Login: student

wachtwoord: Ohey5350

Hands-on sessie: programmeren

University of Antwerp

Tom Hofkens tom.hofkens@uantwerpen.be

Wat is een computerprogramma ?

Een lijst van instructies in een formele taal om een specifieke taak met een computer uit te voeren

Elektrische toestellen

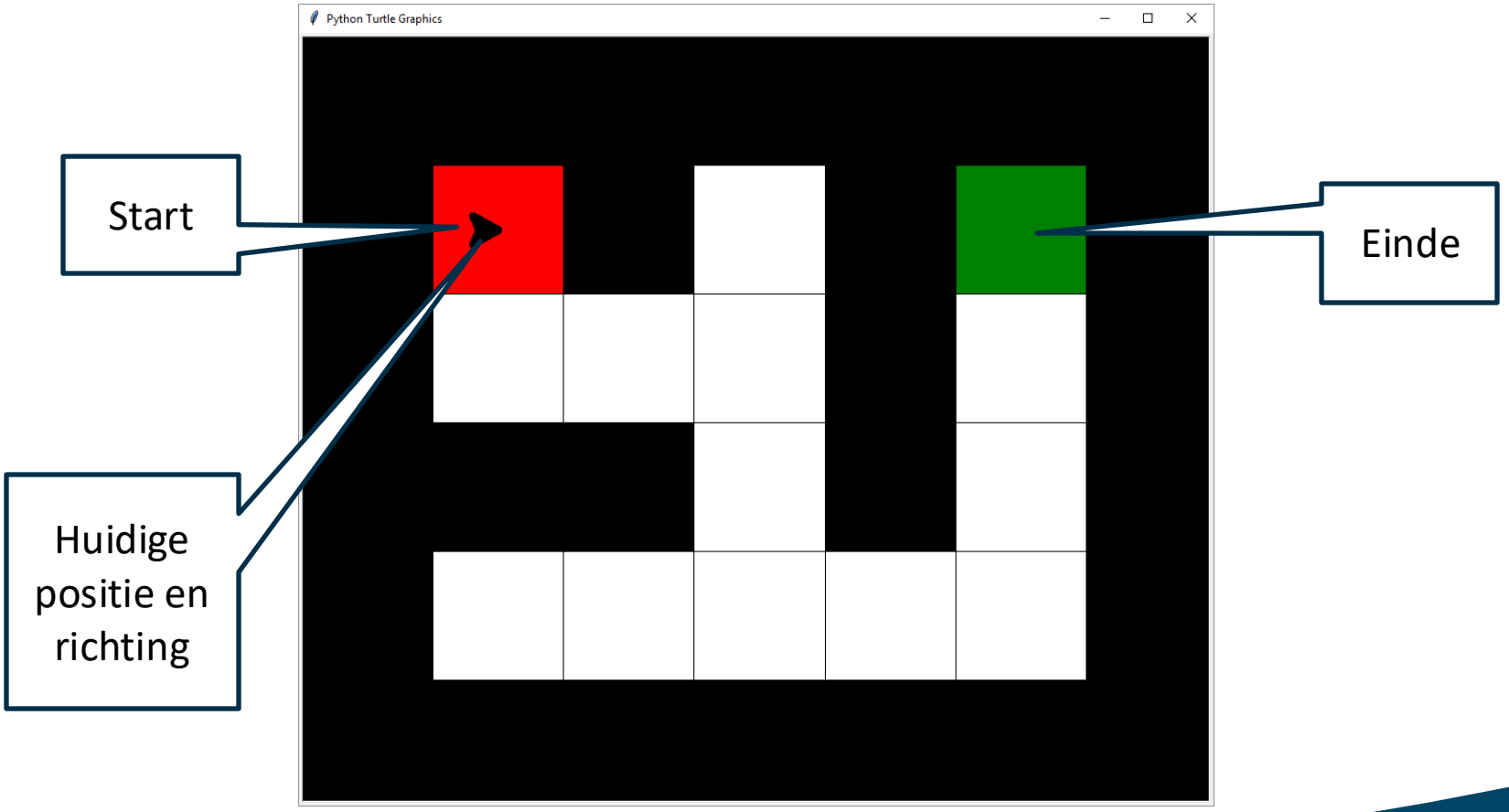
Besturingssysteem

Kantoor software

Games

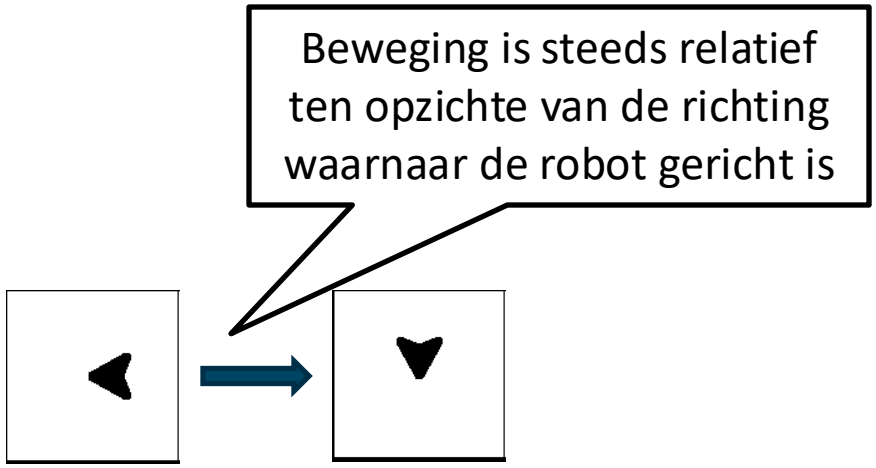
Tijdens deze mini-les maken we …

… een programma in Python waarmee we uit elk (virtueel) labyrint kunnen ontsnappen

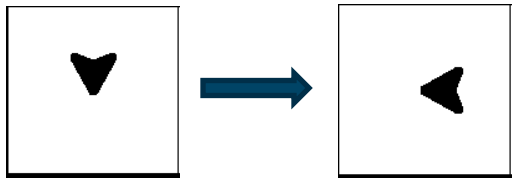


De instructies:

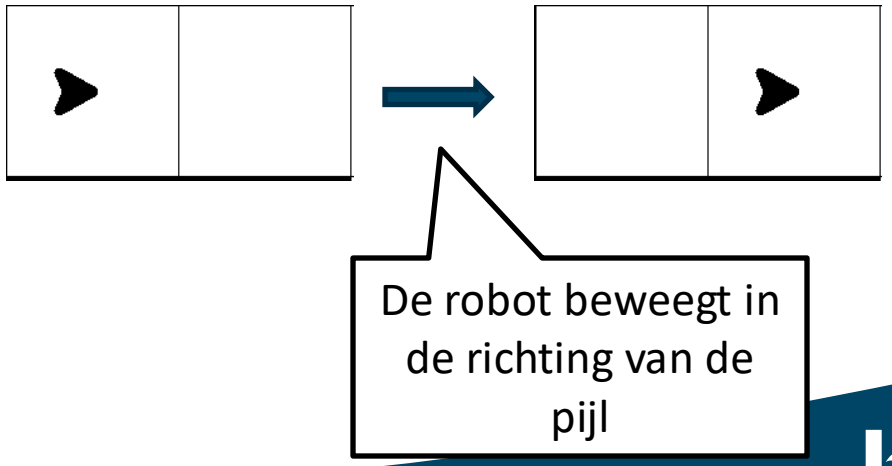
turnLeft()



turnRight()

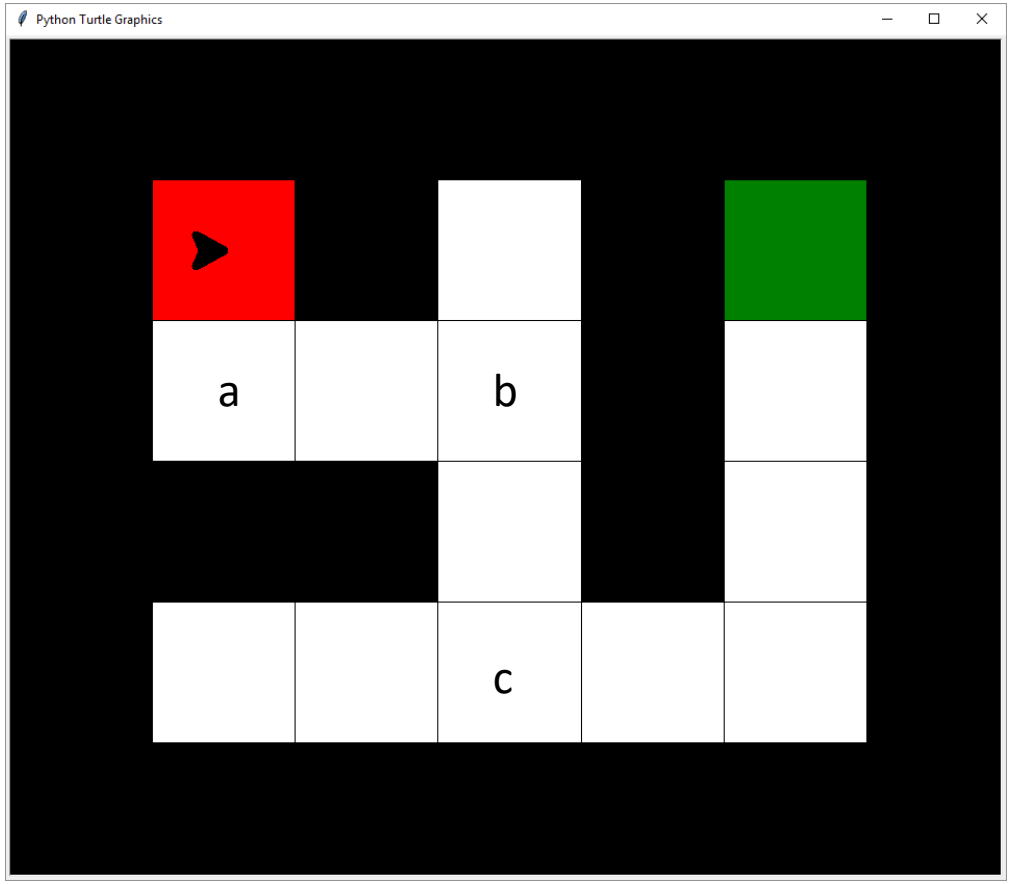


goForward()



Vlugge vraag

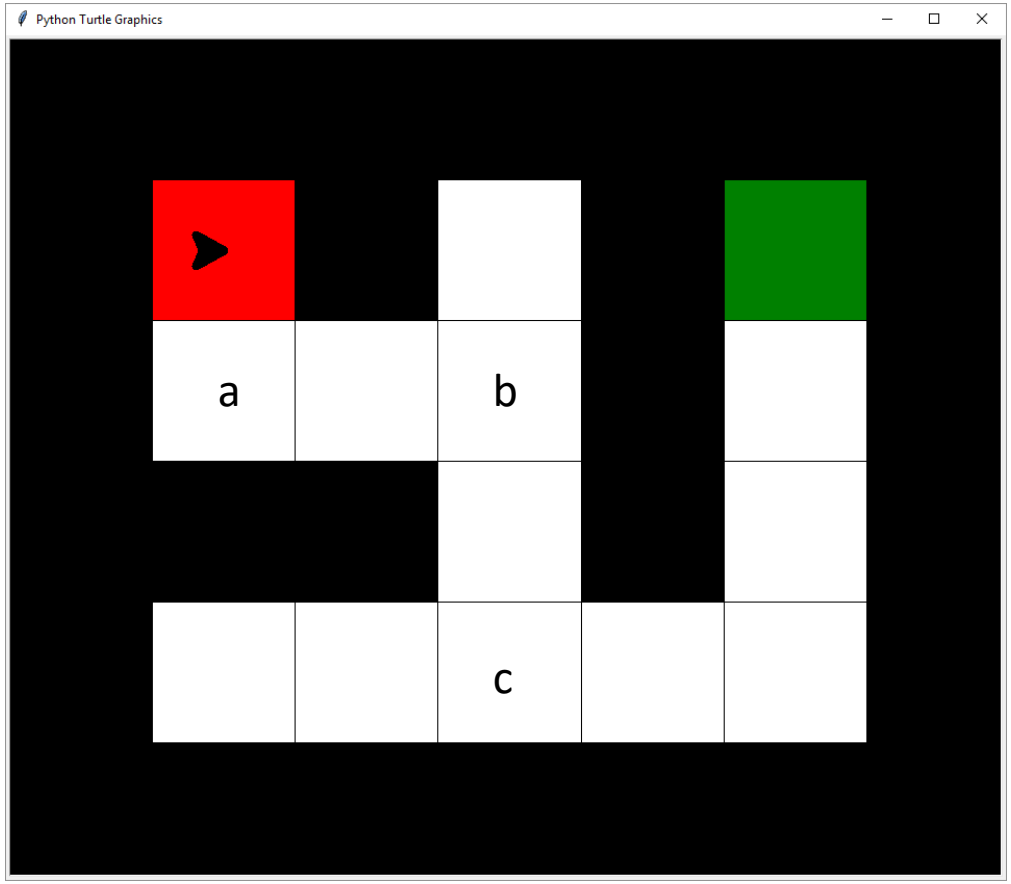
Beschouw volgende beginsituatie; wat is het resultaat van volgende sequentie instructies?



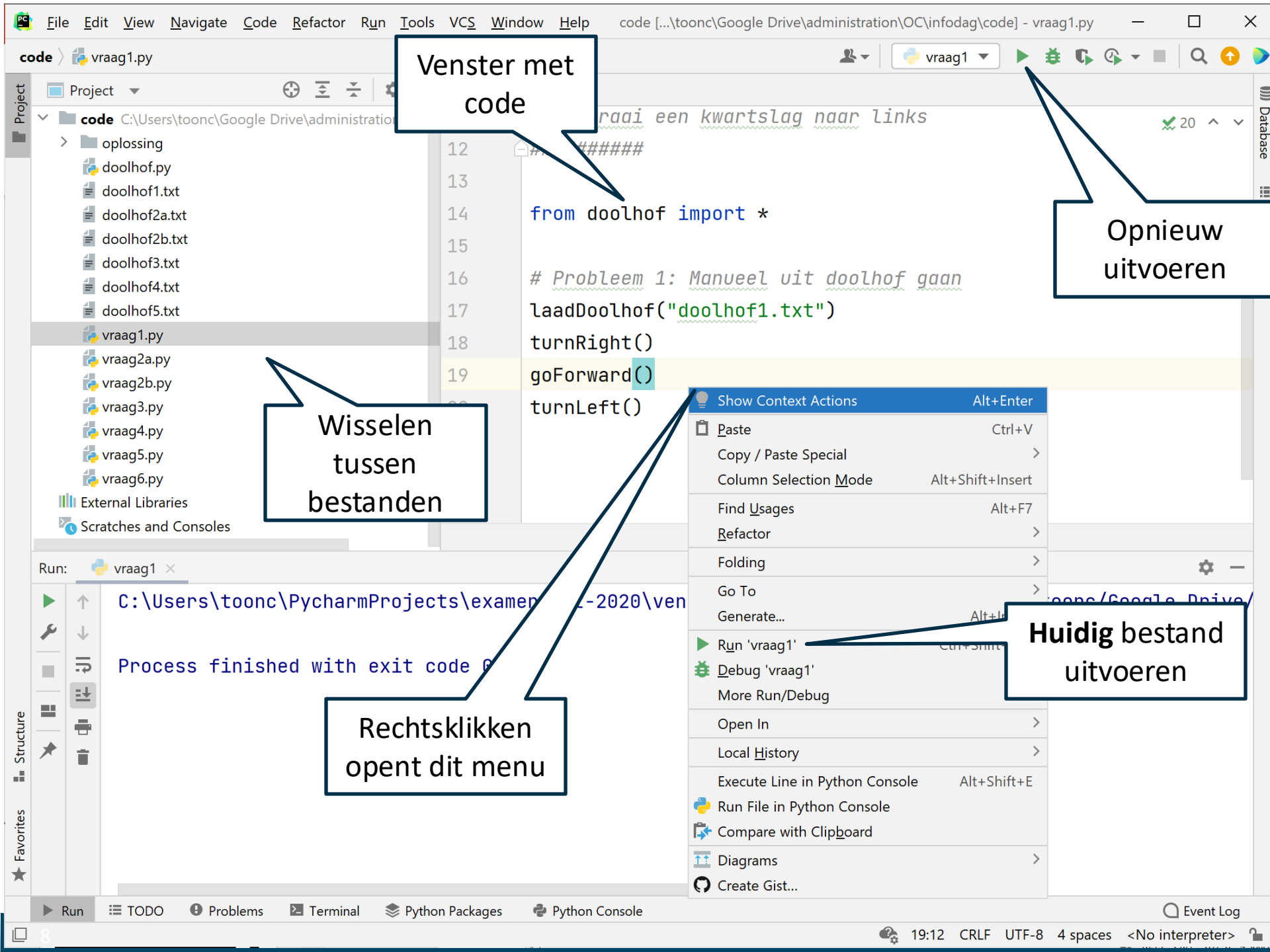

```
turnRight()
goForward()
turnLeft()
goForward()
goForward()
turnRight()
```

Vlugge vraag: Antwoord: **b**

Beschouw volgende beginsituatie; wat is het resultaat van volgende sequentie instructies?

