Kennisnet

Workshop Computational thinking

Auteurs	Jan-Bart Vreede ; Kennisnet LleG ; Sander van Acht ; Lalibel Mohaupt
Team	Stichting Kennisnet j.devreede@kennisnet.nl
Laatst gewijzigd	30 augustus 2022
Licentie	CC Naamsvermelding 3.0 Nederland licentie
Webadres	https://maken.wikiwijs.nl/70012/

Inhoudsopgave

Welkom

Welkom bij deze online workshop over 'Computational thinking' voor leraren!

https://www.youtube.com/embed/SgjR0BV_sIs?list=PLQI9hXCcoK1SS7eN8lv_Lzj91SfE3W2Vu&controls=0&showinfo=0&rel=0

Voor vragen kun je contact opnemen met Remco Pijpers via r.pijpers@kennisnet.nl of 0800 3212233

Van logisch denken naar programmeren

In deze workshop leer je logisch nadenken. Uitdagingen die op het eerste gezicht soms lastig lijken, maken we eenvoudig door ze op te delen in kleinere stukken, die gemakkelijk op te lossen zijn. Deze aanpak is niet alleen handig als je met computers werkt, maar komt ook van pas in het dagelijks leven.

Want *computational thinking* kom je overal tegen, in de wekker die je wakker maakt, het stoplicht dat het verkeer regelt, of de magnetron die je havermout opwarmt. Maar ook in je smartphone, met software om je vingerafdruk te herkennen, tekst om te zetten naar spraak of de route naar je bestemming uit te stippelen.

En laten we de spelletjes niet vergeten. Kijk maar eens naar wat kinderen hebben gemaakt na een eerste kennismaking met *computational thinking*! Aan het einde van deze workshop heb je voldoende kennis om deze wall of fame uit te breiden met jouw eigen game.

Dit leer je in deze workshop

Deze online workshop is ontwikkeld voor jou, leerkracht basisonderwijs of docent voortgezet onderwijs in de onderbouw. *Computational thinking* maakt onderdeel uit van de 21e eeuwse vaardigheden, maar kan op het eerste gezicht best lastig lijken. We introduceren je in de wereld van logisch nadenken en programmeren - en laten je zien dat dat helemaal niet eng of ingewikkeld hoeft te zijn, maar ook vooral heel leuk is!

Onze ambitie is dat je na afloop van deze workshop:

- weet wat *computational thinking* inhoudt en de meerwaarde voor jou en je leerlingen kent
- bekend bent met de belangrijkste termen en concepten van programmeren
- in staat bent om een uitdaging te vertalen in een logisch stappenplan
- de belangrijkste principes toe kan passen om tot een oplossing te komen
- maar vooral: dat je met de oefeningen, opdrachten en programma's die we je aanreiken direct met jouw leerlingen zelf aan de slag kunt!

Zo werkt deze workshop

Deze workshop is opgebouwd uit een introductiemodule, een basismodule, vier verdiepende modules en een overzicht van voorbeeldlessen. We hebben ze gerangschikt in een volgorde van opbouwende complexiteit en raden je daarom aan ze lineair te volgen. Zo leer je stapsgewijs wat *computational thinking* is en hoe je met je eigen leerlingen aan de slag kunt. Maar heb je al wat voorkennis, dan ben je uiteraard ook vrij om direct een module in te duiken die jou het meeste aanspreekt:

---

1. Basismodule Programmeren (duur: ca. 20 minuten)
In deze module leer je de belangrijkste begrippen die we vaak gebruiken in de context van programmeren. Geen abracadabra, maar termen en concepten die je houvast geven wanneer je straks daadwerkelijk aan de slag gaat. De begrippen komen in alle verdere modules terug, dus het is handig om deze als startpunt te gebruiken.

2. Introductiemodule (duur: ca. 10 minuten)
We kunnen ons geen wereld meer voorstellen zonder technologie. Iedereen gebruikt het, maar slechts weinigen weten hoe het werkt. Dat is zonde, want je kunt er zoveel moois mee creëren! Je hoeft geen genie te zijn om het te leren. Maar als je het eenmaal onder de knie hebt... beschik je over een superkracht die niet veel mensen hebben!

3. Verdiepende modules (duur: ca. 30 minuten per module)
In de verdiepende modules leer je stap voor stap de belangrijkste principes van het programmeren. We starten steeds met een eenvoudige analogie met logisch nadenken in het dagelijks leven, laten je zien hoe je dit met leerlingen in de klas kunt behandelen en maken echte computerprogramma's. Hoewel de modules oplopen in complexiteit, blijft de uitleg steeds eenvoudig en zijn de oefeningen in spelvorm.

5. Voorbeelden en voorbeeldlessen (duur: eigen invulling)
Om direct met je eigen leerlingen aan de slag te kunnen bieden we je aan het einde van iedere module al voorbeelden die je in je klas kunt gebruiken. De voorbeelden passen bij de verschillende niveaus die we in de verdiepende modules behandelen. In de laatste module vind je ook nog eens een overzicht van voorbeeldlessen, die je zo in de praktijk kunt brengen.

LET OP: Klik je tijdens het afspelen van een video op 'Volgende' om naar de volgende pagina te gaan, dan blijft de video verder spelen. Zet dus eerst de video op stop voordat je een volgende pagina opent.

We wensen je heel veel plezier en succes met deze workshop! En mocht je vragen hebben, dan kun je natuurlijk altijd direct contact met ons opnemen via r.pijpers@kennisnet.nl of via telefoonnummer 0800 3212233.

Wat weet je eigenlijk al?

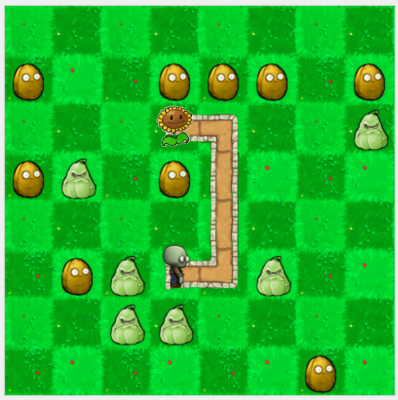
Deze workshop gaat over denken als een robot. Dat houdt in dat je stap voor stap uitvogelt hoe je je doel kunt bereiken. Dat klinkt misschien ingewikkeld, maar dat valt reuze mee. Heb je weleens aan iemand uitgelegd hoe je naar het station loopt? Of cupcakes gebakken op basis van een recept? Dan heb je onbewust een programma geschreven én uitgevoerd!

Het is een kwestie van logisch nadenken en precies aangeven wat je bedoelt. Kijk maar eens naar onderstaande afbeeldingen van een doolhof, waarin een zombie de weg zoekt naar een zonnebloem. Kun jij beredeneren welke route bij welk doolhof hoort? Dan ben je al "computationeel" aan het denken. Verderop in deze workshop leer je bovendien om een simpele oplossing te bedenken waarmee élke zombie in íeder doolhof zijn zonnebloem kan vinden. Dat is best indrukwekkend toch?

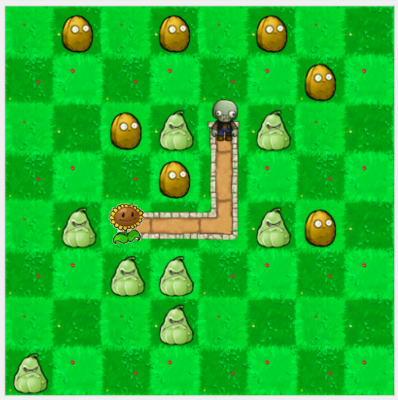
In onderstaande test kun je jouw voorkennis demonstreren!

Vind de weg van zombie naar zonnebloem

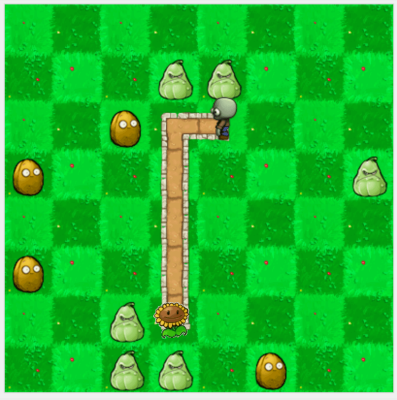
Hier zie je drie doolhoven én drie routebschrijvingen om de zombie naar de zonnebloem te brengen. Lukt het jou om de juiste route naar de juiste afbeelding te slepen?

☐ 

a Beweeg een stap vooruit, draai linksom, beweeg steeds een stap vooruit totdat je bij de zonnebloem bent.

☐ 

b Beweeg een stap vooruit, draai linksom, beweeg drie keer een stap vooruit, draai linksom, beweeg een stap vooruit.

☐ 

c Beweeg een stap vooruit, beweeg een stap vooruit, draai rechtsom, beweeg een stap vooruit, beweeg een stap vooruit.

Advies van Allard Strijker (SLO)

Wat moet je met *computational thinking* op school? We vroegen het aan niemand minder dan Allard

Strijker van SLO, dé expert op het gebied van digitale geletterdheid in het onderwijs. Laten we hem eerst even netjes voorstellen:

https://www.youtube.com/embed/0PYwflziAcg?rel=0&controls=0&showinfo=0&rel=0

Computational thinking?

Wanneer het gaat over mediawijsheid, digitale vaardigheden en 21e eeuwse vaardigheden, rolt de term '*computational thinking*' regelmatig over tafel. Maar wat is het eigenlijk en wat heb je er aan? Allard legt uit wat *computational thinking* is en hoe het verschilt van programmeren:

https://www.youtube.com/embed/videoseries?list=PLQI9hXCcoK1SyXYFGQjUaz3NvkT2-AySc&controls=0&showinfo=0&rel=0

Computational thinking is een manier van denken, waarmee je grote uitdagingen klein en moeilijke oplossingen simpel kunt maken. Het gaat om het combineren van de creativiteit van menselijk denken met de computationele kracht van computers, om antwoorden te vinden voor uiteenlopende vraagstukken in uiteenlopende disciplines.

Vaak denken we meteen aan programmeren, maar dat is slechts één van de manieren om *computational thinking* toe te passen. Net zoals je je muzikaliteit kunt inzetten om piano te leren spelen, maar ook om geluidstechnicus te worden bij concerten of onderzoek te doen naar het menselijk gehoor. Zo is *computational thinking* veel meer, breder en krachtiger dan alleen programmeren.

Computational thinking bestrijkt ook:

- het formuleren van uitdagingen zodat ze op te lossen zijn met computers
- het logisch structureren, analyseren, abstraheren en weergeven van de informatie om oplossingen te vinden
- de meest effectieve en efficiënte stappen vinden om tot een antwoord te komen
- het generaliseren van dit proces naar andere toepassingen

Het gaat over uitdagingen analyseren, informatie structureren en logisch redeneren. Je ontwikkelt ruimtelijk inzicht en probleemoplossend vermogen. Dat zijn handige vaardigheden voor programmeren, maar ze bieden ook een enorme meerwaarde in de rest van je leven. *Computational thinking* geeft bovendien het doorzettingsvermogen en vertrouwen om complexe uitdagingen aan te gaan.

Het biedt een meerwaarde, zelfs als jij of je leerlingen helemaal niks hebben met computers, geen exacte studie willen volgen of een baan ambiëren waarin techniek een primaire rol speelt. Want computers en programma's zijn niet meer weg te denken uit ons leven, ze spelen altijd en overal een rol. En analytisch denken en logisch redeneren komen in alle sectoren en profielen van pas, ook als het om analoge vraagstukken gaat.

Denk niet dat *computational thinking* draait om hacken voor nerds, want dan mis je de leukste aspecten.

Het gaat namelijk ook over het ontwikkelen van je creativiteit, leren samenwerken... en de kick die je krijgt als het je lukt om de computer iets te laten doen dat jij zelf hebt bedacht. Pas op, want die kan verslavend zijn!

21e eeuwse vaardigheden

Computational thinking is een belangrijk onderdeel van de 21e eeuwse vaardigheden, oftewel de kennis en vaardigheden die van belang zijn om leerlingen voor te bereiden op een snel veranderende maatschappij. Dat geldt voor alle sectoren. Kom je het niet tegen op je werk, dan op z'n minst in je privéleven, aldus Allard:

https://www.youtube.com/embed/oM9hyyALtrU?list=PLQI9hXCcoK1SS7eN8lv_Lzj91SfE3W2Vu&controls=0&showinfo=0&rel=0

Het Platform Onderwijs2032 schrijft hierover:

"De impact van nieuwe technologieën op wie we zijn en wat we doen is groot en wordt eerder onder- dan overschat. De hoeveelheid beschikbare informatie neemt exponentieel toe. Technologische ontwikkelingen leiden tot structurele veranderingen op het gebied van werk en in de manier van samenleven in een mondiale maatschappij.

"Toekomstgericht onderwijs maakt leerlingen digitaal vaardig en mediawijs en stelt hen in staat zich op dat vlak te blijven ontwikkelen. Kennis van actuele informatie- en communicatietechnologie (ICT) en zogeheten computational thinking zorgt ervoor dat leerlingen leren begrijpen hoe ze telkens nieuwe technologische diensten en producten kunnen benutten. Leerlingen leren begrijpen welke rol logisch redeneren en programmeren daarin spelen, hoe je digitale informatie kunt duiden en verwerken en hoe je omgaat met (digitale) media en beelden.

"Het Platform vindt dat werken en leren in de digitale wereld en met nieuwe technologieën tot de kern van toekomstgericht onderwijs behoren. Het gaat om vier onderdelen: dat leerlingen ICT-basiskennis opbouwen, informatievaardigheid ontwikkelen, mediawijs worden en leren begrijpen hoe technologie werkt (computational thinking)."

SLO diept het nieuwste model van de 21e eeuwse vaardigheden verder uit op http://curriculumvandetoekomst.slo.nl/21e-eeuwse-vaardigheden/

Na deze workshop

Kennisnet heeft met deze workshop als doel om jou en de hele school enthousiast te maken om met *computational thinking* aan de slag te gaan. Maar na deze workshop ben je er zeker nog niet... Dit is slechts een opstapje.

