1. Zet je code in beweging

In wiskunde en wetenschappen zijn we vaak op zoek naar patronen, modellen en wetmatigheden. In de klas worden tal van algoritmes bestudeerd voor het oplossen van problemen.

Coderen komt er ook op neer algoritmes te programmeren om bepaalde taken automatiseren en samen een oplossing te bieden.

Met de TI-Innovator Rover kunnen we met beweging code visualiseren. Dit maakt het programmeren uitdagender terwijl er gebruik gemaakt wordt van wiskundige en wetenschappelijk concepten.

Zonder in detail te treden van alle mogelijkheden van de TI-Innovator Rover, bekijken we enkele voorbeelden hoe het coderen van de Rover in zijn werk gaat.

Meer info is te vinden op www.education.ti.com/nl of www.education.ti.com/be-nl.

We maken connectie tussen een TI-handheld (TI-Nspire CX II-T (CAS) of TI-84 Plus CE-T PYTHON EDITION) met de code: import ti_rover as rv. Hiermee worden alle methodes(functies) uit de TI Rover-module geïmporteerd voor het object rv.

1.1. Voor- en achteruitrijden

Met de commando's

- rv.forward(1)
- rv.backward(1)

rijdt de Rover 1 unit (10 cm) van het virtuele grid (10 cm) respectievelijk vooruit en achteruit.

Andere eenheden kunnen als volgt toegevoegd worden:

- r.forward(1,"m") 1 meter vooruit
- r.backward(1,"revs") 1 wielomwenteling vooruit

De default-eenheid is grid "units".

1.2. Naar links en rechts draaien

Met de commando's

- rv.left(45)
- rv.right(45)

draait de Rover 45 graden respectievelijk links (tegenwijzerzin) en rechts (wijzerzin).

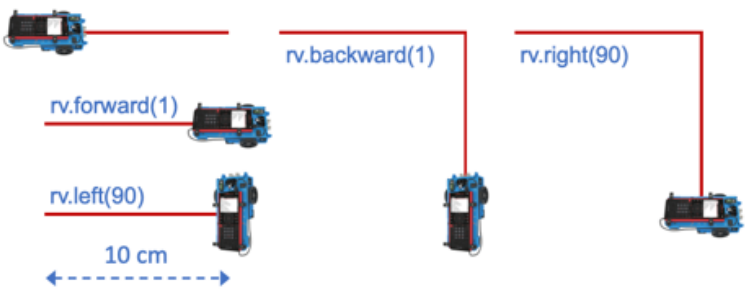
Ook aan deze commando's kan een eenheid toegevoegd worden:

- rv.left(π,"radians") radialen
- rv.right(50,"gradians") 100-delige graden

De default-eenheid is "degrees".

1.3. Een eerste move

```
import ti_rover as rv
rv.forward(1)
rv.left(90)
rv.backward(1)
rv.right(90)
```



1.4. Regelmatige veelhoeken

Een eenvoudige combinatie van deze codes en een for-lus resulteert in het rijden van een pad in de vorm van een regelmatige veelhoek.

Vierkant

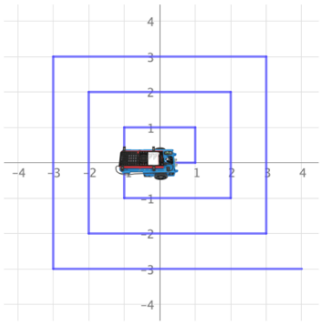
```
import ti_rover as rv

for i in range(4):
    rv.forward(3)
    rv.right(90)
```

Driehoek

```
import ti_rover as rv

for i in range(3):
    rv.forward(3)
    rv.left(120)
```

1.5. Spiralen

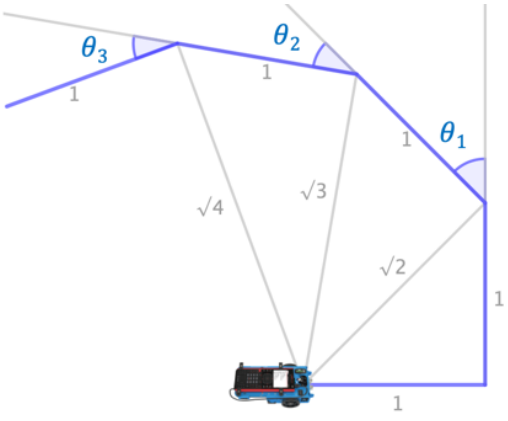
Voorbeeld 1 – Square spiral



```
import ti_rover as rv
for i in range(1,8):
    rv.forward(i)
    rv.left(90)
    rv.forward(i)
    rv.left(90)
```

