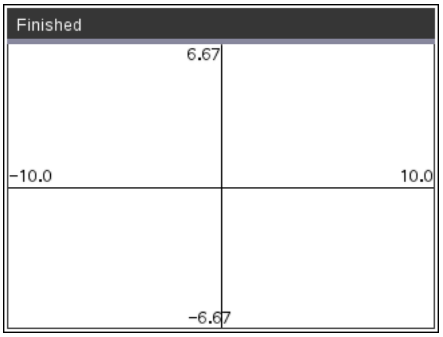
1. Setup Plot-omgeving

De TI PlotLib-module laat toe om grafieken en data-plots te tekenen. Voor deze grafische module moet voor TI-Nspire CX-software de document Preview-mode ingesteld zijn als Handheld.

We starten met het definiëren van een TI PlotLib-object, plt. Het hierna uitvoeren van de functie axes() toont de default instellingen van het plot-venster.

```
import ti_plotLib as plt
plt.axes("on")
```

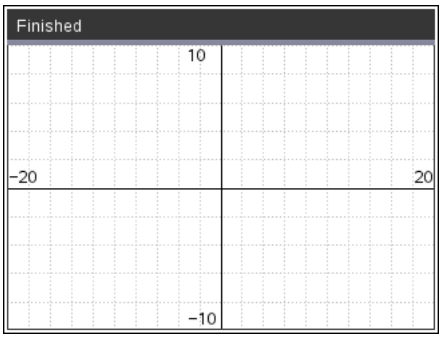


De volgende opties zijn voor axes() beschikbaar:

- "axes" alleen de assen, geen eindwaarden
- "window" alleen de eindwaarden, geen assen
- "off" geen assen en geen eindwaarden

Manueel instellen van het venster doe je met window(xmin,xmax,ymax,ymin) en met de functie grid(xScale,yScale,"style") wordt een grid getekend:

```
import ti_plotLib as plt
plt.window(-20,20,-10,10)
plt.grid(1,1,"dotted")
plt.axes("on")
```

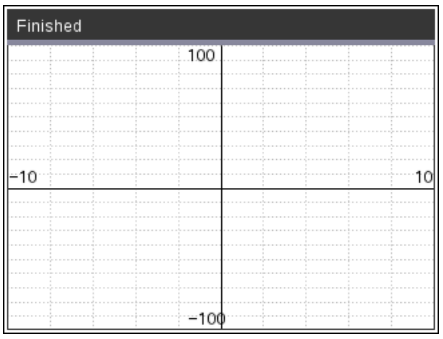


De opties voor de grid-stijl zijn:

- "solid" _____
- "dotted" .....
- "dashed" - - - - -

Deze plt-vensterinstellingen kunnen ook gedefineerd worden als een functie:

```
import ti_plotLib as plt
def venster(xmin,xmax,ymin,ymax,xscale,yscale):
    plt.window(xmin,xmax,ymin,ymax)
    plt.grid(xscale,yscale,"dotted")
    plt.axes("on")
venster(-10,10,-100,100,2,10)
```

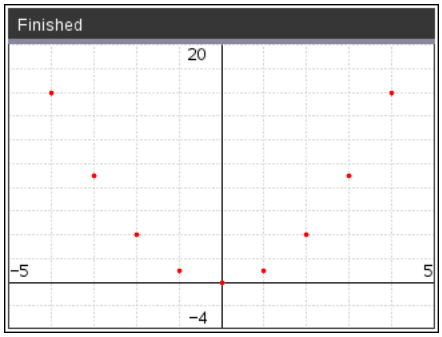


2. Grafieken

Het plotten van een grafiek kan beschouwd worden als het kleuren van de pixels van het scherm, t.o.v. een referentie-assenstelsel, die voldoen aan het verband voorgeschreven door de functiedefinitie.

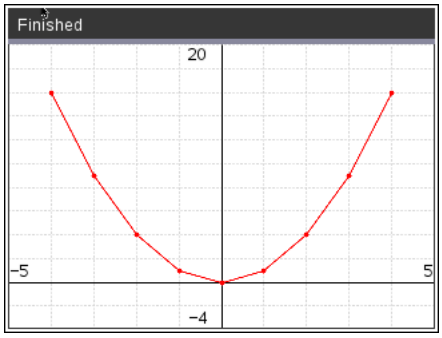
Een voorbeeld voor $f(x) = x^2$.

```
import ti_plotlib as plt

def venster(xmin,xmax,ymin,ymax,xscale,yscale):
    plt.window(xmin,xmax,ymin,ymax)
    plt.grid(xscale,yscale,"dotted")
    plt.axes("on")

def f(x):
    return x**2

xmin=-5 ; xmax=5 ; ymin=-4 ; ymax=20
venster(xmin,xmax,ymin,ymax,1,2)

for x in range(xmin+1,xmax):
    plt.color(255,0,0)
    plt.plot(x,f(x),"o")
```



