xxx Van Scratch naar Python xxx

Status: ai-alternatief, niet getest maar te complex

0 Wat gaan we leren?

In deze eerste les krijg je een overzicht van wat je in de komende weken gaat leren.

We gaan aan de slag met Python: een echte programmeertaal die je stap voor stap gaat begrijpen door te werken met bewegingen, logica en herhalingen.

□□ Wat leer je in deze module?

- □ Begrijpen waarom en hoe je moet **inspringen**
- □ Leren wat **commentaar**
- □ Inzicht krijgen in het gebruik van **if-statements** (beslissingen in je code)
- □ Leren hoe je **herhalingen** maakt met loops (lussen)

□□ Wat is het verschil met Scratch?

In Scratch gebruik je blokjes die je kunt slepen. In Python schrijf je zelf regels code.

Maar de logica is hetzelfde. Kijk maar:

In Scratch	In Python
herhaal 10 keer	`for i in range(10):`
als *voorwaarde* dan	`if voorwaarde:`

In Scratch	In Python
zet x op 100	`x = 100`

□□ Belangrijke basisprincipes

Lees dit vast door, begrijp je het nog niet helemaal dan is er niets aan de hand want we gaan er in deze les in detail op in.

□□ Inspringen (indentation)

Python gebruikt inspringen (meestal 4 spaties) om aan te geven welke regels bij elkaar horen.

```
if x > 10:
    print("x is groter dan 10")  # deze regel hoort bij de if
print("klaar")  # deze regel hoort NIET bij de if
```

□□ Commentaar

Met een hekje `#` kun je uitleg toevoegen aan je code:

```
# Dit is commentaar – de computer doet hier niets mee
x = 5  # We geven x de waarde 5
```

□□ If-statements

Met een `if`-statement laat je de computer beslissingen nemen:

```
if score >= 10:
    print("Je hebt gewonnen!")
```

□□ Loops (herhalingen)

Met een `for`-loop kun je iets meerdere keren herhalen:

```
for i in range(5):
    print("Dit gebeurt 5 keer")
```

□□ Opdracht

xxx

□□ Inleveren

xxx

1 Installatie van Python (Thonny) en pygame

Voordat we kunnen beginnen met programmeren, moeten we een paar dingen installeren.

We gaan werken met **Thonny**, een eenvoudige Python-omgeving die speciaal geschikt is voor beginners.

Daarnaast gebruiken we **pygame** (of pygame zero) om visuele dingen zoals sprites en bewegingen toe te voegen aan onze code.

□□ Leerdoelen

- Installeren van de Thonny IDE voor Python-programmeren
- Installeren van de pygame-library voor grafische toepassingen
- Startcode downloaden en uitvoeren om de basisstructuur van een Python-programma te begrijpen

□□ Stap 1: Thonny installeren

Ga naar de officiële website van Thonny: https://thonny.org

Klik op de juiste downloadknop voor jouw besturingssysteem (Windows/macOS/Linux).

Na installatie kun je Thonny openen. Het ziet er ongeveer zo uit:

Thonny screenshot

Image not found or type unknown

□□ Stap 2: pygame (zero) installeren

In Thonny ga je naar het menu **Tools → Manage packages...**

Typ daar `pgzero` in (dit is de naam van de pygame zero bibliotheek).

Klik op **Install** om het te installeren.

Installeren van pgzero
Image not found or type unknown

□□ Stap 3: Startcode downloaden en uitvoeren

Download de startcode via de volgende link:

□□ startcode-python-scratch.zip

Pak het zip-bestand uit op een logische plek, bijvoorbeeld je bureaublad.

Open het bestand `student.py` in Thonny.

Druk op de groene Run-knop ▶ om het programma uit te voeren.

