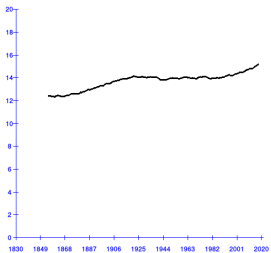
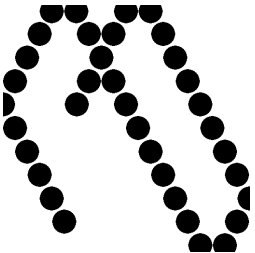
Vorlesung 06

1.5.5 Graphische Ausgabe mit `stddraw`

Das Modul `stddraw.py` bietet einfache Mittel zur graphischen Ausgabe.

Damit können z.B. Graphiken gemalt oder bewegte Bilder dargestellt werden.





`python3 kugel_3.py 0.05 0.095 5`

```
python3 filter-jahr-txk.py 15 < Jena-Wetterdaten.txt | \
python3 jahres_mittel.py | \
python3 male_glatte_kurve_mit_achsen.py 1830 2022 0 20
```

5 Die „Elemente“ des Malens

- Ein Blatt Papier bzw. eine Leinwand.
- Ein Koordinatensystem für die Leinwand.
-

5 Die „Elemente“ des Malens

- Ein Blatt Papier bzw. eine Leinwand.
- Ein Koordinatensystem für die Leinwand.
-

5 Die „Elemente“ des Malens

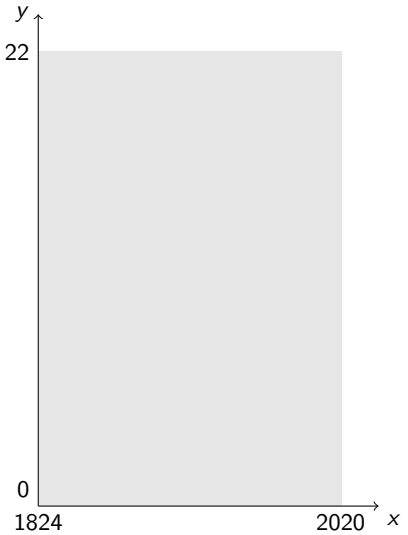
- Ein Blatt Papier bzw. eine Leinwand.
- Ein Koordinatensystem für die Leinwand.
- Ein Stift.

5 Die „Elemente“ des Malens

- Ein Blatt Papier bzw. eine Leinwand.
- Ein Koordinatensystem für die Leinwand.
- Ein Stift.

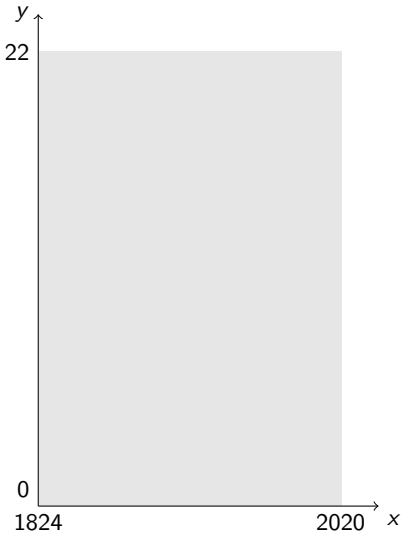
5 Die „Elemente“ des Malens

- Ein Blatt Papier bzw. eine Leinwand.
- Ein Koordinatensystem für die Leinwand.
- Ein Stift.



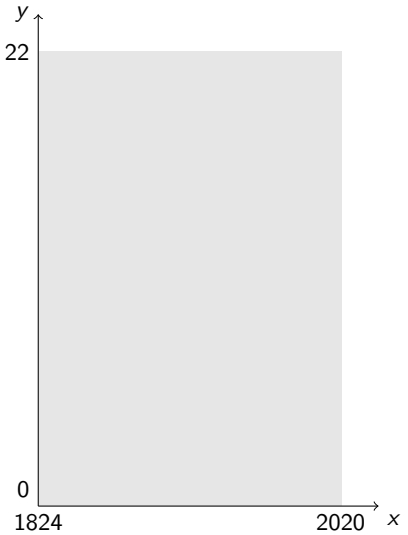
5 Die „Elemente“ des Malens

- Ein Blatt Papier bzw. eine Leinwand.
- Ein Koordinatensystem für die Leinwand.
- Stifte



5 Die „Elemente“ des Malens

- Ein Blatt Papier bzw. eine Leinwand.
- Ein Koordinatensystem für die Leinwand.
- Stifte mit verschiedenen Farben und Größen.

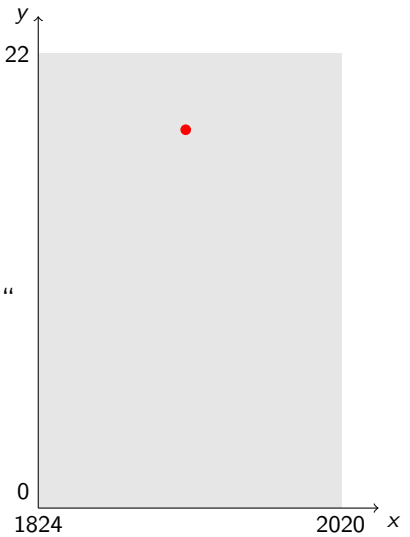


5 Die „Elemente“ des Malens

- Ein Blatt Papier bzw. eine Leinwand.
- Ein Koordinatensystem für die Leinwand.
- Stifte mit verschiedenen Farben und Größen.

„Male roten Punkt mit einem Stift der Größe 2 an Position $(1920, 18.2)$.“

Nimm roten Stift der Größe 2, male Punkt an Position $(1920, 18.2)$.



5 Die „Elemente“ des Malens

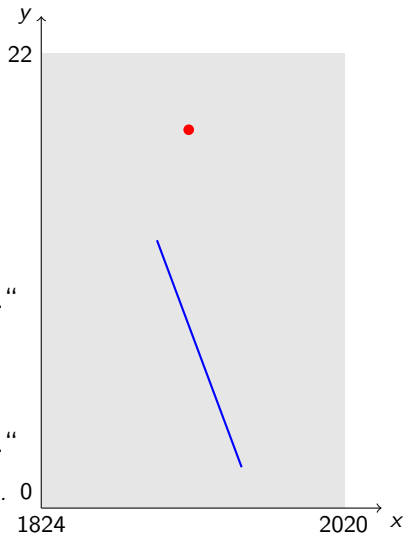
- Ein Blatt Papier bzw. eine Leinwand.
- Ein Koordinatensystem für die Leinwand.
- Stifte mit verschiedenen Farben und Größen.

„Male roten Punkt mit einem Stift der Größe 2 an Position $(1920, 18.2)$.“

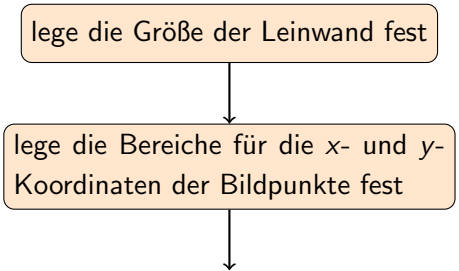
Nimm roten Stift der Größe 2, male Punkt an Position $(1920, 18.2)$.

„Male blauen Strich mit Stift der Größe 1 von $(1899, 13)$ nach $(1954, 2)$.“

Nimm blauen Stift der Größe 1, male Strich von $(1899, 13)$ *nach* $(1954, 2)$.



