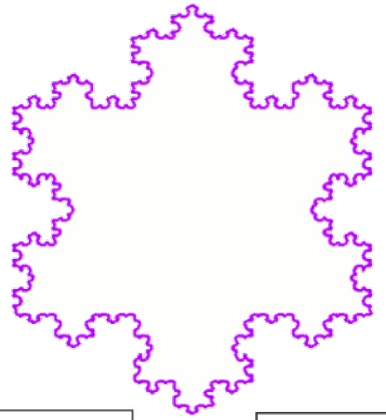
Python in de klas

Koen Stulens – Bert Wikkerink

k-stulens@ti.com

b.wikkerink@gmail.com



T³ NEDERLAND

T³ VLAANDEREN

Goesting in uitdagingen

Hoe kan technologie helpen?

1 Curiosity

> Why? Explore!

> Questions are windows to great instruction!

2 Embrace the mess

> Try, Guess and Explore!

> Learning is an inevitable process of trial & error

3 Reflection

Teach
Learn

> Teaching deserve revision

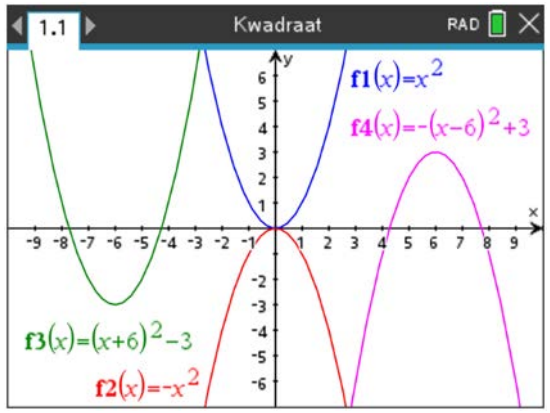
> Learning needs revision

www.ted.com/talks/ramsey_musallam_3_rules_to_spark_learning

Texas Instruments

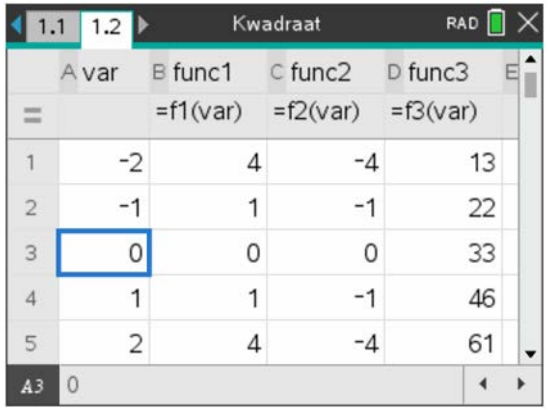
Representaties

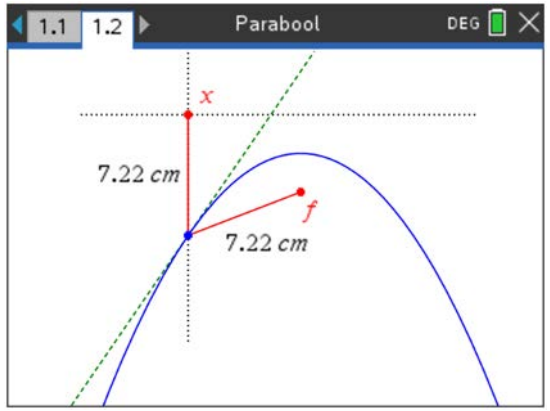
Grafisch - Analytisch



Tekst

Numeriek





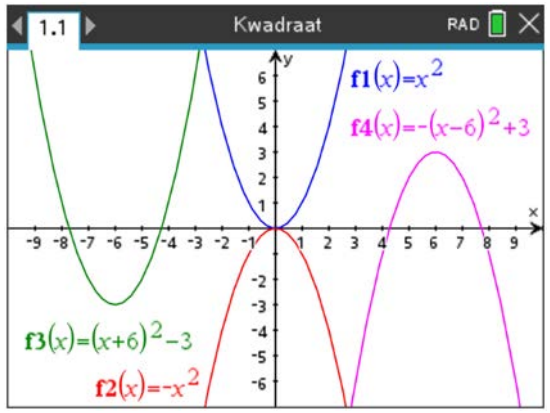
Grafisch - Meetkundig

Algebraïsch

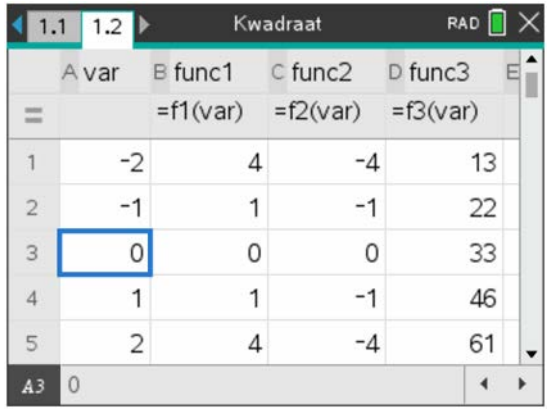
Representaties

Tekst

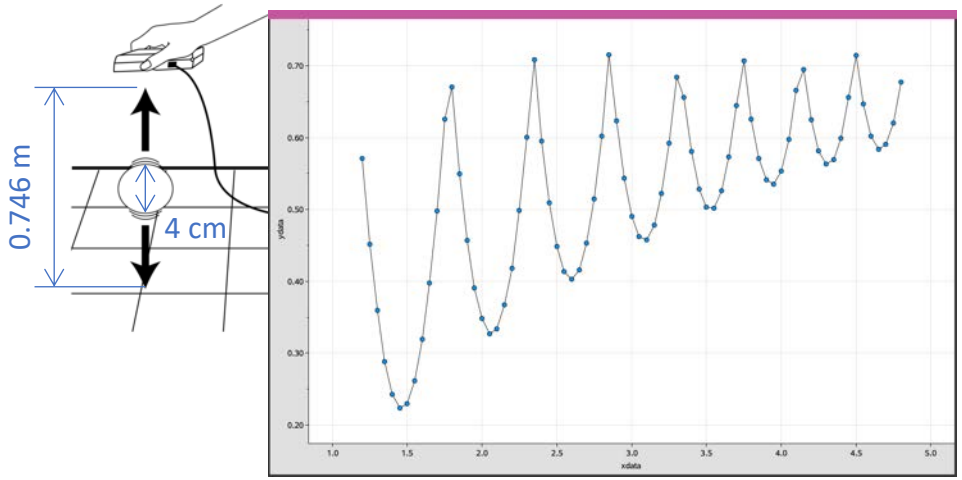
Grafisch - Analytisch

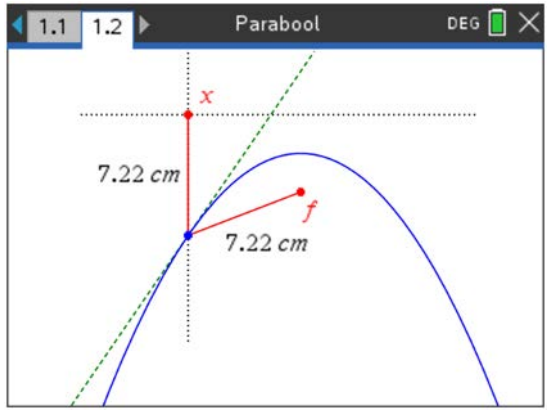


Numeriek



Modelleren



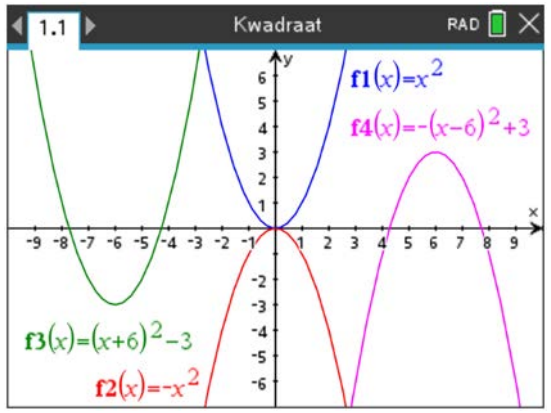


Grafisch - Meetkundig

Algebraïsch

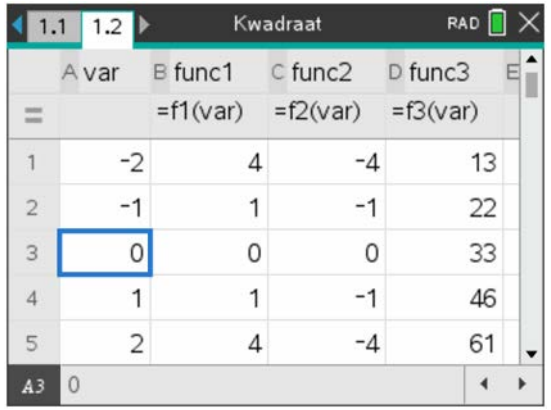
Representaties

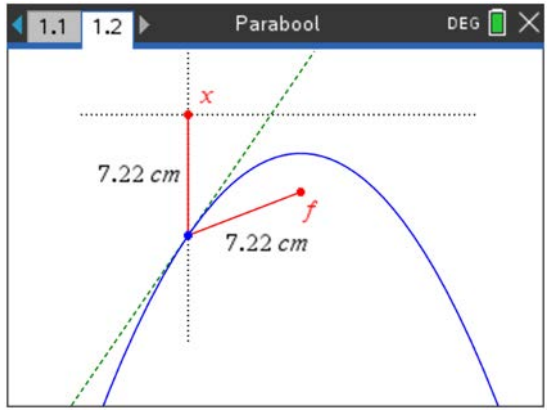
Grafisch - Analytisch



Tekst

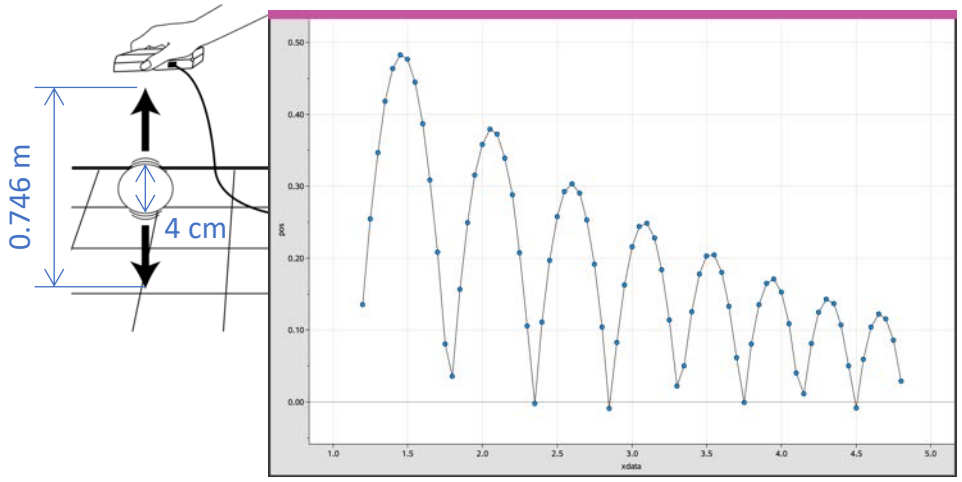
Numeriek





Grafisch - Meetkundig

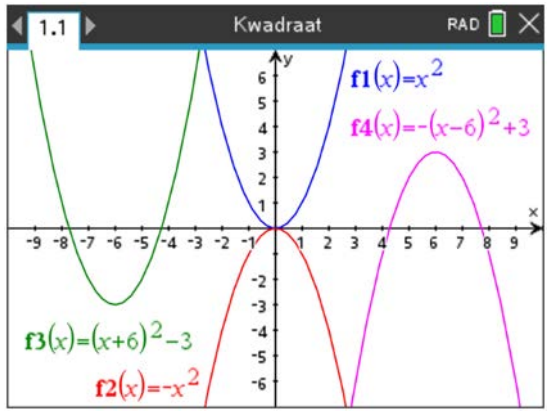
Modelleren



Algebraïsch

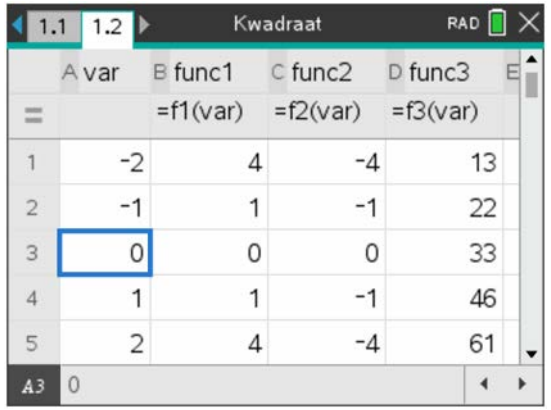
Representaties

Grafisch - Analytisch

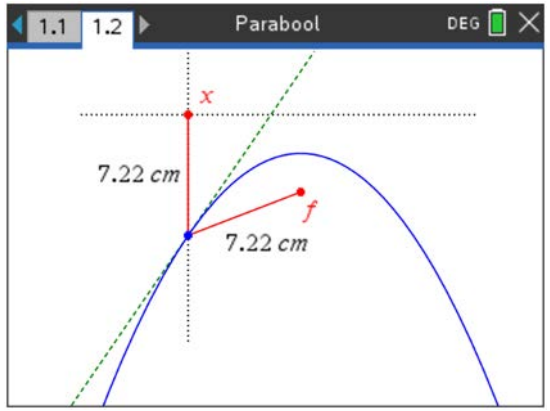


Tekst

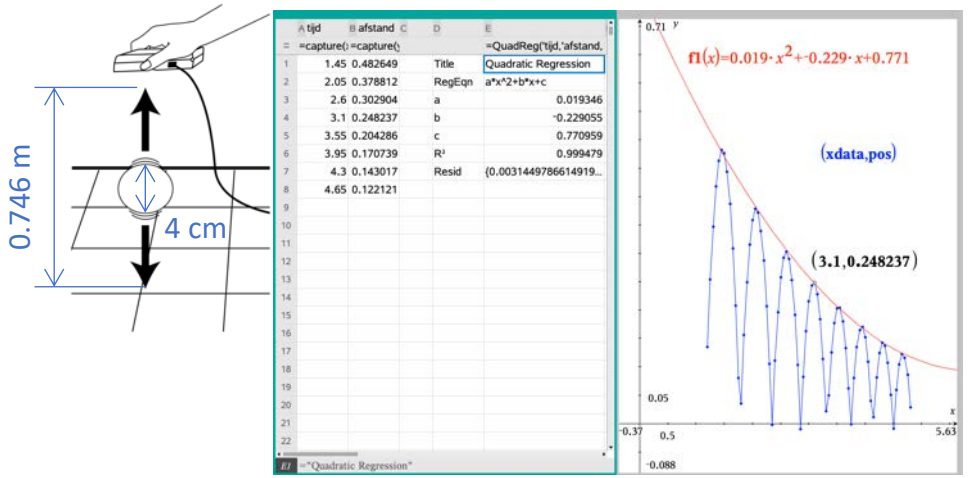
Numeriek



Grafisch - Meetkundig



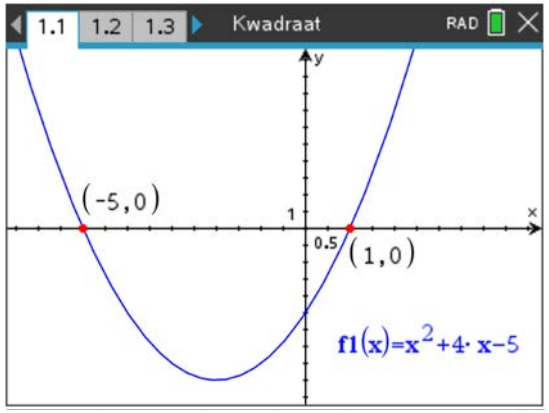
Modelleren



Algebraïsch

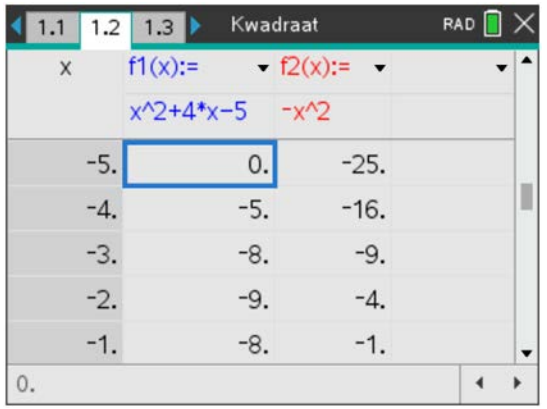
Representaties

Grafisch - Analytisch



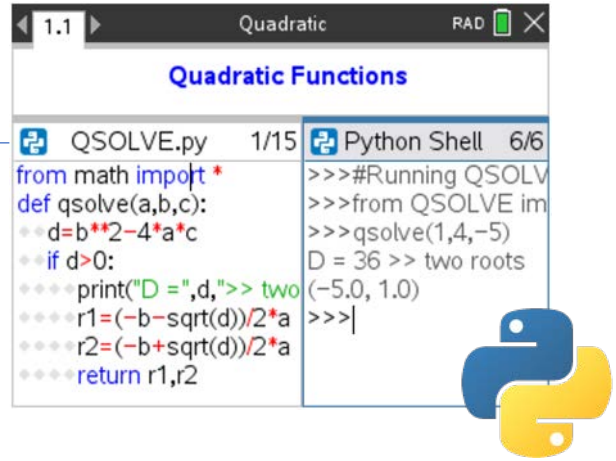
Tekst

Numeriek



Programmeren

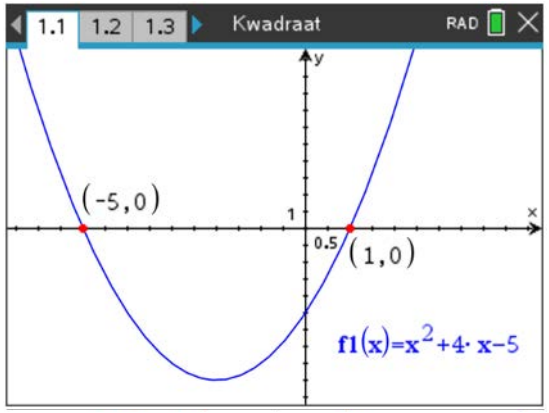
```
QSOLVE.py
from math import *
def qsolve(a,b,c):
    d=b**2-4*a*c
    if d>0:
        print("D =",d,">> two roots")
        r1=(-b-sqrt(d))/2*a
        r2=(-b+sqrt(d))/2*a
        return r1,r2
    elif d==0:
        print("D =",d,">> one root")
        r1=-b/2*a
        return r1
    else:
        print("D =",d,">> no real roots")
    return
```



Algebraïsch

Representaties

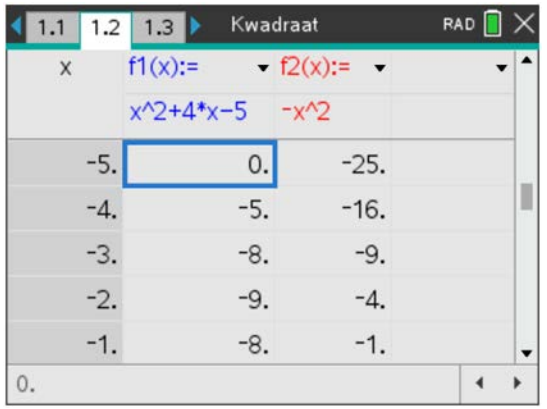
Grafisch - Analytisch



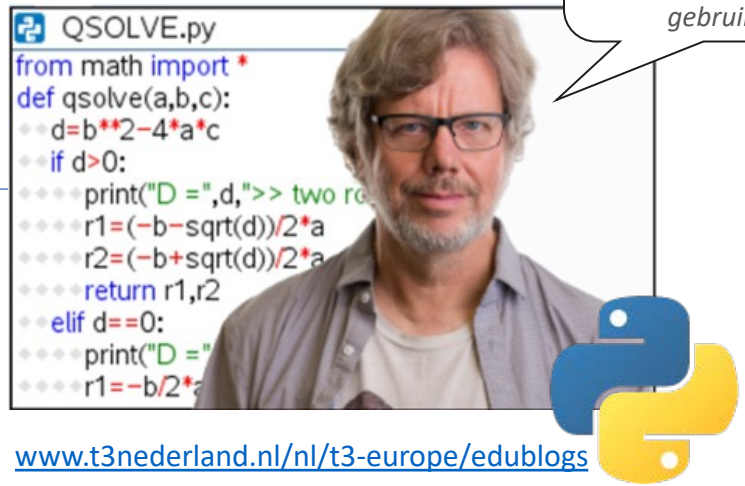

```
QSOLVE.py
from math import *
def qsolve(a,b,c):
    d=b**2-4*a*c
    if d>0:
        print("D =",d,">> two roots")
        r1=(-b-sqrt(d))/2*a
        r2=(-b+sqrt(d))/2*a
        return r1,r2
    elif d==0:
        print("D =",d,">> one root")
        r1=-b/2*a
        return r1
    else:
        print("D =",d,">> no real roots")
    return
```

Tekst

Numeriek

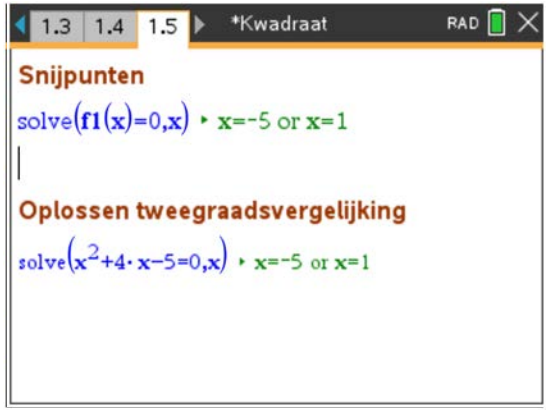


Programmeren



Ik hoop dat leerlingen zo enthousiast worden dat ze Python ook buiten schooltijd gaan gebruiken!

