan 1.

We beginnen met het deel van het programma waar we Python zullen laten rekenen. Omdat een getal steeds bestaat uit de som van 2 voorgaande getallen lopen we vast wanneer we het getal voor  $n=0$  of het getal voor  $n=1$  willen berekenen. Hierbij hebben we geen twee voorgaande getallen die we kunnen optellen. Om dit op te lossen maken we gebruik van het if/elif/else-statement.

Eerst zeggen we aan de hand van 'if' dat het getal gelijk is aan 0 wanneer  $n$  gelijk is aan 0. Dan maken we gebruik van 'elif' om te zeggen dat het getal gelijk is aan 1 bij  $n=1$ .

Nu we dit probleem opgelost hebben, gaan we over naar het berekenen van het getal. Hierbij zullen we gebruik maken van de while-lus. Deze moet blijven lopen tot we bij de generatie komen die we willen bekomen. Oftewel willen we getallen blijven optellen zolang  $\text{generatie} \leq n$ . De variabele 'generatie' hebben we gekozen omdat we in ons achterhoofd het voorbeeld van de konijnen houden. De leerlingen mogen hier natuurlijk een andere naam aan geven. Omdat we eerder de getallen voor  $n=0$  en  $n=1$  al geprogrammeerd hebben, zullen we hier starten bij de tweede generatie.

Elke keer dat de lus doorlopen wordt, zullen we het getal van de volgende generatie berekenen. De generatie zal dus steeds met 1 vermeerderen. Het getal gaan we dan berekenen door de vorige twee getallen op te tellen. Ons getal zal dus gelijk zijn aan  $n_0 + n_1$ . En elke keer dat onze lus doorlopen wordt, zullen we een andere waarde aan  $n_0$  en  $n_1$  toekennen. Dit doen we door  $n_0$  gelijk te stellen aan  $n_1$  en  $n_1$  gelijk te stellen aan  $n_2$ . In onderstaande tabel is een duidelijk overzicht van hoe dit gebeurt voor generatie 2 tot en met generatie 8.

n	2	3	4	5	6	7	8
n0	0	1	1	2	3	5	8
n1	1	1	2	3	5	8	13
getal	1	2	3	5	8	13	21



## Oefening 2 :

Darren zijn mannelijke bijen die geboren worden uit een onbevucht eitje. Ze zijn een maatje groter dan de werkbijen, maar een maatje kleiner dan de koningin. Hun ogen zijn twee keer zo groot als de ogen van de werkbijen en de koningin, maar in tegenstelling tot de werkbijen kunnen darren niet steken.

Darren halen geen honing, geen stuifmeel, voeren geen larven, bouwen geen raten en laten zich het liefst van al voeren door de werkbijen. Hun bijdrage aan de kolonie is om te helpen bij de temperatuurregeling. Wanneer de temperatuur te laag wordt, gaan de darren en de werkbijen warmte genereren door te trillen. Wanneer de temperatuur te hoog wordt, gaan de darren net als de werkbijen met de vleugels wapperen ter ventilatie. Daarnaast bestaat de enige andere taak van de darren erin om te paren met de koningin.

Bij het aantal bijen in een generatie kunnen we een interessant patroon terugvinden:

- Het aantal darren in een bepaalde generatie is gelijk aan het aantal vrouwelijke bijen in de voorgaande generatie.
- Het aantal vrouwelijke bijen in een bepaalde generatie is gelijk aan het aantal vrouwelijke bijen in de voorgaande twee generaties.

In onderstaande tabel zie je een overzicht van de eerste 7 generaties.

Generatie	0	1	2	3	4	5	6	7
Aantal vrouwelijke bijen	0	1	1	2	3	5	8	13
Aantal mannelijke bijen	1	0	1	1	2	3	5	8
Totaal aantal bijen	1	1	2	3	5	8	13	21

Maak nu een programma dat je vraagt naar een generatie en je dan het aantal mannelijke, het aantal vrouwelijke en het totaal aantal bijen van die generatie geeft. Hieronder zie je een voorbeeld waar we Python gevraagd hebben om het aantal bijen van de 7<sup>de</sup> generatie te geven.

```
Shell Clear  
Van welke generatie wil je het aantal bijen weten?7  
Aantal vrouwen:13  
Aantal mannen:8  
Aantal bijen:21
```

Tip: Als je naar de tabel kijkt, kan je hier de rij van Fibonacci in herkennen. Het programma dat je in de vorige oefening maakte, zal je hier dus nodig hebben.

## Mogelijke oplossing:

main.py	Shell
<pre>1 n=int(input("Van welke generatie wil je het aantal bijen weten?")) 2 3 n0=1 4 n1=0 5 n2=1 6 7 if n==0: 8     aantal_mannen=1 9     aantal_vrouwen=0 10 elif n==1: 11     aantal_mannen=0 12     aantal_vrouwen=1 13 else: 14     generatie=2 15     while generatie&lt;=n: 16         generatie=generatie+1 17         aantal_mannen=n0+n1 18         aantal_vrouwen=n1+n2 19         n0=n1 20         n1=aantal_mannen 21         n2=aantal_vrouwen 22 23 aantal_bijen=aantal_vrouwen+aantal_mannen 24 25 print("Aantal vrouwen:"+str(aantal_vrouwen)) 26 print("Aantal mannen:"+str(aantal_mannen)) 27 print("Aantal bijen:"+str(aantal_bijen))</pre>	<pre>Van welke generatie wil je het aantal bijen weten?13 Aantal vrouwen:233 Aantal mannen:144 Aantal bijen:377 &gt;</pre>

### Oefening 3:

Krekels tjirpen door hun vleugels langs elkaar te strijken. Bij de meeste soorten zijn het enkel de mannetjes die het bekende geluid voortbrengen om zo vrouwelijke partners te kunnen aantrekken. Het idee dat het tellen van de tjirpgeluiden, die krekels voortbrengen, ook kan dienen als een informele manier om de temperatuur te bepalen is echter niet nieuw. Het werd oorspronkelijk beschreven in 1897 door de natuurkundige Amos Dolbear, in een artikel met als titel "De krekels als thermometer". Daarin stelde Dolbear aanvankelijk dat de buitentemperatuur een belangrijke bepalende factor is voor de frequentie waarmee krekels tjirpen. Doorheen de jaren werd zijn manier om naar dit verband te kijken echter omgekeerd, mensen tellen nu het aantal tjirpgeluiden om daarmee de temperatuur te bepalen. De wet van Dolbear geeft het verband aan onder de vorm van volgende formule, die aangeeft hoe de temperatuur in graden Fahrenheit ( $T_F$ ) kan geschat worden op basis van het aantal gehoorde tjirpen per minuut ( $N_{60}$ ):





$$T_F = 50 + \left( \frac{N_{60} - 40}{4} \right)$$

Deze formule kan ook herschreven worden om de temperatuur in graden Celsius ( $T_C$ ) te bepalen:

$$T_C = 10 + \left( \frac{N_{60} - 40}{7} \right)$$

Schrijf nu een programma waar de gebruiker gevraagd wordt naar het aantal tjirpen per minuut. Python reageert met de temperatuur in graden Fahrenheit en in graden Celsius.

