Algoritmisch denken

Inleiding

https://www.youtube.com/embed/7p6-s_4IwOE

Samenvatting video

In de video wordt er uitgelegd dat ik een probleem heb, namelijk dat ik honger heb.
Dan bedenk ik een algoritme (een paar stappen) dat ik kan uitvoeren om tot een oplossing te komen.

Kort samengevat is dit de situatie:
Probleem: Ik heb honger (mijn maag is leeg)
Oplossing: Een gevulde maag

Het algoritme dat ik heb bedacht om van het probleem tot de oplossing te komen is het volgende:

1. Ik zoek eten
2. Ik bereid dit eten
3. Ik eet het eten op

Tijdens het uitvoeren van mijn algoritme merk ik dat ik een nieuw probleem heb, ik weet niet hoe ik het eten kan bereiden.
Gelukkig had iemand al een algoritme (een stappenplan) op de achterkant van het noedelpakje geschreven hoe ik dit kan doen.

Over dit onderdeel

In dit onderdeel zullen we zien:

- Wat algoritmes zijn
- Wat programma's zijn
- Hoe we zelf algoritmes kunnen opstellen

Algoritmes

Wat is een algoritme?

> **Wat moet je kennen en kunnen na dit deel?**
> - Weten wat een algoritme is
> - Voorbeelden kunnen geven waar we algoritmes tegen komen in het dagelijkse leven.

Bekijk de volgende video's die uitleggen wat een algoritme is:

https://www.youtube.com/embed/tnFpYaZRyTQ

https://www.youtube.com/embed/RUxT_H4uqcY

Eigenschappen algoritme

Aan de hand van de video's kunnen we de volgende eigenschappen toekennen aan een algoritme:

> **1.** Het is een eindige reeks van instructies
> **2.** Waarvan de volgorde belangrijk is
> **3.** Die je uitvoert om tot een gewenst einddoel te raken of probleem op te lossen

Hieronder bekijken we elke eigenschap wat nauwkeuriger.

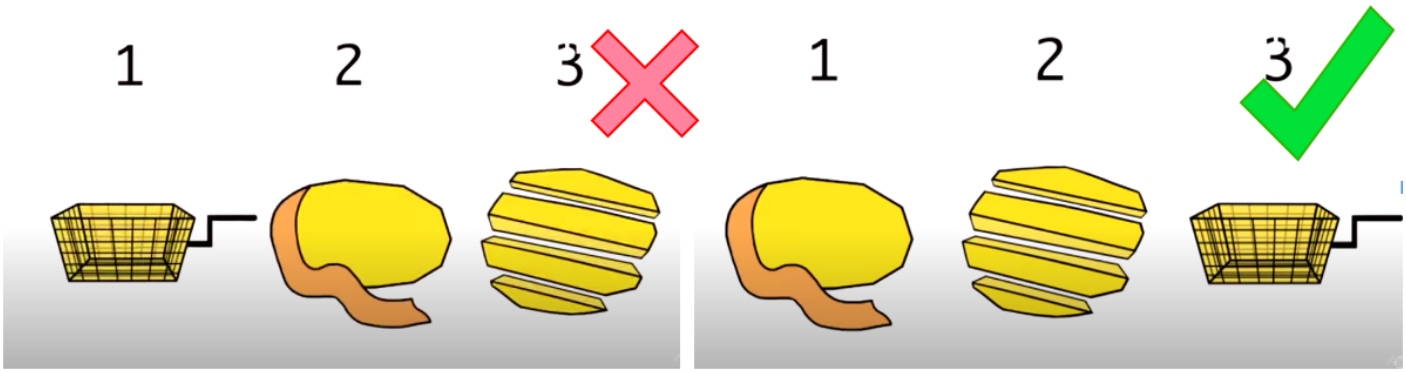
1. Het is een eindige reeks van instructies

Een algoritme bestaat dus uit een aantal instructies of stappen die je moet volgen.
Het is ook een eindige reeks instructies. Eindig betekent dat de instructies niet oneinding lang doorlopen.
Er is een bepaald punt waar je alle instructies hebt uitgevoerd.

2. Waarvan de volgorde belangrijk is

De volgorde waarin de stappen worden uitgevoerd is belangrijk!

In de eerste video wordt duidelijk gemaakt dat als je aardappelen eerst frituurt, dan schilt en dan pas in frieten snijd je geen goede frieten zult maken.



3. Die je uitvoert om tot een gewenst einddoel te raken of probleem op te lossen.

De reden dat je een algoritme uitvoert is om van een beginsituatie naar een eindsituatie te raken.

Je hebt bijvoorbeeld vuile kleren, en je wilt ze proper maken.

Beginsituatie: vuile kleren
Gewenste eindsituatie (doel): propere kleren

Je zou dan het algoritme (de stappen) van de 2e video kunnen volgen om de was te doen.
Deze stappen leggen uit hoe je van de beginsituatie naar de eindsituatie raakt.

> **Belangrijk**
> Vaak worden ook de woorden "Probleem" en "Oplossing" gebruikt in plaats van beginsituatie en eindsituatie.
> In het voorbeeld hierboven zou dit dan zo zijn:
> **Probleem**: vuile kleren
> **Oplossing**: propere kleren

Oefeningen

1. Culinair algoritme

In de eerste video wordt er gesproken over een "culinair algoritme".

Wat wordt er hiermee bedoeld denk je?
Welk ander woord kan je gebruiken om dit te beschrijven?

Als het formulier hieronder niet werkt kan je hier klikken.

2. Volgorde van de stappen

In de video hieronder wordt er nog een culinair algoritme uitgelegd.
Bekijk de stappen nauwkeurig.

https://www.youtube.com/embed/rb0o0F_7RzM

Probeer nu zelf de stappen in de juiste volgorde te zetten zoals beschreven in de video.
(Je kan de blokjes slepen in de juiste volgorde of de pijltjes achteraan elk blokje gebruiken)

Als het formulier hieronder niet werkt kan je hier klikken.

3. Welke andere algoritmes ken je?

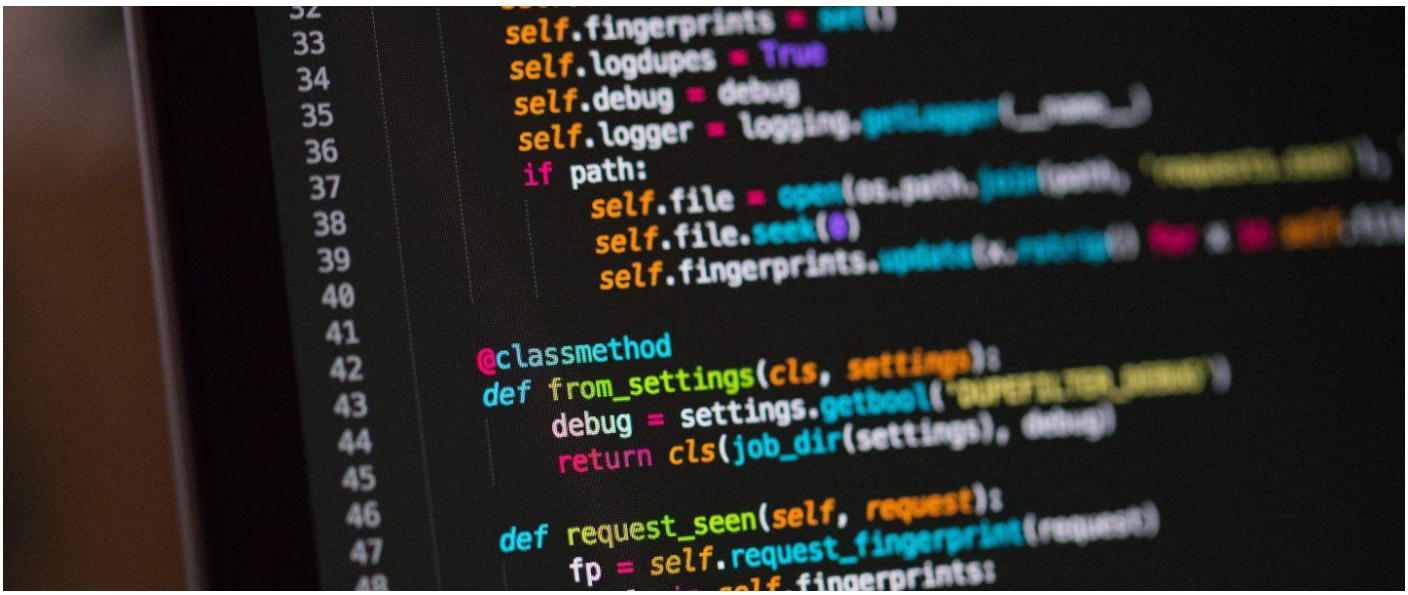
We hebben hierboven al een paar voorbeelden aangehaald van algoritmes:
- Een recept (een kookalgoritme)
- Het inladen van een wasmachine

Welke andere algoritmes ken je vanuit je dagelijkse leven?
Kies er een uit en typ dit neer in het formulier dat hieronder is gelinkt.
Schrijf dan ook kort uit wat de verschillende stappen zijn in dat algoritme.

Als het formulier hier onder niet werkt kan je hier klikken.

Wat is een programma?

Wat moet je kennen en kunnen na dit deel?
- Weten wat een (computer)programma is
- Weten wat het verschil is tussen een algoritme en een computerprogramma



Wat is een programma?

Zoals verteld in het onderdeel computer onderdelen voert de processor in een computer instructies uit. De computer kan zelf niet nadenken maar kan alleen instructies uitvoeren die hem gegeven worden.

Een computer is dus een ideaal hulpmiddel om algoritmes automatisch te laten uitvoeren want algoritmes zijn gewoon een lijst van instructies.

Computertaal

Maar computers verstaan geen gewone spreek- of schrijftaal.
Daarom moeten algoritmes eerst in een speciale computertaal of programmeertaal geschreven worden.
Op deze manier begrijpt de computer de stappen in het algoritme.

> Wanneer een algoritme is neergeschreven op een manier dat de computer dit begrijpt noemen we dit een**(computer)programma**.
> **Alle** computerprogrammas zijn geschreven op basis van een algoritme.

Verschil algoritme (computer)programma

Algoritme	**Programma**
Een algoritme is een beschrijving van hoe je een probleem kan oplossen.	Een programma is een implementatie van een algoritme. (Het algoritme is neergeschreven in een programmeertaal die de computer begrijpt)
De computer verstaat dit niet en kan dit niet uitvoeren	De computer kan dit wél uitvoeren en verstaan

Programmeertalen

Er zijn verschillende types programmeertalen met elk hun eigen doelen.

Een videospel zal bijvoorbeeld in een andere programmeertaal geschreven worden dan een website.
Bij verzekeringsmaatschappijen gebruiken ze speciale programmeertalen om statistische berekeningen te maken.

Een paar populaire programmeertalen

Naam	**Waarvoor de taal gebruikt wordt**
Java	Desktop applicaties, Games, Android apps
C++	Programma's die zeer snel moeten draaien en niet groot mogen zijn, Games, Desktop applicaties
Python	Desktop applicaties, websites, wetenschappelijke berekeningen
C#	Desktop applicaties, Websites, Games
Javascript	Voornamelijk websites
PHP	Websites
R	Statistische berekeningen
MATLAB	Statistische en wetenschappelijke berekeningen
Swift	Wordt vooral gebruikt om Apple apps te schrijven
Objective-C	Wordt vooral gebruikt om Apple apps te schrijven

Oefening

Probleemoplossend denken

Inleiding

> **Wat moet je kennen en kunnen na dit deel?**
> - Weten welke stappen er bij probleemoplossend denken gebruikt worden.
> - Weten in welke volgorde de stappen uitgevoerd worden.

Probleemoplossend denken

Algoritmisch denken wordt ook probleemoplossend denken genoemd.
Als je een probleem hebt begin je best niet willekeurig wat dingen uit te proberen om tot je doel te komen maar denk je best even na welke stappen je kan ondernemen.

Bij probleemoplossend denken gebruiken we een paar stappen om tot een goed algoritme te komen.

1. Probleemdefinitie
2. Analyse
3. Algoritme opstellen
4. Programma schrijven
5. Testen en documenteren

Elk van deze stappen zullen we nu nader bekijken in de volgende stukken.

1. Probleemdefinitie

> **Wat moet je kennen en kunnen na dit deel?**
> - Een probleem en een doel kunnen identificeren.

In deze stap probeer je eerst je probleem en je doel uit te zoeken.
Deze stap wordt soms ook de probleemstelling genoemd.

Lego voorbeeld

Probleem: Je hebt een hoop Legoblokjes maar je weet niet hoe je een kasteel kan bouwen.
Doel: Je wilt een kasteel van Legoblokjes hebben.

Oefeningen

Oefening 1

Oefening 2

Stel het is lunchpauze en je wilt je lunch eten.
Welke van deze afbeeldingen zou het probleem zijn, en welke het doel?

2. Analyse

> **Wat moet je kennen en kunnen na dit deel?**
> - Weten wat je moet doen in een analyse

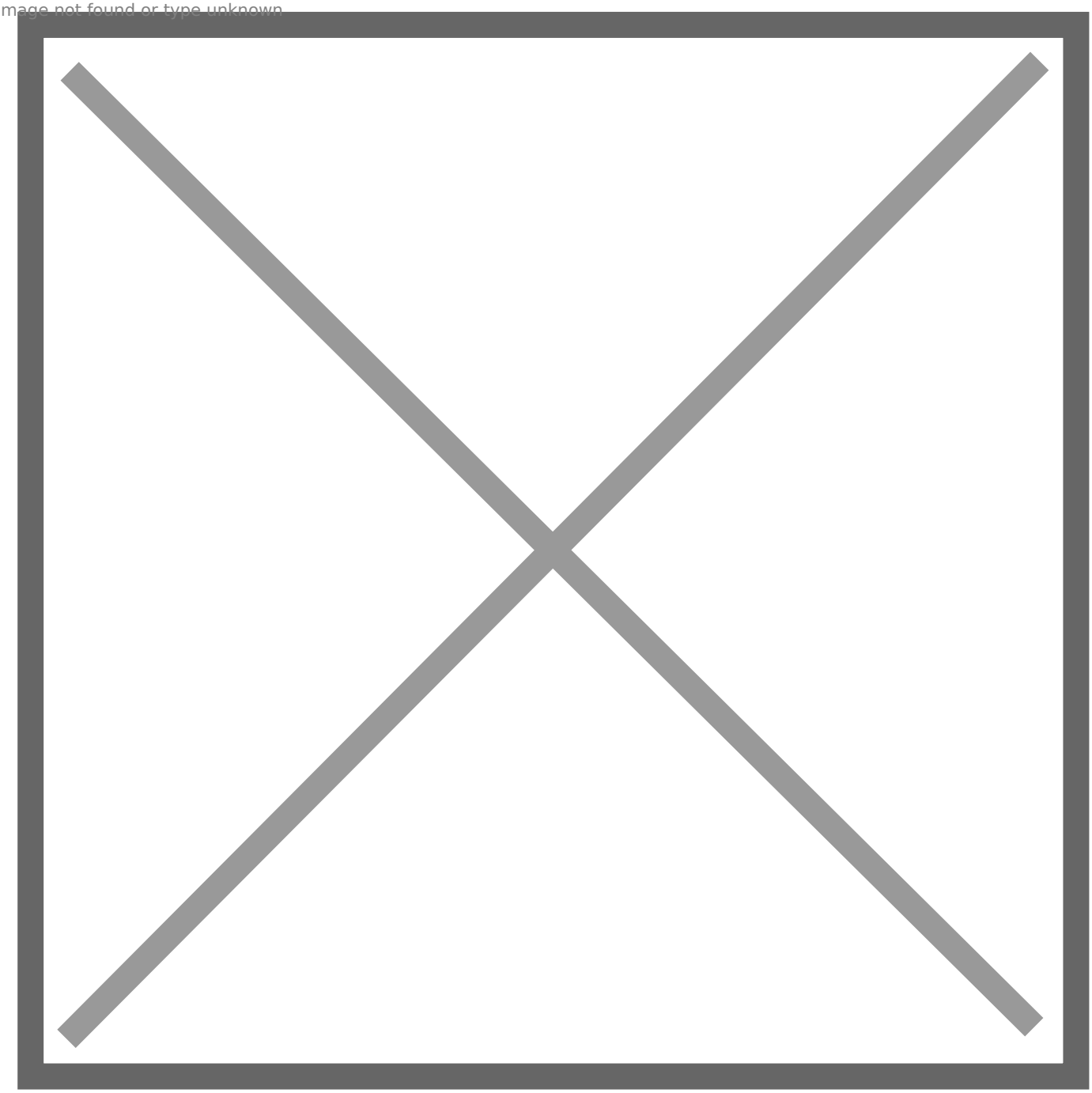
Analyse

- Wat heb je tot je beschikking?
- Welke verwerking moet er met deze beschikbare onderdelen gebeuren?

