Blok 1 van Scratch naar Python

- Scratch 1
- Scratch 2
- Van Scratch naar Python
- Pak de Kaas
- Kennis-check Blok 1

Scratch 1

*Deze pagina biedt een **stap-voor-stap handleiding** om te leren programmeren in Scratch door het bouwen van een **doolhofspel**. Je volgt de lessen door middel van **uitlegvideo's**, maakt **opdrachten direct in Scratch** en leert daarbij belangrijke programmeerconcepten zoals beweging, loops, if-then-else statements, variabelen en het resetten van de speltoestand. De cursus begeleidt je van de introductie en het opzetten van het startproject tot het afronden van je eigen unieke versie van het spel en het reflecteren op wat je hebt geleerd.*

Introductie en start

Welkom! We gaan je stap voor stap helpen om Scratch te begrijpen, zodat je zelf leert programmeren. Maar wat gaan we eigenlijk doen?

Wat kun je verwachten?

- Je bekijkt korte uitlegvideo's van Felienne, een docent aan de TU Delft.
- Je maakt opdrachten direct in Scratch.
- Je leert stap voor stap hoe je een eigen doolhofspel bouwt.

Als je nog meer wilt weten van deze module dan kan je hier meer informatie vinden.

Spelregels

- Je bouwt het spel stap voor stap op. Daarna maak je je eigen variant.
- Het inleveren van andermans werk is fraude en kan ertoe leiden dat je een extra module moet doen.
- Maar het is ook écht leuk om je eigen spel te maken! Kom je er even niet uit, dan mag je natuurlijk wel een medestudent om hulp vragen.
- Voor deze opdrachten gebruiken we (nog) geen AI — dat komt later.
- Bij elke opdracht lever je iets in: soms is dat je code (of een screenshot daarvan), soms een antwoord op één of meer vragen. Ook die antwoorden schrijf je in je eigen woorden. Door zelf na te denken over de vragen, begrijp je de stof beter — en zo word je uiteindelijk een goede softwaredeveloper.

Voorbereiding: Open dit project in Scratch:

https://scratch.mit.edu/projects/96709199/#editor

(of download het bestand hier: Doolhof Start.sb3)

Taal aanpassen

Standaard staat de taal op English, deze module is in het **Nederlands** dus als je de taal wil aanpassen dan kan dat onder **settings** - **laguage**.

Je zou dit startscherm moeten zien:

image.png
Image not found or type unknown

Bekijk de uitlegvideo

In deze video legt Felienne uit wat je gaat bouwen en hoe je aan de slag kunt. Let goed op de uitleg over hoe Scratch werkt.

https://www.youtube.com/embed/JcIVNv4VZv8?si=8FT4Hu648QqKqrCY

Alternatieve link: Klik hier als de video hierboven niet werkt

□□ Opdracht

Beschrijf in één zin in je eigen woorden wat je denkt dat je gaat maken.

□□ Inleveren

Typ je zin in het tekstvak hieronder. Gebruik je eigen woorden en denk aan wat je in het spel gaat bouwen.

Stap 1 – Beweging

We beginnen met de besturing van je hoofdpersonage, Giga. In deze stap leer je hoe je Giga kunt laten bewegen met de pijltjestoetsen.

Wat leer je?

- Bewegen over de X-as (links en rechts).

- Bewegen over de Y-as (omhoog en omlaag).

Bekijk de uitlegvideo

https://www.youtube.com/embed/IGtsy24vbiE?si=ZYnQwgoNBWrrlWw_

Alternatieve link: Bekijk de video via HiDrive

□□ Opdracht

Hoeveel richtingen kan jouw poppetje bewegen aan het einde van deze stap?

□□ Inleveren

Vul in het tekstvak in hoeveel richtingen jouw sprite beweegt.

Stap 2 – De lus (loop)

In deze stap leren we hoe je een herhaling maakt met een lus. Hiermee kun je code telkens opnieuw uitvoeren zolang een bepaalde voorwaarde klopt.

Wat leer je? Je gebruikt een herhaalblok om acties te blijven herhalen.

Bekijk de uitlegvideo

https://www.youtube.com/embed/OFzSUt23Lho?si=r63G16qrEFFOA2YY

Alternatieve link: Bekijk de video via HiDrive

□□ Opdracht

Leg in je eigen woorden uit wat een lus (loop) doet in een programma.

□□ Inleveren

Typ je uitleg over de lus in het tekstvak.

Stap 3 – Als-dan-anders

We voegen nu logica toe: wat moet er gebeuren als iets gebeurt? Je leert nu het *als-dan-anders*-blok gebruiken.

Bekijk de uitlegvideo

https://www.youtube.com/embed/6f9TeuFZshM?si=XWdXdtFBhWh0XuD8

Alternatieve link: Bekijk de video via HiDrive

□□ Opdracht

Wat doet een als-dan-anders (if-then-else)-blok in je code? Leg het uit in je eigen woorden.

□□ Inleveren

Typ je uitleg in het tekstvak.

Stap 4 – Terug naar startpositie

Als Giga een muur raakt, willen we dat hij teruggaat naar het begin. In deze stap leer je hoe je zijn positie reset.

Bekijk de uitlegvideo

https://www.youtube.com/embed/lRX_n2S0Hwk?si=YNeZtufi6l11jnJX

Alternatieve link: Bekijk de video via HiDrive

□□ Opdracht

Welke coördinaten (X en Y) gebruikt jouw spel om Giga terug te zetten naar de start?

□□ Inleveren

Typ de X- en Y-positie in het tekstvak.

Stap 5 – De sleutel

Je spel is bijna klaar! In deze stap voeg je een sleutel toe. Als Giga deze aanraakt, is het spel gewonnen.

Bekijk de uitlegvideo

https://www.youtube.com/embed/UoqZhd4ggn0?si=j8a5GLqdgiNfATHc

Alternatieve link: Bekijk de video via HiDrive

□□ Opdracht

Maak een screenshot van je hele browser waarin je de gemaakte code laat zien.

□□ Inleveren

Upload de screenshot met jouw Scratch-code.

Stap 6 – Monster toevoegen

We maken het spannend! Voeg een monster toe dat je moet ontwijken. Laat het monster bewegen.

Bekijk de uitlegvideo

https://www.youtube.com/embed/6sYbPEse-jc?si=ugD4jr0WJ0vxC-8N

Alternatieve link: Bekijk de video via HiDrive

□□ Opdracht

Laat het monster bewegen. Lever een screenshot in van de code die het monster laat bewegen.

□□ Inleveren

Upload de screenshot van de code van het monster.

Stap 7 – Game over

Als het monster Giga raakt, is het spel afgelopen. Je leert nu hoe je het spel opnieuw kunt laten starten.

Bekijk de uitlegvideo

https://www.youtube.com/embed/LR5vDCgcLMM?si=Kj7SCUr9BADcQYGp

Alternatieve link: Bekijk de video via HiDrive

□□ Opdracht

Laat zien dat Giga teruggezet wordt naar het begin wanneer hij de muur of het monster raakt.

□□ Inleveren

Lever een screenshot in van de code die dit laat zien.

Stap 8 – Alles terugzetten

Als het spel opnieuw begint, moeten zowel het monster als Giga teruggezet worden naar hun beginpositie. Pas ook de snelheid aan als het spel te moeilijk is.

Bekijk de uitlegvideo

https://www.youtube.com/embed/N76K2ZfCP3o?si=Yf6DBrKI71vHcx2V

Alternatieve link: Bekijk de video via HiDrive

□□ Opdracht

Pas de snelheid van het monster aan en laat zien hoe je alles terugzet.

□□ Inleveren

Lever een screenshot in van de code waarin het monster wordt teruggezet en je de snelheid hebt aangepast.

Stap 9 – Punten bijhouden

We voegen nu een score toe met behulp van een variabele.

> Een variabele is een waarde die je kunt opslaan en aanpassen tijdens het spel.

Bekijk de uitlegvideo

https://www.youtube.com/embed/1GTrDEd1ECY?si=_ALBcbZ-dacTo0dq

Alternatieve link: Bekijk de video via HiDrive

Opdracht

Laat zien hoe je de score bijhoudt.

Inleveren

Lever een screenshot in van je code waarin je de score bijhoudt met een variabele.

Stap 10 – Afronden

We maken het spel af en zorgen dat het er goed uitziet. Je kunt je spel opslaan op je laptop of online delen.

Bekijk de uitlegvideo

https://www.youtube.com/embed/cHWgQNulLjU?si=5iQpiZdUwrxSB8UU

Alternatieve link: Bekijk de video via HiDrive

Opdracht

Maak het spel af, voeg iets unieks toe en sla het op.

Inleveren

Lever jouw eigen unieke versie van het spel in.

Stap 11 – Wat heb je geleerd?

Je hebt veel geleerd over programmeren. Nu kijk je terug op wat je allemaal hebt gedaan.

Opdracht

- **Vraag 1:** Waarvoor heb je in dit spel een lus gebruikt?
- **Vraag 2:** Waarvoor heb je een als-dan-anders-blok gebruikt?
- **Vraag 3:** Waarvoor heb je een variabele gebruikt?

□□ Inleveren

Schrijf je drie antwoorden op en lever ze in.

Scratch 2

*Deze pagina biedt een reeks lessen voor het **maken van een platformspel** in Scratch 2, vergelijkbaar met Super Mario. De instructies leiden gebruikers stap voor stap door het proces, beginnend bij **basisbeweging** en voortschrijdend naar complexere elementen zoals **springen op muren**, het vermijden van obstakels, en het **toevoegen en verslaan van monsters**. Elke sectie bevat een **video-tutorial**, opdrachten, en vereist screenshots voor **inlevering en punten**.*

Introductie en start

Wil je precies weten wat je in de module gaat doen, dan kan je hier meer informatie vinden.

Spelregels

Weet je nog wat de spelregels waren, deze helden nog steeds?

Aan de slag....

We slaan een paar stappen over en gaan direct aan de slag met het maken van een soort 'Mario-spel'.