Wil je concreet in je eigen school aan de slag met het ontwikkelen van een leerlijn 'Computational thinking', dan kun je handig gebruik maken van de door Kennisnet en SLO ontwikkelde leerlijn Programmeren: http://kn.nu/leerlijnprogrammeren

Voor volledige professionalisering kun je onder meer bij de volgende organisaties terecht.

Marktaanbod

CODESTARTER	Op **Codestarter** vind je een uitgebreid overzicht van lesmateriaal en activiteiten over programmeren.
4PIP	**4PIP** verzorgt lezingen en worskhops op scholen aan ouders, leraren en leerlingen over mediawijsheid, games design en programmeren.
CODECULT	**Codecult** is een programmeerclub en summerschool voor kinderen en verzorgt workshops en cursussen over programmeren voor het po en vo.
CODER(DOJO);	In **CoderDojo**'s leren kinderen van 7 tot en met 17 leren programmeren van vrijwilligers, op lokaties door heel Nederland .
	CodeUur organiseert met hulp van vrijwilligers workshops in groep 7 en 8 van de basisschool over programmeren en organiseert tevens events rondom dit thema.
DIGITAL PLAYGROUND	Bij **Digital Playground** kunnen jongeren tussen de 12 en 19 jaar workshops en cursussen volgen over programmeren in Amsterdam, Den Haag, Rotterdam en in overleg op andere lokaties.
digiVITA.	**DigiVita** stimuleert meisjes en vrouwen om met ICT aan de slag te gaan door verschillende activiteiten binnen en buiten de school, zoals een bezoek aan het bedrijfsleven.

Gamescool biedt cursussen gamedesign voor leraren po, vo en mbo, die voldoende basis bieden om er zelf mee aan de slag te gaan op school.

IQMaak verzorgt cursussen, trainingen, workshops en gastlessen voor kinderen, hun ouders en leraren over programmeren, elektronica en maakonderwijs.

Materialen

De **Bendoo Box** biedt materialen met een lesboekje voor kinderen om aan de hand van voorbeelden en experimenten aan de slag te gaan met techniek, van hardware tot software.

Bomberbot is een educatieve game (online platform) met lesmaterialen voor kinderen van 8 tot en met 14 jaar om programmeren en computational thinking te leren.

CodeKlas is een inspiratieboek door Pauline Maas over programmeren met kinderen, voor leraren en ouders van kinderen in het basis- en voortgezet onderwijs.

ScratchWeb.nl verzamelt en schrijft lesmateriaal om de belangrijkste concepten van programmeren te leren, dat vrij te gebruiken.

Squla biedt een lesmodule om spelenderwijs te leren programmeren voor basisschoolleerlingen in groep 4 tot en met 8.

The Foos biedt een educatieve game (online platform en app) met een en lesmaterialen om kinderen vanaf de kleuterschool bekend te maken met computational thinking.

Naast deze aanbieders vind je nog veel meer informatie over *computational thinking*, lesmateriaal, evenementen en andere partners bij de netwerkorganisatie **Mediawijzer.net**.

Introductie

Meer dan je denkt

Wat leuk dat je bent begonnen aan de introductie van *computational thinking*!

Maar... wat hebben jij en ik daar eigenlijk mee te maken? Meer dan je denkt! Kijk maar eens naar de volgende video.

https://www.youtube.com/embed/WAU3z1MQId4?list=PLQI9hXCcoK1SS7eN8lv_Lzj91SfE3W2Vu&controls=0&showinfo=0

Programmeren is leren denken

Twijfel je nog steeds of dit iets is voor jou? Weet dan, dat mensen als Bill Gates, Mark Zuckerberg en Steve Jobs deze stap toejuichen!

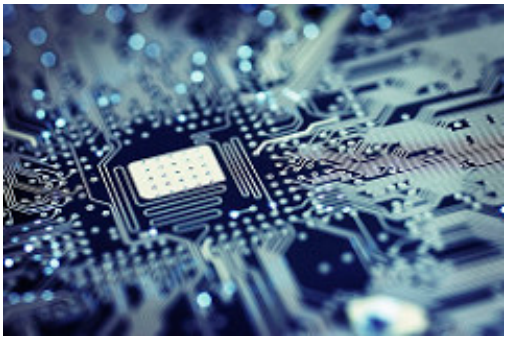
Steve Jobs zei ooit: "Iedereen zou moeten leren om te programmeren, want zo kun je leren hoe je moet denken."

Computational thinking gaat niet over ingewikkelde problemen en ingewikkelde oplossingen. Integendeel! Het is een manier van denken, waarmee je grote uitdagingen klein kunt maken en moeilijke oplossingen simpel. Je kunt het gebruiken om mensen te helpen en een verschil te maken in de wereld. Dat maakt het een van de belangrijkste vaardigheden van de 21e eeuw.

Je bent nooit te oud om die vaardigheid te leren en nooit te jong om er aan te beginnen. *Computational thinking* komt iedereen van pas, ook jou.

Maar waarom is dat zo belangrijk? En wat heb je nodig om het te leren? En wat bereik je daar dan mee? Dat leggen we uit in deze module.

Computers zijn overal



"We zijn volledig afhankelijk van technologie. Van communicatie tot telebankieren," zegt Will.I.Am van de Black Eyed Peas, "...en toch zijn er maar weinigen die computercode kunnen lezen en schrijven."

Vanaf het moment dat we wakker worden tot we gaan slapen, gebruiken we computers en programma's. We kunnen er zelfs bijna niet meer zonder. Alleen hebben de meesten van ons geen idee hoe ze werken. Er zijn slechts heel weinig mensen bekend met *computational thinking*.

Maar juist die mensen maken de mooiste producten en diensten. Denk maar eens aan die momenten op je dag waarop je gebruik vaakt van technologie. Van de stoplichten op straat tot social media op je smartphone. Kun jij je nog een leven voorstellen zonder al deze innovaties?

Er is geen enkele sector die niet radicaal verandert door technologie. Zelfs als je professioneel sport, of in de showbizz werkt, heb je ICT nodig. Overal spelen computers een belangrijke rol, van natuurwetenschap tot zorg en van economie tot kunst.

De autobranche wordt bijvoorbeeld sterk beïnvloed door de opkomst van ict. Sleutelen, kruipolie en vieze handen zijn bijna verleden tijd. In de toekomst werken monteurs met een laptop.

Computational thinking leert je om niet alleen gebruik te maken van alle nieuwe mogelijkheden, maar ook om ze zélf te creëren. Zelfs als je geen programmeur wilt worden, heb je hier profijt van, welk beroep je ook kiest. Want het leert je om logisch te redeneren en gestructureerd te denken, het vergroot je probleemoplossend vermogen, ruimtelijk inzicht en je creativiteit.

Dus als jij bekend bent met *computational thinking*, geeft jou dat een enorme voorsprong!

Maar wat heb je eigenlijk nodig om te beginnen met *computational thinking*? Wat moet je kunnen en kennen?

Wat is waar?

Wat heb je nodig om met computational thinking aan de slag te gaan? Vier beroemdheden geven vier antwoorden:

1. "Optellen, aftrekken... dat is het wel zo'n beetje," aldus Bill Gates van Microsoft.
2. "De tafels van vermenigvuldiging, die zijn ook wel handig," aldus Tony Hsieh van Zappos.
3. "Lezen, meer hoef je niet te kunnen," aldus Jack Dorsey van Twitter.
4. "Alleen een computer, verder niks," aldus Drew Houston van Dropbox.

Wie van de heren heeft gelijk?

- ○ Geen van bovenstaande antwoorden is waar.
- ○ 1 en 2.
- ○ 3.

○ 4.

○ 1, 2, 3 en 4.

Voor iedereen van belang

'Computational thinking is voor iedereen van belang', vindt Allard. Want als je het in alle beroepen nodig hebt, is het gek dat het niet in alle vakken terug komt. Moeten we dan allemaal aan de slag met *computational thinking*?

https://www.youtube.com/embed/90YOIOIUy40?rel=0&controls=0&showinfo=0&rel=0

Je hoeft het alleen te willen

In het begin kan *computational thinking* best intimiderend zijn. Maar leren lopen, praten en rekenen ging ook niet vanzelf. Je hoeft niet superslim, een whizzkid of een nerd te zijn. Het is voldoende als je kunt optellen en aftrekken. Vermenigvuldigen kan ook handig zijn. Voor de rest is het aan jou: als jij het wilt, kun jij het leren!

Je hoeft toch ook geen genie te zijn om te lezen?

Bovendien hoef je niet alles te kunnen en te kennen om er profijt van te hebben. "Leren programmeren betekende voor mij niet dat ik alles van ICT wilde weten. Absoluut niet! Ik begon er alleen maar aan omdat ik één simpel dingetje wilde doen: iets leuks maken voor mezelf en mijn zussen," zegt Mark Zuckerberg van Facebook. "Dus schreef ik een klein programmaatje, dat ik steeds verder uitbreidde. En als ik iets nieuws wilde weten, zocht ik het op in een boek of op internet."

Programmeren kun je vergelijken met het bespelen van een instrument of beoefenen van een sport. In het begin lijkt het lastig, maar na een tijdje krijg je het door. En als je het eenmaal door hebt... dan wordt het leuk!

Je hoeft het alleen te willen

Zet de woorden in de goede volgorde. Let goed op hoofdletters en leestekens.

- [ ] **a** ik
- [ ] **b** leren!
- [ ] **c** Computational
- [ ] **d** kan
- [ ] **e** thinking

Jouw geheime superkracht

Stel je voor dat jij een computerprogramma kunt schrijven, waarmee je miljoenen mensen bereikt en hun leven blijvend verandert, hoe gaaf is dat? Of je nu rijk wilt worden of de wereld wilt veranderen, *computational thinking* helpt je om jouw doel te bereiken. Het is een ontzettend krachtige vaardigheid.

"De programmeurs van morgen zijn de tovenaars van de toekomst. Het is alsof jij magische krachten hebt die niemand anders heeft," zegt Gabe Newell van Valve.

En kinderen? Die vinden het namelijk fan-tas-tisch. Nou ja, de meesten dan...

https://www.youtube.com/embed/5rLgI-b4BcU?rel=0&controls=0&showinfo=0&rel=0

Bovendien vergroot deze superpower de kansen van leerlingen om een baan te vinden. Geen gewone baan. Maar een baan op een kantoor met een drumstel, waar een chefkok je lunch maakt en je op een skateboard door de gang gaat. Want er zijn veel te weinig mensen die verstand hebben van *computational thinking*.

"Goeie programmeurs zijn rocksterren. Zo zit het," zegt Will.I.Am van de Black Eyed Peas.

Jij wilt toch ook een rockster zijn met superkrachten? Aan de slag!

Basismodule programmeren

Welkom

Programmeren is het schrijven van een programma met instructies voor een computer. Dat klinkt misschien ingewikkeld, maar dat hoeft het helemaal niet te zijn.

Want instructies geven doe je de hele dag door. Aan andere mensen, bijvoorbeeld als je aan iemand uitlegt hoe je die lekkere cupcakes bakt. Maar ook aan computers, bijvoorbeeld als je de oven instelt. In feite ben je in beide gevallen aan het programmeren.

In deze module gaan we aan de slag met instructies voor een computer aan de hand van een alledaags en menselijk voorbeeld. We leggen namelijk aan Olaf de Robot uit hoe je een oer-Hollandse boterham met hagelslag smeert.

Zo leer je de belangrijkste verschillen Ã©n overeenkomsten tussen het geven van instructies aan een persoon versus een computer. Ondertussen maak je kennis met de vijf belangrijkste begrippen die komen kijken bij programmeren. Net zoals het handig is om wegwijs te zijn in de keuken voordat je leert koken.

De vijf begrippen die we in deze module behandelen zijn:

- Programmeertaal
- Algoritme
- Statement
- Runnen
- Debuggen

Laat je niet van de wijs brengen door de moeilijke woorden, want je zult zien dat wij mensen een stuk slimmer zijn dan Olaf de Robot.

Een recept in mensentaal

Stel je eens voor: op vakantie in Barcelona heb je afgelopen zomer Maria leren kennen. Jullie raakten meteen bevriend. Nu is ze naar Nederland gekomen en stuurt je een bericht in het Spaans. Je begrijpt er niks van, maar als je de tekst voorlegt aan de robot van Google Translate, krijg je de volgende vertaling:

"Hoe maak je toch zo'n lekkere boterham met hagelslag? Je vertelde er zo enthousiast over toen je zo'n heimwee had! Ik wil er graag één proeven, maar ik weet het recept niet. De ingrediënten heb ik al wel in huis en ik sta nu aan het aanrecht. Wil je me helpen?"

Natuurlijk wil je Maria graag kennis laten maken met deze Nederlandse lekkernij en je stuurt haar het recept.

Hoe smeer je een boterham met hagelslag?

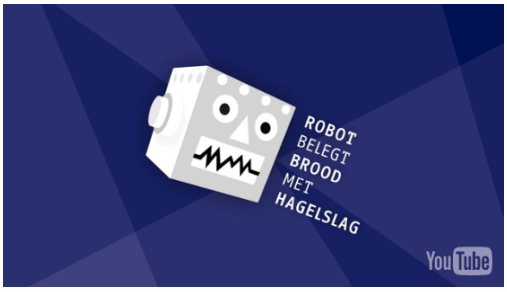
Schrijf in onderstaand tekstvak het recept voor een boterham met hagelslag. Op het aanrecht liggen de volgende benodigdheden:

- een zak brood
- een pak hagelslag
- een pakje boter
- een bord
- een mes

Wat zijn de stappen die Maria moet volgen? Je mag er van uit gaan dat Maria zelf de Nederlandse tekst voorlegt aan de robot van Google Translate die alles naar het Spaans vertaalt.

Klik op 'Controleer antwoord' om ons antwoordmodel te zien.

Een computer met hagelslag



Realiseer je je dat je net een programma hebt gemaakt voor een mens?

Dit recept schreven we in het Nederlands en Maria las het in het Spaans. Het maakt niet uit in welke menselijke taal we communiceren. Want ons brein en dat van Maria werken op precies dezelfde manier, met stroompjes tussen hersencellen.