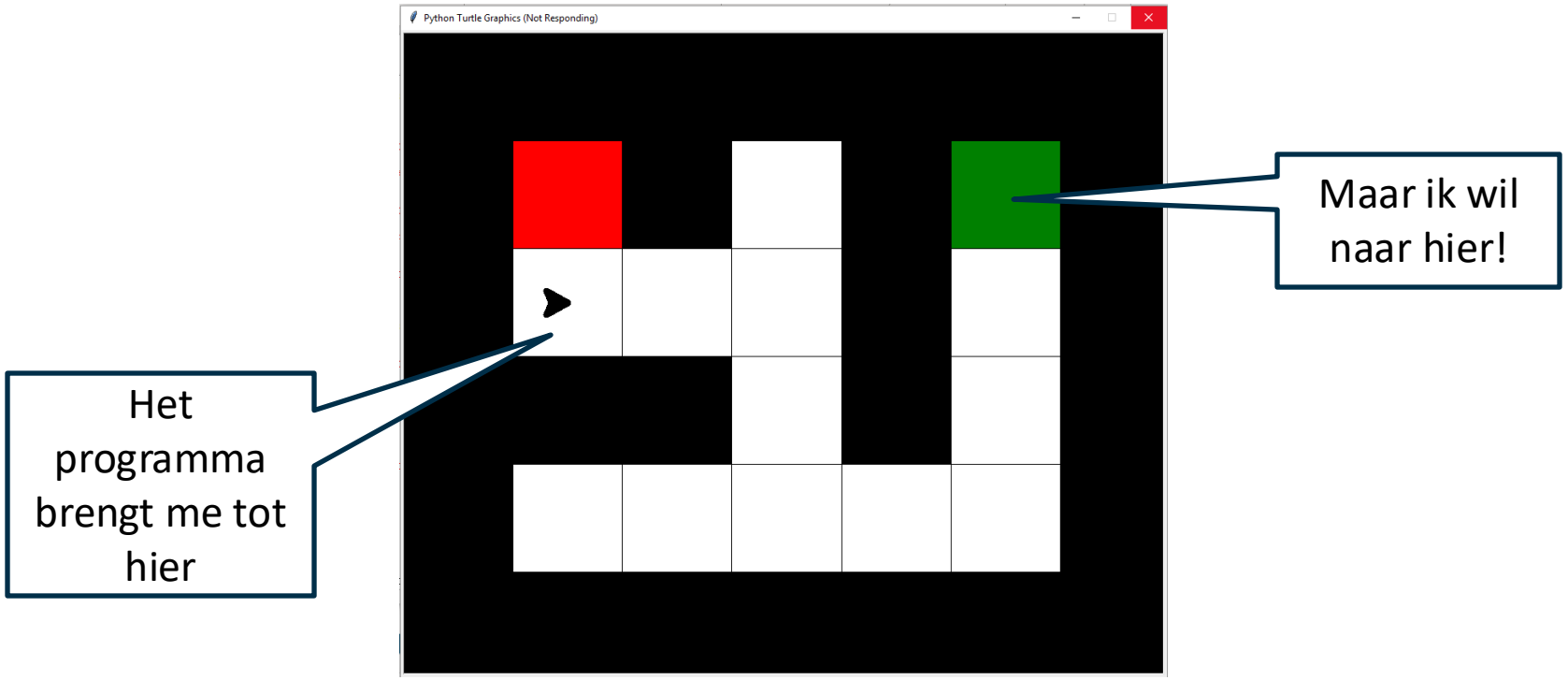



```
turnRight()
goForward()
turnLeft()
goForward()
goForward()
turnRight()
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
code [...\toonc\Google Drive\administration\OC\infodag\code] - vraag1.py
code vraag1.py
vraag1
Project
code C:\Users\toonc\Google Drive\administratio
oplossing
doolhof.py
doolhof1.txt
doolhof2a.txt
doolhof2b.txt
doolhof3.txt
doolhof4.txt
doolhof5.txt
vraag1.py
vraag2a.py
vraag2b.py
vraag3.py
vraag4.py
vraag5.py
vraag6.py
External Libraries
Scratches and Consoles
raai een kwartslag naar links
######
from doolhof import *
# Probleem 1: Manueel uit doolhof gaan
laadDoolhof("doolhof1.txt")
turnRight()
goForward()
turnLeft()
Run: vraag1
C:\Users\toonc\PycharmProjects\examen -2020\ven
Process finished with exit code 0
Show Context Actions Alt+Enter
Paste Ctrl+V
Copy / Paste Special
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Refactor
Folding
Go To
Generate...
Run 'vraag1'
Debug 'vraag1'
More Run/Debug
Open In
Local History
Execute Line in Python Console Alt+Shift+E
Run File in Python Console
Compare with Clipboard
Diagrams
Create Gist...
Run TODO Problems Terminal Python Packages Python Console Event Log
19:12 CRLF UTF-8 4 spaces <No interpreter>
Venster met code
Opnieuw uitvoeren
Wisselen tussen bestanden
Rechtsklikken opent dit menu
Huidig bestand uitvoeren

Opdracht #1 (vraag1.py)

We zijn al een eind op weg; maak het programma verder af!



Oplossing opdracht #1

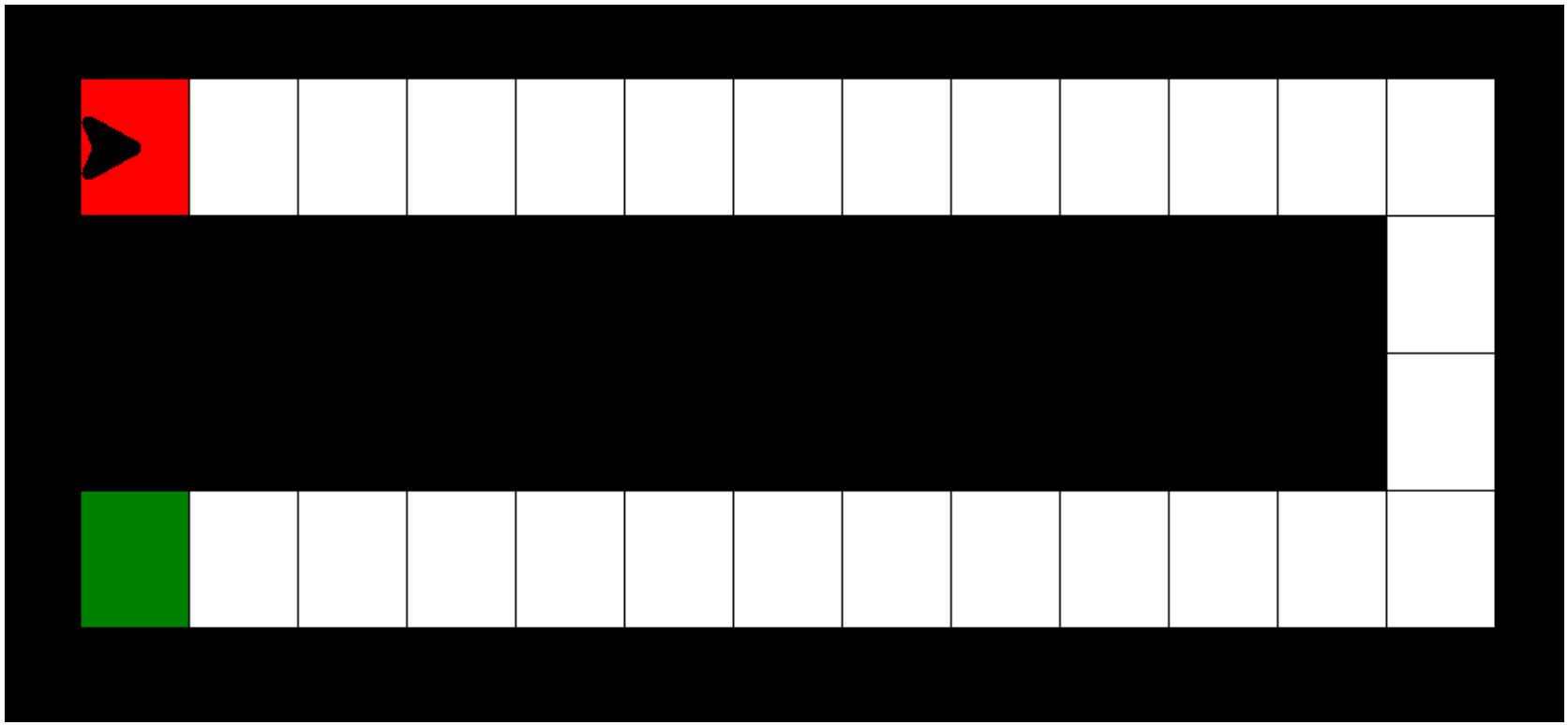
Volgende sequentie van instructies brengt je naar de uitgang:

```
from doolhof import *

# Probleem 1: Manueel uit doolhof gaan
laadDoolhof("doolhof.txt")
turnRight()
goForward()
turnLeft()
goForward()
goForward()
turnRight()
goForward()
goForward()
turnLeft()
goForward()
goForward()
turnLeft()
goForward()
goForward()
goForward()
```

We vallen in herhaling …

Laad nu het volgende doolhof (vraag2a.py)

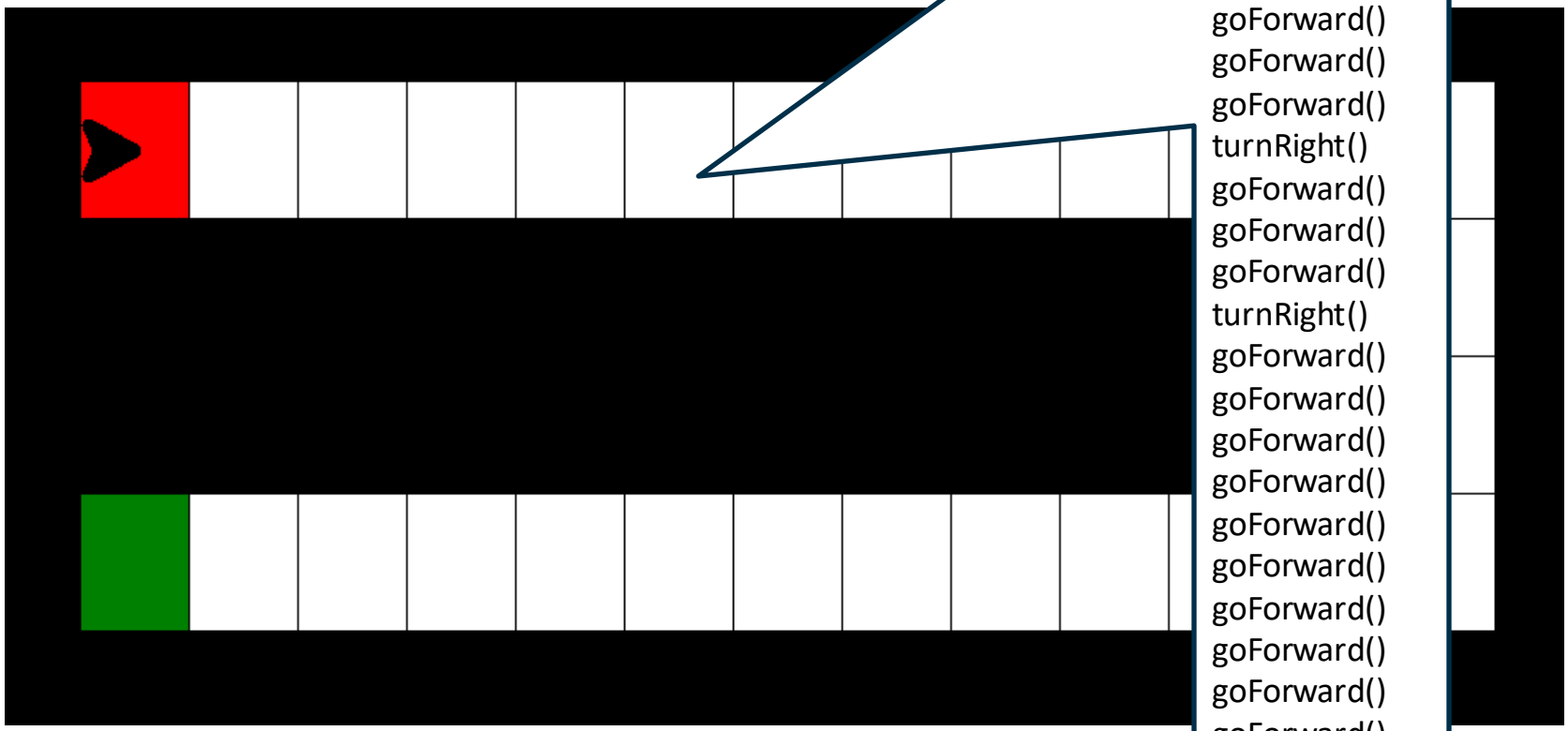
```
# stap 2a: gebruik while
laadDoolhof("doolhof2a.txt")
```



We vallen in herhaling …

Laad nu het volgende doolhof (vraag2a.py)

```
# stap 2a: gebruik while
laadDoolhof("doolhof2a.txt")
```

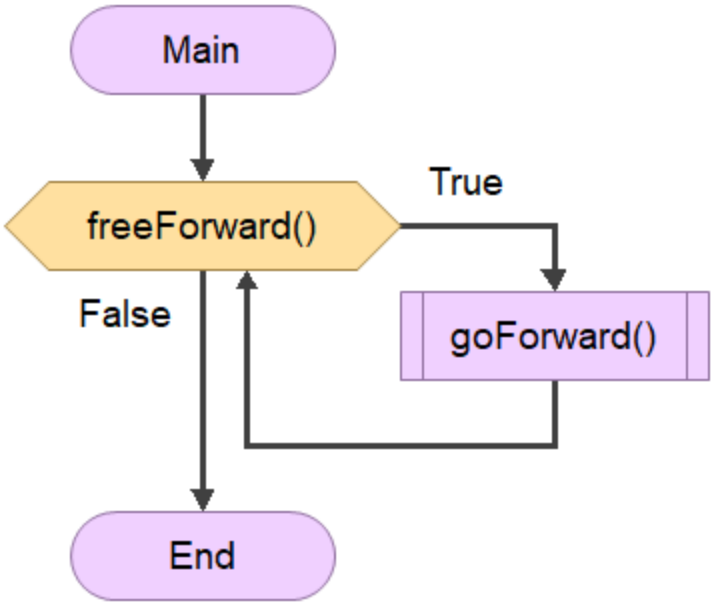


```
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
turnRight()
goForward()
goForward()
goForward()
turnRight()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
```

Herhalingslus

"Zolang je rechtdoor kan gaan, ga rechtdoor"