We beginnen met deze 'code': Giga Platformer Start.sb3

We gaan code gebruiken die we in *Scratch 1* hebben gemaakt. Hiervoor gebruiken we de 'rugzak'. Dit wordt in de video uitgelegd.

https://www.youtube.com/embed/zRWO9vWytPA

Alternatieve link: https://my.hidrive.com/lnk/ApKeEML7P

□□ Opdracht

Bekijk de video en laat Giga vloeiend bewegen met een *herhaal-als*-blok; in het als-blok herhaal je de beweging zolang de toets is ingedrukt.

□□ Inleveren

Een screenshot van je 'code'.

Springen

We gaan Giga laten springen. In de video wordt uitgelegd hoe.

https://www.youtube.com/embed/iw7YvogC5Uo

Alternatieve link: https://my.hidrive.com/lnk/dHHmCfJHa

Aan het eind van de video krijg je de uitdaging om de springbeweging vloeiender te maken.

□□ Opdracht

Maak de springbeweging en probeer deze vloeiender te maken.

□□ Inleveren

Een screenshot van je 'code'.

Muur

We gaan een muur in het veld plaatsen en het spel zo maken dat we op de muur kunnen springen.

Bekijk de video voor instructies.

Aan het eind blijven we in de lucht hangen. Dat is niet de bedoeling. Kun jij dat oplossen?

https://www.youtube.com/embed/vOAI_YY6Ntk

Alternatieve link: https://my.hidrive.com/lnk/KwzJbR6t3

□□ Opdracht

Zorg ervoor dat Giga niet meer in de lucht blijft zweven als je springt.

□□ Inleveren

Een screenshot van je 'code'.

Niet meer door de muur

We lopen dwars door de muur heen en dat is niet de bedoeling. We gaan dat in deze stap oplossen.

Staan we op de muur, dan kunnen we opeens niet meer bewegen. Dat kun jij vast oplossen door naar de Y-positie te kijken.

Bekijk de video voor instructies.

https://www.youtube.com/embed/L8q4OR8n_yw

Alternatieve link: https://my.hidrive.com/lnk/cLNtywaNd

□□ Opdracht

Kun jij ervoor zorgen dat Giga, als hij op de muur staat, nog steeds kan bewegen?

□□ Inleveren

Een screenshot van je 'code'.

Van links en rechts

Staan we op de muur, dan kun je nu wel bewegen omdat we met behulp van de Y-positie controleren of we op de muur staan.

Maar als we vanaf de rechterkant teruglopen, kunnen we nog door de muur heen. Kun jij dat oplossen?

https://www.youtube.com/embed/mluq9_b4xSI

Alternatieve link: https://my.hidrive.com/lnk/JYozfTzCk

□□ Opdracht

Kun jij ervoor zorgen dat Giga niet meer door de muur heen kan lopen als hij terugloopt?

□□ Inleveren

Een screenshot van je 'code'.

Lopen door het level

We moeten het platform laten bewegen om door het level heen te lopen. We gaan de muur laten bewegen zodat het lijkt alsof we door het level heen lopen. Dat is best een beetje ingewikkeld, want de muur moet pas bewegen als we ongeveer in het midden staan.

In de video wordt dit uitgelegd en voorgedaan.

https://www.youtube.com/embed/Cg9T9HoI7tY

Alternatieve link: https://my.hidrive.com/lnk/MvsuRrGFy

□□ Opdracht

Kun jij de muur, als die uit beeld is, opnieuw laten verschijnen?

□□ Inleveren

Een screenshot van je 'code'.

Tijd voor monsters

We gaan een 'monster' (ook wel 'vijand' genoemd) maken. Deze beweegt net als de muur, zodat het lijkt alsof je door het level loopt.

We gaan het 'monster' laten bewegen.

https://www.youtube.com/embed/jlPn-XL4T_c

Alternatieve link: https://my.hidrive.com/lnk/egNP9MPj2

□□ Opdracht

Kun jij het monster heen en weer laten bewegen?

□□ Inleveren

Een screenshot van je 'code'.

Monster bewegen

Het monster beweegt, maar blijft nog aan de muur plakken.

https://www.youtube.com/embed/CQby8IzbkZg

Alternatieve link: https://my.hidrive.com/lnk/Z0y7vVvHj

□□ Opdracht

Laat het monster wachten tot de muur op positie 100 staat en laat het monster dan pas verschijnen.

□□ Inleveren

Een screenshot van je 'code'.

Game Over!

Als het monster je raakt, ben je 'af'. Als je af bent, stopt het spel.

https://www.youtube.com/embed/dM9a4XprB7o

Alternatieve link: https://my.hidrive.com/lnk/IrCnwcFwx

□□ Opdracht

Zet 'Game Over' in beeld als je af bent, zodat je weet dat je 'af' bent.

□□ Inleveren

Een screenshot van je 'code'.

Aanvallen

We gaan het spel zo aanpassen dat we het monster kunnen 'verslaan'.

https://www.youtube.com/embed/CHWy9lLV4bA

Alternatieve link: https://my.hidrive.com/lnk/dvVi8f5dQ

□□ Opdracht

Als Giga tegen het monster aanloopt, is het 'Game Over', maar als je op het monster springt, moet het monster worden geplet. Schrijf hiervoor de code (tip: dit lijkt op de code van de muur waarop we kunnen springen).

□□ Inleveren

Een screenshot van je 'code'.

Monster verdwijnt

Als je het monster 'plet', wil je dat het geen 'Game Over' is. Het monster moet worden geplet en daarna verdwijnen.

We maken een aanpassing zodat het monster weer terugkomt nadat het is geplet.

https://www.youtube.com/embed/dV4ivXfP2q0

Alternatieve link: https://my.hidrive.com/lnk/Eolifu5Sn

□□ Opdracht

Oeps, we hebben nog een bug. Het monster verschijnt weer terug, maar is nog steeds geplet. Kun jij dit oplossen?

□□ Inleveren

Een screenshot van je 'code'.

Stop als je geplet bent

We moeten ervoor zorgen dat zodra het monster is geplet, het niet meer beweegt.

https://www.youtube.com/embed/a9ubC4LTcrs?t=3s

Alternatieve link: https://my.hidrive.com/lnk/DSscj5iKz

□□ Opdracht

Kun jij het monster stilzetten zodra het is geplet?

□□ Inleveren

Een screenshot van je 'code'.

Klaar

We zijn klaar!

https://www.youtube.com/embed/0xFosaKxcgY

Alternatieve link: https://my.hidrive.com/lnk/XZVISW0jy

□□ Opdracht

Je hebt 12 screenshots ingeleverd. Als deze goed zijn, heb je 12 x 5 = 60 punten.

Voor een vinkje moet je 84 punten of meer hebben.

Laat het hele spel aan een docent zien. Je kunt tot 35 punten krijgen.

De docent kan vragen stellen over hoe jouw spel werkt.

Het spel moet aan de volgende voorwaarden voldoen:

Verplicht (+25 punten)

- Je kunt door het spel heen bewegen: naar links en naar rechts.
- Je kunt niet door de muur heen lopen, maar je kunt er wel op springen.
- Je hebt een monster; als je die raakt, is het 'Game Over'. Er verschijnt een 'Game Over'-melding en het spel stopt.
- Je kunt het monster pletten: je ziet het geplette monster, waarna het verdwijnt en even later weer terugkomt.

□□ Inleveren

In het tekstveld tik je in dat je klaar bent. De docent zal samen met een aantla sutdenten het werk doornemen en goedkeuren.

--

###

Youtube Links

1. https://www.youtube.com/watch?v=zRWO9vWytPA
2. https://www.youtube.com/w atch?v=iw7YvogC5Uo
3. https://www.youtube.com/watch?v=vOAI_YY6Ntk
4. https://www.youtube.com/watch?v=L8q4OR8n_yw
5. https://www.youtube.com/watch?v=mluq9_b4xSI
6. https://www.youtube.com/watch?v=Cg9T9HoI7tY
7. https://youtu.be/jlPn-XL4T_c
8. https://www.youtube.com/watch?v=CQby8IzbkZg
9. https://www.youtube.com/watch?v=dM9a4XprB7o
10. https://www.youtube.com/watch?v=CHWy9lLV4bA
11. https://www.youtube.com/watch?v=dV4ivXfP2q0

12. https://www.youtube.com/watch?v=a9ubC4LTcrs&t=3s
13. https://www.youtube.com/watch?v=0xFosaKxcgY

--

Van Scratch naar Python

Status: alles uitgevoerd en getest

*Deze webpagina, getiteld "Van Scratch naar Python", dient als een tutorial om beginners te helpen overstappen van de visuele programmeertaal Scratch naar de tekstgebaseerde taal Python. De lessen behandelen belangrijke programmeerconcepten zoals **indentatie** (hoe codeblokken worden herkend in Python), het gebruik van **commentaar** om code te verduidelijken, en de implementatie van **if-statements** voor beslissingslogica en **loops** (zoals `for`- en `while`-loops) voor herhalende acties. Door middel van praktische opdrachten met een bewegende stip (sprite) leren gebruikers deze concepten toe te passen en steeds complexere bewegingspatronen, zoals stuiteren, vierkante bewegingen en spiralen, te creëren in de Thonny Python-omgeving, waarbij ook het gebruik van de `pygame` library en een specifieke `scratch_lib.py` wordt uitgelegd.*

0 Wat gaan we leren?

We gaan code maken.

We gaan daarvoor Python gebruiken.

Wat gaan we leren:

- waarom en hoe we inspringen in Python.
- wat commentaar in code is
- hoe we in Python een if-statement maken
- hoe we in python een loop (lus) maken

In deze lessen worden de volgende Scratch blokken in geschreven code omgezet.

IF - THEN	IF - THEN - ELSE	FOR LOOP (repeat)
Image.png not found or type unknown	Image.png not found or type unknown	Image.png not found or type unknown

Opdracht

Leg in eigen woorden uit:

1. wat is het verschil tussen een `if-then` en een `if-then-else` ?
2. Waarvoor gebruik je een for-loop (of een repeat; dat is hetzelfde)?