Als het om computers gaat, spreken we over **programmeertalen** zoals Javascript of Python. Maar welke taal je ook kiest, net als bij ons verwerken ze

instructies als universele signalen. Het brein van een robot bestaat echter uit bits en bytes.

Om een simpel gesprekje met Maria te voeren, hoef je niet te weten hoe haar hersenen werken. Hetzelfde geldt voor computers, ook daar kunnen tegenwoordig stees eenvoudiger mee communiceren.

Maar een robot heeft minder kennis van de wereld om ons heen. Daarom moeten onze instructies extra duidelijk zijn. Wat we daarmee bedoelen, ontdek je door het recept van een boterham met hagelslag aan te passen voor Olaf de Robot.

Hoe smeert Olaf de Robot een boterham met hagelslag?

We helpen je vast op weg met een lijst instructies. Je hoeft ze alleen nog in de goede volgorde te slepen voor Olaf de Robot.

- [ ] **a** Pak het pak hagelslag op
- [ ] **b** Smeer de boter met het mes op de boterham
- [ ] **c** Maak de zak met brood open
- [ ] **d** Haal met het mes wat boter uit het kuipje
- [ ] **e** Pak het mes op
- [ ] **f** Zet het pak hagelslag neer
- [ ] **g** Leg het mes neer
- [ ] **h** Strooi de hagelslag op de boterham
- [ ] **i** Maak het kuipje boter open
- [ ] **j** Pak een boterham uit de zak met brood

☐ **k** Leg de boterham op het bord

Proef op de som

Gefeliciteerd, je hebt net je eerste programma geschreven voor een computer!

Merk je op, dat er in essentie weinig verschil is tussen de twee programma's? Beide recepten zijn opgedeeld in stappen. Die voor Olaf de Robot zijn alleen wat gedetailleerder. En als we een recept schrijven voor een computer noemen we dat een **algoritme**. De stappen waar het uit bestaat heten **statements**.

Wat vind je van het programma? Denk je dat Olaf de Robot met deze instructie succesvol een boterham met hagelslag kan smeren? Of voorzie je onderdelen in het recept waar Olaf wellicht niet goed mee uit de voeten kan?

Laten we ons programma eens **runnen** om te testen hoe het onze robot vergaat. Dat betekent dat we de set instructies uitvoeren, in dit geval door de video te starten. Kijk maar wat er gebeurt.

https://www.youtube.com/embed/nTV3LXl2GGU?rel=0&controls=0&showinfo=0

Hoe gaat een robot om met menselijke instructies?

Kijk goed hoe Olaf met de instructies omgaat. Wat valt je op?

Denk bijvoorbeeld aan verschillen tussen de instructies die we hebben geschreven voor Maria en voor Olaf. Denk je dat Maria ze hetzelfde uitgevoerd zou hebben? Wat doet Olaf anders en hoe komt dat?

Over apen en vlooien

Zoals je ziet, heeft Olaf de Robot veel concretere uitleg nodig dan Maria om een boterham met hagelslag te smeren. Tijdens het runnen van de algoritmes voor de klas ging er heel wat mis. Als een computer onverwacht of onjuist gedrag vertoont, doordat de instructies niet helemaal kloppen, heet dat een error.

Het opsporen van fouten in een computerprogramma noemen we **debuggen**. Letterlijk betekent dat "ontvlooien". Het houdt in dat we met een stofkam stap voor stap door de instructies lopen om fouten op te sporen en te herstellen. Net zo lang tot alle stappen duidelijk genoeg zijn en de computer met alle mogelijke situaties overweg kan.

In ons geval moeten we er bijvoorbeeld voor zorgen dat Olaf alles dat hij oppakt na gebruik weer keurig neerlegt, dat hij alles dat hij opent, na afloop weer netjes sluit. Maar ook dat we Olaf leren om om te gaan met onverwachte omstandigheden. We moeten bijvoorbeeld controleren of Olaf het mes niet aan de scherpe maar de botte kant vasthoudt. En uitleggen wat de robot moet doen als het pak hagelslag leeg is.

Als het programma helemaal getest is en Olaf in alle gevallen een boterham met hagelslag oplevert, noemen we het *monkeyproof*. Het houdt in dat het zo duidelijk uitgelegd is, dat zelfs een aap de instructies op kan volgen zonder er een puinhoop van te maken.

Fouten en oplossingen

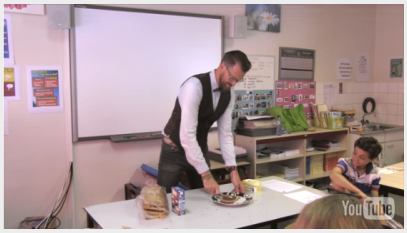
Hieronder zie je afbeeldingen van 3 errors:

1. Olaf kreeg niet alle boter met het mes uit het kuipje
2. Olaf pakte het boterkuipje niet op omdat hij de broodzak nog vast had
3. Olaf kon de boterham niet snijden zonder die vast te houden

Hoe hadden we die errors kunnen voorkomen? Onderaan de pagina vind je de oplossingen die je naar de fouten kunt verslepen.

☐  **a** Duidelijk aangeven wat de computer met welke hand moet doen

☐  **b** Goed aangeven hoe lang, hoe veel of hoe vaak de computer iets moet doen

☐		**c**	Alles dat je oppakt of opent, ook weer terugleggen of sluiten

Kennischeck

Wat is er simpeler dan het smeren van een boterham met hagelslag? In deze module hebben we het uitgelegd aan een mens en aan een computer.

Wij, Maria en Olaf spreken verschillende talen. Maar wat er precies in hun menselijke of digitale brein gebeurt, hoeven wij niet te begrijpen om eenvoudige instructies te geven.

In beide gevallen communiceerden we met een stappenplan. Maria kon met een paar complexe aanwijzingen uit de voeten, omdat wij mensen al heel veel weten en begrijpen van hoe de wereld in elkaar zit. Voor Olaf moesten we veel preciezer beschrijven wat de bedoeling was. Een robot heeft geen voorkennis, hij snapt niet wat er gebeurt en doet precies wat hij opgedragen krijgt. Eigenlijk zijn wij mensen dus veel slimmer.

In feite is dit waar het bij programmeren om draait: in kleine, duidelijke stappen aan een computer uitleggen om een grote, complexe handeling uit te voeren. Zoals je hebt gemerkt is dat vooral een kwestie van logisch nadenken. In de volgende modules gaan we aan de slag met voorbeelden in de wereld om je heen, oefeningen in de klas en opdrachten om online te programmeren.

Daarbij maak je steeds gebruik van de begrippen die we zojuist hebben behandeld. Heb jij nog goed in je hoofd wat ze betekenen?

Welke uitleg hoort bij welk begrip?

Links zie je de begrippen die we in deze module hebben behandeld, rechts de uitleg die er bij hoort. Sleep jij de juiste koppels bij elkaar?

☐	1. Programmeertaal	**a**	... is een weergave van de instructies die we aan een robot kunnen geven. Net zoals Nederlands en Spaans een weergave zijn van hoe wij denken.
☐	2. Algoritme	**b**	... is een verzameling van eenvoudige stappen die beschrijven hoe je een complexe handeling uit kunt voeren. Net als een recept.

☐	3. Statement	**c**	... is een ander woord voor het uitvoeren van je programma. Alle instructies die je onder elkaar hebt opgeschreven, worden in die volgorde achter elkaar gerealiseerd. In één keer, van de eerste tot de laatste stap, tenzij er onderweg een error optreedt ;)
☐	4. Runnen	**d**	... is een simpele opdracht voor een computer. Bijvoorbeeld: "pak op", "zet een stap" of "leg neer".
☐	5. Debuggen	**e**	... is het nalopen van alle instructies, om fouten op te sporen en te herstellen. Net zo lang tot de lijst van alle stappen klopt en compleet is.

Verdieping 1: Denk in stappen als een robot

Stapje voor stapje

Computers hebben veel meer rekenkracht dan jij en ik om ingewikkelde problemen op te lossen... en tegelijkertijd hebben ze de grootst mogelijke moeite met instructies die voor ons gesneden koek zijn. Dat komt omdat computers anders denken.

https://www.youtube.com/embed/atDJq8kw_Kg?list=PLQI9hXCcoK1SS7eN8lv_Lzj91SfE3W2Vu&controls=0&showinfo=0&rel=0

De eerste stap

Denk eens terug aan toen je vanochtend wakker werd. Wist je al precies hoe je reis naar school er uit zou zien? Bij welk stoplicht je zou moeten wachten, waar je over zou steken en of er misschien ergens een wegversperring was? Dat is onwaarschijnlijk.

Vond je het spannend om op te staan en aan je dag te beginnen? Vast niet. Want je loste eventuele uitdagingen pas op, als je ze tegenkwam. Het kostte je daarom ook maar weinig moeite. Toch heb je je bestemming bereikt.

Zoals Martin Luther King al zei: "als je vertrouwen hebt, hoef je niet de hele trap te zien om de eerste stap te zetten".

Met programmeren is het net zo. Robots denken niet vooruit. Ze handelen gewoon, stap voor stap. En dat kun jij ook, let maar op.

In deze module gaan we aan de slag met:

- **decompositie**: je leert hoe je een uitdaging op kunt delen in stappen
- **representatie**: je vertaalt de instructies naar stappen die een computer begrijpt
- **algoritme**: je schrijft je eerste computerprogramma!

Laten we beginnen met een spel.

Blind in het labyrint

Van huis naar school reizen is een eitje - en eigenlijk best wel saai. Het wordt al wat spannender als je niet kunt zien waar je bent en iemand anders je instructies geeft. Dit is het idee achter Labyrint, een

populair spelprogramma uit 1987 van de Vara.

René staat in een doolhof en moet naar het midden lopen zonder de grenzen te raken. Maar hij ziet alleen een leeg blauw vlak, de rest is voor hem onzichtbaar. Paul kan de lijnen wel waarnemen en geeft op afstand instructies.

In de volgende video zie je hoe dat gaat.

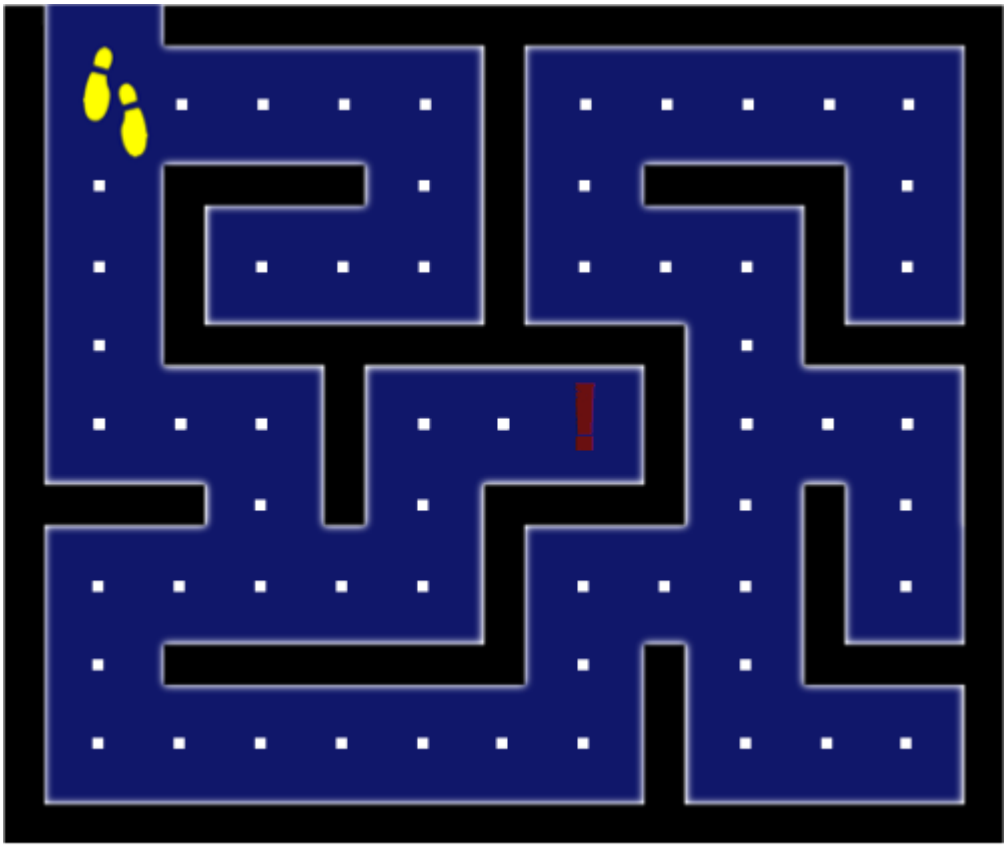
https://www.youtube.com/embed/8JfMwJJbifc?rel=0&controls=0&showinfo=0

Laten wij het eens proberen. Net als Paul, kijken we eerst welke route we gaan volgen.

Bepaal de route

Laten wij het eens proberen.