□□ Uitleg van de structuur

- `draw()` – hiermee teken je iets op het scherm
- `update()` – hiermee kun je bewegingen of acties herhalen
- `screen.blit()` – hiermee plaats je een afbeelding (sprite)

□□ Opdracht

- Installeer Thonny op je computer
- Installeer de pygame zero bibliotheek via Manage Packages
- Download en open het bestand `student.py`
- Run het programma en kijk wat er gebeurt

□□ Inleveren

Maak een screenshot van het programma in actie (groene stip of sprite op het scherm zichtbaar).

Maak ook een screenshot van je Thonny-scherm met geopende code.

2 De stuiterbal

In deze les maken we een simpele animatie: een groene bal beweegt naar rechts en stuitert terug zodra hij de rand raakt.

Hiermee leer je hoe Python werkt met inspringing (indentation) en vergelijk je dat met hoe Scratch blokken groepeert.

□□ Leerdoelen

- Begrijpen van het concept ‘inspringen’ (indentation) in Python en het belang ervan
- Vergelijken van Python-code met Scratch-blokken
- Leren hoe je een eenvoudige animatie maakt waarbij een bal stuitert

□□ Voorbeeldcode

```
from pgzrun import go
from pygame import Rect

WIDTH = 600
HEIGHT = 400

x = 100
y = 200
richting = "rechts"

def draw():
    screen.clear()
    screen.draw.filled_circle((x, y), 30, "green")

def update():
    global x, richting
```

```
    if richting == "rechts":
        x += 5
    if x >= WIDTH:
        richting = "links"
    if richting == "links":
        x -= 5

go()
```

ℹ Uitleg van de code

- `x` bepaalt de horizontale positie van de bal
- `update()` beweegt de bal en verandert de richting bij de rand
- `if ...:` werkt als “als ... dan” in Scratch
- **Let op de inspringing!** Alles wat bij een `if` hoort moet 4 spaties inspringen

□□ Vergelijking met Scratch

Scratch	Python
als x > 500 dan □ verander richting	if x > 500: richting = "links"
herhaal steeds	def update():

□□ Opdracht

- Voer de code uit in Thonny
- Verander de beginsnelheid naar 10
- Laat de bal de andere kant op starten (richting = "links")
- Voeg een extra `print(x)` toe om te zien hoe x verandert

□□ Extra uitdaging

- Laat de bal ook stuiteren aan de linkerkant (x <= 0)
- Verander de kleur als de bal van richting verandert

□□ Inleveren

Maak een screenshot van je bal terwijl hij stuitert.

Schrijf in je eigen woorden wat `if x >= WIDTH:` doet, en wat er zou gebeuren als je de inspringing vergeet.

3 De stuiterbal – heen en weer

In deze les laten we de bal netjes heen en weer bewegen tussen de linker- en rechterrand van het scherm.

Je leert hoe je if-statements en vergelijkingsoperatoren combineert om dit voor elkaar te krijgen.

□□ Leerdoelen

- Gebruik van if-statements om bewegingen te controleren
- Begrijpen van vergelijkingsoperatoren in Python (zoals `>`, `<`, `==`)
- Implementeren van logica om een object heen en weer te laten bewegen

□□ Codevoorbeeld

```
from pgzrun import go

WIDTH = 600
HEIGHT = 400

x = 100
y = 200
richting = "rechts"

def draw():
    screen.clear()
    screen.draw.filled_circle((x, y), 30, "green")

def update():
```

```
    global x, richting
    if richting == "rechts":
        x += 5
        if x >= WIDTH - 30:
            richting = "links"
    if richting == "links":
        x -= 5
        if x <= 30:
            richting = "rechts"

go()
```

ℹ Uitleg van de code

- `richting` bepaalt of we naar links of rechts bewegen
- Als de bal de rechterrand raakt, verandert `richting` naar "links"
- Als de bal de linkerrand raakt, verandert `richting` naar "rechts"
- We gebruiken `if`-statements én vergelijkingen zoals `>=` en `<=`

□□ Opdracht

- Voer de code uit en controleer of de bal heen en weer beweegt
- Verander de waarde 30 naar 50 – wat gebeurt er?
- Maak de bal groter of kleiner en pas de randen aan

□□ Extra uitdaging

- Laat de bal ook van kleur veranderen telkens als hij van richting verandert
- Laat de bal sneller of langzamer bewegen bij elke botsing

□□ Inleveren

Maak een screenshot van het spel waarin de bal duidelijk een andere richting op beweegt.