Inleveren

Een antwoord op de twee vragen, in eigen woorden (geen AI)!

Maak een txt bestand en schrijf daarin je antwoorden.

1 Installatie Python (Thonny)

We hebben geprogrammeerd in Scratch en we gaan nu programmeren in een echte programmeertaal: Python.

We gaan echte code maken, maar daarvoor moeten we eerst wat zaken installeren.

We gaan gebruikmaken van Thonny

Download

Pak het bestand uit en zet het op een plek die voor jou logisch is, bijvoorbeeld op je bureaublad.

Installatie *pygame* Library

Programmeertalen hebben libraries (ook wel 'packages' genoemd). Deze libraries bevatten code die jij kunt gebruiken.

Wij gaan de *pygame* library installeren omdat we die straks nodig hebben.

Als je Thonny opstart, ga dan naar *Tools - Manage packages...*

image.png
Image not found or type unknown

Zoek dan naar *pygame*

image.png
Image not found or type unknown

Klik op *pygame* en daarna op de knop *Install*

Download code

Download de code startcode-python-scratch.zip

Pak de code uit, start Thonny en open het bestand `student.py`

image.png
Image not found or type unknown

Druk op het groene 'run'-symbool en kijk wat er gebeurt.

Het programma wordt regel voor regel van boven naar beneden uitgevoerd.

Uitleg code

Hieronder zie je de uitleg. Het kan zijn dat je niet alles in één keer begrijpt, maar probeer in ieder geval de rode uitleg te begrijpen.

Regel 1

Hier worden libraries ingeladen. Dit zijn stukjes code die al klaar zijn en die in het bestand `scratch_lib.py` staan.

Regel 3

Hier wordt de sprite gemaakt en op een positie gezet. Let op dat positie (0, 0) linksboven is (en niet in het midden zoals bij Scratch).

Regel 5

Hier maken we een functie waarmee de sprite wordt bewogen. Dit is nodig om de library te kunnen gebruiken.

Regel 6

Hiermee bewegen we de sprite 10 pixels naar rechts en 0 pixels naar beneden.

Regel 7

We pauzeren een aantal frames.

Regel 9

Hiermee starten we het spel.

□□ Opdracht

Probeer de getallen op regel 6 eens aan te passen en kijk wat er gebeurt.

Verander de getallen zodanig dat de groene stip van linksboven **diagonaal** richting rechtsonder beweegt.

Inleveren

Maak een screenshot van de code die jij hebt aangepast zodat de groene stip diagonaal van linksboven naar rechtsonder beweegt.

2 De stuiterbal

In deze opdracht leer je hoe je een sprite (een groene stip) van links naar rechts kunt laten bewegen **en** hoe je met een `if`-statement de richting verandert zodra de sprite de rechterkant van het scherm bereikt.

Begincode

Je gebruikt de volgende code als uitgangspunt:

```
from scratch_lib import create_sprite, move, run_animation, get_x

# Maak de sprite en zet hem links op het scherm
sprite = create_sprite("green_dot.png", 0, 300)

# Variabele om te onthouden of we naar rechts bewegen
moving_right = True

def animate():
    global moving_right  # We gaan deze variabele aanpassen

    # Haal de huidige x-positie op
    x = get_x(sprite)

    # TODO: Als x groter of gelijk is aan 550, verander moving_right naar False
    # if ???:
```

```
    #    moving_right = False

    # Beweeg de sprite op basis van de richting
    if moving_right:
        move(sprite, 5 , 0 )
    else:
        move(sprite, 0 , 0 )

# Start de animatie
run_animation([sprite], animate, steps=1000)
```

Als je deze code uitvoert, zie je dat de groene stip van links naar rechts beweegt, maar **hij stopt niet of verandert niet van richting**. Hij verdwijnt uit beeld.

ℹ Wat is inspringen in Python?

Inspringen = blok = identation

In Python is de **inspringing** (ook wel *indentatie* genoemd) heel belangrijk. Python gebruikt inspringen om aan te geven welke code bij elkaar hoort.

Als je bijvoorbeeld een `if`-statement gebruikt, dan moet de code die daarbij hoort **een stukje naar rechts inspringen** (meestal 4 spaties).

```
if x >= 100:
    move(sprite, 5, 0)  # deze regel hoort bij het if-blok

# dit staat buiten het if-blok
print("Ik ben klaar!")
```

◻◻ Vergelijking met Scratch

In Scratch zie je blokken zoals "*als ... dan*" of "*herhaal ...*". De blokken die **in** zo'n constructie staan, vallen daar letterlijk *in*. Ze zijn visueel naar binnen geschoven.

In Python doe je dat met spaties:

- De **buitenste structuur** (zoals `if` of `for`) sluit je af met een dubbele punt `:`.
- De regels die **bij dat blok horen**, zet je eronder en laat je 4 spaties naar rechts inspringen.

Als je dit vergeet, krijg je in Python een foutmelding zoals:

```
IndentationError: expected an indented block
```

✅ Juiste voorbeeld

```
if moving_right:
    move(sprite, 5 , 0 )
```

❌ Fout voorbeeld

```
if moving_right:
move(sprite, 5 , 0 )
```

Controleer dus goed dat de regels die bij een `if` of `for` horen, netjes zijn ingesprongen.

Wat gaan we doen?

We willen dat de bal **van richting verandert** als hij de rechterkant bereikt (bijvoorbeeld bij `x = 550`).

Daarvoor heb je een paar dingen nodig:

De variabele `moving_right` die onthoudt of de sprite naar rechts beweegt (`True`) of niet (`False`).

Een `if`-statement (regel 15, 16 en 17) die controleert of de `x`-waarde van de sprite groter is dan 550.

Als dat zo is, moet de sprite naar links bewegen in plaats van naar rechts (regel 19, 20, 21, 22 en 23).

De code is nog niet helemaal af.

Commentaar

In de code zie je af en toe een hekje `#` aan het begin van de regel staan.

Dit betekent dat dit **commentaar** is. De regel wordt **niet** uitgevoerd. Het dient om jou als programmeur te helpen begrijpen wat de code doet.

Opdracht

Haal het hekje weg op regel 16 en 17, maar zorg ervoor dat de uitlijning goed blijft: voor de `if` **vier** spaties en op de regel `moving_right = False` **acht** spaties.

Op de plaats van de `???` plaats je nu de juiste conditie. Je vergelijkt of de x-positie van de sprite groter of gelijk is aan 550.

In Python ziet dat er als volgt uit:

```
if var_a >= 550:
```

`var_a` is een variabele. Plaats deze conditie in de code en vervang `var_a` door de juiste variabele die de x-positie bevat.

→ Test je code. Geen foutmeldingen? OK!

Wat gebeurt er nu als de x-positie 550 is? Precies — de bal staat stil!

Kijk nog eens goed naar het `if`-statement op regel 20 t/m 23 en probeer de code aan te passen zodat de bal niet meer stil staat als hij positie 550 heeft bereikt, maar dat hij terug beweegt.

Gebruik daarna een tweede `if`-statement om te bepalen **hoe** de sprite moet bewegen:

- Als `moving_right` `True` is → beweeg naar rechts.
- Anders, dus als `moving_right` `False` is → beweeg naar links.

Denk eraan: een positief getal beweegt de sprite vooruit, een negatief getal beweegt hem achteruit.

Inleveren

Maak een screenshot van de aangepaste code.

3 De stuiterbal – heen en weer

In deze opdracht breiden we de vorige oefening uit. De groene stip moet nu niet alleen van links naar rechts bewegen, maar ook weer **terug naar links** als hij de rechterrand heeft bereikt, en daarna **weer naar rechts** als hij de linkerrand bereikt.

Begincode

Je gebruikt de volgende code als uitgangspunt. Deze lijkt op de vorige, maar nu gaan we twee richtingen controleren.

```
from scratch_lib import create_sprite, move, run_animation, get_x

# Maak de sprite en zet hem links op het scherm
sprite = create_sprite("green_dot.png", 0, 300)

# Variabele om te onthouden of we naar rechts bewegen
moving_right = True

def animate():
    global moving_right

    x = get_x(sprite)

    # Keer om als de sprite de rechterkant raakt
    if x >= 550:
        moving_right = False

    # TODO: Voeg hier een extra if-statement toe:
    # Als de sprite aan de linkerkant is (x <= 0), dan moet moving_right weer True worden

    if moving_right:
        move(sprite, 5, 0)
    else:
        move(sprite, -5, 0)

run_animation([sprite], animate, steps=1000)
```

Als we een if maken dan kennen we de volgende vergelijking

==	is gelijk aan?
<	is kleiner dan?
>	is groter dan?
<=	is kleiner of gelijk aan?
>=	is groter of gelijk aan?
!=	is ongelijk aan?

if-then-else - Vergelijking met Scratch

If-loop

image.png
Image not found or type unknown

if-then-else loop

image.png
Image not found or type unknown

Wat moet je doen?

Je gaat nu een extra `if`-statement toevoegen die controleert of de bal de **linkerkant** van het scherm heeft bereikt (dus bij `x <= 0`).

Als dat zo is, verander dan de waarde van `moving_right` weer naar `True`. Daardoor beweegt de sprite weer naar rechts.

□□ Opdracht

- Voeg onder de eerste `if`-statement een tweede `if`-statement toe.
- Controleer of `x <= 0`.
- Als dat zo is, zet `moving_right = True`.
- Test je code. Werkt het? Dan beweegt de bal nu heen en weer!

□□ Tip

Als je wilt, kun je bij beide `if`-statements ook een `print()` toevoegen, zodat je in het log kunt zien wanneer de richting verandert.

```
if x >= 550:
    moving_right = False
    print("Rechterkant bereikt – keer om")

if x <= 0:
    moving_right = True
    print("Linkerkant bereikt – keer om")
```

Inleveren

Maak een screenshot van jouw code waarin je beide `if`-statements hebt toegevoegd en de sprite heen en weer beweegt.