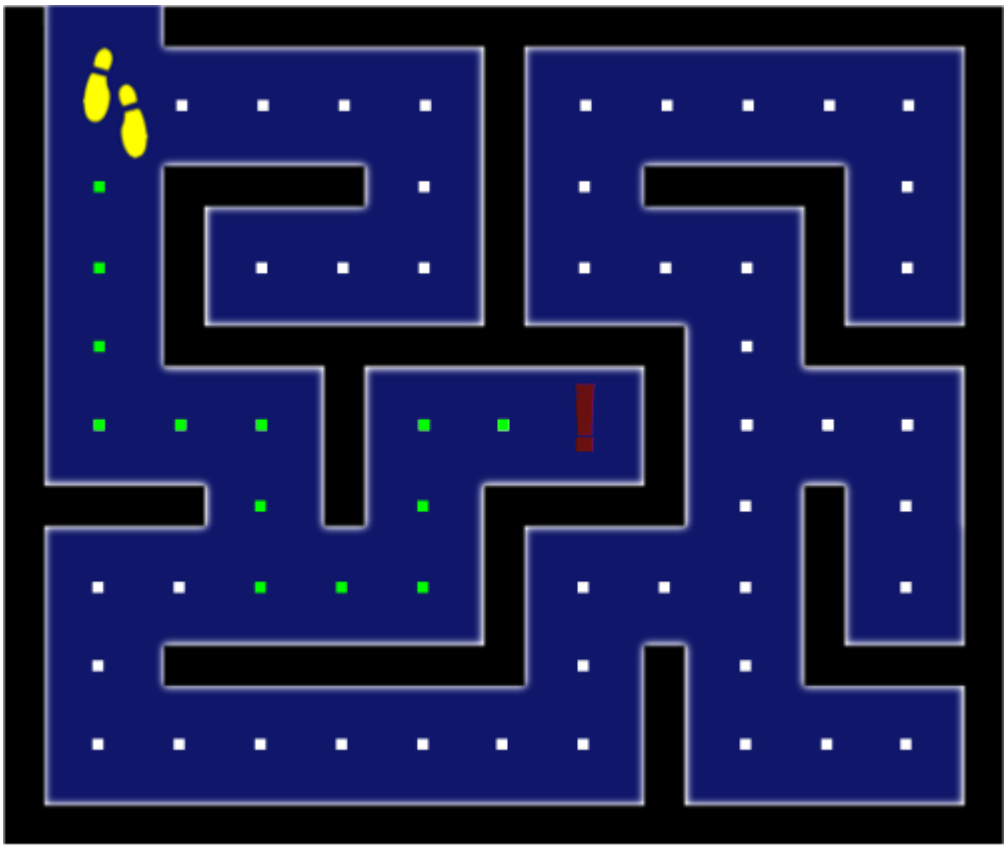
Hieronder zie je jouw labyrint, met linksboven de gele laarzen als startpunt en in het midden ons doel. Net als Paul, kijken we eerst welke route we gaan volgen. Zie je 'm al? Markeer nu de vijf witte punten waar René links- of rechtsom moet draaien om bij het uitroepteken te komen.



Aantal antwoorden: 5

Lopen, lopen, linksom

Zojuist hebben we bepaald wat de beste route is om van de start naar het doel te komen. Nu maken we instructies voor Paul om dit plan uit te voeren.



Vertaal de route naar instructies

In de afbeelding hierboven zie je nogmaals het labyrint. De juiste route is al voor je aangegeven in het groen. Iedere stip is gelijk aan een stap van de René. Zet de instructie om het doel te bereiken in de juiste volgorde.

Let goed op of je links- of rechtsom moet gaan. Als je dit lastig vindt, helpt het om je voor te stellen dat je zelf de laarzen aantrekt en je neus de juiste kant op draait.

- [ ] **a** Lopen, lopen
- [ ] **b** Lopen, lopen, rechtsom, lopen, lopen, linksom
- [ ] **c** Lopen, lopen, linksom, lopen, lopen, rechtsom
- [ ] **d** Lopen, lopen, lopen, lopen, linksom

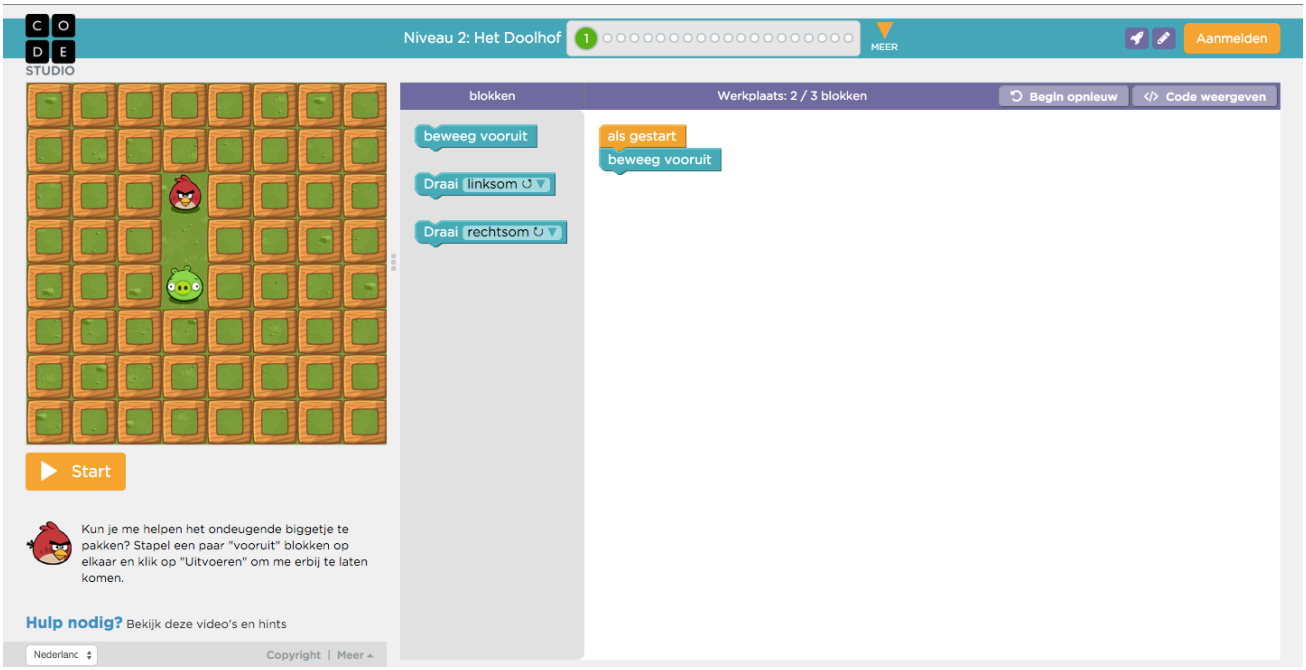
Een labyrint als programmeeromgeving

In het voorbeeld van het labyrint is RenÃ© de robot en Paul de programmeur.

Ons doolhof is een eenvoudige **representatie** van dit spel, waarbij iedere stip op de weg overeenkomt met een stap voor de robot. Als programmeertaal maken we gebruik van instructies zoals "lopen", "linksom" en "rechtsom". Op basis hiervan hebben wij een strategie bepaald om de robot naar het miden te laten lopen en die vervolgens vertaald naar een **algoritme**.

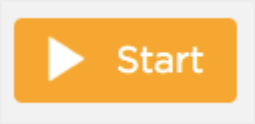
Het opdelen van een uitdaging in kleinere stappen, noemen we **decompositie**.

Ongemerkt heb je zojuist al geoefend met een aantal belangrijke concepten van computational thinking. Daarmee wordt het nu tijd voor het echte werk: biggetjes in een labyrint! Hieronder zie je een programmeeromgeving, met de onderdelen en mogelijkheden die we behandeld hebben in de Basismodule programmeren. Weet je nog hoe ze heten en hoe je ze gebruikt?



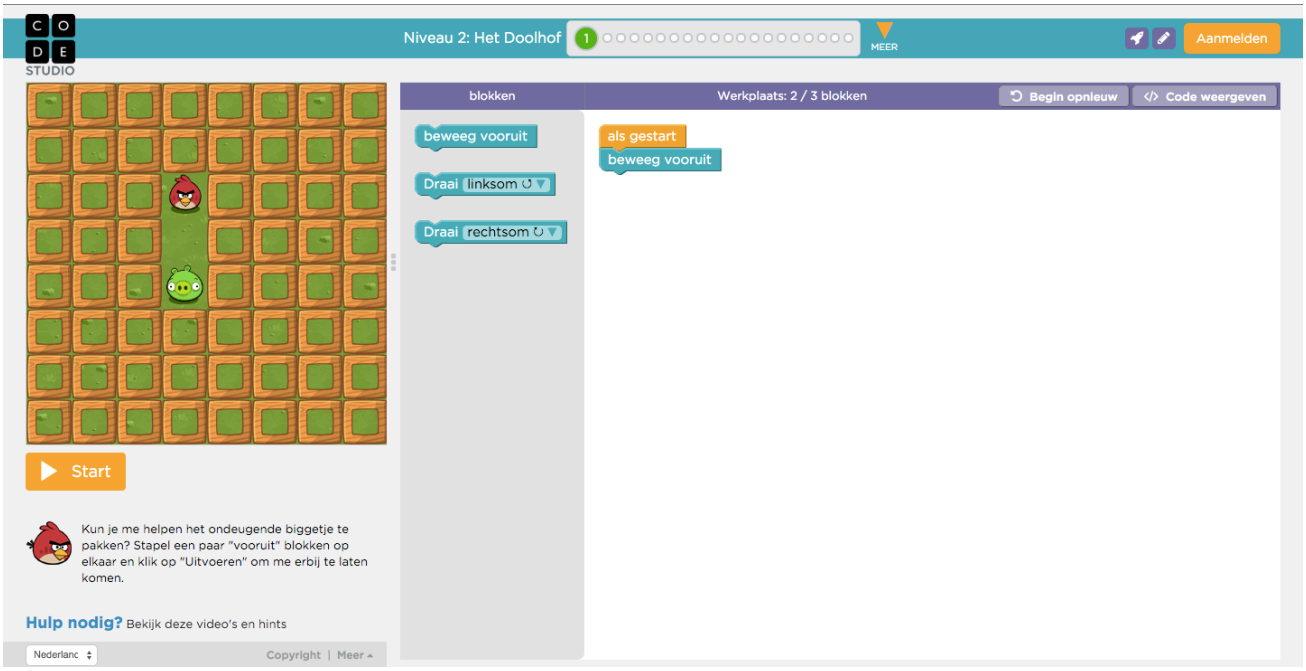
Zo werkt de programmeeromgeving

Sleep de juiste uitleg bij de onderdelen van je programmeeromgeving.

☐		**a**	Je kunt je programma "runnen" door op de startknop te drukken.
☐		**b**	Een enkele instructie in het programma, noemen we een "statement".
☐		**c**	Je maakt een "algoritme" door instructies aan elkaar te plakken onder het oranje startblok.
☐		**d**	Dit is je "programmeertaal", oftewel de set van alle mogelijke instructies die je kunt gebruiken.
☐		**e**	Is het niet goed gegaan, dan krijg je tips om te "debuggen".

Biggetjes vangen in het labyrint

Tijd voor het echte werk. Lukt het jou om de biggetjes te vangen? Klik op de afbeelding om een poging te wagen! Probeer in ieder geval 1 puzzel op te lossen, als je op dreef bent, mag je door tot en met de 5e.



Inspiratie voor jouw les!

► Lesidee in de klas

Het labyrint is heel geschikt om te oefenen in de klas. Afhankelijk van de opstelling, zijn er allerlei routes te bedenken waar je instructies voor kunt maken. Door met de tafels te schuiven kun je zelfs een echt doolhof bouwen.

1. Kies één leerling die de robot gaat spelen en één leerling die de programmeur is.
2. Positioneer de robot op het startpunt en doe hem of haar een blinddoek om.
3. De andere leerlingen kiezen een doel om naartoe te lopen, bijvoorbeeld de deur, het lichtknopje of de boekenkast.
4. De programmeur geeft hardop instructies, die de robot uitvoert.
5. Als dit goed gaat, kunnen we het wat moeilijker maken door het hele programma van te voren op het bord te schrijven en het daarna te "runnen" door de stappen één voor één voor te lezen aan de robot.
6. Bespreek met de klas wat er wel of niet goed gaat en hoe dat komt.

► Lesidee online

Een alternatieve online opdracht is Kodekraker. Dit is een ideale leeromgeving om het denken in stappen mee te oefenen.

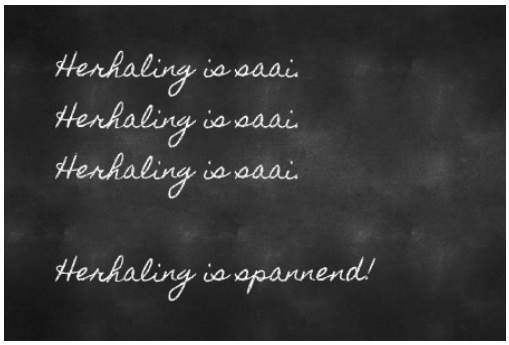
Verdieping 2: Maak het makkelijk met herhalingen

Yes, alwéér hetzelfde!

Herhaling. Voor ons mensen is het al gauw vervelend - en soms zelfs irritant. Computers kunnen er echter prima mee uit de voeten. Maar stel dat we die gegevens combineren... Ligt daar een kans?

https://www.youtube.com/embed/0V7U9_5kzyQ?list=PLQI9hXCcoK1SS7eN8lv_Lzj91SfE3W2Vu&controls=0&showinfo=0&rel=0

Herhaling: saai of spannend?



Wij mensen zien herhaling vaak als saai, sleur of zelfs straf. Denk maar eens aan de keren dat je moest nablijven op school om regels op het bord te schrijven. We zijn er niet goed in en proberen het zoveel mogelijk te vermijden met slimme uitwegen.

Met computers is het andersom. Die blinken juist uit in herhaling; het is de essentie van wat ze doen.

Maar als wij mensen aan computers opdrachten geven om te herhalen, dan wordt het spannend! Let maar eens op.

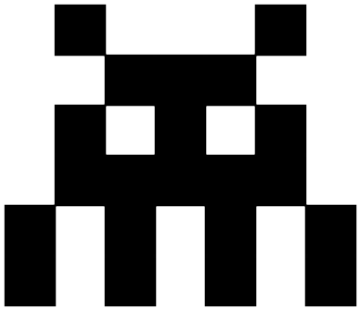
In deze module gaan we aan de slag met:

- **patronen**
- **abstractie**
- **herhalingen**

... en monsters. Ga maar verder als je durft.

Hoe teken je een monster?

Laten we beginnen met het tekenen van een monster. Voor het gemak nemen we er één exemplaar dat niet al te angstaanjagend is, zoals het voorbeeld rechts.