```
while freeForward():
    goForward()
```



Je kan niet enkel 1 instructie herhalen (goForward()), maar ook een heel programma

het te herhalen stuk springt in

Nieuwe functies

	Waar (True) als
freeForward()	… de plek voor je
freeLeft()	… links van je
freeRight()	… rechts van je
	vrij is; anders onwaar (False)
foundExit()	Waar als je het eindpunt bereikte
	anders onwaar
not *functie()*	Waar als *functie()* onwaar is en omgekeerd

Herhalingslus

```
goForward()
g
g
g
goForward()
goForward()
goForward()
goForward()
goForward()
goForward()
...
```

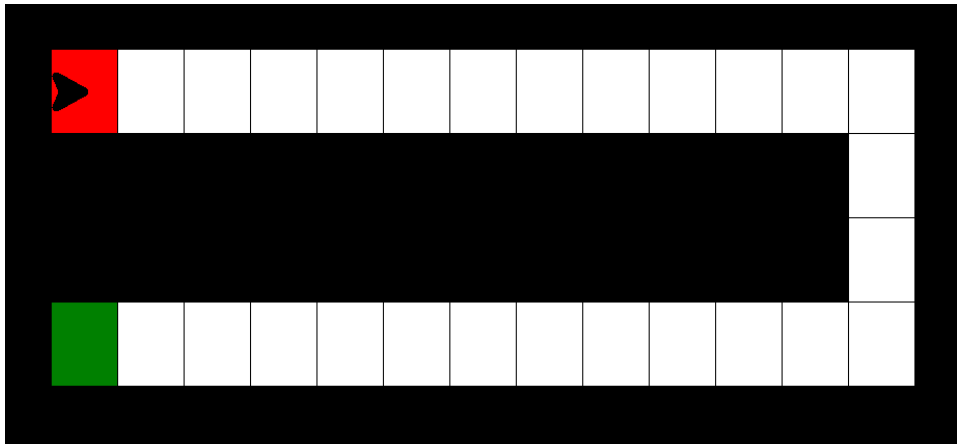
```
while freeForward():
    goForward()
turnRight()
...
```

Het te herhalen stuk springt in; gebruik <TAB>

Geen hoofdletter

Denk aan ":"

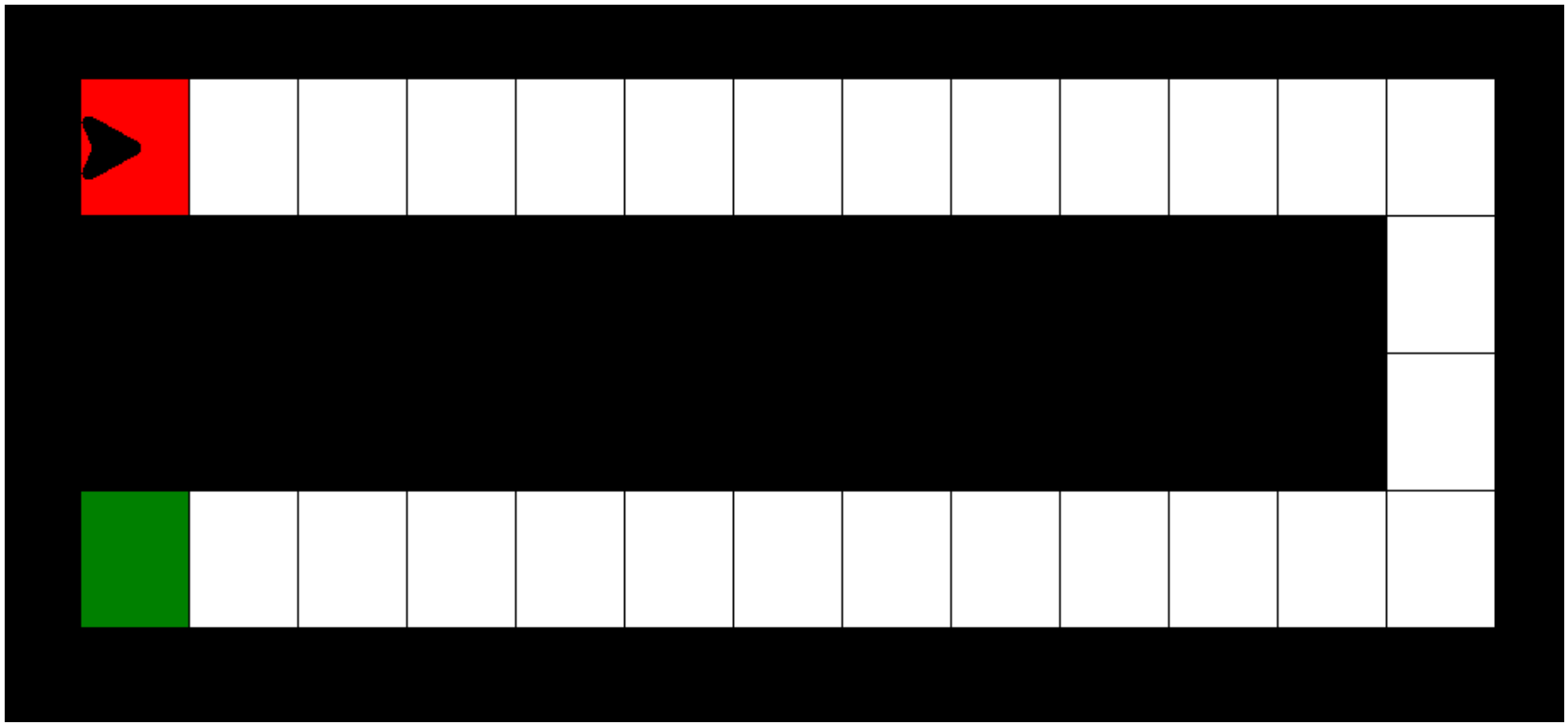
Te herhalen stuk gedaan? Terug naar links! <Backspace>



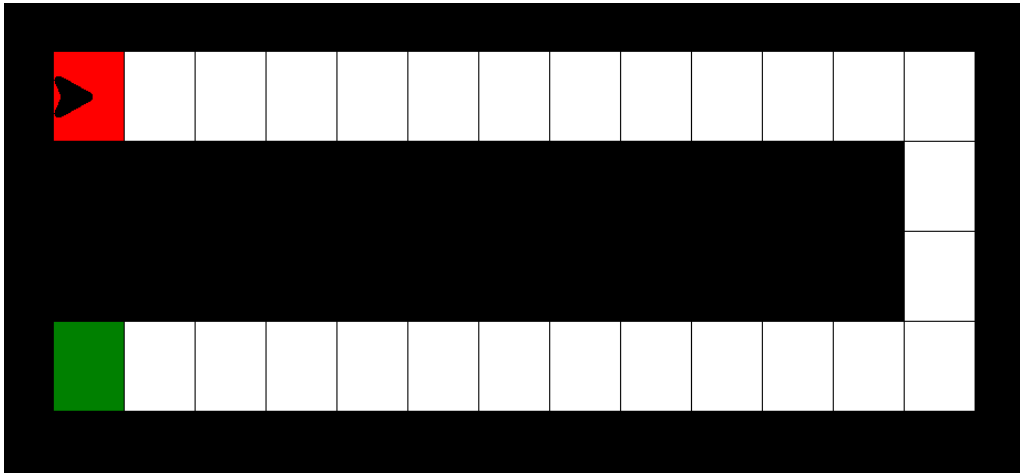
Opdracht #2a (vraag2a.py)

Laad nu het doolhof en ga naar de uitgang

```
# stap 2a: gebruik while
laadDoolhof("doolhof2a.txt")
```

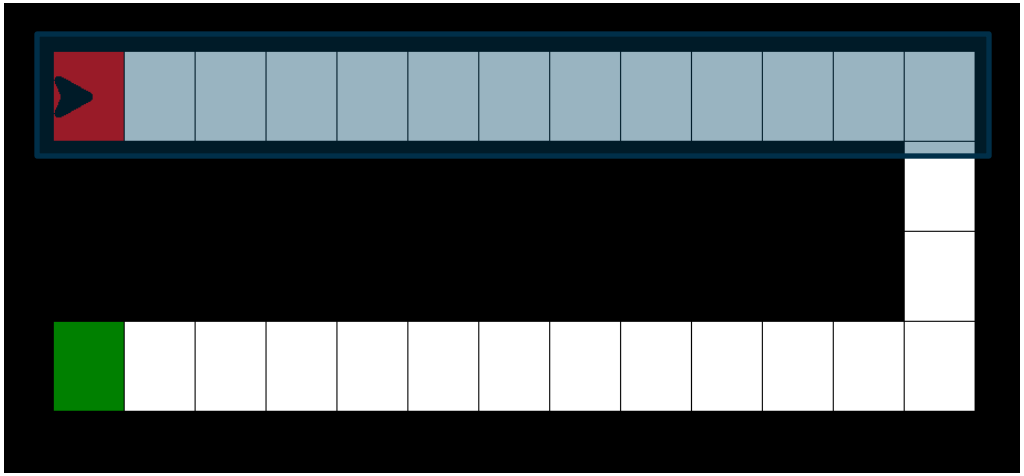


Oplossing #2a



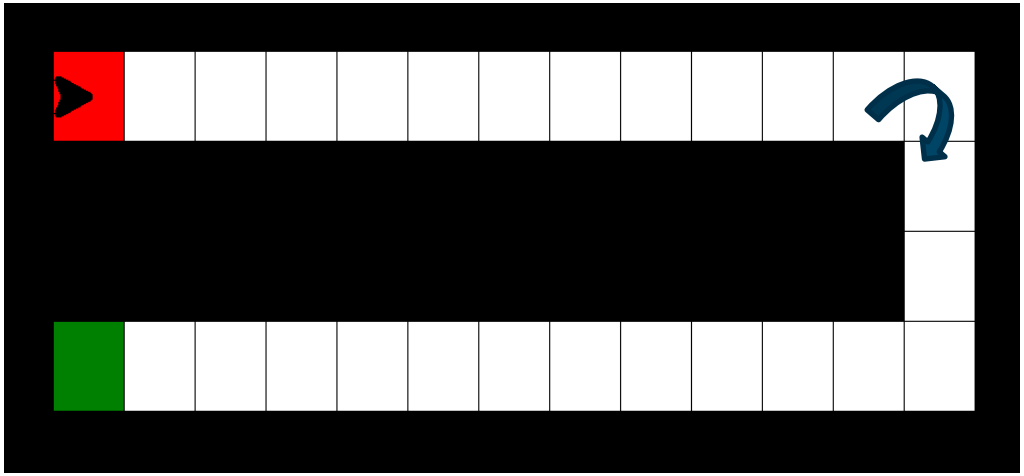
```
# stap 2a: gebruik while
laadDoolhof("doolhof2a.txt")

while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
```

Oplossing #2a



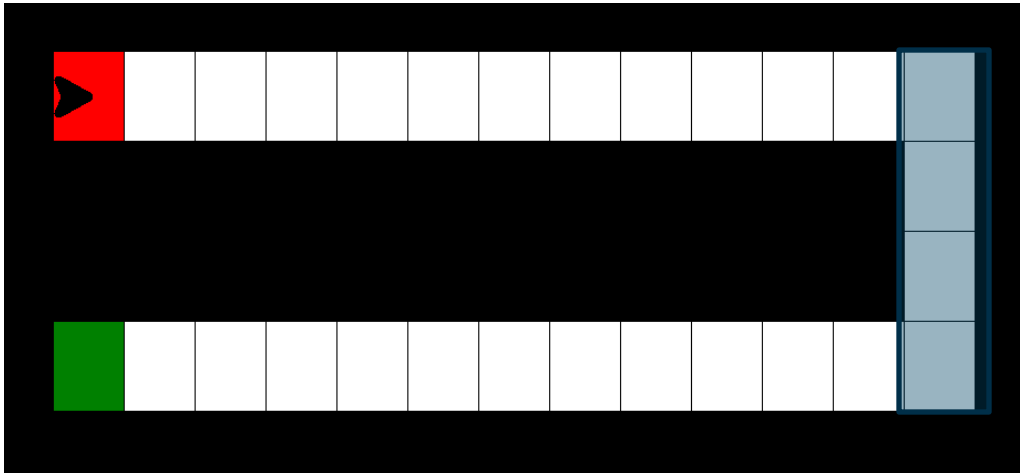
```
# stap 2a: gebruik while
laadDoolhof("doolhof2a.txt")

while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
```

Oplossing #2a



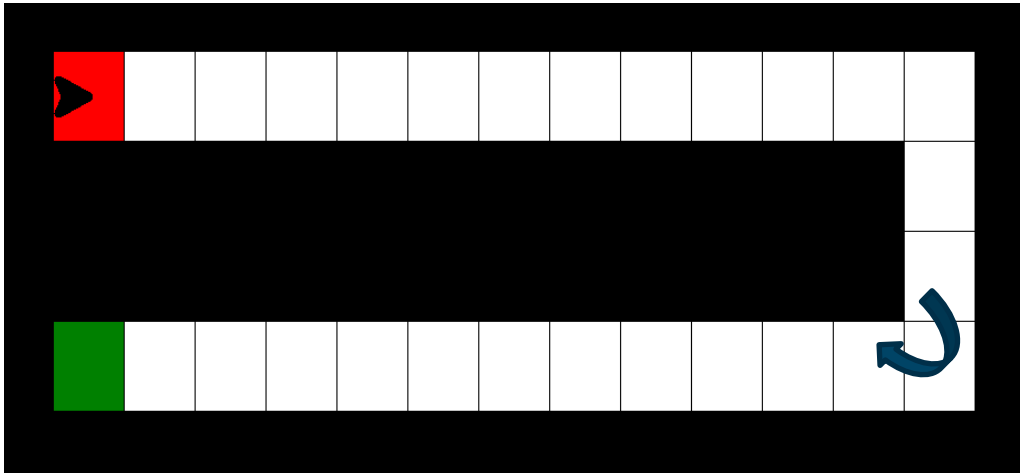
```
# stap 2a: gebruik while
laadDoolhof("doolhof2a.txt")

while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
```

Oplossing #2a



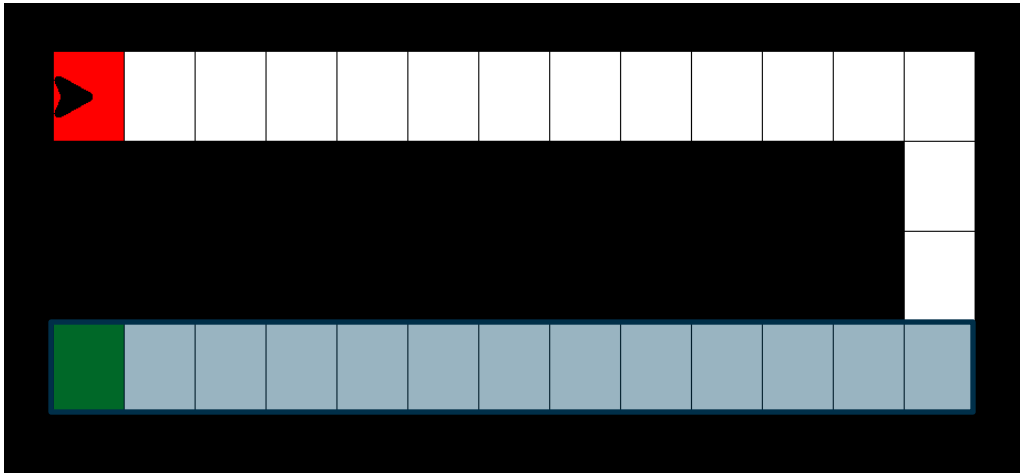
```
# stap 2a: gebruik while
laadDoolhof("doolhof2a.txt")

while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
```

Oplossing #2a



```
# stap 2a: gebruik while
laadDoolhof("doolhof2a.txt")