Lego voorbeeld

Ter beschikking:
We hebben witte blokken, gele blokken, grijze blokken, enz...
Mochten we zien dat we geen blauwe blokken hebben kunnen we dus later bij het opstellen van het algoritme niet zeggen “bouw een blauwe toren”, want daarvoor hebben we niet alle benodigde materiaal.

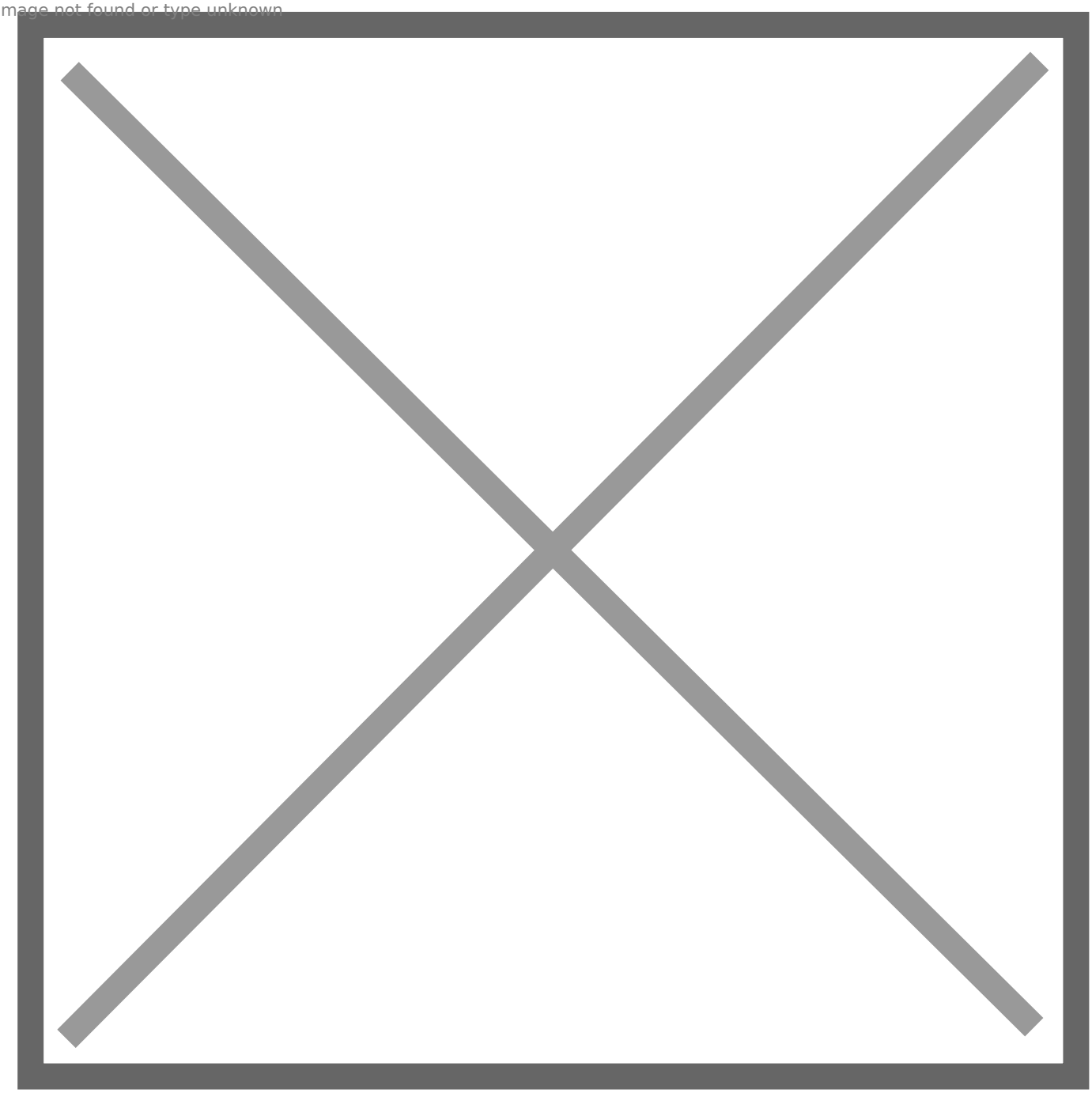
Image not found or type unknown



Welke verwerking moet er gebeuren?
De blokken moeten zo worden neergezet dat we uiteindelijk een lego kasteel hebben.

- We gaan de blokken nog niet écht neer zetten maar moeten wel ongeveer weten wat we er mee gaan doen.
  We weten dus bijvoorbeeld dat we ze niet gaan gebruiken om in het rond te gooien, maar dat we ze gaan neer zetten en op elkaar zetten.

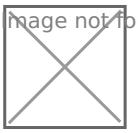
Image not found or type unknown



Kook voorbeeld

Als we bijvoorbeeld iets willen koken kunnen we in de keuken zien welke gereedschappen er zijn dat we kunnen gebruiken.

Image not found or type unknown

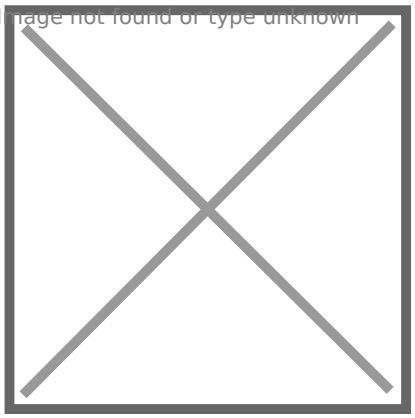


Mocht het hier blijken dat we geen deegrol hebben, dan weten we al dat het moeilijk gaat zijn om een recept op te stellen waar we een platte deeg bodem nodig hebben.

3. Algoritme opstellen

> **Wat moet je kennen en kunnen na dit deel?**
> - Weten wat een deelprobleem is
> - Zelf een simpel algoritme kunnen opstellen

Ons probleem opsplitsen in deelproblemen (decompositie)



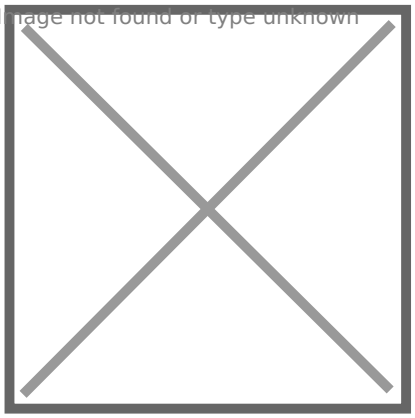
Als je probleem wat groter is dan maak je niet 1 groot algoritme maar deel je het probleem op in kleinere stukken. Deze noemen deelproblemen.
Op deze manier heb je meerder kleinere problemen die apart makkelijker op te lossen zijn dan een groot probleem in één keer.

Als je terugdenkt aan de video die in de inleiding staat (waar ik noedels maak).
Dan had ik dit algoritme ontworpen:

1. Ik zoek eten
2. Ik bereid het eten
3. Ik eet het eten op

Maar stap 2 is veel te vaag beschreven want ik wist niet hoe ik de noedels moest bereiden.
Stap 2 is dus een deelprobleem waarvoor dus een stappenplan moet worden gemaakt.

Een stappenplan (algoritme) opstellen voor elk deelprobleem



Je gaat nu voor elk deelprobleem stap voor stap uitleggen hoe je dit deelprobleem kan oplossen.
Voor stap 2 in het noedelprobleem waren dit de stappen:

1. Doe 200ml water in een pannetje
2. Doe de kruiden erbij
3. Wacht tot het water kookt
4. Voeg dan de noedels toe
5. Wacht tot het water is weggekookt

Nu weten we exact hoe we de noedels moeten bereiden.
Het volledige algoritme voor de video is dus dit:

1. Ik zoek eten
2. **Ik bereid het eten**:
    1. Doe 200ml water in een pannetje
    2. Doe de kruiden erbij
    3. Wacht tot het water kookt
    4. Voeg dan de noedels toe
    5. Wacht tot het water is weggekookt
3. Ik eet het eten op

Deelproblemen in deelproblemen

Het kan zijn dat in een deelprobleem nog andere deelproblemen zitten (En daarin nog eens enz...). Hiervoor moet dan ook een algoritme worden opgesteld.

Stel dat ik niet wist hoe ik 200 ml water in het pannetje moest doen, dan had dit het volledige algoritme kunnen zijn:

1. Ik zoek eten
2. Ik bereid het eten:
    1. **Doe 200ml water in een pannetje**:
        1. Zoek een pannetje en een maatbeker
        2. Doe water in de maatbeker tot het water even hoog staat als de 200 ml lijn.
        3. Doe dit water in het pannetje
    2. Doe de kruiden erbij
    3. Wacht tot het water kookt
    4. Voeg dan de noedels toe
    5. Wacht tot het water is weggekookt
3. Ik eet het eten op

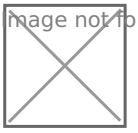
Lego voorbeeld

Deelproblemen: We gaan ons lego kasteel niet in 1 keer helemaal bouwen, maar we bouwen *onderdeel per onderdeel*.
Een deelprobleem in ons kasteel zou kunnen zijn:

- Hoe bouwen we een torentje?
- Of hoe bouwen we de poort?

Dit zijn kleinere problemen binnen ons hoofdprobleem.

Stappenplan:
Om bijvoorbeeld een torentje te bouwen zou dit ons algoritme kunnen zijn.



4. Programma schrijven

> **Wat moet je kennen en kunnen na dit deel?**
> - Weten dat een programma een vertaling is van een algoritme voor een computer.
> - Weten dat deze vertaling gebeurt aan de hand van een programmeertaal of computertaal.

Programma schrijven

Nadat je hebt bepaald welke stappen er moeten worden uitgevoerd om van je probleem tot je doel te komen kan je dit automatiseren door een computer het algoritme te laten uitvoeren.

Maar zoals gezegd in het deel “Wat is een programma?” kunnen we niet gewoon ons algoritme aan de computer geven.
Het algoritme moet eerst worden omgezet in een taal die de computer verstaat. Een programmeertaal, of *code*.

Met deze code beschrijven we elke stap in het algoritme.

Een algoritme is dus zo neergeschreven dat mensen het kunnen verstaan, en een programma is datzelfde algoritme vertaald naar een computertaal zodat een computer dat kan verstaan.

Lego voorbeeld

Voor deze robot is een speciaal programma geschreven gebaseerd op een algoritme om een lego torentje te bouwen.

Voorbeeld algoritme & code

Een voorbeeld van het algoritme en de code zou dit kunnen zijn:

Image not found or type unknown

> **Opgelet!** Dit is geen echt algoritme en geen echte code, het is maar een voorbeeld om aan te tonen hoe verschillend deze 2 er kunnen uitzien.
> Het programma is geschreven in de programmeertaal Python.

5. Testen en documenteren

Wat moet je kennen en kunnen na dit deel?
- Weten waarom je test
- Weten waarom je documenteert

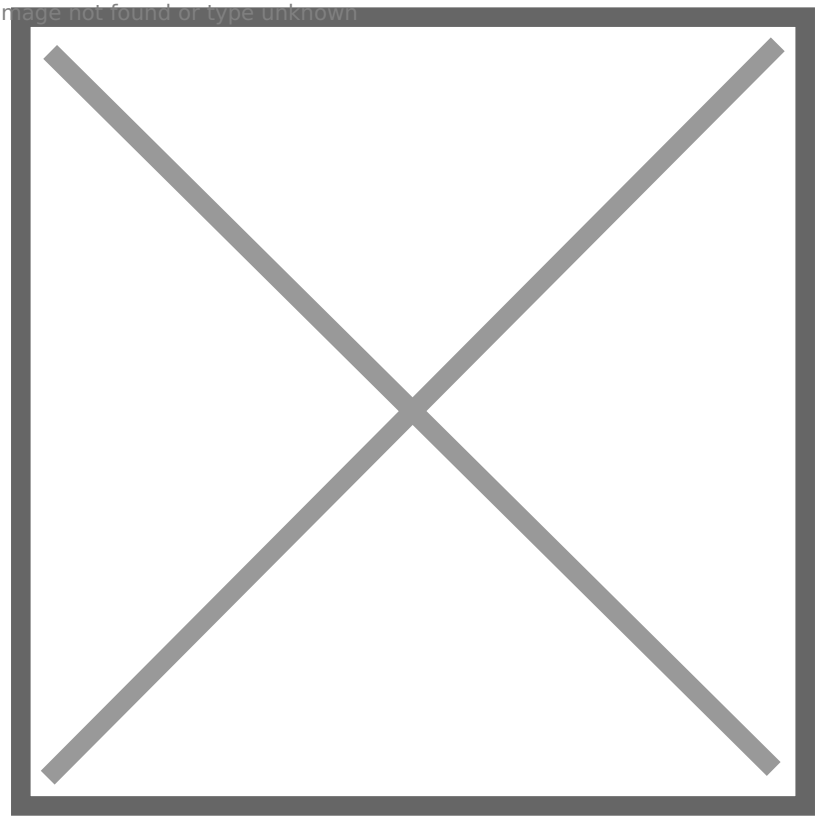
Testen

Nadat we ons programma hebben geschreven moeten we testen of het werkt.

Lego voorbeeld

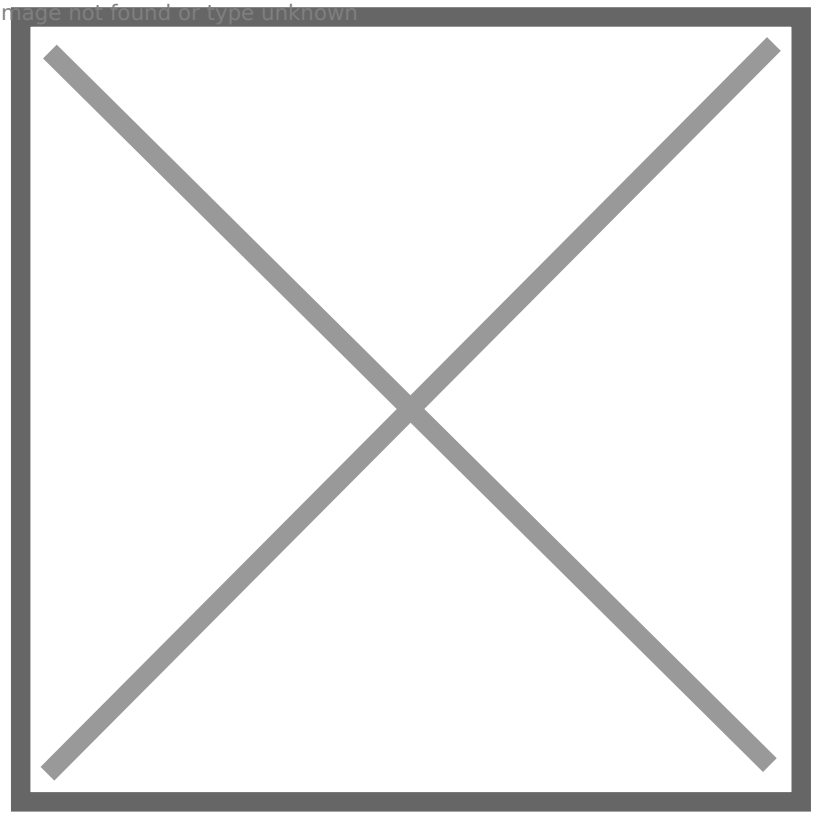
Werkt ons programma ook als we blauwe blokken gebruiken in plaats van rode? En met gele?