Bekijk het eens goed en probeer na te denken over de volgende vragen:

- hoe zou je het aanpakken om dit monster zelf te tekenen?
- wat heb je nodig om aan iemand (die jou niet kan zien!) uit te leggen hoe hij of zij dit monster kan tekenen?
- hoe zou je een computer uitleggen om dit monster te tekenen?

Wat heb je nodig om een monster te tekenen?

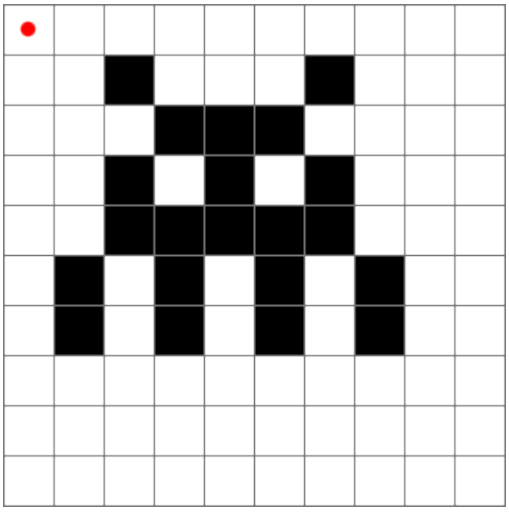
Let bij het geven van je antwoord op de volgende zaken:

- Uit welke vormen het monster is opgebouwd
- Welke materialen je nodig hebt om dit monster te (laten) tekenen
- Hoe je aan kunt geven welk deel van de tekening is ingekleurd

Je hoeft de stappen zelf niet te beschrijven, maar wat je nodig hebt om dat te kunnen doen.

Een monsterlijke weergave

We kunnen de tekening van het monster **representeren** als een vel ruitjespapier, waarop een aantal vierkanten is ingekleurd.



De vakjes geven houvast om het monster na te tekenen. Het is vooral een kwestie van goed kijken en tellen. Eigenlijk niet anders dan het labyrint uit de vorige oefening.

De vakjes geven zelfs zoveel houvast, dat je iemand anders kunt uitleggen om deze tekening te maken, zonder dat hij jou (of het monster) kan zien, door gebruik te maken van de volgende instructies:

- ga een vakje naar rechts
- ga een vakje omlaag
- ga een vakje naar links
- ga een vakje omhoog
- kleur dat vakje in

Probeer het maar eens met de volgende oefening.

Teken de eerste rijen van het monster

Als je linksboven start bij de rode stip en de vakjes volgt, welke instructies heb je dan nodig om de eerste drie rijen van het monster te tekenen? Zet ze in de juiste volgorde.

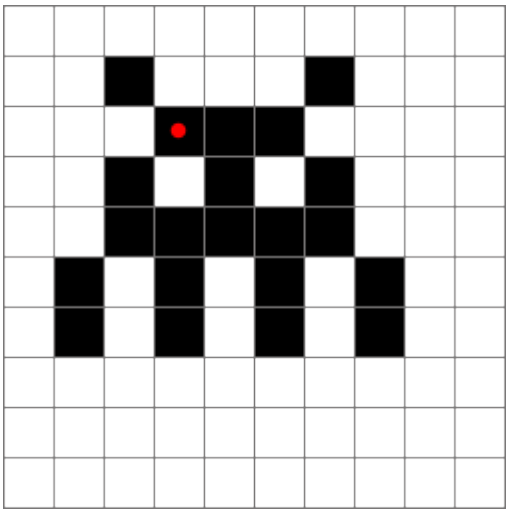
Let op: je gaat steeds verder vanuit het vakje waar je gebleven was.

☐ **a** Ga een vakje naar rechts, ga een vakje naar rechts, ga een vakje naar rechts, ga een vakje naar rechts, kleur dat vakje in.

☐ **b** Ga een vakje naar rechts, ga een vakje naar rechts, ga een vakje omlaag, kleur dat vakje in.

☐ **c** Ga een vakje omlaag, ga een vakje naar links, kleur dat vakje in, ga een vakje naar links, kleur dat vakje in, ga een vakje naar links, kleur dat vakje in.

Een symbolisch monster



In de vorige oefening heb je vast gemerkt dat zinnetjes in menselijke taal niet het handigst zijn om uit te leggen hoe we ons monster kunnen tekenen. De instructies zijn lang en het herhalen van woorden maakt ze onoverzichtelijk.

Het wordt al flink eenvoudiger door gebruik te maken van symbolen:

- → als symbool voor "ga een vakje naar rechts"
- ↓ als symbool voor "ga een vakje omlaag"
- ← als symbool voor "ga een vakje naar links"
- ↑ als symbool voor "ga een vakje omhoog"
- ◈ als symbool voor "kleur dat vakje in"

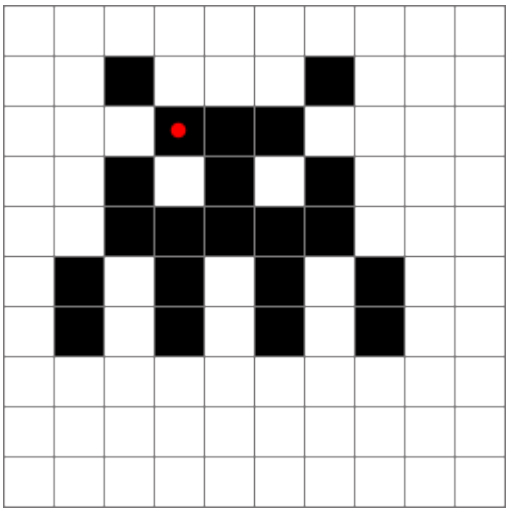
De instructies voor het tekenen van de vierde en vijfde rij kunnen we nu een stuk simpeler weergeven:

→ → ↓ ◈ → → → → ◈ ↓ ← ◈ ← ◈ ← ◈

De volgende rijen van het monster

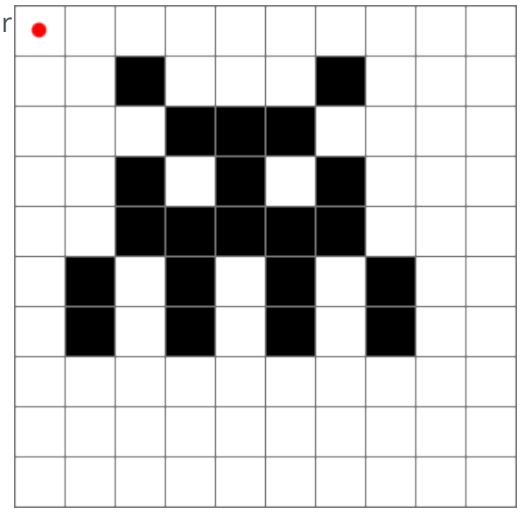
Hieronder zie je het algoritme voor de volgende rijen van het monster. We starten bij de rode stip. Selecteer het vakje waar we eindigen na het volgend van deze stappen:

← ↓ ◈ → → ◈ → → ◈ ↓ ◈ ← ◈ ← ◈ ← ◈ ← ◈ ← ← ↓



Aantal antwoorden: 1

Zombies en zonnebloemen



Als we de tekstuele instructies voor de eerste rijen vertalen naar symbolen en combineren met de rest, kunnen we de bovenste helft van ons monster als volgt representeren.

→ → ↓ ? → → → → ? ↓ ← ? ← ? ← ?

← ↓ ? → → ? → → ? ↓ ? ← ? ← ? ← ? ← ? ← ← ↓

Kijk er nog eens goed naar. Vallen je patronen op die vaker terugkomen?

De herhaling van → → (ga een vakje naar rechts, ga een vakje naar rechts) kunnen we ook weergeven als → 2 (ga twee vakjes naar rechts). Daarmee kunnen we de reeks → → → → versimpelen tot → 4 (ga vier vakjes naar rechts). De laatste herhaling van ← ← kunnen we dan opschrijven als ← 2 (ga twee vakjes naar links).

Feitelijk geven we met het getal aan dat we dezelfde instructie vaker uitvoeren.

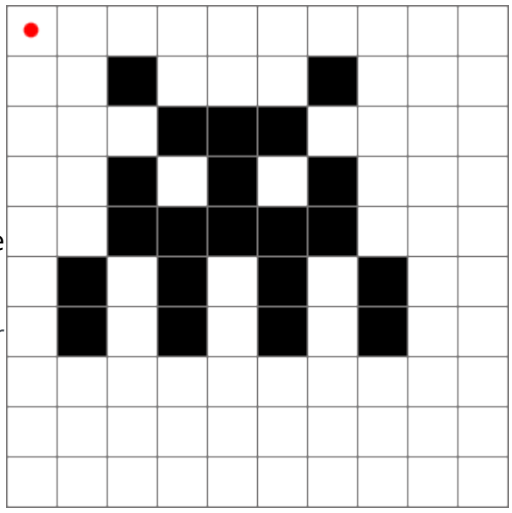
Maak je geen zorgen als het je duizelt met alle pijlen en vakjes. Er wacht een leger monsters op je om dit stap voor stap te oefenen. Klik op onderstaande afbeelding om met behulp van herhalingen de zombie naar de zonnebloem te loodsen... zonder dat je ten prooi valt aan een vleesetende plant!

Probeer ten minste één oefening, als je de smaak te pakken hebt mag je door tot en met de 8e. Klik op de afbeelding om te beginnen

Krijg jij het monster klein?

Heb je de herhaling onder de knie? Nu wordt het spannend! We stoeien nog één keer met ons monster.

Als je goed kijkt naar het algoritme voor de bovenste vijf rijen, valt je misschien nog een patroon op.



→ 2 ↓ ◈ → 4 ◈ ↓ ← ◈ ← ◈ ← ◈

← ↓ ◈ → 2 ◈ → 2 ◈ ↓ ◈ ← ◈ ← ◈ ← ◈ ← ◈ ← 2 ↓

De herhaling van ← ◈ ← ◈ ← ◈ ← ◈ (naar links, kleur in, naar links, kleur in, naar links, kleur in, naar links, kleur in) kunnen we opschrijven als (← ◈) 4 (kleur naar links vier vakjes in).

Het patroon → 2 ◈ → 2 ◈(naar rechts, naar rechts, kleur in, naar rechts, naar rechts, kleur in) kan ook gerepresenteerd worden als (→ 2 ◈) 2 (sla naar rechts steeds een vakje over en kleur de volgende in).

Het algoritme tot nu toe kunnen we ook versimpelen tot:

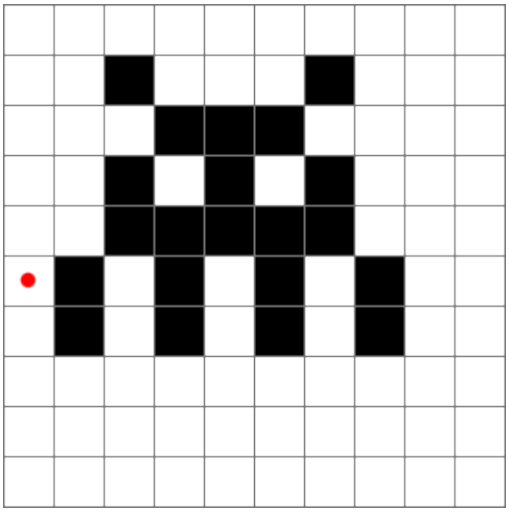
→ 2 ↓ ◈ → 4 ◈ ↓ (← ◈) 3

← ↓ ◈ (→ 2 ◈) 2 ↓ ◈ (← ◈) 4 ← 2 ↓

Laten we eens proberen of we de rest van het monster klein kunnen krijgen.

Hoe klein krijg jij de voeten van het monster?

Kijk nog eens goed naar het monster.



Welke van de onderstaande algoritmes tekenen de voeten van het monster? We beginnen weer vanaf de rode stip.

1. → ■ → → ■ → → ■ → → ■ ↓ ■ ← ← ■ ← ← ■ ← ← ■
2. → ■ (→ → ■) 3 ↓ (■ ← ←) 3 ■
3. → ■ ↓ ■ → ↑ → ■ ↓ ■ → ↑ → ■ ↓ ■ → ↑ → ■ ↓ ■ → ↑
4. (→ ■ ↓ ■ → ↑) 4
5. ↓ → ■ ↑ ■ → ↓ → ■ ↑ ■ → ↓ → ■ ↑ ■ → ↓ → ■ ↑ ■ →
6. (↓ → ■ ↑ ■ →) 4

○ 1 en 2

○ 3 en 4

○ 5 en 6

○ 1, 2, 3, 4, 5 en 6

Slimme zombies

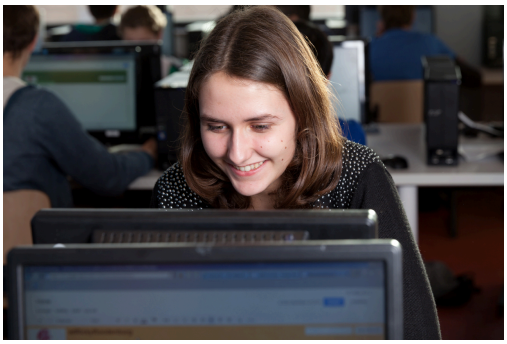
In de voorgaande oefeningen, zijn we steeds op zoek gegaan naar **patronen** in het monster. Op basis daarvan hebben we het **gerepresenteerd** als gekleurde vakjes op ruitjespapier.

Ons algoritme om het te tekenen, maakten we eerst in woorden. Weer zochten we **patronen** en maakten we het eenvoudiger met symbolen. Ook daarin zagen we **herhaling**, die we aangaven met getallen. In de laatste stap combineerden we patronen met herhalingen tot een algoritme dat steeds korter en krachtiger werd. Dat noemen we **abstractie**.

Het is Ã©Ã©n van de meest belangrijke concepten van programmeren - dat in het begin best lastig kan zijn. Laat je daar niet door van de wijs brengen, probeer de laatste oefening met de zombie maar eens. Je zult merken dat je het dan zo te pakken hebt. Klik op de afbeelding om te beginnen. Tot oefening 11 mag je met de zombie aan de slag.

Inspiratie voor jouw les!