#### Mogelijke oplossing:

main.py	  	Shell	
<pre>1 tjirps=int(input("Hoeveel keer per minuut hoor je de krekels tjirpen?")) 2 3 Temperatuur_F=50+((tjirps-40)/4) 4 print('Temperatuur (Fahrenheit): '+str(Temperatuur_F)) 5 6 Temperatuur_C=10+((tjirps-40)/7) 7 print('Temperatuur (Celsius): '+str(Temperatuur_C))</pre>		<pre>Hoeveel keer per minuut hoor je de krekels tjirpen?120 Temperatuur (Fahrenheit): 70.0 Temperatuur (Celsius): 21.42857142857143 &gt;</pre>	

## Oefening 4:

Als je op dit moment op de maan zou staan, zou je gewicht slechts 16,5% zijn van het gewicht dat je op de aarde hebt. Je kunt dit uitrekenen door jouw aardse gewicht te vermenigvuldigen met 0,165.

Stel dat je elk jaar een kilo zwaarder zou worden op aarde gedurende de komende 15 jaren. Wat zou je dan wegen als je elk jaar een bezoekje aan de maan zou brengen? Wat zou dan je gewicht zijn na 15 jaar? Schrijf een programma waarin je een lus gebruikt om voor elk jaar jouw maangewicht te printen.

### Mogelijke oplossing:

main.py	Shell
<pre>1 aarde_gewicht=float(input("Wat is jouw gewicht op de aarde in kg?")) 2 maan_gewicht=aarde_gewicht*0.165 3 print("Jouw gewicht op de maan op dit moment is " + str(maan_gewicht)+"kg") 4 jaar=0 5 for x in range(15): 6     jaar=jaar+1 7     aarde_gewicht=aarde_gewicht+1 8     maan_gewicht=aarde_gewicht*0.165 9     print("Jouw gewicht op de maan na "+str(jaar)+ " jaar is " + str(maan_gewicht) 10         )+ "kg")</pre>	<pre>Wat is jouw gewicht op de aarde in kg?55 Jouw gewicht op de maan op dit moment is 9.075000000000001kg Jouw gewicht op de maan na 1 jaar is 9.24kg Jouw gewicht op de maan na 2 jaar is 9.405000000000001kg Jouw gewicht op de maan na 3 jaar is 9.57kg Jouw gewicht op de maan na 4 jaar is 9.735000000000001kg Jouw gewicht op de maan na 5 jaar is 9.9kg Jouw gewicht op de maan na 6 jaar is 10.065000000000001kg Jouw gewicht op de maan na 7 jaar is 10.23kg Jouw gewicht op de maan na 8 jaar is 10.395000000000001kg Jouw gewicht op de maan na 9 jaar is 10.56kg Jouw gewicht op de maan na 10 jaar is 10.725kg Jouw gewicht op de maan na 11 jaar is 10.89kg Jouw gewicht op de maan na 12 jaar is 11.055kg Jouw gewicht op de maan na 13 jaar is 11.22kg Jouw gewicht op de maan na 14 jaar is 11.385kg Jouw gewicht op de maan na 15 jaar is 11.55kg</pre>

## Oefening 5:

Bedenk een mogelijke opgave voor dit programma. Als je een goede opgave gevonden hebt, geef dan de variabelen een meer passende naam.

main.py	Shell
<pre>1 a = 3.14 2 b = 7 3 c = a*b*b 4 print(c) 5</pre>	<pre>153.86 &gt;</pre>

### Mogelijke oplossing:

Opgave = Bereken de oppervlakte van een cirkel met straal 7.

Aanpassing variabelen; a = pi, b = straal en c = oppervlakte.

main.py	Shell
<pre>1 pi = 3.14 2 straal = 7 3 oppervlakte = pi*straal*straal 4 print(oppervlakte) 5</pre>	<pre>153.86 &gt;</pre>

## Oefening 6:

Schrijf een programma dat de som berekent van de positieve getallen die de gebruiker ingeeft. Hij/zij kan kiezen hoeveel getallen er ingegeven worden. Wanneer de gebruiker een negatief getal of nul intypt, wordt de totaalsom geprint.

### Mogelijke oplossing:

main.py	Run	Shell	Clear
<pre>1 som = 0 2 getal = int(input('Geef een getal: ')) 3 while (getal &gt; 0) : 4     som = som + getal 5     getal = int(input('Geef een getal: ')) 6 else: 7     if getal &lt;= 0: 8         print(som) 9</pre>		<pre>Geef een getal: 5 Geef een getal: 8 Geef een getal: 9 Geef een getal: 14 Geef een getal: -3 36 &gt;</pre>	

## Oefening 7:

Ga opzoek naar het patroon in deze code. Schrijf vervolgens hetzelfde programma maar gebruikmakend van een lus.

main.py	Run	Shell	Clear
<pre>1 print('Ik ben geboren in 2007') 2 print('Het huidige jaar is 2023') 3 print('Dit jaar word/ben ik 16 jaar') 4 print('In 2008 was ik 1 jaar') 5 print('In 2009 was ik 2 jaar') 6 print('In 2010 was ik 3 jaar') 7 print('In 2011 was ik 4 jaar') 8 print('In 2012 was ik 5 jaar') 9 print('In 2013 was ik 6 jaar') 10 print('In 2014 was ik 7 jaar') 11 print('In 2015 was ik 8 jaar') 12 print('In 2016 was ik 9 jaar') 13 print('In 2017 was ik 10 jaar') 14 print('In 2018 was ik 11 jaar') 15 print('In 2019 was ik 12 jaar') 16 print('In 2020 was ik 13 jaar') 17 print('In 2021 was ik 14 jaar') 18 print('In 2022 was ik 15 jaar') 19</pre>		<pre>Ik ben geboren in 2007 Het huidige jaar is 2023 Dit jaar word/ben ik 16 jaar In 2008 was ik 1 jaar In 2009 was ik 2 jaar In 2010 was ik 3 jaar In 2011 was ik 4 jaar In 2012 was ik 5 jaar In 2013 was ik 6 jaar In 2014 was ik 7 jaar In 2015 was ik 8 jaar In 2016 was ik 9 jaar In 2017 was ik 10 jaar In 2018 was ik 11 jaar In 2019 was ik 12 jaar In 2020 was ik 13 jaar In 2021 was ik 14 jaar In 2022 was ik 15 jaar &gt;</pre>	