Image not found or type unknown



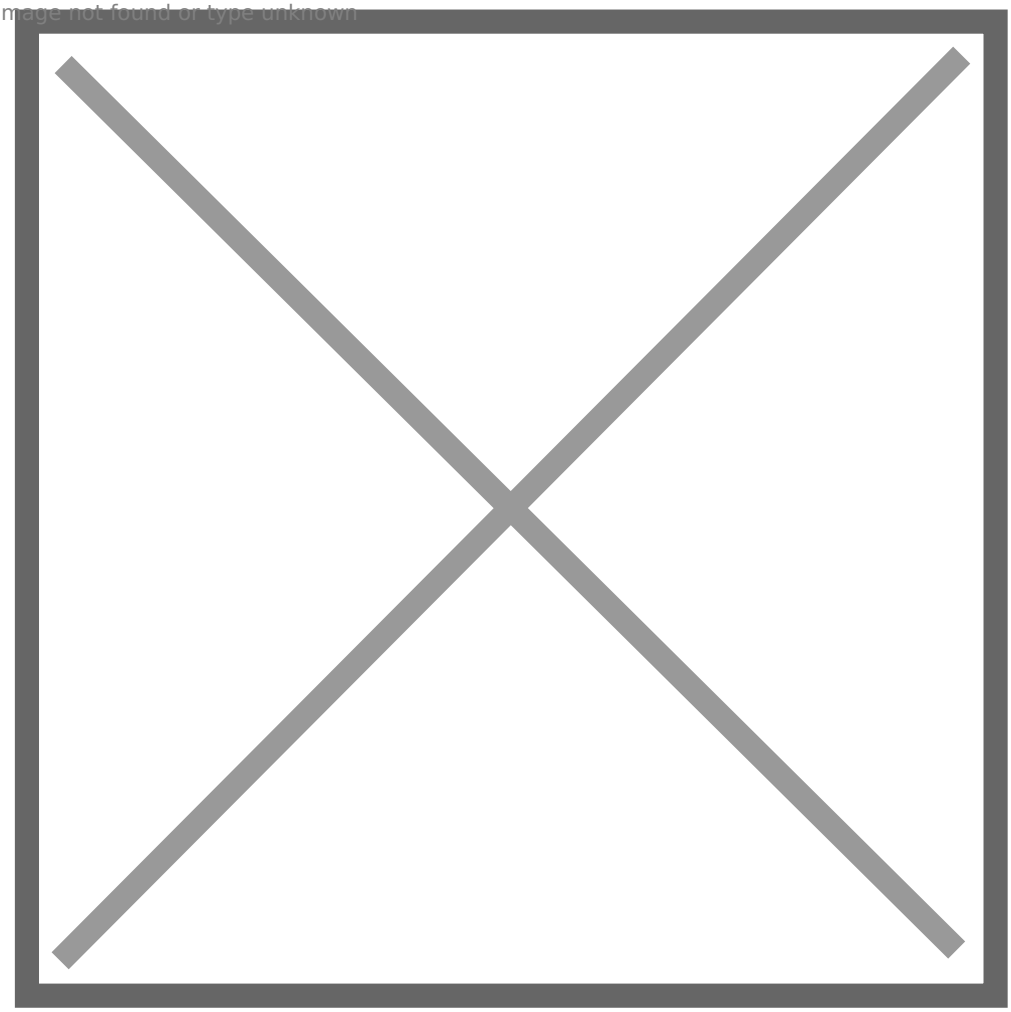
Bouwt ons programma wel een toren? Of bouwt het iets anders?

Image not found or type unknown



Kan onze robot de blokjes nog vinden als ze op deze tafel liggen?

Image not found or type unknown



Documenteren

Na het schrijven van een programma is het belanglijk dat andere mensen je code kunnen begrijpen. Daarom kan het handig zijn om documentatie schrijven die verklaart wat je code doet. Één manier om dit te doen is om commentaar te schrijven in je code.

In de code hieronder is de tekst die achter de 2 schuine strepen staat commentaar. (De tekst die in het grijs staat)
Zelfs zonder te kunnen programmeren weet je nu ongeveer wat de code doet.

```
class MainClass {
  public static void Main (string[] args) {
    var blokken = new List<LegoBlokje>();

    // Hier maak ik 10 nieuwe lego blokjes aan en steek ze in een lijst die
    "blokken" noemt.
    foreach (int i in Enumerable.Range(1, 10)) {
      var nieuwBlokje = new LegoBlokje();
      blokken.Add(nieuwBlokje);
    }
  }
}
```

Iedereen is vergeetachtig

Documenteren is ook handig voor jezelf!
Het gebeurt vaak dat als je een programma hebt geschreven, en je moet er een paar maanden later iets aan veranderen dat je helemaal geen idee meer hebt hoe je code ineen steekt!

Zelf algoritmes ontwerpen

Inleiding

> **Wat moet je kennen en kunnen na dit deel?**
> - Weten wanneer en waarom we een diagram gebruiken om een algoritme voor te stellen.
> - Weten wat een Nassi-Shneiderman diagram is
> - Weten dat een Nassi-Shneiderman diagram uit verschillende types blokken bestaan

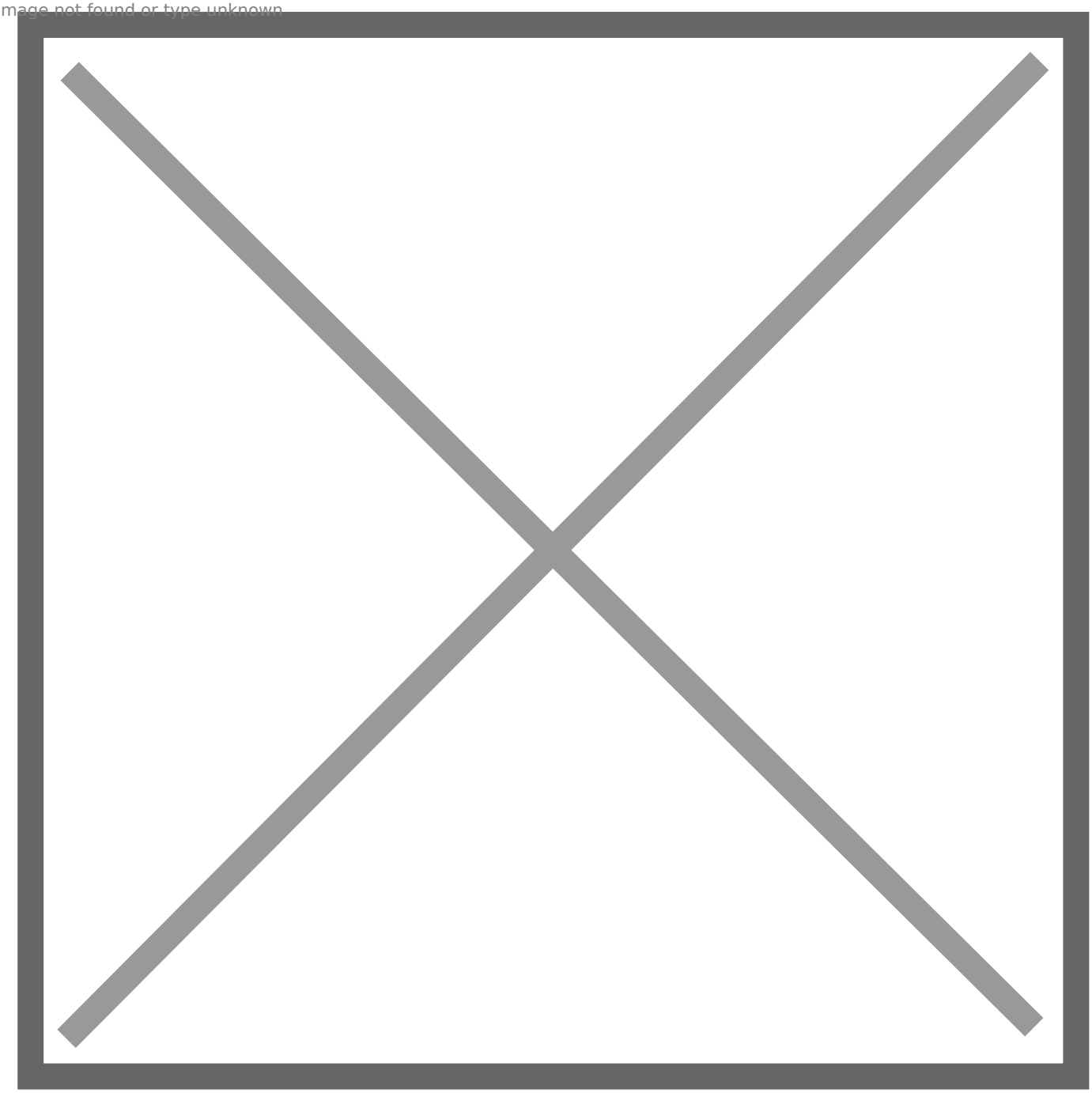
Gewone algoritmes kunnen we nog gewoon met wat tekst beschrijven. Maar vanaf dat het wat ingewikkelder gebruiken we best een gestructureerde manier om ons algoritme te beschrijven.

Door een complex algoritme als een diagram te tekenen wordt het makkelijker te verstaan.
Er zullen ook minder verwarringen voorkomen als iemand anders je diagram leest.

Nassi-Shneiderman diagram

Één zo’n type diagram is een Nassi-Shneiderman diagram.

Image not found or type unknown



> Dit is een voorbeeld van een Nassi-Shneiderman diagram waar alle mogelijke blokken gebruikt zijn.

Blokken

Een Nassi-Shneiderman diagram is samengesteld uit verschillende blokken die elk hun eigen betekenis hebben.
Deze blokken worden dan met elkaar gecombineerd om een algoritme te bouwen.

Proces blok

Image not found or type unknown

Selectie blok

Image not found or type unknown

Iteratie blok

Image not found or type unknown

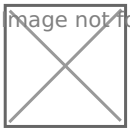
In de volgende stukken zullen we elk blokje bekijken, uitleggen wat hun doel is en hoe we ze kunnen gebruiken.

Proces blok

> **Wat moet je kennen en kunnen na dit deel?**
> - Weten wat een procesblok in een diagram betekent
> - Zelf een diagram kunnen opstellen door procesblokken te gebruiken

https://www.youtube.com/embed/UNfXIYuSyII

Het eerste, en simpelste blokje dat je kan gebruiken in een Nassi-Shneiderman diagram is het procesblok.
Dit duid eigenlijk gewoon een stap aan die moet worden uitgevoerd.



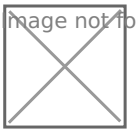
Het ziet er uit als een rechthoek met daarin tekst.
De tekst duid aan wat er moet gebeuren in deze stap.

Blokken combineren

In een algoritme is de volgorde van de stappen belangrijk.
De volgorde waarin je de blokken plaatst is dus ook belangrijk!
De blokken worden van boven naar beneden uitgevoerd.

Omelet voorbeeld

Het algoritme om een omelet te bakken uit het onderdeel “Wat is een algoritme?” kan dus zo getekend worden als een Nassi-Shneiderman diagram:



Hoe proces blokken tekenen?

Om onze diagrammen te tekenen zullen we gebruik maken van draw.io.
Hierin kunnen we digitaal diagrammen tekenen en deze opslagen als afbeeldingen op de computer.

https://www.youtube.com/embed/VXaM4bA3QKY

Oefeningen

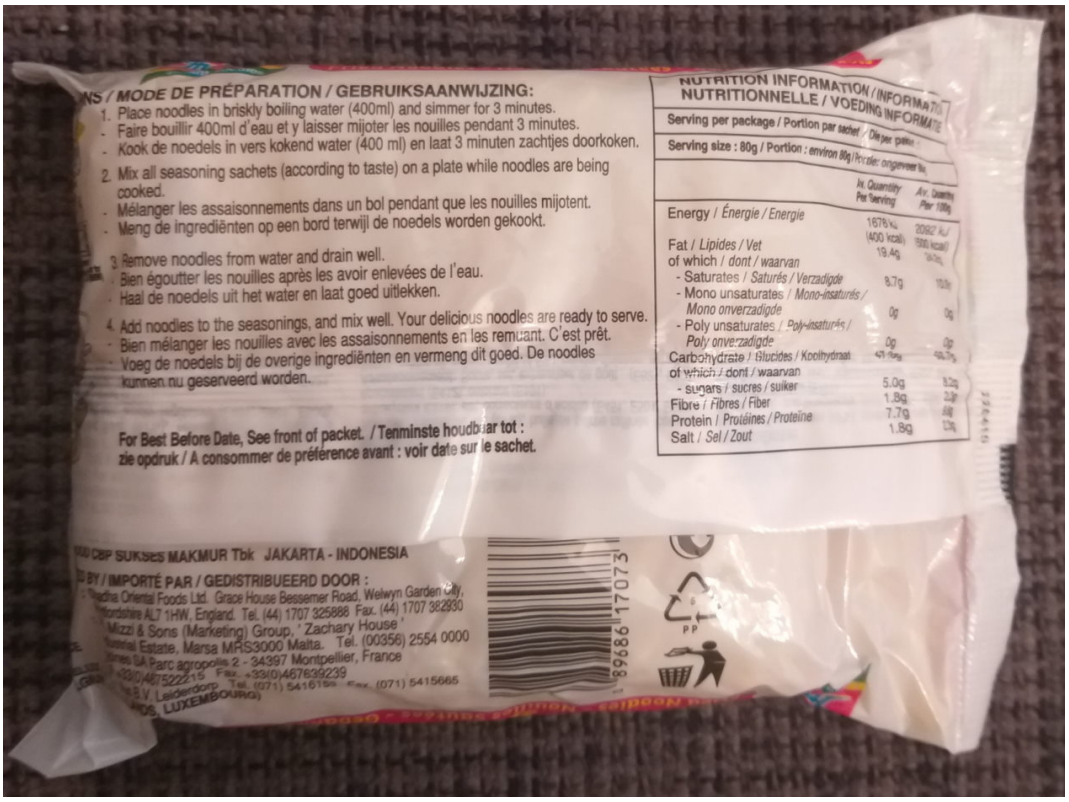
Volg bij het uitvoeren de stappen 1-3 van probleemoplossend denken.
Je schrijft dus de probleemdefinitie en de analyse uit.

Nadien ontwerp je dan je diagram in draw.io

Oefening 1. Noedels maken

Teken een NSD (Nassi–Shneiderman diagram) uit dat de stappen bevat die je moet uitvoeren om een pakket noedels klaar te maken.
De verschillende stappen kan je hieronder op de achterkant van het noedelpakket aflezen.

Gebruik alleen de nuttige tekst.
Je mag ook stappen opdelen als dit het algoritme makkelijker maakt.



Klik op de afbeelding voor een grotere versie.

Oefening 2. Geld afhalen van een bankautomaat

Beeld je in dat je iets wilt gaan kopen waarvoor je cash geld nodig hebt.
Maar je doet je portefeulle open en je ziet dat hij leeg is.
Teken een algoritme uit om dit probleem op te lossen.

Oefening 3. Een boterham met muizenstrontjes smeren

Teken een algoritme uit in een NSD dat uitlegt hoe je een boterham met muizenstrontjes (hagelslag) maakt.
In één van je stappen moet je boter smeren.

Oefening 4. Croque monsieur maken

Teken een algoritme uit in een NSD dat uitlegt hoe je een Croque monsieur (of tosti) maakt.
Een croque monsieur bevat meestal de volgende ingredienten: Brood, ham, kaas.