Vorbereitung zum Malen mit `stddraw` (optional)

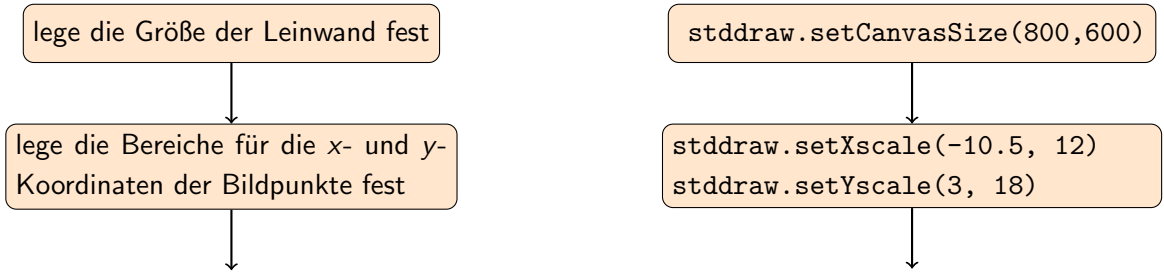



```
stddraw.setCanvasSize(800,600)
stddraw.setXscale(-10.5, 12)
stddraw.setYscale(3, 18)
```

`stddraw.setCanvasSize(breite,hoehe)`	Breite und Höhe (`int`) der Leinwand (in Pixel)
`stddraw.setXscale(xmin,xmax)`	kleinster und größter Wert (`float`) auf der x-Achse der Leinwand
`stddraw.setXscale(xmin,xmax)`	kleinster und größter Wert (`float`) auf der y-Achse der Leinwand

Standardmäßig besteht die Leinwand aus 512×512 Pixeln und die Achsen haben Skalenbereiche $0 \ldots 1$.

Vorbereitung zum Malen mit `stddraw` (optional)



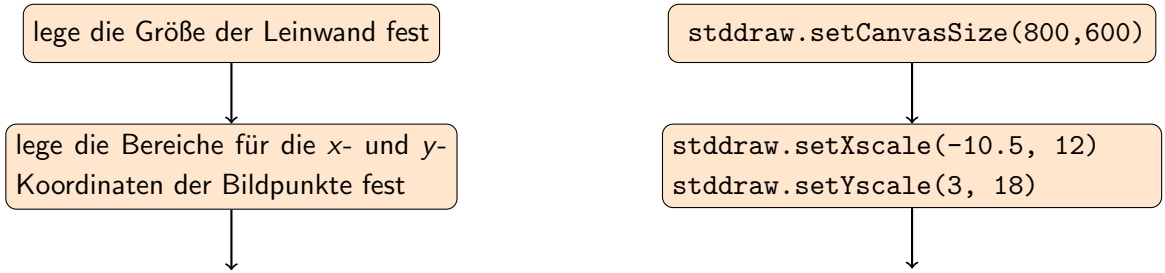
`stddraw.setCanvasSize(breite,hoehe)` Breite und Höhe (`int`) der Leinwand (in Pixel)

`stddraw.setXscale(xmin,xmax)` kleinster und größter Wert (`float`) auf der x-Achse der Leinwand

`stddraw.setXscale(xmin,xmax)` kleinster und größter Wert (`float`) auf der y-Achse der Leinwand

Standardmäßig besteht die Leinwand aus 512×512 Pixeln und die Achsen haben Skalenbereiche $0 \ldots 1$.

Vorbereitung zum Malen mit `stddraw` (optional)



`stddraw.setCanvasSize(breite,hoehe)` Breite und Höhe (`int`) der Leinwand (in Pixel)

`stddraw.setXscale(xmin,xmax)` kleinster und größter Wert (`float`) auf der x-Achse der Leinwand

`stddraw.setXscale(xmin,xmax)` kleinster und größter Wert (`float`) auf der y-Achse der Leinwand

---

Standardmäßig besteht die Leinwand aus 512×512 Pixeln und die Achsen haben Skalenbereiche $0 \ldots 1$.

Die wichtigsten Malfunktionen von `stddraw`

Stiftgröße und -farbe:

`stddraw.setPenRadius(r)`	setze die Stiftdicke auf `r` (Default-Wert von `r` ist 0.005)
`stddraw.setPenColor(c)`	setzt die Farbe des Stiftes auf `c`; `c` kann z.B. `stddraw.BLUE` sein

mit Farben BLACK, BLUE, CYAN, DARK_GRAY, GRAY, GREEN, LIGHT_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW

Punkte und Striche malen:

`stddraw.line(x0, y0, x1, y1)`	male einen Strich von (`x0`,`y0`) zu (`x1`,`y1`)
`stddraw.point(x, y)`	male einen Punkt an (`x`,`y`)
`stddraw.show(t)`	zeige das Bild im standard-drawing-Fenster und warte `t` Millisekunden; falls `t` weggelassen wird, dann warte, bis das Fenster vom Nutzer geschlossen wird

6 Programmier-Auftrag: stelle die Entwicklung der durchschnittlichen Tageshöchsttemperaturen pro Jahr graphisch dar

Der Aufruf

```
python3 filter-jahr-txk.py < Jena-Wetterdaten.txt | python3 jahres_mittel.py
```

liefert die Daten

```
1824 13.50
1825 13.38
1826 13.27
1827 12.92
1828 13.07
1829 10.47
...
```

6 Programmier-Auftrag: stelle die Entwicklung der durchschnittlichen Tageshöchsttemperaturen pro Jahr graphisch dar

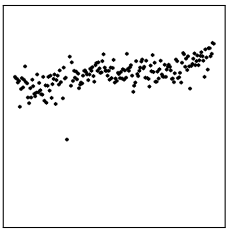
Der Aufruf

```
python3 filter-jahr-txk.py < Jena-Wetterdaten.txt | python3 jahres_mittel.py
```

liefert die Daten

```
1824 13.50
1825 13.38
1826 13.27
1827 12.92
1828 13.07
1829 10.47
...
```

Mögliche graphische Ausgabe:



„Jeder Messwert wird als Punkt gemalt“

6 Programmier-Auftrag: stelle die Entwicklung der durchschnittlichen Tageshöchsttemperaturen pro Jahr graphisch dar

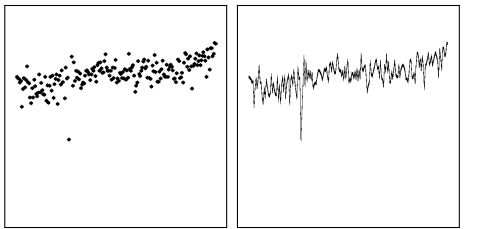
Der Aufruf

```
python3 filter-jahr-txk.py < Jena-Wetterdaten.txt | python3 jahres_mittel.py
```

liefert die Daten

```
1824 13.50
1825 13.38
1826 13.27
1827 12.92
1828 13.07
1829 10.47
...
```

Mögliche graphische Ausgabe:



„Jeder Messwert wird als Punkt gemalt“ oder
„aufeinanderfolgende Messwerte werden als Strich gemalt“.

6 Programmier-Auftrag: stelle die Entwicklung der durchschnittlichen Tageshöchsttemperaturen pro Jahr graphisch dar

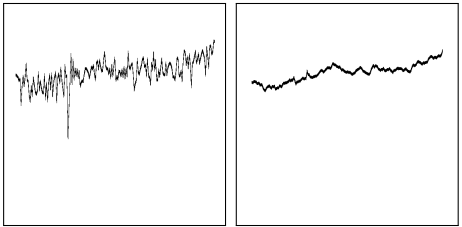
Der Aufruf

```
python3 filter-jahr-txk.py < Jena-Wetterdaten.txt | python3 jahres_mittel.py
```

liefert die Daten

```
1824 13.50
1825 13.38
1826 13.27
1827 12.92
1828 13.07
1829 10.47
...
```

Mögliche graphische Ausgabe:



„Jeder Messwert wird als Punkt gemalt“ oder
„aufeinanderfolgende Messwerte werden als Strich gemalt“.

