Kraak jij de wiskundige en biologische code?

STEM-PROJECT

Naam:

Klas:

Schooljaar: 2022/2023

Inhoudsopgave

1. Introductie

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: 

<u>Doelstellingen:</u>

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.

2. Inleidende opdracht

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.

2.1 Opdracht 1: Set

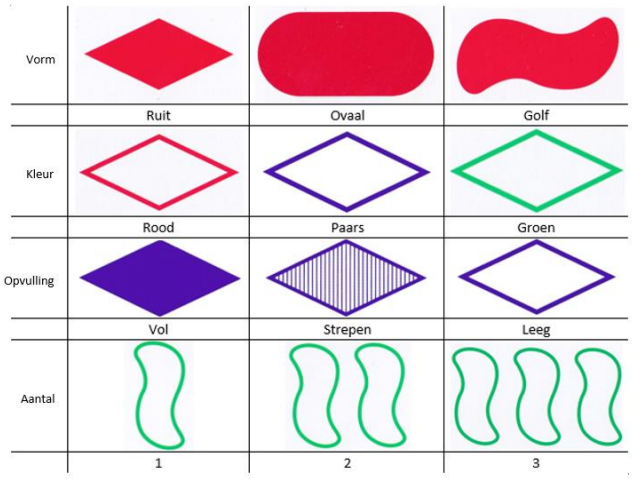
Benodigdheden: kaarten

Tijd: maximum 15 min.

Gecontroleerd: ☐

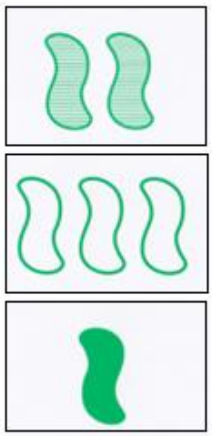
Uitleg spel:

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.)



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.

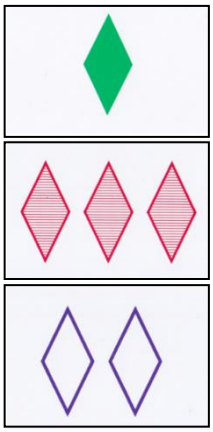
Tijd voor oefeningen

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: .....

Kleur:.....

Opvulling: .....

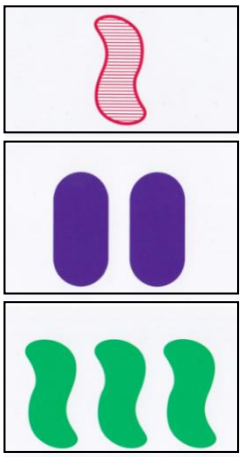
Aantal: .....

Dit is wel/niet een set.

Verantwoording: .....

.....

2.



Vorm: .....

Kleuren: .....

Opvulling:.....

Aantal: .....

Dit is wel/niet een set.

Verantwoording: .....

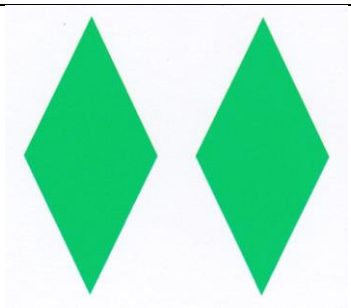
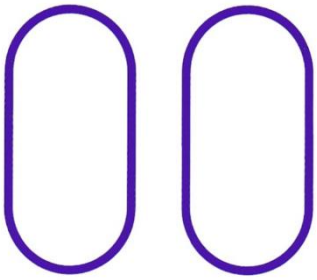
.....

Oefening 2:

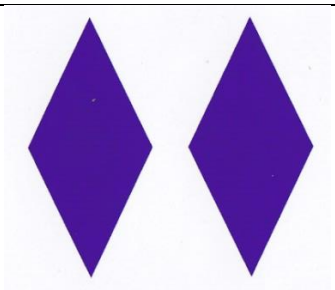
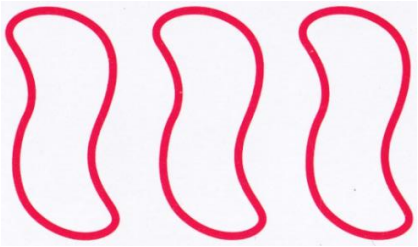
Er zijn steeds twee kaarten gegeven. Ga zelf op zoek naar de derde kaart om de set te vervolledigen. Je kent de eigenschappen waar een set aan moet voldoen.

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

1.

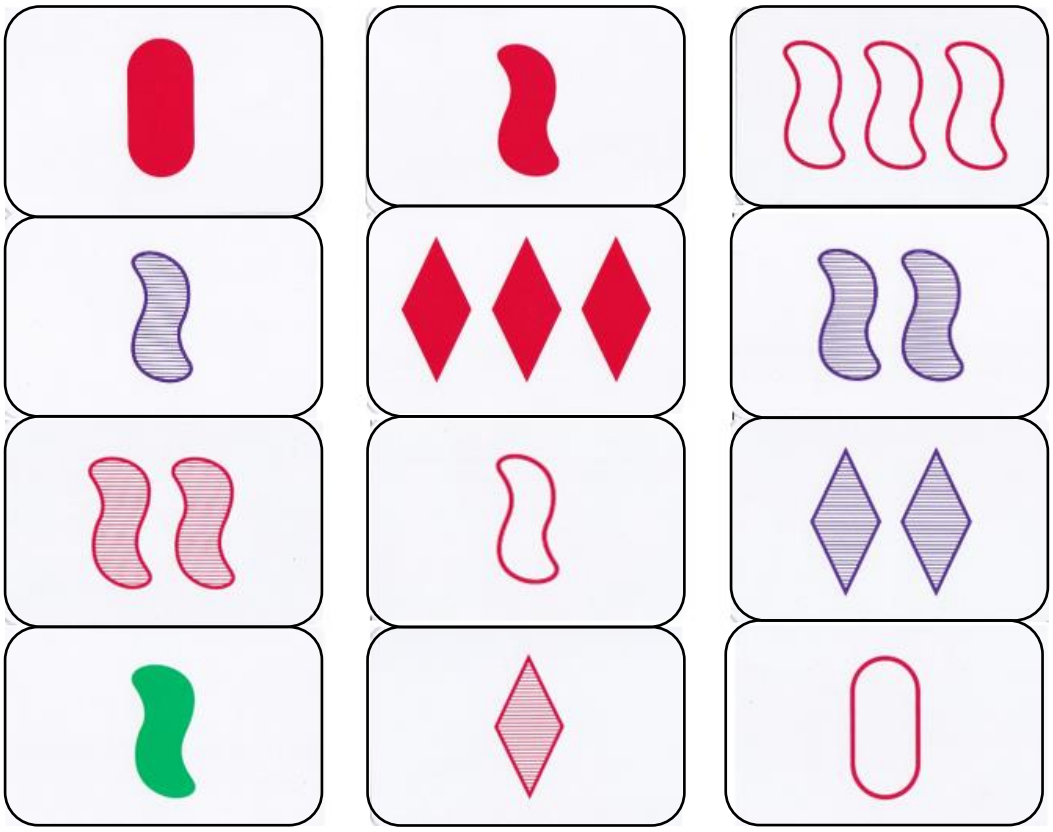
		

2.

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			

2.2 Opdracht 2: driehoeken

Benodigdheden: driehoeken

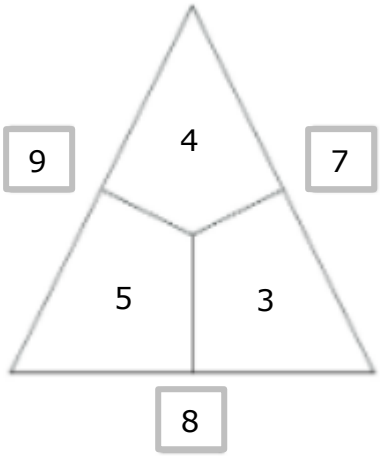
Tijd: maximum 15 min.

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 = ...

Som getallen in de driehoek = ...

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

.....

.....

