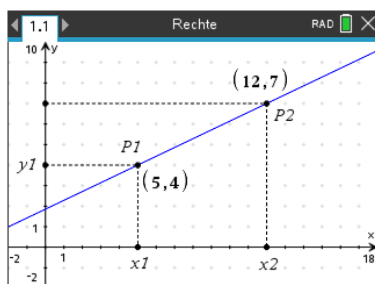


1. Inleidend voorbeeld

We tekenen een rechte in de TI-Nspire CX Graphs App door de punten $P1$ en $P2$.



We bewaren de coördinaten van de punten $P1$ en $P2$ in de volgende TI-Nspire CX variabelen $x1, y1$ en $x2, y2$ als volgt:

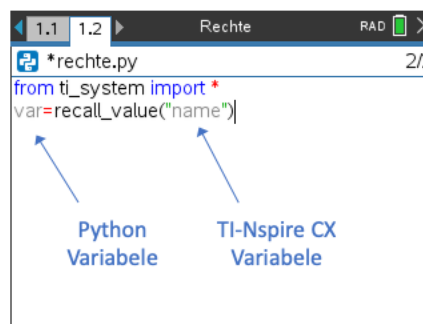
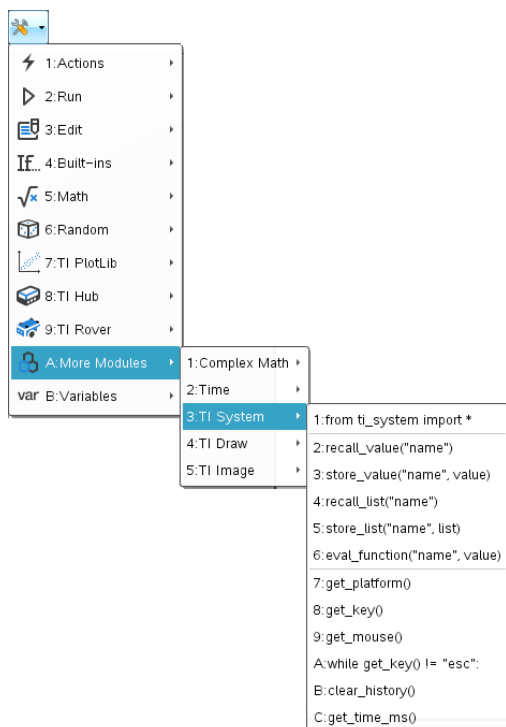
$$P1(x1, y1) \text{ \& } P2(x2, y2).$$

De vergelijking van de rechte is gelijk aan:

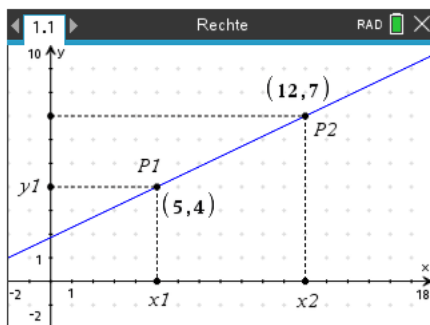
$$y - y1 = \frac{y2 - y1}{x2 - x1}(x - x1).$$

Met de module TI System kan variabelen vanuit TI-Nspire CX geïmporteerd worden in een Python-programma als Python variabelen. Na het importeren van de module `ti_system` selecteren we het commando `recall_value()`.

De onderstaande syntax-template verschijnt die aangeeft hoe de Python variabele te declareren op basis van de TI-Nspire CX variabele.



We berekenen de richtingscoëfficiënt m en bepalen de rechte door de twee punten $P1$ en $P2$.