while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
```

Oplossing #2a



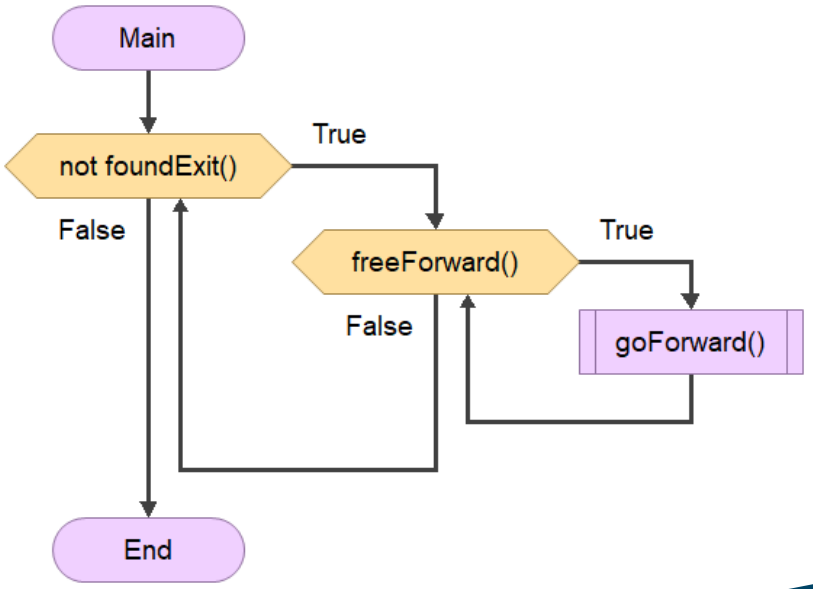
```
# stap 2a: gebruik while
laadDoolhof("doolhof2a.txt")

while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
```

Oplossing #2a : minder code!

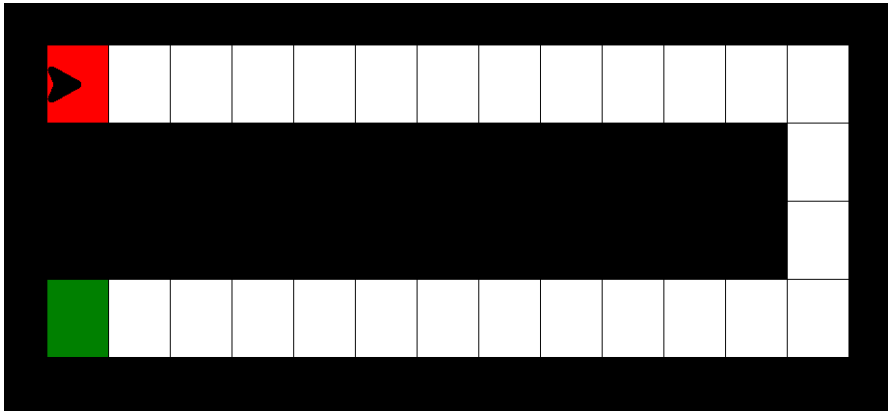
```
# stap 2a: gebruik while
laadDoolhof("doolhof2a.txt")

while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
turnRight()
while freeForward():
    goForward()
```

+ turnRight()

```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

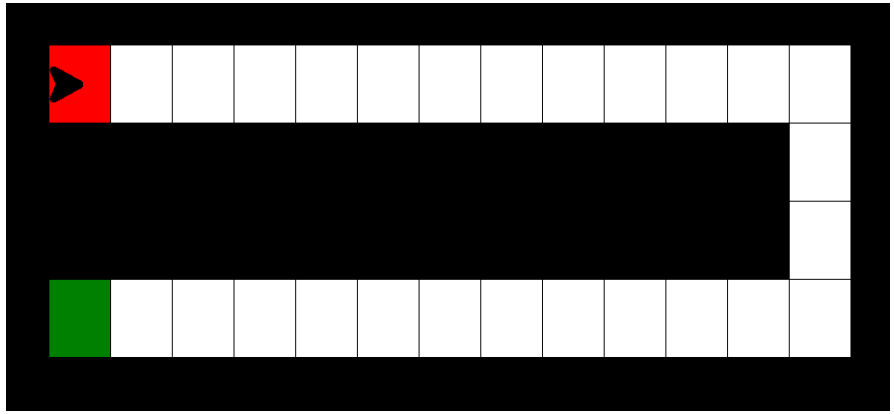


Oplossing #2a - korter



```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

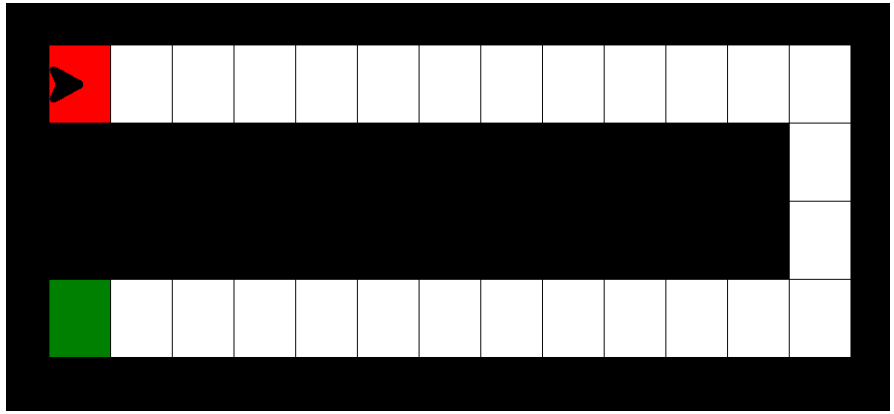
Oplossing #2a - korter



?

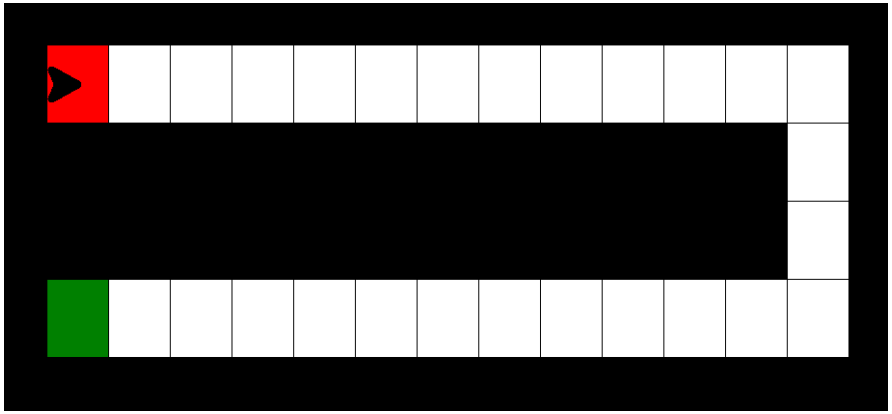
```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

Oplossing #2a - korter



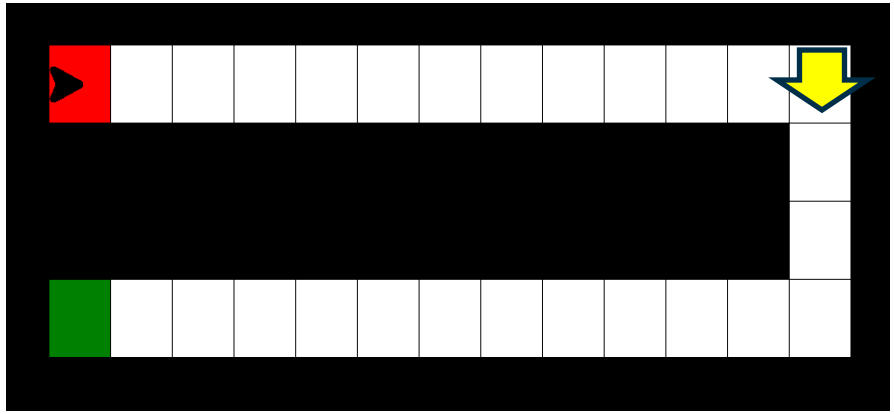
```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

Oplossing #2a - korter



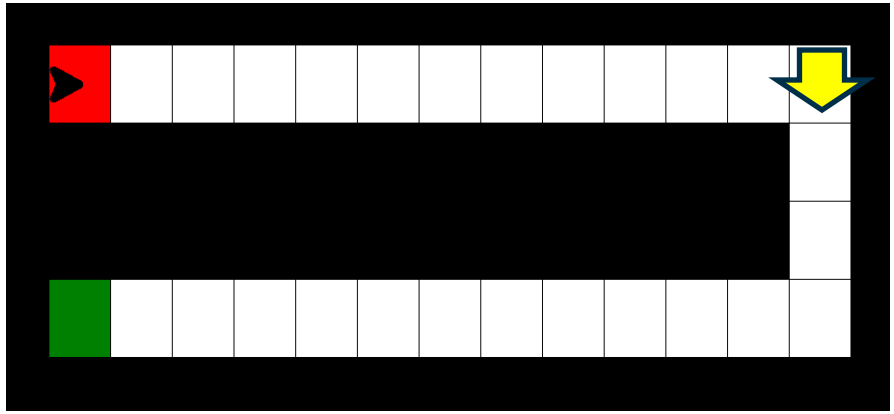
```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

Oplossing #2a
- korter



```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

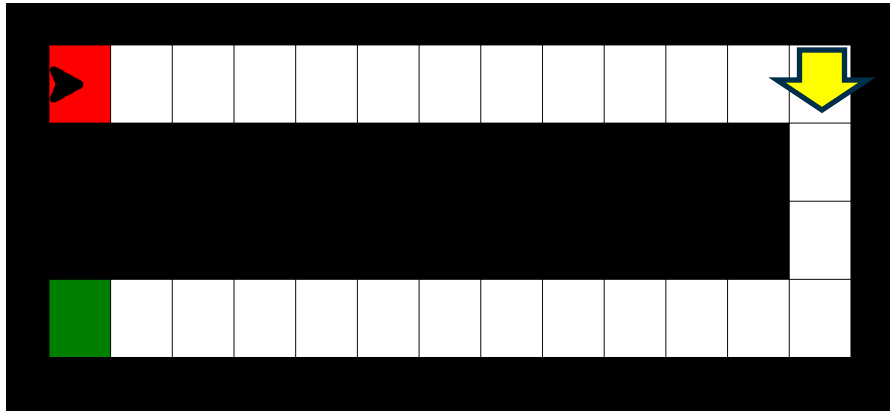
Oplossing #2a - korter



?

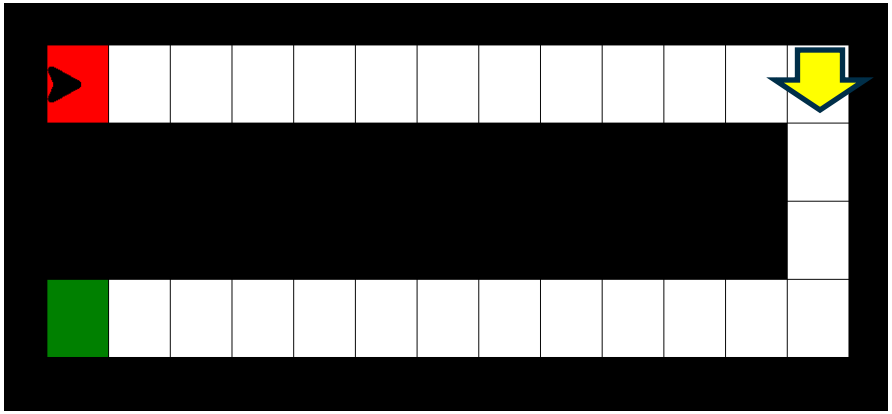
```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

Oplossing #2a
- korter



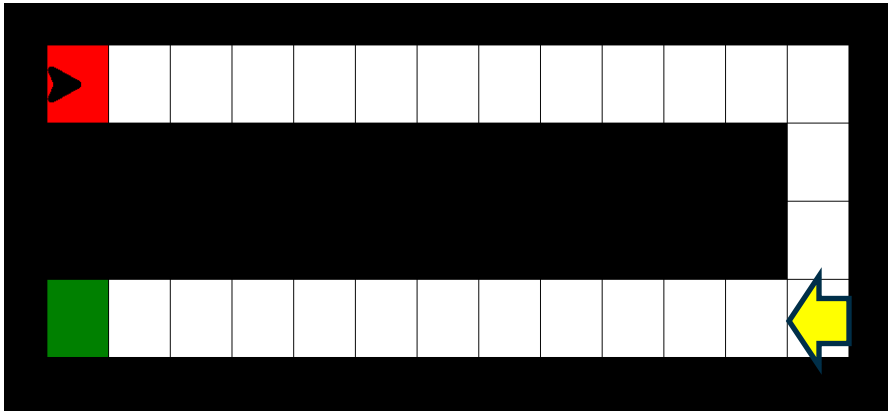
```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

Oplossing #2a - korter



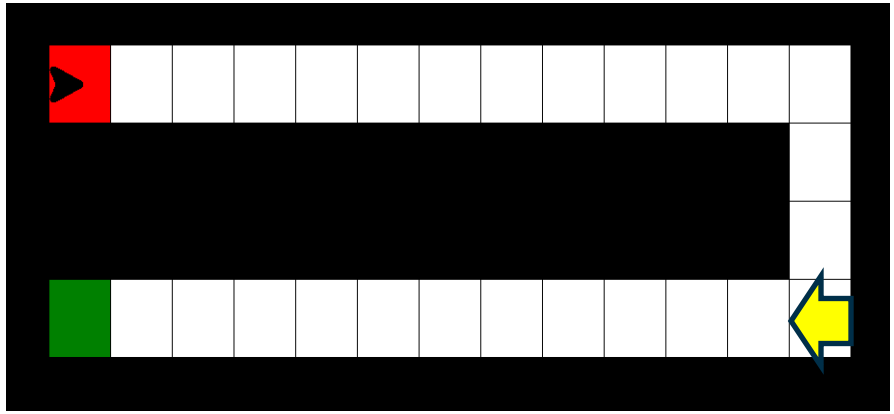
```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

Oplossing #2a
- korter



```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

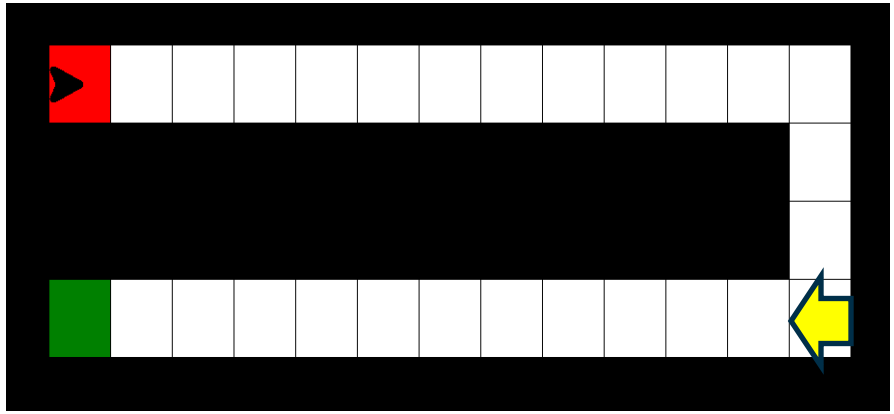
Oplossing #2a - korter



?

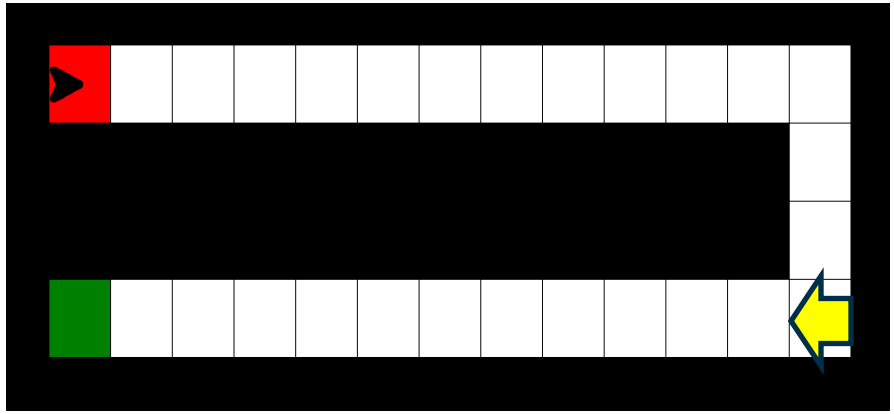
```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