### Mogelijke oplossing:

main.py	Run	Shell	Clear
<pre>1 geboortjaar= int(input('Ik ben geboren in ')) 2 huidige_jaar = int(input('Het huidige jaar is ')) 3 leeftijd = 0 4 leeftijd_nu = huidige_jaar - geboortjaar 5 6 print('Dit jaar word/ben ik',leeftijd_nu,'jaar') 7 while geboortjaar&lt;(huidige_jaar-1): 8     geboortjaar = geboortjaar + 1 9     leeftijd = leeftijd + 1 10    print('In', geboortjaar, 'was ik ',leeftijd,' jaar') 11 12 13 14</pre>		<pre>Ik ben geboren in 2007 Het huidige jaar is 2023 Dit jaar word/ben ik 16 jaar In 2008 was ik 1 jaar In 2009 was ik 2 jaar In 2010 was ik 3 jaar In 2011 was ik 4 jaar In 2012 was ik 5 jaar In 2013 was ik 6 jaar In 2014 was ik 7 jaar In 2015 was ik 8 jaar In 2016 was ik 9 jaar In 2017 was ik 10 jaar In 2018 was ik 11 jaar In 2019 was ik 12 jaar In 2020 was ik 13 jaar In 2021 was ik 14 jaar In 2022 was ik 15 jaar &gt;</pre>	

## Oefening 8:

Bedenk een mogelijke opgave voor dit programma.

```
main.py ⌕ 🌙 Run  
1 age = int(input("Wat is je leeftijd? "))  
2 film = input("Welke film wil je gaan kijken? ")  
3  
4 if age < 3:  
5     print(" Je mag gratis binnen!")  
6 elif age < 13:  
7     print(" Je ticket kost €10.")  
8 else:  
9     print(" Je ticket kost €15.")  
10
```

## Oplossing:

Schrijf een programma dat je laat weten hoeveel je moet betalen in de cinema. Ben je jonger dan 3 jaar, mag je gratis binnen. Ben je tussen 3 en 13, betaal je 10 euro en als je ouder bent, betaal je 15 euro. Laat het programma ook vragen naar welke film je wil kijken.

## Oefening 9:

Schrijf een programma dat een getal van een gebruiker aanneemt en de som van alle getallen van 1 tot een gegeven getal berekent. Stel het getal 6 wordt ingegeven, dan zal de uitkomst 21 zijn. ( $1+2+3+4+5+6=21$ )

## Mogelijke oplossing:

main.py <span>⌕</span> <span>🌙</span> <span>Run</span>	Shell <span>Clear</span>
<pre>1 som = 0 2 gevraagd_nummer = int(input("Wat is je nummer? ")) 3 4 for getal in range(1, gevraagd_nummer + 1): 5     som = som + getal 6 print("De som is: ", som) 7</pre>	<pre>Wat is je nummer? 8 De som is: 36 &gt;</pre>

### Oefening 10:

Bepaal de topvergelijking van een parabool met top T(-4,5) en punt A(-3,2). Zorg ervoor dat je eender welke coördinaten kan invullen.

#### Mogelijke oplossing:

Topvergelijking:  $y = a(x-p)^2 + q$  met T(p,q) en A(x<sub>1</sub>, y<sub>1</sub>)

Coëfficiënt  $a = \frac{q-y_1}{(x_1-p)^2}$  met A(x<sub>1</sub>,y<sub>1</sub>)

main.py	  	Shell	
<pre>1 x_coördinaat_top = int(input('Wat is je x-coördinaat van de top? ')) 2 y_coördinaat_top = int(input('Wat is je y-coördinaat van de top? ')) 3 x_coördinaat_punt = int(input('Wat is je x-coördinaat van het gegeven punt? ')) 4 y_coördinaat_punt = int(input('Wat is je y-coördinaat van het gegeven punt? ')) 5 coëfficiënt = int((y_coördinaat_top - y_coördinaat_punt) / ((x_coördinaat_punt - x_coördinaat_top)**2)) 6 print('y=', coëfficiënt, '(x-(', x_coördinaat_top, '))**2+', y_coördinaat_top) 7</pre>		<pre>Wat is je x-coördinaat van de top? -4 Wat is je y-coördinaat van de top? 5 Wat is je x-coördinaat van het gegeven punt? -3 Wat is je y-coördinaat van het gegeven punt? 2 y= 3 *(x-( -4 ))**2+ 5 &gt;</pre>	

### Oefening 11:

Schrijf een programma dat de gebruiker naar 2 getallen vraagt. Vervolgens laat je Python de getallen vergelijken. Het programma zal dus aangeven of getal 1 groter dan, gelijk aan of kleiner dan getal 2 is.

#### Mogelijke oplossing:

main.py	  	Shell	
<pre>1 nummer1 = int(input('Wat is je eerste nummer? ')) 2 nummer2 = int(input('Wat is je tweede nummer? ')) 3 4 if nummer1 &gt;= nummer2: 5     if nummer1 == nummer2: 6         print(nummer1, 'en', nummer2, 'zijn gelijk') 7     else: 8         print(nummer1, 'is groter dan', nummer2) 9 else: 10    print(nummer1, 'is kleiner dan', nummer2) 11</pre>		<pre>Wat is je eerste nummer? 9 Wat is je tweede nummer? 6 9 is groter dan 6 &gt;</pre>	

## Oefening 12:

Anaïs en Jonas spelen bijna elk weekend een voetbalmatch. Om te kijken wie beter is, vergelijken ze het gemiddeld aantal keer dat ze gescoord hebben tot nu toe. Kan jij een programma schrijven waarin ze elk weekend hun scores ingeven, dat dan zegt wie het beste gespeeld heeft?

### Mogelijke oplossing:

main.py	Run	Shell	Clear
<pre>1 Score_Anaïs=int(input("Hoe vaak scoorde Anaïs tijdens de 1ste wedstrijd?")) 2 punten_Anaïs=0 3 Gemiddelde_Anaïs=Score_Anaïs 4 5 wedstrijd=1 6 7 while punten_Anaïs&lt;=10: 8     wedstrijd=wedstrijd+1 9     punten_Anaïs=int(input("Hoe vaak scoorde Anaïs tijdens de "+str(wedstrijd)+"de     wedstrijd?")) 10    Score_Anaïs=Score_Anaïs+punten_Anaïs 11    Gemiddelde_Anaïs=Score_Anaïs/wedstrijd 12 13 Score_Jonas=int(input("Hoe vaak scoorde Jonas tijdens de 1ste wedstrijd?")) 14 punten_Jonas=0 15 Gemiddelde_Jonas=Score_Jonas 16 17 wedstrijd=1 18 while punten_Jonas&lt;=10: 19     wedstrijd=wedstrijd+1 20     punten_Jonas=int(input("Hoe vaak scoorde Jonas tijdens de "+str(wedstrijd)+"de     wedstrijd?")) 21    Score_Jonas=Score_Jonas+punten_Jonas 22    Gemiddelde_Jonas=Score_Jonas/wedstrijd 23 24 if Gemiddelde_Anaïs==Gemiddelde_Jonas: 25     print("Jullie scoren gemiddeld even vaak en zijn dus beiden even goed.") 26 elif Gemiddelde_Anaïs&lt;Gemiddelde_Jonas: 27     print("Proficiat Jonas, jij scoort gemiddeld vaker dan Anaïs.") 28 else: 29     print("Proficiat Anaïs, jij scoort gemiddeld vaker dan Jonas.") 30</pre>		<pre>Hoe vaak scoorde Anaïs tijdens de 1ste wedstrijd?2 Hoe vaak scoorde Anaïs tijdens de 2de wedstrijd?5 Hoe vaak scoorde Anaïs tijdens de 3de wedstrijd?8 Hoe vaak scoorde Anaïs tijdens de 4de wedstrijd?4 Hoe vaak scoorde Anaïs tijdens de 5de wedstrijd?2 Hoe vaak scoorde Jonas tijdens de 1ste wedstrijd?1 Hoe vaak scoorde Jonas tijdens de 2de wedstrijd?5 Hoe vaak scoorde Jonas tijdens de 3de wedstrijd?6 Hoe vaak scoorde Jonas tijdens de 4de wedstrijd?8 Hoe vaak scoorde Jonas tijdens de 5de wedstrijd?9 Hoe vaak scoorde Jonas tijdens de 6de wedstrijd?1 Proficiat Jonas, jij scoort gemiddeld vaker dan Anaïs. &gt;  </pre>	