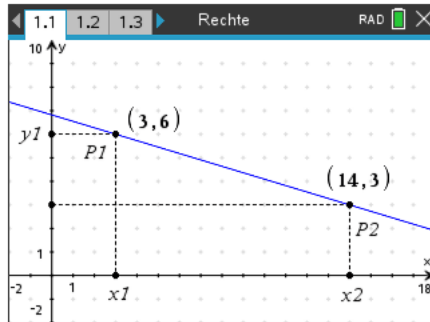
```

rechte.py
from ti_system import *
x1=recall_value("x1")
x2=recall_value("x2")
y1=recall_value("y1")
y2=recall_value("y2")
m=(y2-y1)/(x2-x1)
print("De rico = ",m)
print("De vergelijking door P1 en P2")
print(" y={0:1.4f}(x-{1})+{2}".format(m,x1,y1))
    
```

```

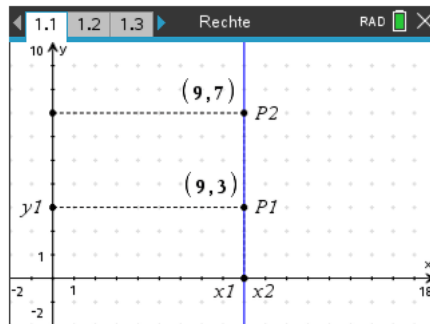
Python Shell
>>>#Running rechte.py
>>>from rechte import *
De rico = 0.4285714285714286
De vergelijking door P1 en P2
y=0.4286(x-5)+4
>>>
    
```

Als we de punten verplaatsen en het programma opnieuw runnen krijgen we de nieuwe vergelijking:



```
Python Shell 6/6
>>>#Running rechte.py
>>>from rechte import *
De rico = -0.2727272727272727
De vergelijking door P1 en P2
y=-0.2727(x-3)+6
>>>|
```

Maar wat met een verticale rechte?



```
Python Shell 10/10
>>>#Running rechte.py
>>>from rechte import *
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
  File "/Users/a0920230/Library/Preferences/Texas Instruments/TI-Nspire CX CAS Premium Teacher Software/python/doc29/rechte.py", line 6, in <module>
ZeroDivisionError: divide by zero
>>>|
```

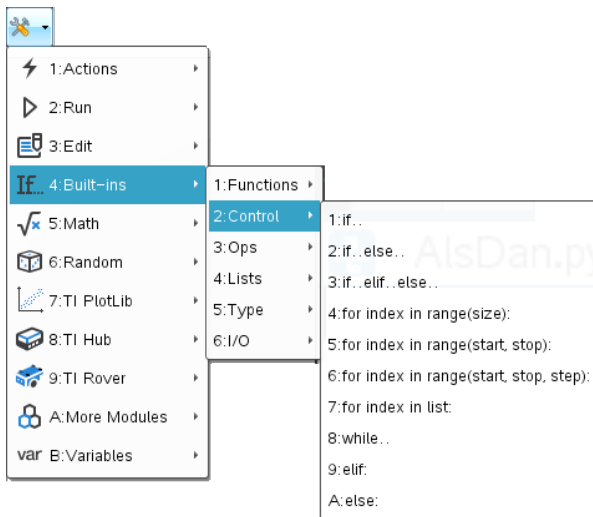
2. If-Statements

Om bovenstaande foutmelding te voorkomen testen we de conditie $x1 == x2$ of $x1 != x2$.

Hiervoor selecteren we in het Control-menu de optie 2: if..else.. en passen ons programma als volgt aan.

Om commentaar in te voeren start je de regel met #.

De template van het if-statement verschijnt automatisch in de Program Editor.



```
Rechte 9/21
*rechte.py
from ti_system import *

# Invoer coördinaten
x1=recall_value("x1")
x2=recall_value("x2")
y1=recall_value("y1")
y2=recall_value("y2")

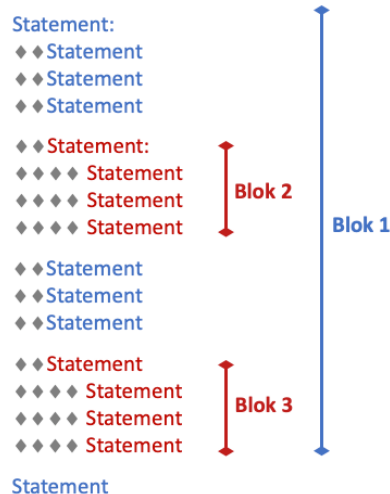
#Rico test
if BooleanExpr:
  **block
else:
  **block
```

```
if BooleanExpr:
♦♦block
else:
♦♦block
```

Een if-statement bestaat uit het sleutelwoord if, gevolgd door een voorwaarde en een dubbelpunt. De statements na het dubbelpunt, moeten in een blok staan. Indien de voorwaarde True is, wordt alle statements in het blok uitgevoerd.

