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?

datasoure

We gaan code maken.

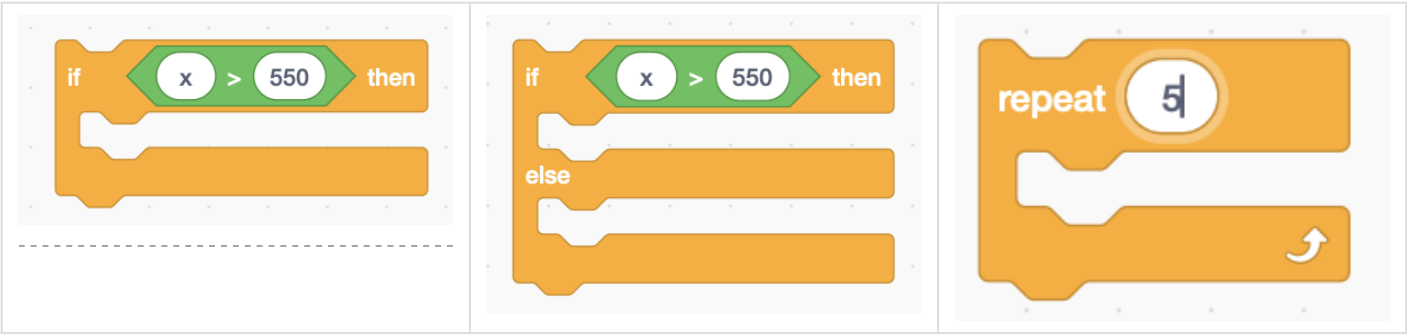
We gaan daarvoor Python gebruiken.

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)



?? 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

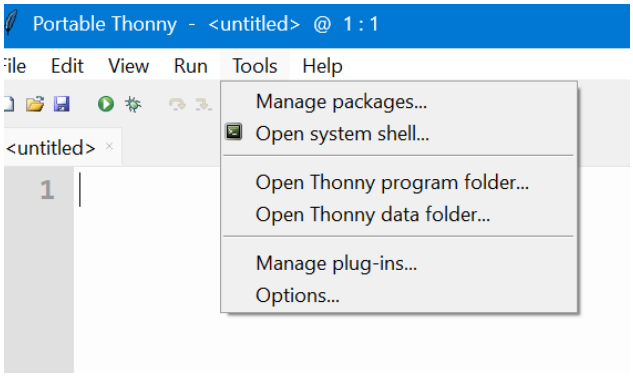
Pak het bestand uit en zet het op een plek die voor jou logisch is, bijvoorbeeld op je bureaublad.

Installatie *pygame* Library

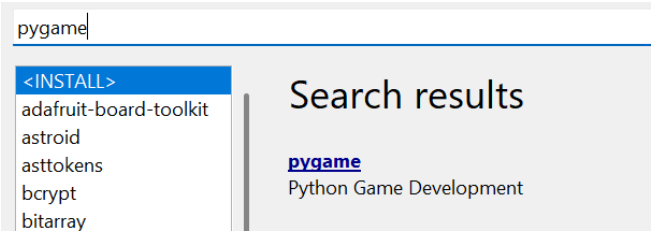
Programmeertalen hebben libraries (ook wel 'packages' genoemd). Deze libraries bevatten code die jij kunt gebruiken.