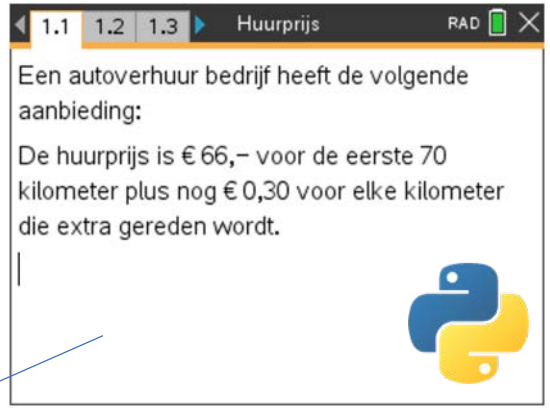
www.t3nederland.nl/nl/t3-europe/edublogs



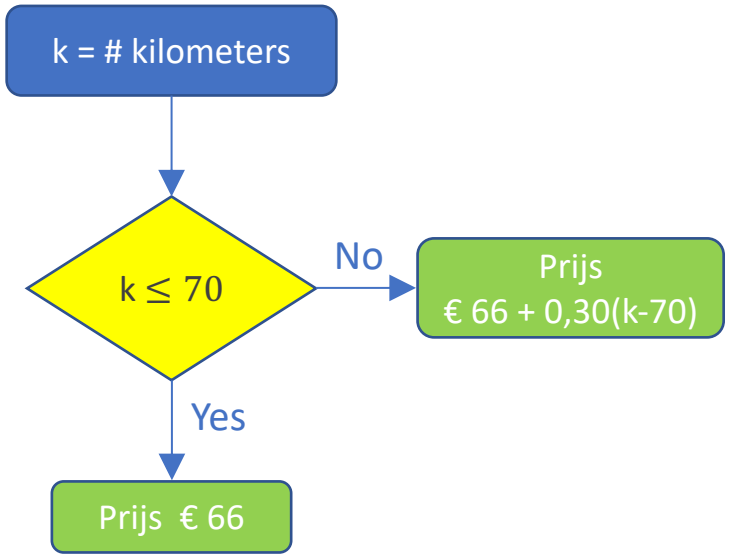
Algebraïsch

Programmeren

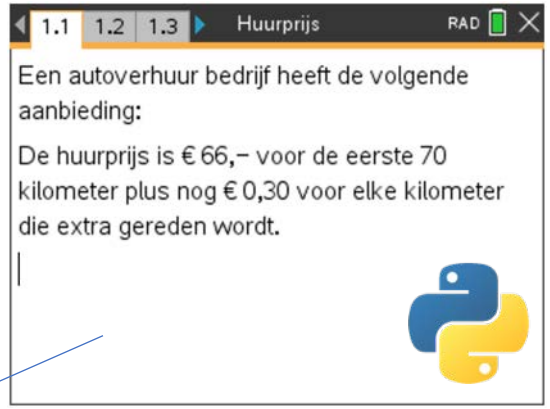
Programmeren



Analyseren

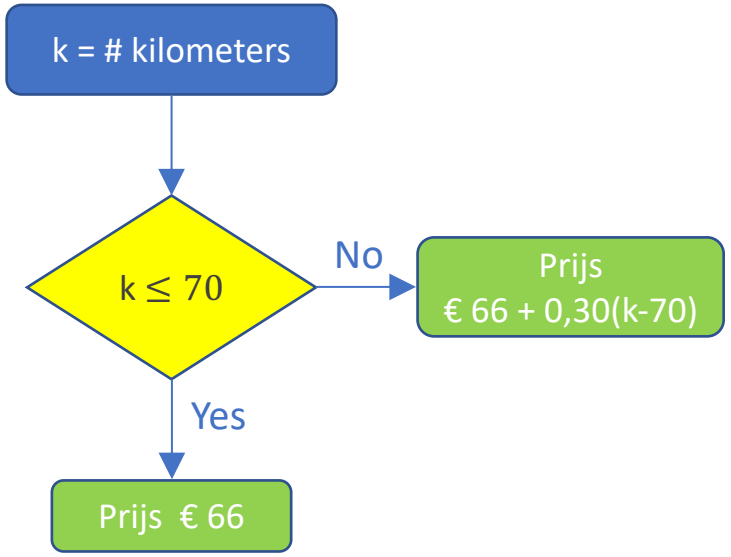


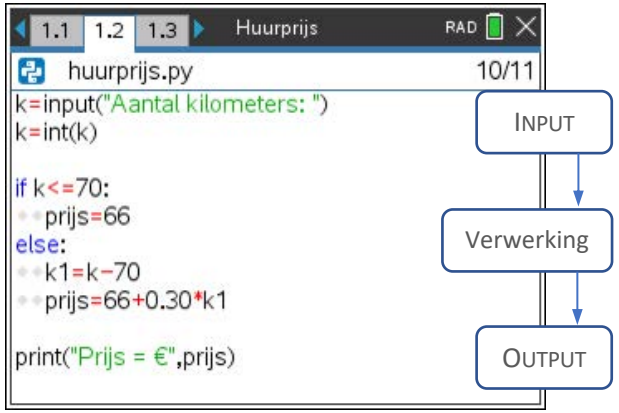
Programmeren



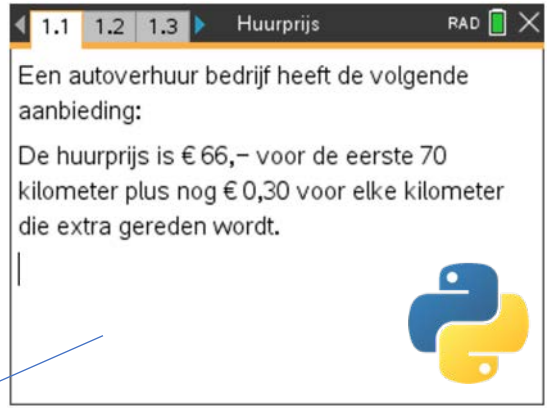
Analyseren

Vertalen

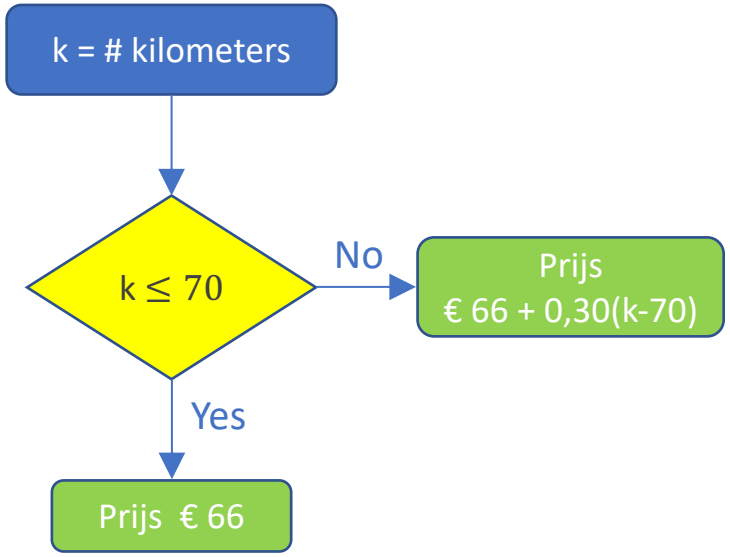


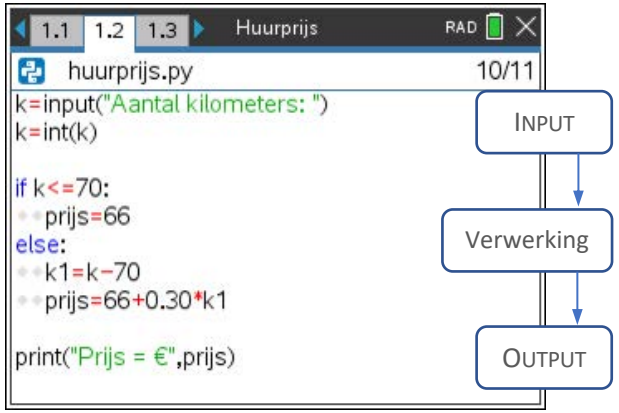


Programmeren



Analyseren → *Vertalen* → *Verwerking*

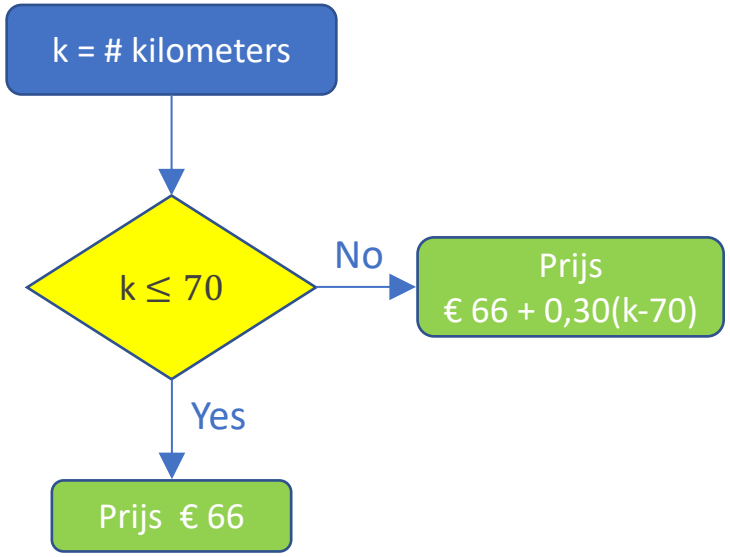


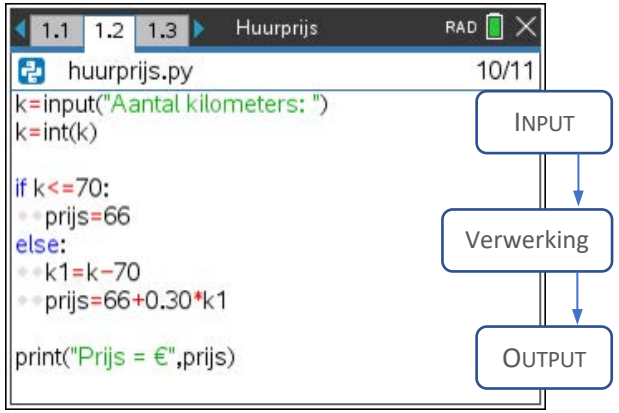


Programmeren

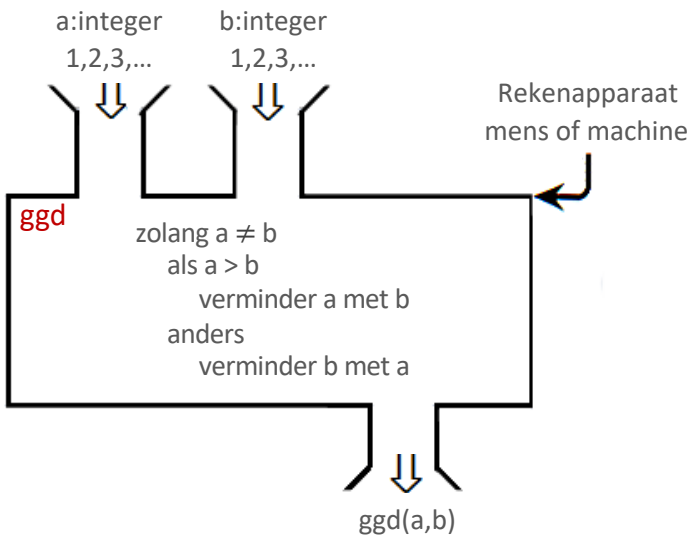
Controle

Analyseren → *Vertalen* → *Verwerking*

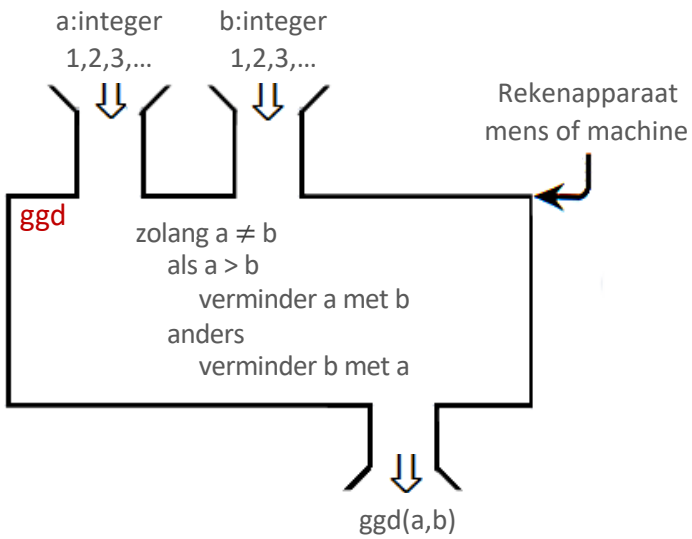


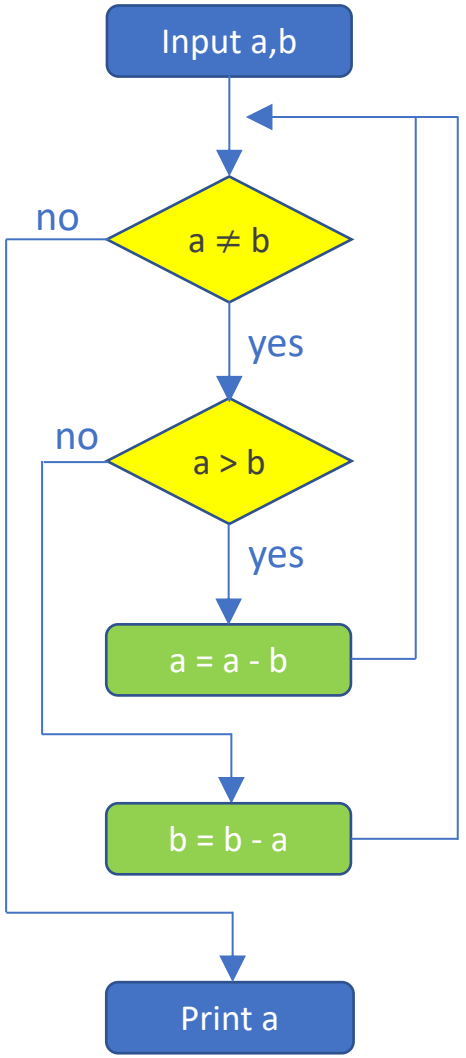


Algoritme van Euclides

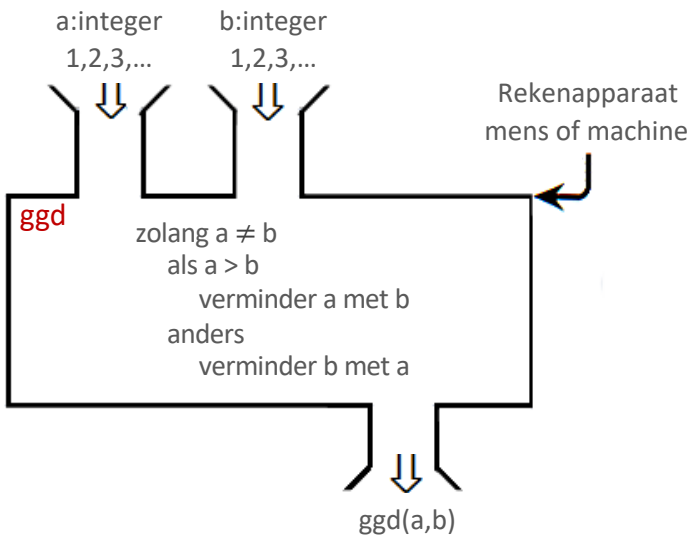


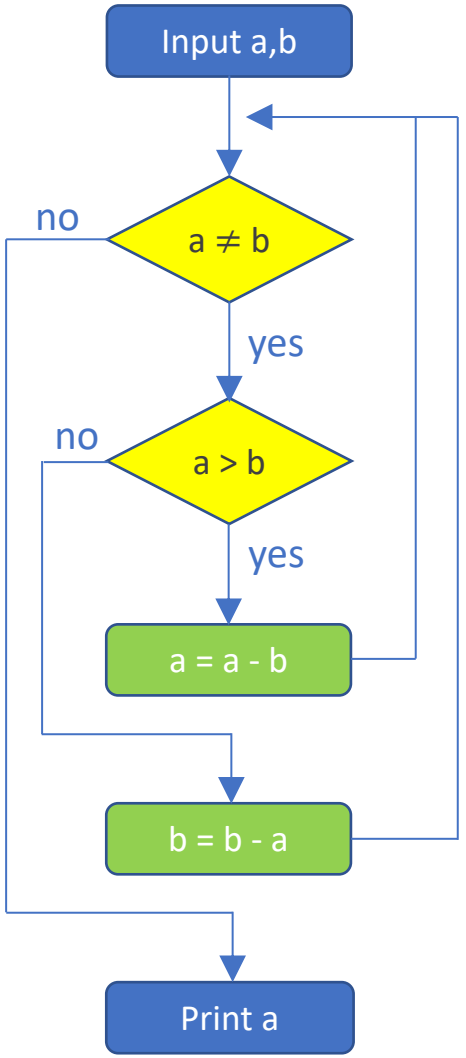
Algoritme van Euclides





Algoritme van Euclides





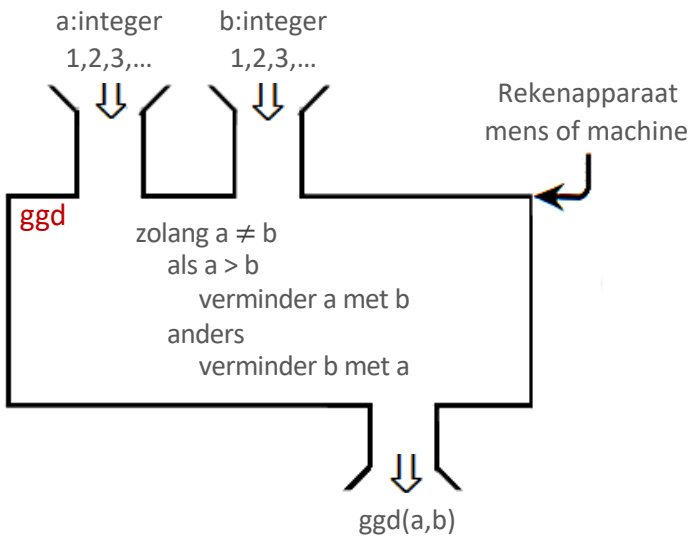
ggd.py

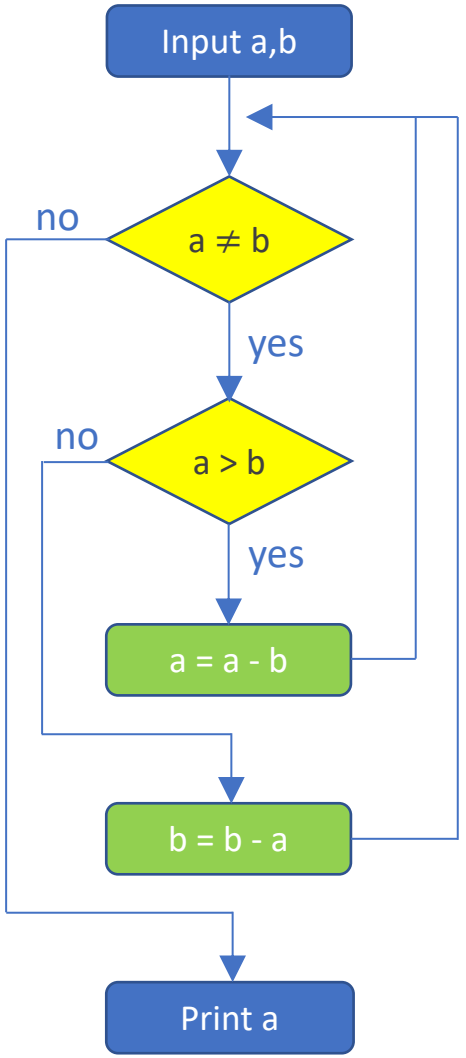
```
a=int(input("a= "))
b=int(input("b= "))

while a!=b:
    if a>b:
        a=a-b
    else:
        b=b-a

print(a)
```

Algoritme van Euclides





ggd.py

```
a=int(input("a= "))
b=int(input("b= "))

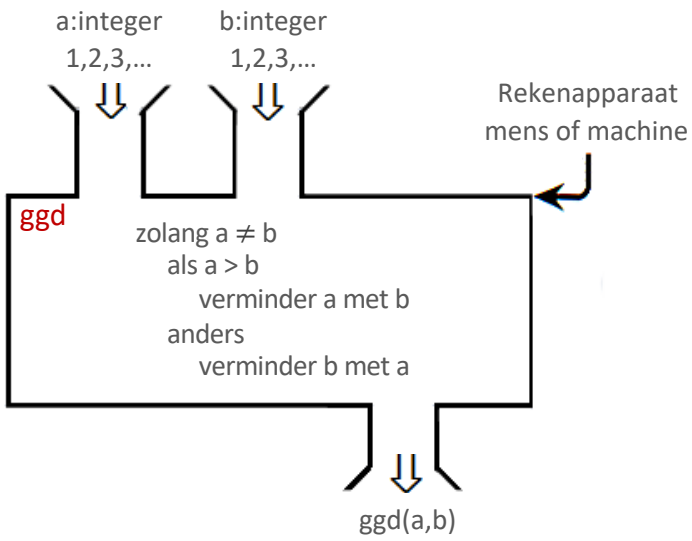
while a!=b:
    if a>b:
        a=a-b
    else:
        b=b-a

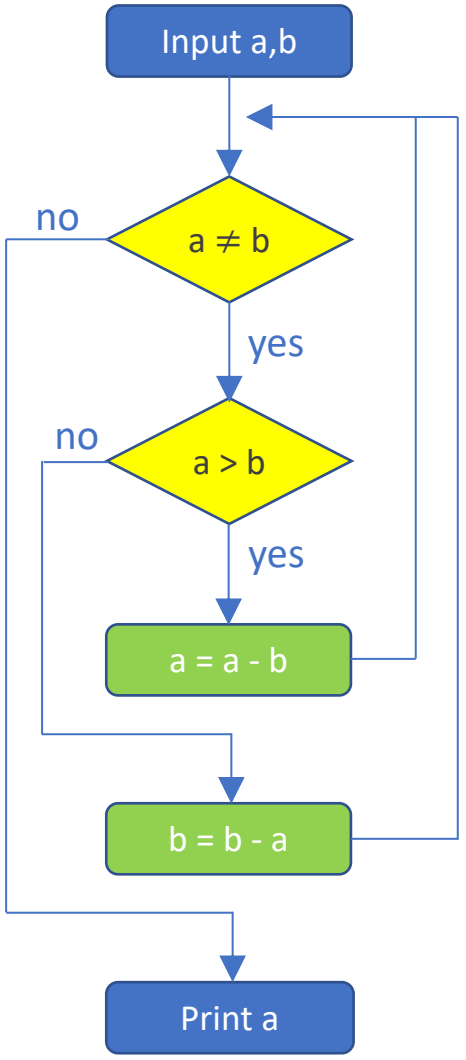
print(a)
```

Python Shell

```
>>>#Running ggd.py
>>>from ggd import *
a= 48
b= 36
12
>>>
```

Algoritme van Euclides





ggd.py

```
a=int(input("a= "))
b=int(input("b= "))

while a!=b:
    if a>b:
        a=a-b
    else:
        b=b-a

print(a)
```