6 Programmier-Auftrag: stelle die Entwicklung der durchschnittlichen Tageshöchsttemperaturen pro Jahr graphisch dar

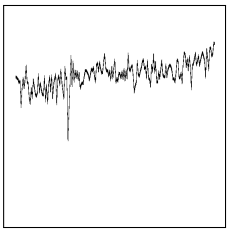
Der Aufruf

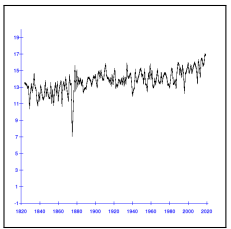
```
python3 filter-jahr-txk.py < Jena-Wetterdaten.txt | python3 jahres_mittel.py
```

liefert die Daten

```
1824 13.50
1825 13.38
1826 13.27
1827 12.92
1828 13.07
1829 10.47
...
```

Mögliche graphische Ausgabe:





„Jeder Messwert wird als Punkt gemalt“ oder

„aufeinanderfolgende Messwerte werden als Strich gemalt“.

Jeder Messwert wird als Punkt gemalt: die Idee

- Lies zeilenweise *Jahr Temperatur* ein.
- Fasse jede Zeile als x- und y-Wert eines Punktes auf und male diese Punkte.
- Beim Malen sind die x-Koordinaten im Bereich $1824 \ldots 2019$,
  und die y-Koordinaten sind im Bereich $0 \ldots 20$.

Die grobe Struktur des Programms

1. Lies den Bereich der x- und y-Koordinaten von der Kommandozeile.
2. Lies alle Punkte von standard input,
   und male sie auf die Leinwand.

Schritt 2 kann auf zwei Arten umgesetzt werden:

(A) Male jeden Punkt direkt nach dem Einlesen.

(B) lies zuerst *alle* Punkte
male danach alle Punkte

Die grobe Struktur des Programms

1. Lies den Bereich der x- und y-Koordinaten von der Kommandozeile.
2. Lies alle Punkte von standard input,
   und male sie auf die Leinwand.

Schritt 2 kann auf zwei Arten umgesetzt werden:

(A) Male jeden Punkt direkt nach dem Einlesen.

(B) lies zuerst *alle* Punkte
male danach alle Punkte

für jede Eingabezeile (Zeile $\hat{=}$ Punkt):
    bestimme den entsprechenden Punkt
    male ihn

Die grobe Struktur des Programms

1. Lies den Bereich der x- und y-Koordinaten von der Kommandozeile.
2. Lies alle Punkte von standard input,
   und male sie auf die Leinwand.

Schritt 2 kann auf zwei Arten umgesetzt werden:

(A) Male jeden Punkt direkt nach dem Einlesen.

für jede Eingabezeile (Zeile $\hat{=}$ Punkt):
    bestimme den entsprechenden Punkt
    male ihn

(B) lies zuerst *alle* Punkte
    male danach alle Punkte

für jede Eingabezeile:
    bestimme den Punkt und speichere ihn

für jeden gespeicherten Punkt:
    male ihn

Das Programm male_punkte_A.py

```
#-------------------------------------------------------------------------------
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als Punkte auf eine Leinwand mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------
import sys, stddraw
# Richte die Achsen der Leinwand ein und bereite den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.01)

# Standard input liefert zeilenweise x- und y-Werte von Punkten.
# Jeder dieser Punkte wird gemalt, die Leinwand wird angezeigt,
# und anschließend wird ein bisschen gewartet.
for zeile in sys.stdin:
  punkt = str.split(zeile)
  x_wert = float(punkt[0])
  y_wert = float(punkt[1])
  stddraw.point(x_wert, y_wert)
  stddraw.show(30)

stddraw.show()
```

Das Programm male_punkte_A.py

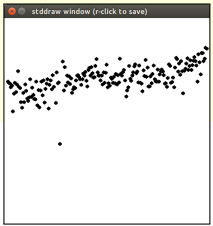
```
#-------------------------------------------------------------------------------
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als Punkte auf eine Leinwand mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------
import sys, stddraw
# Richte die Achsen der Leinwand ein und bereite den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.01)

# Standard input liefert zeilenweise x- und y-Werte von Punkt
# Jeder dieser Punkte wird gemalt, die Leinwand wird angezeig
# und anschließend wird ein bisschen gewartet.
for zeile in sys.stdin:
  punkt = str.split(zeile)
  x_wert = float(punkt[0])
  y_wert = float(punkt[1])
  stddraw.point(x_wert, y_wert)
  stddraw.show(30)

stddraw.show()
```

```
mundhenk@ma3: ~
mm@home: python3 filter-jahr-txk.py < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
```

Das Programm male_punkte_A.py

```
#-------------------------------------------------------------------------------
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als Punkte auf eine Leinwand mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------
import sys, stddraw
# Richte die Achsen der Leinwand ein und bereite den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.01)

# Standard input liefert zeilenweise x- und y-Werte von Punkt
# Jeder dieser Punkte wird gemalt, die Leinwand wird angezeig
# und anschließend wird ein bisschen gewartet.
for zeile in sys.stdin:
  punkt = str.split(zeile)
  x_wert = float(punkt[0])
  y_wert = float(punkt[1])
  stddraw.point(x_wert, y_wert)
  stddraw.show(30)

stddraw.show()
```

```
mundhenk@ma3: ~
mm@home: python3 filter-jahr-txk.py < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
```



Das Programm male_punkte_A.py

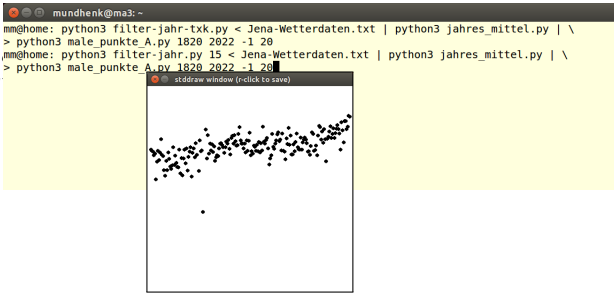
```
#-------------------------------------------------------------------------------
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als Punkte auf eine Leinwand mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------
import sys, stddraw
# Richte die Achsen der Leinwand ein und bereite den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.01)

# Standard input liefert zeilenweise x- und y-Werte von Punkt
# Jeder dieser Punkte wird gemalt, die Leinwand wird angezeig
# und anschließend wird ein bisschen gewartet.
for zeile in sys.stdin:
  punkt = str.split(zeile)
  x_wert = float(punkt[0])
  y_wert = float(punkt[1])
  stddraw.point(x_wert, y_wert)
  stddraw.show(30)

stddraw.show()
```

```
mundhenk@ma3: ~
mm@home: python3 filter-jahr-txk.py < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
mm@home: python3 filter-jahr.py 15 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
```

Das Programm `filter-jahr.py` ist eine Variante von `filter-jahr-txk.py`.
`filter-jahr-txk.py < Jena-Wetterdaten.txt` gibt aus jeder Eingabezeile das Jahr und die Tageshöchsttemperatur aus.
`filter-jahr.py n < Jena-Wetterdaten.txt` gibt aus jeder Eingabezeile das Jahr und Spalte `n` aus.

Das Programm male_punkte_A.py

```
#-------------------------------------------------------------------------------
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als Punkte auf eine Leinwand mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------
import sys, stddraw
# Richte die Achsen der Leinwand ein und bereite den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.01)

# Standard input liefert zeilenweise x- und y-Werte von Punkt
# Jeder dieser Punkte wird gemalt, die Leinwand wird angezeig
# und anschließend wird ein bisschen gewartet.
for zeile in sys.stdin:
  punkt = str.split(zeile)
  x_wert = float(punkt[0])
  y_wert = float(punkt[1])
  stddraw.point(x_wert, y_wert)
  stddraw.show(30)

stddraw.show()
```



Das Programm `filter-jahr.py` ist eine Variante von `filter-jahr-txk.py`.
`filter-jahr-txk.py < Jena-Wetterdaten.txt` gibt aus jeder Eingabezeile das Jahr und die Tageshöchsttemperatur aus.
`filter-jahr.py n < Jena-Wetterdaten.txt` gibt aus jeder Eingabezeile das Jahr und Spalte `n` aus.