4 De vierkante beweging

In deze opdracht leer je hoe je een sprite (de groene stip) kunt laten bewegen in de vorm van een **vierkant**. De sprite moet dus eerst naar rechts, dan naar beneden, dan naar links, en tot slot weer omhoog. Daarna herhaalt hij dit patroon.

Begincode

Je gebruikt de volgende code als uitgangspunt. Deze keer gaan we bijhouden in welke **richting** de sprite moet bewegen, en telkens van richting veranderen als hij een hoekpunt bereikt.

```
from scratch_lib import create_sprite, move, run_animation, get_x, get_y

# Startpositie linksboven
sprite = create_sprite("green_dot.png", 10, 10)

# We gebruiken een getal om de richting bij te houden:
# 0 = rechts, 1 = naar beneden, 2 = naar links, 3 = omhoog
richting = 0

def animate():
    global richting

    x = get_x(sprite)
    y = get_y(sprite)

    Op basis van de richting, kies hoe de sprite moet bewegen
    if richting == 0:        # boven naar rechts bewegen
        move(sprite, 5, 0)
        if x >= 550:
            richting = 1     # volgende richting: aan de rechter kant naar beneden bewegen

    elif richting == 1:      # rechts naar beneden bewegen
        move(sprite, 0, 5)
        if y >= 550:
```

```
        richting = 2     # volgende richting: beneden langs naar links bewegen

    # ToDo maak de code hier af
    # we hebben moeten nog 2 blokjes maken:
    #    beneden langs naar rechts bewegen
    #    linker kant omhoog bewegen.
    # (je kunt het blokje op regel 22-25 kopiëren en aanpassen)



run_animation([sprite], animate, steps=2000)
```

Wat moet je doen?

In de code staat al aangegeven welke stappen moeten worden uitgevoerd. Maar niet alles is compleet.

- Controleer of je begrijpt wat de waarde van `richting` betekent.
- De sprite moet telkens van richting veranderen als hij een hoekpunt van het vierkant heeft bereikt.
- De richtingsveranderingen gebeuren met behulp van een `if` of `elif`-structuur.
- Pas eventueel de getallen 550 aan als jouw sprite kleiner of groter is.

□□ Opdracht

- Vul de `TODO` op regel 27 aan door goed te begrijpen wat elke `if` doet.
- Test je code. Beweegt de sprite in een vierkant? Perfect!

Inleveren

Maak een screenshot van jouw werkende code waarin je laat zien dat de sprite een vierkant loopt.

5 Vierkant met sprongen op elke hoek

In deze opdracht ga je de sprite in een **kleiner vierkant** laten bewegen. Maar dat is nog niet alles: **op elk hoekpunt van het vierkant** springt de sprite vijf keer op en neer. Hiervoor ga je gebruikmaken van een `for`-loop.

Begincode

We hebben de code voor je voorbereid zodat de sprite een kleiner vierkant loopt. Dit vierkant is 100 stappen breed en hoog. Voer deze code uit en kijk wat er gebeurt:

```
from scratch_lib import create_sprite, move, run_animation, get_x, get_y, force_redraw
import time

# Startpositie linksboven
sprite = create_sprite("green_dot.png", 80, 80)

# We gebruiken een getal om de richting bij te houden:
# 0 = rechts, 1 = naar beneden, 2 = naar links, 3 = omhoog
richting = 0

def animate():
    global richting

    x = get_x(sprite)
    y = get_y(sprite)

    if richting == 0:        # naar rechts
        move(sprite, 5, 0)
        if x >= 470:
            # SPRINGEN: plak hier onderstaande code
            richting = 1     # volgende richting: naar beneden

    elif richting == 1:      # naar beneden
        move(sprite, 0, 5)
        if y >= 470:
            # SPRINGEN: plak hier onderstaande code
            richting = 2     # volgende richting: naar links

    elif richting == 2:      # naar links
        move(sprite, -5, 0)
```

```
        if x <= 80:
            # SPRINGEN: plak hier onderstaande code
            richting = 3     # volgende richting: naar boven

    elif richting == 3:      # naar boven
        move(sprite, 0, -5)
        if y <= 80:
            # SPRINGEN: plak hier onderstaande code
            richting = 0     # opnieuw naar rechts

run_animation([sprite], animate, steps=2000)
```

Sprongen op elk hoekpunt

Nu willen we dat de sprite **op elk hoekpunt van het vierkant** vijf keer op en neer springt.

Op en neer betekent dat de sprite eerst iets omhoog en dan weer omlaag beweegt. Dat doen we in een `for`-loop.

□□ Wat is een `for`-loop?

Een `for`-loop gebruik je in Python als je iets **meerdere keren wilt herhalen**. Dat kan bijvoorbeeld zijn: de sprite 5 keer naar rechts bewegen, of 10 keer springen.

□□ De basisvorm van een `for`-loop

```
for i in range(5):
    move(sprite, 5, 0)
```

Wat gebeurt hier?

- `for i in range(5)`: dit betekent dat de code in het blok **5 keer wordt uitgevoerd**.
- De variabele `i` krijgt automatisch de waarden 0, 1, 2, 3 en 4 (vijf keer in totaal).
- Elke keer dat de loop draait, voert Python de ingesprongen regels onder de `for`-regel uit.

□□ Loop - Vergelijking met Scratch

In Scratch gebruik je bijvoorbeeld:

image.png
Image not found or type unknown

> *Herhaal 10 keer → [doe iets]*

Dat is precies hetzelfde idee! De blokken die je in Scratch ín een herhaal-blok sleept, zijn in Python de regels die je moet inspringen (met spaties).

Juiste voorbeeld

```
for i in range(3):
    print("Hallo")
```

Uitvoer:

```
Hallo
Hallo
Hallo
```

i Handig om te weten

Wil je iets 10 keer doen?

```
for i in range(10):
```

Wil je iets maar 1 keer doen? Dan heb je eigenlijk geen loop nodig

Oefening (optioneel)

Wat doet onderstaande code? Probeer het te voorspellen.

```
for i in range(2):
    move(sprite, 0, -20)
    force_redraw()
    time.sleep(0.1)
    move(sprite, 0, 20)
    force_redraw()
    time.sleep(0.1)
```

Antwoord: de sprite springt 2 keer op en neer.

Wat moet je doen?

- Kopieer bovenstaande `for`-loop.

- Plak die op vier plekken in de `animate()`-functie:
  - Vlak voordat `richting = 1` wordt uitgevoerd (na de rechterkant).
  - Vlak voordat `richting = 2` wordt uitgevoerd (na beneden).
  - Vlak voordat `richting = 3` wordt uitgevoerd (na links).
  - Vlak voordat `richting = 0` wordt uitgevoerd (na boven).

□□ Opdracht

- Kopieer en plak de `for`-loop op de juiste plekken in je code (op elk hoekpunt).
- Test je code. De sprite moet netjes in een vierkant bewegen **en** op elke hoek vijf keer op en neer springen.

Inleveren

Maak een screenshot van jouw code waarin de sprite op elk hoekpunt springt.

6 Spring vaker op twee hoeken

In deze korte opdracht breid je je bestaande script uit. De sprite moet nu alleen **rechtsboven** en **linksonder** springen, telkens **vijf keer**. Maar dit keer springt de sprite niet omhoog en omlaag, maar **naar links en naar rechts** (horizontaal).

ℹ Weet je nog wat inspringen is?

Inspringen = blok = identation

> In Python is de **inspringing** (ook wel *indentatie* genoemd) heel belangrijk. Python gebruikt inspringen om aan te geven welke code bij elkaar hoort.

Ook bij een `for`-statement , moet de code die daarbij hoort **een stukje naar rechts inspringen** (meestal 4 spaties).

```
for i in range(5):
    move(sprite, 0, -20)
```

```
    force_redraw()
```

Dus de regels 2 én 3 horen bij het for-blok en worden 5x uitgevoerd.

Vergelijking met Scratch

In Scratch zie je blokken zoals "*als ... dan*" of "*herhaal ...*". De blokken die **in** zo'n constructie staan, vallen daar letterlijk *in*. Ze zijn visueel naar binnen geschoven.

In Python doe je dat met spaties:

- De **buitenste structuur** (zoals `if` of `for`) sluit je af met een dubbele punt `:`.
- De regels die **bij dat blok horen**, zet je eronder en laat je 4 spaties naar rechts inspringen.

Als je dit vergeet, krijg je in Python een foutmelding zoals:

```
IndentationError: expected an indented block
```

Juiste voorbeeld

```
for i in range(5):
    move(sprite, 0, -20)
    force_redraw()
```

Fout voorbeeld

```
for i in range(5):
move(sprite, 0, -20)   # geen inspringing!
force_redraw()
```

Controleer dus goed dat de regels die bij een `if` of `for` horen, netjes zijn ingesprongen.

Wat moet je doen?

- Zoek in je code de momenten waarop de sprite de **rechterbovenhoek** en de **linkeronderhoek** bereikt.
- Op die plekken laat je de sprite 5X (in plaats van 2X) springen.

Zorg dat je deze code **alleen** toevoegt bij de overgang van:

- `richting == 0` → als `y <= 80` (rechtsboven)
- `richting == 2` → als `y >= 470` (linksonder)

Inleveren

Maak een screenshot van de code in Thonny waarop te zien is dat de sprite alleen rechtsboven en linksonder **horizontaal** springt, vijf keer per keer.

7 Spiraal – stap 1

"*vaste richtingen in een patroon*"

In deze opdracht ga je de sprite **steeds twee richtingen bewegen** met een vaste afstand. Je doet dit een paar keer achter elkaar.

Uiteindelijk zal dit het begin worden van een spiraal. Maar eerst leer je het patroon maken: **rechts → omlaag → links → omhoog**.