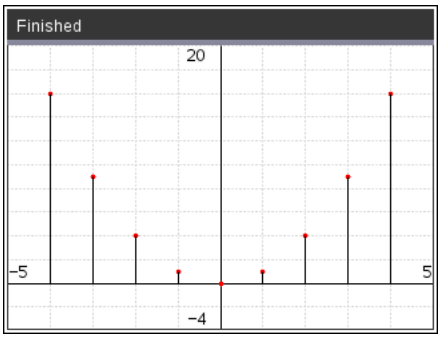
Toevoegen van de onderstaande code verbindt de punten van de grafiek door een lijnstuk:

```
    if x < xmax-1:
        plt.line(x,f(x),x+1,f(x+1))
```



of de volgende code om de hiernaast afgebeelde 3e plot te genereren:

```
    plt.color(0,0,0)
    plt.pen("thin","solid")
    plt.line(x,f(x),x,0)
```



Voor de functie plot(x,y,"mark") zijn de volgende mark-opties beschikbaar:

- "o"
- "+"
- "x"
- "."

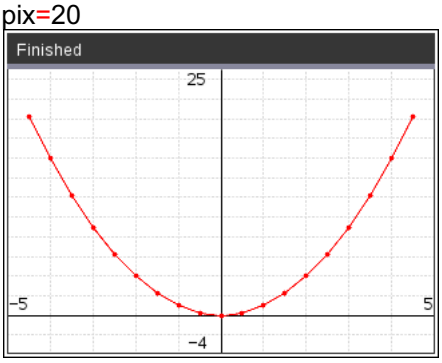
En voor pen("size","style"):

Size	Style
"thin"	"solid"
"medium"	"dotted"
"thick"	"dashed"

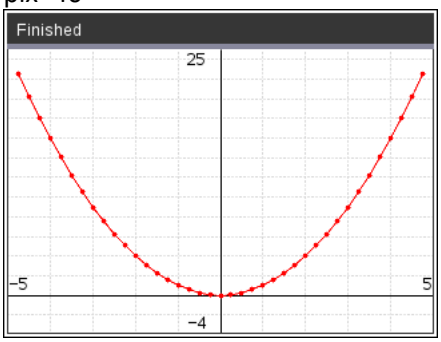
Het verfijnen van de te plotten grafiek kan als volgt:

```
import ti_plotlib as plt

def venster(xmin,xmax,ymin,ymax,xscale,yscale):
    plt.window(xmin,xmax,ymin,ymax)
    plt.grid(xscale,yscale,"dotted")
    plt.axes("on")

def f(x):
    return x**2

xmin=-5 ; xmax=5 ; ymin=-4 ; ymax=25
venster(xmin,xmax,ymin,ymax,1,2)

pix=20
res=(xmax-xmin)/pix
xrange=[xmin+n*res for n in range(1,pix)]

for x in xrange:
    plt.color(255,0,0)
    plt.plot(x,f(x),"o")
    if x < xmax-res:
        plt.line(x,f(x),x+res,f(x+res))
```