Oefening 5. Schoenen knopen

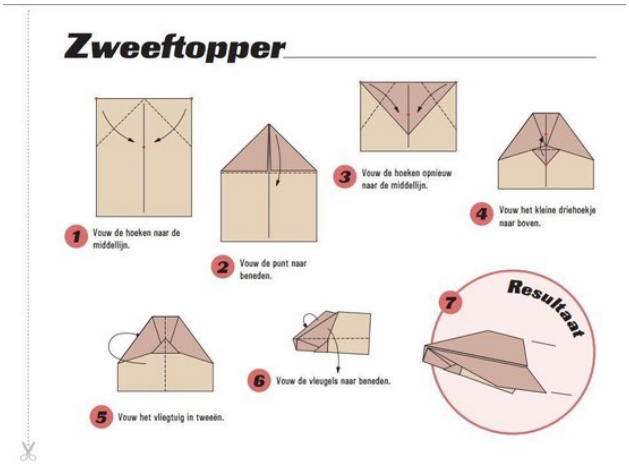
Teken een algoritme uit in een NSD dat de stappen toont om schoenveters te knopen.

https://www.youtube.com/embed/6oHVewk3dRk

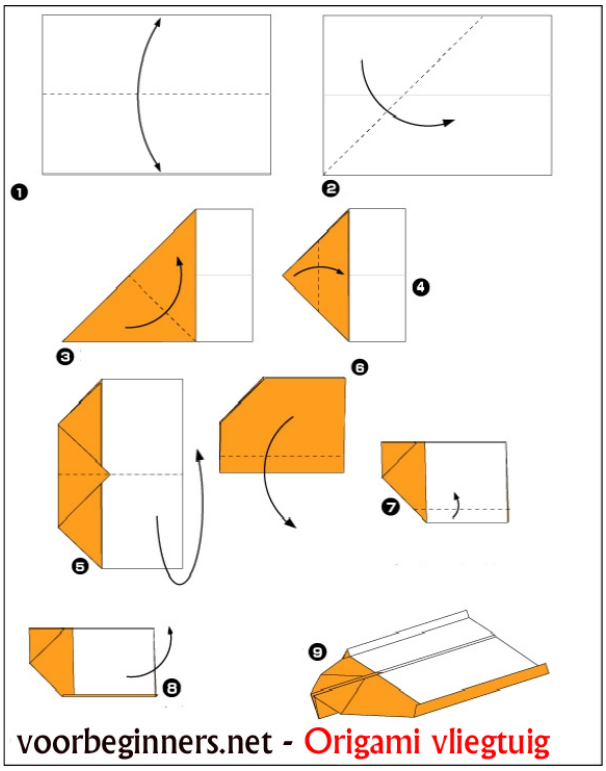
Oefening 6. Papieren vliegtuig vouwen

Teken een algoritme uit in een NSD dat de stappen toont hoe je een papieren vliegtuigje kan vouwen.
Je kan een van de stappenplannen hieronder gebruiken, of je kan je eigen favoriete vliegtuig uitleggen.

Zweeftopper



Origami vliegtuig 2



Meer vliegtuigplannen vind je hier.

Selectie blok

> **Wat moet je kennen en kunnen na dit deel?**
> - Weten wat een selectieblok in een diagram betekent
> - Zelf een diagram kunnen opstellen waarin een selectieblok gebruikt wordt.
> - Het selectieblok gebruiken in combinatie met procesblokken of andere selectieblokken

https://www.youtube.com/embed/4YSkCZsWn94

Het tweede blokje is het selectie blok.
Deze duidt aan dat er een keuze moet gemaakt worden in je algoritme.
Op basis van deze keuze ga je dan sommige dingen wel of net niet doen.
Op deze plek krijg je dan een splitsing. Je zult dan 2 kolommen van proces blokken naast elkaar zien.

Hieronder is de selectieblok in het vet aangeduid:

De selectieblok staat dus tussen andere blokken in.
De blokken worden nog steeds van boven naar beneden uitgevoerd.
In dit voorbeeld is dit de volgorde waarop de blokken worden uitgevoerd:

1. Zoek iets om te eten
2. Dan wordt de selectieblok uitgevoerd
3. Eet de boterham op

Hoe de selectieblok werkt zien we hieronder.

Selectie met 2 keuzes

De simpelste selectieblok heeft maar 2 keuzes, zoals in dit voorbeeld:

Image not found or type unknown

Groene driehoek: De keuze die gemaakt moet worden.
Blauw: De eerste keuzemogelijkheid. (Choco in dit geval)
Rood: De tweede keuzemogelijkheid. (Hagelslag in dit geval)

Als je choco kiest dan word de procesblok onder de blauwe driehoek uitgevoerd.
“Smeer de choco op het brood”.
De procesblokken onder de keuze Hagelsag worden *niet* uitgevoerd.

Maar als je hagelslag kiest dan worden de blokken onder de rode driehoek uitgevoerd.
2 blokken dus: “Smeer border op het brood” en “Strooi hagelslag over de boter”.
Maar de procesblok “Smeer de choco op het brood” wordt *niet* uitgevoerd, omdat die niet onder de rode driehoek staan.

Meer dan 2 keuzes

Een selectieblok kan ook méér dan 2 keuzes hebben.
Bekijk onderstaande video.

https://www.youtube.com/embed/gA-n_Gqd-90

Selectieblokken *in* selectieblokken

Onder de keuzes in een selectieblok kan je dus andere blokken zetten om uit te voeren.
Je kan daar *eender welk type* blok zetten.
Tot nu toe hebben we enkel maar procesblokken onder de keuzes gezet, maar je kan er ook nog eens een selectieblok onder zetten!
Zo krijgen we dus een selectieblok *in* een selectieblok.

Een voorbeeld:

Image not found or type unknown

Hier moeten we dus eerst kiezen of we choco willen of hagelslag.
Als we voor hagelslag kiezen moeten we ook nog eens kiezen of we boter willen of niet.

Deze video legt het verder uit:

https://www.youtube.com/embed/F44uiuWh2ZQ

Hoe een selectieblok tekenen

https://www.youtube.com/embed/B8c7ZxdrXlE

Oefeningen

Volg de stappen in deze oefening.
Nadien kan je onderaan deze pagina zien naar een mogelijke oplossing.

Oefening 1: Een ei koken

Teken het NSD (Nassi–Shneiderman diagram) om een ei te koken.
In je diagram moet je de keuze kunnen maken tussen een hardgekookt of zachtgekookt ei.

Ter informatie:

- Om een **zachtgekookt** ei te hebben moet een ei **4,5 minuten** koken
- Om een **hardgekookt** ei te hebben moet een ei **8 minuten** koken

Oefening 2: Thee maken

1. Diagram om een simpele thee te maken

Bedenk even de situatie waarbij iemand een tas thee bestelt in een café.
Hoe zou het NSD voor het maken van de thee er kunnen uitzien?
Teken dit diagram. (In dit diagram moet je dus nog geen keuze maken)

2. Een keuze toe voegen
In een café kan je normaal gezien eerst kiezen welke thee je juist wilt.
Bijvoorbeeld zwarte thee, fruitthee, muntthee, …
Welke stap in ons diagram zou dan veranderen?

Voeg een stap toe in je diagram waarin je kunt kiezen welke thee je wil.

3. Nog een keuze

Sommige mensen hebben graag suiker in hun thee.

Voeg een stap toe aan je diagram waarin je de keuze kan maken om wel of geen suiker toe te voegen aan de thee.

4. Nog meer keuzes

Voeg ook stappen toe in je diagram voor de volgende keuzes:

- Of je melk in je thee wilt
- Of je citroen in je thee wilt

Oefening 3: De juiste trein nemen

Als je een trein wil nemen naar ergens moet je natuurlijk zien dat je de juiste trein neemt op het juiste perron.
Maak een NSD waarin een persoon kan kiezen om naar een bepaalde stad te gaan.
Dit NSD moet deze persoon dan begeleiden naar het juiste perron.

Hieronder een voorbeeld van mogelijke steden en perrons:

ANTWERPEN-CENTRAAL - VERTREK

Vertrek	Bestemming	Trein	Spoor
Dinsdag 19 april 2022			
12:33	Paris Nord	THALYS THA 9340	23
12:37	Essen(B)	IC IC 2010	22
12:37	Bruxelles Midi	IC IC 3333	24
12:37	Poperinge	IC IC 712	2
12:39	Nivelles	R 1784	14

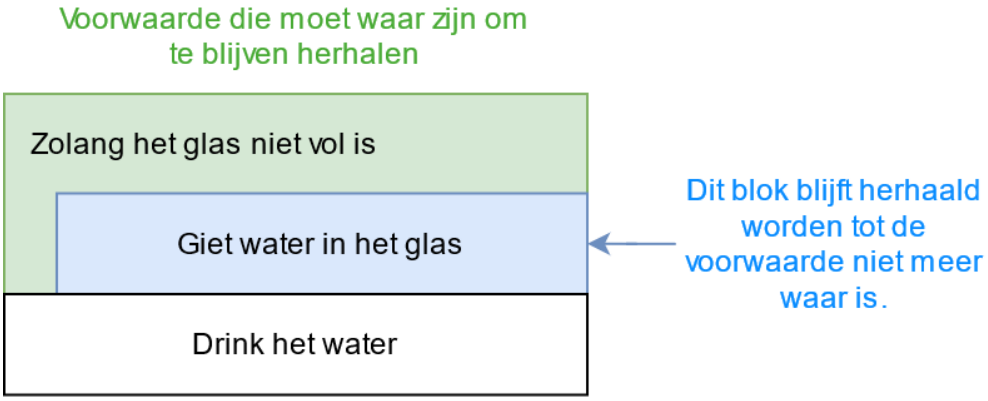
Iteratie blok

> **Wat moet je kennen en kunnen na dit deel?**
> - Weten wat een iteratieblok in een diagram betekent
> - Zelf een diagram kunnen opstellen waarin een iteratie gebruikt wordt.
> - Het iteratieblok gebruiken in combinatie met procesblokken, selectieblokken en andere iteratieblokken

Het woord iteratie betekent hetzelfde als het woord herhaling.
In de iteratie blok zullen we dus bepaalde andere blokken blijven herhalen.

Bekijk de video hieronder voor een uitleg.

https://www.youtube.com/embed/FZIcc1xNpMU



Het deel dat in groen is aangeduid hierboven bevat een voorwaarde (of controle) waarvan het resultaat waar of niet waar is.
Zolang de voorwaarde waar is zullen alle blokken rechts van de groene haak herhaald worden.
In dit geval is dat de procesblok die in het blauw is aangeduid. "Giet water in het glas".
We blijven dus water gieten zo lang het glas niet vol is.

Nadien, als de voorwaarde niet meer waar is, wanneer het glas vol is, dan drinken we het water op.
Deze laatste procesblok zit niet in het iteratieblok en wordt dus niet herhaald.

Oefeningen

Maak een tekening van het diagram voor elke oefening.
Nadien kan je onderaan deze pagina de mogelijke oplossingen bekijken.

Oefening 1: fietsband

Tijdens het fietsen krijgen we een een platte fietsband.
We repareren de band en moeten deze nadien opblazen met onze fietspomp.
Hoe zou een NSD er kunnen uitzien voor deze fietsband op te blazen?

We gaan er van uit dat onze fietspomp een drukmeter heeft en dat de bandendruk 4 bar moet zijn.

Oefening 2: autostrade

Beeld je in dat je op de autostrate aan het rijden bent naar ergens.
Je moet blijven rijden tot je aan je bestemming aankomt.
Maar als je op de autostrade een afrit tegen komt moet je wel de keuze maken of je er af rijd of niet.

Hoe zou je dit opstellen in een NSD?
Denk er aan dat je bepaalde blokken in andere blokken kunt steken.

Oefening 3: kleren sorteren

Stel, je hebt een wasmand vol met kleren die gewassen moeten worden.
Elk kledingstuk in de wasmand moet eerst gesorteerd worden in een hoop van witte kleren en gekleurde kleren.
Hoe zou je dit voorstellen in een NSD?

Oefening 4: verkeerslicht

Een verkeerslicht loopt telkens door de zelfde volgorde van kleuren.
Eerst groen, dan oranje, dan rood. (En dan weer groen)
Stel dit algoritme voor in een NSD.

Bewerk je algoritme zodat het ook de situatie behaldelt wanneer een voetganger op het knopje heeft geduwd om over te gaan.

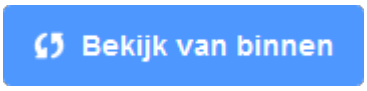
Zelf programma’s schrijven

Inleiding tot Scratch

1. Oefeningen openen & kopiëren

1.1. Een oefening openen

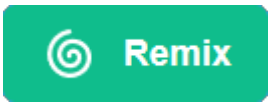
Om een oefening te openen klik je gewoon op de **Bekijk van binnen knop** die in de cursus bij de oefening staat.



De oefening wordt dan geopend in Scratch.
Deze oefening kan je niet bewerken!

1.2. Kopie maken van een oefening

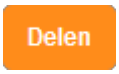
Om een oefening te kunnen bewerken moet je op de **Remix** knop bovenaan klikken.



Na op deze knop te drukken wordt er een privé kopie van de oefening gemaakt.
Deze kan je dan wel bewerken.

1.3. Delen van je oplossing

Nadat je de oefening opgelost hebt moet je deze delen.
Dan wordt de oplossing zichtbaar voor iedereen en dan kan ik deze verbeteren.
De oefening delen doe je door bovenaan op de **Delen** knop te klikken.

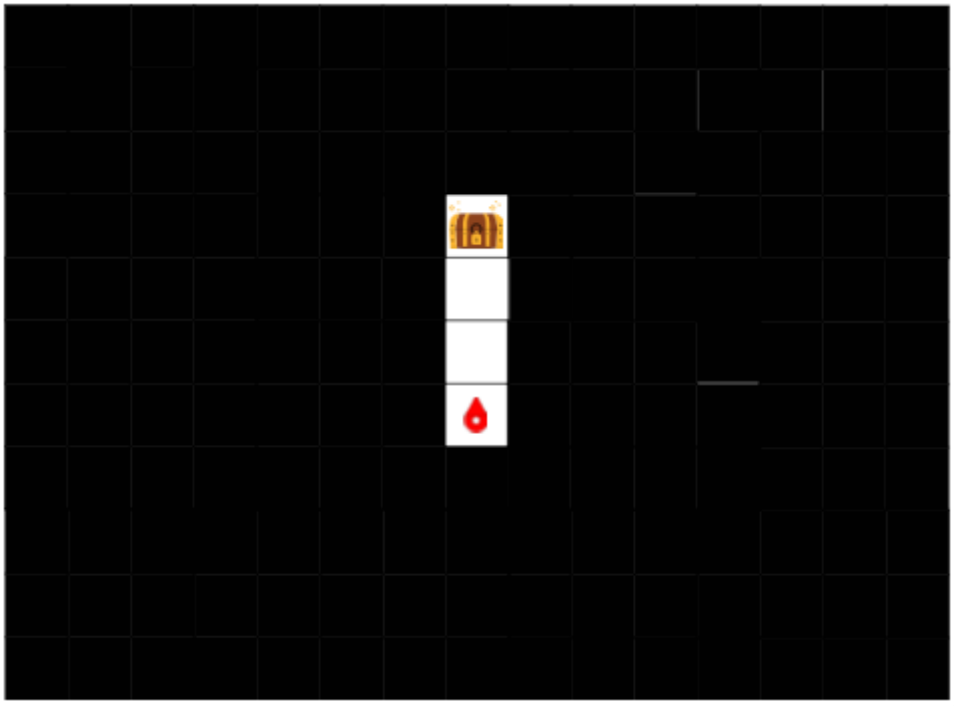


Video die de stappen bovenaan uitlegt

https://www.youtube.com/embed/uRdlhFmeo2o

2. Scratch gebruiken

Na het openen van een oefening krijg je de map van de oefening te zien.
Op deze map zijn er witte vakjes aangeduid. In deze witte vakjes staan er een paar symbolen.



Deze map heeft 4 vakjes met in het bovenste en onderste vakje een paar symbolen.

Symbolen op de map

	Dit is een schatkist. Dit is het doel in elke oefening. Hier moet je karakter op komen te staan.
	Het karakter dat je zult bewegen over de map. Dit karakter kan enkel bewegen over de witte vakken. Als je een van de zwarte vakken raakt dan zal het programma een foutmelding tonen.

Van het begin naar de schatkist raken

Om ons karakter te laten bewegen zullen we dit instructies moeten geven.
Dit kunnen we door gebruik te maken van een paar instructie blokken die het karakter zal uitvoeren.

Deze blokken vinden we onder **Mijn blokken** in de menu links.

Na daar op te klikken krijgen we de volgende blokken te zien:

	Door dit blokje te gebruiken zal het karakter 1 vak vooruit bewegen. In de richting naar waar de punt van het karakter wijst.
	Door dit blokje te gebruiken zal het karakter zijn punt 90 graden naar **links** draaien.
	Door dit blokje te gebruiken zal het karakter zijn punt 90 graden naar **rechts** draaien.

Hoe je programma uitvoeren

Na het programma te openen zie je de volgende blok staan:

Dit is de Start blok.
Onder deze blok kan je andere blokjes gaan hangen. (Deze klikken er in zoals een puzzelstuk)
Je kan deze vanuit het menu links rechtstreeks onder dit gele blokje slepen.
Hier onder kan je dus de roze blokken van uit de **Mijn blokken** menu hangen.

Nadat je je programma geschreven hebt dan kan je op deze knop klikken rechtsboven:

Scratch gaat dan zien welke blokken er onder de Start blok hangen.
En Scratch zal deze blokken dan stap per stap uitvoeren van boven naar beneden.