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

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



Zoek dan naar *pygame*



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`

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

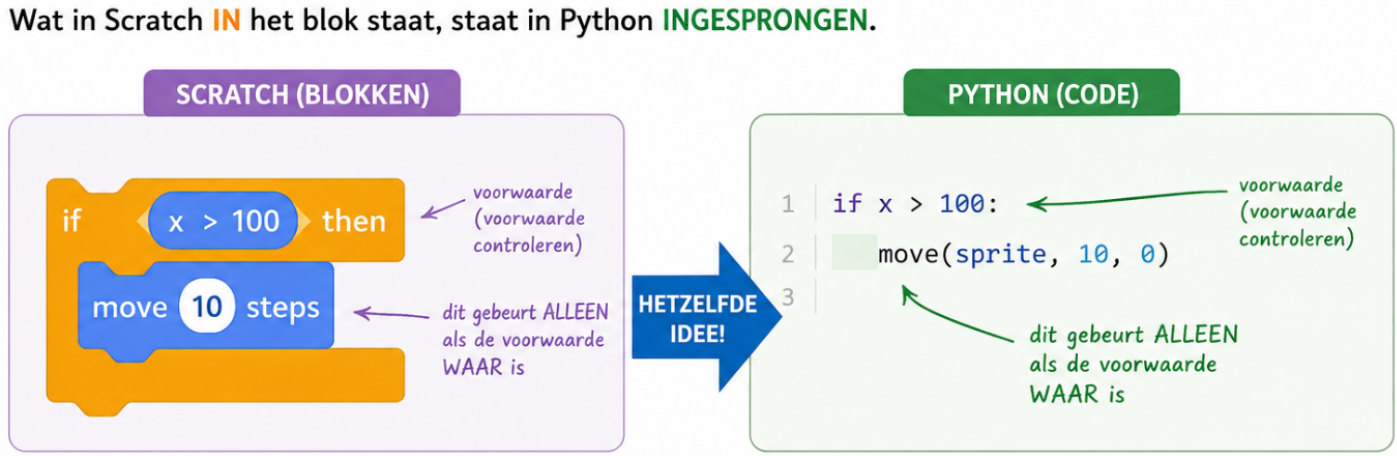
Je gebruikt de volgende code als uitgangspunt:

```
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.

?? 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!")
```

> **TIP**
> In Python gebruik je een dubbele punt (**:**) na de if-regel.
> De regels eronder moeten ingesprongen zijn (meestal met 4 spaties).

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 )
```

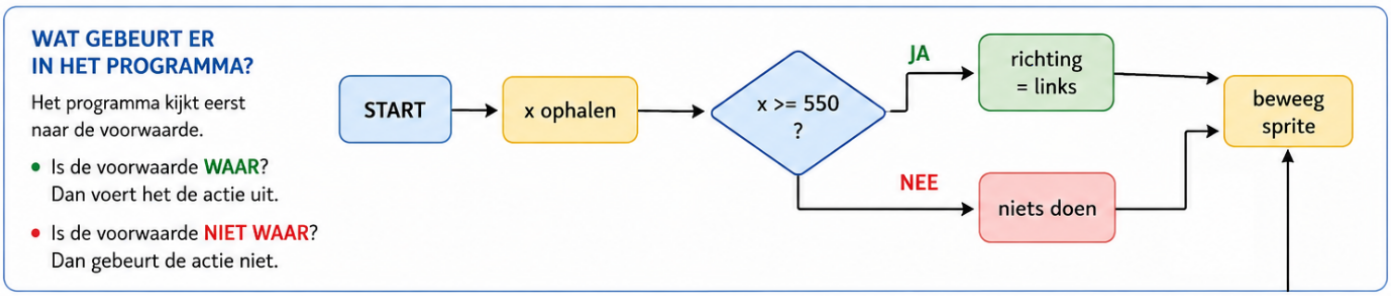
❌ Fout voorbeeld

```
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:s

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.