Oplossing #2a - korter



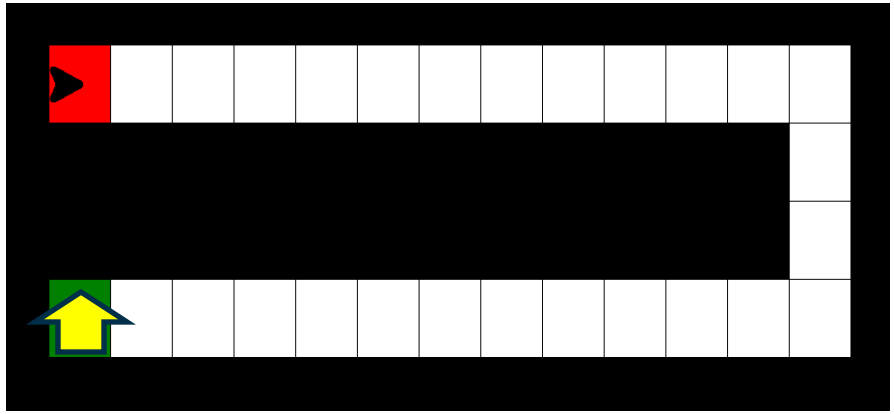
```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

Oplossing #2a
- korter



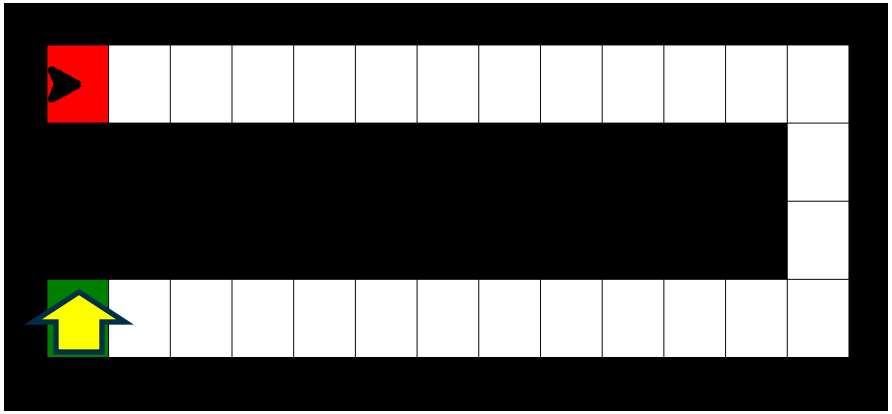
```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

Oplossing #2a
- korter



```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

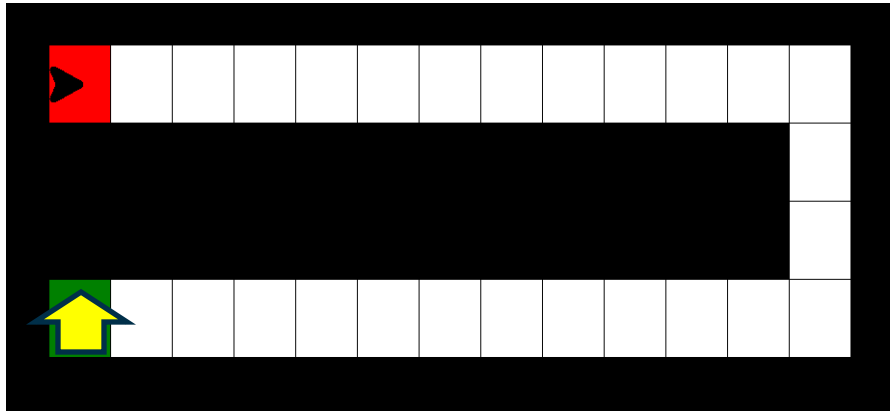
Oplossing #2a - korter



?

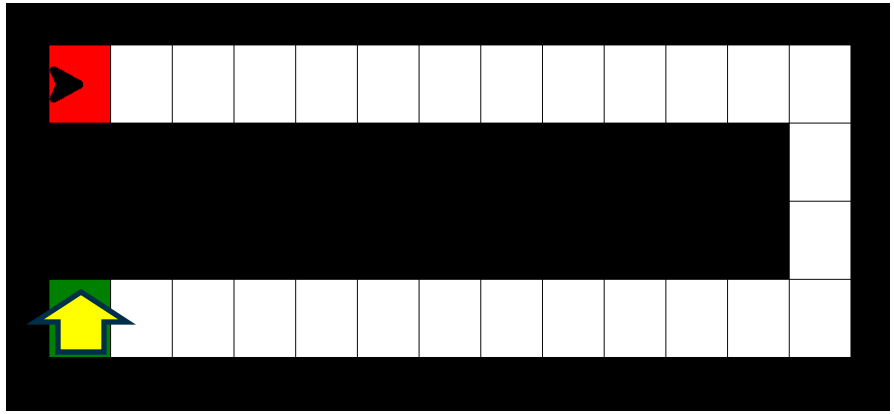
```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

Oplossing #2a
- korter



```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

Oplossing #2a - korter

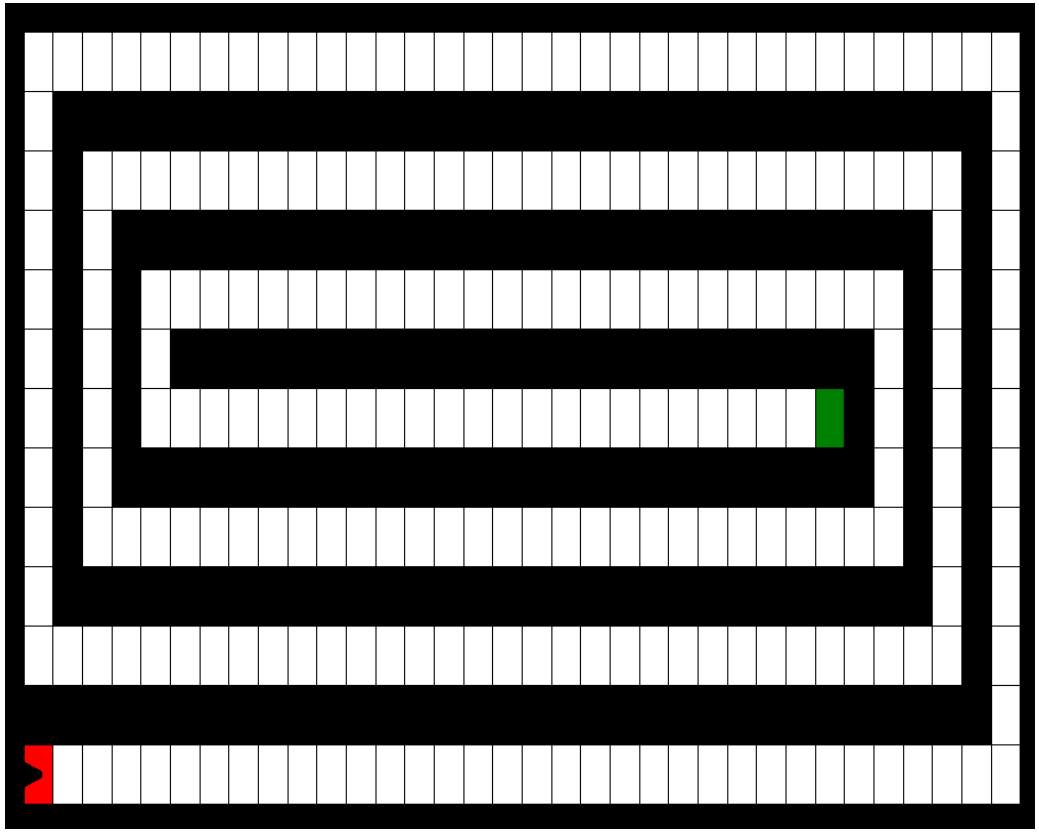


```
while not foundExit():
    while freeForward():
        goForward()
    turnRight()
```

Opdracht #2b (vraag2b.py)

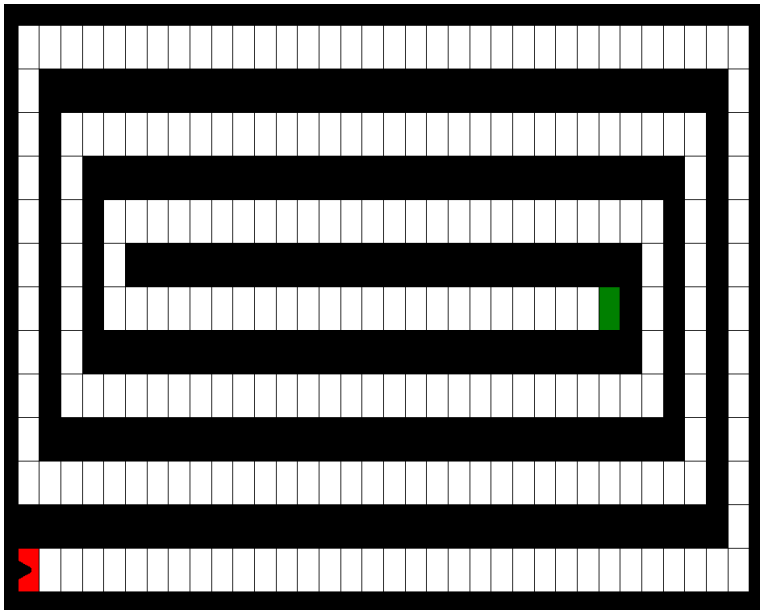
Ga naar het einde van dit doolhof:

```
# stap 2b: while opdracht b
laadDoolhof("doolhof2b.txt")
```



Oplossing #2b

```
# stap 2b: while opdracht b
laadDoolhof("doolhof2b.txt")

while not foundExit():
    while freeForward():
        goForward()
    turnLeft()
```



Willekeurige doolhoven oplossen

Tot nu: programma's om specifieke doolhoven op te lossen.

Kunnen we ook programma's schrijven die elk doolhof oplossen?

- We moeten keuzes kunnen maken die afhangen van de situatie:

  Als links vrij is dan draai naar links

  Anders als rechts vrij is dan draai naar rechts

  Anders ga vooruit

if – elif - else

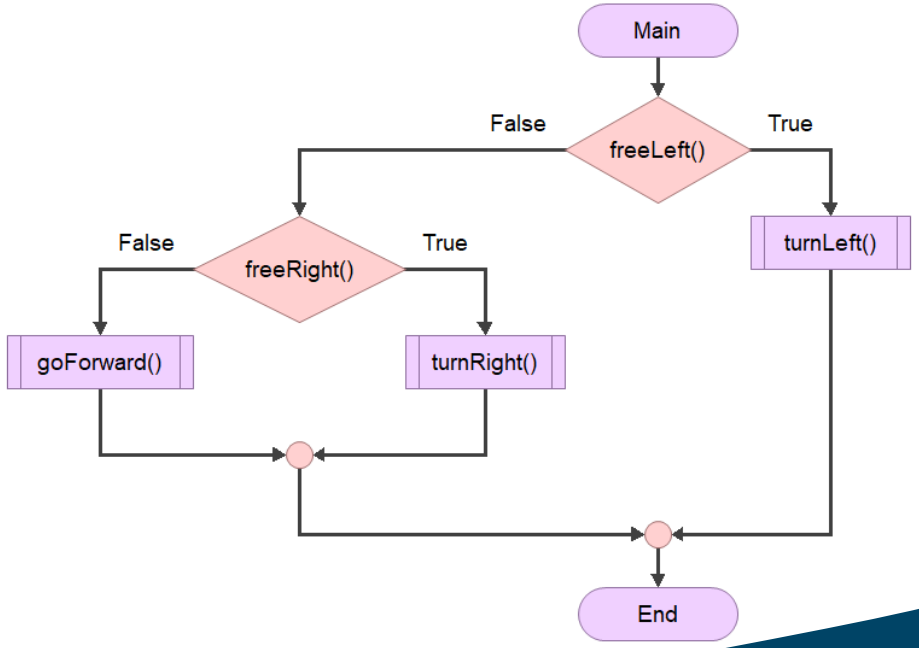
Dit is exact wat if – elif – else doet in Python:

Als links vrij is dan draai naar links

Anders als rechts vrij is dan draai naar rechts

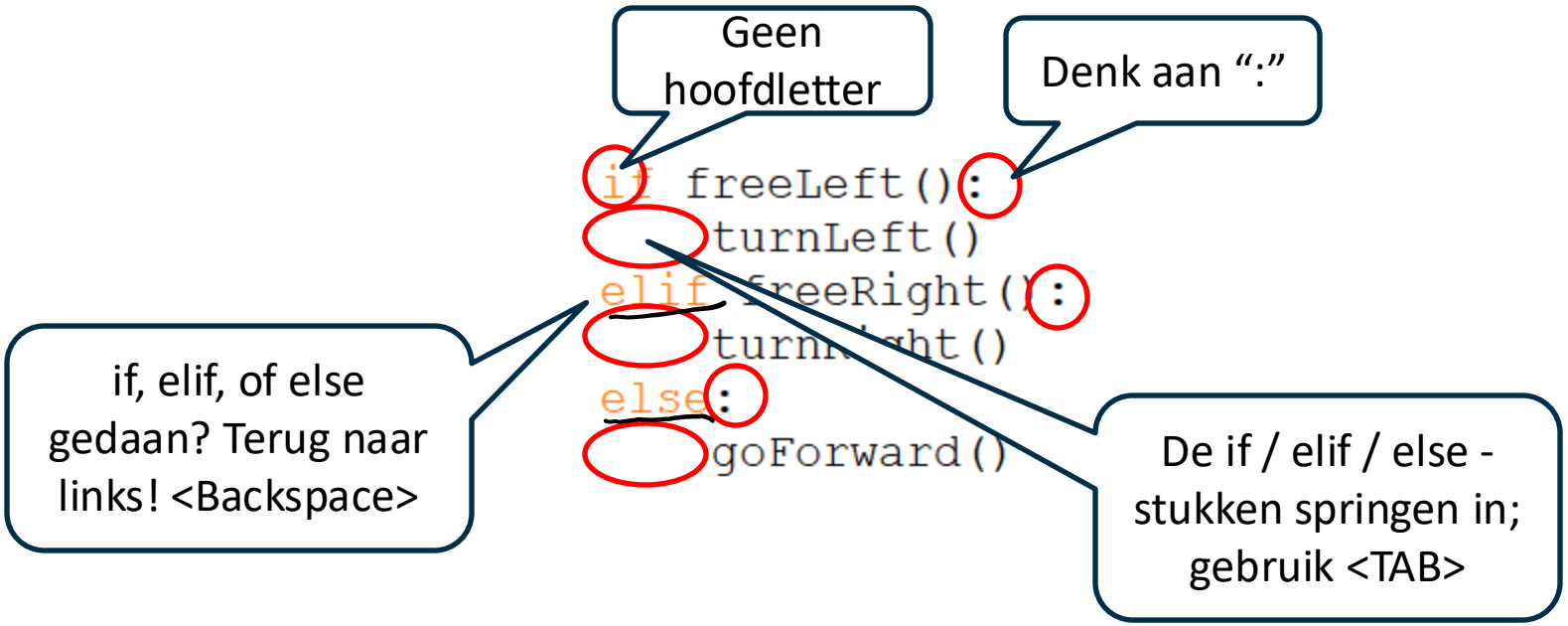
Anders ga vooruit