Das Programm male_punkte_A.py

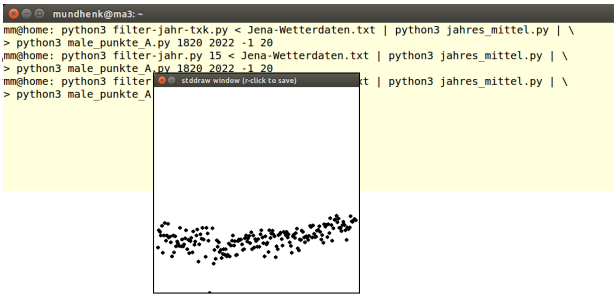
```
#-------------------------------------------------------------------------------
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als Punkte auf eine Leinwand mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------
import sys, stddraw
# Richte die Achsen der Leinwand ein und bereite den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.01)

# Standard input liefert zeilenweise x- und y-Werte von Punkt
# Jeder dieser Punkte wird gemalt, die Leinwand wird angezeig
# und anschließend wird ein bisschen gewartet.
for zeile in sys.stdin:
  punkt = str.split(zeile)
  x_wert = float(punkt[0])
  y_wert = float(punkt[1])
  stddraw.point(x_wert, y_wert)
  stddraw.show(30)

stddraw.show()
```

```
mundhenk@ma3: ~
mm@home: python3 filter-jahr-txk.py < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
mm@home: python3 filter-jahr.py 15 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
mm@home: python3 filter-jahr.py 16 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
```

Das Programm `filter-jahr.py` ist eine Variante von `filter-jahr-txk.py`.
`filter-jahr-txk.py < Jena-Wetterdaten.txt` gibt aus jeder Eingabezeile das Jahr und die Tageshöchsttemperatur aus.
`filter-jahr.py n < Jena-Wetterdaten.txt` gibt aus jeder Eingabezeile das Jahr und Spalte `n` aus.

Das Programm male_punkte_A.py

```
#-------------------------------------------------------------------------------
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als Punkte auf eine Leinwand mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------
import sys, stddraw
# Richte die Achsen der Leinwand ein und bereite den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.01)

# Standard input liefert zeilenweise x- und y-Werte von Punkt
# Jeder dieser Punkte wird gemalt, die Leinwand wird angezeig
# und anschließend wird ein bisschen gewartet.
for zeile in sys.stdin:
  punkt = str.split(zeile)
  x_wert = float(punkt[0])
  y_wert = float(punkt[1])
  stddraw.point(x_wert, y_wert)
  stddraw.show(30)

stddraw.show()
```



Das Programm `filter-jahr.py` ist eine Variante von `filter-jahr-txk.py`.
`filter-jahr-txk.py < Jena-Wetterdaten.txt` gibt aus jeder Eingabezeile das Jahr und die Tageshöchsttemperatur aus.
`filter-jahr.py n < Jena-Wetterdaten.txt` gibt aus jeder Eingabezeile das Jahr und Spalte `n` aus.

Das Programm male_punkte_A.py

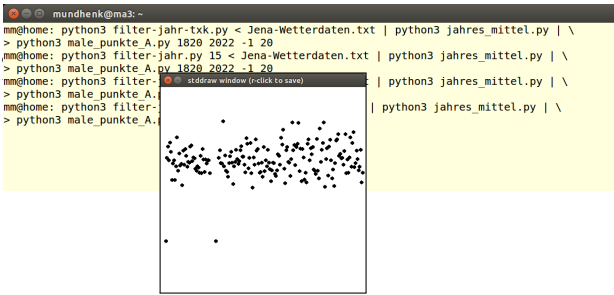
```
#-------------------------------------------------------------------------------
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als Punkte auf eine Leinwand mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------
import sys, stddraw
# Richte die Achsen der Leinwand ein und bereite den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.01)

# Standard input liefert zeilenweise x- und y-Werte von Punkt
# Jeder dieser Punkte wird gemalt, die Leinwand wird angezeig
# und anschließend wird ein bisschen gewartet.
for zeile in sys.stdin:
  punkt = str.split(zeile)
  x_wert = float(punkt[0])
  y_wert = float(punkt[1])
  stddraw.point(x_wert, y_wert)
  stddraw.show(30)

stddraw.show()
```

```
mundhenk@ma3: ~
mm@home: python3 filter-jahr-txk.py < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
mm@home: python3 filter-jahr.py 15 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
mm@home: python3 filter-jahr.py 16 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
mm@home: python3 filter-jahr.py 6 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 3
```

Das Programm `filter-jahr.py` ist eine Variante von `filter-jahr-txk.py`.
`filter-jahr-txk.py < Jena-Wetterdaten.txt` gibt aus jeder Eingabezeile das Jahr und die Tageshöchsttemperatur aus.
`filter-jahr.py n < Jena-Wetterdaten.txt` gibt aus jeder Eingabezeile das Jahr und Spalte `n` aus.

Das Programm male_punkte_A.py

```
#-------------------------------------------------------------------------------
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als Punkte auf eine Leinwand mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------
import sys, stddraw
# Richte die Achsen der Leinwand ein und bereite den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.01)

# Standard input liefert zeilenweise x- und y-Werte von Punkt
# Jeder dieser Punkte wird gemalt, die Leinwand wird angezeig
# und anschließend wird ein bisschen gewartet.
for zeile in sys.stdin:
  punkt = str.split(zeile)
  x_wert = float(punkt[0])
  y_wert = float(punkt[1])
  stddraw.point(x_wert, y_wert)
  stddraw.show(30)

stddraw.show()
```



Das Programm `filter-jahr.py` ist eine Variante von `filter-jahr-txk.py`.
`filter-jahr-txk.py < Jena-Wetterdaten.txt` gibt aus jeder Eingabezeile das Jahr und die Tageshöchsttemperatur aus.
`filter-jahr.py n < Jena-Wetterdaten.txt` gibt aus jeder Eingabezeile das Jahr und Spalte `n` aus.

Das Programm male_punkte_B.py

```
#-------------------------------------------------------------------------------
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als Punkte auf eine Leinwand mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------
import sys, stddraw
# Richte die Achsen der Leinwand ein und bereite den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.01)

# Standard input liefert zeilenweise x- und y-Werte von Punkten.
# Jeder dieser Werte wird einem Array der x-Werte bzw. in einem Array der y-Werte gespeichert.
xwerte = []
ywerte = []
for zeile in sys.stdin:          # Lies jede Zeile von stdin ein.
  punkt = str.split(zeile)       # Die Zeile wird in die beiden Koordinaten (string) geteilt.
  xwerte += [ float(punkt[0]) ]  # Die x-Koordinate wird an das Array der x-Werte angehängt.
  ywerte += [ float(punkt[1]) ]  # Die y-Koordinate wird an das Array der y-Werte angehängt.

for i in range(len(xwerte)):     # Alle Indizes mit x-Werten (und y-Werten) werden durchlaufen.
  stddraw.point(xwerte[i], ywerte[i])  # Jeder Punkt wird gemalt.

# Zeige das Bild an.
stddraw.show()
```

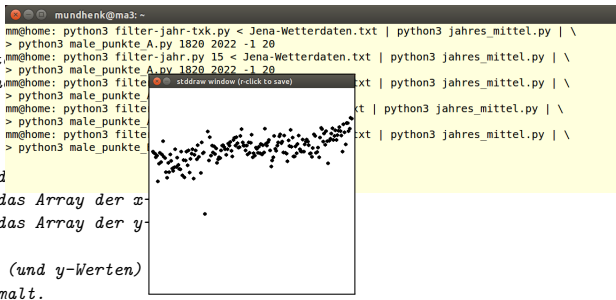
Das Programm male_punkte_B.py

```
#-------------------------------------------------------------------------------------------
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als Punkte auf eine Leinwand mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------------------
import sys, stddraw
# Richte die Achsen der Leinwand ein und bereite den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.01)

# Standard input liefert zeilenweise x- und y-Werte von Punkt
# Jeder dieser Werte wird einem Array der x-Werte bzw. in ein
xwerte = []
ywerte = []
for zeile in sys.stdin:          # Lies jede Zeile von stdin
  punkt = str.split(zeile)       # Die Zeile wird in die beid
  xwerte += [ float(punkt[0]) ]  # Die x-Koordinate wird an das Array der x-Werte angehängt.
  ywerte += [ float(punkt[1]) ]  # Die y-Koordinate wird an das Array der y-Werte angehängt.

for i in range(len(xwerte)):     # Alle Indizes mit x-Werten (und y-Werten) werden durchlaufen.
  stddraw.point(xwerte[i], ywerte[i])  # Jeder Punkt wird gemalt.

# Zeige das Bild an.
stddraw.show()
```