## Oefening 13:

Een bioloog wil inloggen op zijn favoriete website [www.natuurpunt.be](http://www.natuurpunt.be). Zijn wachtwoord is: IkHouVan3Panda's. Schrijf een programma dat weergeeft of het wachtwoord al dan niet juist is.

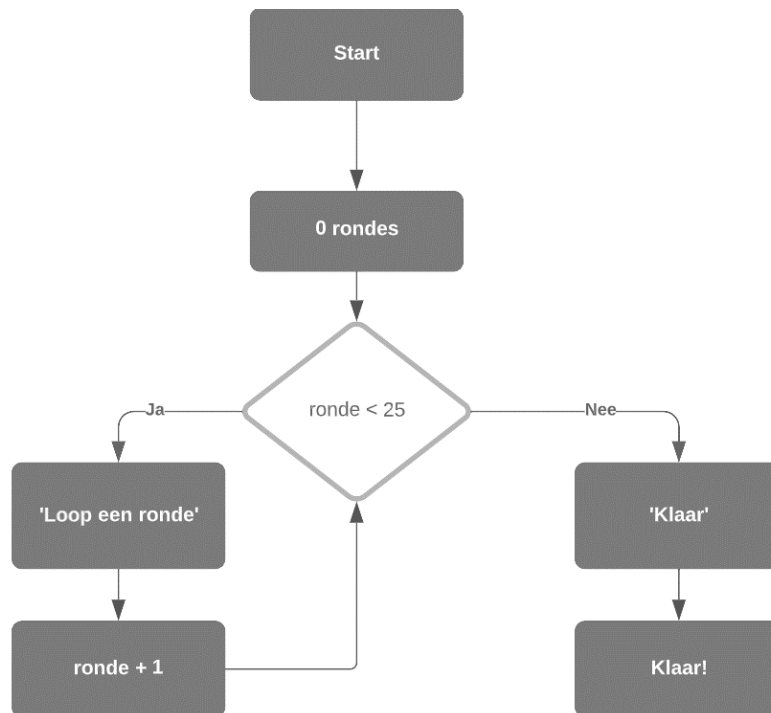
### Mogelijke oplossing:

main.py	Run	Shell	Clear
<pre>1 Wachtwoord = input('Wat is je wachtwoord? ') 2 if Wachtwoord == "IkHouVan3Panda's": 3     print("Juist wachtwoord") 4 else: 5     print("Onjuist wachtwoord") 6</pre>		<pre>Wat is je wachtwoord? IkHouVan3Panda's Juist wachtwoord &gt;  </pre>	



## Oefening 14:

Gegeven onderstaande flowchart. Giet dit in een programma.




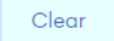
Oplossing programma:

main.py		Shell	Clear
1 # ronde is een teller die het aantal gelopen rondjes bijhoudt		Loop een ronde	
2 # aan de start heb je 0 rondes gelopen		Loop een ronde	
3 ronde = 0		Loop een ronde	
4 # indien je minder dan 25 rondes gelopen hebt, loop dan nog een ronde		Loop een ronde	
5 while ronde < 25:		Loop een ronde	
6     print('Loop een ronde')		Loop een ronde	
7     ronde = ronde + 1		Loop een ronde	
8 # na 25 rondes ben je klaar		Loop een ronde	
9 print('Klaar!')		Loop een ronde	
10		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Loop een ronde	
		Klaar!	
		>	

### Oefening 15:

Schrijf een programma dat vraagt naar een nummer. Vervolgens laat je het programma weergeven uit hoeveel cijfers het nummer bestaat.

#### Mogelijke oplossing:

main.py	  	Shell	
<pre>1 nummer = int(input("Wat is je nummer? ")) 2 tellen = 0 3 while nummer != 0: 4     nummer = nummer // 10 5     tellen = tellen + 1 6 print("Het nummer bestaat uit", tellen, "cijfers.") 7</pre>		<pre>Wat is je nummer? 12345678910 Het nummer bestaat uit 11 cijfers. &gt;  </pre>	

## Oefening 16:

In deze oefening ga je een quiz voor je klasgenoten programmeren. Houd rekening met de volgende spelregels:

- De quiz bestaat uit minimum 3 vragen.
- De speler krijgt 3 levens. Wanneer een vraag fout beantwoord wordt, verliest de speler een leven.
- De speler krijgt pas een nieuwe vraag wanneer de vorige vraag correct beantwoord is.
- Elke keer wanneer de speler iets ingeeft, krijgt hij te zien wat zijn score tot dan toe is en hoeveel levens hij/zij nog over heeft.
- Wanneer de levens van de speler op zijn, stopt de quiz en krijgt de speler zijn uiteindelijke score te zien.

Hieronder vind je 3 voorbeeldvragen met telkens het juiste antwoord. Voel je vrij om zelf vragen te bedenken.