□□ Begincode

```
from scratch_lib import create_sprite, move, pause_for_frames

sprite = create_sprite("green_dot.png", 300, 300)

# Afstand per richting
afstand = 100

# Herhaal het patroon 3 keer
for i in range(3):
    # Beweeg naar rechts
    for j in range(afstand // 5):
        move(sprite, 5, 0)
        pause_for_frames(1)

    # Beweeg naar beneden
```

```
    for j in range(afstand // 5):
        move(sprite, 0, 5)
        pause_for_frames(1)

    # Beweeg naar links
    for j in range(afstand // 5):
        move(sprite, -5, 0)
        pause_for_frames(1)

    # Beweeg naar boven
    for j in range(afstand // 5):
        move(sprite, 0, -5)
        pause_for_frames(1)
```

ℹ Uitleg

Je ziet hierboven een herhaling (een `for`-loop) die het patroon van 4 richtingen 3 keer uitvoert.

Elke richting bestaat uit een eigen `for`-loop waarin de sprite telkens kleine stapjes zet.

De afstand is verdeeld in blokjes van 5 pixels, zodat je het goed kunt zien bewegen.

□□ Opdracht

- Experimenteer met `afstand = 100`. Wat gebeurt er als je die verandert in 200?
- Voeg commentaar toe bij elke richting, zodat je het patroon beter begrijpt.
- Test wat er gebeurt als je `range(3)` verandert in `range(1)` of `range(5)`.

□□ Inleveren

Maak een screenshot van je code én van het pad dat de sprite aflegt in het venster.

8 Spiraal – stap 2

"*kleiner wordende afstanden*"

In de vorige opdracht heb je een patroon gemaakt: de sprite bewoog rechts, omlaag, links, omhoog – en dat een paar keer. Het pad bleef telkens even groot.

Nu gaan we iets nieuws doen: **na elke twee richtingen wordt de afstand kleiner**. Hierdoor lijkt het alsof de sprite langzaam een spiraal naar binnen loopt.

□□ Begincode

```
from scratch_lib import create_sprite, move, pause_for_frames

sprite = create_sprite("green_dot.png", 300, 300)

afstand = 200  # beginafstand

for i in range(5):  # we doen 5 spiraal-lagen
    # naar rechts
    for j in range(afstand // 5):
        move(sprite, 5, 0)
        pause_for_frames(1)

    # naar beneden
    for j in range(afstand // 5):
        move(sprite, 0, 5)
        pause_for_frames(1)

    afstand = afstand - 40  # we maken de afstand kleiner

    # naar links
    for j in range(afstand // 5):
        move(sprite, -5, 0)
        pause_for_frames(1)

    # naar boven
    for j in range(afstand // 5):
        move(sprite, 0, -5)
        pause_for_frames(1)

    afstand = afstand - 40  # opnieuw iets kleiner maken
```

ℹ Uitleg

- We beginnen met `afstand = 200`.
- Na twee richtingen verkleinen we de afstand met 40.
- Daarna bewegen we weer twee richtingen (links en boven).
- We verkleinen opnieuw met 40.

Zo wordt het pad kleiner en kleiner – alsof je een vierkante spiraal naar het midden tekent.

□□ Opdracht

- Pas de waarde `afstand = 200` aan. Wat gebeurt er bij 300?
- Wat als je niet met 40 verkleint, maar met 20?
- Voeg een `print(afstand)` toe onder elke `afstand = afstand - 40`. Wat zie je in de console?

□□ Inleveren

Maak een screenshot van je aangepaste code én van de spiraal die de sprite tekent.

9 Spiraal – stap 3

"*automatisch stoppen*"

Je hebt nu al een mooie spiraal gemaakt waarbij de sprite steeds kleinere vierkantjes loopt. Maar misschien heb je gemerkt: op een gegeven moment is de afstand zo klein dat de sprite nauwelijks nog beweegt of zelfs gekke dingen gaat doen.

Daarom gaan we in deze stap zorgen dat het script **zelf stopt** als de afstand te klein wordt.

□□ Begincode

```
from scratch_lib import create_sprite, move, pause_for_frames
```

```
sprite = create_sprite("green_dot.png", 300, 300)

afstand = 200

# Herhaal zolang de afstand groter is dan 20
while afstand > 20:
    # naar rechts
    for j in range(afstand // 5):
        move(sprite, 5, 0)
        pause_for_frames(1)

    # naar beneden
    for j in range(afstand // 5):
        move(sprite, 0, 5)
        pause_for_frames(1)

    afstand = afstand - 20

    # naar links
    for j in range(afstand // 5):
        move(sprite, -5, 0)
        pause_for_frames(1)

    # naar boven
    for j in range(afstand // 5):
        move(sprite, 0, -5)
        pause_for_frames(1)

    afstand = afstand - 20
```

ℹ Uitleg

- In plaats van `for i in range(5)` gebruiken we nu een **`while`-loop**.
- Die zorgt ervoor dat het patroon doorgaat **zolang de afstand groter is dan 20**.
- Als de afstand kleiner wordt dan of gelijk is aan 20, stopt de loop vanzelf.

□□ Opdracht

- Test de code. Kun je zien waar de sprite stopt?
- Wat gebeurt er als je de `afstand > 20` verandert in `afstand > 40`?
- Voeg een `print(afstand)` toe aan het einde van elke herhaling om te zien wanneer het stopt.
- Kun je de sprite laten starten in het midden van het scherm in plaats van linksboven?

□□ Inleveren

Maak een screenshot van jouw spiraal **die vanzelf stopt** wanneer het midden bereikt is.

10 Reflectie: wat heb je geleerd?

Je hebt nu gewerkt aan meerdere opdrachten waarbij je de sprite steeds meer hebt aangestuurd met code. Je hebt geleerd hoe je met een `for`-loop en `if`-statements herhaling en logica kunt combineren, en je hebt een echte spiraalbeweging gemaakt in Python.

In deze laatste opdracht ga je **terugkijken op je leerproces**. Wat ging goed? Wat vond je moeilijk? En wat wil je nog beter leren?

ℹ Wat is een reflectie?

Reflecteren betekent dat je **nadenkt over wat je gedaan hebt**. Je kijkt niet alleen naar het resultaat, maar vooral naar hoe je het hebt aangepakt en wat je daarvan leert.

□□ Opdracht

Beantwoord de volgende vragen in een kort reflectieverslag (ongeveer 5 zinnen per vraag is voldoende):

1. **Wat ging goed?**
   Geef een voorbeeld van iets dat je zelfstandig hebt opgelost of goed begreep.
2. **Wat vond je lastig?**
   Was er iets dat je niet meteen snapte? Welke opdracht kostte meer tijd dan je dacht?
3. **Noem drie dingen die je hebt geleerd o ver Python.**
   Wat zijn specifieke dingen die je hebt geleerd over Python?

4. **Wat heb je geleerd over if-statements?**
   Beschrijf kort wat je nu beter begrijpt over herhaling of logica in code.
5. **Wat heb je geleerd over loops?**
   Beschrijf kort wat je nu beter begrijpt over herhaling of logica in code.
6. **Wat zou je de volgende keer anders doen?**
   Denk aan hoe je je werk hebt aangepakt, of hoe je omging met fouten.
7. **Waar wil je nog meer mee oefenen?**
   Zijn er onderwerpen waarvan je denkt: dit wil ik nóg beter snappen of meer mee oefenen?
8. **Welke uitleg was duidelijk?**
   Welke opdracht(en) vond je goed uitgelegd?
9. **Welkke uitleg was onduidelijk?**
   Welke opdrachten waren niet duidelijk genoeg uitgelegd en hoe zou jij het ander/beter doen?

□□ Inleveren

Lever je reflectie in als een PDF-bestand.

###

Docenten

opdracht 4

```
from scratch_lib import create_sprite, move, run_animation, get_x, get_y


# Startpositie linksboven
sprite = create_sprite("green_dot.png", 10, 10)


# We gebruiken een getal om de richting bij te houden:
# 0 = rechts, 1 = naar beneden, 2 = naar links, 3 = omhoog
richting = 0
```

```
def animate():
    global richting

    x = get_x(sprite)
    y = get_y(sprite)

    # TODO: Op basis van de richting, kies hoe de sprite moet bewegen
    if richting == 0:        # naar rechts
        move(sprite, 5, 0)
        if x >= 550:
            richting = 1     # volgende richting: naar beneden

    elif richting == 1:      # naar beneden
        move(sprite, 0, 5)
        if y >= 550:
            richting = 2     # volgende richting: naar links

    elif richting == 2:      # naar links
        move(sprite, -5, 0)
        if x <= 0:
            richting = 3     # volgende richting: naar boven

    elif richting == 3:      # naar boven
        move(sprite, 0, -5)
        if y <= 0:
            richting = 0     # opnieuw naar rechts

run_animation([sprite], animate, steps=2000)
```

Pak de Kaas

Status: alles uitgevoerd en getest

*Dit is een lessenserie over een programmeerproject met **Python en Pygame Zero** genaamd "Pak de Kaas". Het leidt leerlingen door zes stappen om een eenvoudig spel te maken waarin een muis kaas verzamelt. De lessen behandelen basisconcepten zoals het weergeven en verplaatsen van **afbeeldingen (sprites)**, het detecteren van **botsingen** tussen objecten, het gebruik van **willekeurige getallen** om de kaas te verplaatsen, het bijhouden van een **score** en het toevoegen van een **tijdslimiet** voor een "Game Over" scenario. Het document bevat ook een **docentenhandleiding** met leerdoelen, potentiële valkuilen en suggesties voor differentiatie en beoordeling.*

0 Wat gaan we leren

We gaan nog een projectje maken met Thonny (uit de vorige les) en bij dit project gaan we gebruik maken van de standaard Python library:

pgzero

Weet je nog hoe je een pacakge installeert in Thonny?

Yep, Tools - Manage Packages en dan pgzero zoeken en installeren.

Als je het niet meer weer weet kijk dan even naar de vorige les.

1 Pak de Kaas – Les 1

Je hebt pgzero in Thonny geinstalleerd? Nee kijk dan bij de vorige stap.

In deze les gaan we met Python en Pygame Zero leren hoe je plaatjes (sprites) op het scherm kunt zetten.

Je leert hoe je het scherm wist, en hoe je sprites op een bepaalde positie tekent.

Wat gaan we doen?