Voorbeeld 2 – Pythagorean spiral

```
from math import *
import ti_rover as rv
rv.forward(1)
rv.left(90)
rv.forward(1)
for i in range(1,4):
    h=atan(1/sqrt(i))
    rv.left(h,"radians")
    rv.forward(1)
```



$$\theta_1 = \tan^{-1}\left(\frac{1}{\sqrt{1}}\right)$$

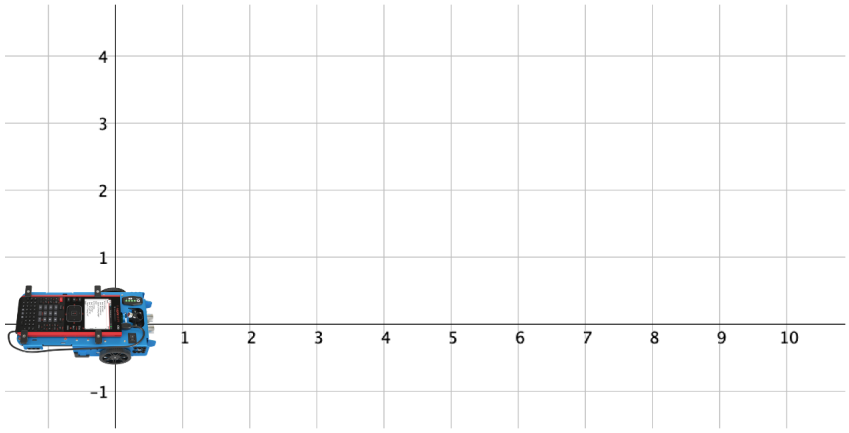
$$\theta_2 = \tan^{-1}\left(\frac{1}{\sqrt{2}}\right)$$

$$\theta_3 = \tan^{-1}\left(\frac{1}{\sqrt{3}}\right)$$

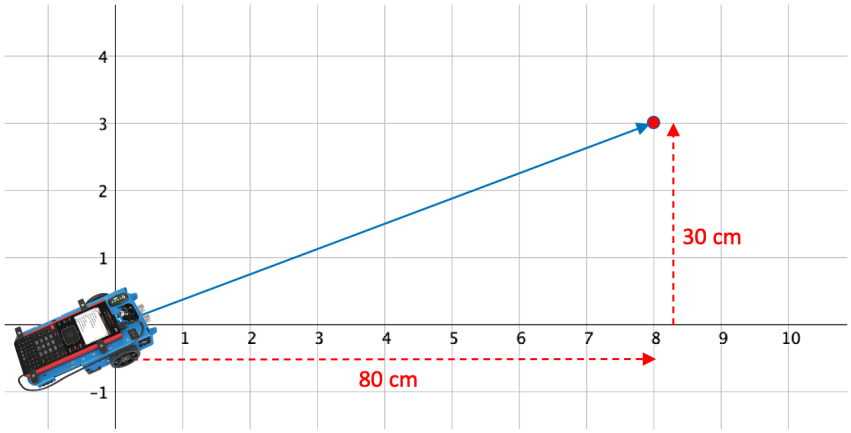
$$\vdots$$

2. Rijden via coördinaten

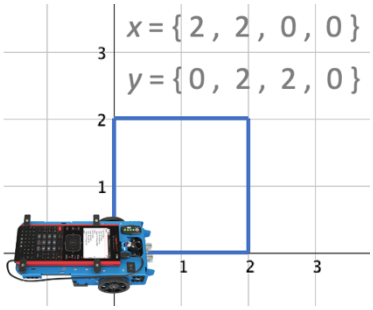
Met de code rv.to_xy(x,y) geef je de Rover de instructie naar het punt met coördinaten (x,y) te rijden in het virtuele grid met de Rover als volgt startend in de oorsprong:



De code `import ti_rover as rv` `rv.to_xy(8,3)` laat de Rover eerst gepast draaien en dan naar het punt met coördinaten (8,3) rijden.