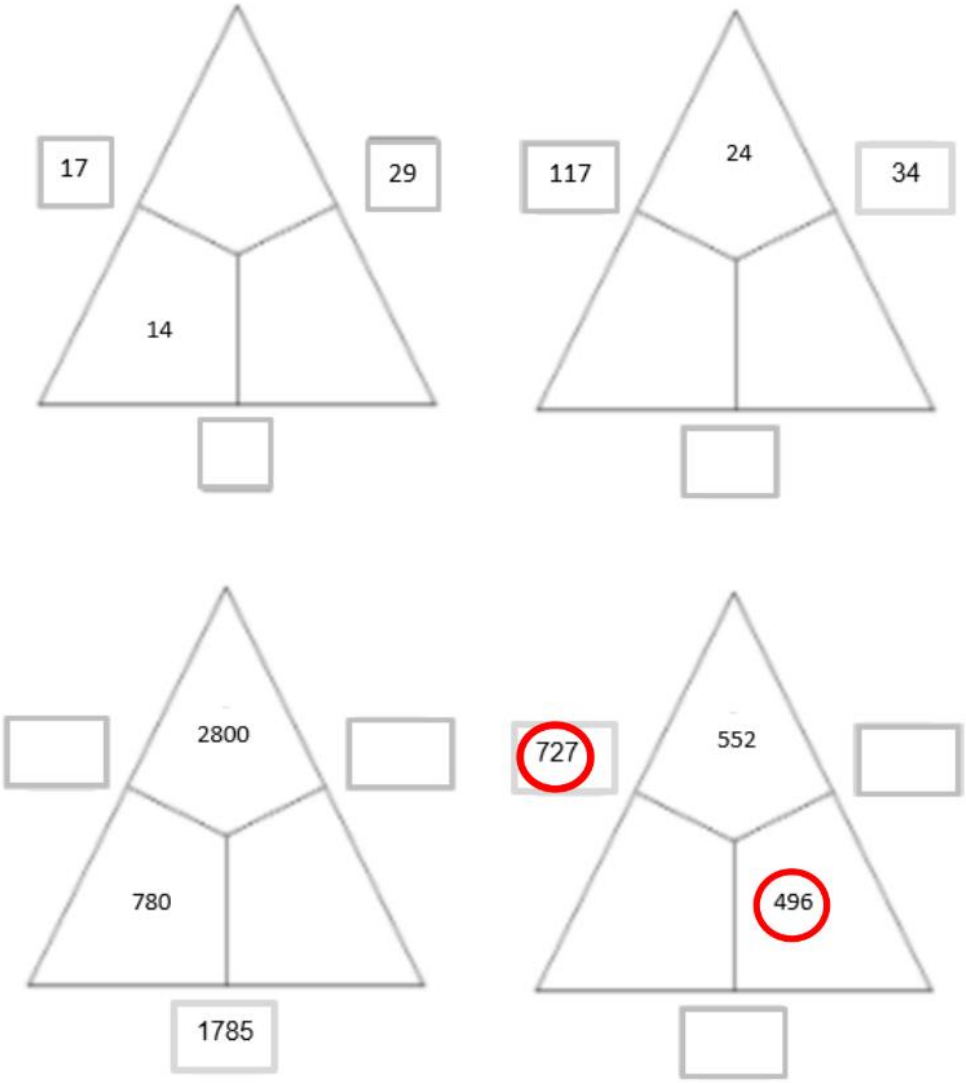
.....

.....

.....

Tijd voor oefeningen

Vervolledig volgende driehoeken.



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 ..... van de buitensom.

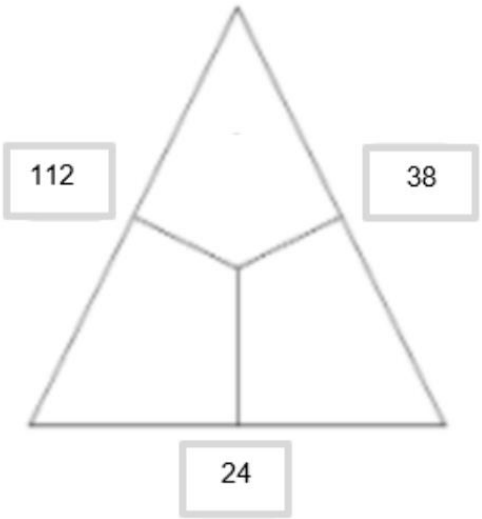
2. Zoek een verband tussen de binnensom en de omcirkelende getallen (zie voorgaande opdracht). Welk verband bestaat er tussen deze componenten?

.....

.....

Extra:

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



2.3 Opdracht 3: smartgames

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!

3. Het stappenplan

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.

3.1 Decompositie

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.

-
-
-
-
-

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

.....

.....

.....

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

Decompositie is .....

.....

.....

.....

Tijd voor oefeningen

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

Dierlijke cel	Plantaardige cel

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

-
-
-
-
-

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

3.2 Patroonherkenning

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

<u>Tijd voor oefeningen</u>: patronen herkennen en aanvullen

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

a) 3 → 2 → 6 → 0 → 9 → -2 → 12 → …… → ……→ …… → …… → ……

Patroon: .....

b) 0 → 1 → 7 → 17 → 30 → 45 → …… → …… → …… → …… → ……

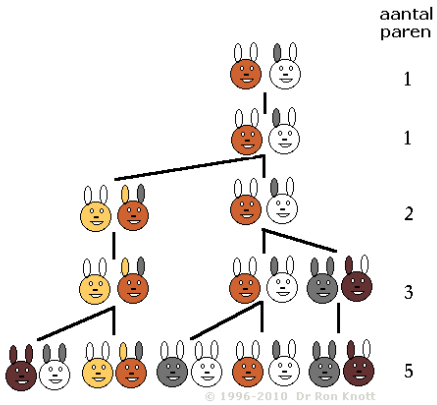
Patroon.....

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 (uiteraard) met twee, maar al snel had hij er drie. Na het bestuderen van vermenigvuldigen van deze dieren kwam hij tot de reeks:

0 → 1 → 1 → 2 → 3 → 5 → 8 → 13 → …… → …… → …… → …… → ……

Patroon: .....



Figuur 1: rij van Fibonacci

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:

```
def fib(N):
    if N < 2:
        return 1
    return (fib(N-1) + fib(N-2))

resultaat = fib(11)
print(resultaat)
```

```
144
>
```

Figuur 2: 11de getal van de rij van Fibonacci

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

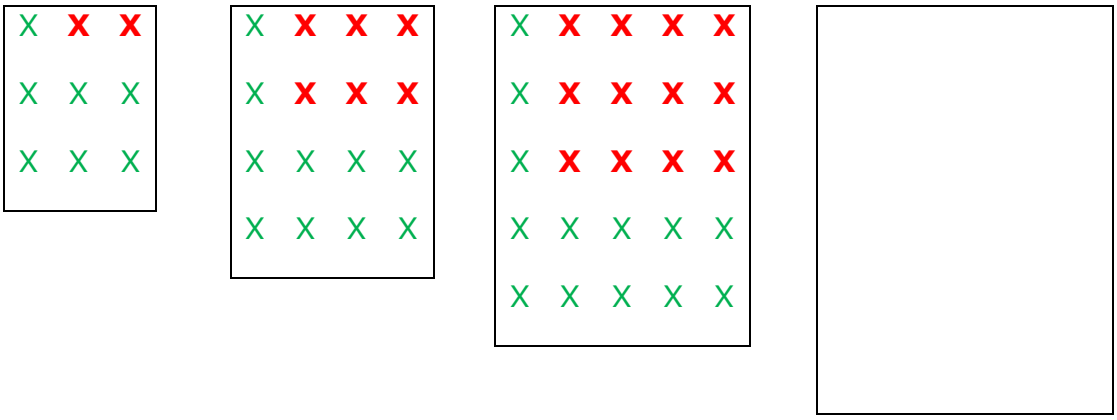
```
def fib(N):
    if N < 2:
        return 1
    return (fib(N-1) + fib(N-2))

resultaat = fib(44)
print(resultaat)
```

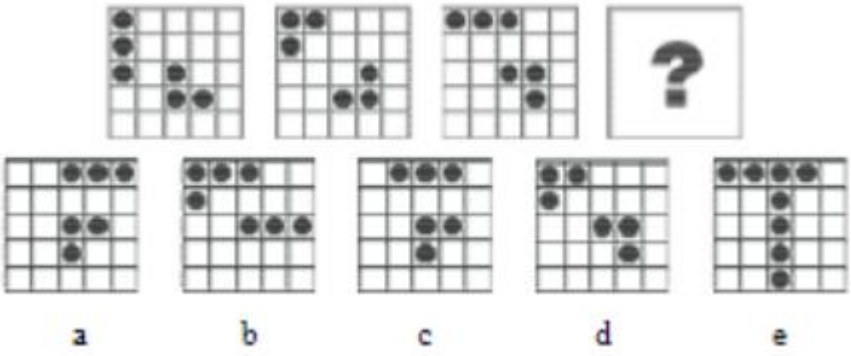
```
1134903170
>
```

Figuur 3: 44ste getal van de rij van Fibonacci

d) Teken het eerst volgende patroon:



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



Patronen zijn .....