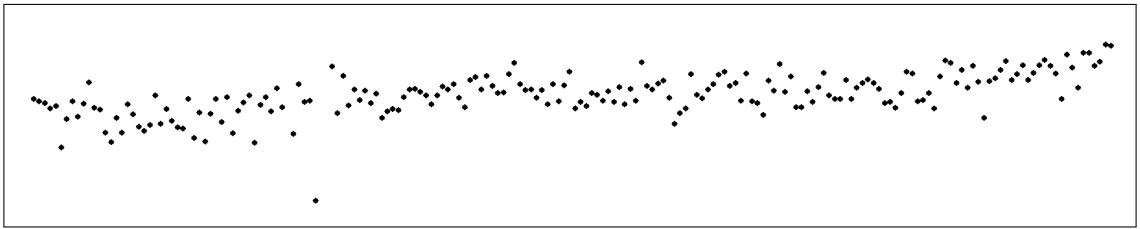
```
mundhenk@ma3: ~
mm@home: python3 filter-jahr-txk.py < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
mm@home: python3 filter-jahr.py 15 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
mm@home: python3 filter-jahr.py 16 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 20
mm@home: python3 filter-jahr.py 6 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_A.py 1820 2022 -1 3
mm@home: python3 filter-jahr.py 15 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_punkte_B.py 1820 2022 -1 20
```

Das Programm male_punkte_B.py

```
#-------------------------------------------------------------------------------------------
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als Punkte auf eine Leinwand mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------------------
import sys, stddraw
# Richte die Achsen der Leinwand ein und bereite den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.01)

# Standard input liefert zeilenweise x- und y-Werte von Punkt
# Jeder dieser Werte wird einem Array der x-Werte bzw. in ein
xwerte = []
ywerte = []
for zeile in sys.stdin:          # Lies jede Zeile von stdin
  punkt = str.split(zeile)       # Die Zeile wird in die beid
  xwerte += [ float(punkt[0]) ]  # Die x-Koordinate wird an das Array der x
  ywerte += [ float(punkt[1]) ]  # Die y-Koordinate wird an das Array der y

for i in range(len(xwerte)):     # Alle Indizes mit x-Werten (und y-Werten)
  stddraw.point(xwerte[i], ywerte[i])  # Jeder Punkt wird gemalt.

# Zeige das Bild an.
stddraw.show()
```

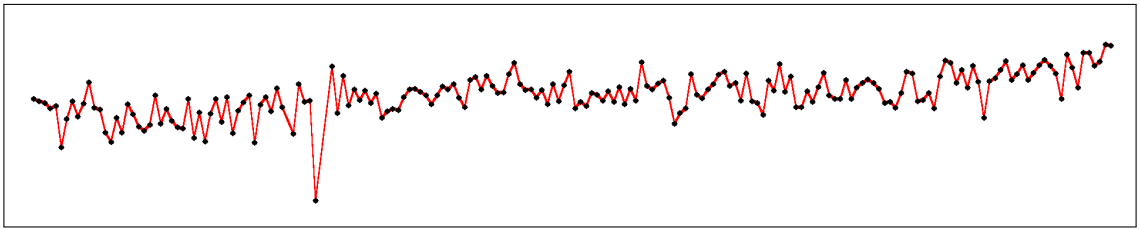


Aufeinanderfolgende Messwerte werden als Strich gemalt: die Idee



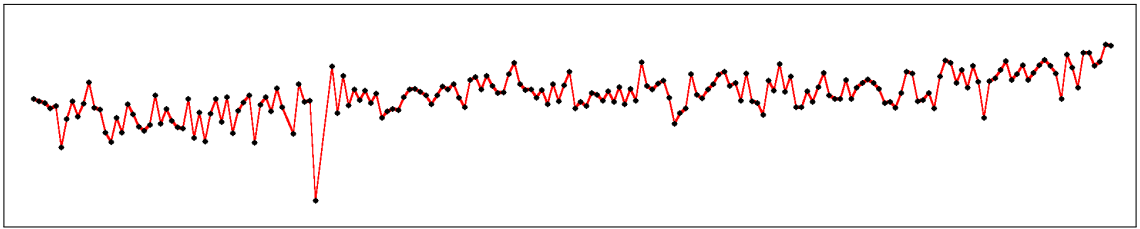
- Lies alle Zeilen *Jahr Temperatur* ein.
- Fasse jede Zeile als x- und y-Wert eines Punktes auf.
- Male zwei aufeinanderfolgende Punkte als Strich.

Aufeinanderfolgende Messwerte werden als Strich gemalt: die Idee



- Lies alle Zeilen *Jahr Temperatur* ein.
- Fasse jede Zeile als x- und y-Wert eines Punktes auf.
- Male zwei aufeinanderfolgende Punkte als Strich.

Aufeinanderfolgende Messwerte werden als Strich gemalt: die Idee



- Lies alle Zeilen *Jahr Temperatur* ein.
- Fasse jede Zeile als x- und y-Wert eines Punktes auf.
- Male zwei aufeinanderfolgende Punkte als Strich.

Das Programm male_kurve.py

```
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als mit Strichen verbundene Punkte auf eine Leinwand  mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------------------------------------
import sys, stddraw
# Bereite die Leinwand und den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.14)

# Wir lesen zuerst alle Punkte von standard input.
xwerte = []
ywerte = []
for zeile in sys.stdin:
  punkt = str.split(zeile)
  xwerte += [ float(punkt[0]) ]
  ywerte += [ float(punkt[1]) ]

# Wir zeichnen jeweils einen Strich zwischen zwei aufeinanderfolgenden Punkten.
# Da der letzte Punkt keinen "Nachfolger" hat, ist er kein Startpunkt eines Strichs.
for i in range(len(xwerte)-1):
  stddraw.line( xwerte[i], ywerte[i], xwerte[i+1], ywerte[i+1] )

# Zeige das Bild an.
stddraw.show()
```

Das Programm male_kurve.py

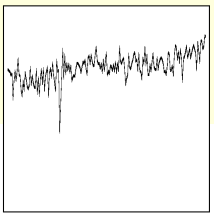
```
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als mit Strichen verbundene Punkte auf eine Leinwand  mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------------------------------------
import sys, stddraw
# Bereite die Leinwand und den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.14)

# Wir lesen zuerst alle Punkte von standard input.
xwerte = []
ywerte = []
for zeile in sys.stdin:
  punkt = str.split(zeile)
  xwerte += [ float(punkt[0]) ]
  ywerte += [ float(punkt[1]) ]

# Wir zeichnen jeweils einen Strich zwischen zwei aufeinanderfolgenden Punkten.
# Da der letzte Punkt keinen "Nachfolger" hat, ist er kein Startpunkt eines Strichs.
for i in range(len(xwerte)-1):
  stddraw.line( xwerte[i], ywerte[i], xwerte[i+1], ywerte[i+1] )

# Zeige das Bild an.
stddraw.show()
```

```
mundhenk@ma3: ~
mm@home: python3 filter-jahr.py 15 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_kurve.py 1820 2022 -1 20
```