► Lesidee in de klas



De oefening met het tekenen van patronen is heel geschikt voor in de les. In de lagere klassen kun je beginnen met tekenen van letters op een klein raster, in de hogere met steeds complexere monsters op grotere vellen. Ook is het mogelijk om gebruik te maken van meedere kleuren, door een instructie te bedenken voor de wisseling. Laat de leerlingen in groepjes werken en kijk of ze elkaars monsters kunnen tekenen op basis van de algortimes - dus zonder de afbeelding te hebben gezien. Als ze er wat langer mee bezig zijn, zul je merken dat ze zelf herhalingen ontdekken en op zoek gaan naar een eenvoudigere wijze van noteren.

► Lesidee online

Online kun je ook aan de slag met de biggen uit de vorige module (tot en met oefening 9) of de bij die honing zoekt (tot en met oefening 12).

Verdieping 3: De voordelen van voorwaarden

Instructie met een voorwaarde

Als je de vorige modules hebt doorlopen, *dan* mag je deze video kijken. Dat is de voorwaarde om deze video te te kijken.

Want door slim gebruik te maken van regels, kun je bepalen wat er wel óf juist niet mag gebeuren, in situaties die je van te voren niet kent.

https://www.youtube.com/embed/7KCvWHjxKDc?list=PLQI9hXCcoK1SS7eN8lv_Lzj91SfE3W2Vu&controls=0&showinfo=0&rel=0

Als je computer slim is, dan ...

Jij bent slim. Je kunt al denken in stappen en je hebt geoefend met herhalingen. Je weet hoe je Olaf de Robot een boterham kunt laten smeren, hoe je biggen vangt in een doolhof en hoe je een zombie naar een zonnebloem brengt. Jij kunt een computer laten doen wat je wilt, hoe gaaf is dat?

Maar... tot nu toe hebben we alleen problemen opgelost, waar we zÃ©lf het antwoord al op wisten. Zou het niet beter zijn, als we iets van onze intelligentie aan de computer over kunnen brengen? Zodat we niet alles voor hoeven te kauwen en zo'n zombie zelf kan bepalen wat er moet gebeuren?

Laat dat nu niet zo moeilijk zijn. Als je de regels van een spelletje kaarten snapt, dan ben je al halverwege.

In deze module leer je namelijk over:

- **voorwaarden**
- **herhalingen**
- **abstractie**

Als je naar de volgende pagina gaat, *dan* laten we je zien hoe dat werkt!

Zolang jij de regels mag bepalen

Je denkt het vast wel eens: "Als ik het voor het zeggen had...!"

Als het om programmeren gaat, ben jij de baas. Jij mag de regels bepalen en de computer kan alleen maar volgen. Er komt zelfs geen valsspelen aan te pas.

Laten we beginnen met een potje pesten. Dat heb je vast wel eens gespeeld. Voordat je begint, is het belangrijk dat iedereen weet hoe het spelletje gaat. Weet jij hoe de regels precies zijn?

De regels van een kaartspel

Ken jij de regels van een potje pesten? Sleep de juiste woorden op de lege plekken in de tekst.

Weet je niet meer precies hoe het gaat? Je mag best even spieken op wikikids.

We gaan uit van een kaartspel van 52 kaarten met plaatjes en jokers.

- Deel om de beurt aan iedere speler een gedekte kaart (met de rug naar boven), ☐. De rest leg je in een gedekte stapel op tafel.
- Als ☐, dan begint het spel. Herhaal de volgende stappen:
- - Als ☐, dan slaat de speler een beurt over, anders
- - Als ☐, dan moet hij 2 kaarten pakken of 5 voor een joker.
- - Als ☐, dan legt hij die op tafel, anders trekt hij een kaart van de stapel.
- - Als ☐, dan mag de speler nog een kaart op tafel leggen.
- - Als ☐, dan mag de speler de kaarten van iemand

a. de kaart een 8 is

b. de kaart een pestkaart is én de speler er zelf geen heeft

c. je de bovenste kaart op tafel legt

d. totdat iedereen 10 kaarten heeft

e. de speler een kaart heeft met dezelfde kleur of waarde

f. de kaart een 7 is

g. de kaart een 10 is

h. één van de spelers geen

anders bekijken.

- - Als ☐, dan mag de speler bepalen met welke kleur we verder spelen.
- totdat ☐.
- Als ☐, dan ben je de winnaar, anders heb je verloren.

kaarten meer heeft

i. jij als eerste al je kaarten kwijt bent

j. de kaart een Boer is

Spelen, totdat je hebt gewonnen

De spelregels beginnen met een **herhaling**: geef 10 keer een willekeurige kaart aan iedere speler. Dat is een makkie! Want als je van te voren weet hoe vaak iets moet gebeuren, kun jij dat immers al met twee simpele instructies regelen. EÃ©n stap voor de actie zelf (deel een kaart uit) en Ã©Ã©n voor de herhaling (doe dat 10 keer).

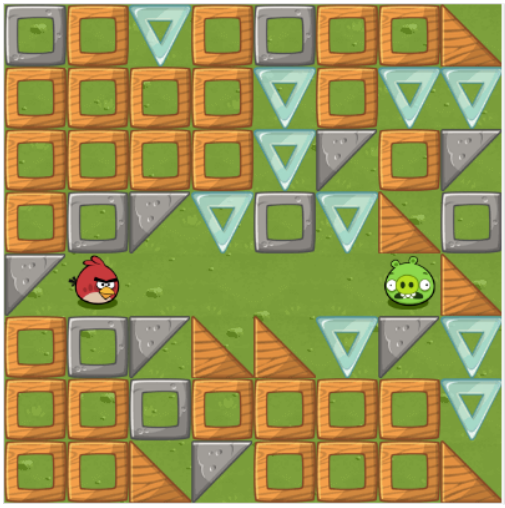
Daarna speel je net zolang totdat iemand heeft gewonnen. Hoeveel rondes daarvoor nodig zijn, kun je vooraf niet bepalen. Dat hangt af van toeval en strategie. Welke kaarten krijg je gedeeld, hoe lang duurt het potje en natuurlijk: wie zal er winnen? Juist deze factoren maken het spelletje zo leuk. Hoe vaak je aan de beurt zal zijn, is onmogelijk te voorspellen. Maar je weet wel wat de regel is: je gaat door, *totdat* iemand geen kaarten meer heeft.

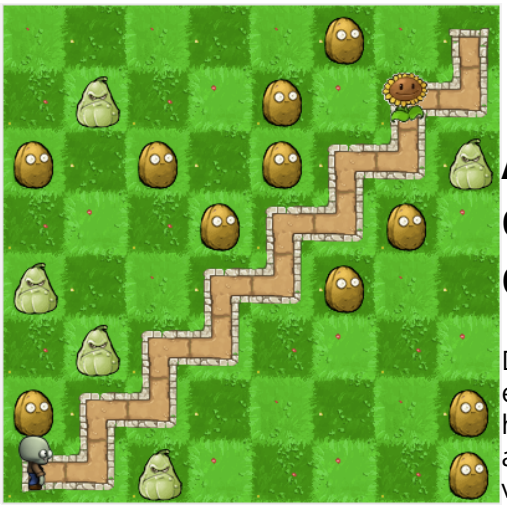
Regels met "totdat", gebruiken we heel veel. Let er maar eens op. Maak je huiswerk, *totdat* het af is. Je mag buitenspelen, *totdat* de lantaarnpalen aan zijn. Je blijft aan tafel zitten, *totdat* je bord leeg is.

Wij mensen lappen zulke regels soms het liefste aan ons laars. Maar computers kunnen er hartstikke goed mee overweg. Ze herhalen onvermoeibaar steeds opnieuw jouw stappen, net zo vaak tot ze mogen stoppen. Of dat nu 1, 8, 734 of nog veel vaker is, dat maakt niet uit. Jij hoeft alleen te bedenken wat er moet gebeuren en vooral: wanneer het genoeg is. Dat laatste noemen we een **voorwaarde**. Het is het stukje van je regel na "totdat".

Zo'n totdat-instructie maakt ons doolhof een fluitje van een cent. De boze vogel beweegt vooruit, *totdat* hij bij het biggetje is. De zombie zigzagt, *totdat* hij de zonnebloem bereikt. Dat maakt programmeren voor jou een stuk gemakkelijker, probeer het maar eens.

In de programmeeromgeving herken je een totdat-statement aan de roze kleur. Het voorbeeld rechts betekent bijvoorbeeld: *herhaal* de stap "beweeg vooruit", *totdat* je bij het biggetje bent. Klik op de afbeelding om het biggetje en de zonnebloem te bereiken met totdat-instructies. Je mag oefenen tot en met opdracht 13.





Als de bij bij de bloem is, dan ...

De meeste regels van een potje pesten hebben echter een andere vorm. Het is je vast opgevallen dat ze vaak bestaan uit "Als ... dan ... anders...". *Als* de kaart een Aas is, *dan* is de vorige speler weer aan de beurt, *anders* mag de volgende.

Regels met "als-dan-anders" gebruik je elke dag door. *Als* het mooi weer is, *dan* trek je een korte broek aan, *anders* draag je een dikke trui. *Als* het een doordeweekse dag is, *dan* ga je naar school, *anders* mag je lekker uitslapen. We gebruiken ze om keuzes te maken, meestal over dingen die we van te voren nog niet wisten.

Als een computer zulke regels kent, *dan* kan hij zelf beslissingen nemen, *anders* moet jij dat van te voren doen. Juist dat laatste is niet altijd handig en soms zelfs niet eens mogelijk. Gelukkig kennen programmeertalen ook een "als-dan-anders"-constructie. Ditmaal is de **voorwaarde** het stuk van de regel tussen "als" en "dan".

Duik maar eens in de volgende opdracht. Deze bij is op zoek naar nectar. Die kan hij echter alleen uit een bloem halen. Wij weten van te voren niet wat er achter het wolkje zit. Kun jij de bij helpen met de juiste regels?

In de programmeeromgeving herken je een als-dan-statement aan de blauwe kleur. Het voorbeeld rechts betekent bijvoorbeeld: *als* je op de bloem bent, *dan* doe je de stap "haal nectar", *anders* doe je de stap "maak honing". Het laatste deel, vanaf "anders", is overigens niet verplicht en mag ook leeg blijven.

Probeer ten minste Ã©Ã©n puzzel op te lossen met als-dan-statements, als het goed lukt, mag je verder gaan tot en met de 5e. Klik op de afbeelding om te beginnen.

De kracht zit in de combinatie

Handig zijn ze hÃ¨, die voorwaarden?

Door ze te combineren, worden ze nÃ³g krachtiger. Als je ze slim gebruikt, kun je uit *ieder* doolhof ontsnappen. Hoe groot of ingewikkeld het ook is. Zelfs als je van te voren niet weet hoe het er uit ziet of wat de juiste weg is.

Merk je op hoe voorwaarden je zo ook helpen om je programma's **abstracter** te maken?

En jij hebt nu het gereedschap in handen om precies dat te doen. Met een paar instructies loods je vurige vogels, zotte zombies en bezige bijen naar hun doel met maar een paar stappen. Kleinduimpje is er niks bij.

Kijk maar eens of jij je superkrachten kunt gebruiken om prehistorische eikels te verzamelen voordat het ijs smelt. Daar draai jij je hand toch niet meer voor om? Want door voorwaarden te gebruiken, heb je maar een paar stappen nodig. Klik op de afbeelding om te beginnen. *Als* het je lukt tot oefening 20, *dan* ben jij al een superslimme programmeur!

Inspiratie voor jouw les!

► Lesidee in de klas

Kaartspelletjes zijn ideaal om aan de slag te gaan met voorwaarden. Bekende spelletjes zoals Ezelen of Oorlog zijn heel geschikt om te oefenen en ontdekken dat ze grotendeels zijn opgebouwd uit "doe-iets-totdat"- en "als-dan-anders"-constructies. Ook hierbij is het handig om in groepjes te werken en van te voren af te spreken dat iedere instructie een van de volgende vormen heeft:

- doe <iets>
- doe <iets> totdat <voorwaarde>
- als <voorwaarde> dan <doe iets> anders <doe iets anders>*

** Merk op dat bij een "als-dan-anders"-constructie het laatste deel ook leeg mag zijn, zodat je alleen "als-dan" overhoudt.*

Als het concept duidelijk is, kunnen ze de proef op de som nemen door zelf een spelletje te bedenken, de regels op te schrijven en een ander groepje te laten testen.

► Lesidee online

Online kun je verder met de zombie en de zonnebloem (tot en met oefening 10) of de boer die modder

schept (tot en met oefening 11).

Verdieping 4: Patronen herkennen (functies)

De sleutel naar succes

Hoe maak je snel een krachtig computerprogramma? Oefenen helpt. Maar deze truc is nog handiger! Herken de patronen en gebruik ze opnieuw.

https://www.youtube.com/embed/FSQRDqQiInc?list=PLQI9hXCcoK1SS7eN8lv_Lzj91SfE3W2Vu&controls=0&showinfo=0&rel=0

Kijk mama, zonder handen!

Herinner jij je nog hoe het was om te leren fietsen? Eerst had je iemand nodig om uit te leggen hoe je op moest stappen en met je mee te lopen om te voorkomen dat je viel.

Nu heb je vast geen zijwieltjes meer. Je hebt de handelingen van het fietsen al zo vaak gedaan, dat ze routine zijn geworden. Zelfs als je 's ochtends nog niet helemaal wakker bent, kom je zonder kleerscheuren in de klas. Sterker nog, je hebt zelfs denkkracht over om nog wat woordjes te oefenen in je hoofd, of gezellig te kletsen met een vriend(in).

Je staat er nooit bij stil, maar eigenlijk is dat best indrukwekkend. Zeker als je nog even terugdenkt aan die eerste keer. Wat leek dat toen moeilijk! Nu hoeft niemand je meer te vertellen hoe het precies moet. Vaak weet je niet eens meer precies welke stappen je volgt. Je stapt gewoon op en fietst weg.

Zouden we dat ook kunnen bereiken met computers? Laten we het proberen!