Video die de stappen bovenaan uitlegt

https://www.youtube.com/embed/CbKYPYT-d_s

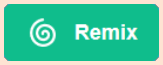
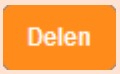
De sequentie

> **Wat moet je kennen en kunnen na dit deel?**
> - Een eenvoudig programma kunnen opstellen in Scratch om tot een bepaald doel te raken.
> - Begrijpen dat de volgorde van de uit te voeren stappen belangrijk is.
> - Begrijpen dat je een probleem op verschillende manieren kan oplossen en daar de beste manier kunnen kiezen

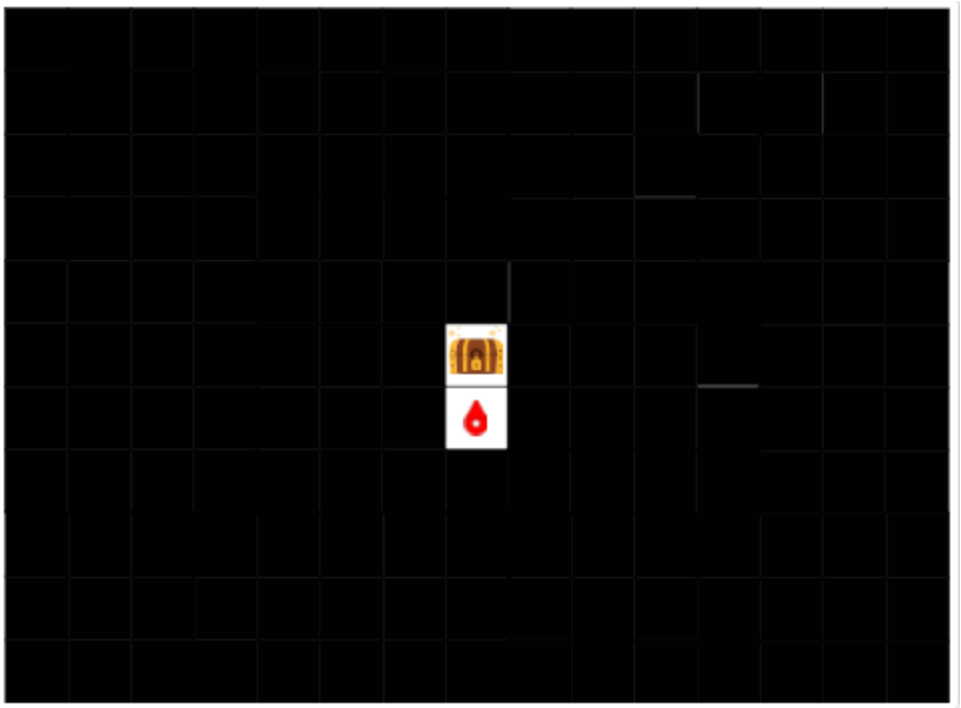
Wanneer we een programma schrijven moeten we tegen de computer zeggen wat hij juist moet doen.
Dit moeten we doen aan de hand van een algoritme (een stappenplan).

Bekijk nog eens de eigenschappen van een algoritme.

Wanneer we een algoritme opstellen is het dus belangrijk dat de stappen in de juiste volgorde staan.
Dit is belangrijk om om te letten in de volgende oefeningen.
De volgorde van de stappen moet juist zijn want anders zal je niet tot de juiste oplossing komen in deze oefeningen.

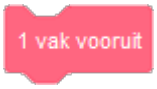
> **Belangrijk!**
> Druk na het openen van een oefening altijd eerst op de 
>
> knop om de oefening te kopiëren.
>
> En als je klaar bent klik je op de 
>
> knop bovenaan.

Sequentie 1

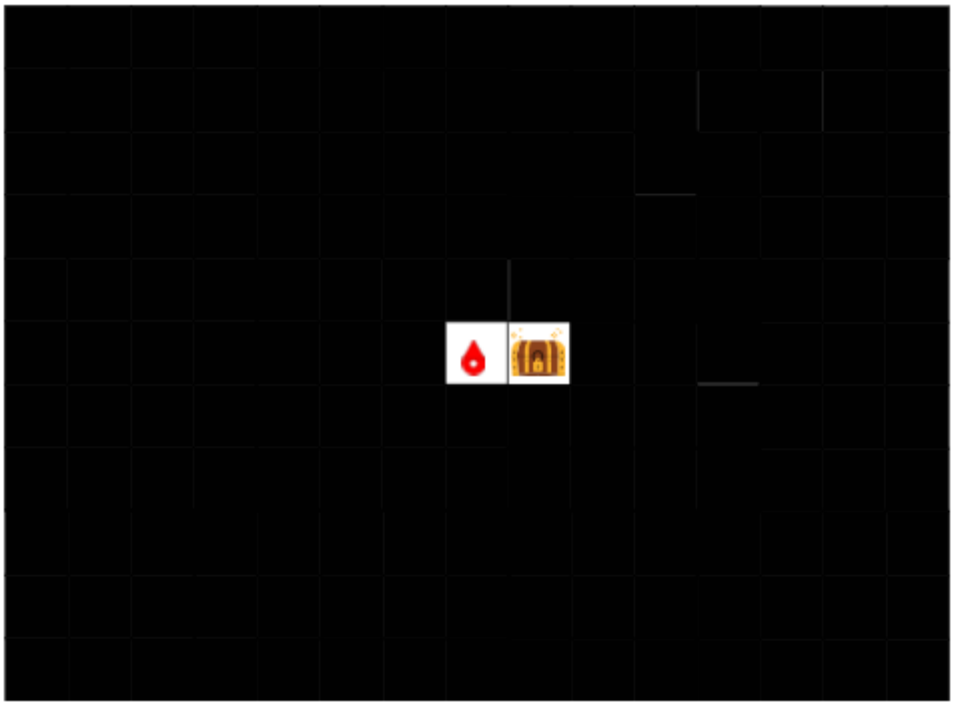


Bekijk van binnen

In deze oefening moet je maar 1 blokje gebruiken.
Het blokje om vooruit te stappen:

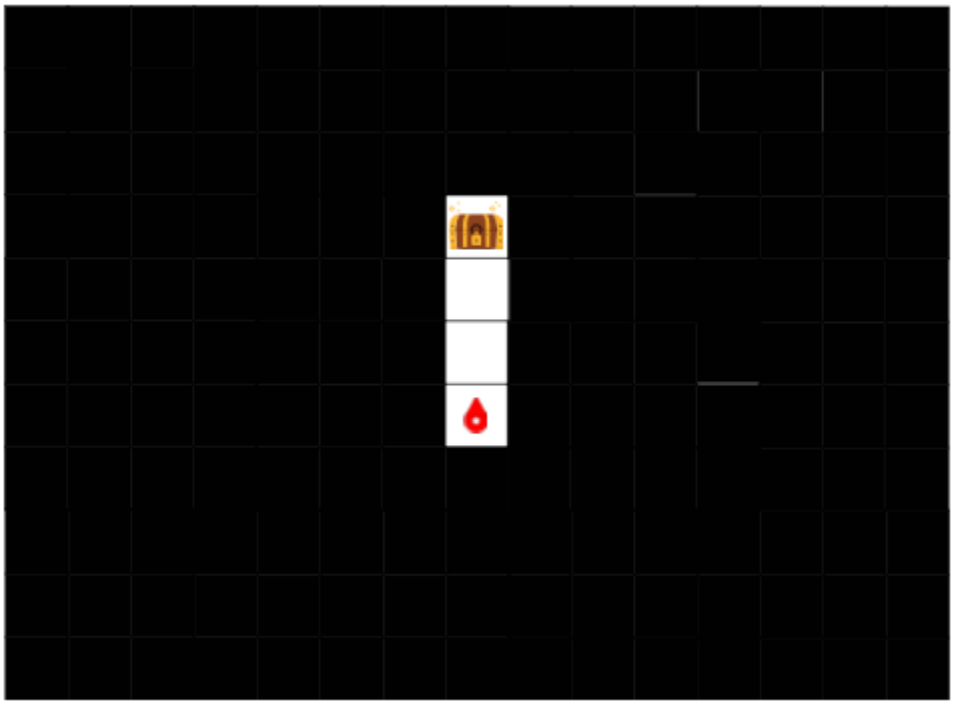


Sequentie 2



Bekijk van binnen

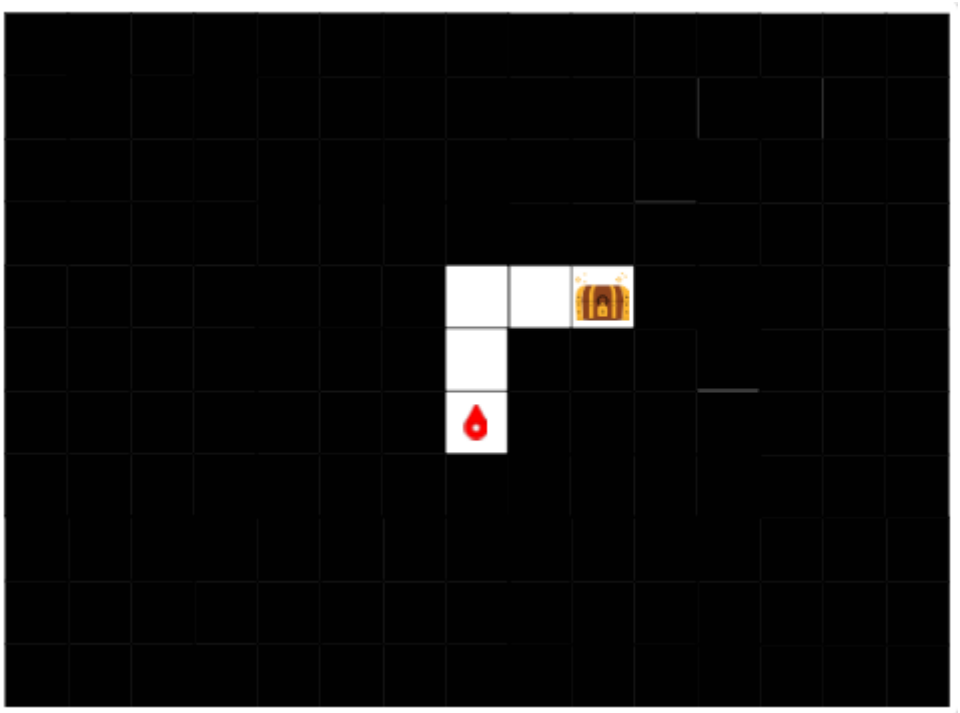
Sequentie 3



Bekijk van binnen

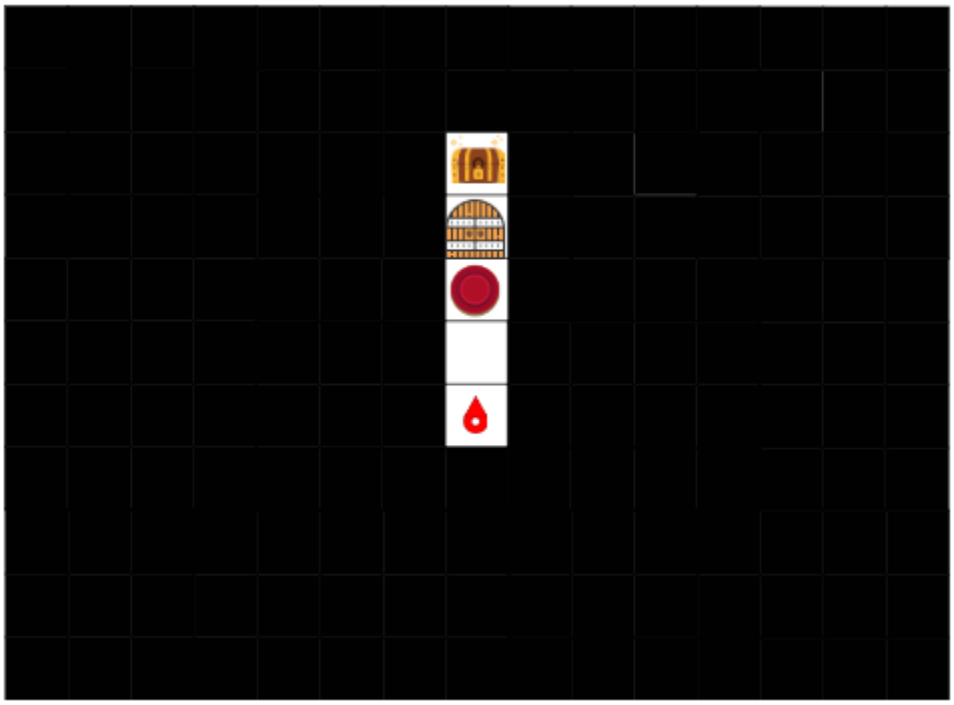
In deze oefening heb je ook een van de volgende blokjes nodig:

Sequentie 4



Bekijk van binnen

Sequentie 5



Bekijk van binnen

In deze oefening zijn er 2 nieuwe symbolen op de map en is er ook 1 nieuw blokje.

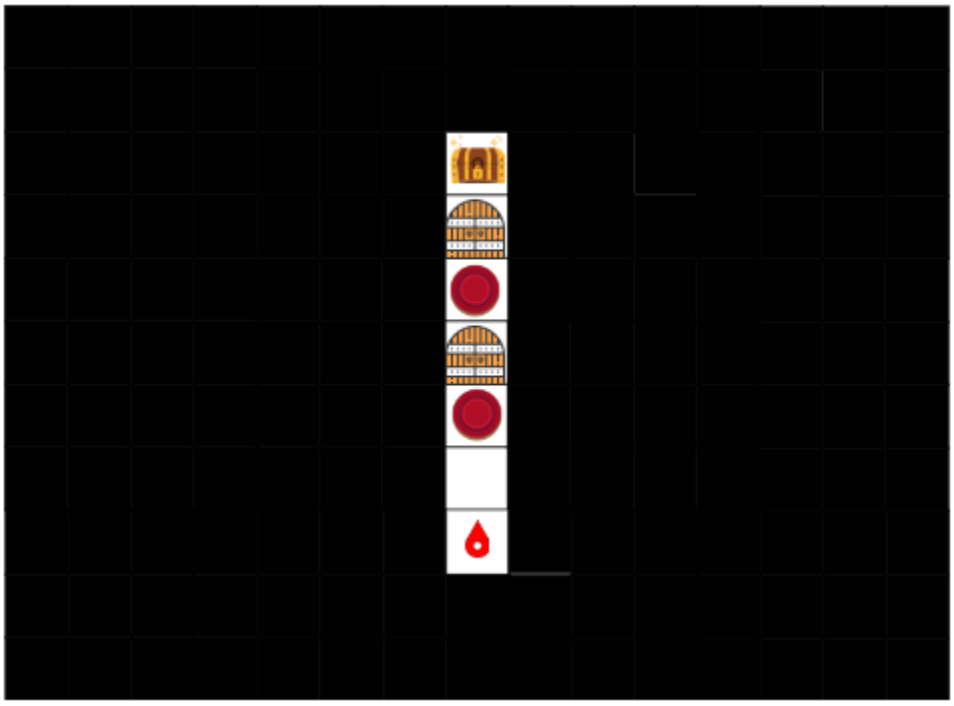
Nieuwe mapsymbolen

	Dit symbool duid een poort aan op de map. Hier kan je niet door. Als je het blokje **1 vak vooruit** zou gebruiken hier dan zal je programma een foutmelding tonen.
	Dit symbool duid een een knop aan. Als je op deze knop drukt dan zal de poort verdwijnen.