Schrijf in één zin uit waarom er twee `if`-blokken zijn en wat hun functie is.

4 De vierkante beweging

In deze les gaan we de sprite laten bewegen in een vierkant patroon: eerst naar rechts, dan naar beneden, dan naar links en dan weer omhoog.

Je leert hoe je meerdere richtingen beheert met `if`-statements en hoe je beweging op basis van coördinaten controleert.

□□ Leerdoelen

- Creëren van bewegingen in een vierkant patroon
- Gebruik van meerdere `if`-statements om richtingen te bepalen
- Versterken van begrip over coördinatensystemen en beweging in twee dimensies

□□ Codevoorbeeld

```
from pgzrun import go

WIDTH = 600
HEIGHT = 400

x = 100
y = 100
richting = "rechts"

def draw():
    screen.clear()
    screen.draw.filled_circle((x, y), 30, "orange")

def update():
    global x, y, richting
```

```
    if richting == "rechts":
        x += 5
        if x >= 500:
            richting = "beneden"
    elif richting == "beneden":
        y += 5
        if y >= 300:
            richting = "links"
    elif richting == "links":
        x -= 5
        if x <= 100:
            richting = "boven"
    elif richting == "boven":
        y -= 5
        if y <= 100:
            richting = "rechts"


go()
```

ℹ Uitleg van de code

- De sprite beweegt per update in een bepaalde richting
- Als de sprite een hoekpunt bereikt, verandert de richting
- De volgorde is: rechts → beneden → links → boven → herhaal
- We gebruiken meerdere `elif`-blokken voor duidelijke richtinglogica

□□ Opdracht

- Voer de code uit en bekijk de beweging van de sprite
- Verander de vierkantsgrootte (bijvoorbeeld naar 400 bij 200)
- Laat de sprite sneller of trager bewegen door de stappen aan te passen

□□ Extra uitdaging

- Laat de sprite een ander kleurtje krijgen bij elke hoek

- Tel hoe vaak de sprite een vierkant heeft voltooid (met een `teller`)

□□ Inleveren

Maak een screenshot van het spel waarin je sprite zich in een vierkant beweegt.

Schrijf in één zin hoe de richting wordt aangepast en waarom de sprite stopt of draait bij een bepaalde waarde.

5 Vierkant met sprongen op elke hoek

In deze les voegen we sprongen toe aan het vierkante bewegingspatroon.

Telkens als de sprite een hoek bereikt, laat hij een korte sprong omhoog en omlaag zien.

We gebruiken een **`for`-loop** om deze herhaalde sprongbeweging te programmeren.

□□ Leerdoelen

- Introductie tot `for`-loops in Python
- Automatiseren van herhaalde acties met behulp van loops
- Vergelijken van loops in Python met herhalingsblokken in Scratch

□□ Codevoorbeeld

```
from pgzrun import go
import time

WIDTH = 600
HEIGHT = 400

x = 100
y = 100
```

```
richting = "rechts"

def draw():
    screen.clear()
    screen.draw.filled_circle((x, y), 30, "orange")

def spring():
    global y
    for i in range(5):
        y -= 10
        time.sleep(0.05)
        y += 10
        time.sleep(0.05)

def update():
    global x, y, richting

    if richting == "rechts":
        x += 5
        if x >= 500:
            richting = "beneden"
            spring()
    elif richting == "beneden":
        y += 5
        if y >= 300:
            richting = "links"
            spring()
    elif richting == "links":
        x -= 5
        if x <= 100:
            richting = "boven"
            spring()
    elif richting == "boven":
        y -= 5
        if y <= 100:
            richting = "rechts"
            spring()

go()
```