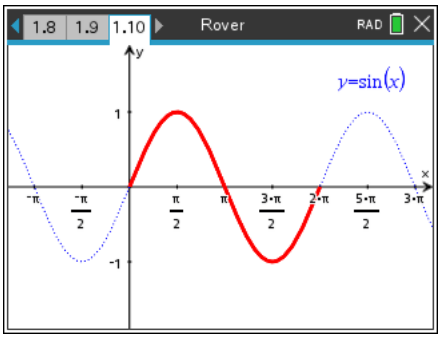
M.b.v. een for-lus rijden we het volgende vierkant:




```
import ti_rover as rv

x=[2,2,0,0]
y=[0,2,2,0]

for i in range(len(x)):
    rv.to_xy(x[i],y[i])
```

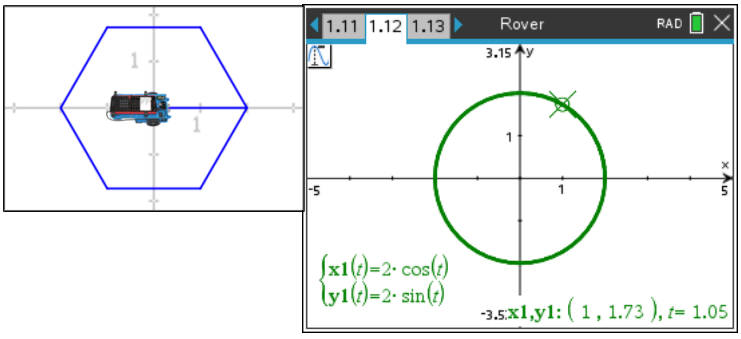
3. Het rijden van functies

Voorbeeld 1 – A sin ride




```
from math import *
import ti_rover as rv

points = 10
scale = 2*pi/points
xcoor=[i*scale for i in range(points+1)]

for i in xcoord:
    rv.to_xy(i,sin(i))
```

Voorbeeld 2 – In een cirkel rijden

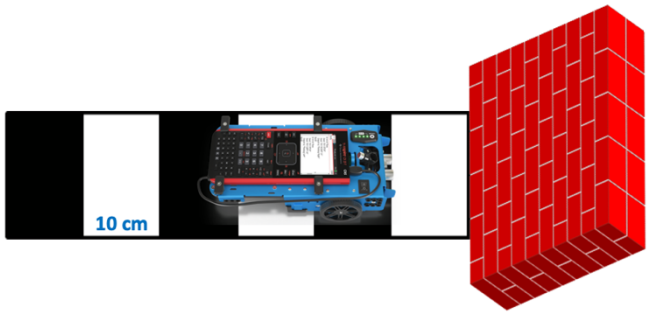



```
from math import *
import ti_rover as rv

points = 6
scale = 2*pi/points
parameter=[i*scale for i in range(points+1)]

for t  in parameter:
    x=2*cos(t)
    y=2*sin(t)
    rv.to_xy(x,y)
```

4. Metend rijden

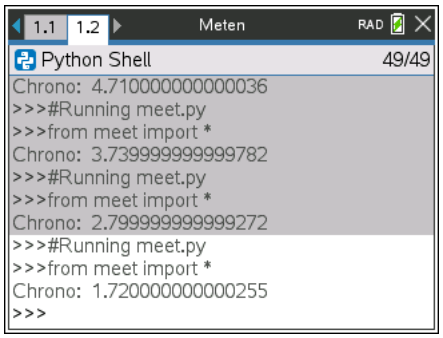
Aan de voorkant van de Rover is een ranger bevestigd waarmee de afstand (in meter) van de Rover tot een object gemeten wordt. Indien de rover te dicht komt met b.v. een muur brengt de methode stop() de Rover tot stilstand.