Nieuwe blokjes

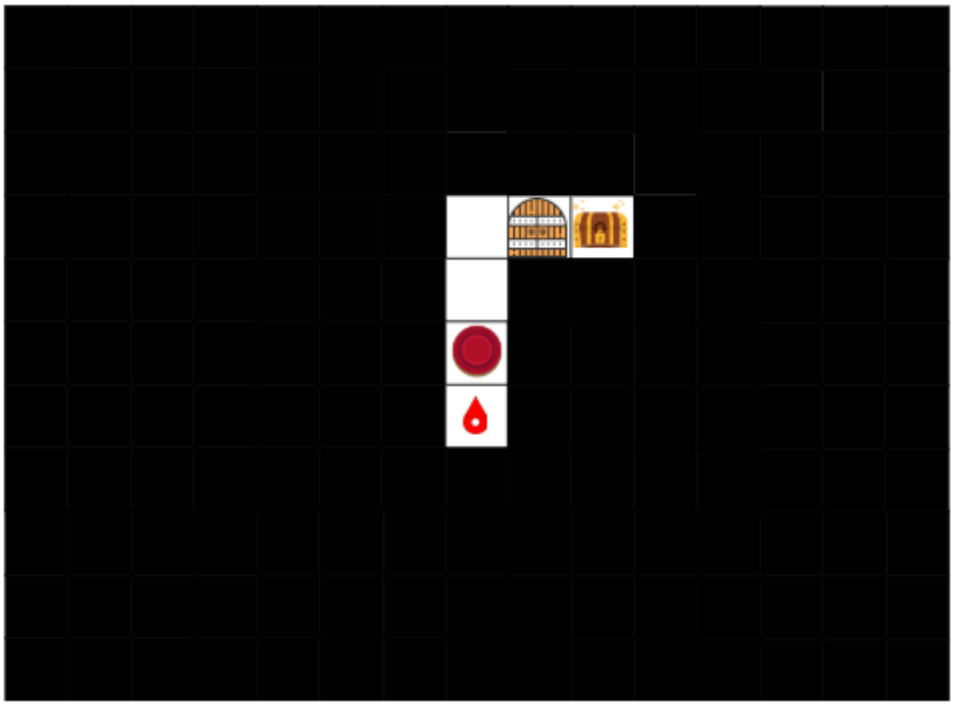
	Als het rode karakter op de knop staat en je gebruikt dit blokje, dan zal de knop worden ingeduwd. Als de knop is ingeduwd dan zal de poort verdwijnen. Dan kan je dus doorlopen.

Sequentie 6



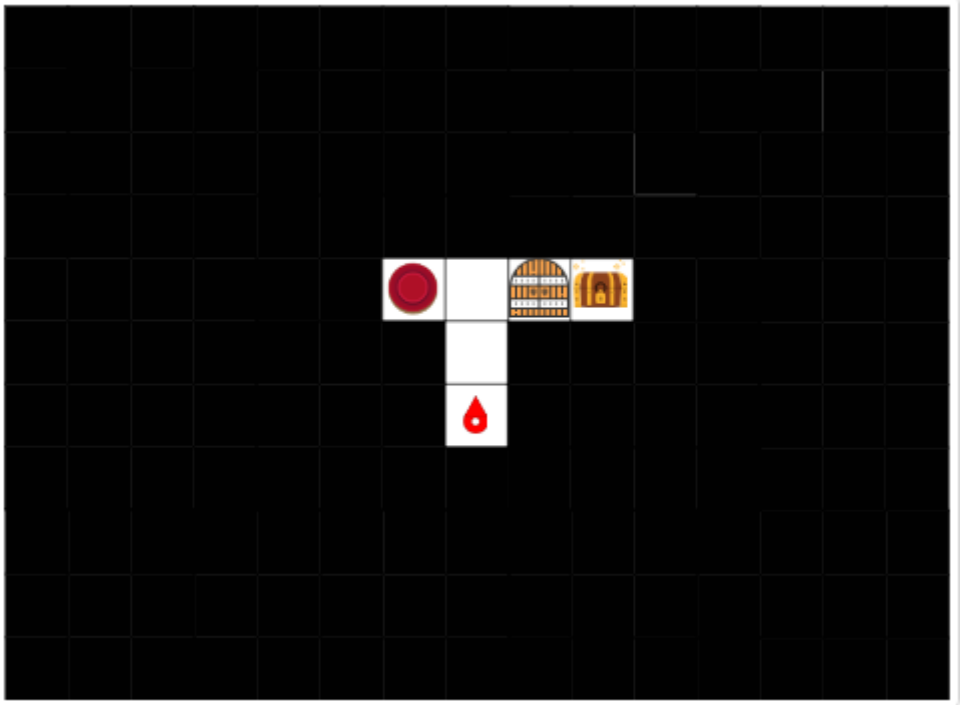
Bekijk van binnen

Sequentie 7



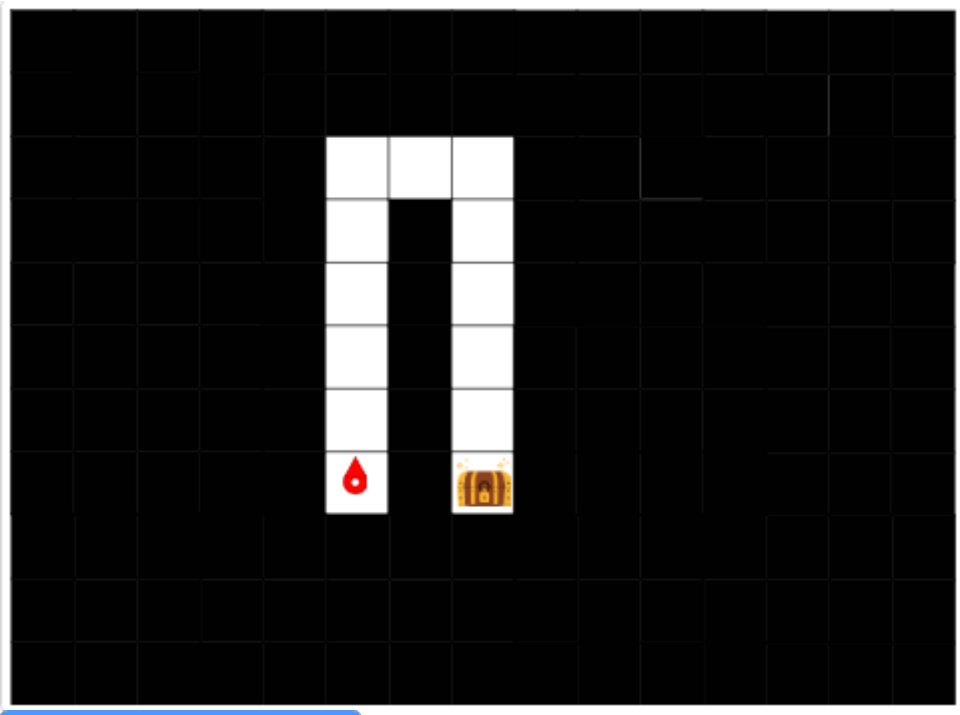
Bekijk van binnen

Sequentie 8



Bekijk van binnen

Sequentie 9



Bekijk van binnen

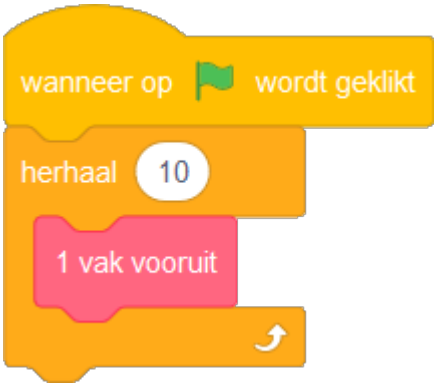
In deze oefening heb je veel blokjes nodig..
In het volgende deel, Iteraties, zien we hoe dit makkelijker kan met minder blokjes.

De iteratie

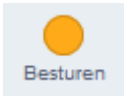
Wat moet je kennen en kunnen na dit deel?
- Patronen kunnen vinden in een probleem
- Een herhaling kunnen gebruiken in een programma
- Je programma zo kort (en simpel) mogelijk maken door het gebruik van een herhaling of een geneste herhaling.

In deze oefeningen zullen we bepaalde stappen meerdere keren moeten uitvoeren.
Als we bijvoorbeeld 10 keer een stap vooruit moeten zetten zou het veel werk zijn om 10 keer een blokje **1 vak vooruit** te zetten.
Het zou makkelijker zijn als we gewoon zouden kunnen zeggen "**Herhaal 10 keer** het blokje **1 vak vooruit**".

Dat kunnen we doen met het **Herhaal** blokje:
Nu wordt het blokje **1 vak vooruit** 10 keer uitgevoerd.
Het is dus alsof er 10 keer na elkaar het blokje **1 vak vooruit** staat.



Dit blokje kan je links vinden in de sectie "Besturen".

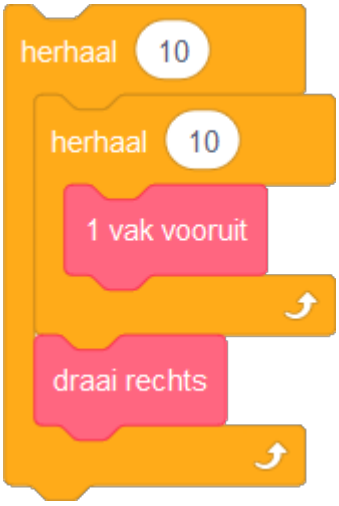


Binnenin het **Herhaal** blokje kan je méérdere blokjes zetten.

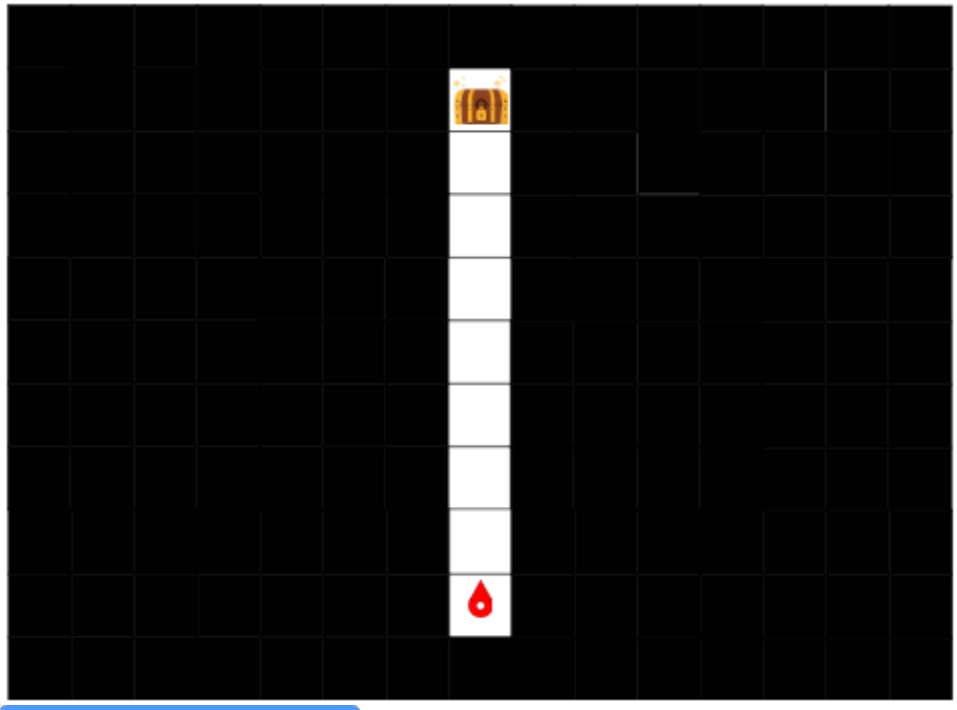
Nu wordt er 10 keer na elkaar de blokjes **1 vak vooruit** en **draai links** uitgevoerd.

Geneste blokken

Je kan het **Herhaal** blokje ook binnenin een ander **Herhaal** blok zetten. Dit noemen ze blokken "nesten".
Dit ga je nodig hebben vanaf oefening 6.