ℹ Uitleg van de code

- `for i in range(5):` herhaalt iets 5 keer
- De functie `spring()` laat de sprite op en neer bewegen
- We gebruiken `time.sleep()` om het effect van een korte sprong te laten zien
- De sprong wordt telkens uitgevoerd bij een hoek van het vierkant

□□ Vergelijking met Scratch

Scratch	Python

herhaal 5 keer
□ verander y met -10
□ wacht 0.05 sec
□ verander y met 10
	`for i in range(5):`
	`y -= 10`
	`sleep(0.05)`
	`y += 10`

□□ Opdracht

- Voer de code uit en bekijk het sprongeffect
- Pas `range(5)` aan naar `range(3)` of `range(10)` en bekijk het verschil
- Verander de spronghoogte (bijv. 20 pixels)

□□ Extra uitdaging

- Laat de sprite een geluid afspelen bij elke sprong (indien geluid is ingesteld)
- Laat alleen bij de rechterbovenhoek een sprong uitvoeren

□□ Inleveren

Maak een screenshot van je sprite bij een hoek van het vierkant.

Schrijf in één zin uit wat de `for`-loop doet in de functie `spring()`.

6 Spring vaker op twee hoeken

In deze les laat je de sprite **alleen op bepaalde hoeken** van het vierkant springen. Zo leer je hoe je **voorwaardelijke logica** (if-statements) combineert met herhaling (loops).

□□ Leerdoelen

- Verfijnen van bewegingen met behulp van loops en conditionele logica
- Begrijpen van geneste structuur: een `if`-statement binnen een functie met een loop
- Toepassen van eerder geleerde concepten in een nieuwe context

□□ Codevoorbeeld

```
from pgzrun import go
import time

WIDTH = 600
HEIGHT = 400

x = 100
y = 100
richting = "rechts"

def draw():
    screen.clear()
    screen.draw.filled_circle((x, y), 30, "purple")

def spring():
    for i in range(3):
        global y
        y -= 15
        time.sleep(0.05)
        y += 15
        time.sleep(0.05)

def update():
    global x, y, richting

    if richting == "rechts":
```

```
        x += 5
        if x >= 500:
            richting = "beneden"
            spring()
    elif richting == "beneden":
        y += 5
        if y >= 300:
            richting = "links"
            # geen sprong hier
    elif richting == "links":
        x -= 5
        if x <= 100:
            richting = "boven"
            spring()
    elif richting == "boven":
        y -= 5
        if y <= 100:
            richting = "rechts"
            # geen sprong hier

go()
```

ℹ Uitleg van de code

- De `spring()`-functie wordt nu **alleen** op twee hoeken uitgevoerd
- Bijvoorbeeld: rechterbovenhoek en linkeronderhoek
- Je gebruikt dus `if`-logica om te kiezen *wanneer* je de loop uitvoert

□□ Opdracht

- Laat je sprite alleen springen bij de hoeken rechtsboven en linksonder
- Verander de `range()` in de `spring()`-functie naar een groter of kleiner getal
- Laat de kleur veranderen tijdens de sprong (optioneel: met een extra variabele)

□□ Extra uitdaging

- Tel in een variabele hoeveel keer er gesprongen is en toon dit met `screen.draw.text()`
- Maak het springen afhankelijk van een variabele zoals `hoek_nummer`

□□ Inleveren

Maak een screenshot van de sprite terwijl hij springt bij één van de actieve hoeken.

Beschrijf in 1 of 2 zinnen hoe je bepaalt **waar** de sprong wel of niet plaatsvindt.

7 Spiraal – stap 1

In deze les gaan we de sprite in een spiraal laten bewegen. Bij elke stap draait hij een hoek op, en de afstand die hij aflegt blijft (voor nu) steeds hetzelfde.

Het effect: een vierkante spiraalvormige beweging.