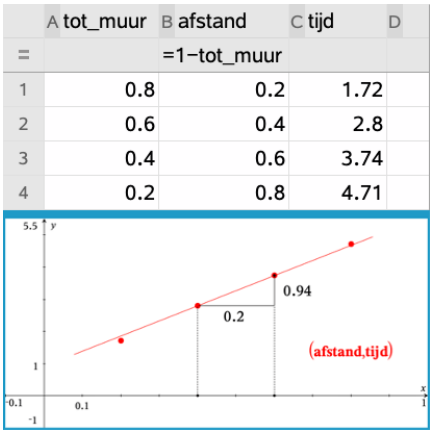
```
import ti_rover as rv
afstand=rv.ranger_measurement()
rv.forward(100)
while afstand > 0.1:
    afstand=rv.ranger_measurement()
rv.stop()
```

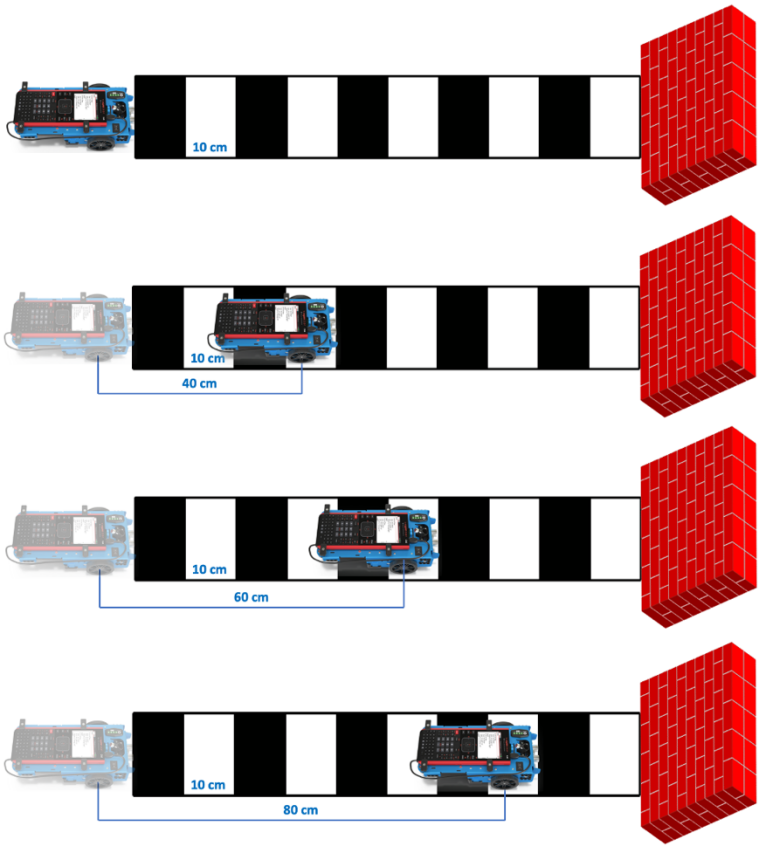


Het volgende programma berekent een ruwe benadering van de snelheid van de eenparig rechtlijnige beweging van de rover. Het statement clock() van de time-module geeft de processor tijd als een float-getal, uitgedrukt in seconden.

```
from time import *
import ti_rover as rv

afstand=rv.ranger_measurement()

start=clock()
rv.forward(100)

tot_muur=0.6

while afstand > tot_muur:
  afstand=rv.ranger_measurement()

rv.stop()
einde=clock()

print("Chrono: ",einde-start)
```







2.80 seconden

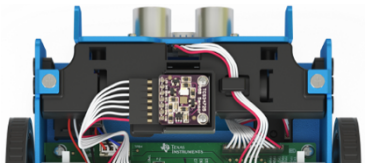
3.74 seconden

4.71 seconden

Het experiment geeft een de benadering $v = \frac{0.2}{0.94} = 0.21\frac{m}{s}$ van de standaard snelheid van de Rover: $0.2\frac{m}{s}$.

De snelheid van de Rover kan varieëeren tussen $0.14\frac{m}{s}$ en $0.23\frac{m}{s}$.

De Rover beschikt ook over een kleuren-sensor waarmee bepaald wordt wat de kleur is waarover de Rover rijdt; Deze kleuren-sensor ledit ook tot uitdagende programmeeropdrachten.