.....

.....

.....

<u>Even over het muurtje kijken in de biologie:</u>

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?

.....

.....

.....

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)

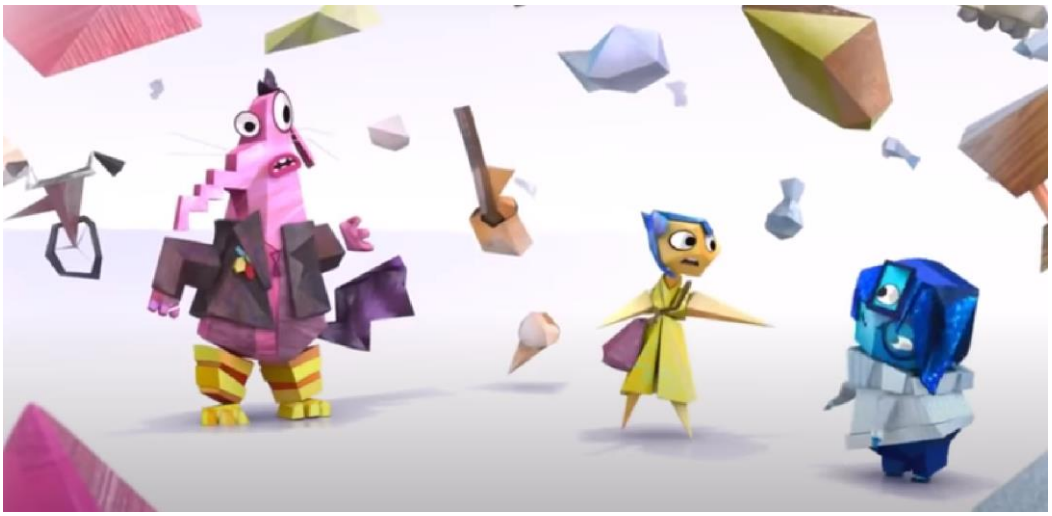
3.3 Abstractie

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

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

.....

.....



Figuur 8: Fragment 'Inside out'

Abstractie is .....

.....

.....

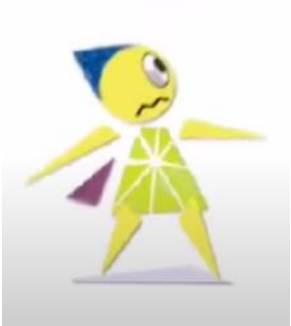
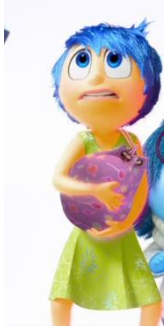
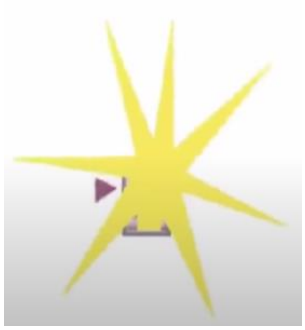
.....

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:

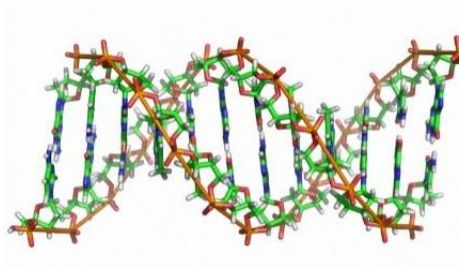
			
1	2	3	4

	Minst abstract ⇔ Meest abstract			
Reeks 1				
Reeks 2				

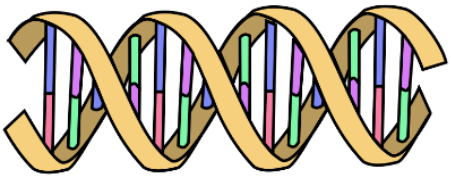
Oefening 2:

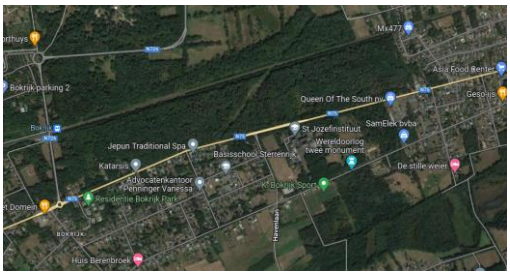
Abstractie is iets dat we vaak in ons leven gebruiken zonder er bij stil te staan. Kan je zelf een aantal voorbeelden geven wanneer we het gebruiken en waarom?

Hieronder staan een aantal afbeeldingen die ter inspiratie kunnen dienen.

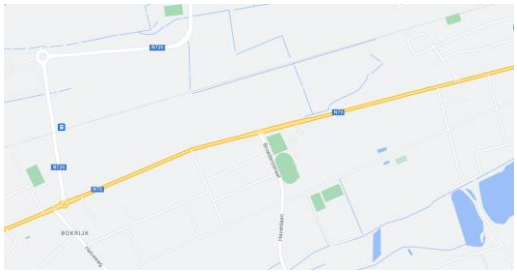


⇒





⇒



.....

.....

.....

.....

.....

.....

.....

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?

3.4 Algoritme

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.

2.

3.

4.

5.

…

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?

.....

Een algoritme is .....

.....

.....

.....

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?

.....

.....

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

.....

.....

.....

.....

.....

Tijd voor oefeningen

Oefening 1:

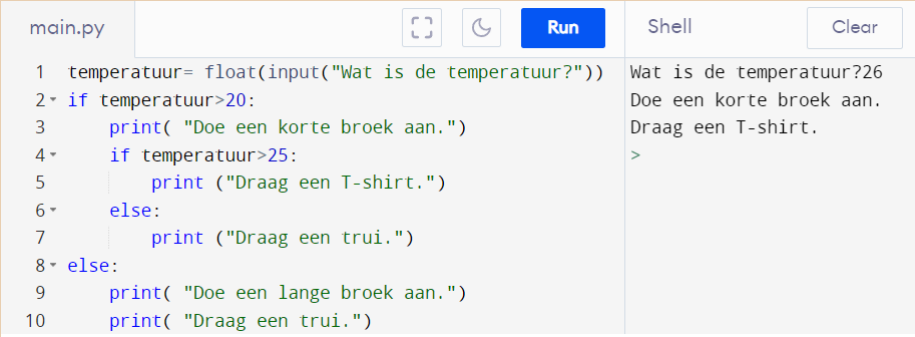
Als het warmer is dan 20°C dan draag ik een korte broek. Wanneer het kouder is dan 20°C draag ik een lange broek. Voor mijn outfit heb ik ook nog een trui of T-shirt nodig. Als het warmer is dan 25°C doe ik een T-shirt aan. Anders draag ik een trui.

Stel hiervoor een algoritme op.

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:



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 gevroren en de wegen glad zijn, kies ik er voor om te voet naar school te gaan. Als het niet heeft gevroren, 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.

3.5 Foutenanalyse

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?

.....

.....

.....

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 .....

.....

.....

.....

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

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.

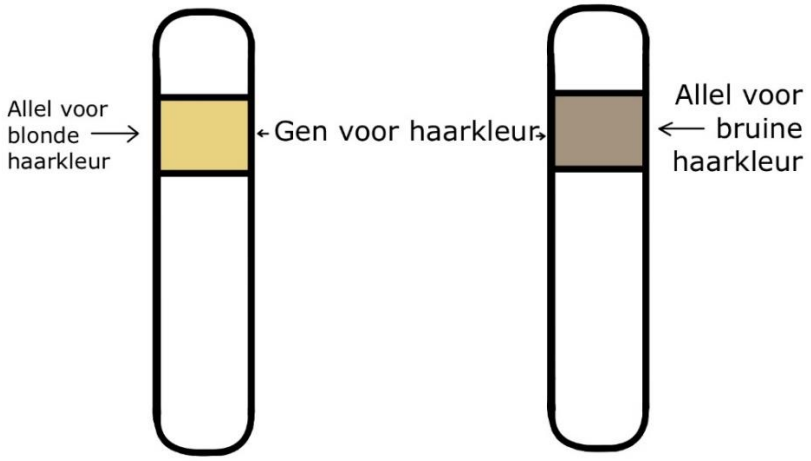
Iedere cel bevat een celkern waarin het erfelijk materiaal of DNA gelegen is. Dat wordt gedragen door verschillende chromosomen die elk opgebouwd zijn uit 2 chromatiden. Het DNA bevat de erfelijke code die ervoor zorgt dat ieder individu uniek is. Deze code wordt gevormd door combinaties van 4 verschillende 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.

