



# Python

---

Informatica

Renske Smetsers



# Belangrijke aspecten bij coderen

---

- ❑ **Volledigheid:** Heeft het eindproduct alles wat verwacht wordt?
- ❑ **Correctheid:** Doet alles het zoals je zou verwachten?
- ❑ **Constructie:** Hoe goed is de code geschreven?
- ❑ **Stijl:** Is de code leesbaar?

Goed programmeren gaat dus **niet alleen** om een werkende **oplossing**, maar juist om een **degelijke aanpak en oplossing**.

...hier wordt je ook op beoordeeld



# Iteraties (Herhalingen)

---

- While loop
- For loop: als je precies weet hoe vaak je moet gaan herhalen



# While loop

---

```
while < boolean expressie >:  
    < acties >
```

*“Zolang een voorwaarde waar is,  
voer de acties uit”*



# While loop: herhalingen

---

- Zolang een voorwaarde waar is, voer de acties uit.

- Structuur:

```
while < voorwaarde >:  
    < acties >
```

- Bijvoorbeeld:

```
while ikHebHonger:  
    eet()
```

```
while not wachtWoordJuist:  
    voerWachtwoordIn()
```



# Strafregels schrijven

## Opdracht:

Druk 3 keer af: “Ik zal voortaan de opdrachten maken.”

```
while < voorwaarde >:  
    < acties >
```

## Uitvoer:

**1<sup>e</sup> keer:** “Ik zal voortaan de opdrachten maken.”

**2<sup>e</sup> keer:** “Ik zal voortaan de opdrachten maken.”


**3<sup>e</sup> keer:** “Ik zal voortaan de opdrachten maken.”

## Het algoritme:

- Zolang nog geen drie keer geschreven
  - Druk strafregel af



# Standaard opbouw while



Op de volgende sheets kijken we hoe dat precies gaat

Vaak met een teller

## De variabele:

1a) Wat is de variabele? Dus wat verandert er steeds?

1b) Geef de variabele een begin waarde (meestal is dit 0)

## De herhaling `while`:

2a) Bepaal de **voorwaarde** voor herhaling: “Zolang **voorwaarde** herhaal ... “

2b) Bepaal wat je steeds wilt herhalen, de **acties**.

2c) Verander de variabele (meestal verhogen met 1).

```
<variabele initialiseren>
```

```
while <voorwaarde>:
```

```
    <acties>
```

```
    <variabele aanpassen>
```



# Strafregels schrijven

## Het algoritme:

- Zolang nog geen drie keer geschreven
  - Druk strafregel af

Je moet wel bijhouden hoe vaak je al geschreven hebt!

## De variabele:

1a) Wat moet je onthouden? Wat verandert er steeds?

`aantal_strafregels_geschreven`

1b) Variabele beginwaarde

`aantal_strafregels_geschreven = 0`





# Strafregels schrijven

## Het algoritme:

- Zolang nog geen drie keer geschreven
  - Druk strafregel af

## De herhaling `while`:

2a) Bepaal de **voorwaarde** voor herhaling:

```
aantal_strafregels_geschreven < 3
```

2b) Bepaal wat je steeds wilt herhalen, de **acties**.

```
print( "Ik zal voortaan de opdrachten maken." )
```

2c) Verander de variabele (meestal verhogen met 1).

```
aantal_strafregels_geschreven += 1
```



# Strafregels

```
<variabele initialiseren>  
while <voorwaarde>:  
    <acties>  
    <variabele aanpassen>
```

## Opdracht:

Druk 3 keer af: "Ik zal voortaan de opdrachten maken."

## Code:

```
aantal_strafregels_geschreven = 0  
  
while aantal_strafregels_geschreven < 3:  
    print( "Ik zal voortaan de opdrachten maken." )  
    aantal_strafregels_geschreven += 1
```



# Tracing tabel: wat gebeurt er eigenlijk?

```
aantal_strafregels = 0

while aantal_strafregels < 3:
    print( "Ik zal voortaan de opdrachten maken." )
    aantal_strafregels += 1
print( "Klaar!" )
```

Code	Waarde van aantal_strafregels	Wat geprint wordt
<b>aantal_strafregels = 0</b>	0	
<b>while aantal_strafregels &lt; 3 )</b>		
<b>print ( "Ik zal ... " )</b>		"Ik zal ..."
<b>aantal_strafregels += 1</b>		
<b>while aantal_strafregels &lt; 3 )</b>		
<b>print ( "Ik zal ... " )</b>		"Ik zal ..."
<b>aantal_strafregels += 1</b>		
<b>while aantal_strafregels &lt; 3 )</b>		
<b>print ( "Ik zal ... " )</b>		"Ik zal ..."
<b>aantal_strafregels += 1</b>		
<b>print( "Klaar!" )</b>		"Klaar!"