```
if freeLeft():
    turnLeft()
elif freeRight():
    turnRight()
else:
    goForward()
```



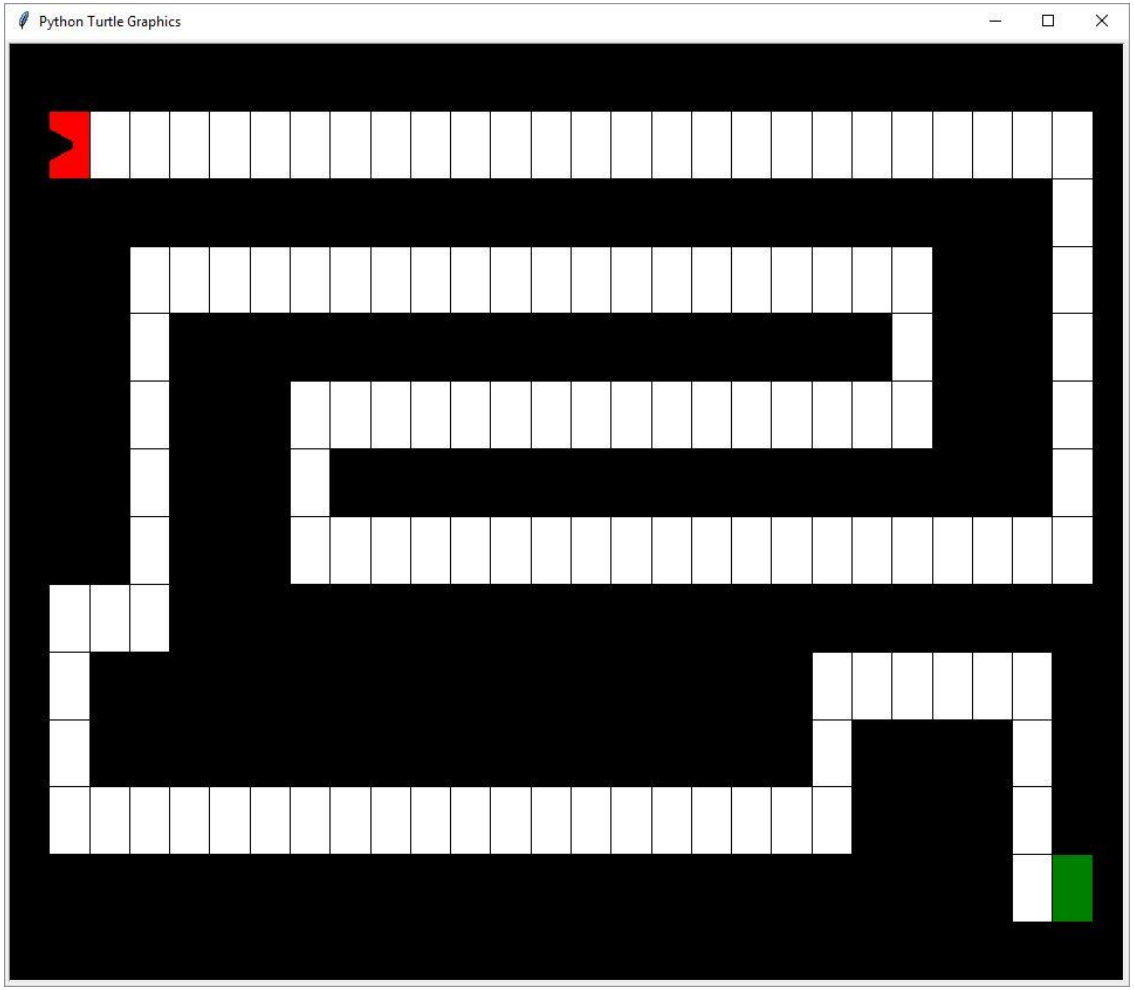
if – elif - else

Dit is exact wat if – elif – else doet in Python:



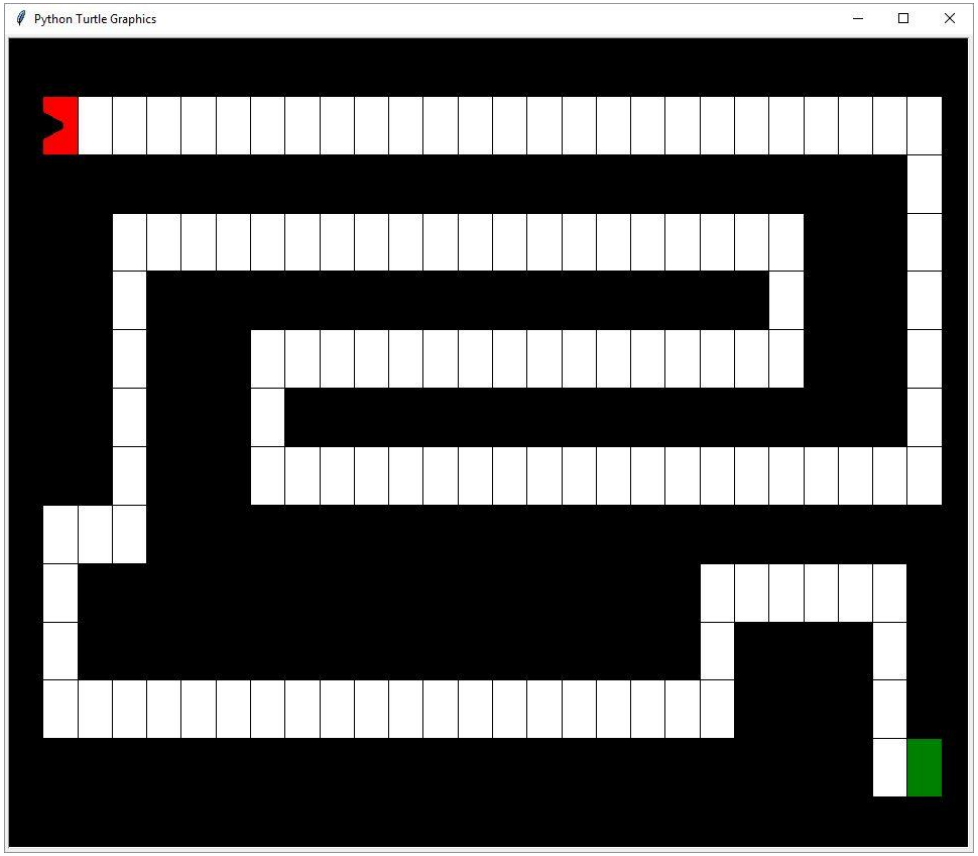
Opdracht #3 vraag3.py

Maak een programma dat doolhof 3 oplost.



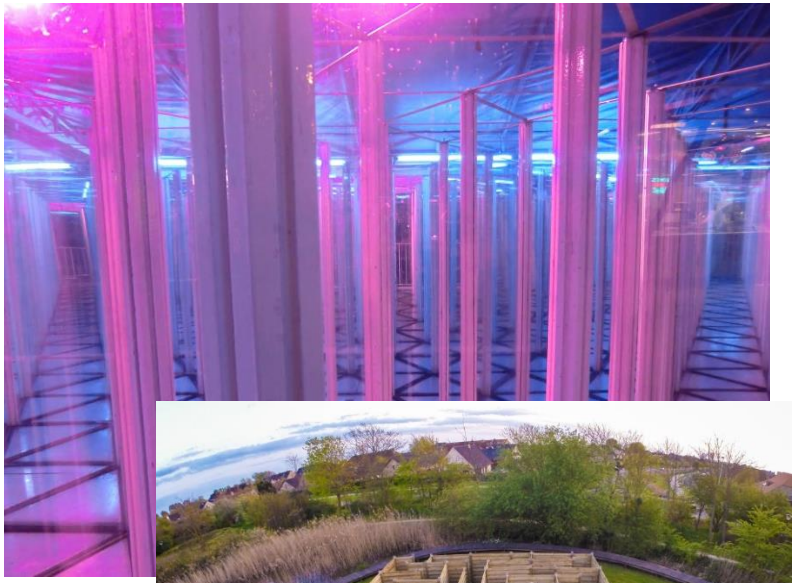
Oplossing #3