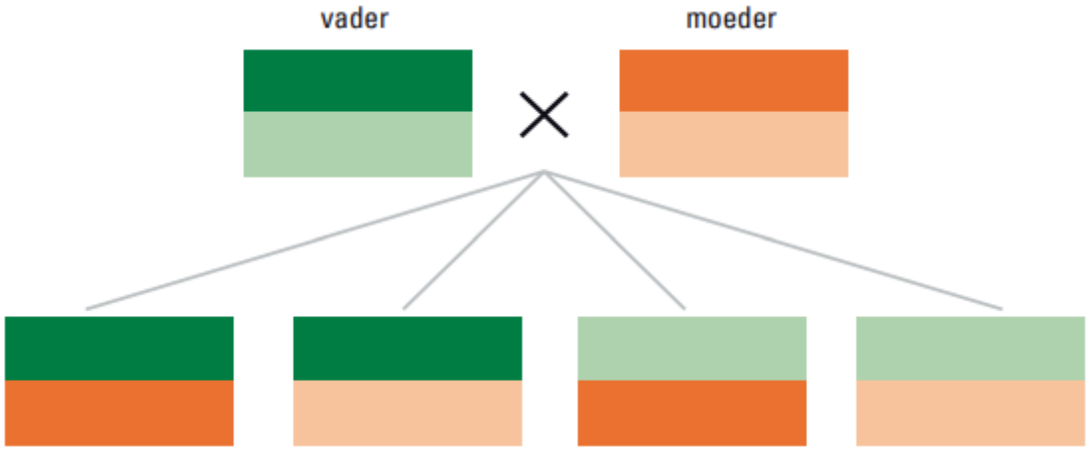
Wanneer er een nakomeling gevormd wordt, zal deze zowel van de moeder als de vader hetzelfde basisgen doorkrijgen maar het resultaat zal anders zijn. Dit komt doordat de code in het gen dat bestaat uit A, C, T en G een andere volgorde heeft bij de moeder en de vader. Deze twee varianten in een gen noemen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

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 $2 = 1$

$$a = b$$

$$a^2 = ab$$

$$2a^2 = a^2 + ab$$

$$2a^2 - 2ab = a^2 - ab$$

$$2(a^2 - ab) = a^2 - ab$$

$$2 = \frac{a^2 - ab}{a^2 - ab}$$

$$2 = 1$$

Welke fout zit er in dit bewijs?

.....

.....

.....

.....

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.

.....

.....

.....

3.6 Overzicht van het stappenplan

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.

SCAN me!

3.7 Algemene oefeningen

Oefening 1:

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

a) Hoe zou je dit doen?

.....

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?

.....

.....

.....

.....

.....

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

.....

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

.....

.....

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

.....

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

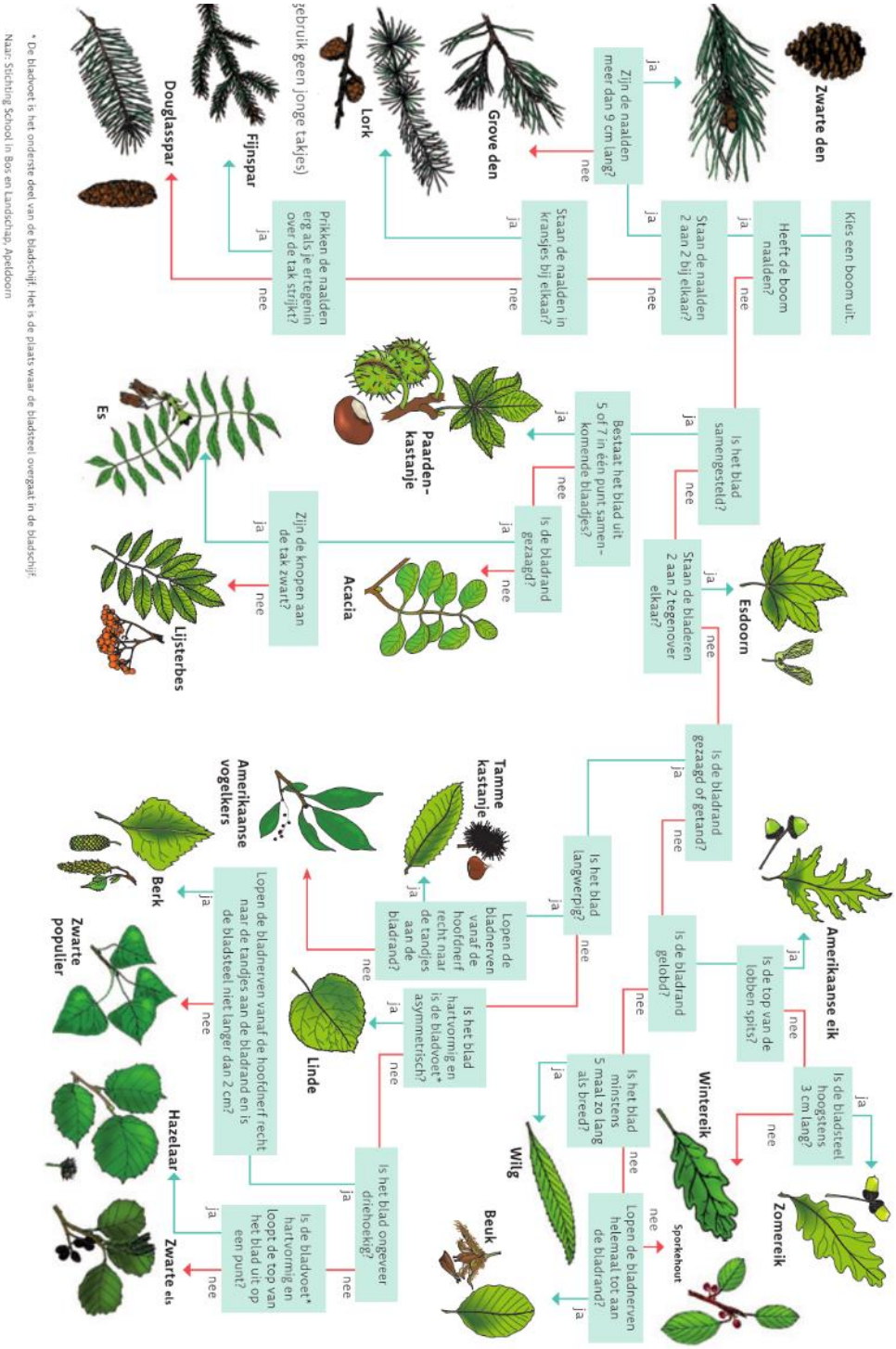
.....

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.

4. Programmeren in Python

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?

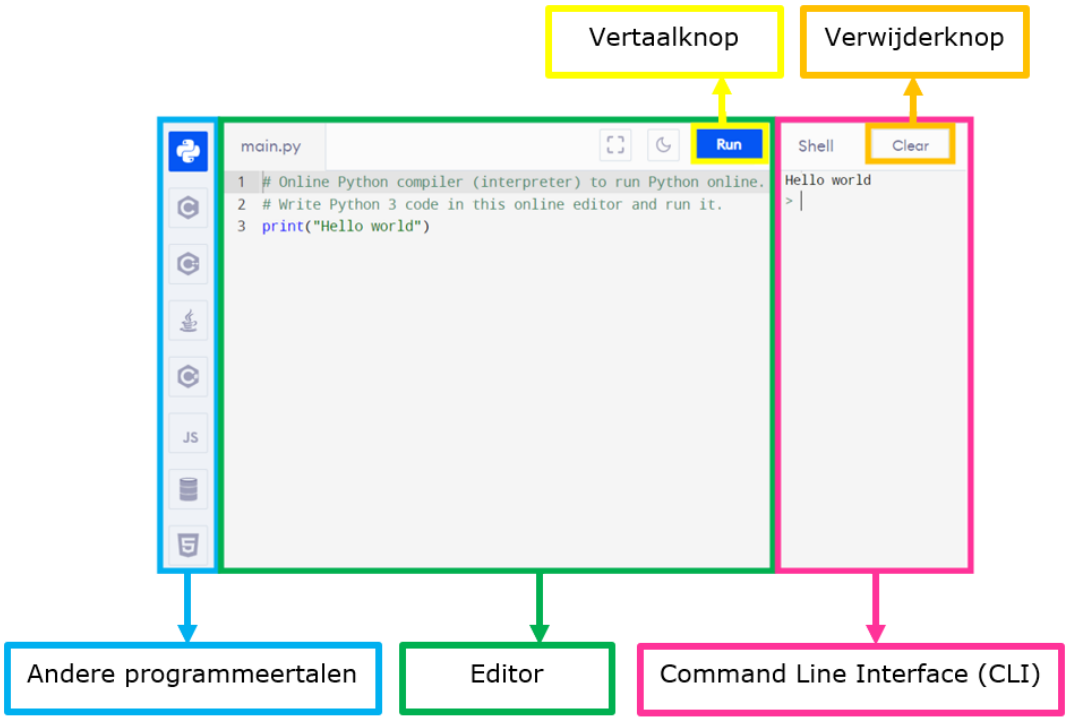
- Het is toegankelijk, laagdrempelig en toch relevant.
- Eenmaal je Python onder de knie hebt, kan je gemakkelijk overstappen naar een andere programmeertaal.
- Python kan door veel mensen worden gebruikt, denk maar aan ingenieurs, wiskundigen, economisten, informatici, …

4.1 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.