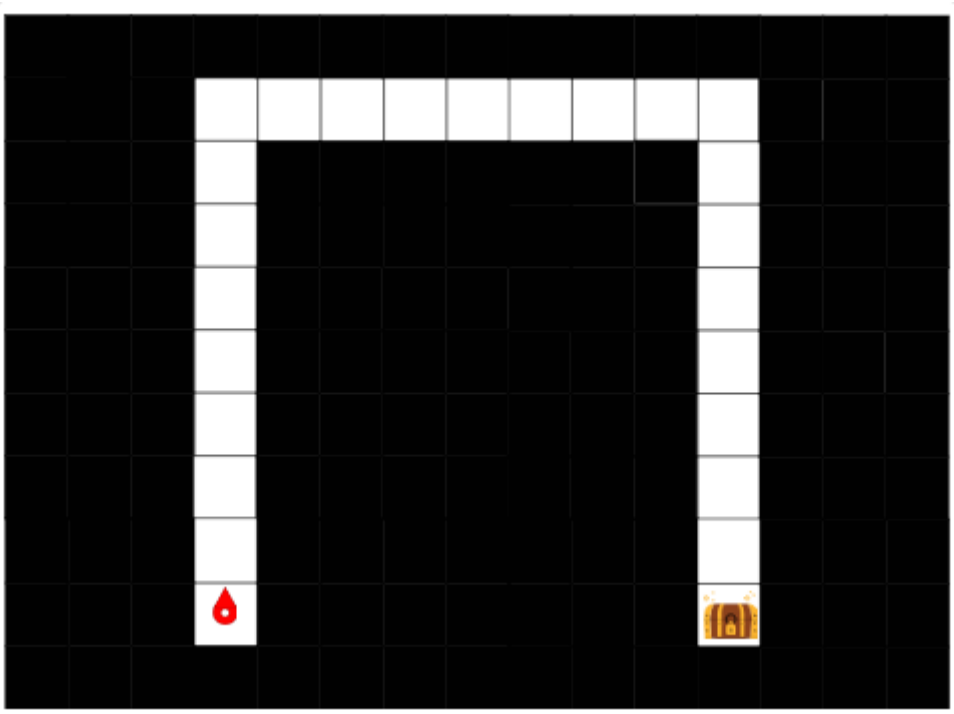


Iteratie 1



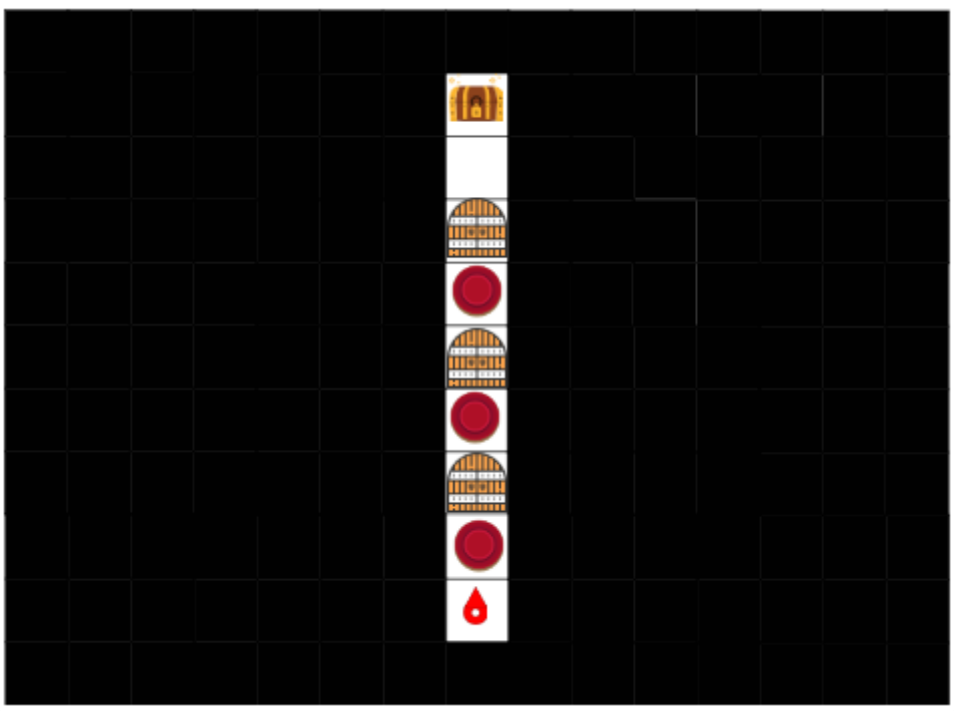
Bekijk van binnen

Iteratie 2



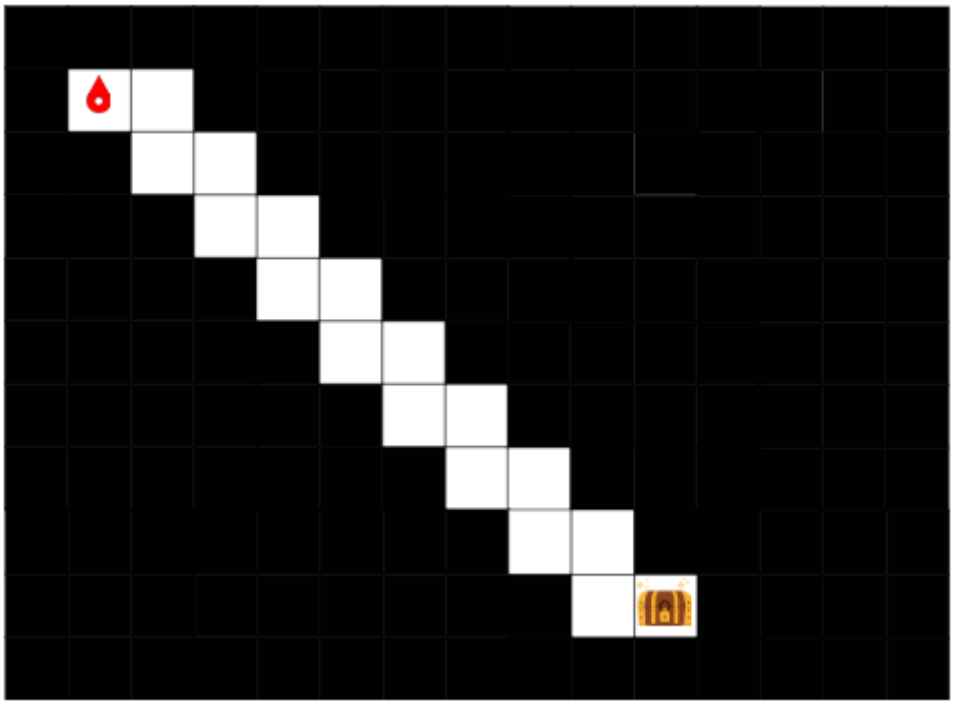
Bekijk van binnen

Iteratie 3



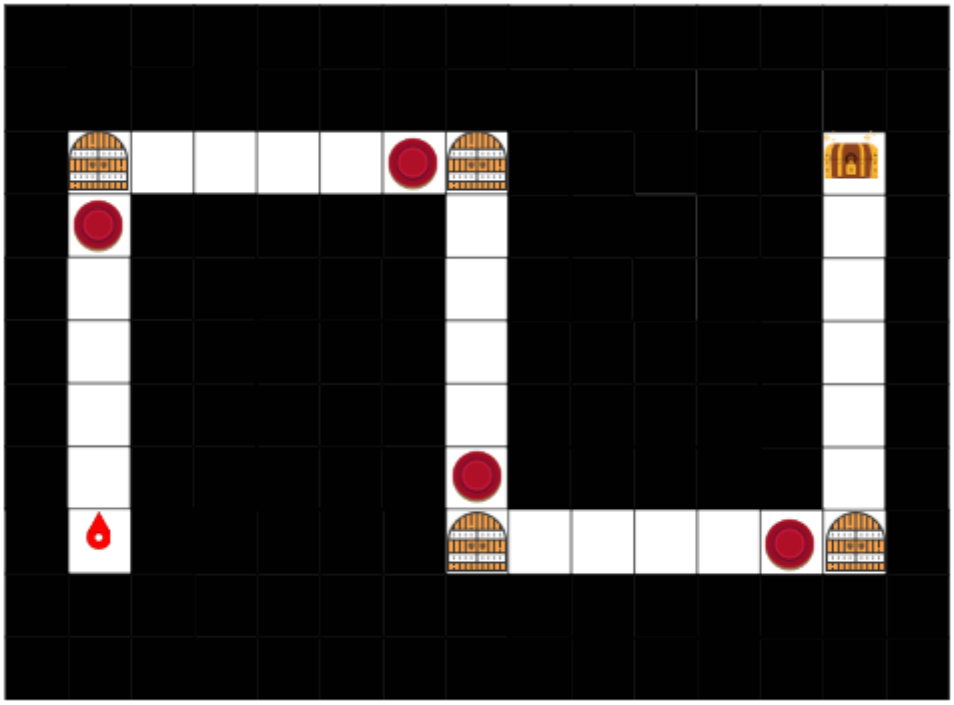
Bekijk van binnen

Iteratie 4



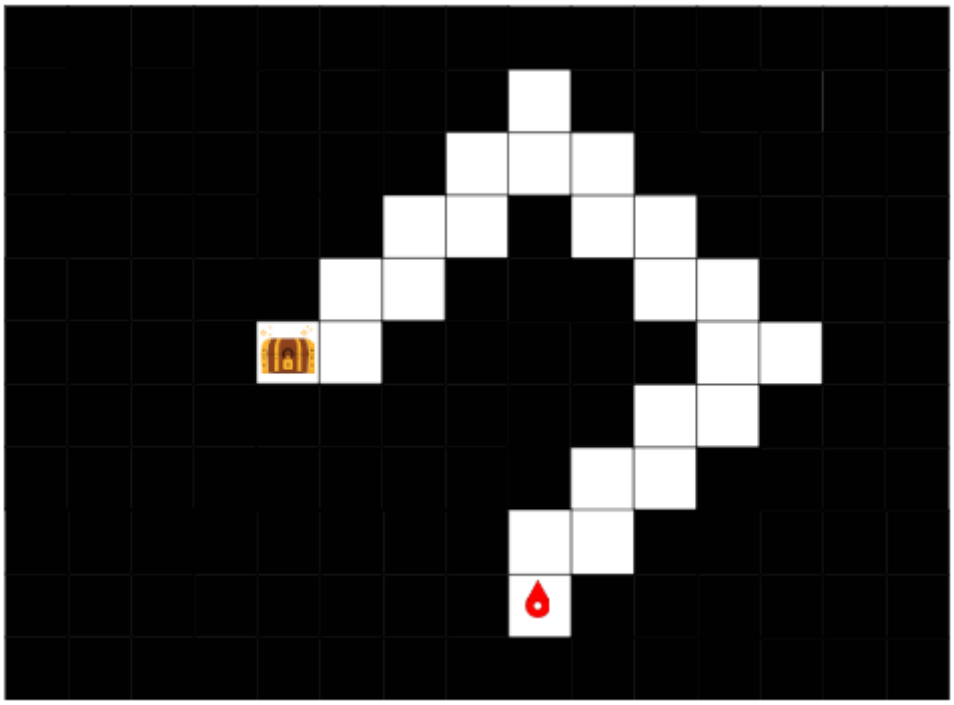
Bekijk van binnen

Iteratie 5



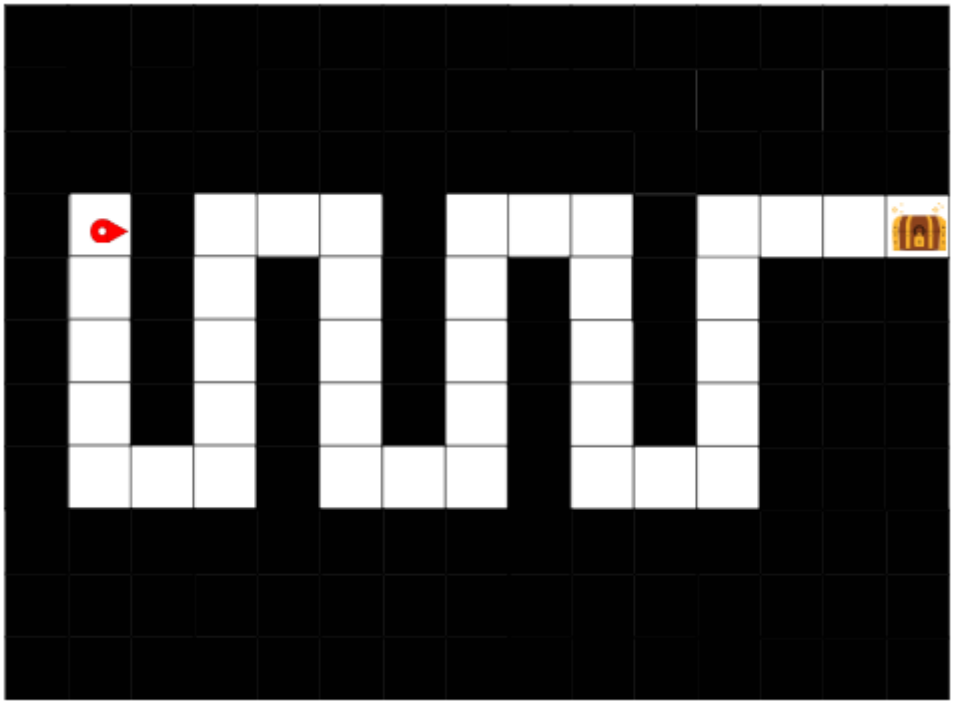
Bekijk van binnen

Iteratie 6



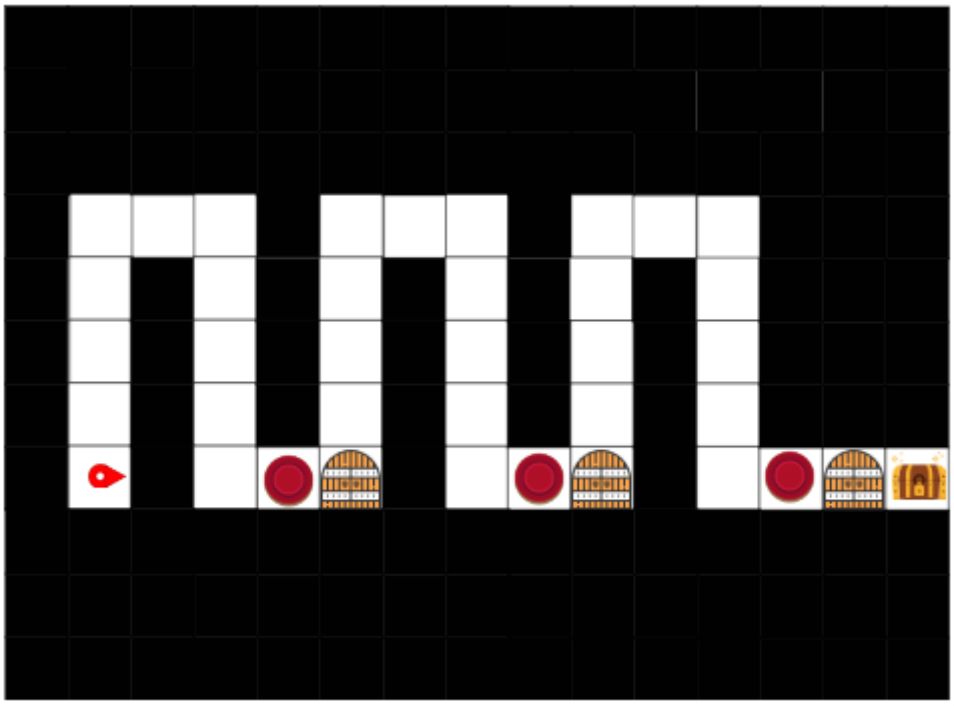
Bekijk van binnen

Iteratie 7



Bekijk van binnen

Iteratie 8



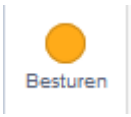
Bekijk van binnen

De selectie

> **Wat moet je kennen en kunnen na dit deel?**
> - Keuzes kunnen maken in je programma met behulp van een selectie.

In dit deel zullen we gebruik maken van een paar nieuwe blokjes in Scratch.
Het **Als-Dan** blokje en het **Als-D**

Deze blokjes vind je in het "Besturen" menu links:



Condities

Bovenaan in elk blokje zie je een donkere zeshoekige ruimte.
Hierin moeren we een **Conditie** (of voorwaarde) zetten.

> Een **Conditie** is een **ja-nee vraag**. Dit type vraag heeft altijd maar 2 mogelijke antwoorden. **Ja** of **Nee**. (Soms worden ook **Waar** of **Onwaar** gebruikt in plaats van **Ja** en **Nee**)

In Scratch kunnen we alle mogelijke **Condities** vinden in het "Waarnemen" menu links:

De conditie die we het meest zullen gebruiken in de oefeningen is de **"Raak ik" conditie**:

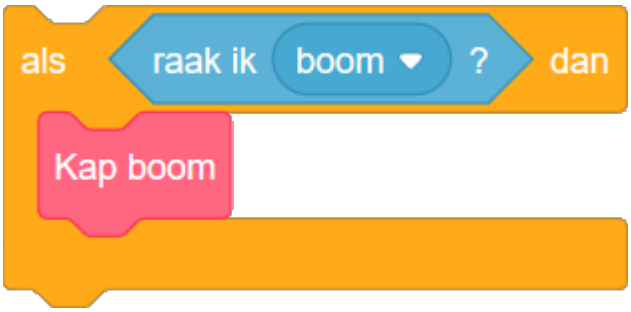
Deze conditie controleert of het huidige karakter een ander iets aanraakt.
Standaard wordt er gecontroleerd of het karakter de muisaanwijzer aanraakt.
Maar dit kunnen we veranderen door op het pijltje te klikken:

We kunnen dus ook kiezen om te controleren of we iets anders aanraken.
Zoals de schatkist of een knop.

In de oefeningen hieronder komt er een nieuw symbool bij, een boom.
Dit symbool wordt later verder uitgelegd.

Als-Dan(-Anders)

Deze condities kunnen we dan gebruiken in de **Als-Dan** of de **Als-Dan-Anders** blok.
De conditie wordt gecontroleerd, en als het resultaat **Waar** is, dan worden de blokjes binnenin het **Als-Dan** blok uitgevoerd.
Als de conditie **Niet waar** is dan worden de blokjes binnenin het **Als-Dan** blok overgeslagen.



In het voorbeeld hierboven wordt er gecontroleerd of ons karakter een boom raakt.
Zo ja, dan wordt de boom omgekapt. (het blokje **Kap boom** wordt wel uitgevoerd).
Anders wordt het blokje **Kap boom** niet uitgevoerd.

In het voorbeeld hierboven wordt **Kap boom** uitgevoerd als de conditie **raak ik boom** waar is.
Als deze niet waar is dan wordt het blokje **1 vak vooruit** uitgevoerd.
Bij het **Als-Dan-Anders** blokje wordt er dus altijd 1 van de 2 blokjes uitgevoerd.

De boom

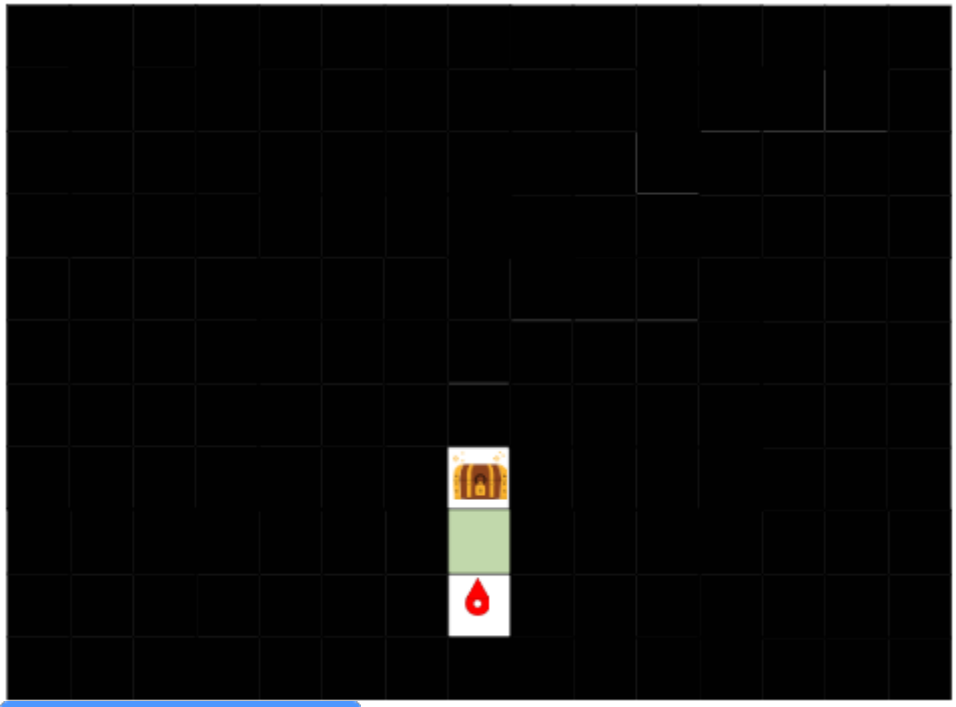
In de oefeningen hieronder komt er dus een nieuw symbool bij, de boom.

Bomen kunnen enkel maar groeien op de **groene vakken** op de kaart.
Om voorbij een boom te raken moet je op het vakje van de boom gaan staan en het **Kap boom** blokje gebruiken.

> **Opgelet! Bomen staan er niet altijd!**
> Elke keer dat je op de start knop drukt worden de bomen opnieuw geplaatst.
> Er is 50% kans dat er een boom zal verschijnen op een groen vak, en 50% kans dat er geen boom zal staan.