Welke beer leeft op de Noordpool? ijsbeer

Wat is het snelste landdier? luipaard



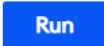
Welk dier is het grootst? blauwe vinvis

### Mogelijke oplossing:

main.py	Run	Shell	Clear
<pre>1 score=0 2 levens=3 3 nog_radend=True 4 5- while nog_radend and levens&gt;0: 6     vraag_1=input("Welke beer leeft op de Noordpool?") 7-     if vraag_1=="ijsbeer": 8         score=score+1 9         print("Goed zo! Jouw score is nu "+str(score)+ " Je hebt nog " + str(levens) + " leven(s)") 10        nog_radend=False 11-     else: 12        levens=levens-1 13        print("Dat is niet juist. Je score is "+str(score)+ " Je hebt nog " + str(levens) + " leven(s)") 14 15 nog_radend=True 16- while nog_radend and levens&gt;0: 17     vraag_1=input("Wat is het snelste landdier?") 18-     if vraag_1=="luipaard": 19         score=score+1 20         print("Goed zo! Jouw score is nu "+str(score)+ " Je hebt nog " + str(levens) + " leven(s)") 21        nog_radend=False 22-     else: 23        levens=levens-1 24        print("Dat is niet juist. Je score is "+str(score)+ " Je hebt nog " + str(levens) + " leven(s)") 25 26 nog_radend=True 27- while nog_radend and levens&gt;0: 28     vraag_1=input("Welk dier is het grootst?") 29-     if vraag_1=="blauwe vinvis": 30         score=score+1 31         print("Goed zo! Jouw score is nu "+str(score)+ " Je hebt nog " + str(levens) + " leven(s)") 32        nog_radend=False 33-     else: 34        levens=levens-1 35        print("Dat is niet juist. Je score is "+str(score)+ " Je hebt nog " + str(levens) + " leven(s)") 36 37 print("Je uiteindelijke score is "+str(score))</pre>		<pre>Welke beer leeft op de Noordpool?ijsbeer Goed zo! Jouw score is nu 1 Je hebt nog 3 leven(s) Wat is het snelste landdier?d Dat is niet juist. Je score is 1 Je hebt nog 2 leven(s) Wat is het snelste landdier?luipaard Goed zo! Jouw score is nu 2 Je hebt nog 2 leven(s) Welk dier is het grootst?e Dat is niet juist. Je score is 2 Je hebt nog 1 leven(s) Welk dier is het grootst?blauwe vinvis Goed zo! Jouw score is nu 3 Je hebt nog 1 leven(s) Je uiteindelijke score is 3 &gt; .</pre>	

### Oefening 17:

Wat doet onderstaand programma? Geef een mogelijke opgave.



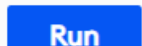
```
main.py     
1 for getal in range (137, 192, 2):  
2     if getal//100 + (getal%100)//10 + getal%10 ==11:  
3         print(getal)  
4
```

### Oplossing:

Er wordt van de oneven getallen, van 137 tot 192, nagegaan of de som van hun cijfers gelijk is aan 11. Indien dat het geval is, wordt het getal afgedrukt op het scherm.

### Oefening 18:

Als je dit programma zou ingeven, krijg je een foutmelding. Welke fout zit in dit programma? Hiervoor mag je **geen** gebruik maken van je computer!

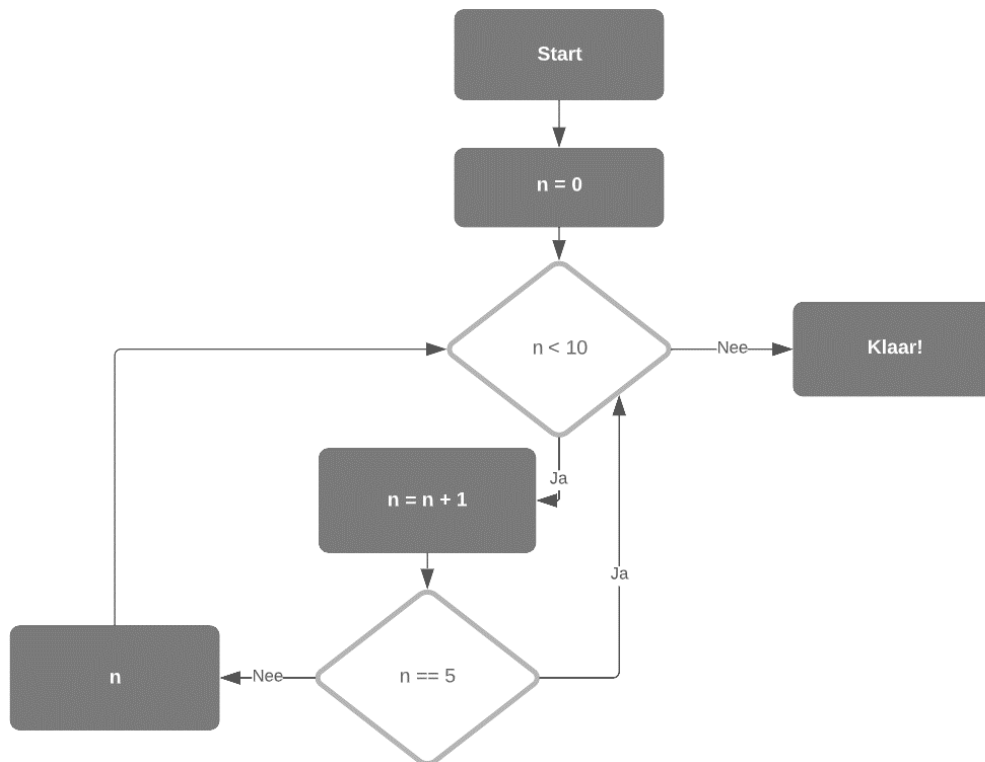
```
main.py     
1 lengte_ribbe = input('Wat is de lengte van de ribbe?')  
2 inhoud = a**3  
3 print(inhoud)  
4
```

### Oplossing:

De functie float() of int() mist. De uitkomst van een input() is altijd een string. Je kan een string niet tot de derde macht verheffen, dus zal je van de lengte van de ribbe een float of integer moeten maken.

### Oefening 19:

Gegeven is een flowchart. Programmeer deze in Python.



### Oplossing:

main.py		Run	Shell	Clear
1	n = 0		5	
2	while n < 10:		>	
3	n = n + 1			
4	if n == 5:			
5	print(n)			
6				

## Oefening 20:

Een perfect getal is een positief geheel getal dat de som is van al zijn delers, behalve zichzelf. Het getal 6 is het eerste perfecte getal, aangezien al de delers van 6 (1, 2 en 3) optellen tot 6.

Kan jij met deze informatie een programma schrijven dat alle perfecte getallen onder een miljoen geeft? Wanneer dit je lukt, zal je zien dat er maar 4 perfecte getallen bestaan onder een miljoen.

Let op: Python moet een miljoen getallen controleren, het kan dus even duren voordat hij het 4<sup>de</sup> getal vindt. De eerste 3 komen normaal wel redelijk vlot tevoorschijn.

### Mogelijke oplossing:

main.py	Run	Shell	Clear
<pre>1  getal=0 2 3  while getal &lt;=1000000: 4      deler=1 5      som=0 6      while deler&lt;getal: 7          if getal%deler==0: 8              som=som+deler 9              deler=deler+1 10         if som==getal and (deler+1)==getal: 11             print(getal) 12     getal=getal+1</pre>		<pre>6 28 496 8128</pre>	

## 6. Bijlagen

### 6.1 Leerplannen

#### 6.1.1 Wiskunde

##### **Leerplannen: D/2021/13.758/045**

LPD 2: De leerlingen onderzoeken interacties tussen wiskunde en andere domeinen via wiskundige toepassingen.

LDP 60: De leerlingen ontwerpen algoritmes om problemen digitaal op te lossen.

- Concepten van computationeel denken: decompositie, patroonherkenning, abstractie en algoritme
- Organisatie, modellering, simulatie en digitale representatie van informatie
- Debuggen
- Principes van programmeren: opeenvolging, herhalingsstructuur, keuzestructuur
- Ingebouwde functies
- Elementen van programmeertalen: variabelen, datatypes, operatoren, parameters, condities, procedures of functies.