4.1.1 Startscherm

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



Onderdelen startscherm	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.

4.1.2 Bewerkingen in Python

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

<u>Oefening:</u>

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		
b)	a – b		
c)	a * b		
d)	a / b		
e)	a**b		
f)	a**(1/b)		
g)	a**(1/c)		
Euclidische deling			Uitkomst
h)	a // b		
i)	a % b		

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	
b)	1 2 + 1 3	

Wat valt je op? Hoe komt dit?

.....

.....

Oefening 2:

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

	In de CLI!	Python's reactie
a)	1325/36	
b)	3612 + 2789	
c)	58**9	
d)	$\sqrt[5]{1048576}$	
e)	11 != 11	
f)	2 <= 100	
g)	'appendix'>'beenmerg'	Verklaring:

Opmerking: Bij de 5de-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:

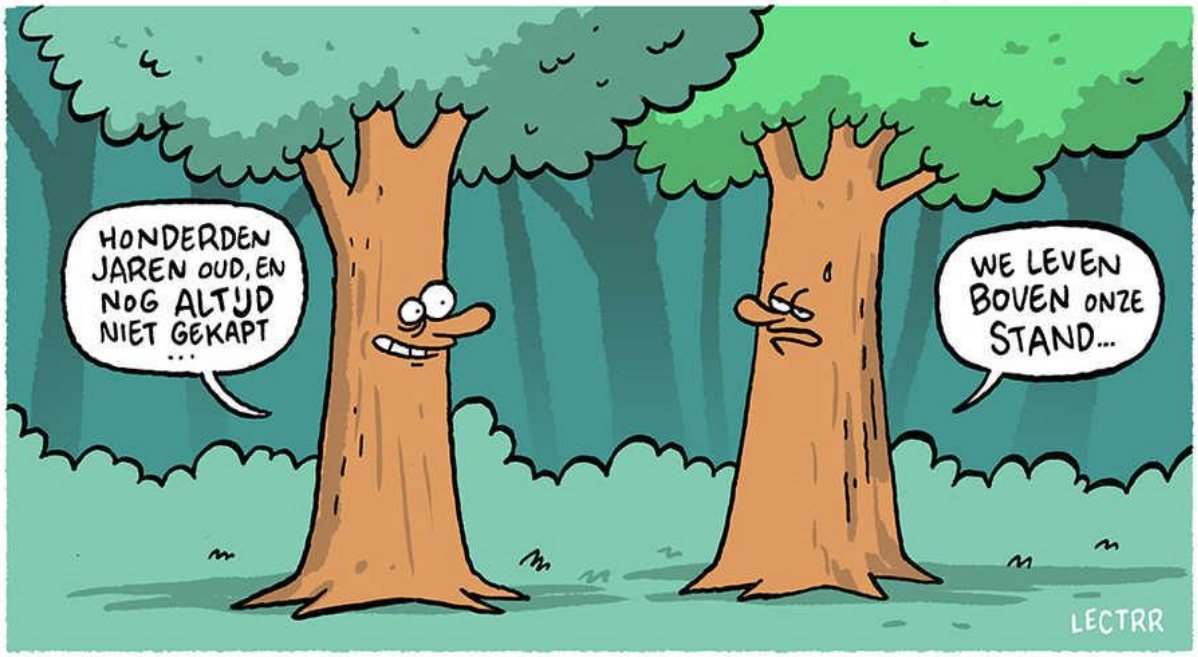
456 bomen worden verdeeld over verschillende percelen. Op ieder perceel kunnen precies 16 bomen geplant worden.

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

.....

.....

.....



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).

4.2 Basisfuncties in Python

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.

4.2.1 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:

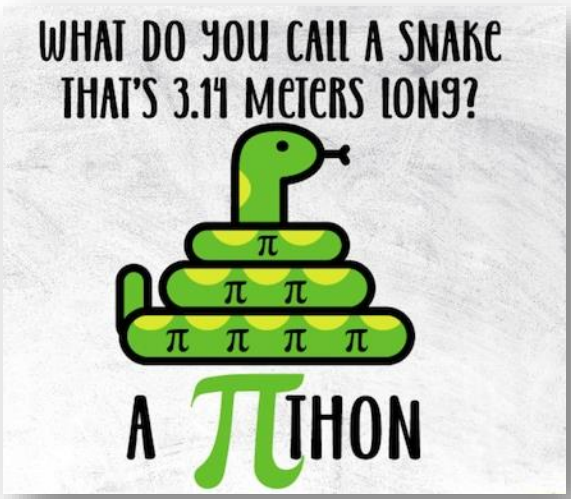
main.py Run

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

Shell Clear

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

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

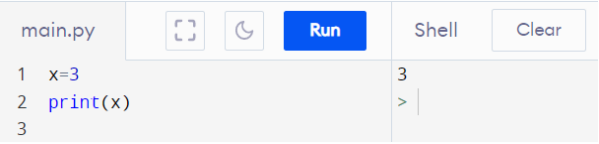


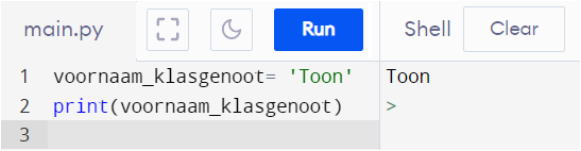
4.2.2 Variabelen

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:





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 aanhalingsteken 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/qkCpMylOYm4
De toekenning van waarden aan variabelen: https://youtu.be/QsJ3t9gXZIE

4.2.3 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. Analoog 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:

main.py | Run | Shell | Clear

```
1  voornaam = input("Wat is je voornaam?")
2  achternaam = input("Wat is je achternaam?")
3  volledige_naam = voornaam + " " + achternaam
4  print(volledige_naam)
5
```

```
Wat is je voornaam?Alexander
Wat is je achternaam?Broux
Alexander Broux
>
```

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

4.2.4 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                                   Run        Shell        Clear
1  getal_1=input("Wat is je eerste getal?")        Wat is je eerste getal?5
2  getal_2= input("Wat is je tweede getal?")       Wat is je tweede getal?8
3  som =getal_1+getal_2                            58
4  print(som)                                      >
5
```

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                                   Run        Shell        Clear
1  getal_1=input("Wat is je eerste getal?")        Wat is je eerste getal?5
2  getal_2 = input("Wat is je tweede getal?")      Wat is je tweede getal?8
3  som = int(getal_1)+int(getal_2)                 13
4  print(som)                                      >
5
```

De functie int(): https://youtu.be/1sn58AmshII
De functie float(): https://youtu.be/wgCC9nfV320