□□ Leerdoelen

- Creëren van een spiraalvormig bewegingspatroon
- Gebruik van vaste richtingen en patronen in loops
- Begrijpen van hoe kleine aanpassingen in code grote visuele effecten kunnen hebben

□□ Codevoorbeeld

```
from pgzrun import go
import time

WIDTH = 600
HEIGHT = 400

x = 300
y = 200
richting = "rechts"
afstand = 50
```

```
def draw():
    screen.clear()
    screen.draw.filled_circle((x, y), 20, "blue")

def update():
    global x, y, richting, afstand

    if richting == "rechts":
        x += afstand
        richting = "beneden"
    elif richting == "beneden":
        y += afstand
        richting = "links"
    elif richting == "links":
        x -= afstand
        richting = "boven"
    elif richting == "boven":
        y -= afstand
        richting = "rechts"

    time.sleep(0.4)  # animatie vertragen

go()
```

ℹ Uitleg van de code

- De sprite beweegt elke keer een vaste afstand in een nieuwe richting
- De richting verandert steeds in de volgorde: rechts → beneden → links → boven → herhaal
- Met `time.sleep()` vertraag je de stappen zodat je het goed kunt volgen

□□ Opdracht

- Voer de code uit – je ziet een sprite een vierkant patroon volgen
- Verander de beginwaarde van `afstand` naar 30 of 70

- Laat het middelpunt (x, y) ergens anders beginnen, bijvoorbeeld linksonder of rechtsboven

□□ Extra uitdaging

- Laat de sprite een spoor tekenen (bijvoorbeeld met een lijst van oude posities)
- Laat de kleur per stap veranderen

□□ Inleveren

Maak een screenshot van de sprite tijdens de spiraalbeweging.

Schrijf in één zin uit wat er gebeurt als je `richting` niet telkens verandert.

8 Spiraal – stap 2

In deze les ga je de sprite niet alleen in een spiraal laten bewegen, maar bij elke stap ook de **afstand iets kleiner maken**.

Het resultaat is een spiraal die langzaam naar het midden krult.

□□ Leerdoelen

- Aanpassen van spiraalbewegingen door afstanden te verkleinen
- Gebruik van variabelen om bewegingen dynamisch te maken
- Versterken van begrip over loops en variabele manipulatie

□□ Codevoorbeeld

```
from pgzrun import go
import time

WIDTH = 600
HEIGHT = 400
```

```
x = 300
y = 200
richting = "rechts"
afstand = 80

def draw():
    screen.clear()
    screen.draw.filled_circle((x, y), 20, "teal")

def update():
    global x, y, richting, afstand

    if richting == "rechts":
        x += afstand
        richting = "beneden"
    elif richting == "beneden":
        y += afstand
        richting = "links"
    elif richting == "links":
        x -= afstand
        richting = "boven"
    elif richting == "boven":
        y -= afstand
        richting = "rechts"

    afstand -= 5  # afstand verkleinen bij elke stap
    if afstand <= 0:
        afstand = 0  # stop op 0, anders negatief

    time.sleep(0.4)

go()
```

ℹ Uitleg van de code

- Bij elke stap verandert de richting én wordt de `afstand` 5 kleiner
- Als de afstand 0 bereikt, stopt de sprite effectief met bewegen
- Deze aanpak simuleert een vierkante spiraal die naar binnen draait

□□ Opdracht

- Voer de code uit en bekijk het effect van de afnemende afstand
- Probeer met `afstand -= 2` of `afstand -= 10` — hoe beïnvloedt dit de spiraal?
- Verander de startpositie (x, y) en bekijk hoe het patroon verschuift

□□ Extra uitdaging

- Teken het spoor van de sprite (gebruik een lijst met posities)
- Laat de sprite stoppen als de afstand kleiner is dan 10

□□ Inleveren

Maak een screenshot van jouw spiraal tijdens of net voor het einde van de beweging.

Schrijf kort op hoe de `afstand -= 5` regel het gedrag van de sprite beïnvloedt.

8 Spiraal – stap 2

In deze les ga je de sprite niet alleen in een spiraal laten bewegen, maar bij elke stap ook de **afstand iets kleiner maken**.