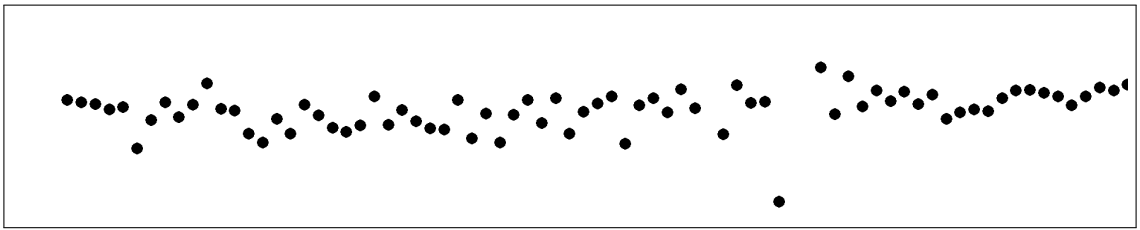
Das Programm male_kurve.py

```
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als mit Strichen verbundene Punkte auf eine Leinwand  mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------------------------------------
import sys, stddraw
# Bereite die Leinwand und den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.14)

# Wir lesen zuerst alle Punkte von standard input.
xwerte = []
ywerte = []
for zeile in sys.stdin:
  punkt = str.split(zeile)
  xwerte += [ float(punkt[0]) ]
  ywerte += [ float(punkt[1]) ]

# Wir zeichnen jeweils einen Strich zwischen zwei aufeinanderfolgenden Punkten.
# Da der letzte Punkt keinen "Nachfolger" hat, ist er kein Startpunkt eines Strichs.
for i in range(len(xwerte)-1):
  stddraw.line( xwerte[i], ywerte[i], xwerte[i+1], ywerte[i+1] )

# Zeige das Bild an.
stddraw.show()
```

```
mundhenk@ma3: ~
mm@home: python3 filter-jahr.py 15 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_kurve.py 1820 2022 -1 20
```

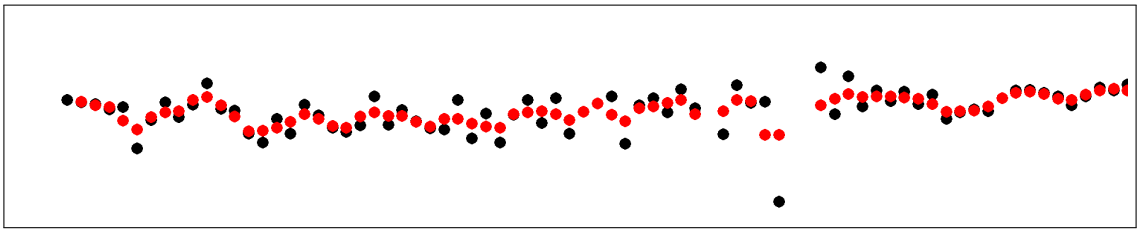


Die Kurve der Messwerte wird geglättet: die Idee



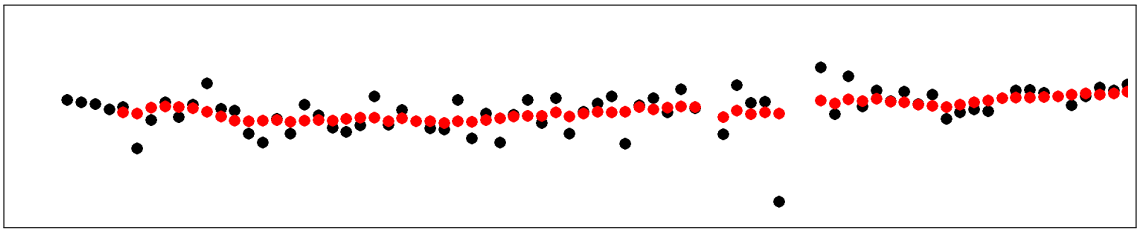
- Lies alle Zeilen *Jahr Temperatur* ein.
- Fasse jede Zeile als x- und y-Wert eines Punktes auf.
- Berechne für jeden y-Wert den Durchschnitt seiner Nachbarschaft (einschließlich ihm selbst).
- Male eine Kurve aus den x-Werten und dem jeweiligen Durchschnitts-Wert.

Die Kurve der Messwerte wird geglättet: die Idee



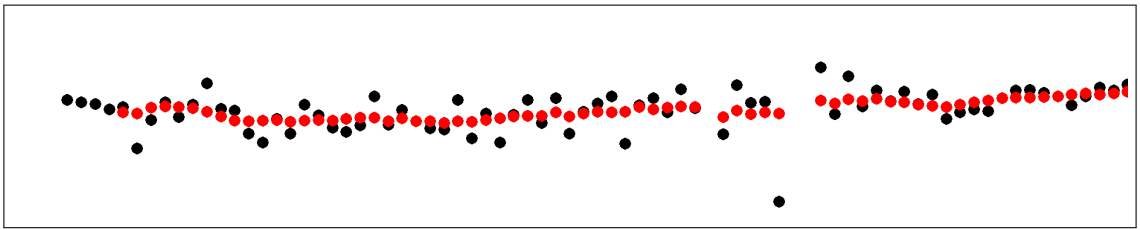
- Lies alle Zeilen *Jahr Temperatur* ein.
- Fasse jede Zeile als x- und y-Wert eines Punktes auf.
- Berechne für jeden y-Wert den Durchschnitt seiner Nachbarschaft (einschließlich ihm selbst).
- Male eine Kurve aus den x-Werten und dem jeweiligen Durchschnitts-Wert.

Die Kurve der Messwerte wird geglättet: die Idee



- Lies alle Zeilen *Jahr Temperatur* ein.
- Fasse jede Zeile als x- und y-Wert eines Punktes auf.
- Berechne für jeden y-Wert den Durchschnitt seiner Nachbarschaft (einschließlich ihm selbst).
- Male eine Kurve aus den x-Werten und dem jeweiligen Durchschnitts-Wert.

Die Kurve der Messwerte wird geglättet: die Idee



- Lies alle Zeilen *Jahr Temperatur* ein.
- Fasse jede Zeile als x- und y-Wert eines Punktes auf.
- Berechne für jeden y-Wert den Durchschnitt seiner Nachbarschaft (einschließlich ihm selbst).
- Male eine Kurve aus den x-Werten und dem jeweiligen Durchschnitts-Wert.

Die Umsetzung der Idee

Wir wollen für jeden y-Wert den Durchschnitt seiner Nachbarschaft aus z.B. 9 Nachbarn berechnen.

- Nimm die 9er-Nachbarschaft um Index `i` aus Array `ywerte` – das ist `ywerte[i-4:i+5]`.
- Berechne den Durchschnitt dieser Werte – das ist `sum(ywerte[i-4:i+5])/9`

- Beachte, dass diese Werte nur für den Indexbereich $4 \ldots \ell - 4$ berechnet werden können, wobei ℓ der größte Index im Array `ywerte` ist.
- Wenn die Striche gemalt werden, werden nur die Punkte bis zum *vorletzten* Punkt mit 9 Nachbarn als Startpunkte benutzt.
  Also ist die Schleife dann `for i in range(4,len(ywerte)-5)`.

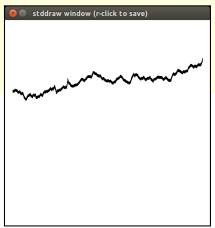
Das Programm male_glatte_kurve.py

```
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als mit Strichen verbundene Punkte auf eine Leinwand  mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------------------------------------
import sys, stddraw
# Bereite die Leinwand und den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.14)

# Wir lesen zuerst alle Punkte von standard input.
xwerte = []
ywerte = []
for zeile in sys.stdin:
  punkt = str.split(zeile)
  xwerte += [ float(punkt[0]) ]
  ywerte += [ float(punkt[1]) ]

# Für jeden Punkt rechnen wir den y-Wert als Durchschnitt seiner y-Nachbarschaft
# der Größe 9 aus.
# Wir zeichnen jeweils einen Strich zwischen zwei aufeinanderfolgenden Punkten.
for i in range(4,len(xwerte)-5):
  stddraw.line( xwerte[i], sum(ywerte[i-4:i+5])/9, xwerte[i+1], sum(ywerte[i-3:i+6])/9 )

# Zeige das Bild an.
stddraw.show()
```