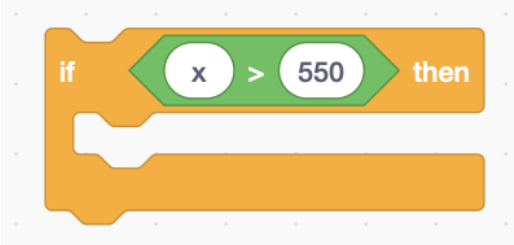
```
from scratch_lib import create_sprite, move, run_animation, get_x

# 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



if-then-else loop

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.

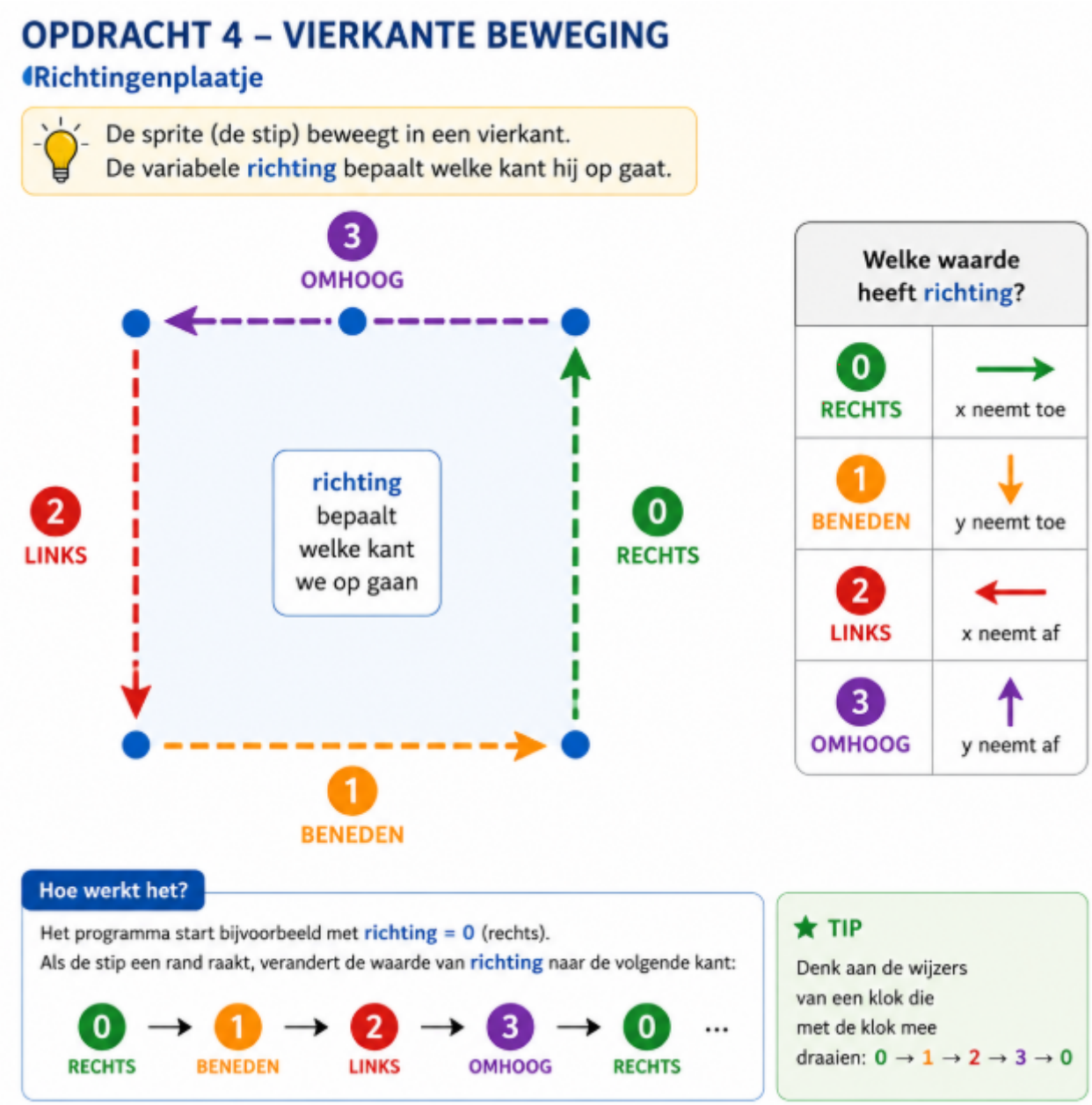
Begincode

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)
```

Hoe werkt dit?



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

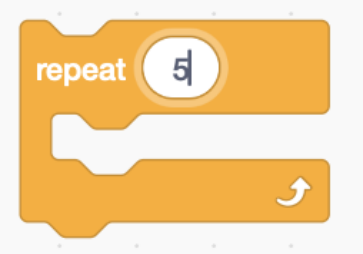
```
for i in range(5):
    move(sprite, 5, 0)
```

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:



> *Herhaal 5 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(5):
    print("Hallo")
```

Uitvoer:

```
Hallo
Hallo
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).

?? 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 2X (in plaats van 5X) 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, run_animation, get_x, get_y, force_redraw
import time

# Startpositie linksboven
sprite = create_sprite("green_dot.png", 20, 20)

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

def animate():
    global richting
    global max_waarde
    global min_waarde

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

    if richting == 0:        # naar rechts
        move(sprite, 5, 0)   # verplaats 5 stapjes over x-as (rechts)
        if x >= 560:         # als we aan de rechterkant zijn dan....
            richting = 1     # pas richting aan: naar beneden

    elif richting == 1:      # naar beneden
        move(sprite, 0, 5)   # verplaats 5 stapjes over y-as (beneden)
        if y >= 560:         # als we onderaan zijn dan....
            richting = 2     # pas richting aan: naar links

    elif richting == 2:      # naar links
        move(sprite, -5, 0)  # verplaats -5 stapjes over y-as (boven)
        if x <= 20:          # als we bovenaan zijn dan ....
            richting = 3     # volgende richting: naar boven

    elif richting == 3:      # naar boven
        move(sprite, 0, -5)  # verplaats -5 over x-as (links)
```