We gaan een muis en een stuk kaas op het scherm laten verschijnen.

image.png
Image not found or type unknown

Hiervoor gebruiken we `screen.blit()` in de `draw()`-functie van Pygame Zero.

□□ Benodigdheden

- Thonny met Pygame Zero geïnstalleerd
- Een map `images/` met daarin twee bestanden: `mouse.png` en `cheese.png`

□□ Startercode

```
# importeer library
import pgzrun

# Spelgrootte
WIDTH = 800
HEIGHT = 600

def draw():
    screen.clear()
    screen.blit("mouse", (150, 150))
    screen.blit("cheese", (50, 50))

#start programma
pgzrun.go()
```

ℹ Uitleg

- De code wordt van **boven** naar **beneden**, regel voor regel uitgevoerd.
- `import pgzrun` hiermee vertellen we dat de library pgrun gaan gebruiken. Sommige computer-comando's 'sdie straks gaan gebruiken staat beschreven in deze library.
- `WIDTH` en `HEIGHT` bepalen hoe groot het venster is
- `draw()` is een speciale functie die automatisch wordt aangeroepen om het scherm te tekenen

- `screen.clear()` is een library functie die het scherm bij elke frame wist
- `screen.blit("mouse", (150, 150))` tekent het plaatje `mouse.png` op positie (150, 150). Dit zijn ook beide library functies.

Opdracht

Pas de coördinaten van de muis en de kaas aan en kijk wat er gebeurt.

- Zet de muis linksboven in beeld
- Zet de kaas rechtonder in beeld

Wat gebeurt er als deze posities gebruikt?
`screen.blit("mouse", (50, 50))`
`screen.blit("cheese", (30, 30))`

> Denk na over de volgorde waarin de commando's worden uitgevoerd.

Inleveren

Maak een screenshot de code met de coordinaten waarbij de muis linksboven in beeld in beeld staat en de kaas rechtsonder.

2 Beweeg de muis

In deze les gaan we de muis laten bewegen met de pijltjestoetsen.

We doen dat door de positie van de muis aan te passen telkens als een toets wordt ingedrukt.

Wat gaan we doen?

We maken twee variabelen `mouse_x` en `mouse_y` om de positie van de muis bij te houden.

In de functie `update()` passen we deze coördinaten aan als je een pijl indrukt.

Startercode

```
# importeer library
import pgzrun

# Spelgrootte
WIDTH = 800
HEIGHT = 600

# Startpositie van de muis
mouse_x = 150
mouse_y = 150

def draw():
    screen.clear()
    screen.blit("mouse", (mouse_x, mouse_y))
    screen.blit("cheese", (50, 50))

def update():
    global mouse_x, mouse_y
    if keyboard.left:
        mouse_x -= 5
    if keyboard.right:
        mouse_x += 5
    if keyboard.up:
        mouse_y -= 5
    if keyboard.down:
        mouse_y += 5

#start programma
pgzrun.go()
```

ℹ Uitleg

- `mouse_x` en `mouse_y` zijn variabelen die bijhouden waar de muis staat
- `update()` wordt meerdere keren per seconde uitgevoerd
- Met `keyboard.left` controleer je of de linkerpijl wordt ingedrukt
- Bij elke toetsdruk wordt de positie een klein stukje aangepast

Opdracht

- Voer de code uit en beweeg de muis met de pijltjestoetsen
- Pas de waarde `5` aan naar een groter of kleiner getal. Wat merk je?
- Probeer ervoor te zorgen dat de muis niet buiten het scherm kan verdwijnen (zie extrat uitleg).
- Als de muis en de kaas op dezelfde positie staan dan verdwijnt de muis onder de kaas. Zorg ervoor dat de muis boven de kaas komt. Zoals hier is weegegven:
image.png
Image not found or type unknown

Extra uitleg

Laat de muis **niet** verder bewegen als hij het scherm uit dreigt te gaan. Voeg hiervoor `if`-statements toe zoals:

```
if keyboard.left and mouse_x > 10:
    mouse_x -= 5
```

Dus hier staat: las de linker pijtjes toets is ingedrukt én de x coördinaat is > 10 zet de postie van de muis dan op de huidige positie - 5. Dus trek 5 van de huisdige posite af.

Doe dit in vier stappen en test **elke** stap.

1. Doe dit eerst voor keyboard.left en test of het werkt.
2. Doe dit ook voor de keyboard.right en test of het werkt.
3. Doe dit daarna ook voor de keyboard.up en test of het werkt.
4. Doe dit tenslotte voor de keyboard.down en test of het werkt.

Inleveren

Maak een screenshot van je aangepaste code waarbij de **muis boven de kaas** beweegt én waarbij de muis **niet uit het scherm** kan bewegen.

3 Botsing met de kaas

In deze les gaan we kijken of de muis de kaas aanraakt.

Daarvoor gebruiken we een `if`-statement en controleren we of de muis en de kaas elkaar overlappen.

Wat gaan we doen?

We maken een rechthoek rondom de muis en rondom de kaas, en gebruiken `colliderect()` om te kijken of ze elkaar raken.

Als de muis de kaas raakt, tonen we een bericht in de console met `print()`.

□□ Startercode

```
# importeer library
import pgzrun

# Spelgrootte
WIDTH = 800
HEIGHT = 600

# Startpositie van de muis en kaas
mouse_x = 150
mouse_y = 150
cheese_x = 50
cheese_y = 50

def draw():
    screen.clear()
    screen.blit("mouse", (mouse_x, mouse_y))
    screen.blit("cheese", (cheese_x, cheese_y))

def update():
    global mouse_x, mouse_y
    if keyboard.left:
        mouse_x -= 5
    if keyboard.right:
        mouse_x += 5
    if keyboard.up:
        mouse_y -= 5
    if keyboard.down:
```

```
        mouse_y += 5

    # Botsing controleren
    mouse_rect = Rect((mouse_x, mouse_y), (50, 50))
    cheese_rect = Rect((cheese_x, cheese_y), (100, 100))

    if mouse_rect.colliderect(cheese_rect):
        print("Gevonden!")
    else:
      print('-')

#start programma
pgzrun.go()
```

ℹ Uitleg

- Een `Rect` is een rechthoek: (x, y, breedte, hoogte)
- `colliderect()` geeft `True` als twee rechthoeken elkaar raken
- In dit voorbeeld zijn muis 50x50 pixels en de kaas beide 100x100 pixels groot.
- Als er een botsing is, toont Python het woord `Gevonden!`

□□ Opdracht

- Probeer de muis met de pijltjestoetsen naar de kaas te bewegen
- Als je `Gevonden!` ziet in de console, werkt de botsing
- De kaas is niet helemaal 100x100 (in alle richtingen). Pas de grootte van de `Rect` aan zodat je alleen een melding "Gevonden!" krijgt als de muis duidelijk de kaas raakt en duidelijk 'op de kaas zit'.
- Zorg er ook voor dat de muis **op** de kaas komt en niet eronder!

□□ Extra uitdaging

Toon een boodschap op het scherm als de muis de kaas heeft gevonden:

```
gevonden = False
```

```
def update():
    ...
    if mouse_rect.colliderect(cheese_rect):
        gevonden = True

def draw():
    ...
    if gevonden:
        screen.draw.text("Gevonden!", (350, 10), fontsize=60, color="red")
```

□□ Inleveren

Maak een screenshot van je console waarin je ziet dat "Gevonden!" verschijnt als de muis de kaas aanraakt.

4 Kaas verspringt

In deze les gaan we ervoor zorgen dat de kaas naar een nieuwe plek springt als de muis hem aanraakt.

We doen dat met de functie `random.randint()` om een willekeurige positie te kiezen.

Wat gaan we doen?

We importeren de `random`-bibliotheek en maken een functie die een nieuwe plek kiest voor de kaas.

Als er een botsing is, roepen we die functie aan en verplaatsen we de kaas.

□□ Startercode

```
import random

WIDTH = 800
HEIGHT = 600

mouse_x = 150
```

```
mouse_y = 150
cheese_x = 50
cheese_y = 50

def nieuwe_kaas_plek():
    x = random.randint(0, WIDTH - 64)
    y = random.randint(0, HEIGHT - 64)
    return x, y

def draw():
    screen.clear()
    screen.blit("mouse", (mouse_x, mouse_y))
    screen.blit("cheese", (cheese_x, cheese_y))

def update():
    global mouse_x, mouse_y, cheese_x, cheese_y

    if keyboard.left:
        mouse_x -= 5
    if keyboard.right:
        mouse_x += 5
    if keyboard.up:
        mouse_y -= 5
    if keyboard.down:
        mouse_y += 5

    mouse_rect = Rect((mouse_x, mouse_y), (64, 64))
    cheese_rect = Rect((cheese_x, cheese_y), (64, 64))

    if mouse_rect.colliderect(cheese_rect):
        cheese_x, cheese_y = nieuwe_kaas_plek()
```

ℹ Uitleg

- `import random` zorgt ervoor dat we willekeurige getallen kunnen gebruiken
- `random.randint(a, b)` geeft een willekeurig getal tussen `a` en `b`
- `WIDTH - 64` zorgt ervoor dat het plaatje niet buiten beeld komt

- Als de muis de kaas raakt, wordt `cheese_x` en `cheese_y` aangepast

□□ Opdracht

- Laat de muis de kaas aanraken en kijk of deze naar een nieuwe plek verspringt
- Probeer het een paar keer en kijk of het altijd binnen het scherm blijft
- Pas de afmetingen van de sprite aan als jouw plaatjes groter of kleiner zijn dan 64x64

□□ Extra uitdaging

Kies een leuk geluidseffect op : https://www.wavsource.com/sfx/sfx.htm

□□ Zet dan ook een bestand, bijvoorbeel `bloop_x.wav` in de (nieuwe) map `sounds/`

Voeg een geluid toe dat afspeelt als de muis de kaas raakt:

```
if mouse_rect.collliderect(cheese_rect):
    sounds.blook_x.play()
    cheese_x, cheese_y = nieuwe_kaas_plek()
```

□□ Inleveren

Maak een screenshot van je werkende code waar de kaas verspringt.