##### **Eindtermen**

4.5: De leerlingen ontwerpen algoritmen op problemen digitaal op te lossen. Met inbegrip van kennis.

- Conceptuele kennis
  - Concepten van computationeel denken: decompositie, patroonherkenning, abstractie en algoritme
  - Organisatie, modellering, simulatie en digitale representatie van informatie
  - Debuggen: testen en bijsturen
  - Principes van programmeren: sequentie, herhalingsstructuur, keuzestructuur
  - Ingebouwde functies
  - Elementen van programmeertalen: variabelen, datatypes, operatoren, parameters, condities, procedures of functies.

- Procedurele kennis
  - Toepassen van principes van computationeel denken: decompositie, patroonherkenning, abstractie, algoritme
  - Toepassen van principes van organisatie, modellering, simulatie en digitale representatie van informatie
  - Toepassen van principes om te debuggen
  - Toepassen van principes van programmeren: sequentie, herhalingsstructuur, keuzestructuur
  - Toepassen van controlestructuren en eenvoudige gegevensstructuren bij het formuleren van algoritmen
  - Toepassen van principes om algoritmen bestaande uit een aantal samenwerkende procedures te ontwerpen en te implementeren in een programmeeromgeving.

## 6.1.2 Biologie

### **Leerplan eerste graad: D/2019/13.758/011**

LPD 9: De leerlingen onderzoeken via een terreinstudie voor een biotoop de onderlinge afhankelijkheid van verschillende organismen en de rol van biotische en abiotische factoren.

### **Leerplan derde graad: D/2014/7841/011**

LPD B23: De chemische structuur van het DNA en RNA bespreken.

LPD B38: Aan de hand van concrete vraagstukken, kruisingsschema's en stambomen, de overerving van codominante allelen, letale allelen, multiële allelen, polygenie, geslachtsgebonden allelen, gekoppelde genen en recombinatie van gekoppelde genen interpreteren en toelichten.

## 6.1.3 ICT

### **Leerplan tweede graad: D/2021/13.758/001**

LPD 3: De leerlingen gebruiken doelgericht en adequaat standaardfunctionaliteiten van digitale infrastructuur en online en offline toepassingen om digitale inhoud te beheren.

- Standaardmethodes voor het beheer van data: gestructureerd bewaren en beheren, schijfhygiëne, grote hoeveelheden bestanden gelijktijdig manipuleren, downloaden, uploaden, back-up, synchroniseren, comprimeren, snelkoppeling
- Inzicht in de opbouw van een gegevensbank
- Zoeken gegevens: filters, sorteren

GLI-ddaa 9: De leerlingen gebruiken doelgericht en adequaat standaardfunctionaliteiten van digitale infrastructuur en toepassingen om digitale rekenbladen te creëren.



- Wenk 8: Je kan in de D-finaliteit geneste ALS-functies inoefenen.

## **6.1.4 STEM**

### **Leerplan tweede graad: D/2021/13.758/034**

LPD S2: De leerlingen analyseren natuurlijke en technische systemen aan de hand van verschillende STEM-concepten.

- ✓ Je kan patronen zoals in kenmerken van organismen, systemen, landschappen ... laten ontdekken als basis voor classificatie, ordening en optimalisering.
- ✓ Grafieken, diagrammen en kaarten kunnen helpen om patronen te ontdekken.

LPD S8: De leerlingen ontwikkelen modellen om te visualiseren, te onderzoeken, op te lossen en te verklaren.

- ✓ Technologische modellen zoals schaalmodellen, technische tekeningen, algoritmes schema's ...komen ook aan bod in STEM-leerplandoel 9.

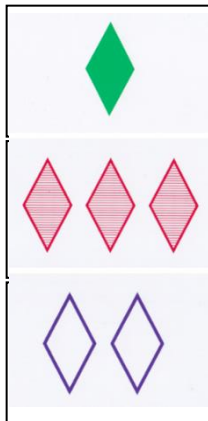
## 6.2 Verbetersleutel set

### Oefening 1:

Vormen onderstaande kaarten een set?

- Bespreek iedere eigenschap.
- Besluit vervolgens of er al dan niet een set gevormd wordt. Duid dit aan.
- Geef tot slot een verantwoording waarom het wel/niet een set is.

1.



Vorm: De vorm is steeds hetzelfde.

Kleur: De kleuren zijn steeds verschillend.

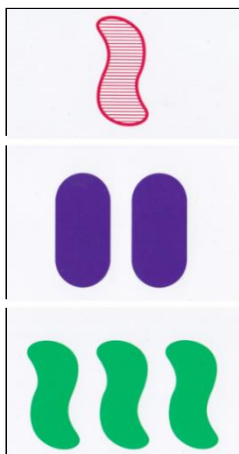
Opvulling: De opvulling is steeds verschillend.

Aantal: Het aantal is steeds verschillend.

Dit is **wel**/niet een set.

Verantwoording: Alle eigenschappen zijn voor de 3 kaarten ofwel gelijkend ofwel verschillend.

2.



Vorm: De vorm is niet overal verschillend maar ook niet overal gelijk.

Kleuren: De kleuren zijn steeds verschillend.

Opvulling: De opvulling is niet overal verschillend maar ook niet overal gelijk.

Aantal: Het aantal is steeds verschillend.

Dit is wel/**niet** een set. Deze conclusie kon al genomen worden bij het bestuderen van de vorm. De andere eigenschappen bekijken was dus eigenlijk overbodig.

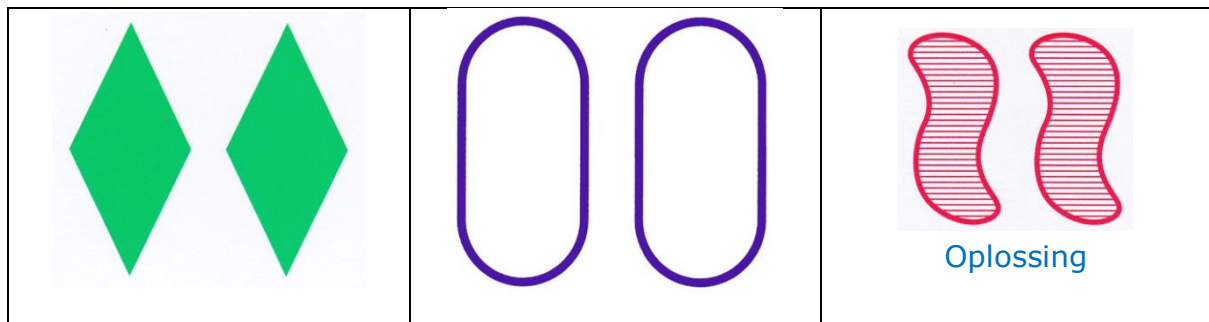
Verantwoording: Niet alle eigenschappen zijn voor de 3 kaarten gelijkend of verschillend.