Python Shell

```
>>>#Running ggd.py
>>>from ggd import *
a= 48
b= 36
12
>>>
```

ggd.py

```
a=int(input("a= "))
b=int(input("b= "))

x,y = a,b

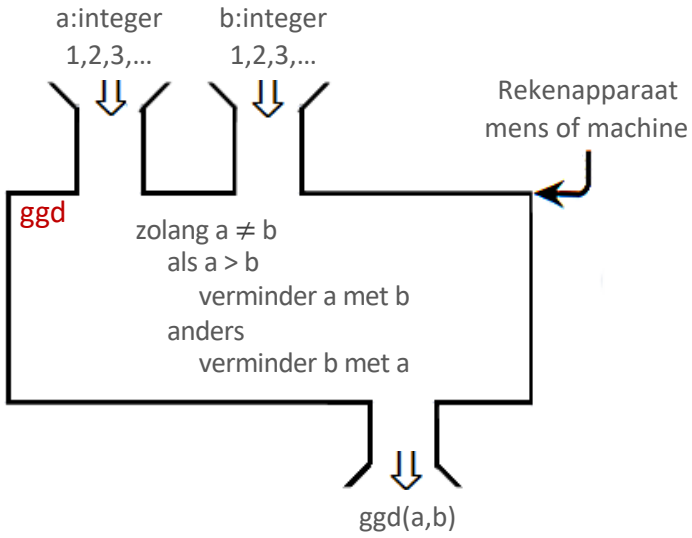
while x!=y:
    if x>y:
        x=x-y
    else:
        y=y-x

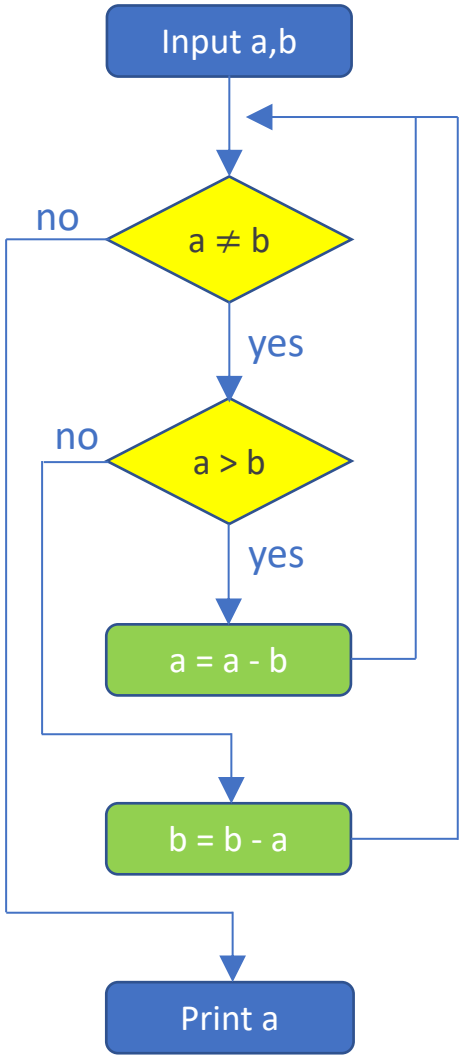
print("De grootste gemene deler van {} en {} = {}".format(a,b,x))
```

Python Shell

```
>>>#Running ggd.py
>>>from ggd import *
a= 48
b= 36
De grootste gemene deler van 48 en 36 = 12
>>>
```

Algoritme van Euclides





ggd.py

```
a=int(input("a= "))
b=int(input("b= "))

while a!=b:
    if a>b:
        a=a-b
    else:
        b=b-a

print(a)
```

Python Shell

```
>>>#Running ggd.py
>>>from ggd import *
a= 48
b= 36
12
>>>
```

ggd.py

```
def ggd(a,b):
    while a!=b:
        if a>b:
            a=a-b
        else:
            b=b-a
    return a
```

Python Shell

```
>>>#Running ggd.py
>>>from ggd import *
>>>ggd(48,36)
12
>>>
```

ggd.py

```
a=int(input("a= "))
b=int(input("b= "))

x,y = a,b

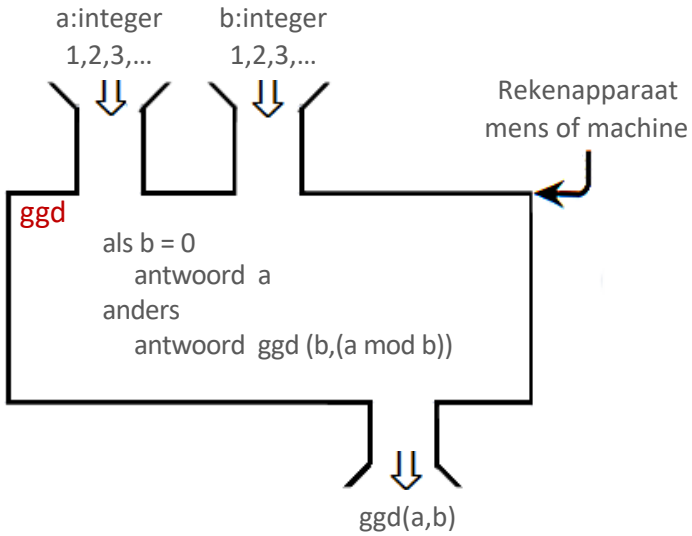
while x!=y:
    if x>y:
        x=x-y
    else:
        y=y-x

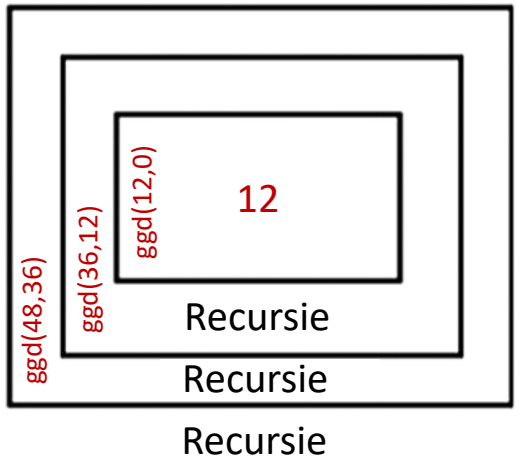
print("De grootste gemene deler van {} en {} = {}".format(a,b,x))
```

Python Shell

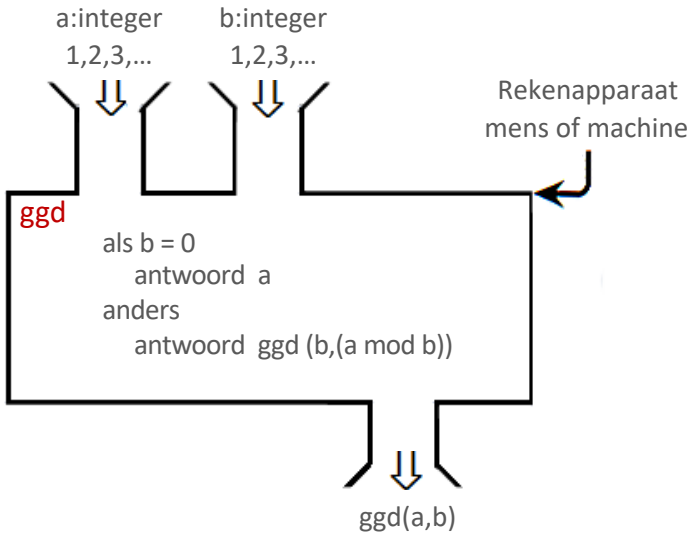
```
>>>#Running ggd.py
>>>from ggd import *
a= 48
b= 36
De grootste gemene deler van 48 en 36 = 12
>>>
```

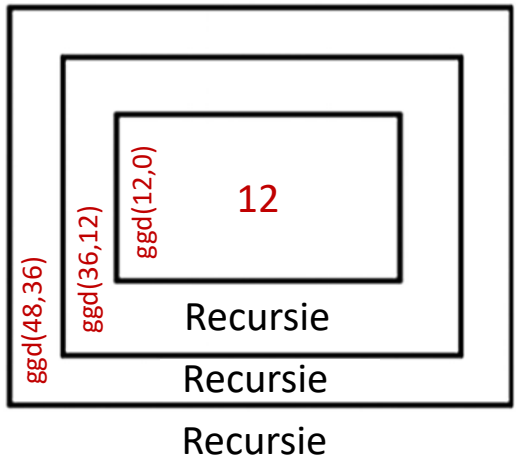
Algoritme van Euclides





Algoritme van Euclides





ggd.py

```
a=int(input("a= "))
b=int (input("b= " ))

def ggd(p,q):
    if q==0:
        return p
    else:
        return ggd(q,p%q)

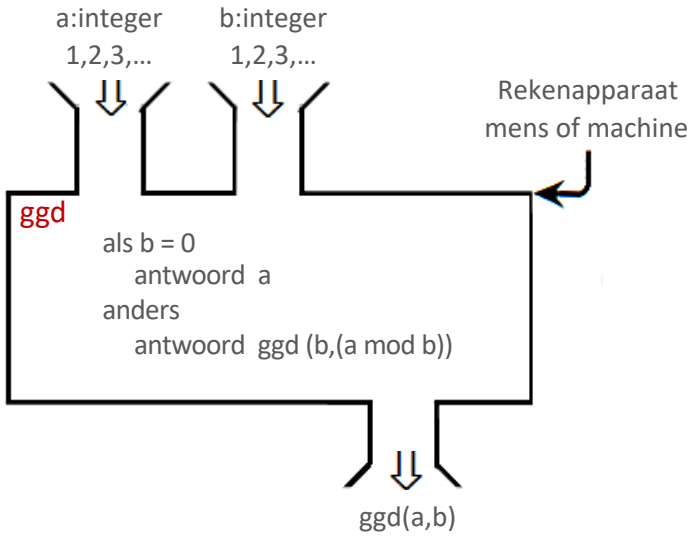
c=ggd(a,b)

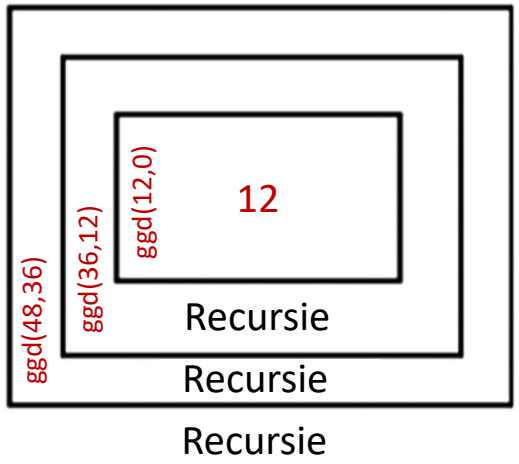
print("De grootste gemene deler van {} en {} = {}".format(a,b,c))
```

Python Shell

```
>>>#Running ggd.py
>>>from ggd import *
a= 48
b= 36
De grootste gemene deler van 48 en 36 = 12
>>>
```

Algoritme van Euclides





ggd.py

```
a=int(input("a= "))
b=int (input("b= " ))

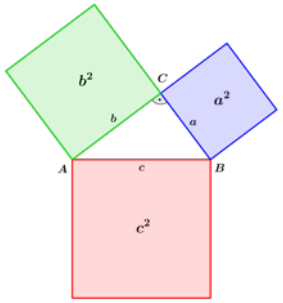
def ggd(p,q):
    if q==0:
        return p
    else:
        return ggd(q,p%q)

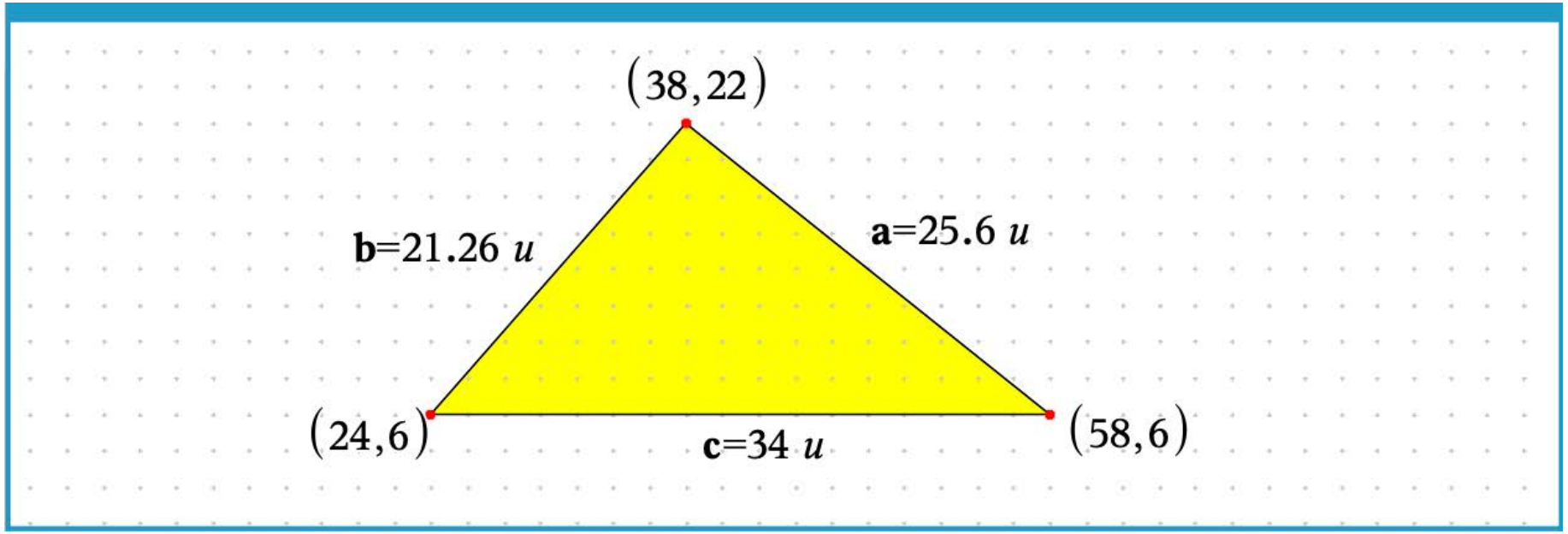
print("De grootste gemene deler van {} en {} = {}".format(a,b,ggd(a,b)))
```

Python Shell

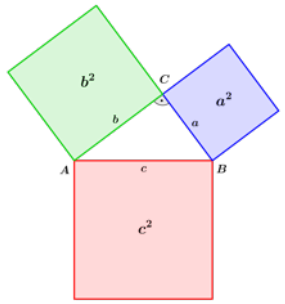
```
>>>#Running ggd.py
>>>from ggd import *
a= 48
b= 36
De grootste gemene deler van 48 en 36 = 12
>>>
```

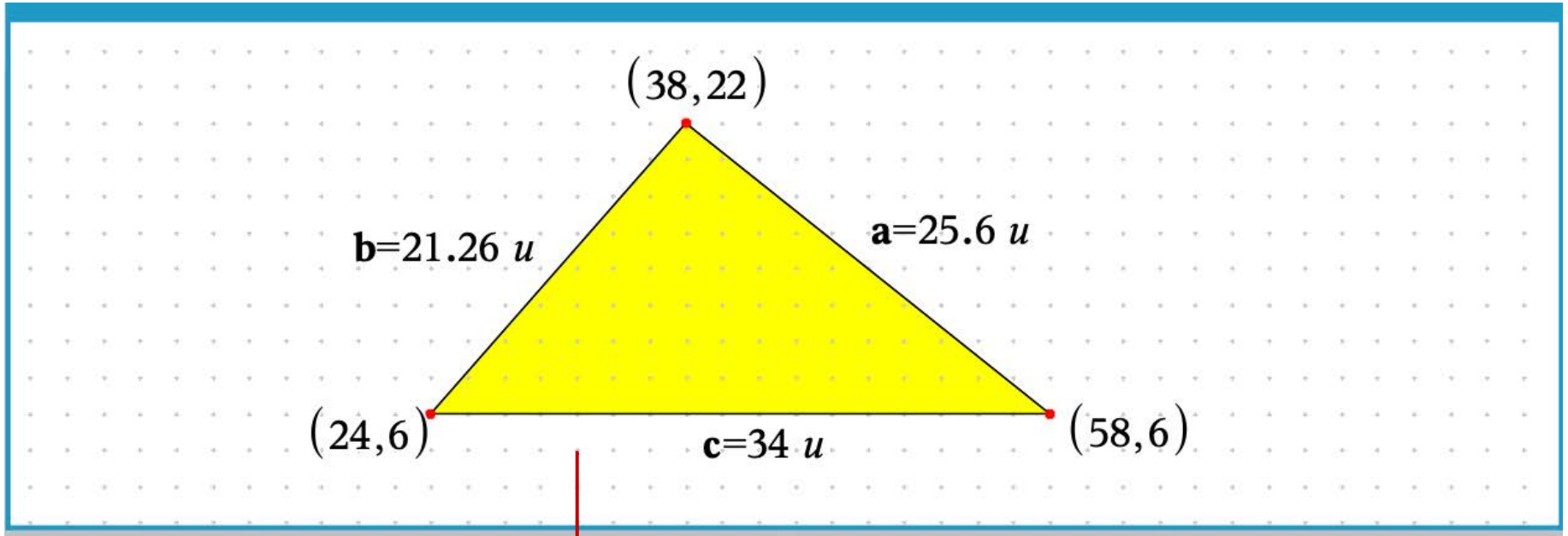
Pythogoras





Pythogoras





Pythagoras.py 14/14

```
from ti_system import *

L=[0]*3
L[0]=recall_value("a")
L[1]=recall_value("b")
L[2]=recall_value("c")
L.sort()

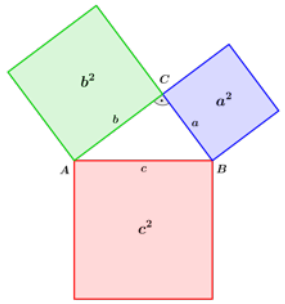
a=round(float(L[0]),2)
b=round(float(L[1]),2)
c=round(float(L[2]),2)
print("De driehoek ∠ {} , {} , {} ".format(a,b,c))
```

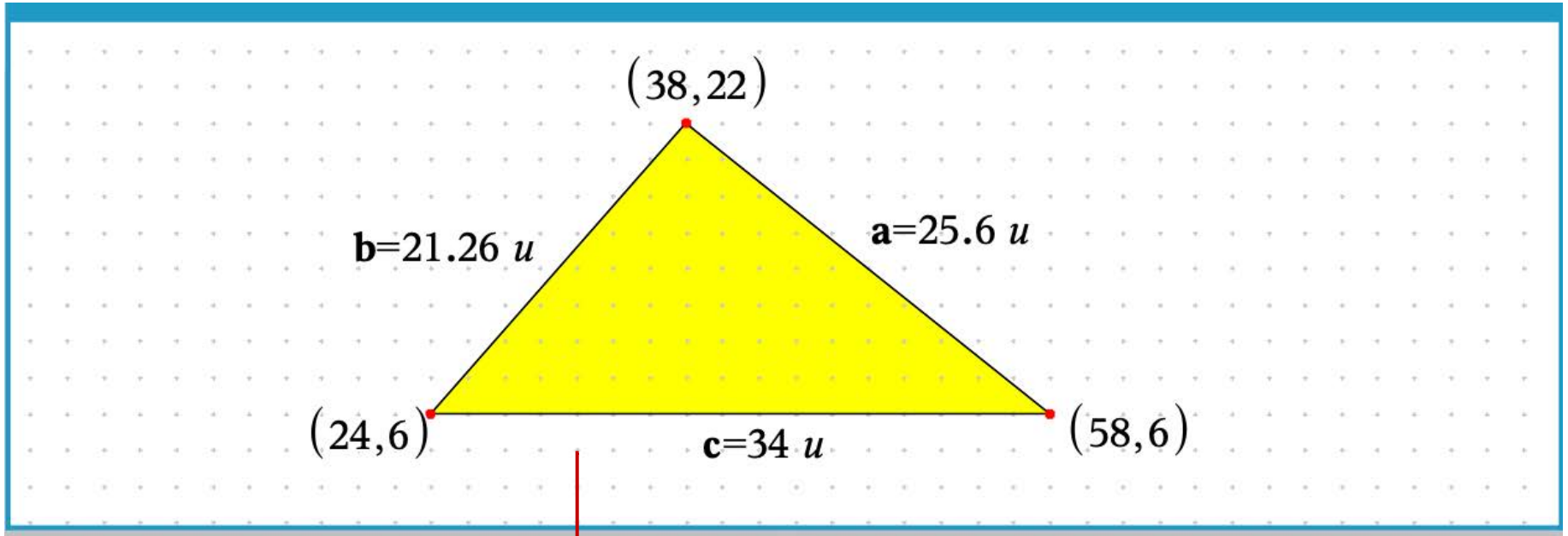
Variabelen uitwisselen

Python Shell 4/4

```
>>>
```

Pythogoras





Pythagoras.py 14/14

```
from ti_system import *

L=[0]*3
L[0]=recall_value("a")
L[1]=recall_value("b")
L[2]=recall_value("c")
L.sort()

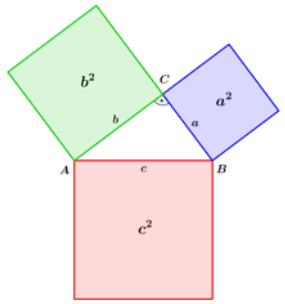
a=round(float(L[0]),2)
b=round(float(L[1]),2)
c=round(float(L[2]),2)
print("De driehoek ∠ {} , {} , {} ".format(a,b,c))
```

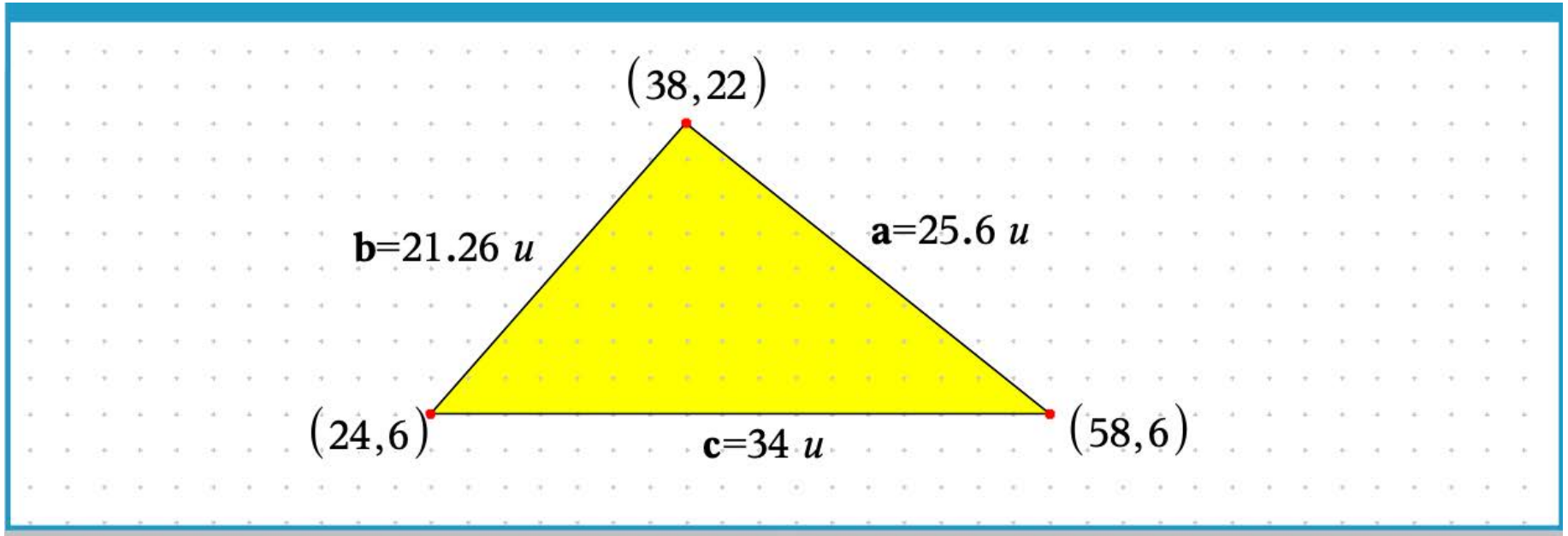
Variabelen uitwisselen

Python Shell 4/4

```
>>>#Running Pythagoras.py
>>>from Pythagoras import *
De driehoek ∠ 21.26 , 25.61 , 34.0
>>>
```