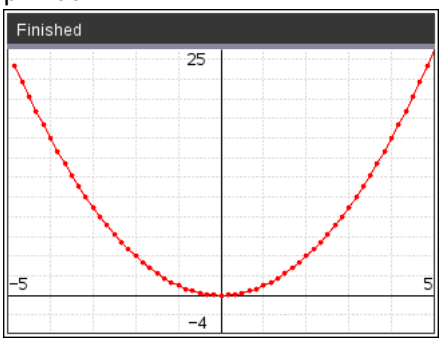
pix=20



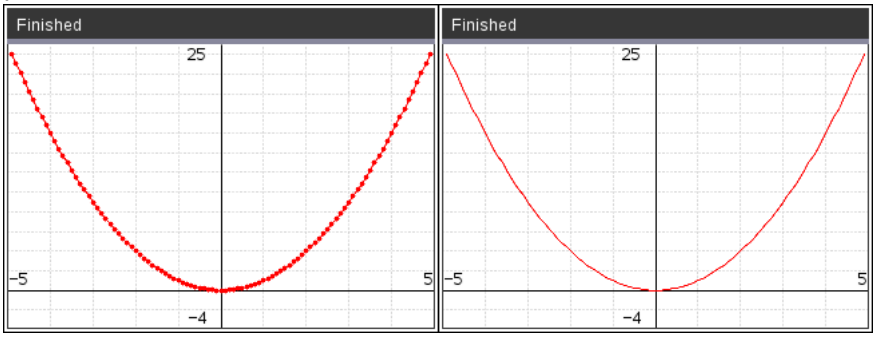
pix=40



pix=60

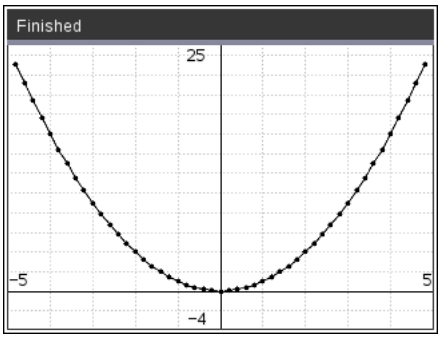


pix=100



I.p.v. voor plot() pixels te gebruiken als argumenten (en eventueel de pixels te verbinden met lijnstukjes), kan plot() ook gebruik maken van twee lijsten als argumenten en dit geeft een connected pixel plot als output.

```
import ti_plotlib as plt

def venster(xmin,xmax,ymin,ymax,xscale,yscale):
    plt.window(xmin,xmax,ymin,ymax)
    plt.grid(xscale,yscale,"dotted")
    plt.axes("on")

def f(x):
    return x**2

xmin=-5 ; xmax=5 ; ymin=-4 ; ymax=25
venster(xmin,xmax,ymin,ymax,1,2)

pix=50
res=(xmax-xmin)/pix
xrange=[xmin+n*res for n in range(1,pix)]
yrange=[f(i) for i in xrange]
plt.plot(xrange,yrange,"o")
```



3. Plotten van data

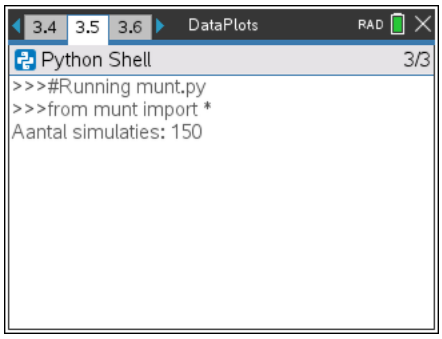
3.1. Kans-simulatie

We simuleren het opwerpen van een muntstuk, een kansexperiment met een binomiale kansverdeling:

- Kans op geen kop: $\frac{1}{4}$
- Kans op één keer kop: $\frac{1}{2}$
- Kans op twee keren kop: $\frac{1}{4}$

Voor het simuleren en herhaaldelijk uitvoeren van het experiment, runnen we de volgende code.

```
from random import *
import ti_plotlib as plt

def venster(xmin,xmax,ymin,ymax,xscale,yscale):
    plt.window(xmin,xmax,ymin,ymax)
    plt.axes("on")
```

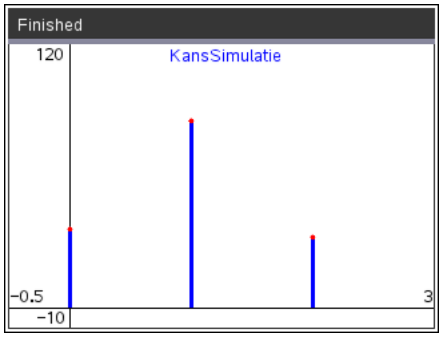


Definitie mogelijke uitkomsten en input aantal herhalingen experiment:

```
kop=[i for i in range(0,3)]
aantal=int(input("Aantal simulaties: "))
```

Herhaaldelijk uitvoeren experiment en opslaan resultaten:

```
for k in range(aantal):
    munt1=randint(0,1)
    munt2=randint(0,1)
    som=munt1+munt2
    kop[som]+=1
```

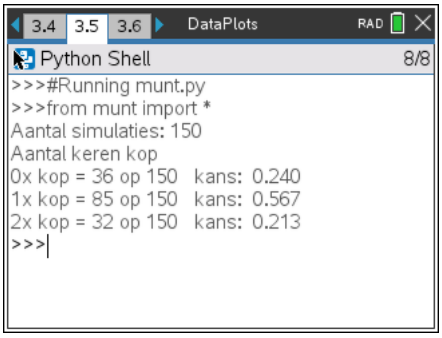


Definitie plot-venster en plotten van een titel:

```
xmin=-0.5 ; xmax=3 ; ymin=-10 ; ymax=int(aantal-20*aantal/100)
venster(xmin,xmax,ymin,ymax,1,10)

plt.color(0,0,255)
plt.title("KansSimulatie")
```

Plotten van de resultaten van de simulatie:

```
for x in range(0,3):
    plt.color(0,0,255)
    plt.pen("medium","solid")
    plt.line(x,0,x,kop[x])
    plt.color(255,0,0)
    plt.plot(x,kop[x],"o")
```