Het resultaat is een spiraal die langzaam naar het midden krult.

□□ Leerdoelen

- Aanpassen van spiraalbewegingen door afstanden te verkleinen
- Gebruik van variabelen om bewegingen dynamisch te maken
- Versterken van begrip over loops en variabele manipulatie

□□ Codevoorbeeld

```
from pgzrun import go
import time
```

```
WIDTH = 600
HEIGHT = 400

x = 300
y = 200
richting = "rechts"
afstand = 80

def draw():
    screen.clear()
    screen.draw.filled_circle((x, y), 20, "teal")

def update():
    global x, y, richting, afstand

    if richting == "rechts":
        x += afstand
        richting = "beneden"
    elif richting == "beneden":
        y += afstand
        richting = "links"
    elif richting == "links":
        x -= afstand
        richting = "boven"
    elif richting == "boven":
        y -= afstand
        richting = "rechts"

    afstand -= 5  # afstand verkleinen bij elke stap
    if afstand <= 0:
        afstand = 0  # stop op 0, anders negatief

    time.sleep(0.4)

go()
```

ℹ Uitleg van de code

- Bij elke stap verandert de richting én wordt de `afstand` 5 kleiner
- Als de afstand 0 bereikt, stopt de sprite effectief met bewegen
- Deze aanpak simuleert een vierkante spiraal die naar binnen draait

□□ Opdracht

- Voer de code uit en bekijk het effect van de afnemende afstand
- Probeer met `afstand -= 2` of `afstand -= 10` — hoe beïnvloedt dit de spiraal?
- Verander de startpositie (x, y) en bekijk hoe het patroon verschuift

□□ Extra uitdaging

- Teken het spoor van de sprite (gebruik een lijst met posities)
- Laat de sprite stoppen als de afstand kleiner is dan 10

□□ Inleveren

Maak een screenshot van jouw spiraal tijdens of net voor het einde van de beweging.

Schrijf kort op hoe de `afstand -= 5` regel het gedrag van de sprite beïnvloedt.

9 Spiraal – stap 3

Tot nu toe liet je de sprite bewegen in een steeds kleinere spiraal. In deze les leer je hoe je het programma automatisch laat stoppen met een `break`-statement.

Zo kun je zelf bepalen wanneer een herhaling moet stoppen op basis van een voorwaarde.

□□ Leerdoelen

- Implementeren van een automatische stopconditie voor loops
- Begrijpen van het gebruik van `break`-statements in Python
- Creëren van programma’s die zelfstandig kunnen stoppen op basis van een variabele

□□ Codevoorbeeld

```
import pgzrun
import time

WIDTH = 600
HEIGHT = 400

x = 300
y = 200
richting = "rechts"
afstand = 80

bewegingen = []

def draw():
    screen.clear()
    screen.draw.filled_circle((x, y), 20, "green")

def update():
    global x, y, richting, afstand

    if afstand <= 5:
        print("Spiraal gestopt.")
        exit()

    if richting == "rechts":
        x += afstand
        richting = "beneden"
    elif richting == "beneden":
        y += afstand
        richting = "links"
    elif richting == "links":
        x -= afstand
        richting = "boven"
    elif richting == "boven":
        y -= afstand
        richting = "rechts"
```

```
    afstand -= 5
    time.sleep(0.4)


pgzrun.go()
```

ℹ Uitleg van de code

- De afstand wordt steeds kleiner
- Als de afstand ≤ 5 is, wordt het programma automatisch gestopt met `exit()`
- Alternatief: gebruik `break` in een `while True`-loop om zelf meer controle te houden

□□ Variant me`while`-loop en `break`

```
while True:
    if afstand <= 5:
        break
    # beweging uitvoeren
    ...
```

□□ Opdracht

- Laat de sprite stoppen zodra de spiraal "te klein" wordt
- Gebruik `exit()` of `break` om dit netjes af te sluiten
- Print “Spiraal afgelopen” op het scherm of in de console

□□ Extra uitdaging

- Laat de sprite een tekst tonen op het scherm zodra hij stopt
- Gebruik een teller om te tonen hoeveel stappen zijn uitgevoerd

□□ Inleveren

Maak een screenshot van de laatste positie van de sprite vóór het stoppen.