Das Programm male_glatte_kurve.py

```
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als mit Strichen verbundene Punkte auf eine Leinwand  mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------------------------------
import sys, stddraw
# Bereite die Leinwand und den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.14)

# Wir lesen zuerst alle Punkte von standard input.
xwerte = []
ywerte = []
for zeile in sys.stdin:
  punkt = str.split(zeile)
  xwerte += [ float(punkt[0]) ]
  ywerte += [ float(punkt[1]) ]

# Für jeden Punkt rechnen wir den y-Wert als Durchschnitt seiner y-Nachbarschaft
# der Größe 9 aus.
# Wir zeichnen jeweils einen Strich zwischen zwei aufeinanderfolgenden Punkten.
for i in range(4,len(xwerte)-5):
  stddraw.line( xwerte[i], sum(ywerte[i-4:i+5])/9, xwerte[i+1], sum(ywerte[i-3:i+6])/9 )

# Zeige das Bild an.
stddraw.show()
```

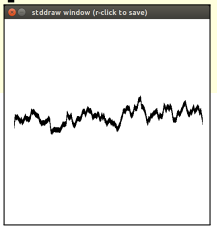
```
mundhenk@ma3: ~
mm@home: python3 filter-jahr.py 15 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_kurve.py 1820 2022 -1 20
mm@home: python3 filter-jahr.py 15 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_glatte_kurve.py 1820 2022 -1 20
```



Das Programm male_glatte_kurve.py

```
# Lies x0, x1, y0 und y1 von der Kommandozeile. Lies Wertepaare x,y von standard input
# und male sie als mit Strichen verbundene Punkte auf eine Leinwand  mit x-Achse x0..x1 und y-Achse y0..y1.
#-------------------------------------------------------------------------------------------------------
import sys, stddraw
# Bereite die Leinwand und den Stift vor.
stddraw.setXscale(float(sys.argv[1]), float(sys.argv[2]))
stddraw.setYscale(float(sys.argv[3]), float(sys.argv[4]))
stddraw.setPenRadius(0.14)

# Wir lesen zuerst alle Punkte von standard input.
xwerte = []
ywerte = []
for zeile in sys.stdin:
  punkt = str.split(zeile)
  xwerte += [ float(punkt[0]) ]
  ywerte += [ float(punkt[1]) ]

# Für jeden Punkt rechnen wir den y-Wert als Durchschnitt seiner y-Nachbarschaft
# der Größe 9 aus.
# Wir zeichnen jeweils einen Strich zwischen zwei aufeinanderfolgenden Punkten.
for i in range(4,len(xwerte)-5):
  stddraw.line( xwerte[i], sum(ywerte[i-4:i+5])/9, xwerte[i+1], sum(ywerte[i-3:i+6])/9 )

# Zeige das Bild an.
stddraw.show()
```

```
mundhenk@ma3: ~
mm@home: python3 filter-jahr.py 15 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_kurve.py 1820 2022 -1 20
mm@home: python3 filter-jahr.py 15 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_glatte_kurve.py 1820 2022 -1 20
mm@home: python3 filter-jahr.py 6 < Jena-Wetterdaten.txt | python3 jahres_mittel.py | \
> python3 male_glatte_kurve.py 1820 2022 0 3
```



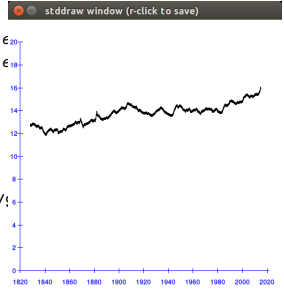
Und abschließend zeichnen wir noch beschriftete Achsen dazu.

Programm male_kurve_mit_achsen.py

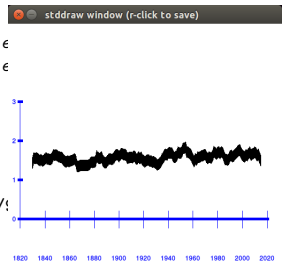
```
# Lies xmin, xmax, ymin und ymax von der Kommandozeile.
# Lies Wertepaare x,y von standard input und male sie als geglätte Kurve eine Leinwand
# mit x-Achse xmin..xmax und y-Achse ymin..ymax.
#-------------------------------------------------------------------------------
import sys, stddraw
# Wir lesen die Punkte von standard input.
# Wir schreiben alle x-Werte in ein Array xwerte, und alle y-Werte in ein Array ywerte.
# Dann ist (xwerte[i],ywerte[i]) jeweils ein Punkt.
xwerte = []
ywerte = []
for zeile in sys.stdin:
  punkt = str.split(zeile)
  xwerte += [ float(punkt[0]) ]
  ywerte += [ float(punkt[1]) ]

# Um die Leinwand vorzubereiten,
# lesen wir von der Kommandozeile die Längen der x- und der y-Achse.
xmin = float(sys.argv[1])
xmax = float(sys.argv[2])
ymin = float(sys.argv[3])
ymax = float(sys.argv[4])
# Zum Malen der Achsen und deren Beschriftung brauchen wir an jeder Seite etwas Rand.
stddraw.setXscale(xmin-10, xmax+10)
stddraw.setYscale(ymin-2, ymax+2)
stddraw.setPenRadius(0.04)
```

```
# Male die x-Achse und die y-Achse des Koordinatensystems (in blau).
stddraw.setPenColor(stddraw.BLUE)
stddraw.line(xmin,ymin, xmax,ymin)
stddraw.line(xmin,ymin, xmin,ymax)

# Beschrifte die Achsen des Koordinatensystems.
if ymax-ymin>=10: d=int(ymax-ymin)//10 # bestimme den Abstand der kleinen Striche auf der y-Achse
else: d=1

for i in range(int(ymin), int(ymax+1), d):  # beschrifte die y-Achse
  stddraw.line(xmin-2,i, xmin+2,i)    # male einen kleinen waagerechten Strich über die y-Achse in Höhe i
  stddraw.text(xmin-5, i, str(i))     # schreibe i links neben die y-Achse in Höhe i

for i in range(int(xmin), int(xmax+1), int(xmax-xmin)//10): # beschrifte die x-Achse
  stddraw.line(i,ymin-0.2, i, ymin+0.2) # male einen kleinen senkrechten Strich an Stelle i der x-Achse
  stddraw.text(i, ymin-1, str(i))       # schreibe i etwas unter die x-Achse an Stelle i

# Die Linien zwischen aufeinanderfolgenden (x,y)-Paaren werden gemalt.
stddraw.setPenColor(stddraw.BLACK)
stddraw.setPenRadius(0.06)
for i in range(len(xwerte)-1):
  stddraw.line( xwerte[i], sum(ywerte[i-4:i+5])/9, xwerte[i+1], sum(ywerte[i-3:i+6])/9 )

stddraw.show()
#--------------------------------------------------------------------------------
```

```
# Male die x-Achse und die y-Achse des Koordinatensystems (in blau).
stddraw.setPenColor(stddraw.BLUE)
stddraw.line(xmin,ymin, xmax,ymin)
stddraw.line(xmin,ymin, xmin,ymax)

# Beschrifte die Achsen des Koordinatensystems.
if ymax-ymin>=10: d=int(ymax-ymin)//10 # bestimme den Abstand der kleinen Striche auf der y-Achse
else: d=1

for i in range(int(ymin), int(ymax+1), d):  # beschrifte die y-Achse
  stddraw.line(xmin-2,i, xmin+2,i)     # male einen kleinen waagerechten Strich über die y-Achse in Höhe i
  stddraw.text(xmin-5, i, str(i))      # schreibe i links neben die y-Achse in Höhe i

for i in range(int(xmin), int(xmax+1), int(xmax-xmin)//10): # beschrifte die x-Achse
  stddraw.line(i,ymin-0.2, i, ymin+0.2) # male einen kleinen senkrechten Strich an Ste
  stddraw.text(i, ymin-1, str(i))       # schreibe i etwas unter die x-Achse an Stelle

# Die Linien zwischen aufeinanderfolgenden (x,y)-Paaren werden gemalt.
stddraw.setPenColor(stddraw.BLACK)
stddraw.setPenRadius(0.06)
for i in range(len(xwerte)-1):
  stddraw.line( xwerte[i], sum(ywerte[i-4:i+5])/9, xwerte[i+1], sum(ywerte[i-3:i+6])/9

stddraw.show()
#------------------------------------------------------------------------------
# python3 filter-jahr.py 15 < Jena-Wetterdaten.txt | python3 jahres_mittel.py  | \
# python3 male_kurve_mit_achsen.py 1820 2022 0 20
#
```



```
# Male die x-Achse und die y-Achse des Koordinatensystems (in blau).
stddraw.setPenColor(stddraw.BLUE)
stddraw.line(xmin,ymin, xmax,ymin)
stddraw.line(xmin,ymin, xmin,ymax)

# Beschrifte die Achsen des Koordinatensystems.
if ymax-ymin>=10: d=int(ymax-ymin)//10 # bestimme den Abstand der kleinen Striche auf der y-Achse
else: d=1

for i in range(int(ymin), int(ymax+1), d):  # beschrifte die y-Achse
  stddraw.line(xmin-2,i, xmin+2,i)     # male einen kleinen waagerechten Strich über die y-Achse in Höhe i
  stddraw.text(xmin-5, i, str(i))      # schreibe i links neben die y-Achse in Höhe i

for i in range(int(xmin), int(xmax+1), int(xmax-xmin)//10): # beschrifte die x-Achse
  stddraw.line(i,ymin-0.2, i, ymin+0.2) # male einen kleinen senkrechten Strich an Ste
  stddraw.text(i, ymin-1, str(i))       # schreibe i etwas unter die x-Achse an Stelle

# Die Linien zwischen aufeinanderfolgenden (x,y)-Paaren werden gemalt.
stddraw.setPenColor(stddraw.BLACK)
stddraw.setPenRadius(0.06)
for i in range(len(xwerte)-1):
  stddraw.line( xwerte[i], sum(ywerte[i-4:i+5])/9, xwerte[i+1], sum(ywerte[i-3:i+6])/9

stddraw.show()
#-------------------------------------------------------------------------------
# python3 filter_jahr.py 6 < Jena-Wetterdaten.txt | python3 jahres_mittel.py  | \
# python3 male_kurve_mit_achsen.py 1820 2022 0 3
#
```