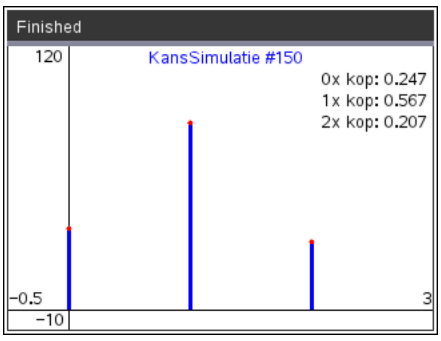


Weergeven van de benaderingen van de kans in de shell.

```
print("Aantal keren kop")
print("0x kop =",kop[0],"op",aantal,"  kans:  {0:2.3f}".format(kop[0]/aantal))
print("1x kop =",kop[1],"op",aantal,"  kans:  {0:2.3f}".format(kop[1]/aantal))
print("2x kop =",kop[2],"op",aantal,"  kans:  {0:2.3f}".format(kop[2]/aantal))
```

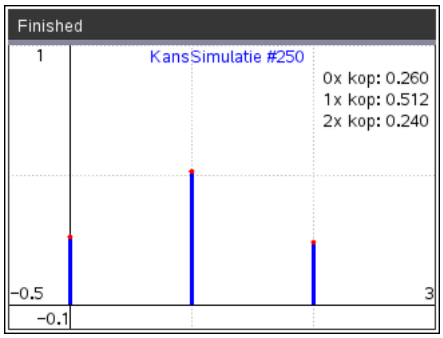
De kansbenaderingen kunnen ook als volgt weergegeven worden in het plot-venster:

```
plt.color(0,0,255)
plt.title("KansSimulatie #{}".format(aantal))
  ⋮
plt.color(0,0,0)
plt.text_at(2,"0x kop: {0:2.3f} ".format(kop[0]/aantal),"right")
plt.text_at(3,"1x kop: {0:2.3f} ".format(kop[1]/aantal),"right")
plt.text_at(4,"2x kop: {0:2.3f} ".format(kop[2]/aantal),"right")
```



In plaats van de resultaten kunnen ook door een kleine aanpassing van de code de kansbenaderingen geplot worden.

```
xmin=-0.5 ; xmax=3 ; ymin=-0.1 ; ymax=1
venster(xmin,xmax,ymin,ymax,1,0.5)
for x in range(0,3):
    plt.color(0,0,255)
    plt.pen("medium","solid")
    plt.line(x,0,x,kop[x]/aantal)
    plt.color(255,0,0)
    plt.plot(x,kop[x]/aantal,"o")
```

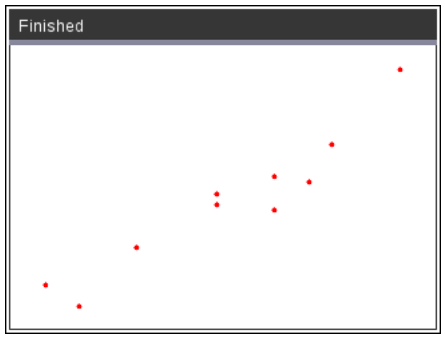


3.2. Puntenwolk

De methode(functie) scatter() plot een puntenwolk van twee datasets (lijsten).
We illustreren dit aan de hand van de lengte en gewicht van tien 18-jarigen:

Lengte	163	185	180	175	168	175	191	180	160	183
Gewicht	60	90	78	81	71	79	104	84	64	83

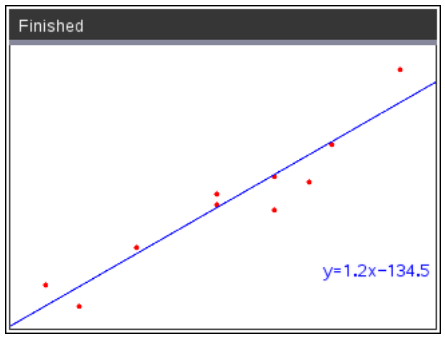
```
import ti_plotlib as plt

lengte=[163,185,180,175,168,175,191,180,160,183]
gewicht=[60,90,78,81,71,79,104,84,64,83]

plt.auto_window(lengte,gewicht)

plt.color(255,0,0)
plt.scatter(lengte,gewicht,"o")
```



Merk op dat in bovenstaande code auto_window() een venster creëert dat alle data bevat; di i.p.v. manueel een venster in te stellen baserend op de data.

De functie lin_reg() plot en berekent de beste lineaire benadering van de puntenwolk.

```
plt.color(0,0,255)
plt.lin_reg(lengte,gewicht,"right")
```



plt.m en plt.b geven de coëficiënten van de regressierechte:

```
Python Shell
>>>plt.m
1.215261958997722
>>>plt.b
-134.4861047835991
```