```
while not foundExit():
    while freeForward():
        goForward()
    if freeLeft():
        turnLeft()
    else:
        turnRight()
```



De rechterhandregel

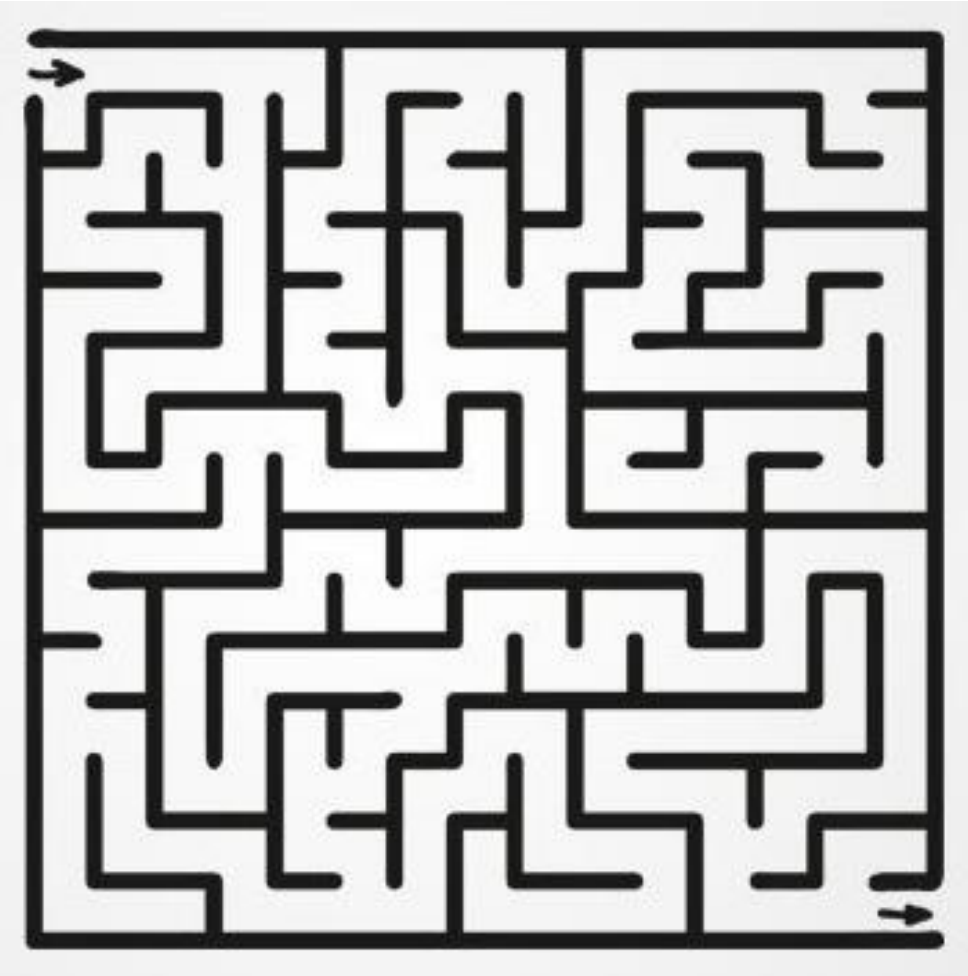
Als in- en uitgang beiden aan de buitenmuur liggen, dan kan je steeds op volgende manier ontsnappen:



Volg steeds het meest rechtse pad

(zorg ervoor dat jouw rechterhand steeds de muur blijft raken)

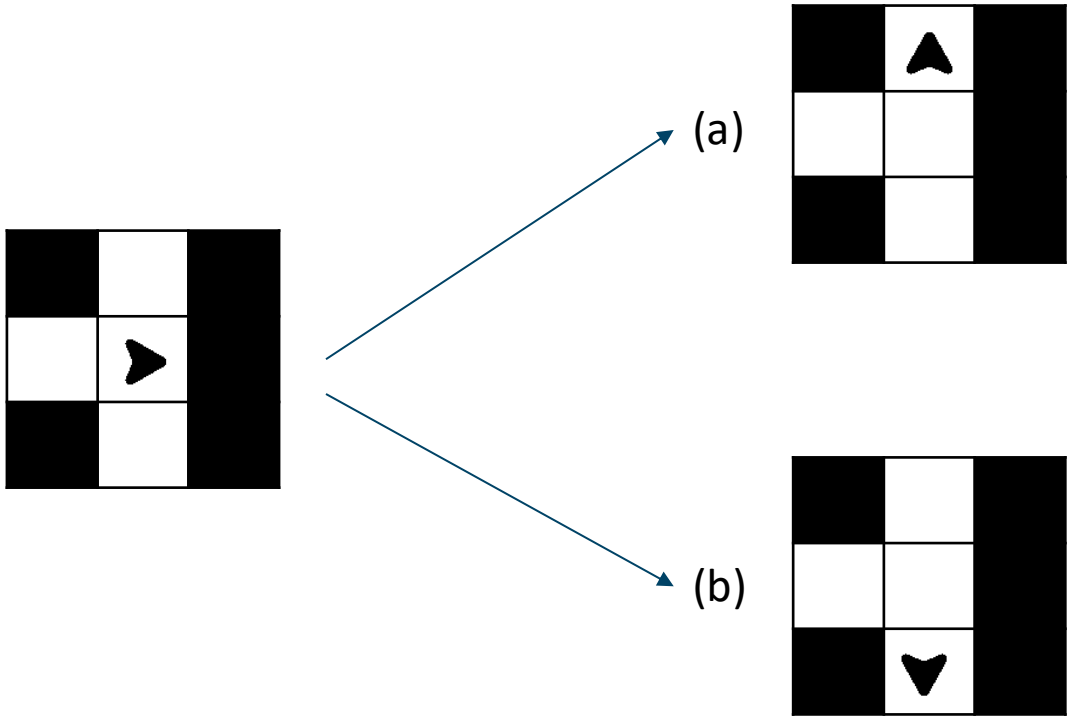
Rechterhandregel: voorbeeld



De rechterhandregel

Volg steeds het meest rechtse pad

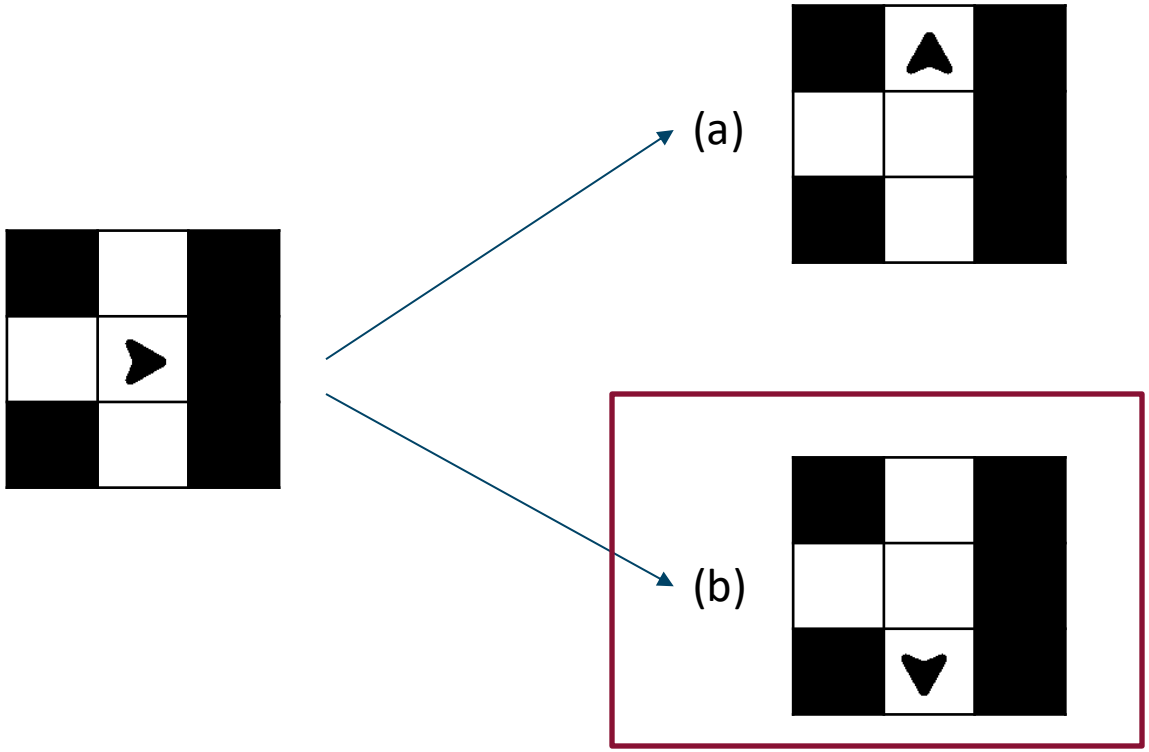
Waarheen in volgend voorbeeld?



De rechterhandregel

Volg steeds het meest rechtse pad

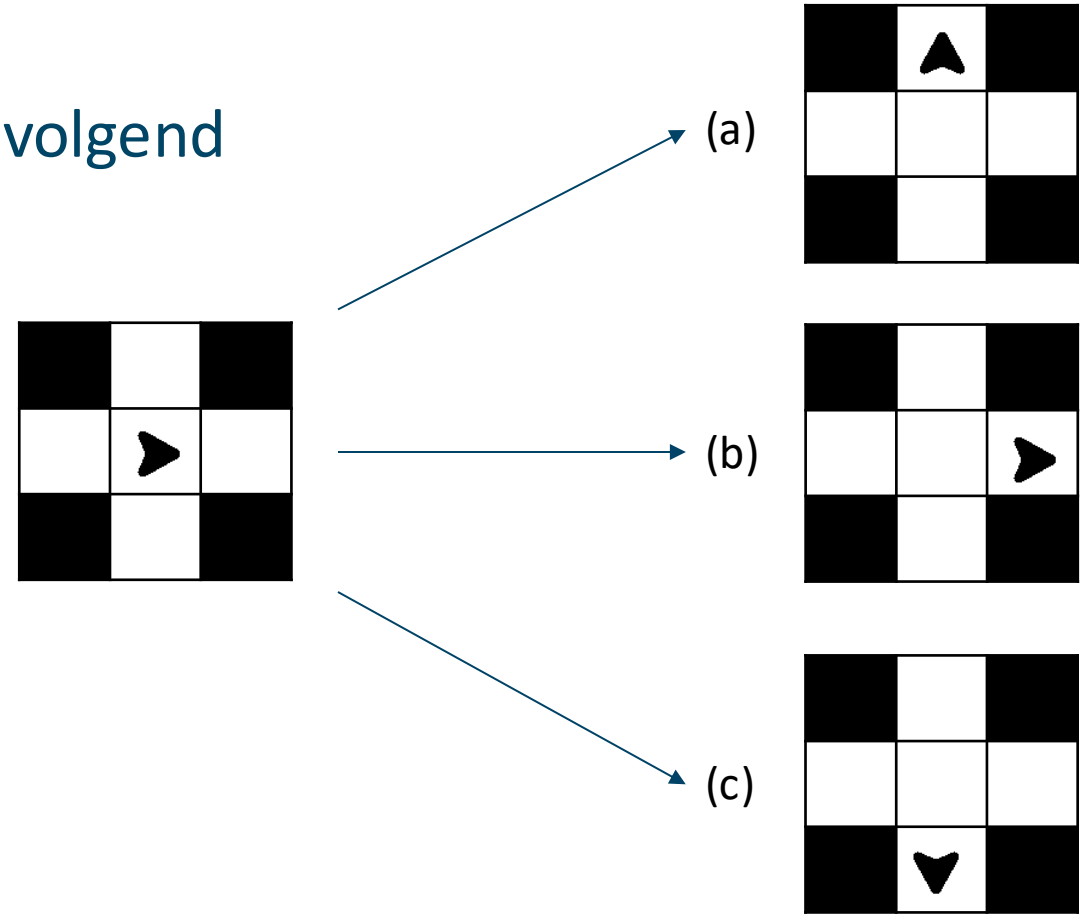
Waarheen in volgend voorbeeld?



De rechterhandregel

Volg steeds het meest rechtse pad

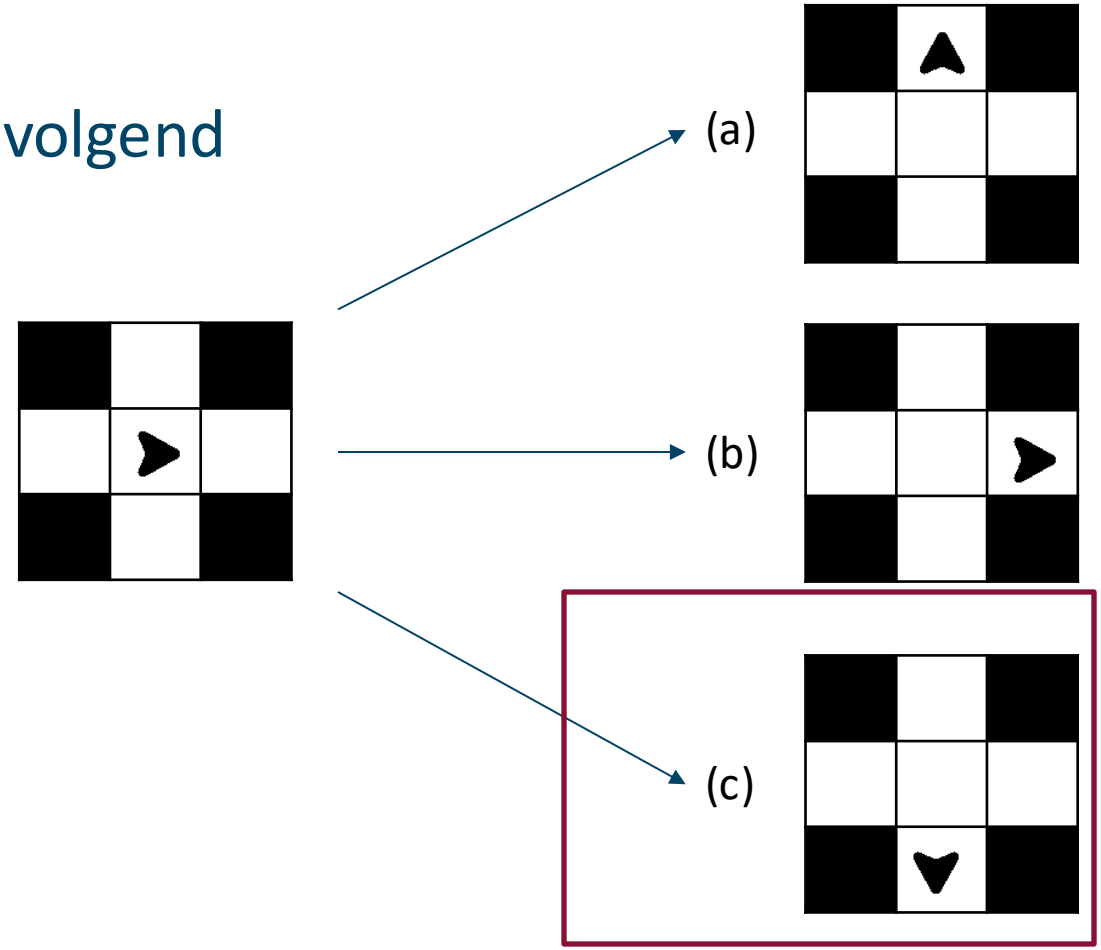
Waarheen in volgend voorbeeld?



De rechterhandregel

Volg steeds het meest rechtse pad

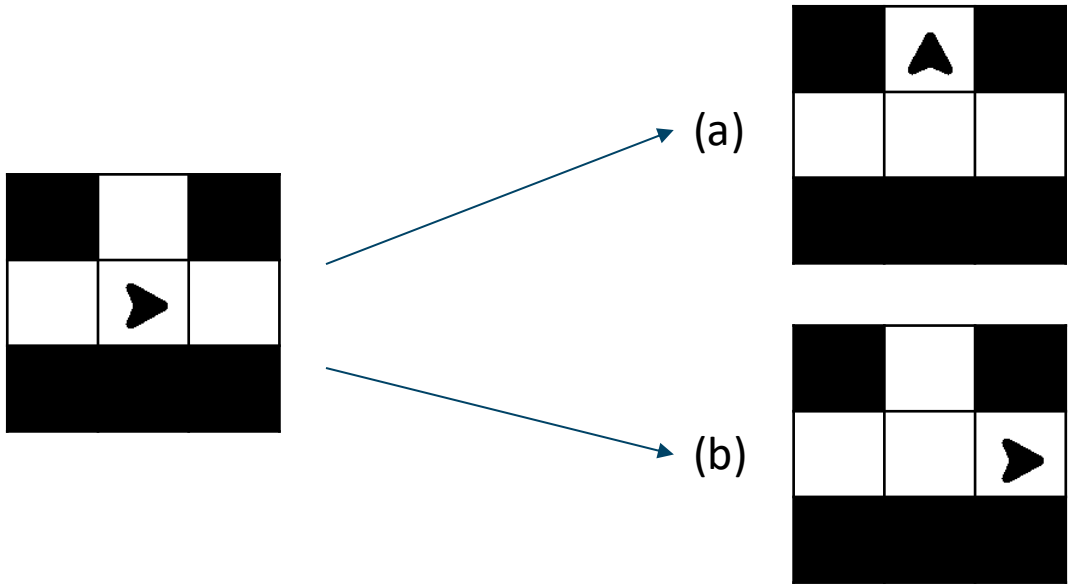
Waarheen in volgend voorbeeld?



De rechterhandregel

Volg steeds het meest rechtse pad

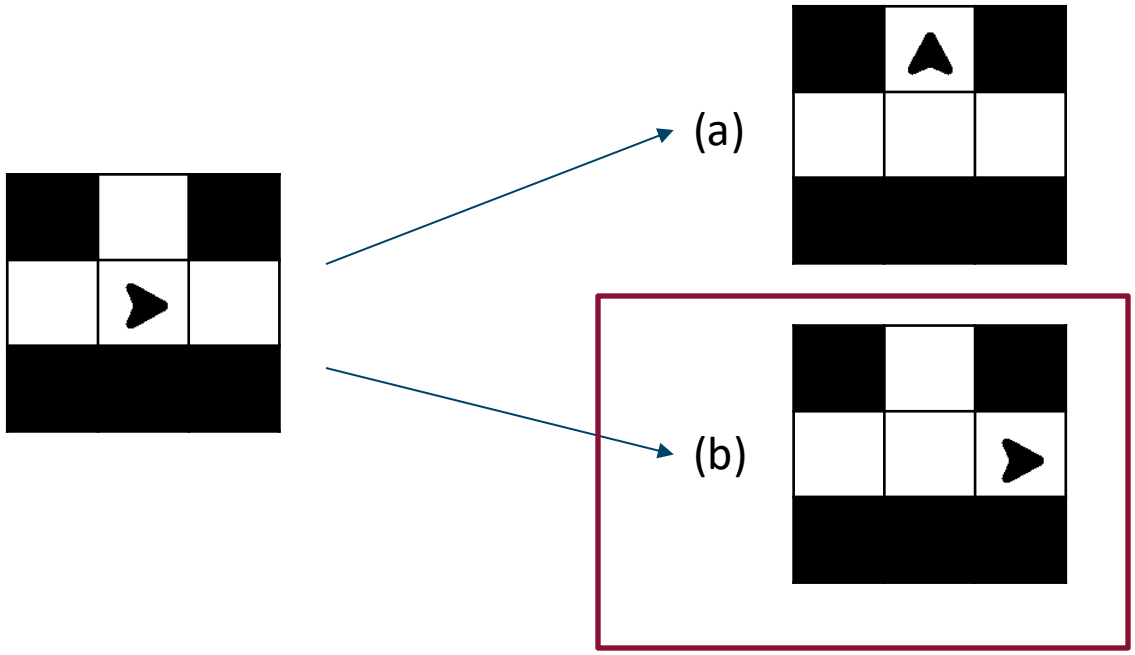
Waarheen in volgend voorbeeld?



De rechterhandregel

Volg steeds het meest rechtse pad

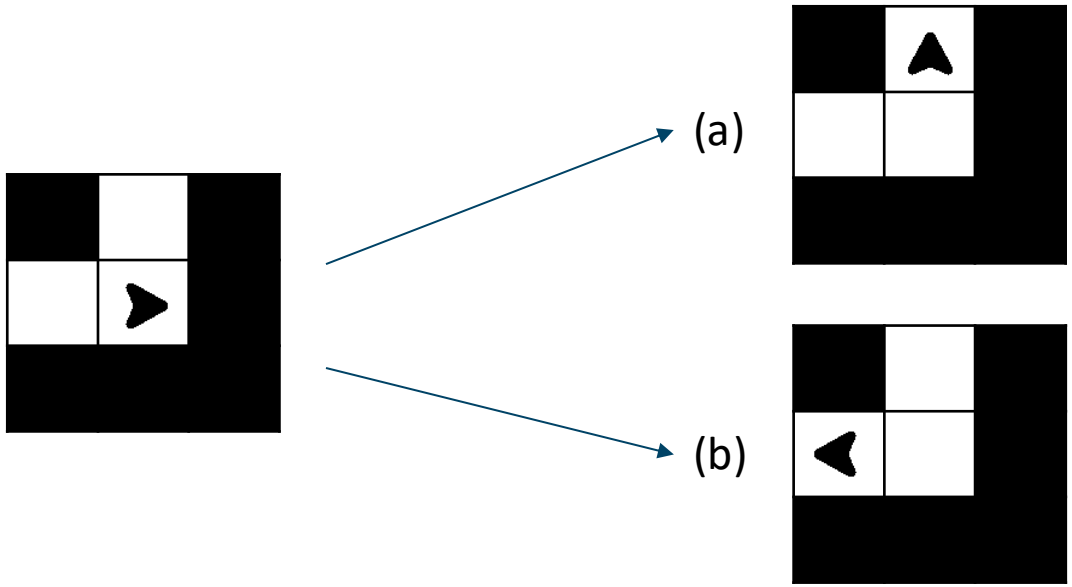
Waarheen in volgend voorbeeld?



De rechterhandregel

Volg steeds het meest rechtse pad

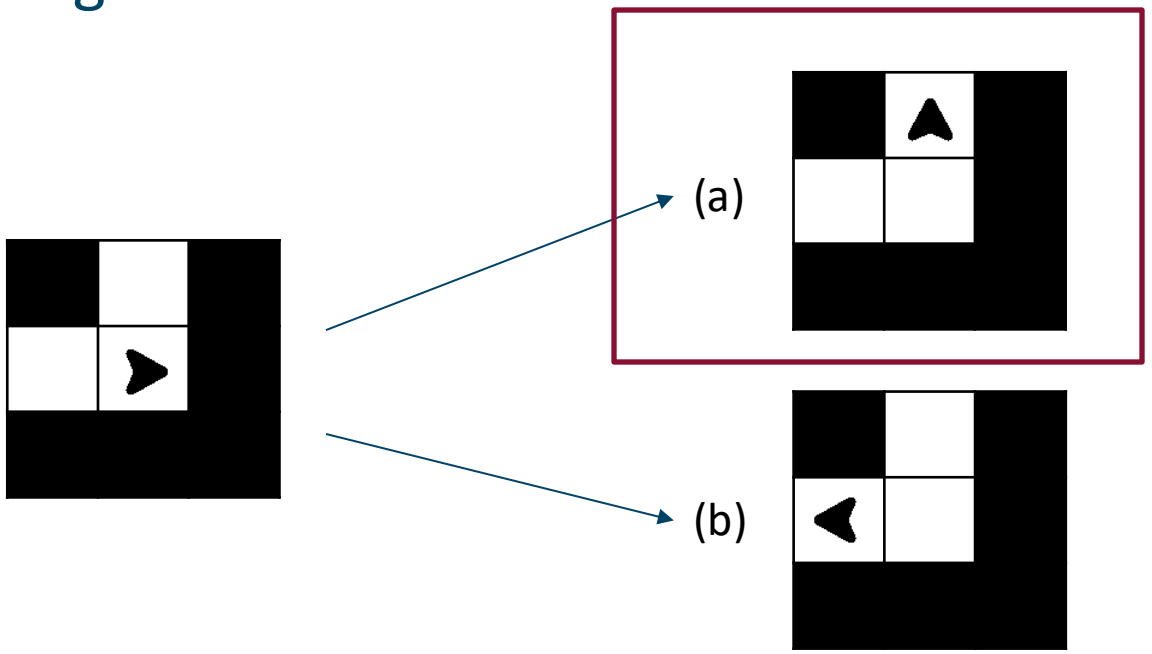
Waarheen in volgend voorbeeld?



De rechterhandregel

Volg steeds het meest rechtse pad

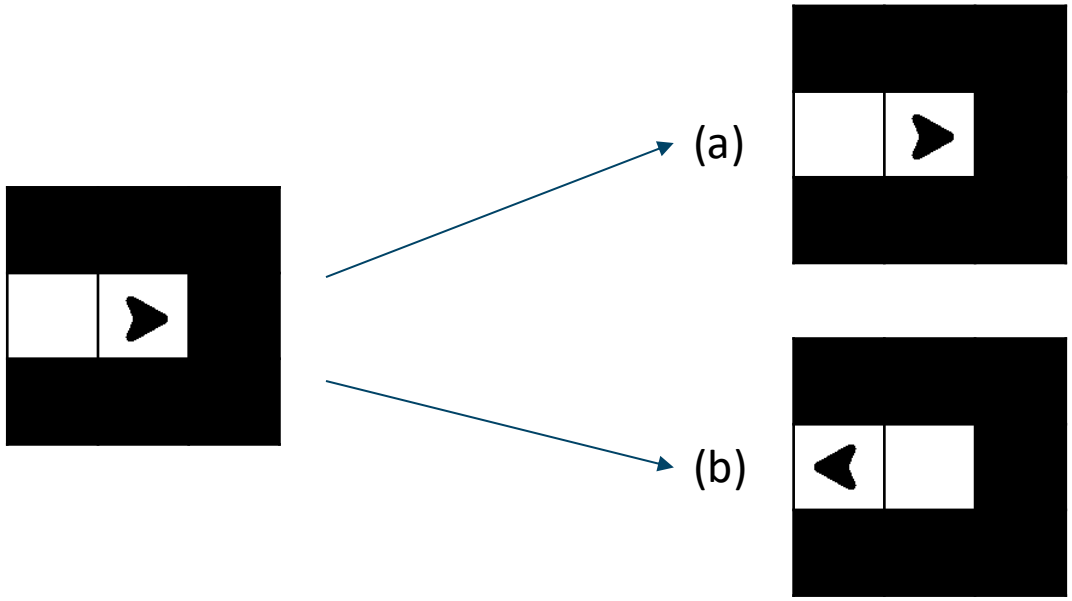
Waarheen in volgend voorbeeld?



De rechterhandregel

Volg steeds het meest rechtse pad

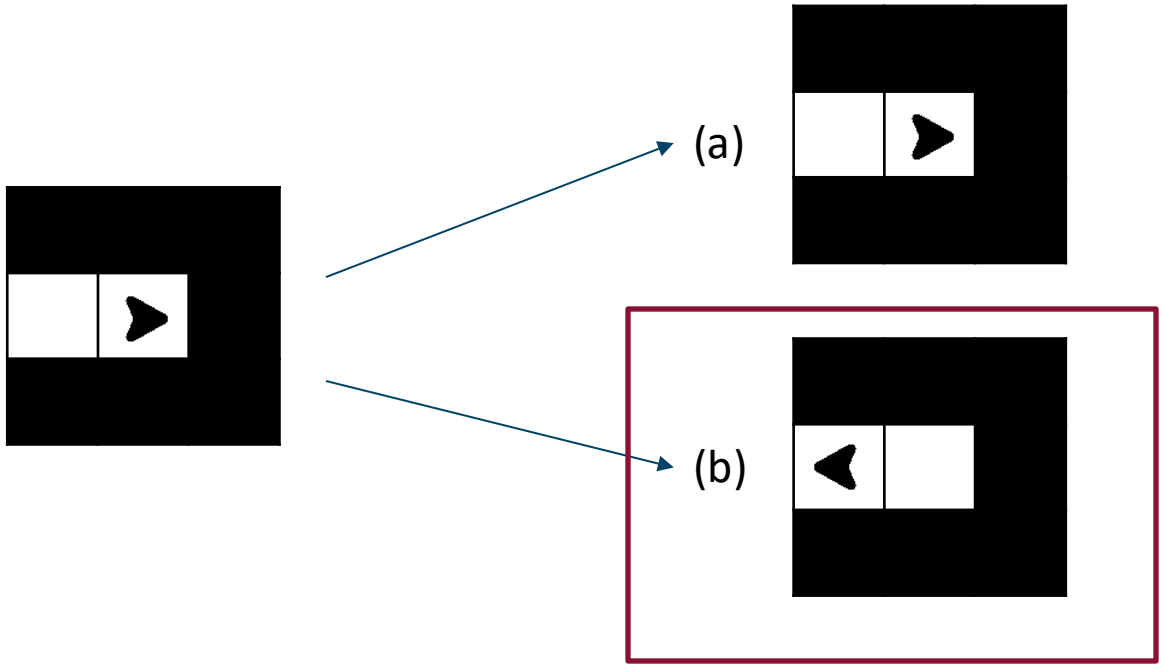
Waarheen in volgend voorbeeld?



De rechterhandregel

Volg steeds het meest rechtse pad

Waarheen in volgend voorbeeld?



Rechterhandregel

Volg steeds het meest rechtse pad

=

Zolang je de uitgang niet bereikte:

als rechts vrij is:

draai naar rechts en vooruit

anders als vooruit vrij is:

vooruit

anders als links vrij is:

draai naar links en vooruit

anders:

draai 180gr en vooruit

Opdracht #4 vraag4.py

Maak een programma dat de rechterhandregel volgt.

Je kan dit testen op doolhof 4:

```
# Vraag 4: Los doorhof 4 op
laadDoolhof("doolhof4.txt")
# laadDoolhof("doolhof5.txt") # 4 gelukt? vervang vorige regel dan door deze!
```

Zolang je de uitgang niet bereikte:

neem het meest rechtse pad

Ben je klaar? Los dan ook eens doolhof5 op!

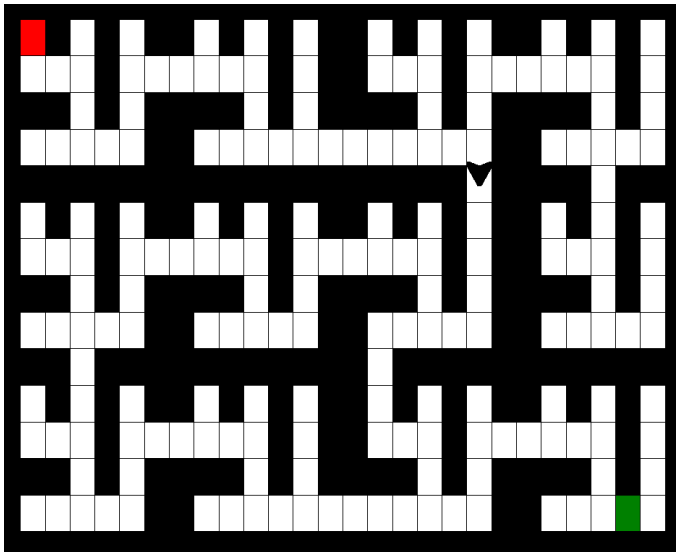
Oplossing #4

Zolang je de uitgang niet bereikte:

zoek meest rechtse doorgang

ga vooruit

```
while(not foundExit()):
    if freeRight():
        turnRight()
        goForward()
    elif freeForward():
        goForward()
    elif freeLeft():
        turnLeft()
        goForward()
    else:
        turnRight()
        turnRight()
        goForward()
```



Samenvatting

Programmeren = stap voor stap instructies geven
om een probleem op te lossen

Belangrijke constructies:

Sequentie van commando's : een voor een uitvoeren

Herhalingslussen : zolang … herhaal …

Voorwaardelijke uitvoering : if … elif … else …

Extras:

Variabelen om te onthouden en te rekenen

Informatica

Meer dan alleen programmeren

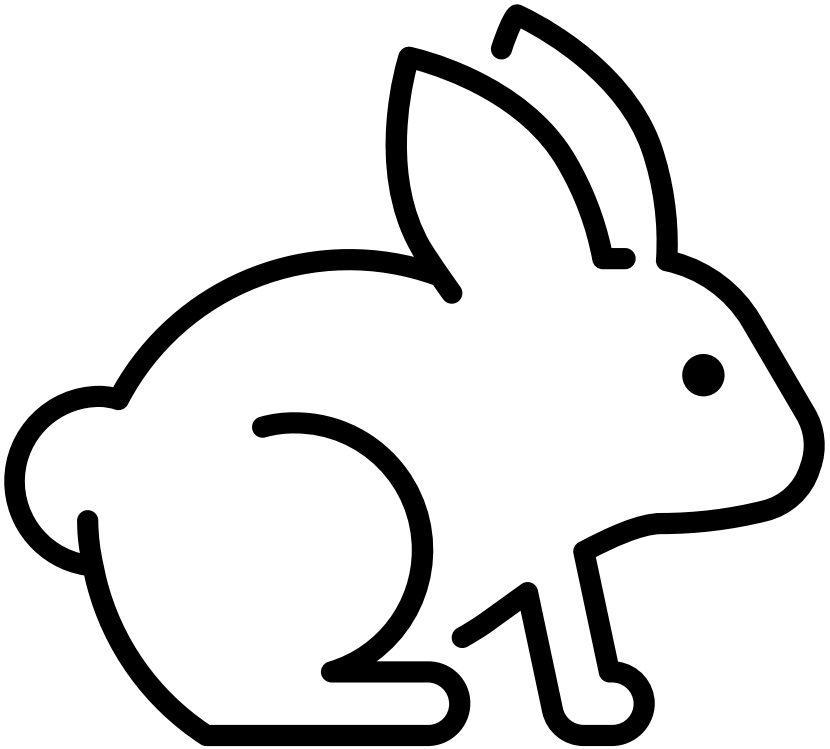
- Algoritme: rechterhandregel
  - Aantal stappen in slechtste geval?
  - Efficiënter algoritme mogelijk?
  - Wat als in- en uitgang niet aan de buitenkant liggen?
  - Hoe het *kortste* pad vinden?

- Ook: datastructuren, besturingssystemen, netwerken, gedistribueerde systemen, databanken, …

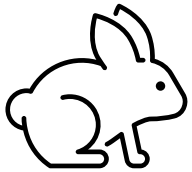
Bedankt voor jullie aanwezigheid!

Tot volgend jaar?

Extra materiaal voor de snelle programmeurs …



Variabelen



Stel dat we willen onthouden hoeveel stappen we namen

- Moeten dingen kunnen onthouden
- Kan met behulp van *variabelen*

Een variabele is een naam voor een geheugenplaats

- Zelfgekozen naam
- We kunnen waarden toekennen, aanpassen, uitlezen

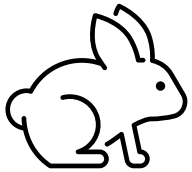
```
steps=0
```

initializatie

```
steps=steps+1
```

Waarde aanpassen

Tel het totaal aantal stappen

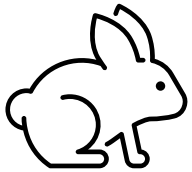


```
steps=0
while not foundExit():
    steps=steps+1
    if freeRight(): # right is free
        turnRight()
        goForward()
    elif freeForward(): # not right, but forward is free
        goForward()
    elif freeLeft():  # not right, not forward, but left free
        turnLeft()
        goForward
    else: # neither right, nor forward, nor left are free
        turnLeft()
        turnLeft()
        goForward()

print("Aantal stappen:",steps)
```

Print geeft de *waarde* van uitdrukkingen of variabelen; hier de waarde van de *string* “Aantal stappen:” en de waarde van de variabele steps

Opdracht #5

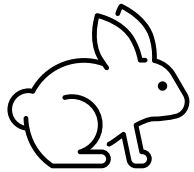
vraag5.py



Tel hoeveel maal de robot:

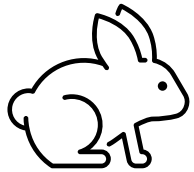
- Naar links draait
- Naar rechts draait

Opdracht #5 : Oplossing



```
steps=0
right=0
left=0
while not foundExit():
    steps=steps+1
    if freeRight(): # right is free
        turnRight()
        right=right+1
        goForward()
    elif freeForward(): # not right, but forward is free
        goForward()
    elif freeLeft():  # not right, not forward, but left free
        turnLeft()
        left=left+1
        goForward
    else: # neither right, nor forward, nor left are free
        turnLeft()
        turnLeft()
        left=left+2
        goForward()

print("Aantal stappen:",steps)
print("Rechts:",right,"links:",left)
```

Variabelen gebruiken in *if* en *while*



We kunnen bewerkingen doen met een variabele:

```
cost = 3*left+2*right+steps
    #kostprijs afhankelijk van de operatie
```

vermenigvuldiging

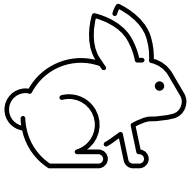
We kunnen testen of een variabele of uitdrukking kleiner, gelijk, groter is dan een waarde of een andere variabele of uitdrukking

Let op! 2x = om te testen op gelijkheid!

```
if left > right:
    print("Meer draaien naar links dan naar rechts")
elif left == right:
    print("Even veel draaien links als rechts")
else:
    print("Meer draaien rechts dan links")
```

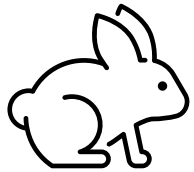
Opdracht #6

vraag6.py



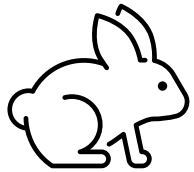
Test welk algoritme het grootste aantal stappen nodig heeft in het doolhof: de rechterhand regel of de linkerhand regel?

Opdracht #6 Oplossing



```
stepsRHR=0
while not foundExit():
    stepsRHR=stepsRHR+1
    if freeRight(): # right is free
        turnRight()
        goForward()
    elif freeForward(): # not right, but forward is free
        goForward()
    elif freeLeft():  # not right, not forward, but left free
        turnLeft()
        goForward()
    else: # neither right, nor forward, nor left are free
        turnLeft()
        turnLeft()
        goForward()

# Right-hand rule
laadDoolhof("doolhof4.txt")
# Now do the Lefthand rule and count steps
stepsLHR=0
while not foundExit():
    stepsLHR=stepsLHR+1
    if freeLeft(): # left is free
        turnLeft()
        goForward()
    elif freeForward(): # not left, but forward is free
        goForward()
    elif freeRight():  # not left, not forward, but right free
        turnRight()
        goForward()
    else: # neither right, nor forward, nor left are free
        turnRight()
        turnRight()
        goForward()

# Decide which one is larger and print output
if stepsLHR==stepsRHR:
    print("Beiden nemen hetzelfde aantal stappen")
elif stepsLHR<stepsRHR:
    print("Linkerhand regel is efficienter:",stepsLHR,"vs.",stepsRHR)
else:
    print("Rechterhand regel is efficienter:",stepsRHR,"vs.",stepsLHR)
```

Opdracht #6 Oplossing



```
stepsRHR=0
while not foundExit():
    stepsRHR=stepsRHR+1
    if freeRight(): # right is free
        turnRight()
        goForward()
    elif freeForward(): # not right, but forward is free
        goForward()
    elif freeLeft():  # not right, not forward, but left free
        turnLeft()
        goForward()
    else: # neither right, nor forward, nor left are free
        turnLeft()
        turnLeft()
        goForward()

# Right-hand rule
laadDoolhof("doolhof4.txt")
# Now do the Lefthand rule and count steps
stepsLHR=0
while not foundExit():
    stepsLHR=stepsLHR+1
    if freeLeft(): # left is free
        turnLeft()
        goForward()
    elif freeForward(): # not left, but forward is free
        goForward()
    elif freeRight():  # not left, not forward, but right free
        turnRight()
        goForward()
    else: # neither left, nor forward, nor right are free
        turnRight()
        turnRight()
        goForward()

# Decide which one is larger and print output
if stepsLHR==stepsRHR:
    print("Beiden nemen hetzelfde aantal stappen")
elif stepsLHR<stepsRHR:
    print("Linkerhand regel is efficienter:",stepsLHR,"vs.",stepsRHR)
else:
    print("Rechterhand regel is efficienter:",stepsRHR,"vs.",stepsLHR)
```