4.2.5 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                                   Run        Shell        Clear
1  rekeningsaldo = 123                             Het bedrag op je rekening bedraagt 123 euro.
2  zin = "Het bedrag op je rekening bedraagt " + str(rekeningsaldo) + ' euro.'   >
3  print(zin)
4
```

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

4.2.6 type()

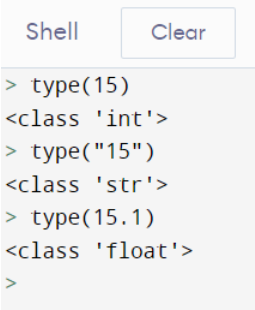
Datatypes brengen structuur en efficiëntie in de programmeercode. Ook Python gebruikt een aantal standaard datatypes zoals integer, float, string,… Het is in eerdere voorbeelden al duidelijk geworden dat “7” + “93” niet gelijk is aan 7 + 93.

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:



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.

Oefening 2:

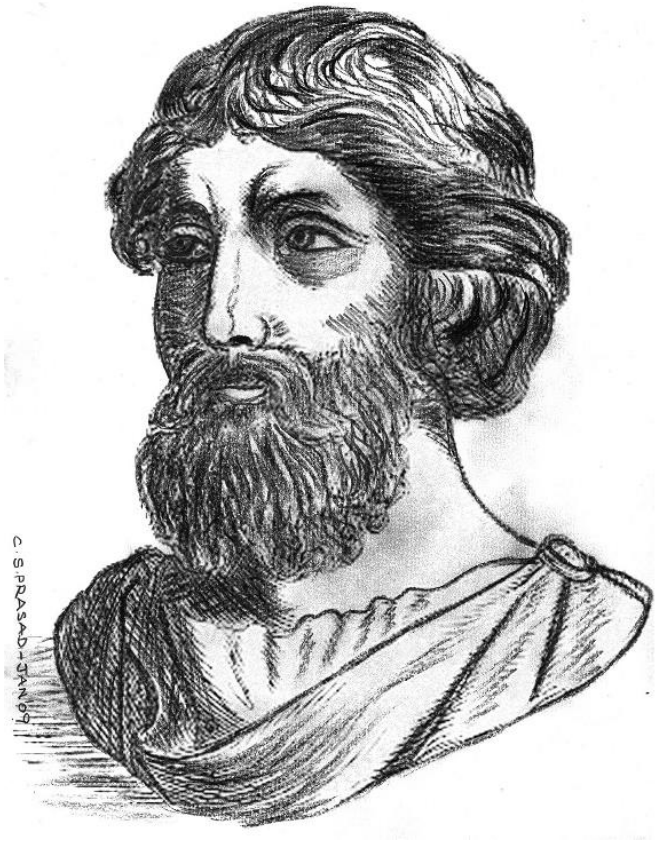
Maak een programma waarmee Python de volgende 3 vragen aan je stelt:

- 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

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.



Oefening 4:

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

Programma 1:

```
main.py                                   Run
1  print('Baby's hebben geen knieschijven')
```

.....

.....

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)
```

.....

.....

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)
```

.....

.....

Programma 4:

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

.....

.....

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.

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.

Oefening 7:

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

Programma 1:

main.py

```
aantal_kinderen=int(input("Hoeveel kinderen gaan er naar het pretpark?"))
aantal_volwassenen=int(input("Hoeveel volwassenen gaan er naar het pretpark?"))

prijs=aantal_kinderen*30+aantal_volwassenen*45

print("Je moet "+str(prijs)+" euro betalen.")
```

.....

.....

.....

.....

.....

Programma 2:

main.py

```
aantal_boeken=int(input("Hoeveel boeken koopt de winkel aan?"))

prijs_boeken=aantal_boeken*15
prijs_vervoer=1*3+(aantal_boeken-1)*0.75

totaal_prijs=prijs_boeken+prijs_vervoer

print("De winkel moet in totaal "+str(totaal_prijs)+" betalen.")
```

.....

.....

.....

.....

.....

4.3 ALS/DAN-functies

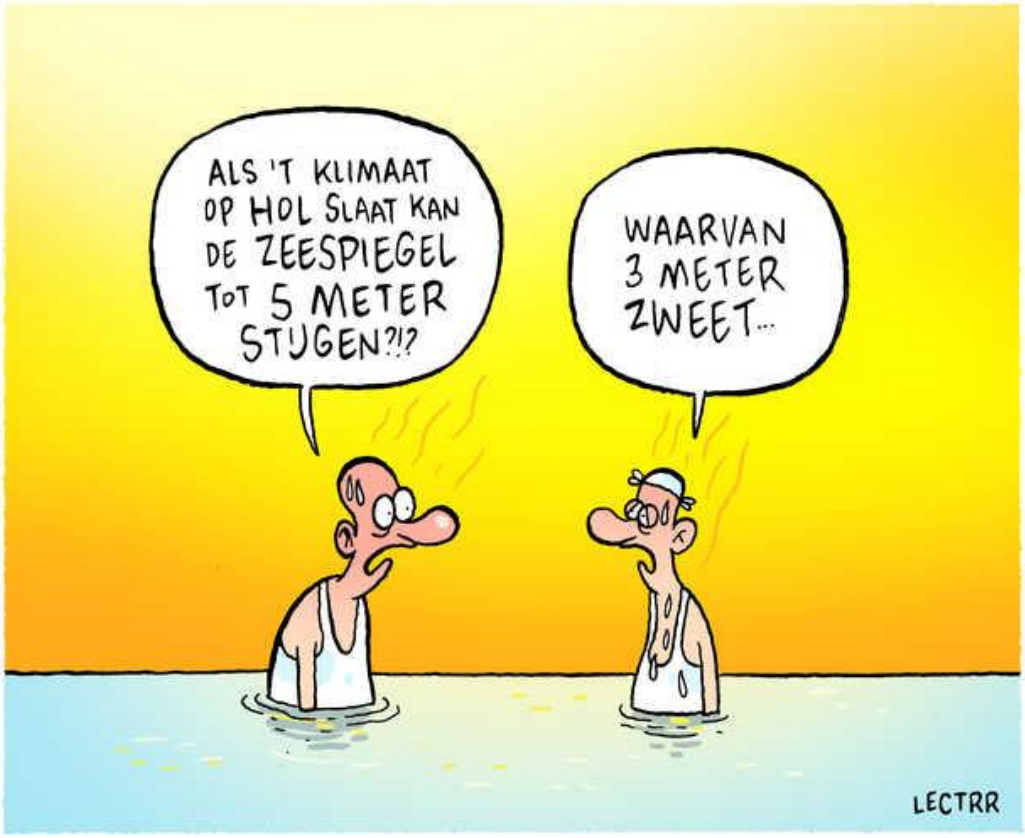
4.3.1 Booleaanse expressie

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.

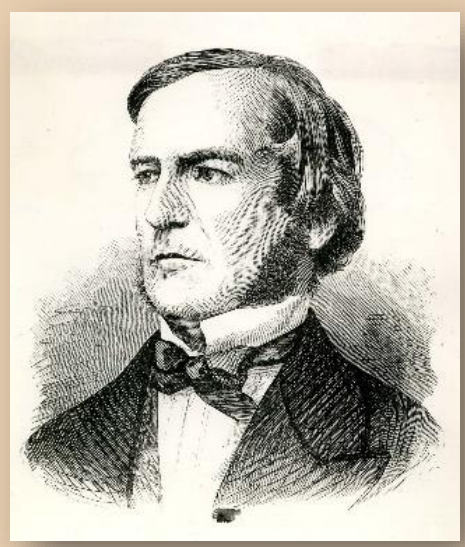
Voorbeeld: ALS $\underbrace{\text{mijn lichaamstemperatuur stijgt}}_{\text{Booleaanse expressie}}$ DAN begin ik te zweten.



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 16de 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

4.3.2 if-statement of enkelvoudige keuzestructuur

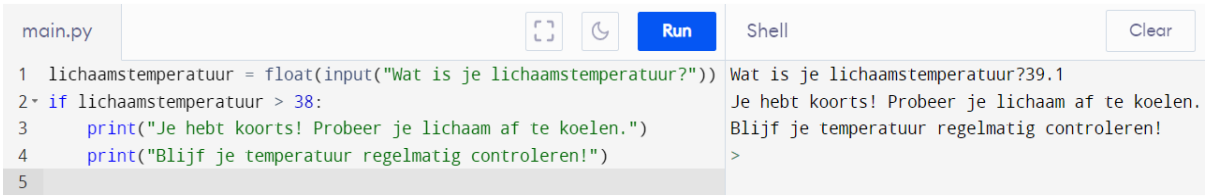
Het if-statement komt overeen met de ALS/DAN-bewering. Een statement is een stukje programmeercode dat een, door de computer uit te voeren, taak beschrijft. Bij het if-statement beschrijft men welke taken er uitgevoerd moeten worden, wanneer de booleaanse expressie evolueert naar de waarde **True**.

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.



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/lD_cYtxyyZA

4.3.3 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 | 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     print("Je temperatuur is normaal. Maak er een leuke dag van!")
6
```

```
Wat is je lichaamstemperatuur?36.7
Je temperatuur is normaal. Maak er een leuke dag van!
>
```

In codetaal:

if << booleaanse expressie >> :

<< taken >> } True

else:

<< alternatieve taken >> } False

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

4.3.4 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:

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.