# Wat doet deze code?

---

```
teller = 0
while teller < 100:
    print( "teller: ", str(teller) )
    teller = teller * 1
```

Drukt af:

teller: 0

teller: 0

teller: 0

teller: 0

...

--stopt dus nooit: CRASH!! -



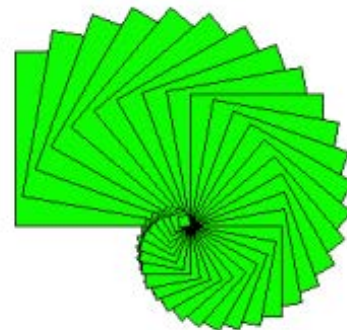
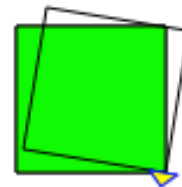
# Iteraties: FOR loop

---

# Bepaalde en onbepaalde herhaling

## □ **While-loop:** Onbepaalde herhaling:

- Zolang ... herhaal
- Aantal keren herhalen afhankelijk van een voorwaarde
- Zorg dat while-loop eindigt! Voorkom oneindige loop!

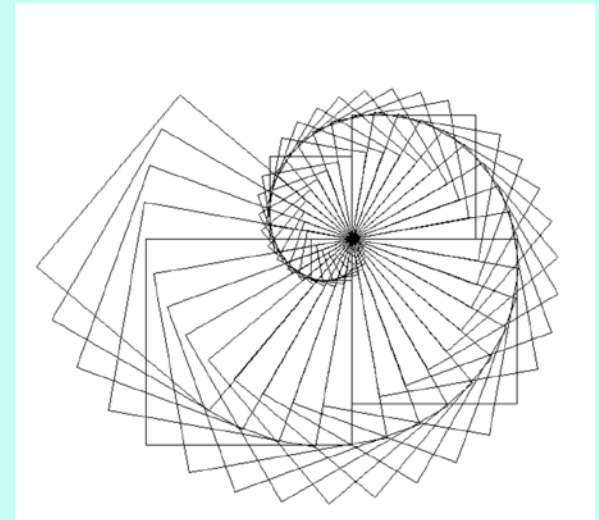


## □ **For-loop:** Bepaalde (of tellende) herhaling:

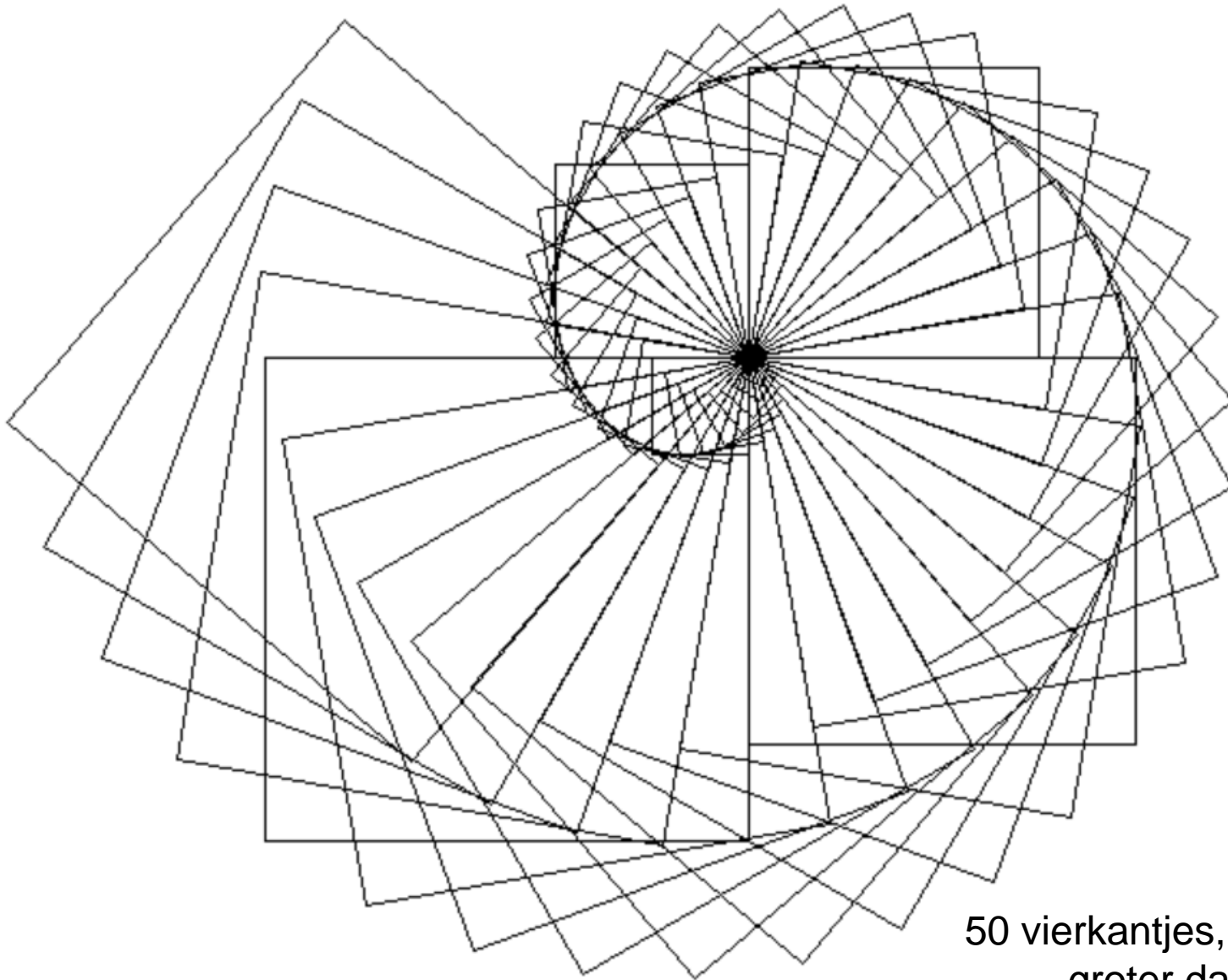
- Herhaal x keer
- Aantal herhalingen vooraf precies bekend
- Voorbeeld 6 keer