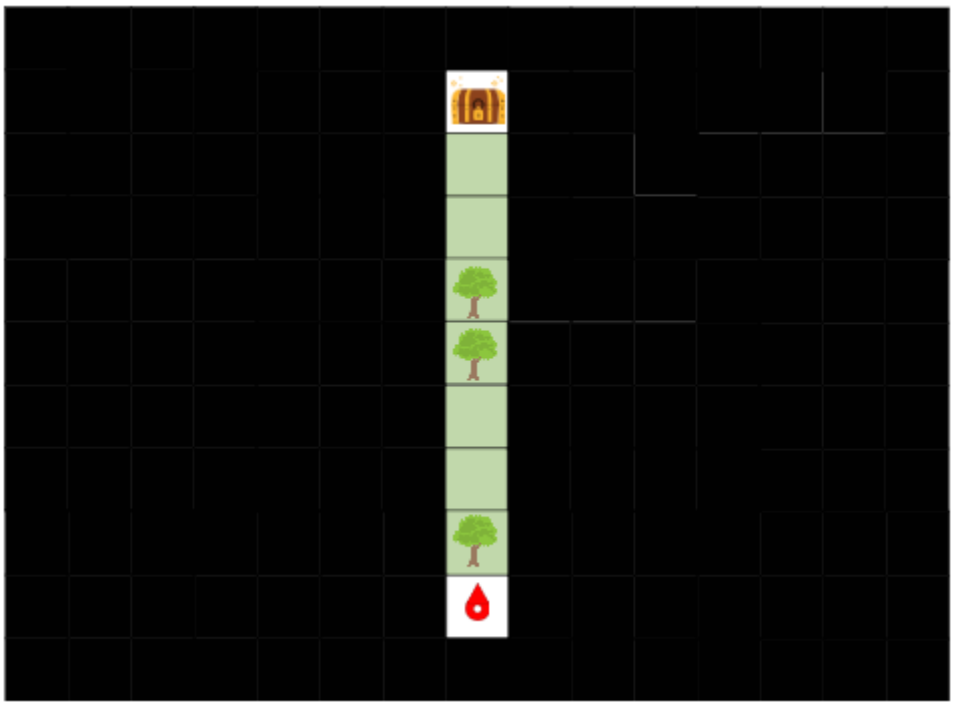
Je kan enkel het blokje **Kap boom** gebruiken als er een boom staat.
Je zal dus om deze oefeningen op te lossen het **Als-Dan** blokje of het **Als-Dan-Anders** blokje moeten gebruiken.

Selectie 1



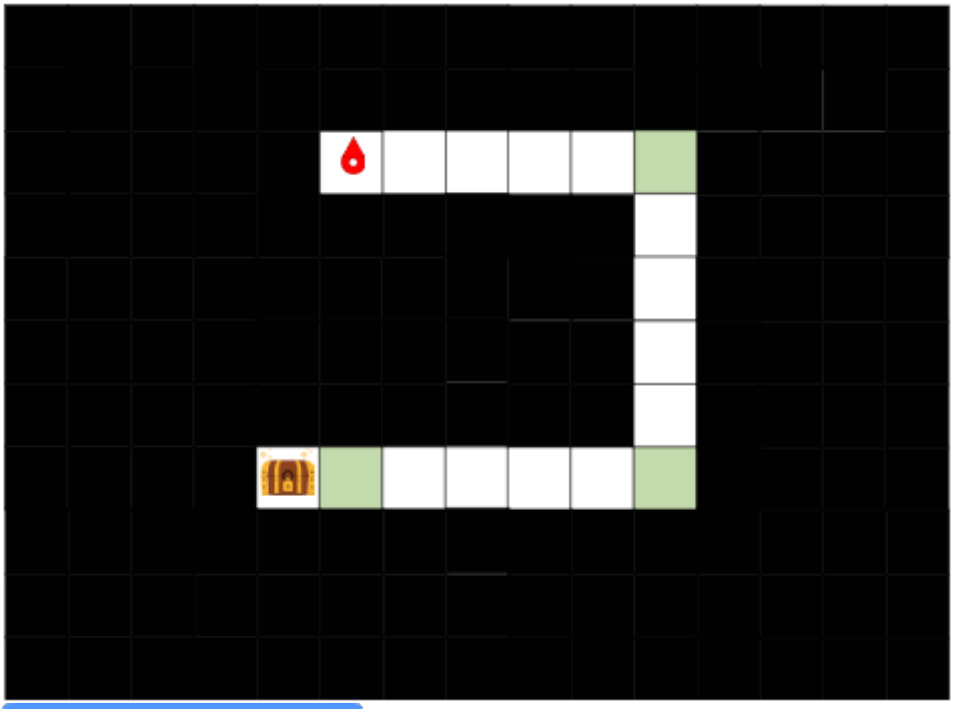
Bekijk van binnen

Selectie 2



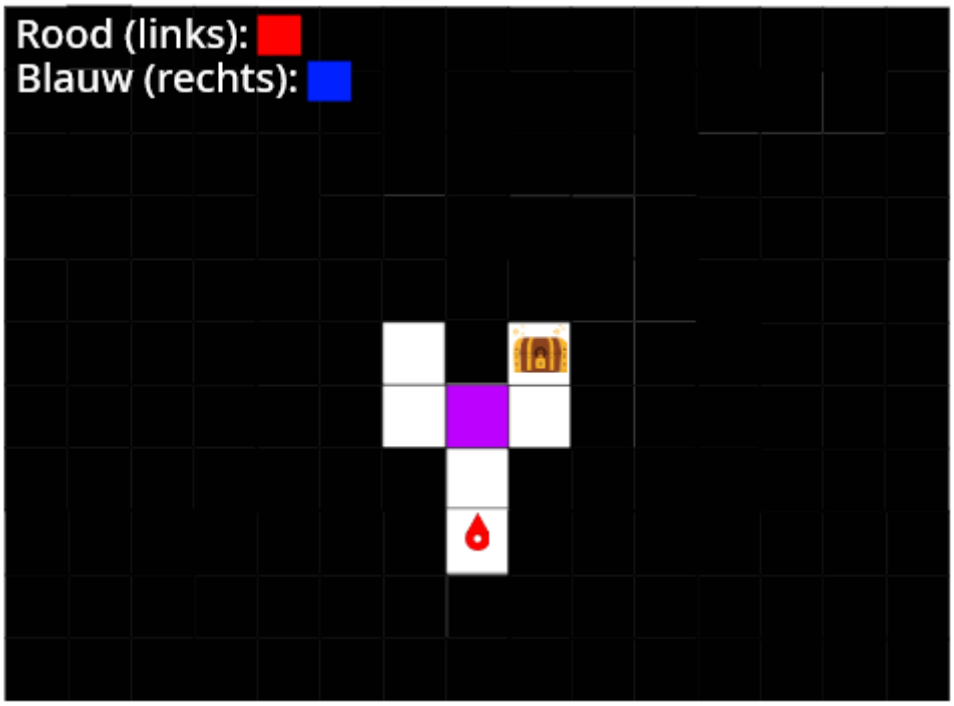
Bekijk van binnen

Selectie 3



Bekijk van binnen

Selectie 4



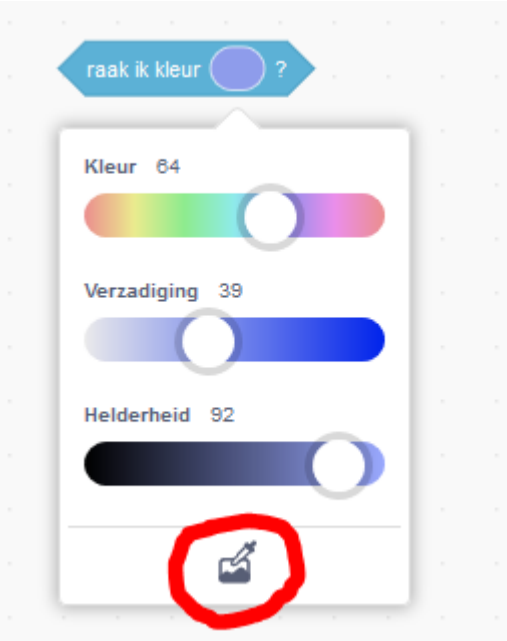
Bekijk van binnen

In deze oefening zie je een paars vak staan.
Als je op de **groene vlag** klikt zal dit paarse vak veranderen naar rood of blauw.
Als het vak **rood** is dan moet je de linkse afslag nemen.
Als het vak **blauw** is dan moet je de rechtse afslag nemen.

Probeer eens meerdere keren op de groene vlag te klikken. Het pad is niet altijd het zelfde! Maar het programma dat je schrijft moet wel altijd tot het einde raken.

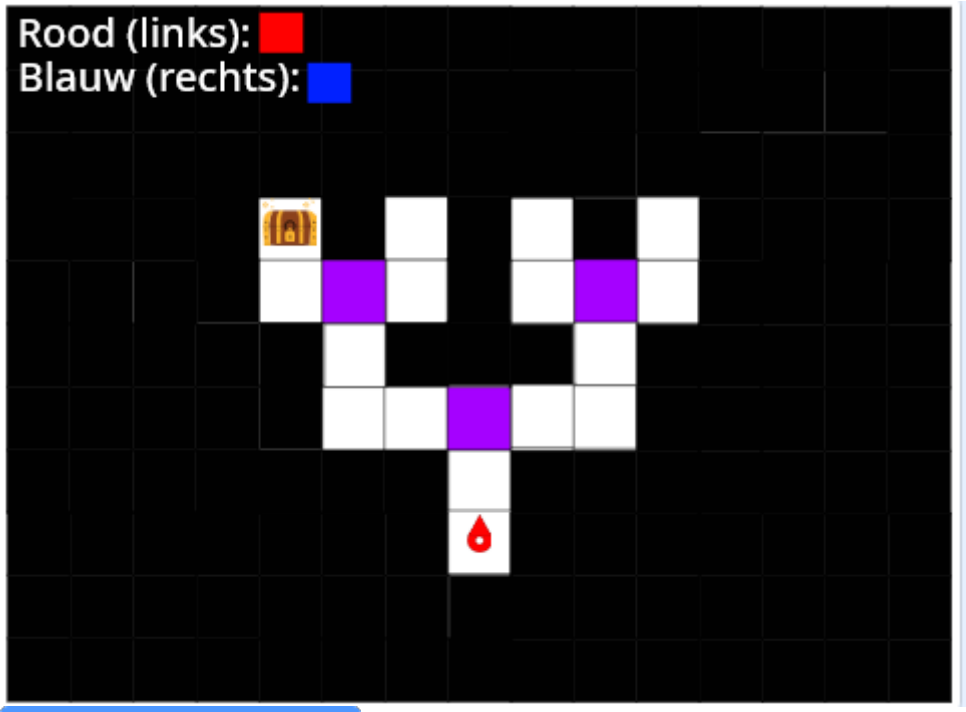
Om te controleren of je op een rood of een blauw vak staat kan je deze **Conditie** gebruiken:

Als je op het bolletje klikt met de kleur in dan kan je een andere kleur kiezen:



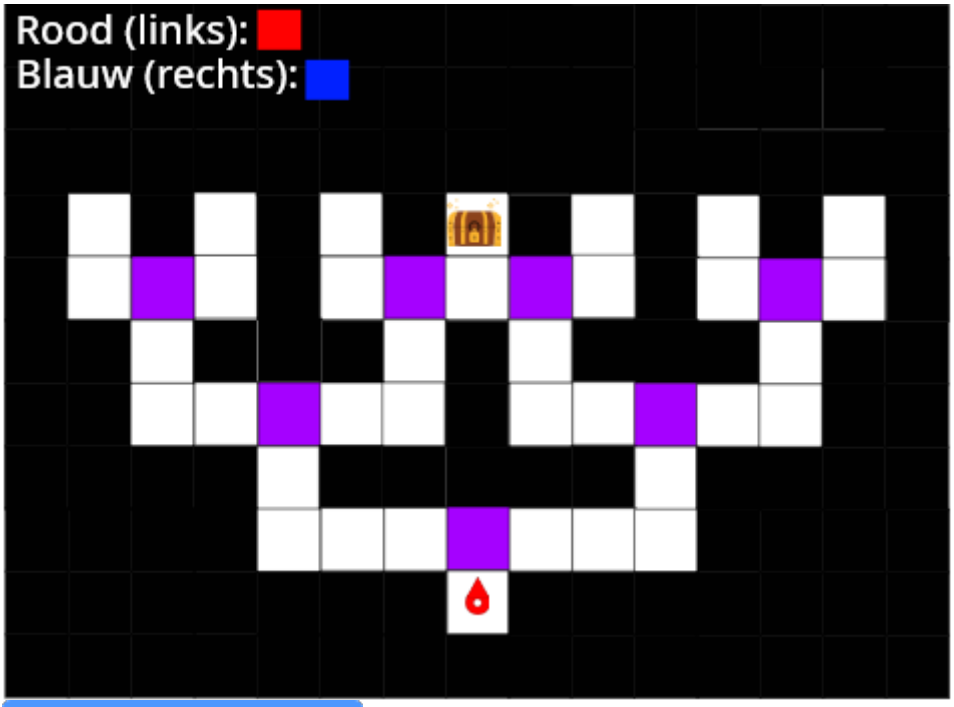
Als je dan op de onderste knop klikt (hierboven aangeduid met een rode cirkel) dan kan je een andere kleur aanduiden.
Je kan dan de rode of blauwe kleur linksboven op de map selecteren.

Selectie 5



Bekijk van binnen

Selectie 6



Bekijk van binnen

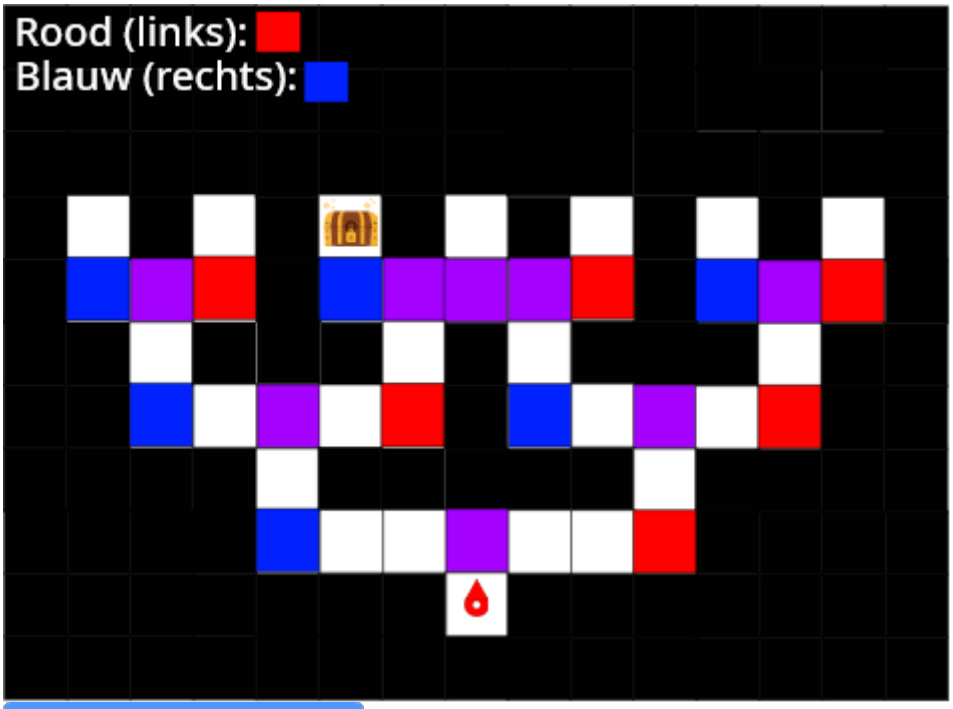
Selectie 7

Deze oefening kan je op meerdere manieren oplossen.
Een van deze blokjes kan daarin helpen:

Het linkse blokje herhaalt oneindig en het rechtse herhaalt totdat een bepaalde conditie waar is.

In het herhaal tot blokje kan je een **Conditie** zetten.
Het zal de blokjes binnenin zich blijven herhalen tot de conditie waar is.
Dan stopt de herhaling.

Probeer je oplossing zo kort mogelijk te maken!
Je kan deze oefening oplossen door maar 6 blokken en 2 condities te gebruiken.



Bekijk van binnen