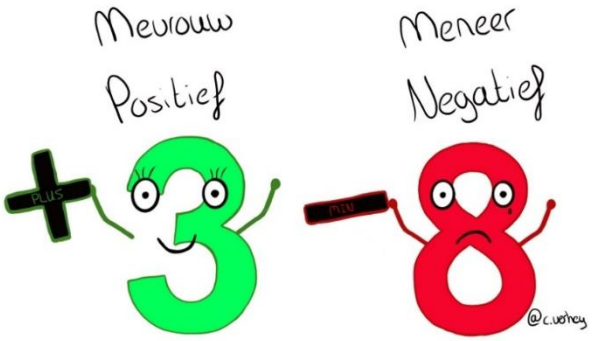
Statements

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.

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.

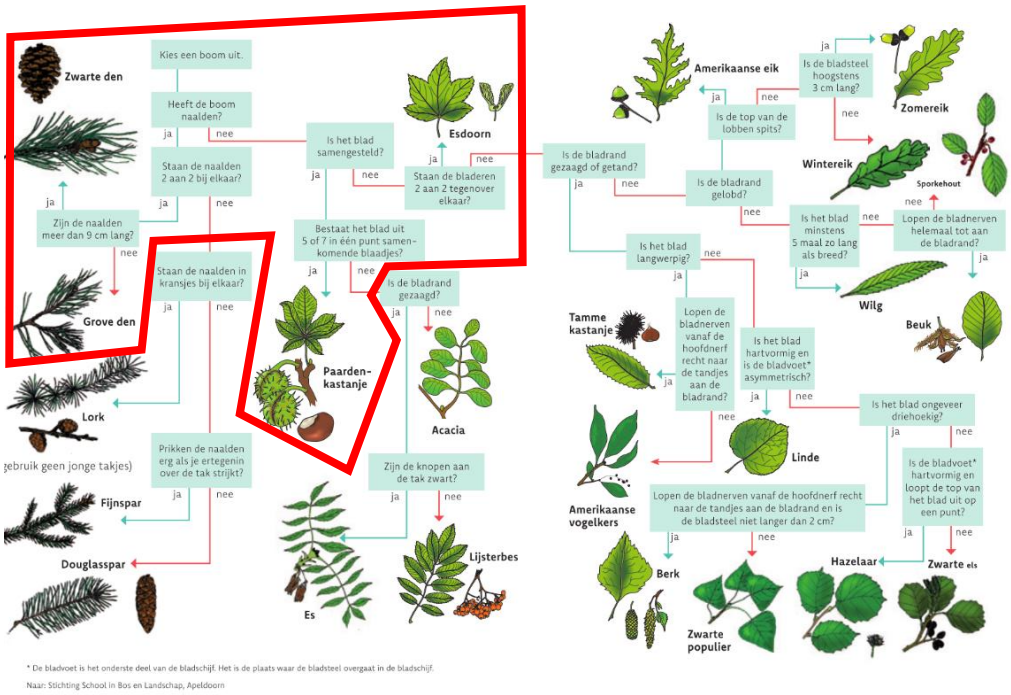
Oefening 3:

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.

Oefening 4:

Bij de oefeningen omtrent computationeel denken heb je gezien dat er determinatie-apps bestaan. In deze oefening zal je zelf een stuk van de determinatie-app moeten programmeren. Het te programmeren deel staat omkaderd. Indien nodig kan je terug gaan kijken naar pagina 36.

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



Figuur 21: Determinatietabel

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.

Oefening 6:

Schrijf een if/elif/else-keten die na het vragen van iemands geboortejaar 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 eender welk jaar. Test je programma op minstens 3 verschillende leeftijden.

Oefening 7:

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

main.py Run

```
getal=int(input("Met hoeveel personen kom je dineren?"))

if getal>8:
    print("Onze excuses, u moet nog even wachten op een tafel.")
else:
    print("Uw tafel is gereed, volgt u maar.")
```

.....

.....

.....

.....

.....

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.

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

Oefening 10:

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

Programma 1:

main.py Run

```
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
```

.....

.....

.....

.....

.....

Programma 2:

```
main.py                                                    Run
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.")
```

.....

.....

.....

.....

.....

4.3.5 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		
False		

Oefenvoorbeeld:

```
main.py                                                    Run
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.')
```

```
Shell                                                      Clear
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		
False		

Oefenvoorbeeld:

main.py

```
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
```

Shell

```
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:

Shell

```
> 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

4.3.6 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

4.3.7 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	
not(True) or False	
True and False or True	

Geef hieronder de hiërarchie van de operatoren.

1. not
2.
3.

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 volgende jaartallen:

1300	Schrikkeljaar
2016	Schrikkeljaar
1800	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.'

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

```
x=str(input("Wat is de waarde van x?"))
y=str(input("Wat is de waarde van y?"))

if x==3 and y==4
        print("x is 3")
    print("y is 4")
if x>2 end y<5:
print ("x>2")
print ("y<5")
if x<4 and y>3:
    print ("x<4")
        print("y>3")
```

Oefening 5:

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

	Resultaat (True/False)
not(True and not(False))	
not(True or False)	
not(True) or not(False)	
False or False and True or not(False) and not(True)	
True or not(True) and False or not(False)	

4.4 Lussen in Python

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.

4.4.1 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

```
# De manier waarop we herhalingen schrijven zonder lussen.
getal_1=1
if getal_1<=6:
    print(getal_1)
getal_2=2
if getal_2 <=6:
    print(getal_2)
getal_3 = 3
if getal_3<=6:
    print(getal_3)
getal_4=4
if getal_4<=6:
    print(getal_4)
getal_5=5
if getal_5<=6:
    print(getal_5)
getal_6=6
if getal_6<=6:
    print(getal_6)
getal_7=7
if getal_7<=6:
    print(getal_7)
# De manier waarbij er wel gebruik wordt gemaakt van de while-lus
getal = 1
while getal <=6:
    print(getal)
    getal = getal +1
```

voorwaarde

```
1
2
3
4
5
6

1
2
3
4
5
6
```

Oefenvoorbeeld:

main.py | Run | Shell | Clear

```
# Met dit programma kan je een aantal positieve gehele getallen optellen.
# Vanaf dat er een negatief geheel getal wordt gegeven, zal de lus eindigen.
getal = 0
som = 0
while getal >=0:
    som = som + getal
    getal = int(input("Geef een een geheel getal: "))
print(som)
```

```
Geef een een geheel getal: 5
Geef een een geheel getal: 6
Geef een een geheel getal: -2
11
>
```

De while-lus: https://youtu.be/o7iJ6aslY1Q

4.4.2 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                                                    Run        Shell   Clear
1 print('x' in 'desoxyribonucleïnezuur')                              True
2 print('a' in 'desoxyribonucleïnezuur')                              False
3 print('xy' in 'desoxyribonucleïnezuur')                             True
4 print('yx' in 'desoxyribonucleïnezuur')                             False
5                                                                   >
```

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

4.4.3 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.

SCAN ME

https://youtu.be/pzjs6XWvrP0

Voorbeeld:

main.py 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!
>
```

<u>Opdracht</u>: 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.

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.

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".

Did You Know...

1 x 8 + 1 = 9
12 x 8 + 2 = 98
123 x 8 + 3 = 987
1234 x 8 + 4 = 9876
12345 x 8 + 5 = 98765
123456 x 8 + 6 = 987654
1234567 x 8 + 7 = 9876543
12345678 x 8 + 8 = 98765432
123456789 x 8 + 9 = 987654321

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

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

Oefening 4:

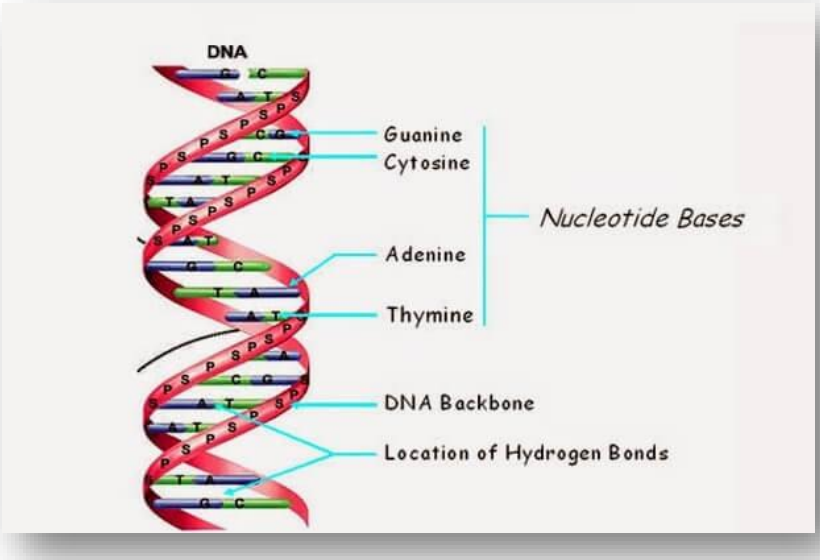
Schrijf een programma dat vraagt om twee woorden. Druk vervolgens alle letters af die de woorden gemeen hebben. Je mag hoofletters beschouwen als verschillend van kleine letters, maar iedere letter die je rapporteert, mag slechts één keer gerapporteerd worden (bijvoorbeeld: de strings "een" en "peer"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

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.



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 2de macht verheffen. Als ik onderstaande code gebruik zal ik echter niet tot mijn gewenste resultaat komen.

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

Programma 2:

Zoek de 5 fouten in dit voorbeeld.