Schrijf in één zin uit waarom en wanneer de `break` of `exit()` wordt uitgevoerd.

10 Reflectie: wat heb je geleerd?

Je hebt in de afgelopen lessen veel geleerd over programmeren met Python. In deze afsluitende les kijk je terug op wat je hebt gedaan, wat je hebt geleerd en waar je nog verder in wilt groeien.

Door hierover na te denken, wordt je bewuster van je eigen leerproces en kun je gerichter verder leren.

□□ Leerdoelen

- Terugblikken op de geleerde concepten en vaardigheden
- Identificeren van persoonlijke groeipunten en uitdagingen
- Formuleren van doelen voor verdere ontwikkeling in programmeren

□□ Reflectieopdracht

Beantwoord de volgende vragen in een apart document of onderaan je Python-bestand als commentaar.

- □□ Wat vond je het leukst om te maken of uit te proberen?
- □□ Wat vond je lastig? Hoe heb je dat opgelost?
- □□ Wat heb je geleerd over Python (of over programmeren in het algemeen)?
- □□ Waar ben je trots op in jouw eindresultaat?
- □□ Wat zou je de volgende keer anders willen doen of verbeteren?

□□ Extra vragen (optioneel)

- □□ Waar herken je elementen van Scratch terug in Python?
- □□ Heb je zelf iets toegevoegd aan je code dat niet in de lessen stond?
- □□ Zou je dit project aan een andere leerling aanraden? Waarom (niet)?

□□ Inleveren

- Lever je reflectie in bij je docent (tekstbestand, screenshot of commentaar in code)
- Lever ook je laatste versie van je spiraalproject in

□□ Voorbeeld van reflectie als commentaar

```
# Reflectie:
# Ik vond het leuk dat je de sprite kon laten bewegen met steeds kleinere stappen.
# Het lastigst vond ik de richting goed laten wisselen, maar met hulp lukte het.
# Ik heb geleerd hoe je loops en if-statements kunt combineren.
# Ik ben trots dat mijn spiraal echt stopt aan het eind.
# Volgende keer wil ik proberen met meerdere sprites te werken.
```

---

1 Thonny installeren

□□ Je moet eerst **Thonny** installeren. Dat is een eenvoudige Python-omgeving voor beginners.

Ga naar https://thonny.org en volg de installatie-instructies voor jouw besturingssysteem (Windows, Mac of Linux).

□□ Als Thonny geïnstalleerd is, kun je het openen en een nieuw Python-bestand aanmaken. Plak onderstaande code in je bestand:

```
import pygame
import time

# Begin pygame en maak scherm
pygame.init()
screen = pygame.display.set_mode((500, 500))

# Laad een afbeelding (zorg dat 'bal.png' in dezelfde map staat)
ball = pygame.image.load("bal.png")
x = 100
y = 100
```

```
# Oneindige lus (druk op sluiten om te stoppen)
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Wis het scherm
    screen.fill((255, 255, 255))

    # Teken de bal
    screen.blit(ball, (x, y))

    # Verplaats de bal
    x += 1
    y += 1

    # Wacht even en werk het scherm bij
    pygame.display.flip()
    time.sleep(0.01)

pygame.quit()
```

Je hebt een afbeelding nodig met de naam `bal.png` in dezelfde map als je script. Maak bijvoorbeeld een klein rondje in Paint of een andere tekenapp, sla het op als `bal.png` en zet het in dezelfde map.

Klaar? Druk op **Run** (de groene afspeelknop) in Thonny. Als het goed is, zie je een venster waarin een bal diagonaal beweegt.

Revision #9
Created 22 May 2025 13:04:06 by Max
Updated 6 June 2025 21:06:09 by Max