Leg kort uit wat `random.randint()` doet en waarom je `WIDTH - 64` gebruikt.

5 Score bijhouden

In deze les gaan we een score bijhouden: elke keer als de muis de kaas raakt, telt de score één punt op.

Deze score tonen we ook op het scherm.

Wat gaan we doen?

We maken een variabele `score` die begint op 0 en steeds verhoogd wordt bij een botsing.

In de `draw()`-functie tekenen we de score linksboven in het scherm.

Startercode

```
# importeer library
import pgzrun
import random

WIDTH = 800
HEIGHT = 600

mouse_x = 150
mouse_y = 150
cheese_x = 50
cheese_y = 50
score = 0

def nieuwe_kaas_plek():
    x = random.randint(0, WIDTH - 64)
    y = random.randint(0, HEIGHT - 64)
    return x, y

def draw():
    screen.clear()
    screen.blit("mouse", (mouse_x, mouse_y))
    screen.blit("cheese", (cheese_x, cheese_y))
    screen.draw.text(f"Score: {score}", (10, 10), fontsize=40, color="white")

def update():
    global mouse_x, mouse_y, cheese_x, cheese_y, score

    if keyboard.left:
        mouse_x -= 5
    if keyboard.right:
        mouse_x += 5
    if keyboard.up:
        mouse_y -= 5
    if keyboard.down:
        mouse_y += 5
```

```
    mouse_rect = Rect((mouse_x, mouse_y), (64, 64))
    cheese_rect = Rect((cheese_x, cheese_y), (64, 64))

    if mouse_rect.colliderect(cheese_rect):
        cheese_x, cheese_y = nieuwe_kaas_plek()
        score += 1

#start programma
pgzrun.go()
```

ℹ Uitleg

- `score = 0` zet de score aan het begin op nul
- Elke keer als de muis de kaas aanraakt, wordt de score verhoogd met `score += 1`
- `screen.draw.text()` toont tekst op het scherm

□□ Opdracht

- Speel het spel een paar keer en kijk of de score steeds verder oploopt
- Pas de tekstkleur of positie aan van de score
- Maak de tekst groter of kleiner door `fontsize` aan te passen

□□ Extra uitdaging

Laat de kleur van de tekst veranderen bij een bepaalde score:

```
kleur = "black"
if score >= 5:
    kleur = "red"
screen.draw.text(f"Score: {score}", (10, 10), fontsize=40, color=kleur)
```

□□ Inleveren

Maak een screenshot van het spel waarbij de score zichtbaar is (minimaal 3 punten).

Leg in een zinnetje uit wat `score += 1` betekent.

6 Tijdslimiet

In deze les gaan we een tijdslimiet toevoegen. De speler heeft bijvoorbeeld 30 seconden om zoveel mogelijk kaas te pakken.

Aan het einde tonen we “Game Over” en stoppen we het spel.

Wat gaan we doen?

We maken een teller `tijd_over` die elke seconde met 1 omlaag gaat. Als de tijd op is, stopt het spel.

We tonen de tijd linksboven in beeld naast de score.

□□ Startercode

```
# importeer library
import pgzrun
import random

WIDTH = 800
HEIGHT = 600

mouse_x = 150
mouse_y = 150
cheese_x = 50
cheese_y = 50
score = 0
tijd_over = 30
game_over = False

def nieuwe_kaas_plek():
    x = random.randint(0, WIDTH - 64)
    y = random.randint(0, HEIGHT - 64)
    return x, y

def draw():
```

```
    screen.clear()
    screen.blit("mouse", (mouse_x, mouse_y))
    screen.blit("cheese", (cheese_x, cheese_y))
    screen.draw.text(f"Score: {score}", (10, 10), fontsize=40, color="white")
    screen.draw.text(f"Tijd: {tijd_over}", (10, 50), fontsize=40, color="blue")
    if game_over:
        screen.draw.text("Game Over", center=(WIDTH//2, HEIGHT//2), fontsize=60, color="red")

def update():
    global mouse_x, mouse_y, cheese_x, cheese_y, score

    if game_over:
        return

    if keyboard.left:
        mouse_x -= 5
    if keyboard.right:
        mouse_x += 5
    if keyboard.up:
        mouse_y -= 5
    if keyboard.down:
        mouse_y += 5

    mouse_rect = Rect((mouse_x, mouse_y), (64, 64))
    cheese_rect = Rect((cheese_x, cheese_y), (64, 64))

    if mouse_rect.colliderect(cheese_rect):
        cheese_x, cheese_y = nieuwe_kaas_plek()
        score += 1

def verlaag_tijd():
    global tijd_over, game_over
    if tijd_over > 0:
        tijd_over -= 1
    if tijd_over == 0:
        game_over = True

clock.schedule_interval(verlaag_tijd, 1.0)
```

```
#start programma
pgzrun.go()
```

ℹ Uitleg

- `tijd_over` begint op 30 (seconden)
- `clock.schedule_interval(verlaag_tijd, 1.0)` zorgt ervoor dat elke seconde de functie `verlaag_tijd()` wordt aangeroepen
- Als `tijd_over` op 0 staat, verandert `game_over` in `True` en stopt het spel
- In de `draw()`-functie tonen we de resterende tijd en, als het spel voorbij is, de tekst “Game Over”

□□ Opdracht deel 1

- Laat het spel lopen en probeer zoveel mogelijk punten te halen binnen de tijd
- Pas de tijd aan naar 10 of 60 seconden – wat vind je leuker?
- Laat bij “Game Over” ook de eindscore groter in beeld zien.

□□Opdracht deel 2 (opnieuw uitvoeren)

- Zorg ervoor dat de **muis niet uit het beeld** kan worden bewogen (zoals we bij opgave 2 hebben gedaan).
- Zorg ervoor dat de muis **op de kaas** en niet onder de kaas verdwijnt.

□□ Extra uitdaging

Voeg een herstart-mogelijkheid toe met de `R`-toets:

```
def on_key_down(key):
    global score, tijd_over, game_over, mouse_x, mouse_y, cheese_x, cheese_y
    if key == keys.R:
        score = 0
        tijd_over = 30
        game_over = False
        mouse_x, mouse_y = 150, 150
```

```
    cheese_x, cheese_y = nieuwe_kaas_plek()
```

□□ Inleveren

1. Maak een screenshot van het spel als de tijd op is en je “Game Over” ziet en zet een mooie score neer!
2. Bewaar je code als een bestand (in Thonny, file - save as...) en lever het .py bestand met de code in.

7 Eindopdracht

□□ Eindopdracht

Kies één (of meerdere) van de volgende uitbreidingen en voeg die toe aan je spel:

- □□ Zet meerdere stukjes kaas tegelijk op het scherm (gebruik ee `for`-loop)
- □□ Voeg een “giftige” kaas toe: als je die pakt, verlies je punten
- □□ Voeg geluiden toe voor eten, botsing of game over
- □□ Verander het uiterlijk van de muis of de achtergrond
- □□ Laat het spel moeilijker worden naarmate de tijd verstrijkt (bijv. snellere muis of bewegende kaas)

□□ Bedenk zelf ook een uitbreiding? Schrijf het plan op, laat het goedkeuren door je docent en priobeer het te maken!

> Bij deze opgave mag je AI gebruiken!

□□ Inleveren

Lever één van de volgende dingen in:

- Een werkend Python-bestand (.py) waarin jouw uitbreiding is verwerkt
- Een screenshot van je spel én;

- Uitleg in tekst (.txt) wat je hebt gedaan.

8 Reflectie

In deze les kijk je terug op wat je hebt gemaakt, én krijg je de kans om je spel op een leuke manier uit te breiden of aan te passen.

□□ Reflectie

Als je terugkijkt naar deze opgave.

Vraag je zelf de volgende dingen af:

- Wat vond je het leukste om te doen?
- Wat vond je moeilijk of lastig om te begrijpen?
- Welke onderdelen van Python begrijp je nu beter?
- Waar ben je trots op.

□□ Inleveren

1. Lever je reflectie in met de antwoorden op de (reflectie) vragen in een PDF bestand.

> Let op: gebruik je eigen woorden en wees specifiek!

###

 # Docenten

□□ *Docentenhandleiding*

□□ Overzicht

- **Doelgroep:** Leerlingen van 12-15 jaar (instapniveau Python)
- **Duur:** 6 lessen van ±45-60 minuten
- **Software:** Thonny + Pygame Zero
- **Spelconcept:** Een muis beweegt met de pijltjestoetsen en pakt steeds opnieuw verschijnende kaasjes

□□ Leerdoelen

- Begrijpen hoe coördinaten werken in een 2D-scherm
- Gebruik van `draw()` en `update()` in een animatie
- Werken met variabelen voor positie en beweging
- Detecteren van botsingen met `Rect` en `colliderect()`
- Gebruik van `random` en herhaalde logica
- Score bijhouden en tonen
- Reflecteren op eigen code en uitbreidingen bedenken

□□ Lesoverzicht

Les	Onderwerp	Nieuwe concepten
1	Muis en kaas tekenen	`draw()`, `screen.blit()`, coördinaten
2	Muis beweegt met toetsen	`keyboard.left`, `update()`, grenzen
3	Botsing detecteren	`Rect()`, `colliderect()`
4	Kaas verspringt op random plek	`random.randint()`, logica
5	Score bijhouden	`score += 1`, `screen.draw.text()`
6	Reflectie & uitbreiden	Reflecteren, creatief uitbreiden

⚠ Valkuilen

- `colliderect()` werkt niet → verkeerde grootte/positie van Rect

- Sprites bewegen niet vloeiend → `update()` mist `force_redraw()` (indien nodig)
- Kaas verschijnt buiten het scherm → randomwaarden buiten bereik
- Score telt verkeerd → `score += 1` buiten juiste `if`-blok

□□ Differentiatie

Voor snelle leerlingen

- Voeg meerdere stukjes kaas toe tegelijk
- Laat een “slechte” kaas verschijnen die de score verlaagt
- Gebruik afbeeldingen en laat muis draaien in richting

Voor langzamere leerlingen

- Werk eerst zonder `Rect`, laat botsing handmatig triggeren
- Gebruik alleen horizontale beweging
- Geef startscripts per les met al werkende onderdelen

□□ Beoordeling (optioneel)

Criterium	Omschrijving	Score (1-5)
Spel werkt	Muis beweegt, kaas wordt gepakt, score telt	□□
Codekwaliteit	Variabelen zijn logisch, overzichtelijk	□□
Creativiteit	Leerling heeft iets extra’s toegevoegd (geluid, extra levels, design)	□□
Reflectie	Leerling beantwoordt de reflectievragen met inzicht	□□

□□ Lesaanpak & tips

- Laat leerlingen direct runnen en testen na elke wijziging
- Gebruik klassikale codebesprekingen met testvoorbeelden

- Moedig leerlingen aan om “stukjes code” zelf te veranderen
- Laat ze uitleggen wat ze doen: peer programming werkt goed bij deze opdracht

□□ Benodigdheden

- Thonny + Pygame Zero geïnstalleerd
- Afbeeldingen: muis.png, cheese.png (optioneel)
- Toetsenbord met pijltjestoetsen
- Basiskennis Python (variabelen, `if`, `def`)

Kennis-check Blok 1

Kennis-check Blok 1

Om je voor te bereiden op de multiple choice kennis-check kun je voor je zelf proberen de ondersrtaand evragen te beantwoorden.