```
main.py                    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
```

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.

4.5 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.

Oefening 2 :

Darren zijn mannelijke bijen die geboren worden uit een onbevrucht 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 7de 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.

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 krekel 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.

Bekijk deze video: https://youtu.be/36kxhO_ki8o

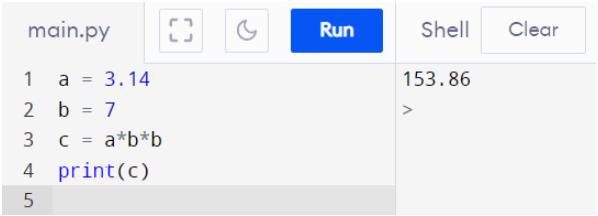
Oefening 4:

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.

Oefening 5:

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



Opgave= .....

.....

Aanpassing variabelen: a= .....

b= .....

c= .....

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.

Oefening 7:

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

main.py | Run | Shell | Clear

```python
print('Ik ben geboren in 2007')
print('Het huidige jaar is 2023')
print('Dit jaar word/ben ik 16 jaar')
print('In 2008 was ik 1 jaar')
print('In 2009 was ik 2 jaar')
print('In 2010 was ik 3 jaar')
print('In 2011 was ik 4 jaar')
print('In 2012 was ik 5 jaar')
print('In 2013 was ik 6 jaar')
print('In 2014 was ik 7 jaar')
print('In 2015 was ik 8 jaar')
print('In 2016 was ik 9 jaar')
print('In 2017 was ik 10 jaar')
print('In 2018 was ik 11 jaar')
print('In 2019 was ik 12 jaar')
print('In 2020 was ik 13 jaar')
print('In 2021 was ik 14 jaar')
print('In 2022 was ik 15 jaar')
```

```
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
>
```

Oefening 8:

Bedenk een mogelijke opgave voor dit programma.

main.py | Run

```python
age = int(input("Wat is je leeftijd? "))
film = input("Welke film wil je gaan kijken? ")

if age < 3:
    print(" Je mag gratis binnen!")
elif age < 13:
    print(" Je ticket kost €10.")
else:
    print(" Je ticket kost €15.")
```

.....

.....

.....

.....

.....

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)

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.

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.

Oefening 12:

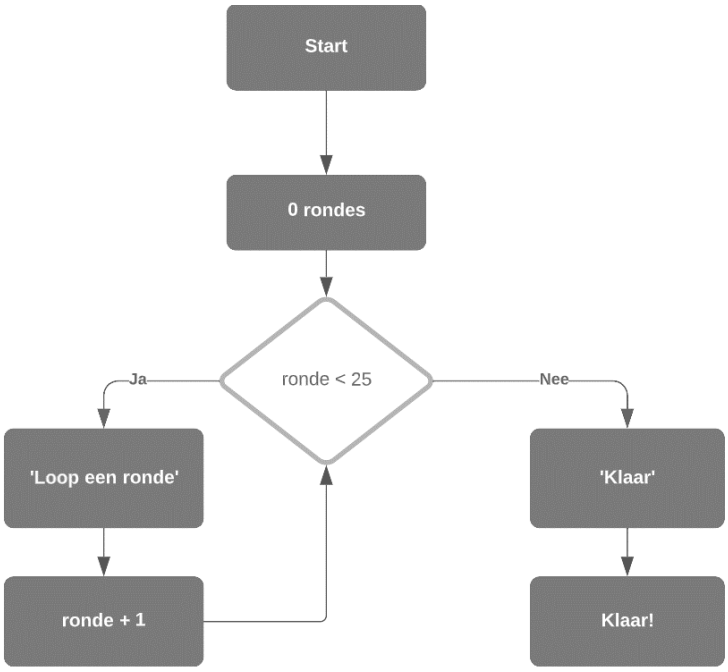
Anaïs en Jonas spelen bijna elk weekend een voetbalmatch. Om te kijken wie beter is, vergelijken ze het gemiddeld aantal keer dat ze gescoord hebben tot nu toe. Kan jij een programma schrijven waarin ze elk weekend hun scores ingeven, dat dan zegt wie het beste gespeeld heeft?

Oefening 13:

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

Oefening 14:

Gegeven onderstaande flowchart. Giet dit in een programma.



Oefening 15:

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

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

Oefening 17:

Wat doet onderstaand programma? Geef een mogelijke opgave.

main.py

```
for getal in range (137, 192, 2):
    if getal//100 + (getal%100)//10 + getal%10 ==11:
        print(getal)
```

.....

.....

.....

.....

.....

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

```
lengte_ribbe = input('Wat is de lengte van de ribbe?')
inhoud = a**3
print(inhoud)
```

.....

.....

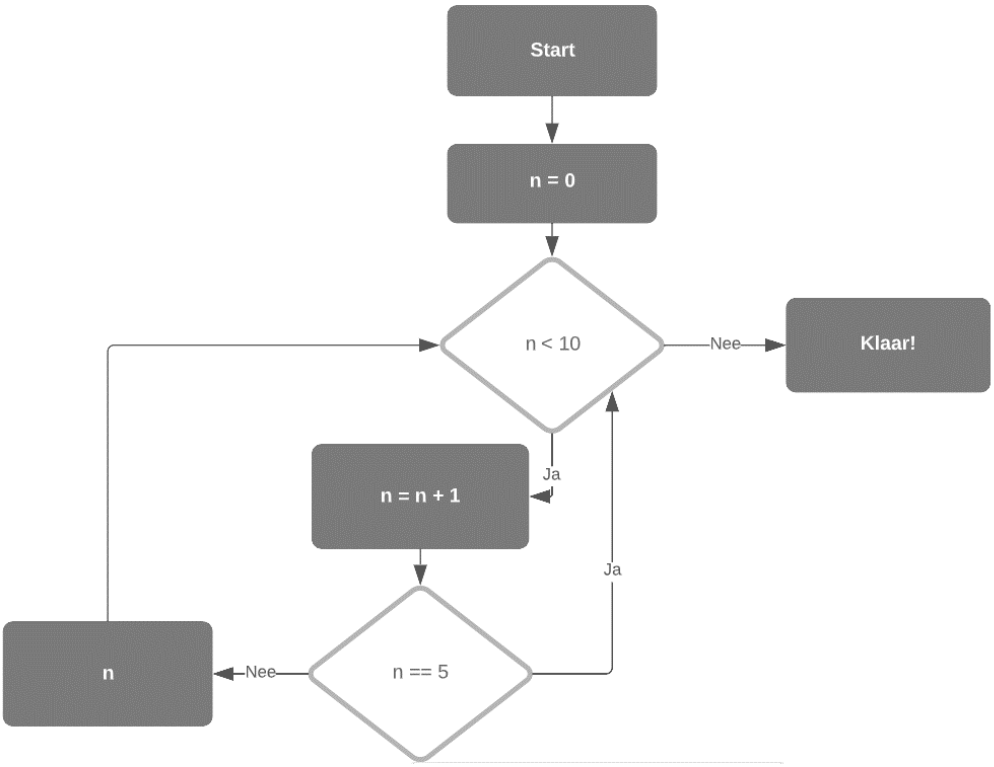
.....

.....

.....

Oefening 19:

Gegeven is een flowchart. Programmeer deze in Python.



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 4de getal vindt. De eerste 3 komen normaal wel redelijk vlot tevoorschijn.