In deze module gaan we aan de slag met:

- **(sub)routines**
- **patronen**
- **herhalingen**
- **abstractie**
- **variabelen**

Schrik niet van deze moeilijke woorden, want we gaan gewoon zingen en tekenen. Doe je mee?

Je tante in Marokko

Laten we beginnen met een liedje: 'Ik heb een tante in Marokko'. Ken je dat? Luister maar eens naar Jelle, die het voorzingt.

In de opdracht hieronder hebben we de tekst voor je afgedrukt. Die is best lang voor een kinderliedje! Toch heb je het zo geleerd. Hoe kan dat?

Als je goed luistert, valt je op dat het liedje heel veel herhalingen bevat. De losse regels die vaker voorkomen hebben we al voor je vervangen door (x2).

Maar er is nog meer...

Ik heb een tante in Marokko

Kijk eens goed naar de tekst. Zie je groepjes regels die vaker terugkomen? Zet er een prikker op door ze aan te klikken.

Ik heb een tante in Marokko en die komt. (x2)
Ik heb een tante in Marokko, een tante in Marokko,
een tante in Marokko en die komt.

Zing ik a ja, jippie, jippie, jee. (x2)
Zing ik a ja, jippie, a ja jippie,
a ja, jippie, jippie, jee.

En ze komt op twee kamelen als ze komt. (x2)
En ze komt op twee kamelen, ze komt op twee kamelen,
ze komt op twee kamelen als ze komt.

Zing ik a ja, jippie, jippie, jee. (x2)
Zing ik a ja, jippie, a ja jippie,
a ja, jippie, jippie, jee.

En we drinken coca cola als ze komt. (x2)
En we drinken coca cola, we drinken coca cola,
we drinken coca cola als ze komt.

Zing ik a ja, jippie, jippie, jee. (x2)
Zing ik a ja, jippie, a ja jippie,
a ja, jippie, jippie, jee.

Maar dan gaat ze met het treintje weer naar huis. (x2)
Maar dan gaat ze met het treintje, dan gaat ze met het treintje,
dan gaat ze met het treintje weer naar huis.

Zing ik a ja, jippie, jippie, jee. (x2)
Zing ik a ja, jippie, a ja jippie,
a ja, jippie, jippie, jee.

Zing ik a ja, jippie, jippie, jee. (x2)
Zing ik a ja, jippie, a ja jippie,
a ja, jippie, jippie, jee.

Zing ik a ja, jippie, jippie, jee. (x2)
Zing ik a ja, jippie, a ja jippie,
a ja, jippie, jippie, jee.

Aantal antwoorden: 6

De functie van een refrein

De stukjes in een liedje die vaker terugkomen, noemen we in de muziek een refrein. We hadden onze tekst veel korter weer kunnen geven. Hier boven zie je hoe we in het blauw met "Refrein:" aangeven dat hier een paar regels volgen, die we later zullen herhalen. Na het tweede couplet hoeven we het niet opnieuw uit te schrijven, maar kunnen we volstaan met "Refrein".

Ik heb een tante in Marokko en die komt. (x2)
Ik heb een tante in Marokko, een tante in Marokko,
een tante in Marokko en die komt.

Refrein:
Zing ik a ja, jippie, jippie, jee. (x2)
Zing ik a ja, jippie, a ja jippie,
a ja, jippie, jippie, jee.

En ze komt op twee kamelen als ze komt. (x2)
En ze komt op twee kamelen, ze komt op twee kamelen,
ze komt op twee kamelen als ze komt.

Refrein

En we drinken coca cola als ze komt. (x2)
En we drinken coca cola, we drinken coca cola,
we drinken coca cola als ze komt.

Refrein

Maar dan gaat ze met het treintje weer naar huis. (x2)
Maar dan gaat ze met het treintje, dan gaat ze met het treintje,
dan gaat ze met het treintje weer naar huis.

Refrein (x3)

Er is niemand die het woord "refrein" gaat zingen, maar we weten allemaal wat het betekent en hoe het werkt. Als je het nog een paar keer oefent, krijg je het zelfs amper meer uit je hoofd. Voor je het weet zit je het te fluiten op de fiets, zonder dat je het in de gaten hebt. Het refrein is **routine** geworden.

Een refrein is een **patroon** in muziek, dat vaker terug komt. We leggen Ã©Ã©n keer uit hoe het gaat en in het vervolg keer je steeds terug naar die uitleg.

Simpel toch? Waarschijnlijk vertellen we je hier niets nieuws. Maar wist je dat je met refreinen ook kunt tekenen?

Hoe vorm je een vorm?

Met programmeren kun je veel meer dan eikels zoeken met een eekhoorn. Wat dacht je bijvoorbeeld van kunst?

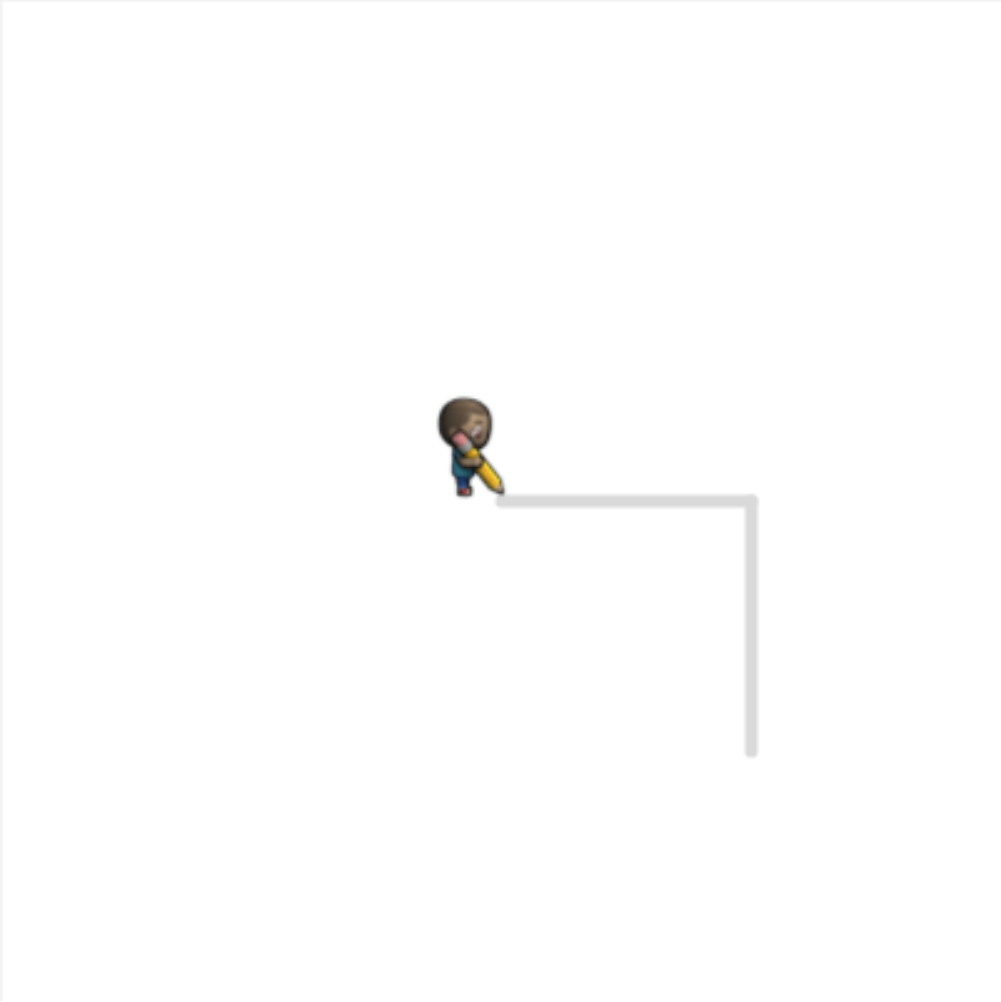
Kijk maar eens naar de volgende oefening, waarin een artistiek ventje vierkanten en driehoeken tekent.

Dat doet hij met twee nieuwe instructies:

- De kunstenaar kan voor- of achteruit bewegen over een bepaalde afstand in pixels
- De kunstenaar kan links- of rechtsom laten draaien met een bepaald aantal graden

Weet je niet zeker hoe ver de kunstenaar moet gaan of draaien? Voer je programma dan gewoon even uit om te testen wat er gebeurt. Met het schuifje tussen de schildpad en de haas kun je bovendien zelf bepalen hoe langzaam of snel het gaat.

Lukt het jou om hiermee vierkanten en driehoeken te tekenen? Probeer in ieder geval tot en met oefening 6 te komen. Klik op de afbeelding om te beginnen aan je eerste digitale kunstwerk!



Van refrein naar functie

In de vorige opdracht heb je flink wat vormen getekend, steeds op dezelfde manier. Je kunt een herhaling gebruiken om drie of vier keer rechtsaf te gaan. Maar in iedere opdracht begon je weer opnieuw.

Je hebt intussen vast gemerkt dat programmeurs lui zijn. Als ze te vaak hetzelfde doen, dan maken ze het makkelijker. Als iets ingewikkeld, foutgevoelig of inefficiÃ«nt is, dan maken ze het simpeler. Ironisch genoeg noemen ze dat met een moeilijk woord **abstractie**...

Voel je al waar we naartoe willen? Jouw driehoeken en vierkanten zijn eigenlijk refreinen! Het zou immers veel handiger zijn als je ze Ã©Ã©n keer beschrijft en er daarna alleen nog maar naar hoeft te verwijzen. De refreinen van computers worden ook wel **(sub)routines** genoemd, of **functies**.

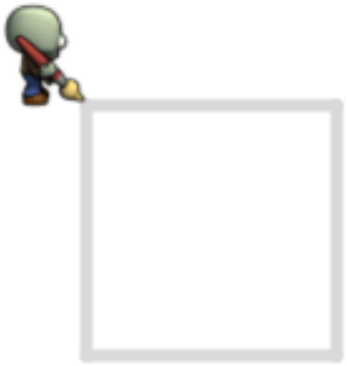
In de programmeeromgeving herken je ze aan een lichtgrijs blok. Iedere functie heeft een eigen naam, die je ziet staan in de groene balk en beschrijft wat de functie doet. Dat is handig wanneer je er meer dan Ã©Ã©n gebruikt. Bijvoorbeeld "teken een vierkant" en "teken een driehoek".

In het grijze blok staat de *uitleg* van de functie. Net als in een liedje moet je zelf aangeven wanneer het refrein gezongen moet worden. Dat doe je met een hele eenvoudige instructie: "teken een vierkant". Als de computer zo'n groene instructie tegenkomt, gaat hij op zoek naar de uitleg van die functie en voert die stappen uit.

Als je eenmaal in een functie hebt uitgelegd hoe je een vierkant tekent, dan kun je het daarna zo vaak gebruiken als je wilt. Je hoeft er niet meer over na te denken. Je hoeft zelfs niet precies te begrijpen hoe het werkt. Als je een functie ziet met de naam "teken een cirkel", heb je vast al een aardig idee wat die functie doet toch?

Probeer het maar eens met de volgende opdracht. Omdat je nu al aardig wat stappen hebt geleerd, zijn ze onder het kopje "Blokken" in categorieÃ«n opgedeeld. Onder "Acties" vind je de stappen om te bewegen en te draaien, onder "Functies" de stap om een functie aan te roepen en onder "Wiskunde" staan je herhalingen.

Kun jij met functies de gevraagde figuren tekenen? Soms ben je al met Ã©Ã©n stap klaar! Probeer tot en met de 5e oefening te komen. Je kunt meteen beginnen met de functies door op de afbeelding te klikken.



Cowboy Billy Boem

Laten we nog eens naar een liedje kijken. Ken je '[Cowboy Billy Boem]'? Het is opgebouwd uit een patroon dat steeds hetzelfde is, maar nét even anders.

En wie rijdt er op z'n paard door de prairie?
Het is Cowboy Billy Boem, door de boeven zeer gevreesd.
Er is nooit in het Wilde Westen 'n cowboy geweest,
die zo dapper was als Cowboy Billy Boem

En van je hotsie, knotsie, knetter, van je jippie, jippie, jee,
Maar zijn paard was zeer vermoeid en die wou niet verder mee.
Maar hij moest de boeven vangen dus nam hij een ander beest
en nu mag je zelf bedenken wat voor 'n beest dat is geweest.

En wie rijdt er op z'n koe door de prairie?
Het is Cowboy Billy Boem, door de boeven zeer gevreesd.
Er is nooit in het Wilde Westen 'n cowboy geweest,
die zo dapper was als Cowboy Billy Boem.

En van je hotsie, knotsie, knetter, van je jippie, jippie, jee,
Maar zijn koe was zeer vermoeid en die wou niet verder mee.
Maar hij moest de boeven vangen dus nam hij een ander beest
en nu mag je zelf bedenken wat voor 'n beest dat is geweest.

En wie rijdt er op z'n schaap door de prairie?

Het is Cowboy Billy Boem, door de boeven zeer gevreesd.
Er is nooit in het Wilde Westen 'n cowboy geweest,
die zo dapper was als Cowboy Billy Boem.

En van je hotsie, knotsie, knetter, van je jippie, jippie, jee,
Maar zijn schaap was zeer vermoeid en die wou niet verder mee.
Maar hij moest de boeven vangen dus nam hij een ander beest
en nu mag je zelf bedenken wat voor 'n beest dat is geweest.

Enzovoort.

Cowboy Billy Boem

Kijk eens goed naar de tekst van 'Cowboy Billy Boem' en beantwoord de volgende twee vragen:

- wat is het grootste patroon dat je ziet?
- wat is daarin de kleinste afwijking?

Hetzelfde maar dan anders

De tekst van 'Cowboy Billy Boem' is in essentie steeds hetzelfde. Als je aan het einde bent beland, verzin je een nieuw beest en begin je weer opnieuw. Het paard vervang je simpelweg door een ander dier.

En wie rijdt er op z'n _____ door de prairie?
Het is Cowboy Billy Boem, door de boeven zeer gevreesd.
Er is nooit in het Wilde Westen 'n cowboy geweest,
die zo dapper was als Cowboy Billy Boem

En van je hotsie, knotsie, knetter, van je jippie, jippie, jee,
Maar zijn _____ was zeer vermoeid en die wou niet verder mee.
Maar hij moest de boeven vangen dus nam hij een ander beest
en nu mag je zelf bedenken wat voor 'n beest dat is geweest.