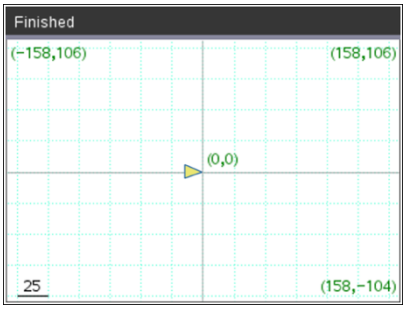
5. Rover versus Turtle

Met de Python Turtle-module kunnen we de Rover voor de voorgaande TI Python programma's simuleren, uitgezonderd deze i.v.m meten. Download en installeer de Turtle-module:

- education.ti.com/nl/product-resources/turtle-module/nspire-python of
- education.ti.com/nl-be/product-resources/turtle-module/nspire-python.

Het runnen van code gebruikmakend van de Turtle-module kan enkel voor de Handheld Page Size.

De Turtle-module maakt gebruik van de volgende scherminstellingen: (xmin,xmax) = (-158,158) en (ymin,ymax) = (-104,106).



Het beginnen met tekenen start met het aanmaken van een Turtle-object:

```
from turtle import *
rv=turtle()
```

Bij start van een programma bevindt rv zich in de oorsprong. Door het toevoegen, na het definiëren van het Turtle-object, van de volgende commando's

```
rv.hideturtle()
rv.hidegrid()
```

verbergen we de Turtle en het grid(assen en schaal inclusief). Voor de output van de onderstaande programma's werden de Turtle en het grid verborgen; zonder de commando's telkens in de code te vermelden.

Voor het simuleren van de Rover-programma's maken we gebruik van de volgende gelijkaardige code:

- rv.forward(distance) rv.backward(distance)
- rv.left(angle_degrees) r.right(angle_degrees)
- rv.clear() wist alle gemaakte tekeningen.
- rv.penup() en rv.pendown() laten toe rv te bewegen zonder te tekenen.
- rv.pencolor(r,g,b) en rv.pensize(value 1, 2 of 3) brengen wat kleur en variatie in het tekenen.

Enkele programma's

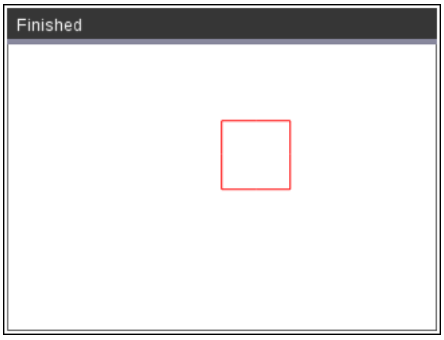
TI-Innovator Rover

Vierkant

```
import ti_rover as rv
for i in range(4):
    rv.forward(3)
    rv.right(90)
```

Turtle Rover

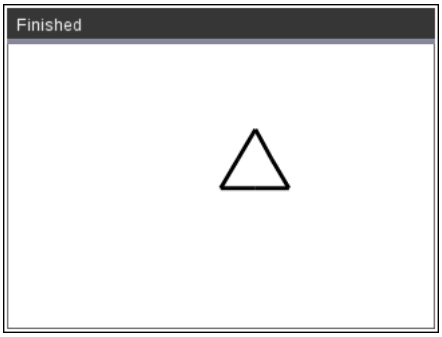
```
from turtle import *
rv=Turtle()
rv.pencolor(255,0,0)
for i in range(4):
    rv.forward(50)
    rv.left(90)
```



Driehoek

```
import ti_rover as rv
for i in range(3):
    rv.forward(3)
    rv.left(120)
```

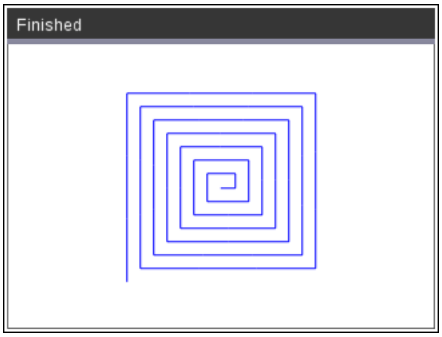
```
from turtle import *
rv=Turtle()
rv.pensize(2)
for i in range(3):
    rv.forward(50)
    rv.left(120)
```