Pythogoras





Pythagoras.py 16/16

```
from ti_system import *

L=[0]*3
L[0]=recall_value("a");L[1]=recall_value("b");L[2]=recall_value("c")
L.sort()

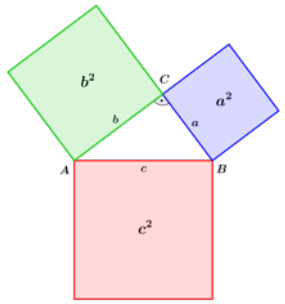
a=round(float(L[0]),2)
b=round(float(L[1]),2)
c=round(float(L[2]),2)
print("De driehoek ∠ {} , {} , {} ".format(a,b,c))

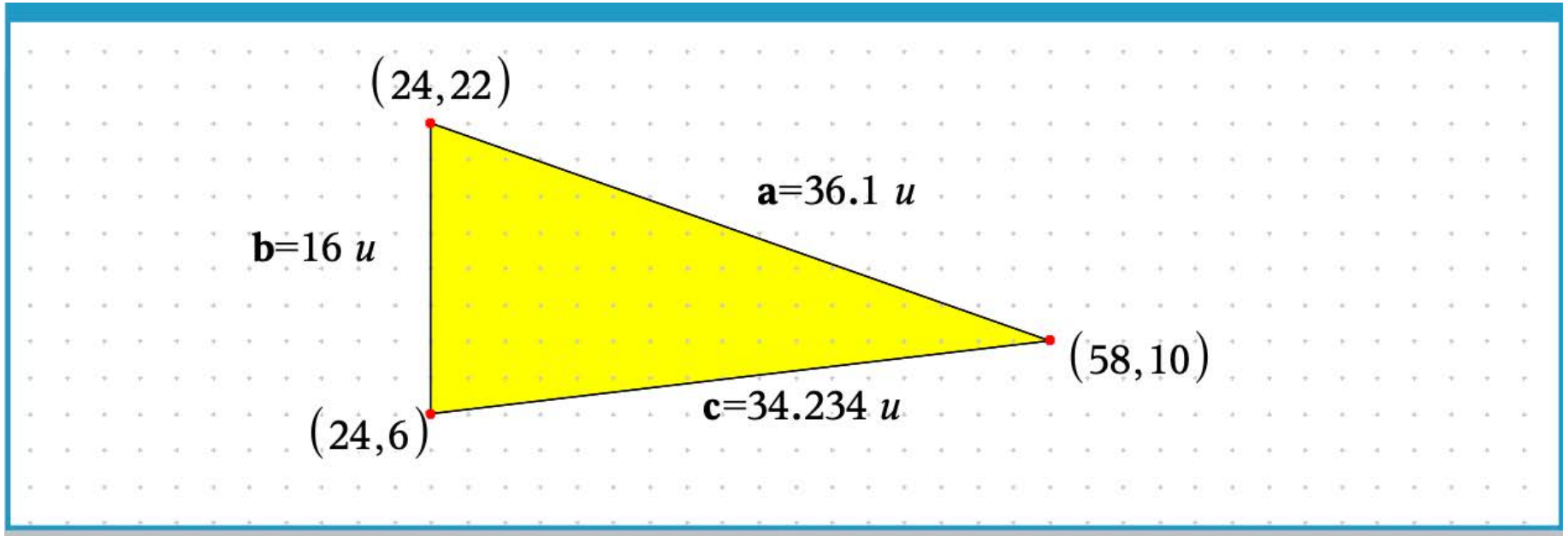
if round(L[2]**2/(L[0]**2+L[1]**2),4)==1:
    print("is een rechthoekige driehoek")
else:
    print("is geen rechthoekige driehoek")
```

Python Shell 8/8

```
>>>#Running Pythagoras.py
>>>from Pythagoras import *
De driehoek ∠ 21.26 , 25.61 , 34.0
>>>#Running Pythagoras.py
>>>from Pythagoras import *
De driehoek ∠ 21.26 , 25.61 , 34.0
is geen rechthoekige driehoek
>>>
```

Pythogoras





Pythagoras.py 11/16

```
from ti_system import *

L=[0]*3
L[0]=recall_value("a");L[1]=recall_value("b");L[2]=recall_value("c")
L.sort()

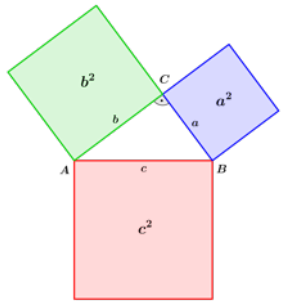
a=round(float(L[0]),2)
b=round(float(L[1]),2)
c=round(float(L[2]),2)
print("De driehoek ∠ {} , {} , {} ".format(a,b,c))

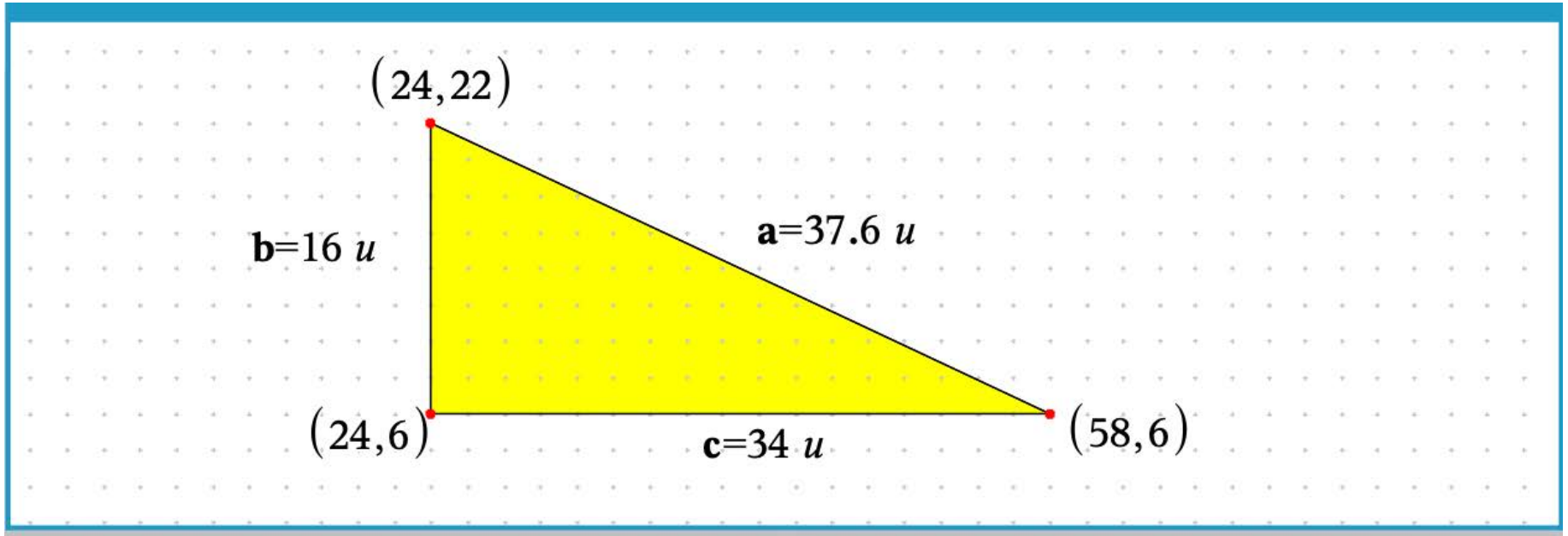
if round(L[2]**2/(L[0]**2+L[1]**2),4)==1:
    print("is een rechthoekige driehoek")
else:
    print("is geen rechthoekige driehoek")
```

Python Shell 12/12

```
>>>#Running Pythagoras.py
>>>from Pythagoras import *
De driehoek ∠ 21.26 , 25.61 , 34.0
>>>#Running Pythagoras.py
>>>from Pythagoras import *
De driehoek ∠ 21.26 , 25.61 , 34.0
is geen rechthoekige driehoek
>>>#Running Pythagoras.py
>>>from Pythagoras import *
De driehoek ∠ 16.0 , 34.23 , 36.06
is geen rechthoekige driehoek
>>>
```

Pythogoras





Pythagoras.py 13/16

```
from ti_system import *

L=[0]*3
L[0]=recall_value("a");L[1]=recall_value("b");L[2]=recall_value("c")
L.sort()

a=round(float(L[0]),2)
b=round(float(L[1]),2)
c=round(float(L[2]),2)
print("De driehoek ∠ {} , {} , {} ".format(a,b,c))

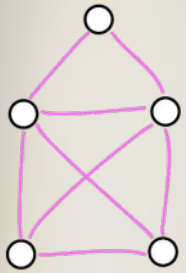
if round(L[2]**2/(L[0]**2+L[1]**2),4)==1:
    print("is een rechthoekige driehoek")
else:
    print("is geen rechthoekige driehoek")
```

Python Shell 16/16

```
>>>#Running Pythagoras.py
>>>from Pythagoras import *
De driehoek ∠ 21.26 , 25.61 , 34.0
>>>#Running Pythagoras.py
>>>from Pythagoras import *
De driehoek ∠ 21.26 , 25.61 , 34.0
is geen rechthoekige driehoek
>>>#Running Pythagoras.py
>>>from Pythagoras import *
De driehoek ∠ 16.0 , 34.23 , 36.06
is geen rechthoekige driehoek
>>>#Running Pythagoras.py
>>>from Pythagoras import *
De driehoek ∠ 16.0 , 34.0 , 37.58
is een rechthoekige driehoek
>>>
```

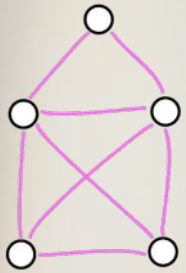
Problem Solving

Meten van hoeken



Problem Solving

Meten van hoeken



1.1 1.2 Rover RAD

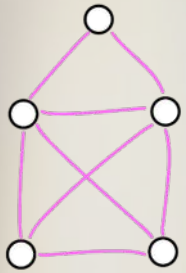
rover.py 8/11

```
import ti_rover as rv

rv.forward(3)
```

Problem Solving

Meten van hoeken



1.1 1.2 Rover RAD

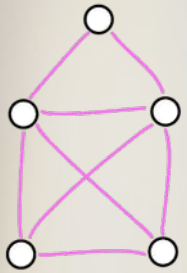
rover.py 8/8

```
import ti_rover as rv

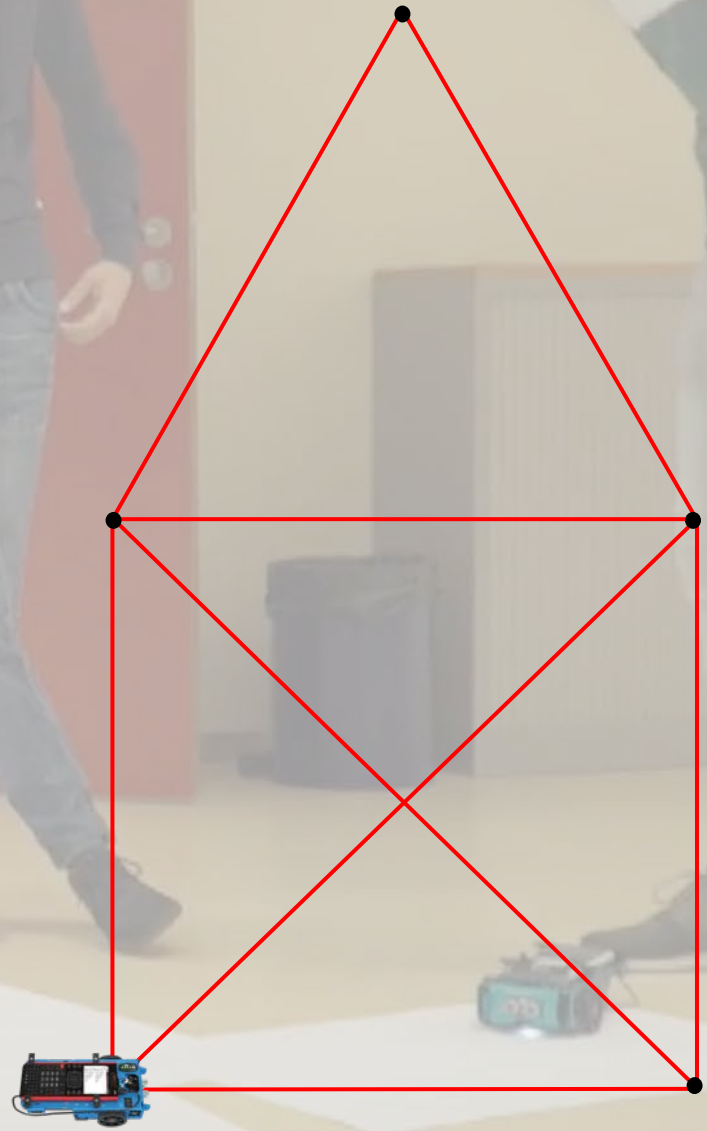
for i in range(4):
    rv.forward(3)
    rv.right(90)
```

Problem Solving

Meten van hoeken

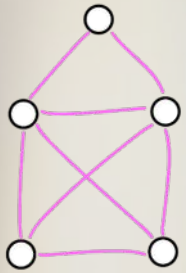


```
import ti_rover as rv
rv.forward(a)
rv.left(90)          rv.right(120)
rv.forward(a)        rv.forward(a)
rv.left(90)          rv.right(75)
rv.forward(a)        rv.forward(sqrt(2)*a)
rv.right(120)        rv.right(135)
rv.forward(a)        rv.forward(sqrt(2)*a)
```



Problem Solving

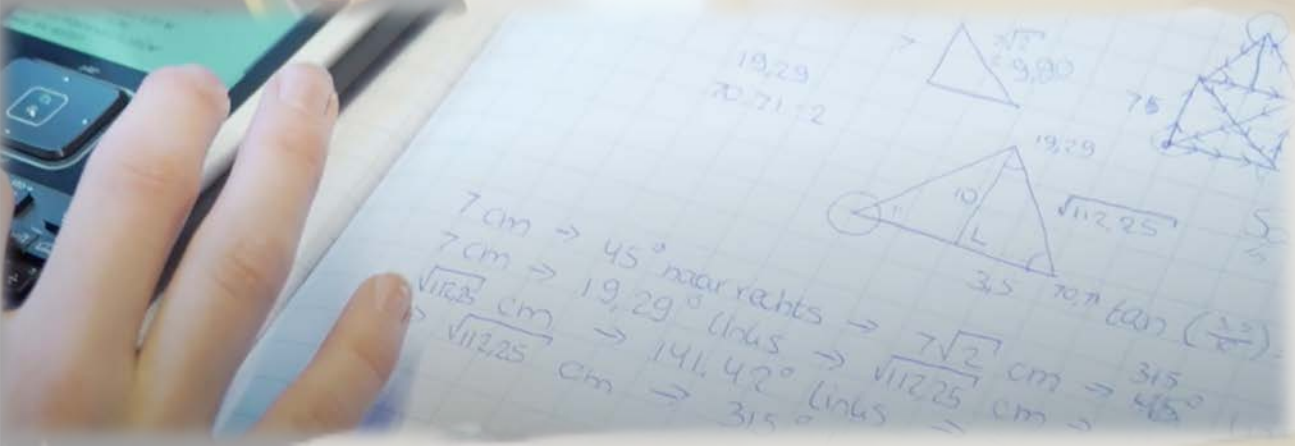
Meten van hoeken

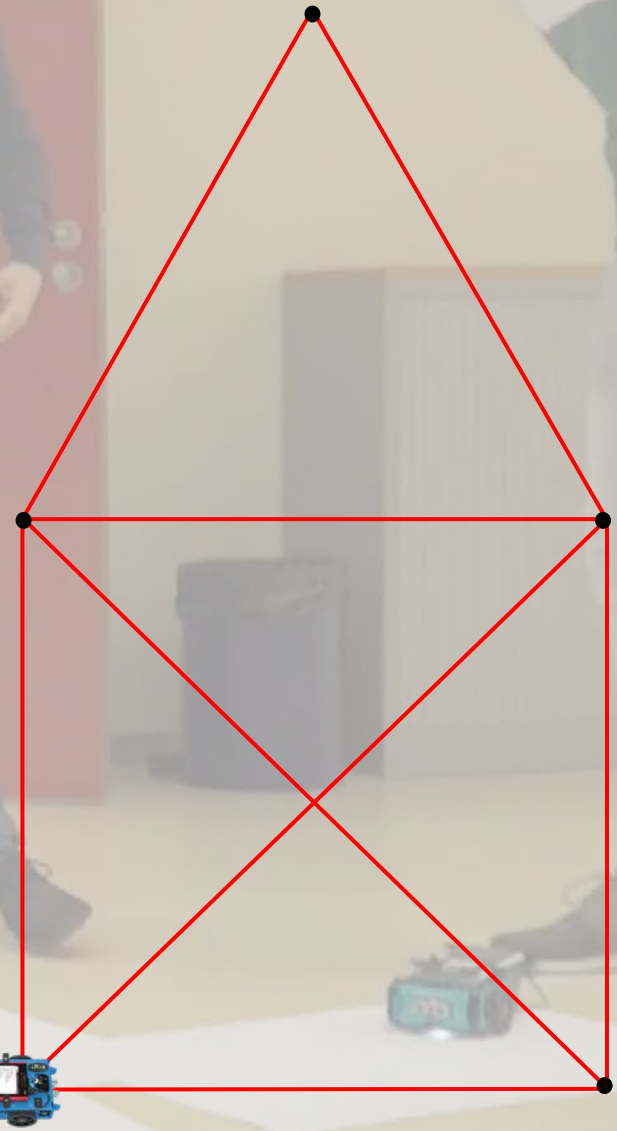


Voor veel leerlingen is het een stimulans om te leren programmeren. De directe respons vinden ze leuk, het geeft een gevoel van beloning. Ze zijn heel fanatiek bezig met deze wiskundige activiteit, fanatieker dan met het maken van wiskundesommetjes!

Bert Wikkerink
Docent Wiskunde
CSG Ludger Drachten

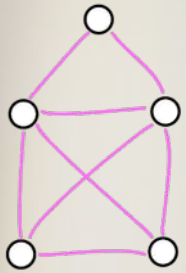
www.t3nederland.nl/nl/t3-europe/edublogs





Problem Solving

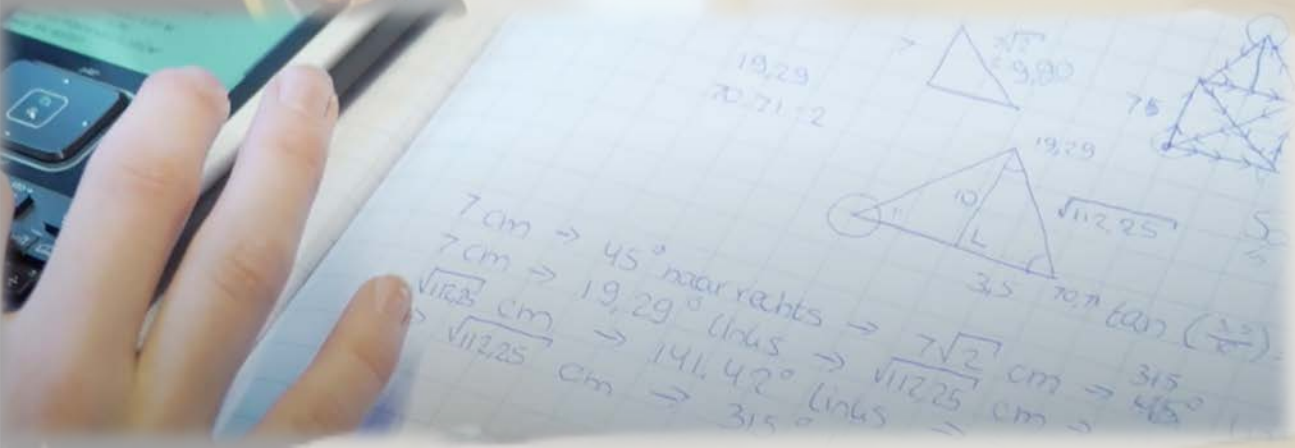
Meten van hoeken

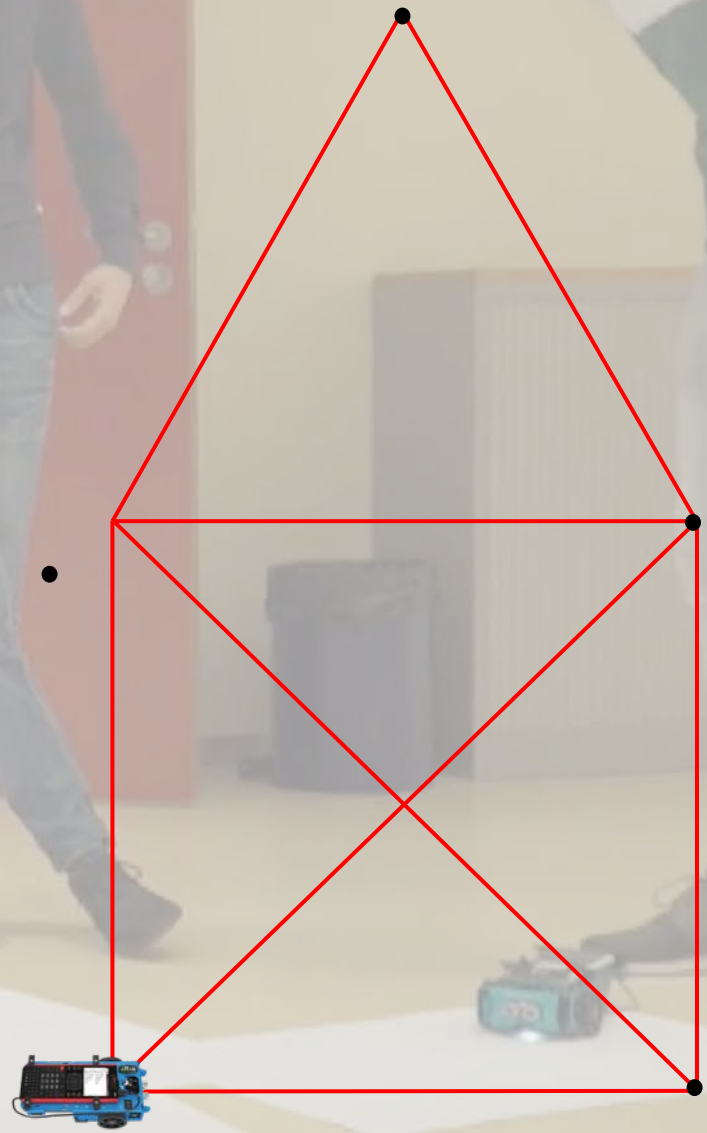


Deze projecten sluiten mooi aan bij het wiskundeonderwijs. Leerlingen vragen wel eens wat is wiskunde en waar heb ik dat voor nodig? Nou hier zie je een hele mooie verbinding tussen wiskunde en de werkelijkheid, waardoor het leereffect groter is."