Stel nu dat we een klein en een groot vierkant willen tekenen. En als we echt iets moois gaan maken, hebben we misschien ook wel een heel klein, een middelmatig en een heel groot vierkant nodig. Het kan toch niet de bedoeling zijn om voor ieder formaat een functie te schrijven?

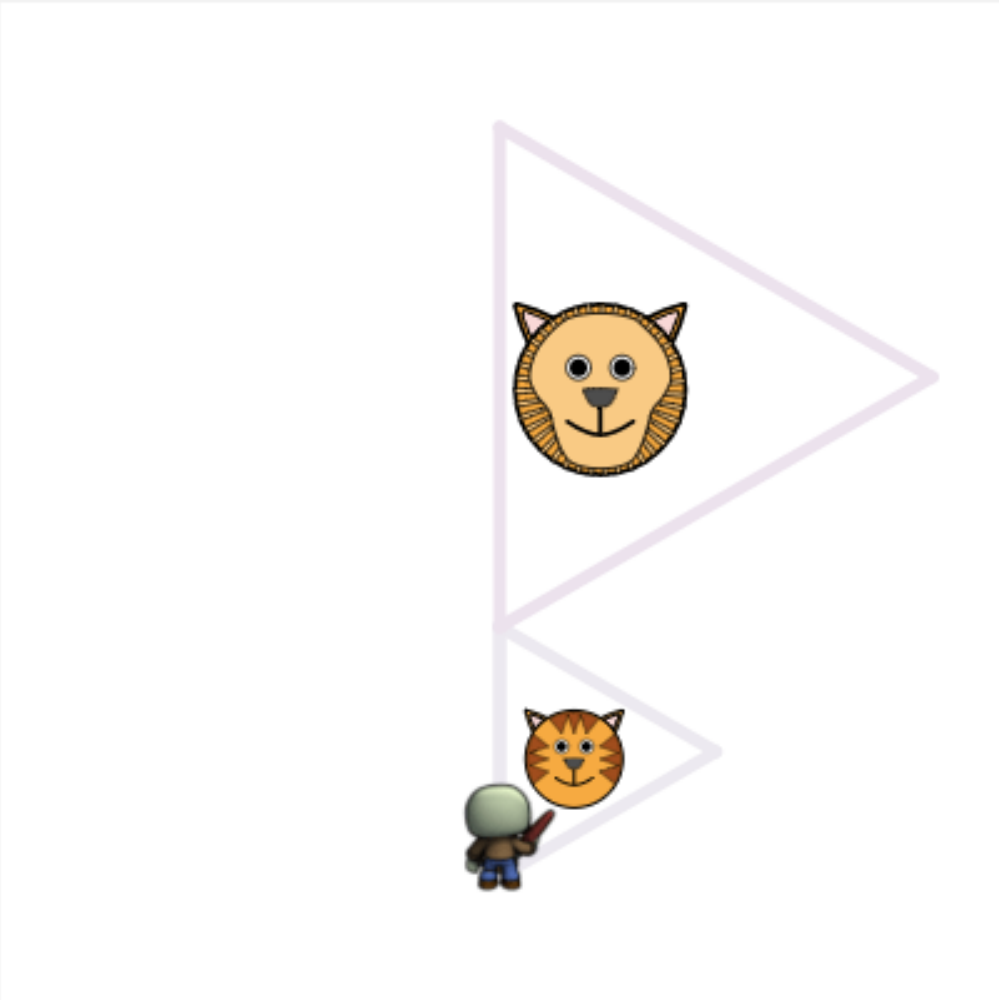
Herhalingen zijn handig als je patronen *precies* hetzelfde zijn. Maar met functies kun je ook uit de voeten als ze *bijna* hetzelfde zijn.

In de programmeeromgeving herken je zo'n functie aan het blauwe sterretje. In dit geval zie je aan de naam dat je een vierkant kunt tekenen *met een bepaalde lengte*. Hoe lang dat is, wordt niet genoemd. Want in de stap zelf lees je "ga vooruit met *lengte* pixels".

Als we de opdracht geven om een vierkant te tekenen, geven we aan wat de lengte is. Het programma hiernaast, maakt eerst een klein vierkant met lente 50 en daarna een grote met lengte 200. Hoe we het vierkant tekenen, is steeds *precies hetzelfde*. Maar de lengte van de zijden is *variabel*.

Wanneer we het hebben over een "lengte" met de waarde "50" of een "kleur" met de waarde "groen" of een andere combinatie van een naam en een waarde, noemen we dat een **variabele**. In de programmeeromgeving herken je die aan de paarse kleur.

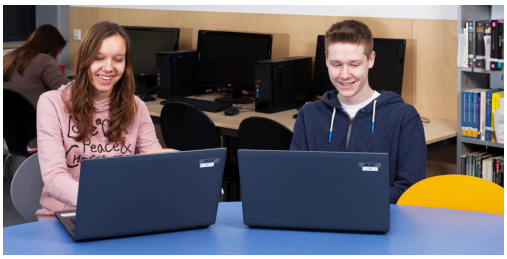
Daarmee zijn we aanbeland bij de laatste opdracht: kun jij vierkanten, driehoeken en zelfs huizen tekenen van verschillende grootte? Klik op de afbeelding om te starten - en je mag door tot het eind!



Inspiratie voor jouw les!

► Lesidee in de klas

Veel kinderliedjes bevatten herhalingen al dan niet met variatie. Van 'Ik zag twee beren' tot 'Old Mac Donald', of 'Een potje met vet'.



Maar kijk ook eens verder dan muziek. Patronen kom je overal tegen. Denk bijvoorbeeld aan technieken in sport of choreografie in dans. Met name die laatste is heel geschikt om herhalingen in te herkennen, functies te maken van bepaalde *moves*, met of zonder variatie.

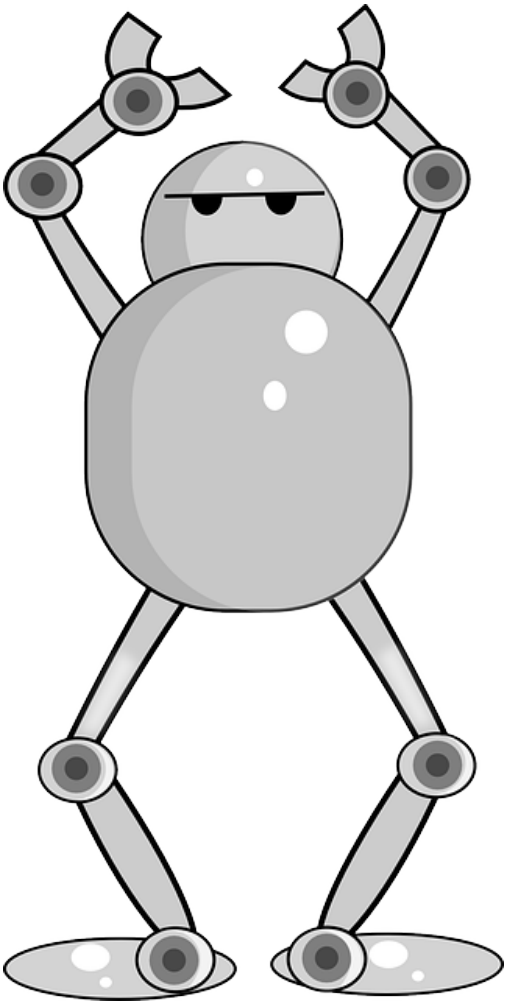
► Lesidee online

Online kun je verder oefenen met patronen met de kunstenaar (tot het einde) of Anna en Elsa (idem).

...en dan wordt het routine!

Gefeliciteerd!

Je hebt alle modules doorlopen en de belangrijkste begrippen van computational thinking onder de knie. Merk je hoe het denken als een computer routine is geworden...? Vanaf nu mag jij jezelf een programmeur noemen!




Certificate of Completion
This certificate is awarded to
jou, als programmeur
for successful completion of
The Hour of Code
and demonstrating an understanding of
the basic concepts of Computer Science.
Hadi Partovi
Hadi Partovi, Co-founder and Chief Executive Officer, Code.org
C O
D E
www.code.org
ANYONE CAN LEARN COMPUTER SCIENCE • CODE.ORG
KEEP UP THE
GREAT
WORK
KEEP LEARNING
To learn beyond your first hour, visit Code.org
Diane Tang and Ben Smith made the generous gift to sponsor your learning.

Voorbeeldlessen

Hoe nu verder?

Stel: je wilt met *computational thinking* aan de slag in jouw klas. Hoe pak je dat eigenlijk aan? Allard helpt je op weg met handige tips!

https://www.youtube.com/embed/ulA4GFZ-SGI?rel=0&controls=0&showinfo=0&rel=0

Een overzicht

Hopelijk hebben we je enthousiast gemaakt over *computational thinking*? Om direct met je leerlingen aan de slag te gaan in de klas, bieden we je een overzicht van kant-en-klare voorbeeldlessen op verschillende niveaus. Veel succes!

Lessen over *computational thinking* zonder computer

In deze lessen van Codekinderen gaan leerlingen aan de slag met *computational thinking*, maar zonder gebruik te maken van een computer. De principes en de werking van een computer worden zo tastbaar.

Lessen over *computational thinking* zonder programmeren

In deze lessen van Codekinderen maken leerlingen 'gewone' objecten interactief. Ze leren dat *computational thinking* een vaardigheid die je in staat stelt iets te bedenken en zelf te maken.

Lessen over *computational thinking* met programmeren

In deze lessen van Codekinderen maken leerlingen spelenderwijs hun eerste computerprogramma's.

Complete lesprogramma's en programmeeromgevingen

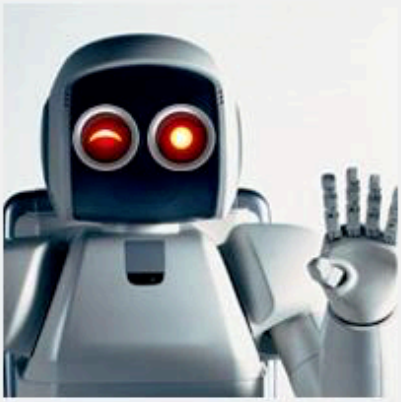
Op studio.code.org zijn bovendien volledige lesprogramma's beschikbaar met off- en online opdrachten voor leerlingen van verschillende leeftijden en niveaus.

Leerlijn programmeren

En wil je concreet in je eigen school aan de slag met het ontwikkelen van een leerlijn 'Computational thinking', dan kun je handig gebruik maken van de door Kennisnet en SLO ontwikkelde leerlijn Programmeren: http://kn.nu/leerlijnprogrammeren

Bee-Bot

Ko de Kraker



RoboMind

Belangrijke begrippen

Abstractie: het verkleinen van complexiteit en het vergroten van efficiëntie van een representatie of algoritme voor een probleem, vaak door irrelevante informatie en herhaling van instructies te reduceren.

Algoritme: een reeks opeenvolgende instructies om een probleem op te lossen of doel te bereiken, vergelijkbaar met een recept om een maaltijd te bereiken.

Debuggen: het opsporen en oplossen van fouten in een computerprogramma.

Decompositie: het opdelen van een probleem in kleinere delen die gemakkelijker te bevatten en op te lossen zijn.

Error: treedt op wanneer een computer onverwacht of onjuist gedrag vertoont, doordat de instructies niet helemaal kloppen.

Herhaling: een deel van een computerprogramma dat meerdere keren herhaald wordt, ook wel lus of iteratie genoemd.

Patroon: een beschrijving, of weergave van een probleem of oplossing die meerdere keren herhaald of hergebruikt kan worden .

Programmeertaal: formele taal die gebruikt wordt om instructies te communiceren aan een computer.

Representatie: de weergeven van een probleem of oplossing in een andere vorm, bijvoorbeeld een tekening, tekst, reeks symbolen of instructies.

Runnen: het uitvoeren van een computerprogramma.

Statement: een eenvoudige instructie voor een computerprogamma om een actie uit te voeren.

Subroutine: een reeks instructies die meerdere keren uitgevoerd wordt tijdens het runnen van een computerprogramma.

Variabele: wordt gebruikt om informatie te bewaren, met een naam en een waarde, zoals "leeftijd" in combinatie met "8".

Voorwaarde: een vergelijking om te bepalen of iets wel of niet gedaan moet worden, bijvoorbeeld "als... dan ..." of "zolang ... doe... " of "herhaal ... tot ....".

Over dit lesmateriaal

Colofon

Auteurs	Jan-Bart Vreede ; Kennisnet LleG ; Sander van Acht ; Lalibel Mohaupt
Team	Stichting Kennisnet j.devreede@kennisnet.nl

Laatst gewijzigd	30 augustus 2022 om 08:30

Aanvullende informatie over dit lesmateriaal

Van dit lesmateriaal is de volgende aanvullende informatie beschikbaar:

Leerniveaus	VMBO gemengde leerweg, 2, VWO 2, VMBO gemengde leerweg, 3, VMBO basisberoepsgerichte leerweg, 1, VMBO theoretische leerweg, 1, HAVO 1, VMBO gemengde leerweg, 1, VMBO theoretische leerweg, 2, PO groep 8, VMBO basisberoepsgerichte leerweg, 2, PO groep 7, VWO 1, HAVO 3, VWO 3, VSO, Speciaal basisonderwijs, VMBO theoretische leerweg, 3, VMBO basisberoepsgerichte leerweg, 3, SO, HAVO 2
Leerinhoud en doelen	Techniek breed, Ict-route, Maatschappijleer, Techniek, Computervaardigheden/ICT
Eindgebruiker	leraar
Studiebelasting	4 uur en 0 minuten
Trefwoorden	abstraheren, analyseren, computational thinking, creativiteit, probleemoplossend vermogen, programmeren, representeren, ruimtelijk inzicht, samenwerken, structureren

Bronnen