Als je op de vraag klikt dan verschijnt het antwoord. Op die manier kun je zelf jou antwoord controleren.
Begrip je d euitleg neit, vraag dan aan de docent om extra uitleg.

Scratch 1

Wat is een lus in programmeren en waarom is het handig?

Een lus (of "loop") is een blokje in Scratch dat zorgt dat een stukje code steeds opnieuw wordt uitgevoerd. Bijvoorbeeld: als je wilt dat een sprite (Giga) blijft bewegen zolang het spel bezig is, kun je een "herhaal" of "herhaal zolang" blok gebruiken. Dit is handig, omdat je dan niet elke stap apart hoeft te programmeren – het gebeurt automatisch steeds opnieuw.

Wat is een als-dan-anders blok (if-then-else)?

Een als-dan-anders-blok in Scratch controleert of iets waar is. Als dat zo is, doet het programma het eerste stukje code. Anders doet het iets anders. Bijvoorbeeld: *als* de muis is ingedrukt, *dan* laat een sprite (Giga) iets zeggen, *anders* doet hij niets of zegt hij iets anders. Zo kan je programma reageren op wat er gebeurt.

Wat is een variabele en hoe kun je die gebruiken om de score bij te houden?

Een variabele is een blokje waarmee je informatie kunt opslaan, zoals een getal dat kan veranderen. In een spel kun je een variabele maken die "score" heet. Aan het begin zet je die op 0. Elke keer dat Giga bijvoorbeeld een sleutel aanraakt, verhoog je de score met 1. Zo weet het spel hoeveel punten je hebt.

Wat is het verschil tussen gewoon “als…dan” en “als…dan…anders”?

“Als…dan” voert alleen code uit *als* de voorwaarde waar is. Als het niet waar is, gebeurt er helemaal niks. **"Als…dan…anders”** voert óf de ene actie uit als de voorwaarde waar is, óf een andere actie als de voorwaarde niet waar is. Dus:

- Met “als…dan” doe je iets óf niks.
- Met “als…dan…anders” doe je altijd iets: óf de ene actie óf de andere.

Scratch 2

Wat betekent het als een sprite op x: 0 en y: 0 staat?

Dat betekent dat de sprite precies in het **midden van het speelveld** staat.
In Scratch is (0, 0) het **middelpunt van het scherm**.

- De x-coördinaat (horizontaal) is dan in het midden van links naar rechts.
- De y-coördinaat (verticaal) is in het midden van boven naar beneden.

In welke richting beweegt een sprite als je de x-coördinaat groter maakt?

Dan beweegt de sprite **naar rechts**.

- Hoe groter de x-waarde, hoe verder naar rechts de sprite op het scherm staat.
- Als je de x-coördinaat juist kleiner maakt, beweegt de sprite naar links.

Wat gebeurt er als je bij x een heel grote waarde gebruikt, zoals 5000?

De sprite verdwijnt **buiten beeld**.

- Het Scratch-scherm heeft grenzen: ongeveer van x = -240 (helemaal links) tot x = 240 (helemaal rechts).
- Als je een waarde gebruikt zoals x = 5000, dan staat de sprite ver buiten het zichtbare scherm, dus je ziet hem niet meer.

Hoe kun je zorgen dat een sprite niet buiten het scherm beweegt?

Gebruik een **“als…dan”**-blok om te controleren of x of y binnen een bepaalde grens.

```
als x-positie < 240 dan
  wijzig x met 10
```

Van Scratch naar Python

Waarom is inspringen (indentatie) belangrijk in Python, terwijl Scratch dat niet nodig heeft?

Omdat Python aan de hand van inspringen (spaties) bepaalt welke code bij elkaar hoort.
Als je dat niet goed doet, begrijpt Python niet wat je bedoelt, en krijg je een foutmelding.

Voorbeeld:

```
if x > 10:
print("x is groot")  # fout: geen inspringing
```

```
if x > 10:
    print("x is groot")  # goed: ingesprongen
```

Leg uit wat een `#` in Python doet

Een `#` wordt gebruikt om commentaar toe te voegen in je code. Alles wat na het `#` staat, wordt genegeerd door de computer. Het is alleen bedoeld voor de programmeur zelf, om uit te leggen wat de code doet.

Voorbeeld:

```
# Dit is een commentaarregel
x = 5  # We geven x de waarde 5
```

Hoe ziet een if-then-else er uit in Python?

In Python gebruik je `if`, `else` (en soms `elif`). De code die bij elke voorwaarde hoort moet ingesprongen staan.

Voorbeeld:

```
if x > 10:
    print("x is groter dan 10")
else:
    print("x is 10 of kleiner")
```

Wat doet een for-loop in Python?

Een `for`-loop herhaalt een blokje code een vast aantal keren.

Voorbeeld:

```
for i in range(5):
    print("Hallo")
```

Noem verschillen tussen Python en Scratch

Scratch	Python
Werkt met blokken die je sleept	Werkt met tekst die je zelf typt
Geen spaties of haakjes nodig	Spaties en dubbele punten zijn belangrijk
Visueel en kleurrijk	Tekstgebaseerd, je moet meer onthouden
Makkelijk te starten zonder fouten	Foutgevoelig bij typefouten of inspringen

Wat leert de student bij 'Pak de Kaas'?

In dit Python/Pygame Zero-project leren studenten onder andere:

- Sprites op het scherm tonen met `screen.blit()`
- De muis laten bewegen via variabelen en `update()`
- Botsingen detecteren tussen muis en kaas
- Willekeurige verplaatsing van de kaas met `random`

- Score bijhouden bij het verzamelen van kaas
- Tijdslimiet instellen zodat het spel eindigt ("Game Over")

Hoe detecteer je dat de muis de kaas raakt?

Gebruik een `if`-statement met `collide` of vergelijk coördinaten van de muis en kaas, bijvoorbeeld:

```
if mouse_x == cheese_x and mouse_y == cheese_y:
    score += 1
```

Waarom gebruik je willekeurige getallen bij het verplaatsen van de kaas na botsing?

Zodat de kaas steeds op een nieuwe plek verschijnt. Bijvoorbeeld met:

```
import random
cheese_x = random.randint(0, WIDTH)
cheese_y = random.randint(0, HEIGHT)
```

Wat doet `random.randint()` precies?

De functie `random.randint(a, b)` geeft een willekeurig geheel getal terug tussen **a** en **b**, inclusief beide grenzen.

Voorbeeld:

```
import random
getal = random.randint(1, 10)
print(getal)
```

Dit print een willekeurig getal tussen 1 e

Pak de Kaas

Hoe detecteer je dat de muis de kaas raakt?

Gebruik een `if`-statement met `collide` of vergelijk coördinaten van de muis en kaas, bijvoorbeeld:

```
if mouse_x == cheese_x and mouse_y == cheese_y:
    score += 1
```

Waarom gebruik je willekeurige getallen bij het verplaatsen van de kaas na botsing?

Zodat de kaas steeds op een nieuwe plek verschijnt. Bijvoorbeeld met:

```
import random
cheese_x = random.randint(0, WIDTH)
cheese_y = random.randint(0, HEIGHT)
```

Wat doet `random.randint()` precies?

De functie `random.randint(a, b)` geeft een willekeurig geheel getal terug tussen **a** en **b**, inclusief beide grenzen.

Voorbeeld:

```
import random
getal = random.randint(1, 10)
print(getal)
```

Dit print een willekeurig getal tussen 1 en 10, zoals 3 of 9. Handig voor bijvoorbeeld het willekeurig verplaatsen van een sprite in een spel.

Wat gebeurt er als je de muis buiten het scherm laat bewegen?

Als je de variabelen `mouse_x` of `mouse_y` zo aanpast dat de sprite buiten het scherm komt, dan **verdwijnt de muis uit beeld**.

- Het spel zelf blijft werken, maar de speler ziet de muis niet meer.
- De sprite staat dan op een coördinaat die buiten het zichtbare venster valt (bijvoorbeeld x = -50 of y = 1000).

Hoe zorg je ervoor dat de muis niet buiten het spel beweegt? Gebruik je een loop of een if-then?

Om te voorkomen dat de muis buiten het scherm beweegt, gebruik je een `if`**-statement**, geen `loop`. Daarmee controleer je of de muis nog binnen de grenzen is voordat je de positie verandert.

Voorbeeld:

```
if mouse_x < WIDTH - 50:
    mouse_x += 5
if mouse_x > 0:
    mouse_x -= 5
```

Opdracht

Maak nu de kennis-check.

Inleveren

Aan het einde van de kennis-check ontvang je een certificaat. Maak een schermafdruk en lever deze in.

--