Een blok is een verzameling van gegroepeerde statements. De Whitespace voor de statements geeft aan welke statements tot eenzelfde blok. Statements die op dezelfde positie starten, m.a.w. regels met dezelfde inspringingen, behoren tot hetzelfde blok.

Merk op dat if-statements niet afgesloten worden met een End-statement maar begin en einde is gebaseerd op gelijke inspringingen. In TI-Technologie wordt de Whitespace aangeduid met ♦♦.



Een if-statement kan uitgebreid worden met een else-statement met eventueel een elif-statement tussenin.

Voor ons programma geeft dit het volgende:

<pre>1.1 1.2 1.3 Rechte RAD 7/20 rechte.py from ti_system import * # Invoer coördinaten x1=recall_value("x1") x2=recall_value("x2") y1=recall_value("y1") y2=recall_value("y2")</pre>	<pre>1.1 1.2 1.3 Rechte RAD 18/18 rechte.py #Vergelijking if x1==x2: ♦♦ print("De vergelijking door P1 en P2") ♦♦ print(" x={}".format(x1)) else: ♦♦ m=(y2-y1)/(x2-x1) ♦♦ print("De vergelijking door P1 en P2") ♦♦ print(" y={0:1.4f}(x-{{1}})+{{2}}".format(m,x1,y1))</pre>	<pre>1.1 1.2 1.3 Rechte RAD 5/5 Python Shell >>>#Running rechte.py >>>from rechte import * De vergelijking door P1 en P2 x=9 >>> </pre>
--	---	--

Met een extra elif-statement kunnen we ook de vergelijking van een horizontale rechte eleganter weergeven:

	<pre>1.1 1.2 1.3 Rechte RAD 19/21 rechte.py #vergelijking if x1==x2: ♦♦ print("De vergelijking door P1 en P2") ♦♦ print(" x={}".format(x1)) elif y1==y2: ♦♦ print("De vergelijking door P1 en P2") ♦♦ print(" y={}".format(y1)) else: ♦♦ m=(y2-y1)/(x2-x1) ♦♦ print("De vergelijking door P1 en P2") ♦♦ print(" y={0:1.4f}(x-{{1}})+{{2}}".format(m,x1,y1))</pre>	<pre>1.1 1.2 1.3 Rechte RAD 5/5 Python Shell >>>#Running rechte.py >>>from rechte import * De vergelijking door P1 en P2 y=5 >>> </pre>
--	---	--

Om de structuur van Python-code goed op een rij te hebben, is het belangrijk steeds aandacht te hebben op hoe de structuur van een Python-programma gebaseerd is op inspringingen (indents of whitespace).

Nog twee voorbeelden om de structuur van de `if`-statements te illustreren.

1. Bepaal de absolute waarde van een reëel getal basierend op de definitie $|x| = \begin{cases} x & \text{als } x \leq 0 \\ -x & \text{als } x < 0 \end{cases}$.

Graph of $f_1(x) = |x|$

```
abs.py
x=float(input("x = "))

if x >= 0:
    **abs = x
else:
    **abs = -x

print("De absolute waarde")
print("van {} is {}".format(x,abs))
```

```
Python Shell
>>>#Running abs.py
>>>from abs import *
x = 3,14
De absolute waarde
van 3.14 is 3.14
>>>#Running abs.py
>>>from abs import *
x = -3,14
De absolute waarde
van -3.14 is 3.14
>>>|
```

2. Bereken de functiewaarde voor de volgende stuksgewijs gedefinieerde functie $f_1(x) = \begin{cases} x + 2 & \text{als } x \leq -2 \\ 0 & \text{als } -2 < x < 2 \\ x - 2 & \text{als } x \geq 2 \end{cases}$.

Graph of $f_1(x) = \begin{cases} x+2, x \leq -2 \\ 0, -2 < x < 2 \\ x-2, x \geq 2 \end{cases}$

```
trap.py
x=float(input("x = "))

if x <= -2:
    **y = x+2
elif x > -2 and x < 2:
    **y = 0
else:
    **y = x-2

print("f1({}) = {}".format(x,y))
```

```
Python Shell
>>>#Running trap.py
>>>from trap import *
x = 4,25
f1(4,25) = 2,25
>>>#Running trap.py
>>>from trap import *
x = -2
f1(-2,0) = 0,0
>>>|
```