# For loops

- ▣ Bepaalde (of tellende) herhaling toepassen



# For loop: bepaald aantal herhalingen



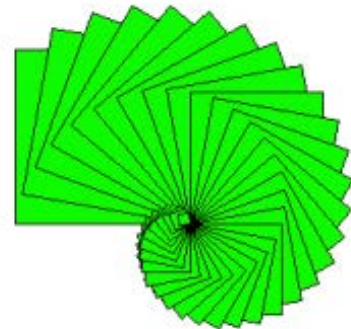
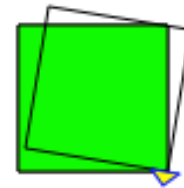
50 vierkantjes, ieder steeds iets  
groter dan de vorige



# Bepaalde en onbepaalde herhaling

## □ **While-loop:** Onbepaalde herhaling:

- Zolang ... herhaal
- Aantal keren herhalen afhankelijk van een voorwaarde
- Zorg dat while-loop eindigt! Voorkom oneindige loop!



## □ **For-loop:** Bepaalde (of tellende) herhaling:

- Herhaal x keer
- Aantal herhalingen vooraf precies bekend
- Voorbeeld 6 keer



# Robot besturen - voorbeeld

- Stel we willen een robot een rondje laten lopen
- Maar een robot loopt niet vanzelf
- Hoe kunnen we dit doen?

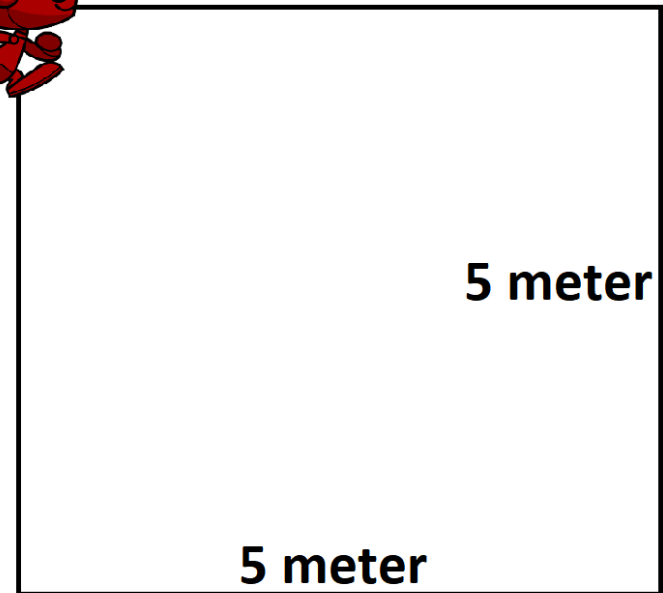
Algoritme:

STAP STAP STAP STAP STAP  
RECHTS

STAP STAP STAP STAP STAP  
RECHTS

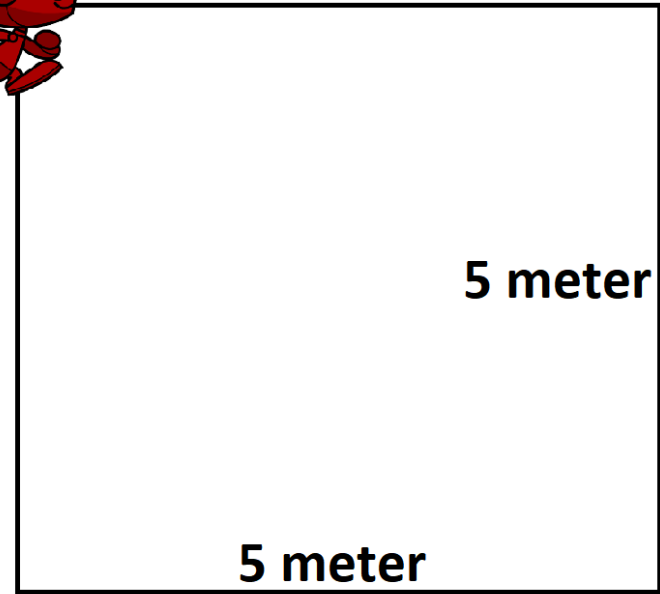
STAP STAP STAP STAP STAP  
RECHTS

STAP STAP STAP STAP STAP  
(RECHTS)





# Robot besturen - voorbeeld



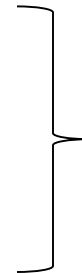
Kunnen we dit slimmer opschrijven?

```
import turtle

def loopVooruitEnDraai( ):
    turtle.forward(5)
    turtle.right(90)

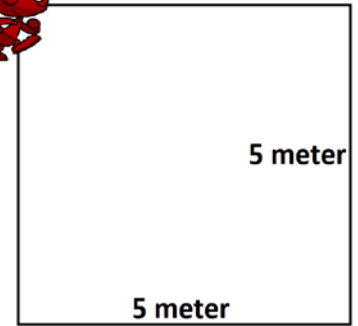
loopVooruitEnDraai()
loopVooruitEnDraai()
loopVooruitEnDraai()
loopVooruitEnDraai()

turtle.done()
```



4x hetzelfde!!

```
Herhaal 4 keer:
    loopVooruitEnDraai()
```



# Robot besturen - voorbeeld

**for** loop: een bepaald aantal keren iets herhalen

```
import turtle

def loopVooruitEnDraai( ):
    turtle.forward(5)
    turtle.right(90)

for i in range(4):
    loopVooruitEnDraai()

turtle.done()
```

Teller: mag je noemen wat je wilt

Hier het aantal keren herhalen

Wat herhaalt moet worden.  
Let op: wel inspringen!



# Goede algoritmen

---

“Een algoritme is goed als het werkt”

Is dit echt zo?

```
import turtle

for i in range(4004):
    turtle.forward(100)
    turtle.right(90)

turtle.done()
```

Probeer efficiënt algoritme te schrijven!

Is vooral belangrijk bij zoeken en sorteren



# For loop voor herhalingen

---

```
for x in range(5):  
    print (“Dit wordt 5 keer afgedrukt!”)
```

## Resultaat

```
Dit wordt 5 keer afgedrukt!  
Dit wordt 5 keer afgedrukt!  
Dit wordt 5 keer afgedrukt!  
Dit wordt 5 keer afgedrukt!  
Dit wordt 5 keer afgedrukt!
```



# For loop

---

Bijvoorbeeld voor het afdrukken van een rij getallen:

```
for x in range(5):  
    print( x )
```

Resultaat:

0

1

2

3

4



# Geneste for loop – loop in een loop

- Als je bijvoorbeeld coördinaten wilt afdrukken

```
for x in range(5):  
    for y in range(5):  
        print(x,y)
```

- Resultaat:

<b>0 0</b>	<b>1 0</b>		<b>4 0</b>
<b>0 1</b>	<b>1 1</b>		<b>4 1</b>
<b>0 2</b>	<b>1 2</b>	<b>...</b>	<b>4 2</b>
<b>0 3</b>	<b>1 3</b>		<b>4 3</b>
<b>0 4</b>	<b>1 4</b>		<b>4 4</b>





# For met range parameters:

`range (begin, einde, stapgrootte)`

**begin:** de eerste waarde (telt wel mee)  
**einde:** de laatste waarde (telt zelf **niet** mee)  
**stapgrootte:** stap

```
for x in range(5):  
    print(x)
```

Resultaat:

0  
1  
2  
3  
4

```
for x in range(1,11):  
    print(x)
```

Resultaat:

1  
2  
3  
..  
10

```
for x in range(1,11,3):  
    print(x)
```

Resultaat:

1  
4  
7  
10