```
        if y <= 20:          # als we aan de linkerkant zijn dan....
            richting = 0     # opnieuw naar rechts


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

?? Uitleg

- De run_animation draait 1000x en telkens wordt de sprite verplaatst.
- Als de richting 0 is dan verplaatst de sprite naar rechts omdat de **x waarde** van de sprite met 5 wordt **verhoogd**.
- Als de richting 1 is dan verplaatst de sprite naar beneden omdat de **y waarde** van de sprite met 5 wordt **verhoogd**.
- Als de richting 2 is dan verplaatst de sprite naar link omdat de **x waarde** van de sprite met 5 wordt **verlaagd**.
- Als de richting 3 is dan verplaatst de sprite naar boven omdat de **y waarde** van de sprite met 5 wordt **verlaagd**.
- Telkens als de x groter is dan **560** of kleiner dan **20** wordt de richting veranderd.
- Telkens als de y groter is dan **560** of kleiner dan **20** dan wordt de richting veranderd.
- De vaiabele min_waarde en max_waarde gerbuiken we nog niet.

?? Opdracht

1. Vervang de 560 op regel 23 en regel 28 door de variabele max_value en geef de max_value de juiste waarde zodat de code hetzelfde blijft doen.
2. Test je code!
3. Daarna vervang de 20 op regel 33 en regel 38 door de min_waarde en geef de min_waarde de jusite waarde zodat de code hetzelfde blijft doen.
4. Test je code!
5. Stel we maken maken min_waard 120 en max_waarde 460, wat gebeurt er?
6. Test je code!

Heb je al een idee hoe je een soort spiraal kan maken zodat het vierkant dat het bolletje maakt steeds kleiner wordt?

? Inleveren

Vertel kort wat er gebeurd als je stap 5 van de opdracht hebt uitgevoerd.

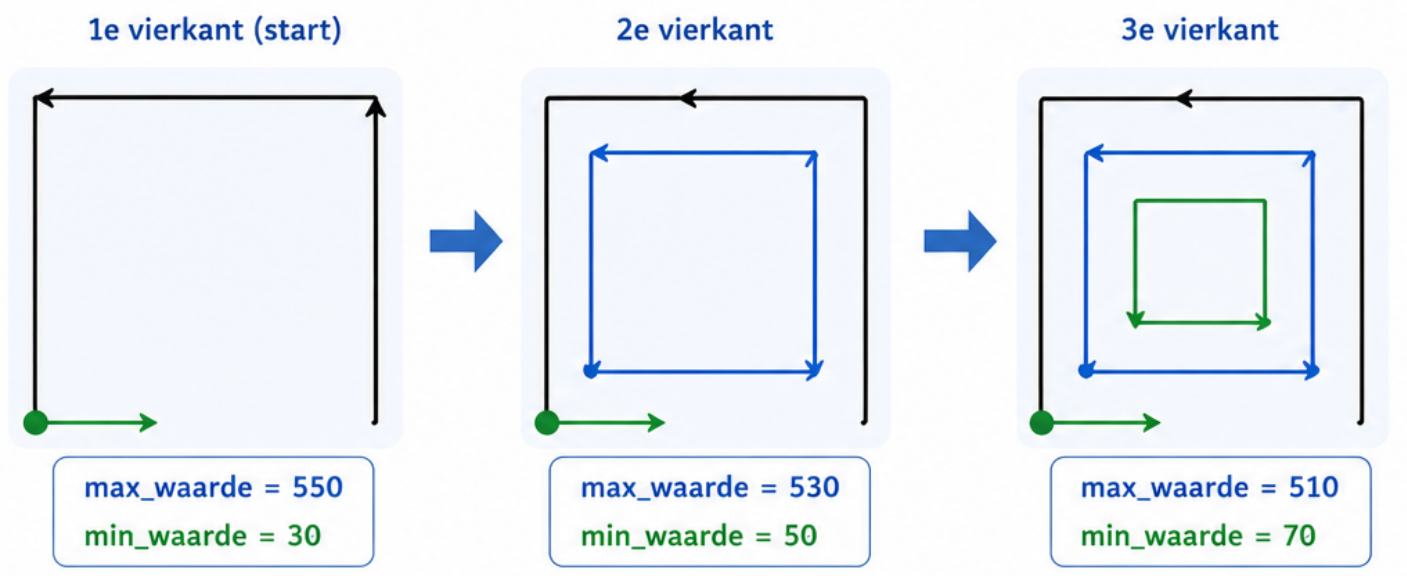
Lever txt-bestand (notepad/kladblok) in.