## Oefening 2:

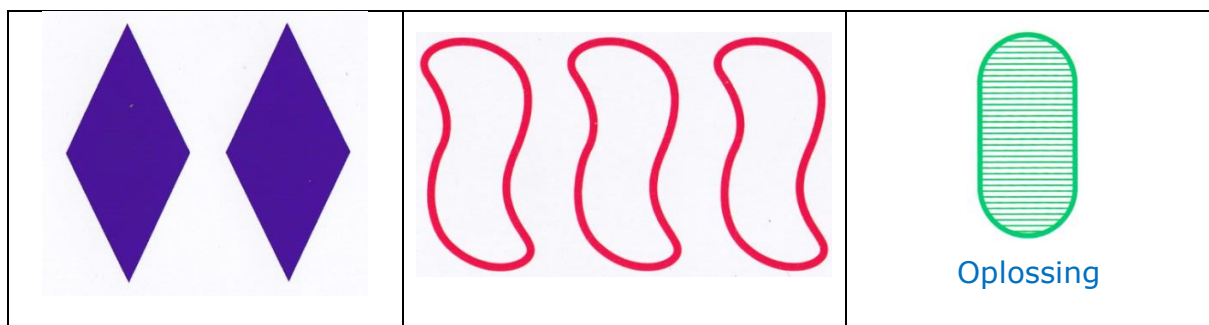
Er zijn steeds twee kaarten gegeven. Ga zelf op zoek naar de derde kaart om de set te vervolledigen. Je kent de eigenschappen waar een set aan moet voldoen.

- Teken in het laatste vakje de ontbrekende kaart van de set. Let op elke eigenschap!

1.

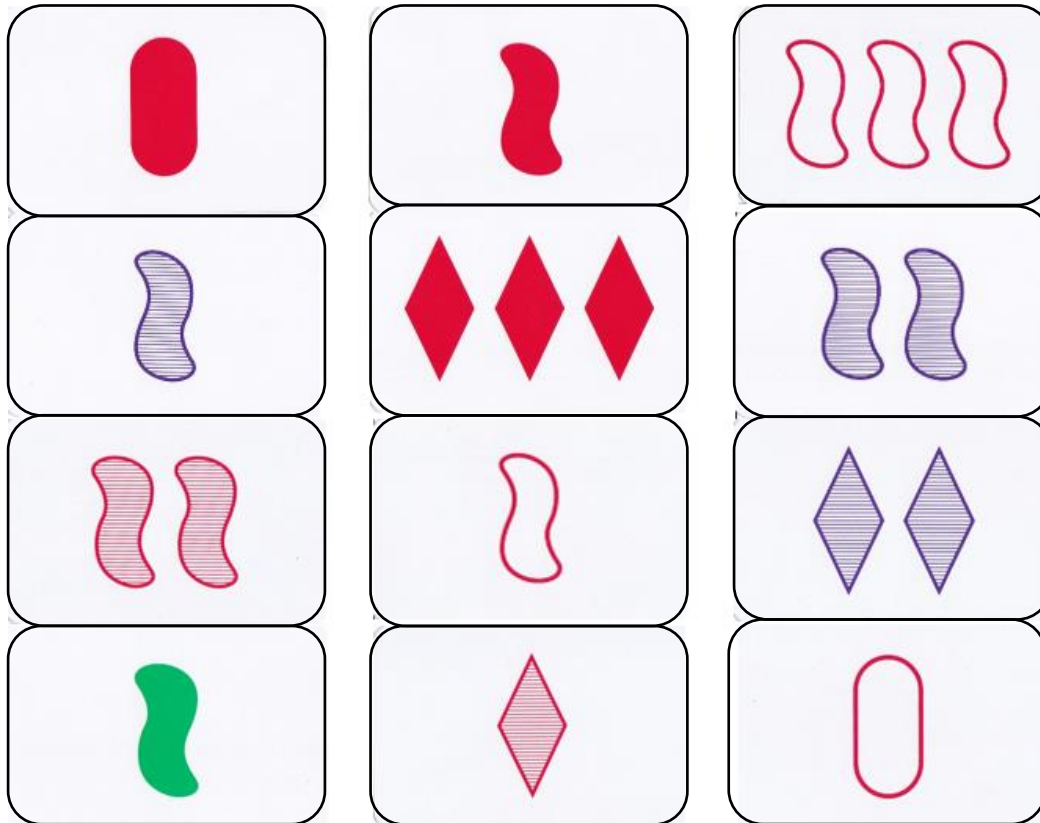


2.



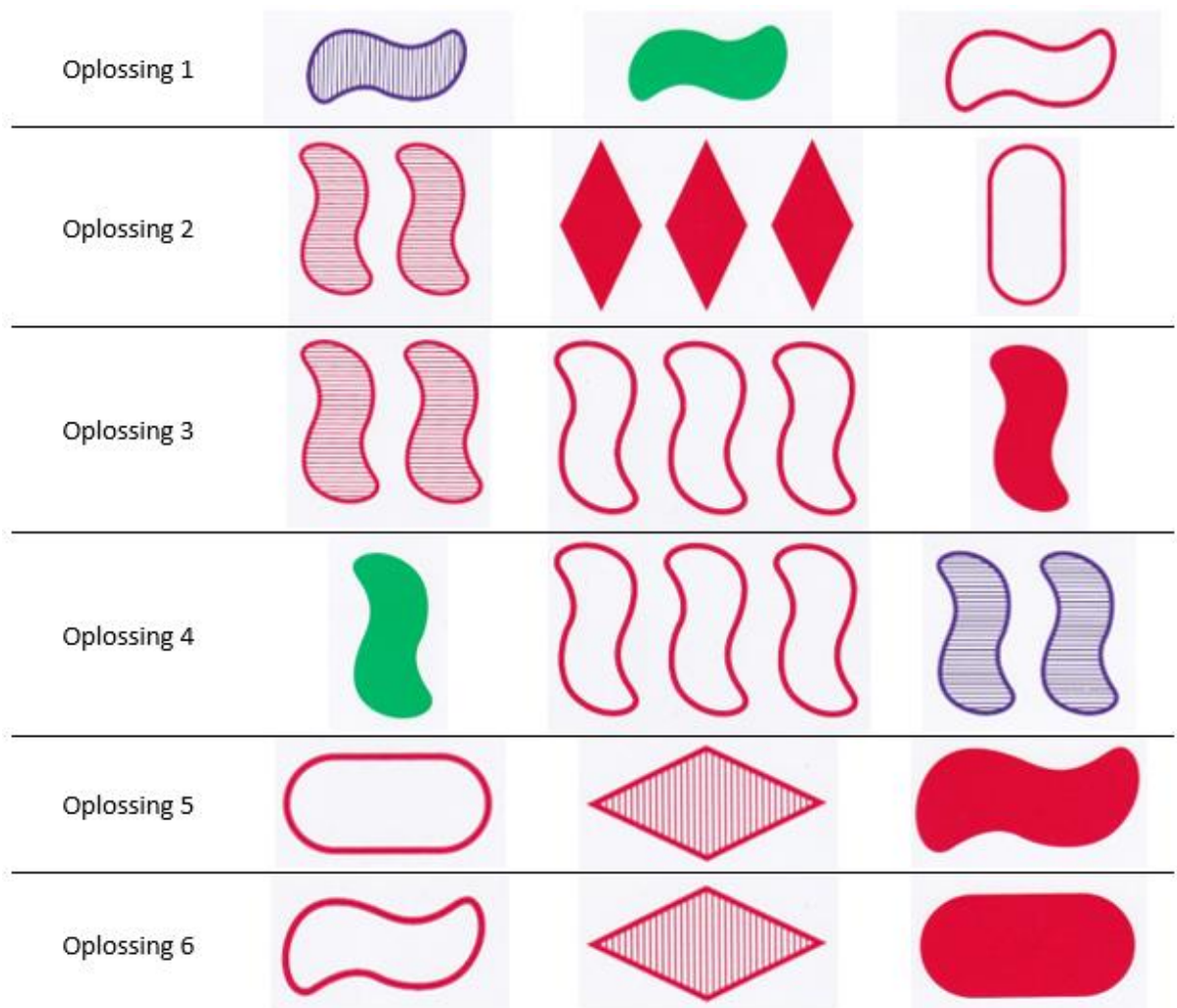
### Oefening 3:

Hieronder zijn 12 kaarten gegeven. De bedoeling is dat jullie hier zo veel mogelijk sets in terug vinden. Hieronder staat een tabel waar je de figuren kan in overtekenen en zo de set zichtbaar kan maken.



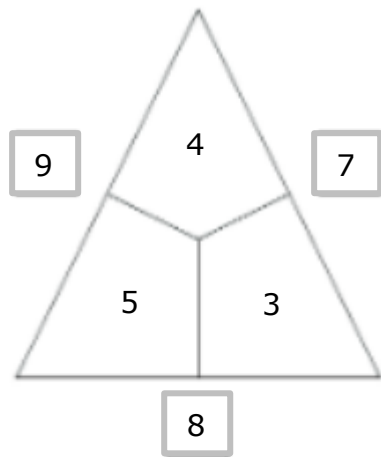
Teken 3 sets die je kan terug vinden.

## Oplossing



## 6.3 Verbetersleutel driehoeken

Voorbeeld van een correct ingevulde driehoek:



**Beantwoord volgende vragen:**

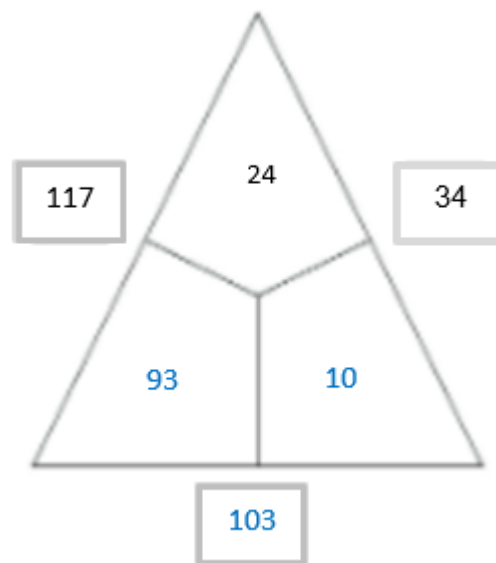
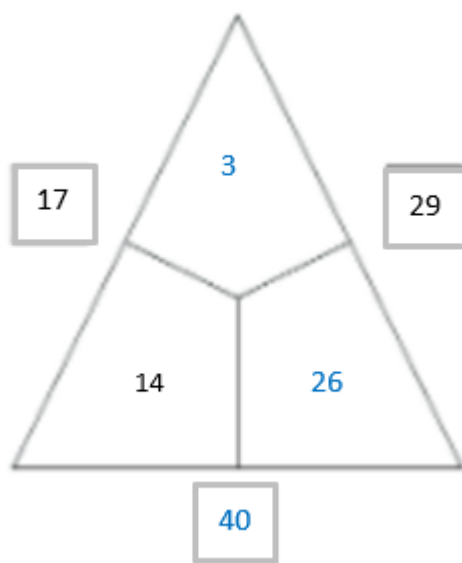
Som getallen in de vierkanten = 24

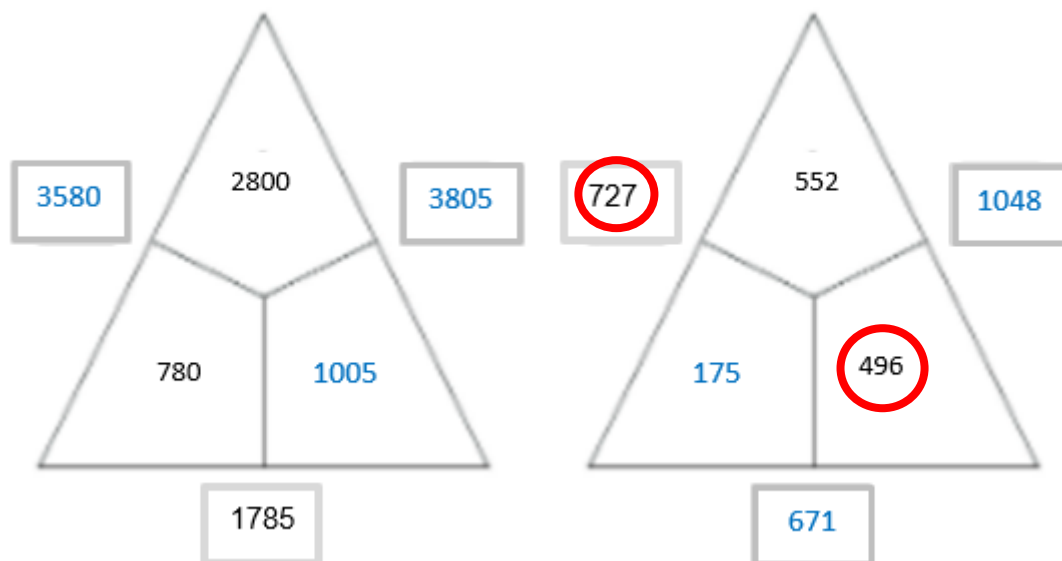
Som getallen in de driehoek = 12

Welk verband vind je tussen de som van de getallen in de vierkantjes en de som van de getallen binnen de driehoek?

De som van de getallen binnen de driehoek is de helft van de som van de getallen in de vierkantjes.

Vervolledig volgende driehoeken. In te vullen





### Verbanden:

Om dergelijke driehoeken vlot op te lossen wordt gebruik gemaakt van 2 verbanden. Probeer beide verbanden aan te vullen/op te sporen.

1. De binnensom is de helft van de buitensom.
2. Zoek een verband tussen de binnensom en de omcirkelende getallen (zie voorgaande opdracht). Welk verband bestaat er tussen deze componenten?   
Neem de binnensom en trek daar een getal dat buiten de driehoek staat vanaf. Dit is het getal dat tegenover dit getal in de driehoek moet komen.