Ina Everts
Locatiedirecteur

www.t3nederland.nl/nl/t3-europe/edublogs





Problem Solving

Eenparig rechtlijnige beweging

```
import ti_rover as rv
a=rv.ranger_measurement()
alist=[]
rv.forward(100)
while a>0.1:
    a=rv.ranger_measurement()
    alist=alist+a
```

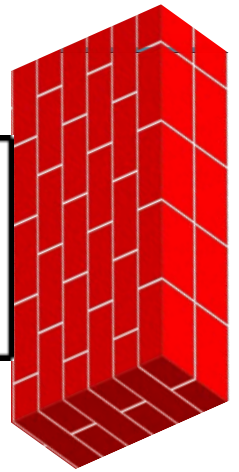
→ alist (python)

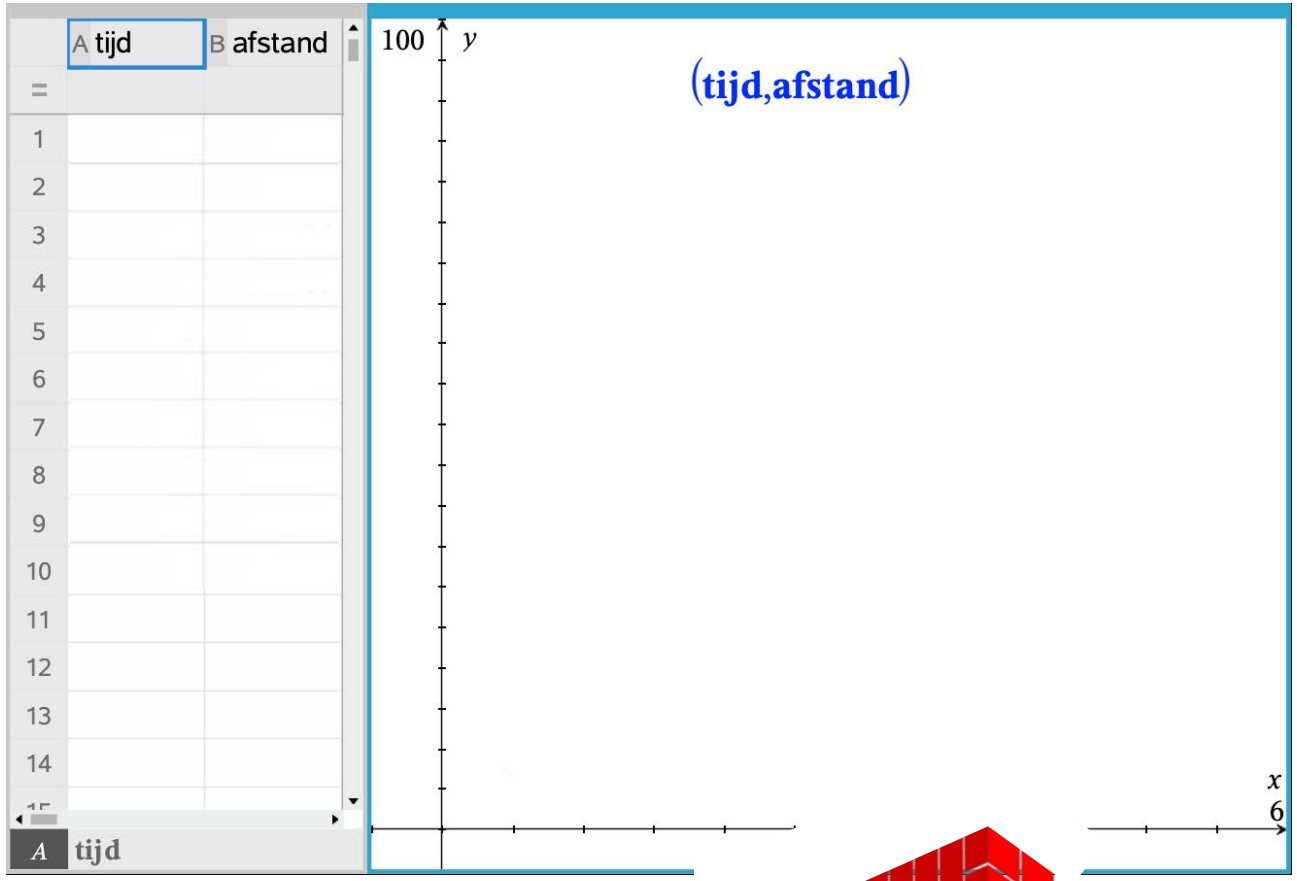
```
from ti_system import *
store_list("afstand",alist)
```

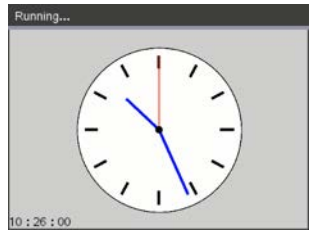
TI-Nspire var Python var

```
from time import *
t=localtime()          (2020, 6, 20, 12, 34, 49, 5, 172, 1)
h=localtime()[3]       12
m=localtime()[4]       34
s=localtime()[5]       49
```

→ tijd (TI-Nspire)







Problem Solving

Eenparig rechtlijnige beweging

```
import ti_rover as rv
a=rv.ranger_measurement()
alist=[]
rv.forward(100)
while a>0.1:
    a=rv.ranger_measurement()
    alist=alist+a
```

→ alist (python)

```
from ti_system import *
store_list("afstand",alist)
```

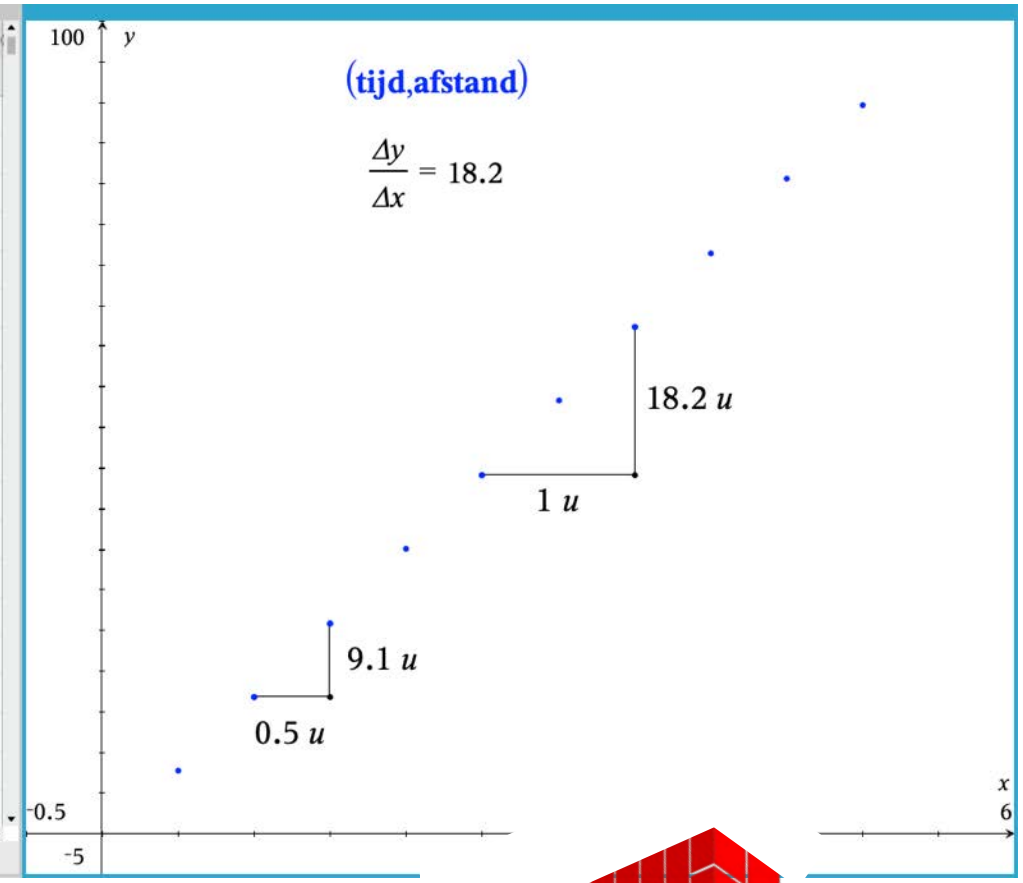
TI-Nspire var Python var

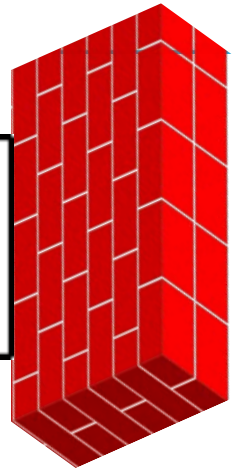
```
from time import *
t=localtime()          (2020, 6, 20, 12, 34, 49, 5, 172, 1)
h=localtime()[3]       12
m=localtime()[4]       34
s=localtime()[5]       49
```

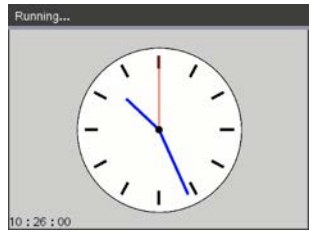
→ tijd (TI-Nspire)

	A tijd	B afstand
1	0.5	7.73
2	1	16.84
3	1.5	25.94
4	2	35.05
5	2.5	44.15
6	3	53.26
7	3.5	62.36
8	4	71.47
9	4.5	80.57
10	5	89.68
11		
12		
13		
14		
15		
16		
17		
18		
19		

A tijd







Problem Solving

Eenparig rechtlijnige beweging

```
import ti_rover as rv
a=rv.ranger_measurement()
alist=[]
rv.forward(100)
while a>0.1:
    a=rv.ranger_measurement()
    alist=alist+a
```

→ alist (python)

```
from ti_system import *
store_list("afstand",alist)
```

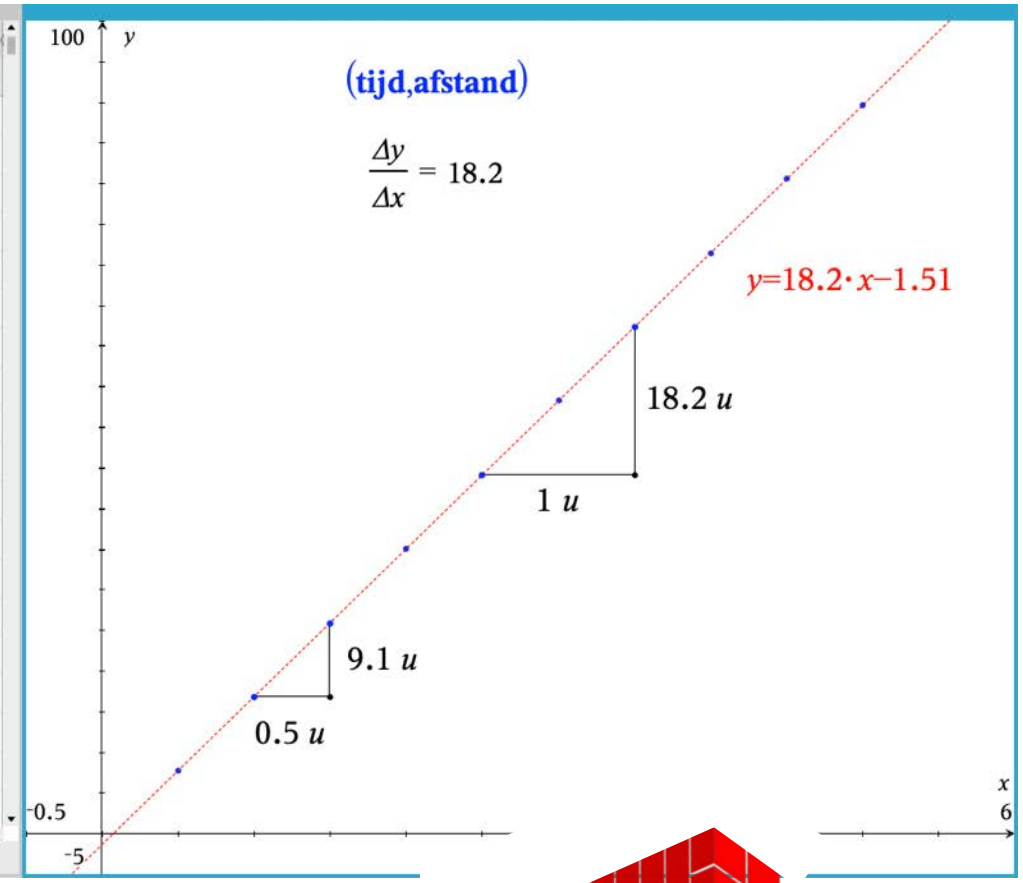
TI-Nspire var Python var

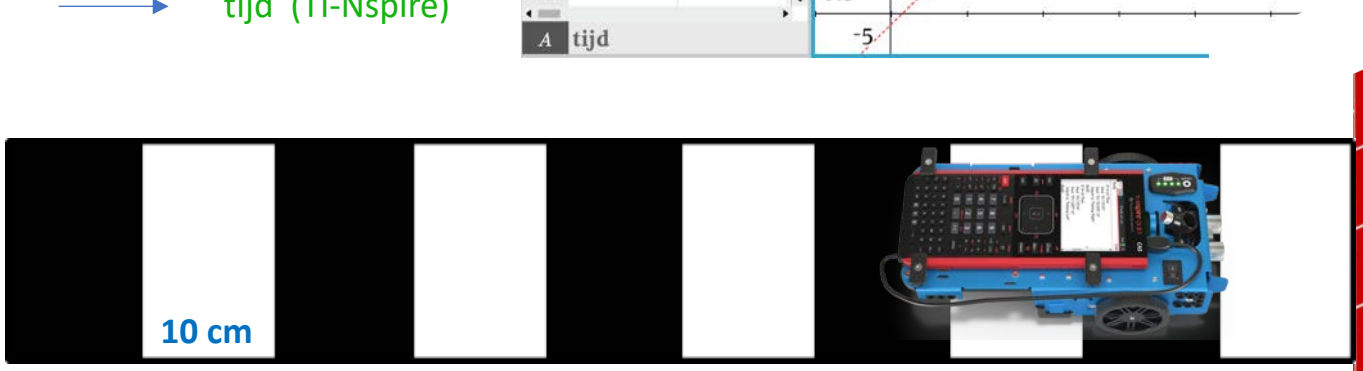
```
from time import *
t=localtime()          (2020, 6, 20, 12, 34, 49, 5, 172, 1)
h=localtime()[3]       12
m=localtime()[4]       34
s=localtime()[5]       49
```

→ tijd (TI-Nspire)

	A tijd	B afstand
1	0.5	7.73
2	1	16.84
3	1.5	25.94
4	2	35.05
5	2.5	44.15
6	3	53.26
7	3.5	62.36
8	4	71.47
9	4.5	80.57
10	5	89.68
11		
12		
13		
14		
15		
16		
17		
18		
19		

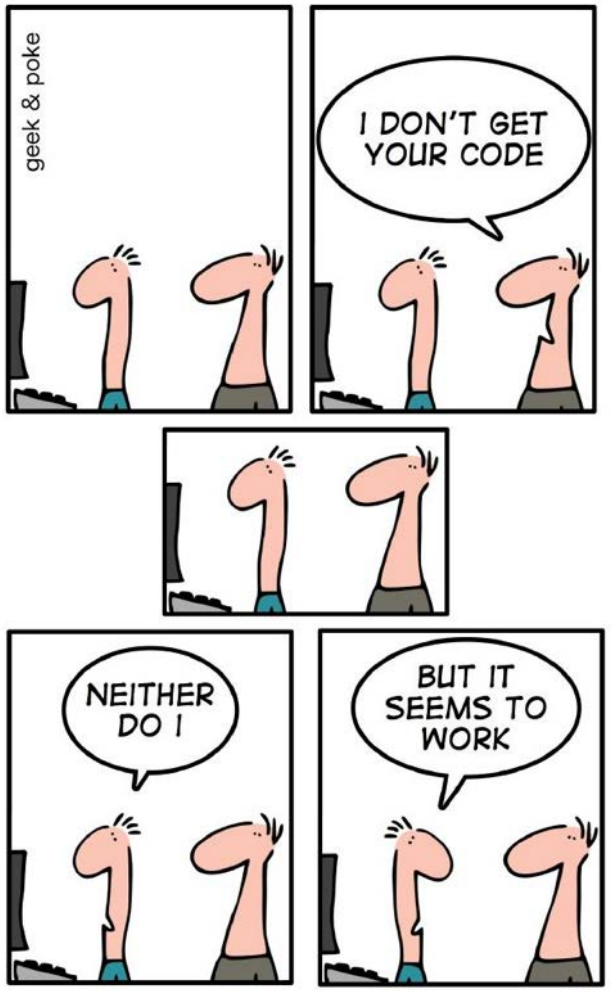
A tijd





Programmeren

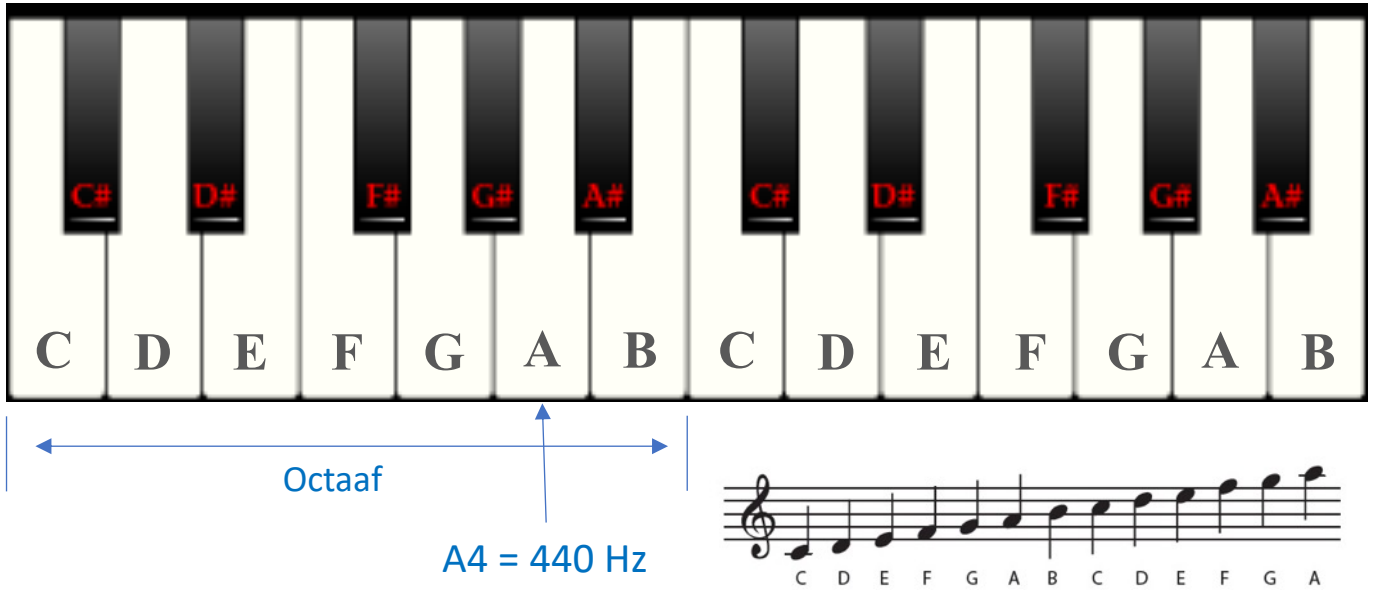
- Programmeren motiveert leerlingen abstracte concepten te onderzoeken!
- Programmeren stimuleert leerlingen tot het creatief oplossen van problemen!
- Programmeren geeft leerlingen een beter beeld van de hun omringde digitale wereld!



STEM PROJECTEN

Geluid

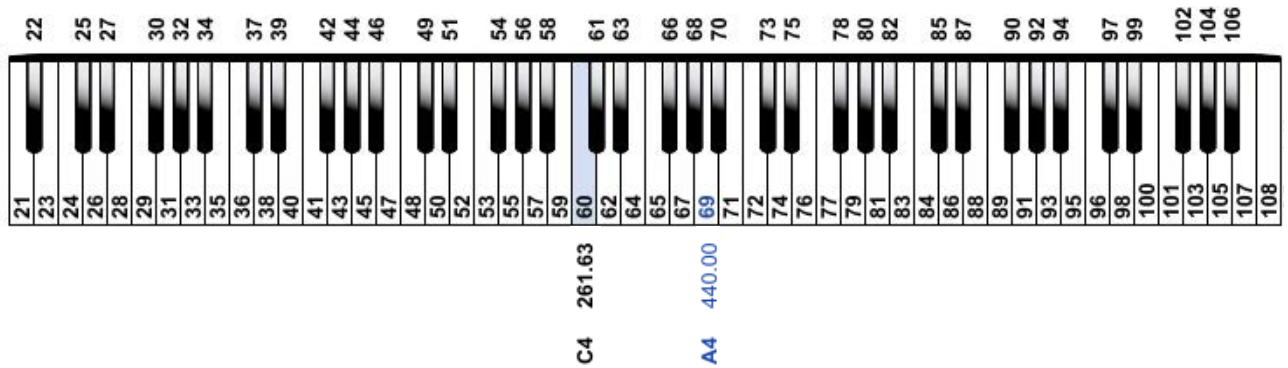
Voor ons oor is de afstand tussen 220 Hz en 440 Hz gelijk aan de afstand tussen 110 Hz en 220 Hz, een octaaf.



Het MIDI-getal (Musical Instrument Digital Interface) voor A4 is 69.