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 in een soort spiraal naar binnen loopt.



Wat gebeurt er?

1. We tekenen een vierkant met de buitenranden: `max_waarde` (rechts en onder) en `min_waarde` (links en boven).
2. Daarna maken we het vierkant kleiner: `max_waarde -= 20` en `min_waarde += 20`
3. Dan tekenen we opnieuw. Zo ontstaat de spiraal naar binnen.

De kern van de code

```
max_waarde = 550
min_waarde = 30
while max_waarde > min_waarde:
    teken_vierkant(max_waarde, min_waarde)
    max_waarde -= 20    # 20 van rechts en onder eraf
    min_waarde += 20    # 20 van links en boven erbij
```

Waarom werkt dit? Omdat het gebied waarin we tekenen elke keer kleiner wordt: we halen 20 pixels weg van rechts en onder (`max_waarde -= 20`) en we halen 20 pixels weg van links en boven (`min_waarde += 20`).

? Begincode

```
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", 20, 20)

# We gebruiken een getal om de richting bij te houden:
# 0 = rechts, 1 = naar beneden, 2 = naar links, 3 = omhoog
richting = 0
max_waarde=560
min_waarde=20

def animate():
    global richting
    global max_waarde
    global min_waarde

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

    if richting == 0:        # naar rechts
        move(sprite, 5, 0)   # verplaats 5 stapjes over x-as (rechts)
        if x >= max_waarde:  # als we aan de rechterkant zijn dan....
            richting = 1     # pas richting aan: naar beneden

    elif richting == 1:      # naar beneden
        move(sprite, 0, 5)   # verplaats 5 stapjes over y-as (beneden)
        if y >= max_waarde:         # als we onderaan zijn dan....
            richting = 2     # pas richting aan: naar links

    elif richting == 2:      # naar links
        move(sprite, -5, 0)  # verplaats -5 stapjes over y-as (boven)
        if x <= min_waarde:  # als we bovenaan zijn dan ....
            richting = 3     # volgende richting: naar boven

    elif richting == 3:      # naar boven
        move(sprite, 0, -5)  # verplaats -5 over x-as (links)
```

```
        if y <= min_waarde:  # als we aan de linkerkant zijn dan....
            richting = 0     # opnieuw naar rechts
                             # plaats hier code om de spiraal kleiner te maken

run_animation([sprite], animate, steps=4000)
```

?? Uitleg

□□ De code is bijna hetzelfde als de vorige code. Wil je toch nog een keer uitleg, vraag dan AI:

Prompt

```
Kan je deze code uitleggen?


<plak hier de code>
```

Snap je de uitleg niet? Stel dan een vervolgvraag of vraag of de AI het eenvoudiger kan uitleggen.

?? Opdracht

De opdracht is om vlak nadat de ricthing is veranderd naar richting 0 (naar rechts), maken we de min_waarde iets groter en de max_waarde iets kliener. Dat doen we dan elke keer weer als de richting naar 0 (naar rechts) veranderd.

□□ Vraag aan AI hoe je de waarde van een variabele groter en kleiner maakt. Bijvoorbeeld door deze prompt te gebruiken:

Prompt

```
Hoe maak ik de waarde van de variabele in Python iets groter?
```

? Inleveren

Maak een screenshot van je aangepaste code (alleen het stukje dat je hebt aangepast is goed).

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

Gebruik de code die je in de vorige opdracht heb gemaakt.

?? Uitleg

We verlagen telkens onze `max_waarde` en we vergogen telkens onze `min_waarde`.

Om dit te stoppen stellen we een grens, bijvoorbeeld:

De `max_waarde` mag niet onder de 300 komen en

De `min_waarde` mag niet boven de 200 komen.

?? Opdracht

Maak twee condities (met een if) waarbij je:

1. De min_waarde alleen verhoogd als de `min_waarde < 200` is.
2. De max_waarde alleen verlaagd als de `max_waarde > 300` is.

□□ Je mag AI om hulp vragen, maar je moet wel zelf de prompt (vraag) bedenken!

? Inleveren

Maak een screenshot van jouw code met de twee condities.

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?

?? 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. **Welke 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)
```

Revision #51
Created 2025-05-21 11:02:30 UTC by Max
Updated 2026-07-05 08:41:26 UTC by Max