Square spiral

```
import ti_rover as rv
for i in range(1,8):
    rv.forward(i)
    rv.left(90)
    rv.forward(i)
    rv.left(90)
```

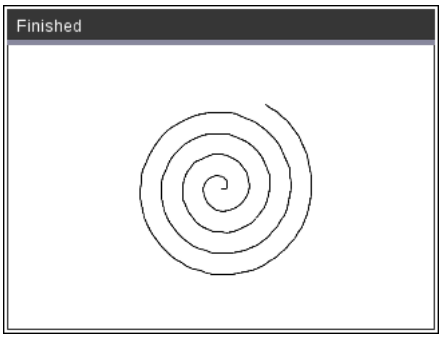
```
from turtle import *
rv=Turtle()
rv.pencolor(0,0,255)
for i in range(1,15):
    rv.forward(i*10)
    rv.left(90)
    rv.forward(i*10)
    rv.left(90)
```



Pythagorean Spiral

```
from math import *
import ti_rover as rv
rv.forward(1)
rv.left(90)
rv.forward(1)
for i in range(1,4):
    h=atan(1/sqrt(i))
    rv.left(h,"radians")
    rv.forward(1)
```

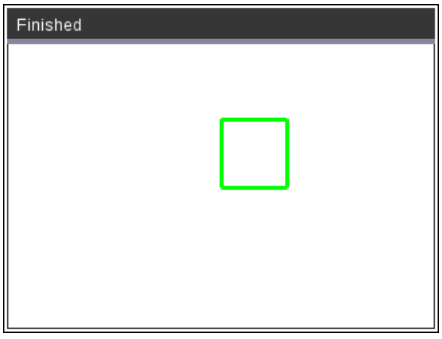
```
from turtle import *
from math import *
rv=Turtle()
rv.forward(5)
rv.left(90)
rv.forward(5)
for i in range(1,200):
    h1=1/sqrt(i)
    h2=atan(h1)*180/pi
    rv.left(h2)
    rv.forward(5)
```



Vierkant xy

```
import ti_rover as rv
x=[2,2,0,0]
y=[0,2,2,0]
for i in range(len(x)):
    rv.to_xy(x[i],y[i])
```

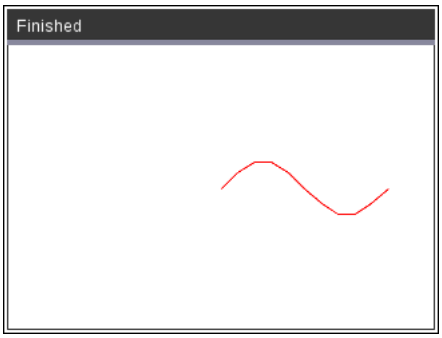
```
from turtle import *
rv=Turtle()
rv.pencolor(0,255,0)
rv.pensize(2)
x=[50,50,0,0]
y=[0,50,50,0]
for i in range(len(x)):
    rv.goto(x[i],y[i])
```



A sin ride

```
from math import *
import ti_rover as rv

points=10
scale=2*pi/points
x=[i*scale for i in range(points+1)]

for i in x:
    rv.to_xy(i,sin(i))
```

```
from turtle import *
from math import *
rv=Turtle()
rv.pencolor(255,0,0)
points=10
scale=2*pi/points
x=[i*scale for i in range(points+1)]
for i in x:
    rv.goto(20*i,20*sin(i))
```



In een cirkel rijden

```
from math import *
import ti_rover as rv

points = 6
scale = 2*pi/points
p=[i*scale for i in range(points+1)]

for t  in parameter:
    x=2*cos(t)
    y=2*sin(t)
    rv.to_xy(x,y)
```

```
from turtle import *
from math import *
rv=Turtle()
rv.pencolor(0,0,255)
points=10
scale=2*pi/points
p=[i*scale for i in range(points+1)]
for t  in p:
    x=50*cos(t)
    y=50*sin(t)
    rv.goto(x,y)
```