Verband frequentie en MIDI-getallen

$$p = 69 + \log_2 \frac{f}{440}.\text{-}2$$

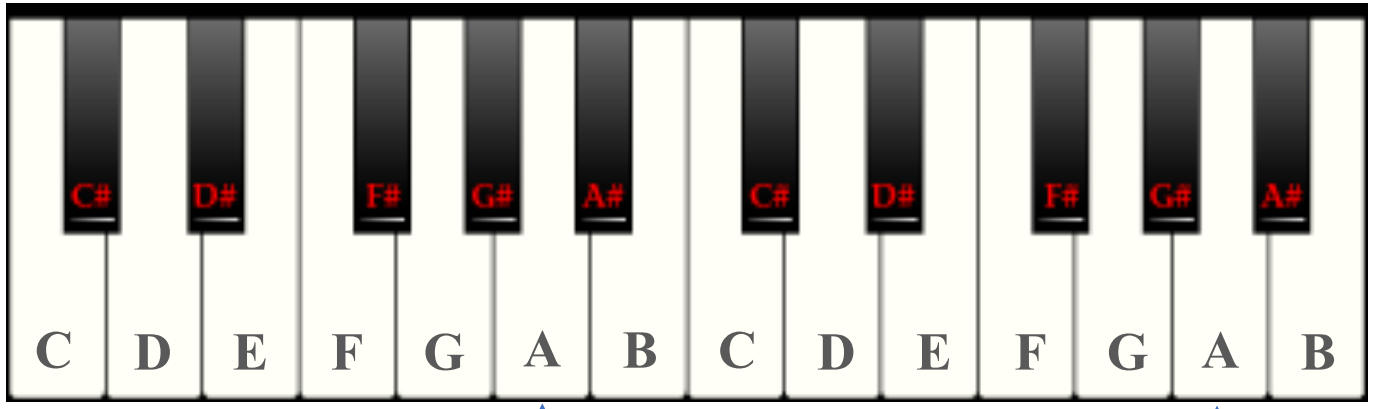


Frequentie vs toonhoogte

toon	0	1	2	3	4	5	6	7	8	9
C	16,35	32,7	65,4	130,8	261,6	523,2	1046	2093	4186	8372
C#	17,32	34,64	69,29	138,5	277,1	554,3	1108	2217	4434	8869
D	18,35	36,7	73,41	146,8	293,6	587,3	1174	2349	4698	9397
D#	19,44	38,89	77,78	155,5	311,1	622,2	1244	2489	4978	9956
E	20,6	41,2	82,4	164,8	329,6	659,2	1318	2637	5274	10548
F	21,82	43,65	87,3	174,6	349,2	698,4	1396	2793	5587	11175
F#	23,12	46,24	92,49	184,9	369,9	739,9	1479	2959	5919	11839
G	24,49	48,99	97,99	195,9	391,9	783,9	1567	3135	6271	12543
G#	25,95	51,91	103,8	207,6	415,3	830,6	1661	3322	6644	13289
A	27,5	55	110	220	440	880	1760	3520	7040	14080
A#	29,13	58,27	116,5	233	466,1	932,3	1864	3729	7458	14917
B	30,86	61,73	123,4	246,9	493,8	987,7	1975	3951	7902	15804

STEM PROJECTEN

Geluid



De toonafstand tussen naastliggende tonen is even groot.

Deze verhouding is gelijk aan $2^{\frac{1}{12}} = 1{,}05946309$.

- A#4 is 1 semitoon hoger dan A4 – A#4 = $440 \cdot 2^{\frac{1}{12}}$ = 466.1 Hz
- B4 is 2 semitonen hoger dan A4 – B4 = $440 \cdot 2^{\frac{2}{12}} = 493.8$ Hz
- A5 is 12 semitonen hoger dan A4 – A5 = $440 \cdot 2^{\frac{12}{12}}$ = 880 Hz

Frequentie vs toonhoogte

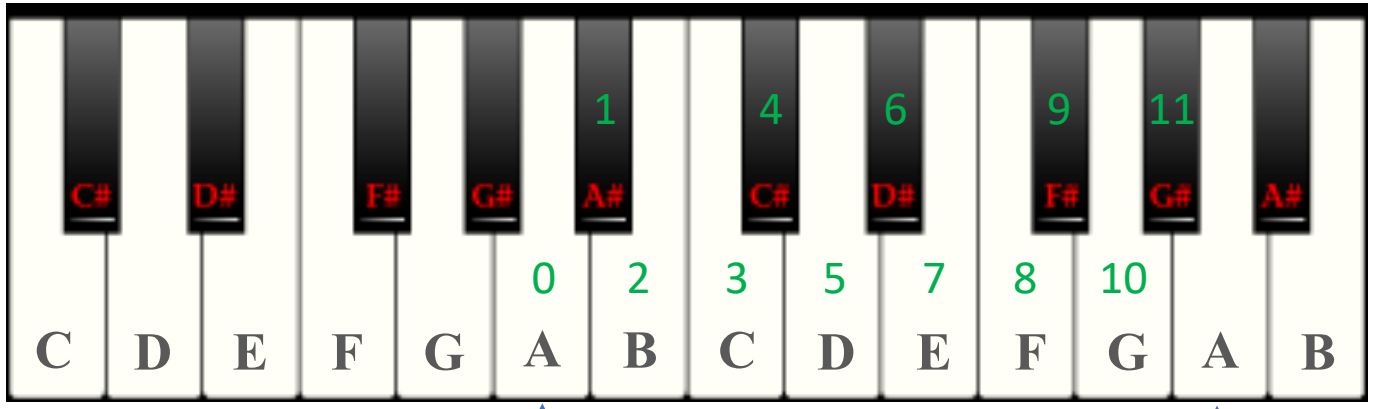
De toonafstand tussen naastliggende tonen is even groot.

Deze verhouding is gelijk aan $2^{\frac{1}{12}} = 1{,}05946309$.

- A#4 is 1 semitoon hoger dan A4 – A#4 = $440 \cdot 2^{\frac{1}{12}}$ = 466.1 Hz
- B4 is 2 semitonen hoger dan A4 – B4 = $440 \cdot 2^{\frac{2}{12}} = 493.8$ Hz
- A5 is 12 semitonen hoger dan A4 – A5 = $440 \cdot 2^{\frac{12}{12}}$ = 880 Hz

STEM PROJECTEN

Geluid



De toonafstand tussen naastliggende tonen is even groot.

Deze verhouding is gelijk aan $2^{\frac{1}{12}} = 1{,}05946309$.

- A#4 is 1 semitoon hoger dan A4 – A#4 = $440 \cdot 2^{\frac{1}{12}}$ = 466.1 Hz
- B4 is 2 semitonen hoger dan A4 – B4 = $440 \cdot 2^{\frac{2}{12}} = 493.8$ Hz
- A5 is 12 semitonen hoger dan A4 – A5 = $440 \cdot 2^{\frac{12}{12}}$ = 880 Hz

$$f = 440 \cdot 2^{\frac{n}{12}}$$

	n	f
A	0	440.
A#	1	466.164
B	2	493.883
C	3	523.251
C#	4	554.365
D	5	587.33
D#	6	622.254
E	7	659.255
F	8	698.456
F#	9	739.989
G	10	783.991
G#	11	830.609

Frequentie vs toonhoogte

De toonafstand tussen naastliggende tonen is even groot.

Deze verhouding is gelijk aan $2^{\frac{1}{12}} = 1{,}05946309$.

- A#4 is 1 semitoon hoger dan A4 – A#4 = $440 \cdot 2^{\frac{1}{12}}$ = 466.1 Hz
- B4 is 2 semitonen hoger dan A4 – B4 = $440 \cdot 2^{\frac{2}{12}}$ = 493.8 Hz
- A5 is 12 semitonen hoger dan A4 – A5 = $440 \cdot 2^{\frac{12}{12}}$ = 880 Hz

STEM PROJECTEN

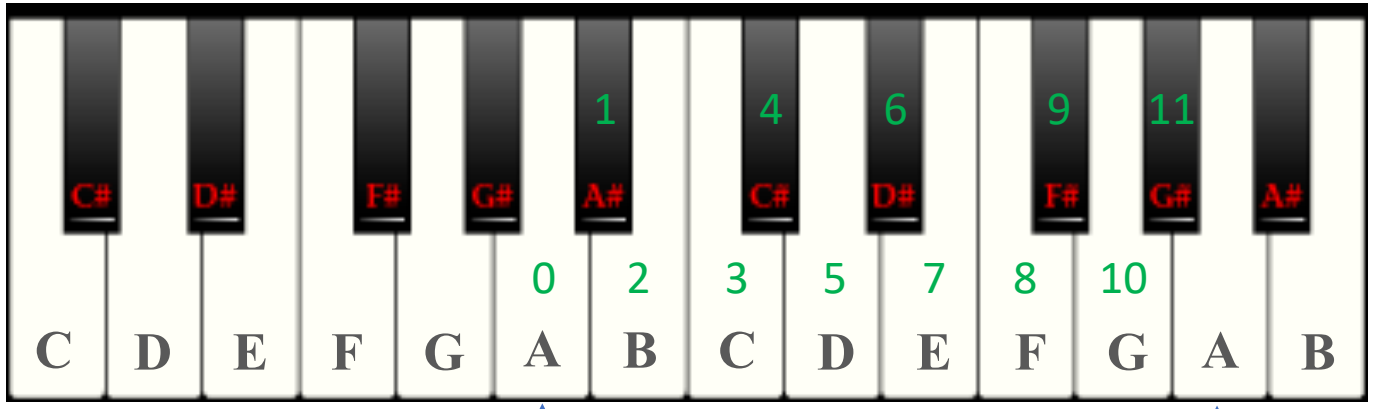
Geluid

geluid.py

```
from ti_hub import *

sound.tone(440,1)
sleep(1.5)

sound.note("A4",1)
sleep(1.5)
```



$$f = 440 \cdot 2^{\frac{n}{12}}$$

A4 = 440 Hz
standaard tuning

x 2

A5 = 880 Hz

Frequentie vs toonhoogte

toon	0	1	2	3	4	5	6	7	8	9
C	16,35	32,7	65,4	130,8	261,6	523,2	1046	2093	4186	8372
C#	17,32	34,64	69,29	138,5	277,1	554,3	1108	2217	4434	8869
D	18,35	36,7	73,41	146,8	293,6	587,3	1174	2349	4698	9397
D#	19,44	38,89	77,78	155,5	311,1	622,2	1244	2489	4978	9956
E	20,6	41,2	82,4	164,8	329,6	659,2	1318	2637	5274	10548
F	21,82	43,65	87,3	174,6	349,2	698,4	1396	2793	5587	11175
F#	23,12	46,24	92,49	184,9	369,9	739,9	1479	2959	5919	11839
G	24,49	48,99	97,99	195,9	391,9	783,9	1567	3135	6271	12543
G#	25,95	51,91	103,8	207,6	415,3	830,6	1661	3322	6644	13289
A	27,5	55	110	220	440	880	1760	3520	7040	14080
A#	29,13	58,27	116,5	233	466,1	932,3	1864	3729	7458	14917
B	30,86	61,73	123,4	246,9	493,8	987,7	1975	3951	7902	15804

STEM PROJECTEN

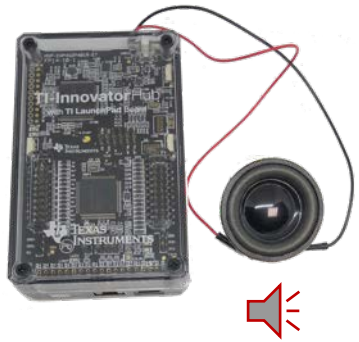
Geluid

geluid.py

```
from ti_hub import *

sound.tone(440,1)
sleep(1.5)

sound.note("A4",1)
sleep(1.5)
```



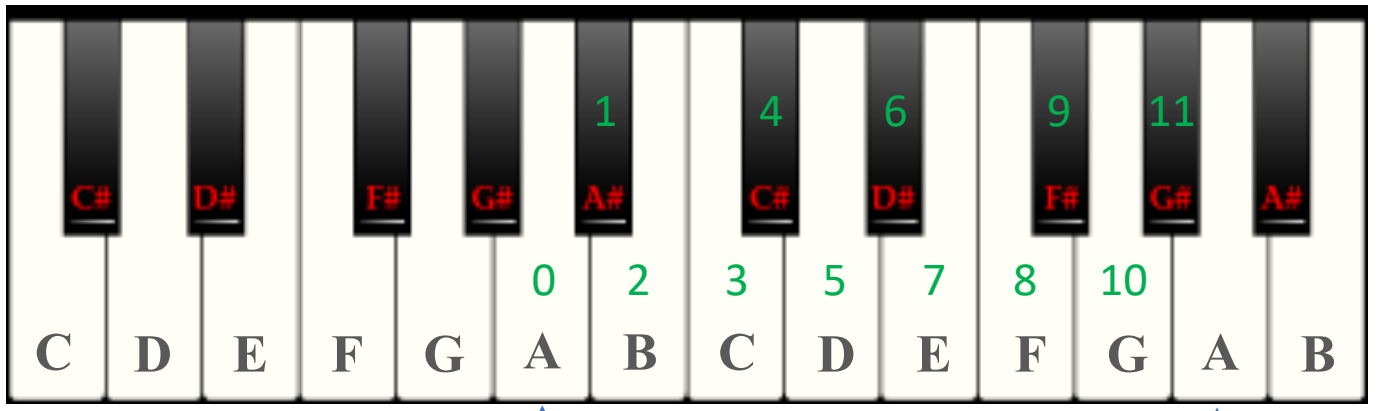
geluid.py

```
from ti_hub import *

vol=speaker("BB 1")

vol.tone(440,1)
sleep(1.5)

vol.note("A4",1)
sleep(1.5)
```



$$f = 440 \cdot 2^{\frac{n}{12}}$$

A4 = 440 Hz
standaard tuning

x 2

A5 = 880 Hz

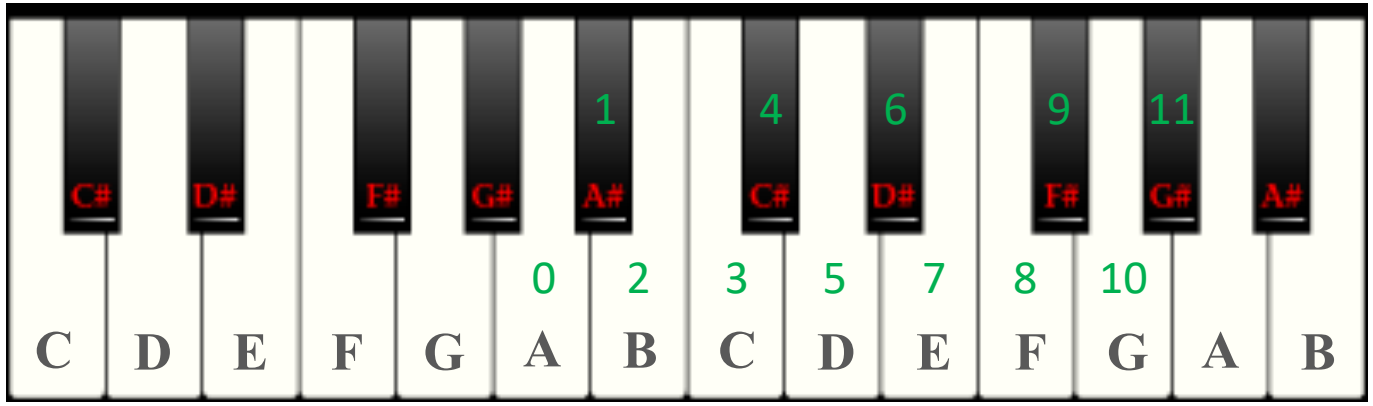
Frequentie vs toonhoogte

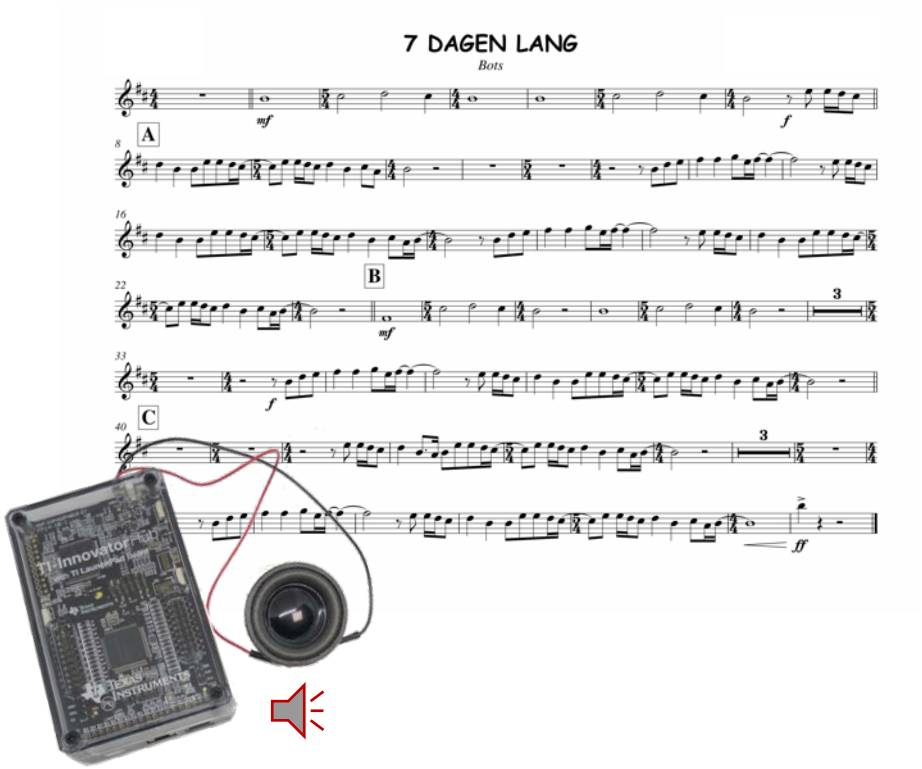
toon	0	1	2	3	4	5	6	7	8	9
C	16,35	32,7	65,4	130,8	261,6	523,2	1046	2093	4186	8372
C#	17,32	34,64	69,29	138,5	277,1	554,3	1108	2217	4434	8869
D	18,35	36,7	73,41	146,8	293,6	587,3	1174	2349	4698	9397
D#	19,44	38,89	77,78	155,5	311,1	622,2	1244	2489	4978	9956
E	20,6	41,2	82,4	164,8	329,6	659,2	1318	2637	5274	10548
F	21,82	43,65	87,3	174,6	349,2	698,4	1396	2793	5587	11175
F#	23,12	46,24	92,49	184,9	369,9	739,9	1479	2959	5919	11839
G	24,49	48,99	97,99	195,9	391,9	783,9	1567	3135	6271	12543
G#	25,95	51,91	103,8	207,6	415,3	830,6	1661	3322	6644	13289
A	27,5	55	110	220	440	880	1760	3520	7040	14080
A#	29,13	58,27	116,5	233	466,1	932,3	1864	3729	7458	14917
B	30,86	61,73	123,4	246,9	493,8	987,7	1975	3951	7902	15804

STEM PROJECTEN

Geluid

$$f = 440 \cdot 2^{\frac{n}{12}}$$





	B	C noot	D tijd
1	0	440.	0.125
2	3	523.251	0.125
3	5	587.33	0.125
4	7	659.255	0.25
5	7	659.255	0.25
6	8	698.456	0.125
7	5	587.33	0.125
8	7	659.255	0.5
9	-200	0.004229	0.125
10	5	587.33	0.125
11	5	587.33	0.0625
12	3	523.251	0.0625
13	2	493.883	0.125
14	3	523.251	0.25
15	0	440.	0.185
16	-2	391.995	0.185

c1 $=440 \cdot 2^{\frac{b1}{12.}}$

muziek.py

```
from ti_system import *
from ti_hub import *

noot=recall_list("noot")
tijd=recall_list("tijd")

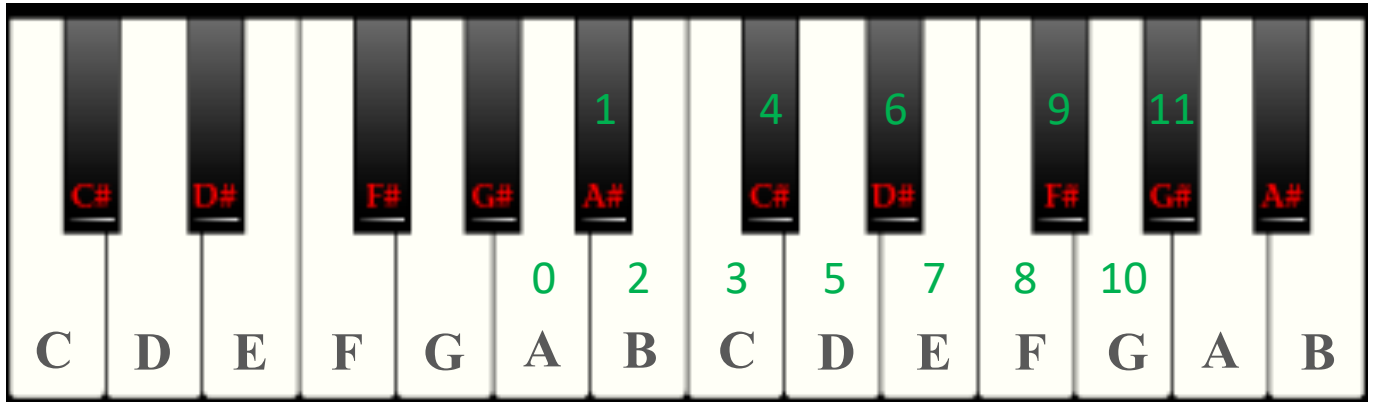
vol=speaker("BB 1")

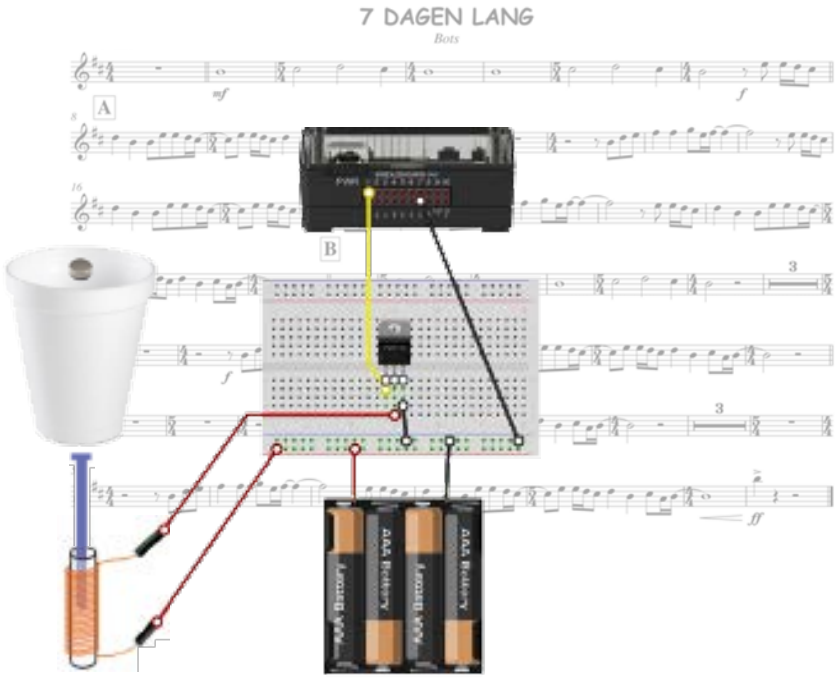
for i in range(len(noot)):
    vol.tone(noot[i],2*tijd[i])
    sleep(2*tijd[i])
```

STEM PROJECTEN

Geluid

$$f = 440 \cdot 2^{\frac{n}{12}}$$





	B	C noot	D tijd
=			
1	0	440.	0.125
2	3	523.251	0.125
3	5	587.33	0.125
4	7	659.255	0.25
5	7	659.255	0.25
6	8	698.456	0.125
7	5	587.33	0.125
8	7	659.255	0.5
9	-200	0.004229	0.125
10	5	587.33	0.125
11	5	587.33	0.0625
12	3	523.251	0.0625
13	2	493.883	0.125
14	3	523.251	0.25
15	0	440.	0.185
16	-2	391.995	0.185

c1 $=440 \cdot 2^{\frac{b1}{12.}}$

muziek.py

```
from ti_system import *
from ti_hub import *

noot=recall_list("noot")
tijd=recall_list("tijd")

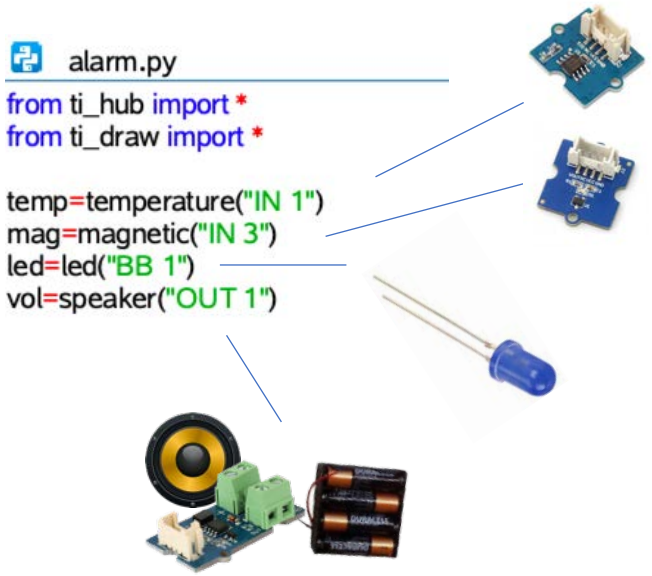
vol=speaker("BB 1")

for i in range(len(noot)):
    vol.tone(noot[i],2*tijd[i])
    sleep(2*tijd[i])
```

STEM PROJECTEN

Pet Car Alarm

Onderzoek de wetenschap achter het broeikaseffect en gebruik wiskunde, programmeren en engineering voor het bouwen van een smart warmte auto-alarm.



Conversie Farenheit ↔ Celcius!

Estimated Vehicle Interior Air Temperature v. Elapsed Time

Elapsed time	Outside Air Temperature (F)					
	70	75	80	85	90	95
0 minutes	70	75	80	85	90	95
10 minutes	89	94	99	104	109	114
20 minutes	99	104	109	114	119	124
30 minutes	104	109	114	119	124	129
40 minutes	108	113	118	123	128	133
50 minutes	111	116	121	126	131	136
60 minutes	113	118	123	128	133	138
> 1 hour	115	120	125	130	135	140

Inside Air Temperature (F)

Courtesy Jan Null, CCM; Department of Geosciences, San Francisco State University

STEM PROJECTEN

Pet Car Alarm

Onderzoek de wetenschap achter het broeikaseffect en gebruik wiskunde, programmeren en engineering voor het bouwen van een smart warmte auto-alarm.

alarm.py

```
from ti_hub import *
from ti_draw import *

temp=temperature("IN 1")
mag=magnetic("IN 3")
led=led("BB 1")
vol=speaker("OUT 1")
```

```
while get_key() != "esc":
    clear()
    t=round(temp.measurement(),1)
    m=mag.measurement()
```

Conversie Farenheit ↔ Celcius!

Estimated Vehicle Interior Air Temperature v. Elapsed Time

Elapsed time	Outside Air Temperature (F)					
	70	75	80	85	90	95
0 minutes	70	75	80	85	90	95
10 minutes	89	94	99	104	109	114
20 minutes	99	104	109	114	119	124
30 minutes	104	109	114	119	124	129
40 minutes	108	113	118	123	128	133
50 minutes	111	116	121	126	131	136
60 minutes	113	118	123	128	133	138
> 1 hour	115	120	125	130	135	140

Inside Air Temperature (F)

Courtesy Jan Null, CCM; Department of Geosciences, San Francisco State University

STEM PROJECTEN

Pet Car Alarm

Onderzoek de wetenschap achter het broeikaseffect en gebruik wiskunde, programmeren en engineering voor het bouwen van een smart warmte auto-alarm.

Conversie Farenheit ↔ Celcius!

Estimated Vehicle Interior Air Temperature v. Elapsed Time

Elapsed time	Outside Air Temperature (F)					
	70	75	80	85	90	95
0 minutes	70	75	80	85	90	95
10 minutes	89	94	99	104	109	114
20 minutes	99	104	109	114	119	124
30 minutes	104	109	114	119	124	129
40 minutes	108	113	118	123	128	133
50 minutes	111	116	121	126	131	136
60 minutes	113	118	123	128	133	138
> 1 hour	115	120	125	130	135	140

Inside Air Temperature (F)

Courtesy Jan Null, CCM; Department of Geosciences, San Francisco State University

alarm.py

```
from ti_hub import *
from ti_draw import *

temp=temperature("IN 1")
mag=magnetic("IN 3")
led=led("BB 1")
vol=speaker("OUT 1")
```

```
while get_key() != "esc":
    clear()
    t=round(temp.measurement(),1)
    m=mag.measurement()
    set_color(0,0,255)
    draw_text(50,80,"HEATH CAR ALARM")
    set_color(0,0,0)
    draw_text(50,120,"Temperature inside car = {}".format(t))

    if m<100:
        draw_text(50,100,"Baby in car")
    else:
        draw_text(50,100,"Baby not in car")

    if t>30 and m<100:
        set_color(255,0,0)
        draw_text(50,140,"Warning! Baby in heat distress")
        led.blink(10,1)
        vol.tone(440,1)
        sleep(0.5)
        vol.tone(880,0.5)
        sleep(0.5)
    else:
        set_color(0,100,0)
        draw_text(50,140,"No problems detected")
```

```
Running...

HEATH CAR ALARM
Baby in car
Temperature inside car = 23.7
No problems detected
```

STEM PROJECTEN

Pet Car Alarm

Onderzoek de wetenschap achter het broeikaseffect en gebruik wiskunde, programmeren en engineering voor het bouwen van een smart warmte auto-alarm.

Conversie Farenheit ↔ Celcius!

Estimated Vehicle Interior Air Temperature v. Elapsed Time

Elapsed time	Outside Air Temperature (F)					
	70	75	80	85	90	95
0 minutes	70	75	80	85	90	95
10 minutes	89	94	99	104	109	114
20 minutes	99	104	109	114	119	124
30 minutes	104	109	114	119	124	129
40 minutes	108	113	118	123	128	133
50 minutes	111	116	121	126	131	136
60 minutes	113	118	123	128	133	138
> 1 hour	115	120	125	130	135	140

Inside Air Temperature (F)

Courtesy Jan Null, CCM; Department of Geosciences, San Francisco State University

alarm.py

```
from ti_hub import *
from ti_draw import *

temp=temperature("IN 1")
mag=magnetic("IN 3")
led=led("BB 1")
vol=speaker("OUT 1")
```

```
while get_key() != "esc":
    clear()
    t=round(temp.measurement(),1)
    m=mag.measurement()
    set_color(0,0,255)
    draw_text(50,80,"HEATH CAR ALARM")
    set_color(0,0,0)
    draw_text(50,120,"Temperature inside car = {}".format(t))

    if m<100:
        draw_text(50,100,"Baby in car")
    else:
        draw_text(50,100,"Baby not in car")

    if t>30 and m<100:
        set_color(255,0,0)
        draw_text(50,140,"Warning! Baby in heat distress")
        led.blink(10,1)
        vol.tone(440,1)
        sleep(0.5)
        vol.tone(880,0.5)
        sleep(0.5)
    else:
        set_color(0,100,0)
        draw_text(50,140,"No problems detected")
```

Running...

HEATH CAR ALARM
Baby in car
Temperature inside car = 23.7
No problems detected

Running...

HEATH CAR ALARM
Baby in car
Temperature inside car = 32.1
Warning! Baby in heat distress

STEM PROJECTEN

Pet Car Alarm

Onderzoek de wetenschap achter het broeikaseffect en gebruik wiskunde, programmeren en engineering voor het bouwen van een smart warmte auto-alarm.

Conversie Farenheit ↔ Celcius!

Estimated Vehicle Interior Air Temperature v. Elapsed Time

Elapsed time	Outside Air Temperature (F)					
	70	75	80	85	90	95
0 minutes	70	75	80	85	90	95
10 minutes	89	94	99	104	109	114
20 minutes	99	104	109	114	119	124
30 minutes	104	109	114	119	124	129
40 minutes	108	113	118	123	128	133
50 minutes	111	116	121	126	131	136
60 minutes	113	118	123	128	133	138
> 1 hour	115	120	125	130	135	140

Inside Air Temperature (F)

Courtesy Jan Null, CCM; Department of Geosciences, San Francisco State University

alarm.py

```
from ti_hub import *
from ti_draw import *

temp=temperature("IN 1")
mag=magnetic("IN 3")
led=led("BB 1")
vol=speaker("OUT 1")
```

```
while get_key() != "esc":
    clear()
    t=round(temp.measurement(),1)
    m=mag.measurement()
    set_color(0,0,255)
    draw_text(50,80,"HEATH CAR ALARM")
    set_color(0,0,0)
    draw_text(50,120,"Temperature inside car = {}".format(t))

    if m<100:
        draw_text(50,100,"Baby in car")
    else:
        draw_text(50,100,"Baby not in car")

    if t>30 and m<100:
        set_color(255,0,0)
        draw_text(50,140,"Warning! Baby in heat distress")
        led.blink(10,1)
        vol.tone(440,1)
        sleep(0.5)
        vol.tone(880,0.5)
        sleep(0.5)
    else:
        set_color(0,100,0)
        draw_text(50,140,"No problems detected")
```

Running...

HEATH CAR ALARM
Baby in car
Temperature inside car = 23.7
No problems detected

Running...

HEATH CAR ALARM
Baby in car
Temperature inside car = 32.1
Warning! Baby in heat distress

Running...

HEATH CAR ALARM
Baby not in car
Temperature inside car = 28.4
No problems detected

www.tistemprojects.com

OOP met Python

Wiskundige objecten

Everthing is an object in Python

- ```
  >>>n=193
  >>>type(n)
  <class 'int'>
  >>>r=1.93
  >>>type(r)
  <class 'float'>
  ```

- ```
  >>>lst=[1,2,3]
  >>>type(lst)
  <class 'list'>
  >>>lst.append(4)
  >>>lst
  [1, 2, 3, 4]
  ```

- ```
  >>>email="k-stulens@ti.com"
  >>>type(email)
  <class 'str'>
  >>>email.upper()
  'K-STULENS@TI.COM'
  ```
  ```

OOP met Python

Wiskundige objecten

*circle.py

class = User Defined Object

```
from math import*
from ti_draw import *

class Circle:
    def __init__(self,rad=10,xcoord=100,ycoord=100):
        self.rad=rad
        self.xcoord=xcoord
        self.ycoord=ycoord

    def __str__(self):
        return 'Circle with radius {} and center point ({},{})'.format(self.rad,self.xcoord,self.ycoord)

    def getArea(self):
        return pi*self.rad**2

    def getCircumference(self):
        return 2*pi*self.rad

    def getCircle(self):
        draw_circle(self.xcoord,self.ycoord,self.rad)
```

→ Eigenschappen

→ Methoden

Everthing is an object in Python

- >>>n=193
  >>>type(n)
  <class 'int'>
  >>>r=1.93
  >>>type(r)
  <class 'float'>

- >>>lst=[1,2,3]
  >>>type(lst)
  <class 'list'>
  >>>lst.append(4)
  >>>lst
  [1, 2, 3, 4]

- >>>email="k-stulens@ti.com"
  >>>type(email)
  <class 'str'>
  >>>email.upper()
  'K-STULENS@TI.COM'

OOP met Python

Wiskundige objecten

Everthing is an object in Python

*circle.py

```
from math import*
from ti_draw import *

class Circle:
    def __init__(self,rad=10,xcoord=100,ycoord=100):
        self.rad=rad
        self.xcoord=xcoord
        self.ycoord=ycoord

    def __str__(self):
        return 'Circle with radius {} and center point ({},{})'.format(self.rad,self.xcoord,self.ycoord)

    def getArea(self):
        return pi*self.rad**2

    def getCircumference(self):
        return 2*pi*self.rad

    def getCircle(self):
        draw_circle(self.xcoord,self.ycoord,self.rad)
```

Python Shell

```
>>>#Running circle.py
>>>from circle import *
>>>c=Circle(25,100,150)
>>>c.rad
25
>>>c.xcoord
100
>>>c.ycoord
150
>>>c.getArea()
1963.495408493621
>>>c.getCircumference()
157.0796326794897
>>>
```

OOP met Python

Wiskundige objecten

Everthing is an object in Python

*circle.py

```
from math import*
from ti_draw import *

class Circle:
    def __init__(self,rad=10,xcoord=100,ycoord=100):
        self.rad=rad
        self.xcoord=xcoord
        self.ycoord=ycoord

    def __str__(self):
        return 'Circle with radius {} and center point ({},{})'.format(self.rad,self.xcoord,self.ycoord)

    def getArea(self):
        return pi*self.rad**2

    def getCircumference(self):
        return 2*pi*self.rad

    def getCircle(self):
        draw_circle(self.xcoord,self.ycoord,self.rad)
```

Python Shell

```
>>>#Running circle.py
>>>from circle import *
>>>c=Circle(25,100,150)
>>>c.rad
25
>>>c.xcoord
100
>>>c.ycoord
150
>>>c.getArea()
1963.495408493621
>>>c.getCircumference()
157.0796326794897
>>>
>>>print(c)
Circle with radius 25 and center point (100,150)
>>>
```

OOP met Python

Wiskundige objecten

Everthing is an object in Python

*circle.py

```
from math import*
from ti_draw import *

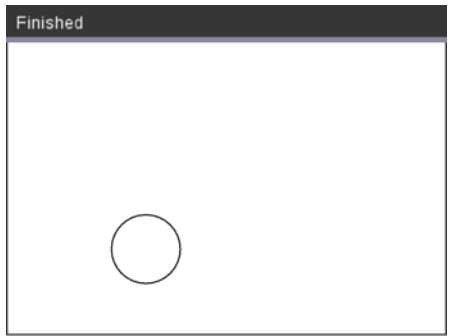
class Circle:
    def __init__(self,rad=10,xcoord=100,ycoord=100):
        self.rad=rad
        self.xcoord=xcoord
        self.ycoord=ycoord

    def __str__(self):
        return 'Circle with radius {} and center point ({},{})'.format(self.rad,self.xcoord,self.ycoord)

    def getArea(self):
        return pi*self.rad**2

    def getCircumference(self):
        return 2*pi*self.rad

    def getCircle(self):
        draw_circle(self.xcoord,self.ycoord,self.rad)
```



Python Shell

```
>>>#Running circle.py
>>>from circle import *
>>>c=Circle(25,100,150)
>>>c.rad
25
>>>c.xcoord
100
>>>c.ycoord
150
>>>c.getArea()
1963.495408493621
>>>c.getCircumference()
157.0796326794897
>>>
>>>print(c)
Circle with radius 25 and center point (100,150)
>>>
>>>c.getCircle()
>>>
```

OOP met Python

Output (virtual) objecten

Everthing is an object in Python

```
from ti_draw import *
from ti_image import *
from time import *

w,h = get_screen_dim()
hub_front=load_image("front_view")
front_x = w/2 - hub_front.w/2
front_y = h/2 - hub_front.h/2

class led():
    def __init__(self,port):
        self.img = load_image("led_t")
        self.dx = 82
        self.x_off = 68
        if port == "BB 1":
            self.x = self.x_off
        elif port == "BB 2":
            self.x = self.x_off+ self.dx
        elif port == "BB 3":
            self.x = self.x_off+ 2*self.dx
        self.y =160
        self.img.show_image(self.x,self.y)
        hub_front.show_image(0,0)

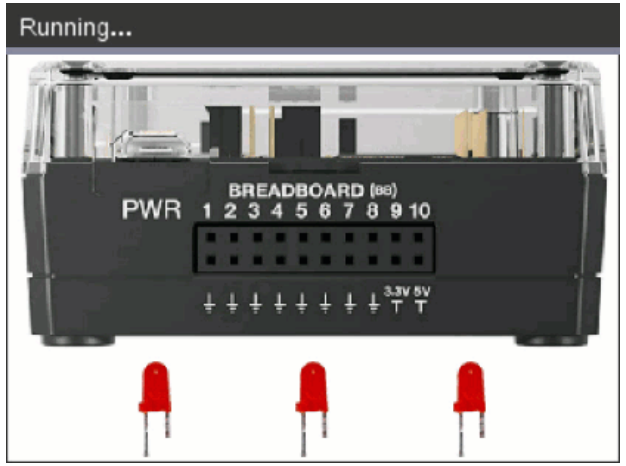
    def on (self):
        self.img.show_image(self.x,self.y)
        hub_front.show_image(front_x,0)
        set_color(255,0,150)
        fill_rect(self.x+3,self.y+5,13,20)
        fill_circle(self.x+9,self.y+6,6)

    def off (self):
        self.img.show_image(self.x,self.y)
        hub_front.show_image(front_x,0)
        hub_front.show_image(front_x,0)
        self.img.show_image(self.x,self.y)
```

```
leds=[]

for n in range(3):
    leds.append (led('BB ' + str(n+1)))

while not escape():
    for aled in leds:
        aled.on()
        sleep_ms(250)
        aled.off()
        sleep_ms(250)
```

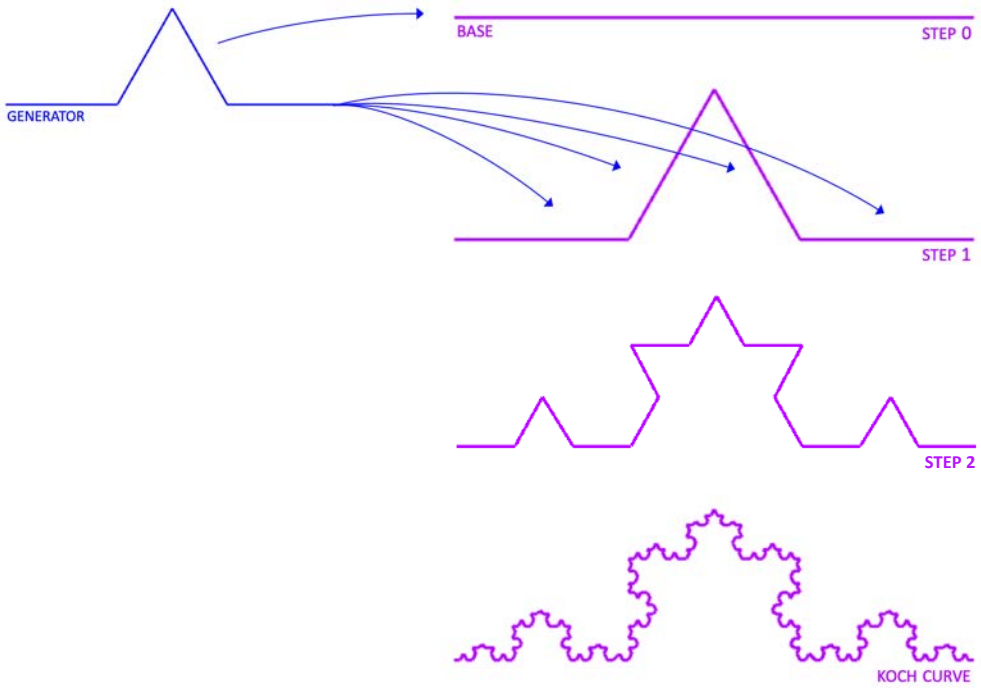


STEAM Projecten

Eigenaardige linaire kronkels

$$T : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

T = a composition of a translation, rotation and a dilation

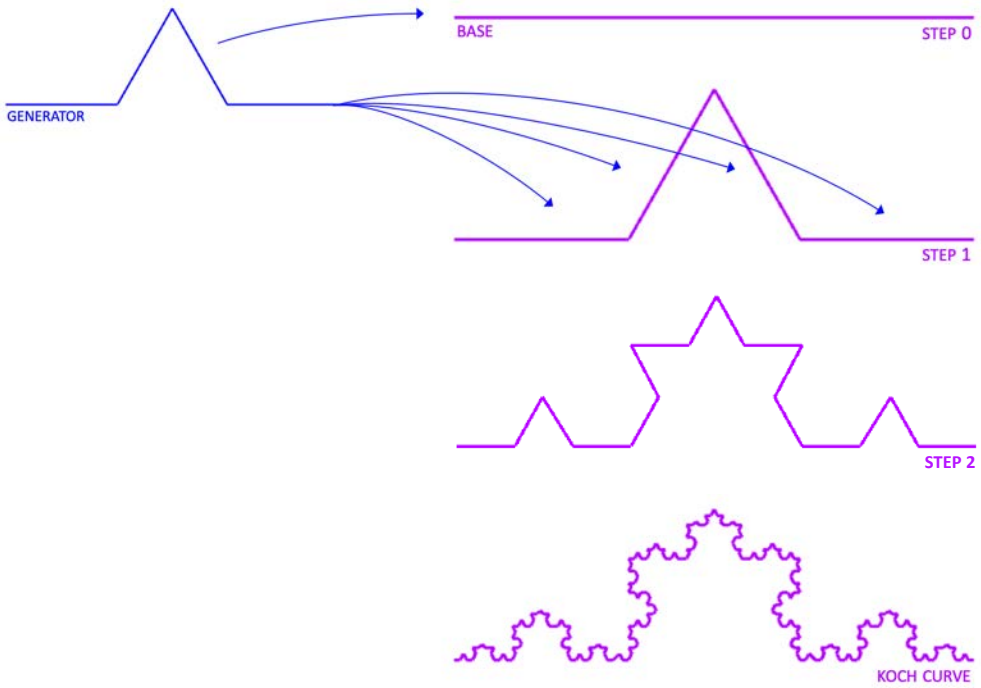


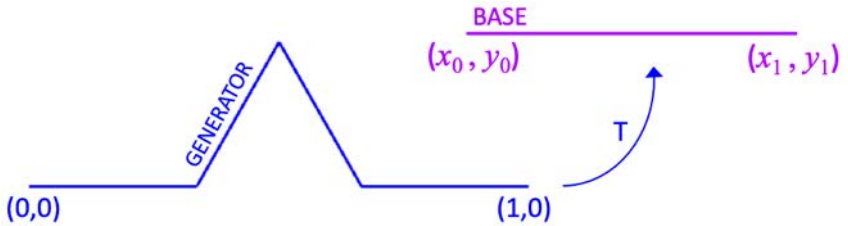
STEAM Projecten

Eigenaardige linaire kronkels

$$T : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

T = a composition of a translation, rotation and a dilation

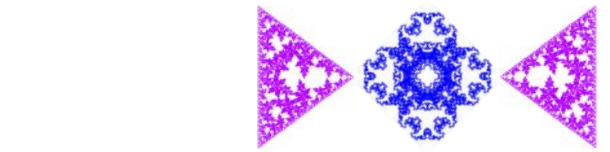




$$T : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} x_1 - x_0 & y_0 - y_1 \\ y_1 - y_0 & x_1 - x_0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_0 \\ y_1 \end{bmatrix}$$

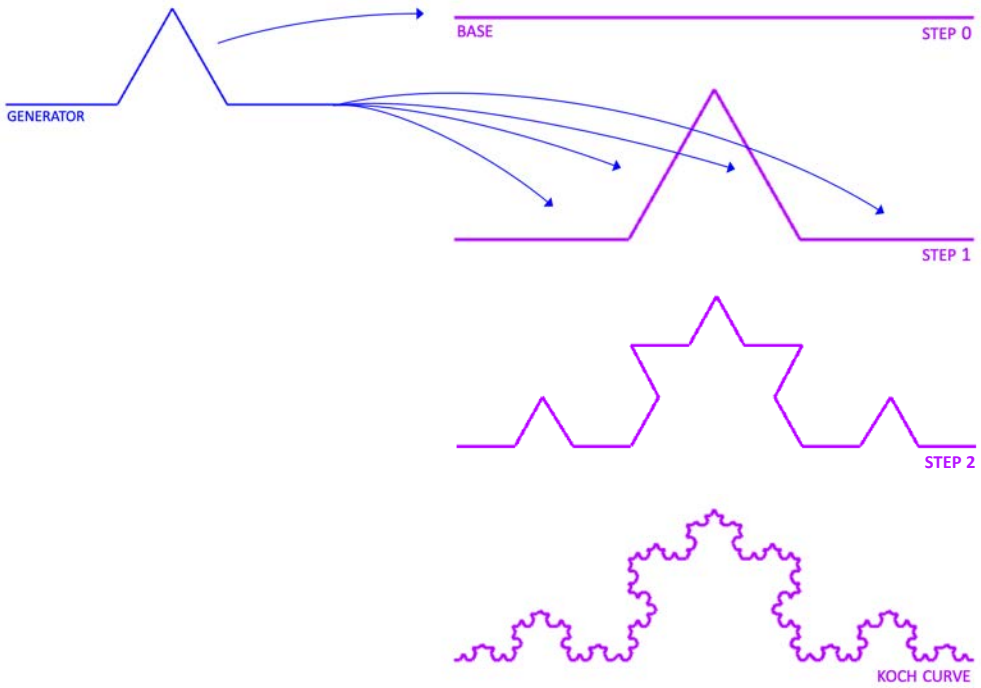
STEAM Projecten

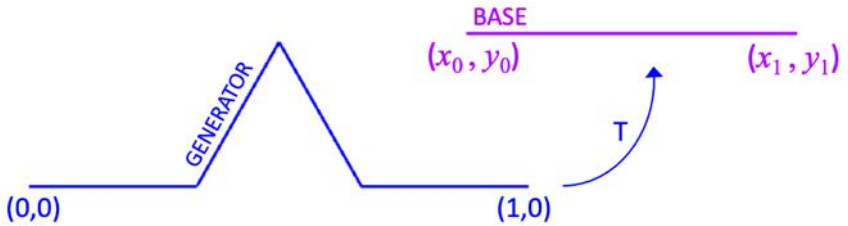
Eigenaardige linaire kronkels



$$T : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

T = a composition of a translation, rotation and a dilation





$$T : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} x_1 - x_0 & y_0 - y_1 \\ y_1 - y_0 & x_1 - x_0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

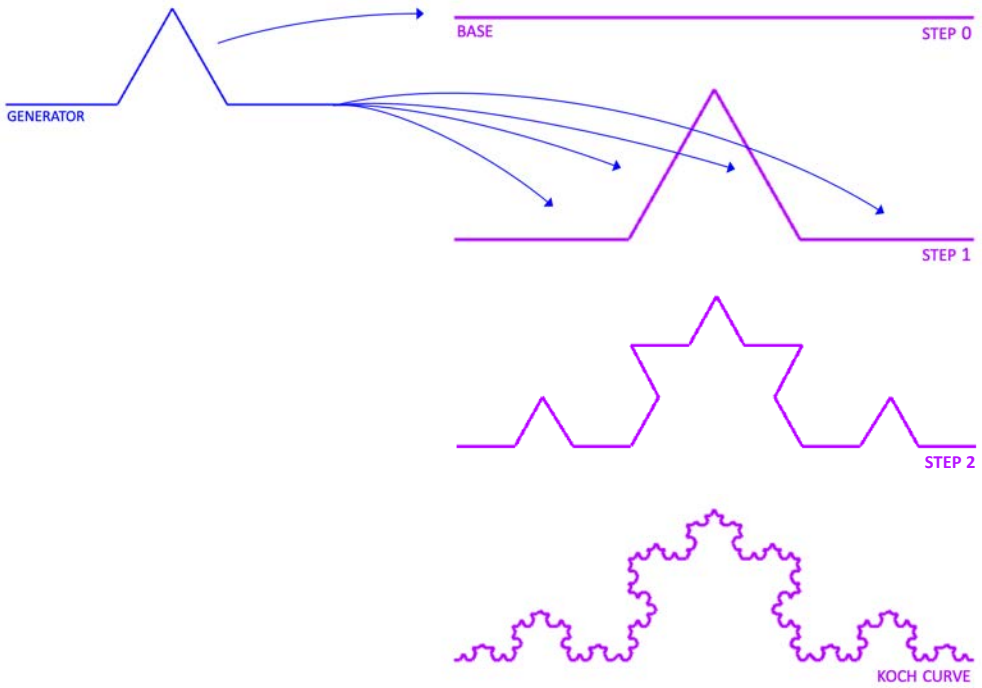
```
# Fractal LinearTransformation based on segment [p1,p2]
def transform(x,y,p1,p2):
    a=p2[0]-p1[0] ; b=p2[1]-p1[1] ; e=p1[0] ; f=p1[1]
    return [a*x-b*y+e,b*x+a*y+f]
```

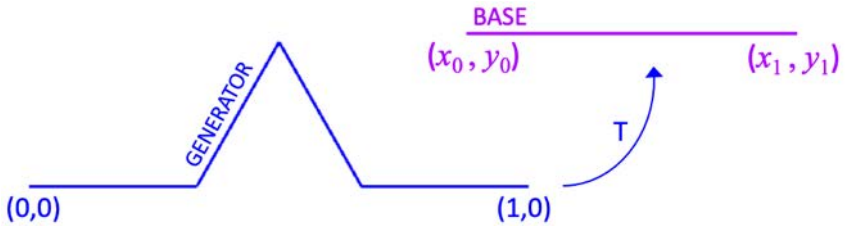
STEAM Projecten

Eigenaardige linaire kronkels

$$T : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

T = a composition of a translation, rotation and a dilation





$$T : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} x_1 - x_0 & y_0 - y_1 \\ y_1 - y_0 & x_1 - x_0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_0 \\ y_1 \end{bmatrix}$$

```
# Fractal LinearTransformation based on segment [p1,p2]
def transform(x,y,p1,p2):
    a=p2[0]-p1[0] ; b=p2[1]-p1[1] ; e=p1[0] ; f=p1[1]
    return [a*x-b*y+e,b*x+a*y+f]

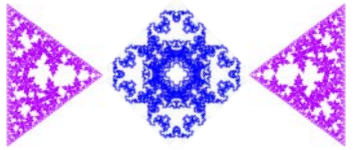

# Fractal Points on segment [p1,p2]
def genpoints(p1,p2):
    points=[]
    for i in range(0,u):
        yield transform(xg[i],yg[i],p1,p2)
```

STEAM Projecten

Eigenaardige linaire kronkels

```
# Fractal LinearTransformation based on segment [p1,p2]
def transform(x,y,p1,p2):
    a=p2[0]-p1[0] ; b=p2[1]-p1[1] ; e=p1[0] ; f=p1[1]
    return [a*x-b*y+e,b*x+a*y+f]
```

```
# Fractal Points on segment [p1,p2]
def genpoints(p1,p2):
    points=[]
    for i in range(0,u):
        yield transform(xg[i],yg[i],p1,p2)
```



```
# Generate Fractal
v=len(xb)

for j in range (v-1):
    set_color(178,0,255)
    pb=[xb[j],yb[j]] ; pe=[xb[j+1],yb[j+1]]

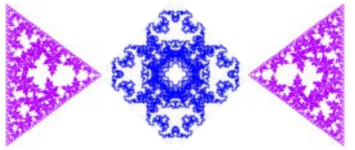
    # Draw Base - Generation 0
    if gen==0:
        draw_segment(pb[0],pb[1],pe[0],pe[1])

    #  Define Fractal Generation
    elif gen>=1:
        fracgen=[[genpoints(pb,pe)]]
        for g in range(0,gen-1):
            fracgen.append([])
            for k in range(0,(u-1)**g):
                pe=next(fracgen[0][k])
                for i in range (0,u-1):
                    pb=pe.copy()
                    pe=next(fracgen[0][k])
                    fracgen[1].append(genpoints(pb,pe))
            fracgen.remove(fracgen[0])

    # Plot Fractal
    if gen>=1:
        for seg in range(0,(u-1)**(gen-1)):
            np=[]
            for i in range(0,u):
                np.append(next(fracgen[0][seg]))
            for j in range(0,u-1):
                draw_segment(np[j][0],np[j][1],np[j+1][0],np[j+1][1])
```

STEAM Projecten

Eigenaardige linaire kronkels



```
# Fractal LinearTransformation based on segment [p1,p2]
def transform(x,y,p1,p2):
    a=p2[0]-p1[0] ; b=p2[1]-p1[1] ; e=p1[0] ; f=p1[1]
    return [a*x-b*y+e,b*x+a*y+f]
```

```
# Fractal Points on segment [p1,p2]
def genpoints(p1,p2):
    points=[]
    for i in range(0,u):
        yield transform(xg[i],yg[i],p1,p2)
```

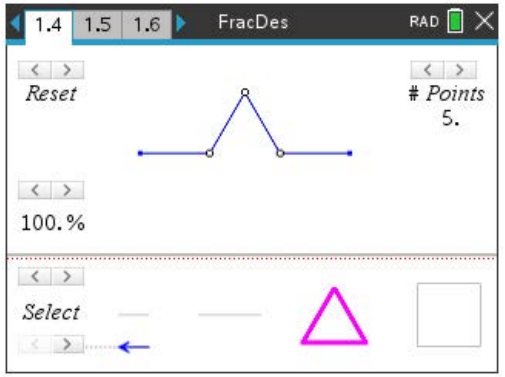
```
# Generate Fractal
v=len(xb)

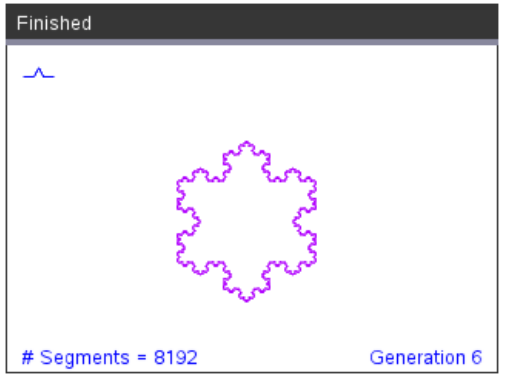
for j in range (v-1):
    set_color(178,0,255)
    pb=[xb[j],yb[j]] ; pe=[xb[j+1],yb[j+1]]

    # Draw Base - Generation 0
    if gen==0:
        draw_segment(pb[0],pb[1],pe[0],pe[1])

    #  Define Fractal Generation
    elif gen>=1:
        fracgen=[[genpoints(pb,pe)]]
        for g in range(0,gen-1):
            fracgen.append([])
            for k in range(0,(u-1)**g):
                pe=next(fracgen[0][k])
                for i in range (0,u-1):
                    pb=pe.copy()
                    pe=next(fracgen[0][k])
                    fracgen[1].append(genpoints(pb,pe))
            fracgen.remove(fracgen[0])

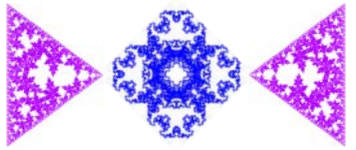
    # Plot Fractal
    if gen>=1:
        for seg in range(0,(u-1)**(gen-1)):
            np=[]
            for i in range(0,u):
                np.append(next(fracgen[0][seg]))
            for j in range(0,u-1):
                draw_segment(np[j][0],np[j][1],np[j+1][0],np[j+1][1])
```





STEAM Projecten

Eigenaardige linaire kronkels



```
# Fractal LinearTransformation based on segment [p1,p2]
def transform(x,y,p1,p2):
    a=p2[0]-p1[0] ; b=p2[1]-p1[1] ; e=p1[0] ; f=p1[1]
    return [a*x-b*y+e,b*x+a*y+f]
```

```
# Fractal Points on segment [p1,p2]
def genpoints(p1,p2):
    points=[]
    for i in range(0,u):
        yield transform(xg[i],yg[i],p1,p2)
```

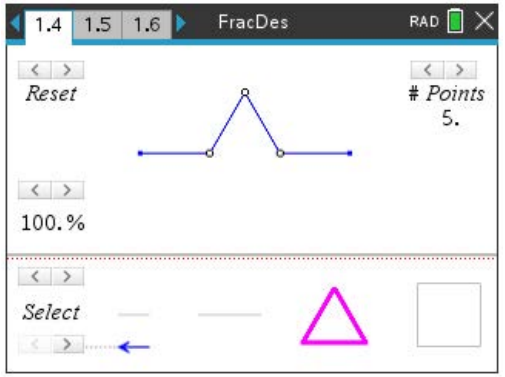
```
# Generate Fractal
v=len(xb)

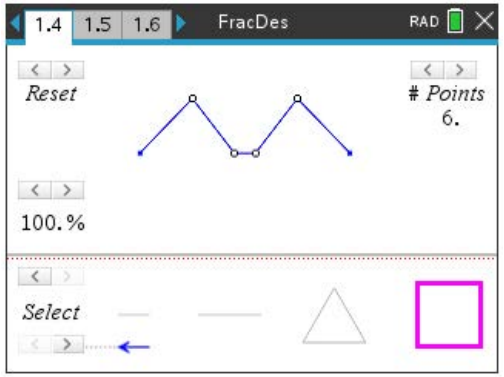
for j in range (v-1):
    set_color(178,0,255)
    pb=[xb[j],yb[j]] ; pe=[xb[j+1],yb[j+1]]

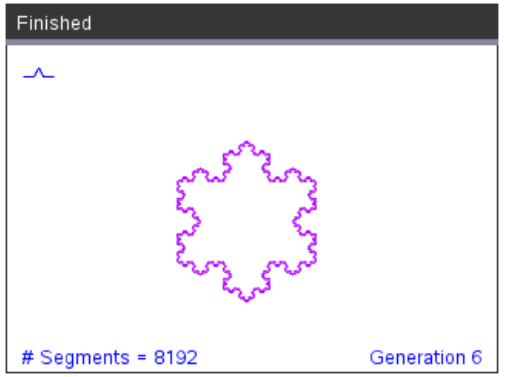
    # Draw Base - Generation 0
    if gen==0:
        draw_segment(pb[0],pb[1],pe[0],pe[1])

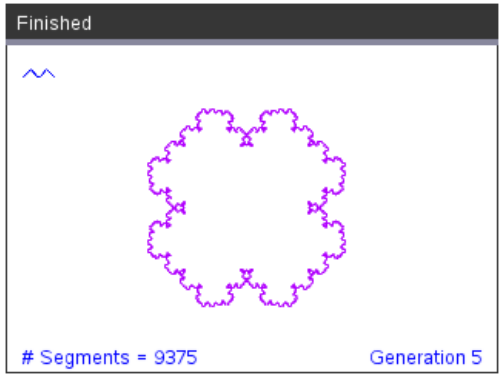
    #  Define Fractal Generation
    elif gen>=1:
        fracgen=[[genpoints(pb,pe)]]
        for g in range(0,gen-1):
            fracgen.append([])
            for k in range(0,(u-1)**g):
                pe=next(fracgen[0][k])
                for i in range (0,u-1):
                    pb=pe.copy()
                    pe=next(fracgen[0][k])
                    fracgen[1].append(genpoints(pb,pe))
            fracgen.remove(fracgen[0])

    # Plot Fractal
    if gen>=1:
        for seg in range(0,(u-1)**(gen-1)):
            np=[]
            for i in range(0,u):
                np.append(next(fracgen[0][seg]))
            for j in range(0,u-1):
                draw_segment(np[j][0],np[j][1],np[j+1][0],np[j+1][1])
```



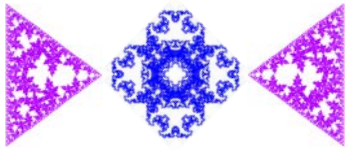






STEAM Projecten

Eigenaardige linaire kronkels



```
# Fractal LinearTransformation based on segment [p1,p2]
def transform(x,y,p1,p2):
    a=p2[0]-p1[0] ; b=p2[1]-p1[1] ; e=p1[0] ; f=p1[1]
    return [a*x-b*y+e,b*x+a*y+f]
```

```
# Fractal Points on segment [p1,p2]
def genpoints(p1,p2):
    points=[]
    for i in range(0,u):
        yield transform(xg[i],yg[i],p1,p2)
```

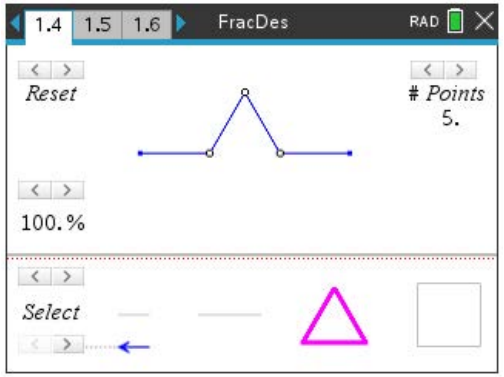
```
# Generate Fractal
v=len(xb)

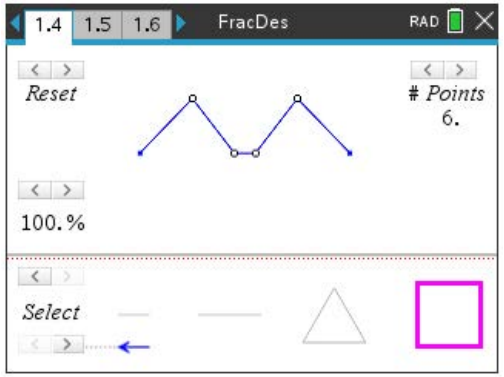
for j in range (v-1):
    set_color(178,0,255)
    pb=[xb[j],yb[j]] ; pe=[xb[j+1],yb[j+1]]

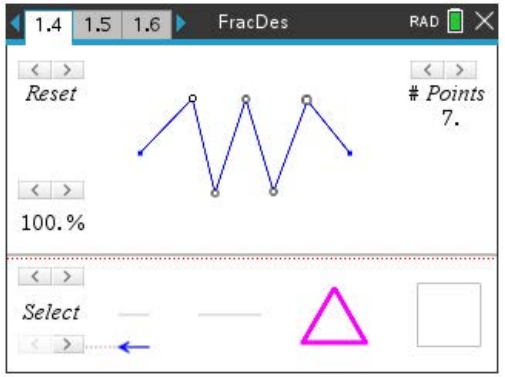
    # Draw Base - Generation 0
    if gen==0:
        draw_segment(pb[0],pb[1],pe[0],pe[1])

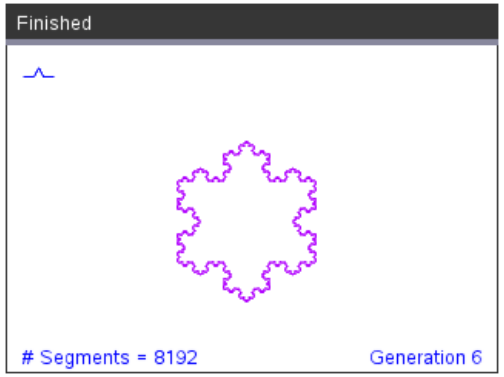
    #  Define Fractal Generation
    elif gen>=1:
        fracgen=[[genpoints(pb,pe)]]
        for g in range(0,gen-1):
            fracgen.append([])
            for k in range(0,(u-1)**g):
                pe=next(fracgen[0][k])
                for i in range (0,u-1):
                    pb=pe.copy()
                    pe=next(fracgen[0][k])
                    fracgen[1].append(genpoints(pb,pe))
            fracgen.remove(fracgen[0])

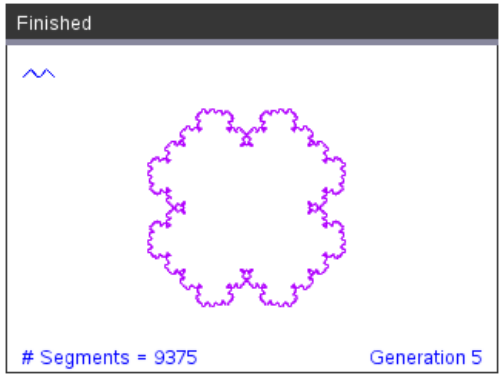
    # Plot Fractal
    if gen>=1:
        for seg in range(0,(u-1)**(gen-1)):
            np=[]
            for i in range(0,u):
                np.append(next(fracgen[0][seg]))
            for j in range(0,u-1):
                draw_segment(np[j][0],np[j][1],np[j+1][0],np[j+1][1])
```

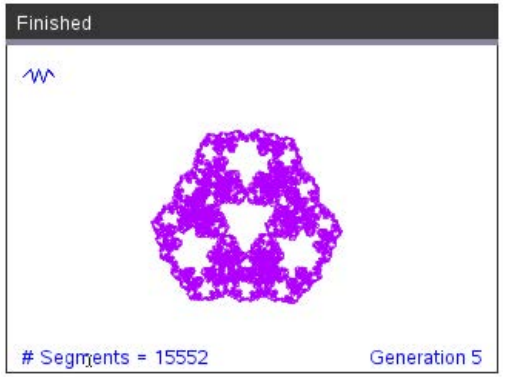












Python in de klas

b.wikkerink@gmail.com
k-stulens@ti.com

T³ NEDERLAND

Teaching is more than the dissemination of content

Teaching should cultivate curiosity and inquiry

Teaching should spark student imagination and creativity

Some skills become more important because technology requires it

Some skills become less important because technology replaces it

Some skills becomes possible because technology allows it

based on Bert Waits - 2000