7 Malen des Graphs einer Funktion

Zum Malen des Graphs einer Funktion muss das Programm die Argument-/Wert-Paare, die `male_kurve_B.py` von standard input einliest, selbst erzeugen.

Wenn der Graph der Funktion $f(x)$ im Bereich $l \leqslant x \leqslant r$ gemalt werden soll,

- muss man festlegen, wieviele Funktionswerte man im Bereich $l \leqslant x \leqslant r$ berechnen will (das Intervall $l \ldots r$ wird in gleichmäßige Abschnitte unterteilt),

- die Argument-Wert-Paare $(x, f(x))$ berechnen und
- die Funktionswerte aufeinanderfolgender Argumente durch eine Linie verbinden.

Man beachte, dass stets nur eine Näherung berechnet wird,
deren Qualität von der Anzahl der berechneten Funktionswerte abhängt.

7 Malen des Graphs einer Funktion

Zum Malen des Graphs einer Funktion muss das Programm die Argument-/Wert-Paare, die `male_kurve_B.py` von standard input einliest, selbst erzeugen.

Wenn der Graph der Funktion $f(x)$ im Bereich $l \leqslant x \leqslant r$ gemalt werden soll,

- muss man festlegen, wieviele Funktionswerte man im Bereich $l \leqslant x \leqslant r$ berechnen will (das Intervall $l \ldots r$ wird in gleichmäßige Abschnitte unterteilt),

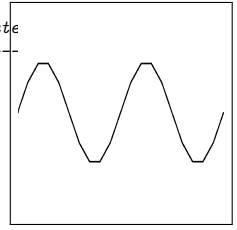
> Wenn Funktionswerte für n Argumente im Intervall $l \leqslant x \leqslant r$ berechnet werden sollen, haben die Argumente Abstand $\frac{l-r}{n-1}$ und sind $l, l+1 \cdot \frac{r-l}{n-1}, l+2 \cdot \frac{r-l}{n-1}, \ldots, \underbrace{l+(n-1) \cdot \frac{r-l}{n-1}}_{=r}$.

- die Argument-Wert-Paare $(x, f(x))$ berechnen und
- die Funktionswerte aufeinanderfolgender Argumente durch eine Linie verbinden.

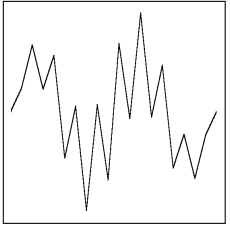
Man beachte, dass stets nur eine Näherung berechnet wird,
deren Qualität von der Anzahl der berechneten Funktionswerte abhängt.

```
#-----------------------------------------------------------------------------------------------------------------
# functiongraph.py (so ähnlich wie im Buch)
#-----------------------------------------------------------------------------------------------------------------
# Der Graph der Funktion f(x) = sin(4x)+sin(20x) wird im Intervall -pi/2..pi/2 gemalt.
# Von der Kommandozeile wird die Sampling-Dichte n (d.h. die Anzahl der zu berechnenden Punkte) eingelesen.
#-----------------------------------------------------------------------------------------------------------------
import math, sys, stddraw
# Lies n von der Kommandozeile
# und erzeuge Arrays für die x-Werte und die y-Werte des Graph der Funktion f.
n = int(sys.argv[1])
x_werte = []
y_werte = []
# Berechne für n x-Werte, die gleichmäßig über das Intervall -pi/2..pi/2 verteilt sind,
# jeweils den zugehörigen y-Wert f(x)=sin(4x)+sin(20x).
# Die x_werte sind -pi/2, -pi/2+1*pi/(n-1), -pi/2+2*pi/(n-1),..., -pi/2+(n-1)*pi/(n-1) .
# Speichere den x-Wert in x_werte und den zugehörigen y-Wert in y_werte.
# Am Ende gilt f(x_werte[i])==y_werte[i] für i in range(n).
for i in range(n):
    x_werte += [ -math.pi/2 + i * math.pi/(n-1) ]
    y_werte += [ math.sin(4*x_werte[i]) + math.sin(20*x_werte[i]) ]
# Richte die Leinwand ein.
stddraw.setXscale(-math.pi/2, math.pi/2)
stddraw.setYscale(-2, 2)
# Male Striche zwischen aufeinanderfolgenden Punkten.
for i in range(n-1):
  stddraw.line(x_werte[i], y_werte[i], x_werte[i+1], y_werte[i+1])
# Zeige das Bild an.
stddraw.show()
```

```
#-------------------------------------------------------------------------------------------------------
# functiongraph.py (so ähnlich wie im Buch)
#-------------------------------------------------------------------------------------------------------
# Der Graph der Funktion f(x) = sin(4x)+sin(20x) wird im Intervall -pi/2..pi/2 gemalt.
# Von der Kommandozeile wird die Sampling-Dichte n (d.h. die Anzahl der zu berechnenden Punkte
#-------------------------------------------------------------------------------------------------------
import math, sys, stddraw
# Lies n von der Kommandozeile
# und erzeuge Arrays für die x-Werte und die y-Werte des Graph der Funktion f.
n = int(sys.argv[1])
x_werte = []
y_werte = []
# Berechne für n x-Werte, die gleichmäßig über das Intervall -pi/2..pi/2 verteilt sind,
# jeweils den zugehörigen y-Wert f(x)=sin(4x)+sin(20x).
# Die x_werte sind -pi/2, -pi/2+1*pi/(n-1), -pi/2+2*pi/(n-1),..., -pi/2+(n-1)*pi/(n-1) .
# Speichere den x-Wert in x_werte und den zugehörigen y-Wert in y_werte.
# Am Ende gilt f(x_werte[i])==y_werte[i] für i in range(n).
for i in range(n):
    x_werte += [ -math.pi/2 + i * math.pi/(n-1) ]
    y_werte += [ math.sin(4*x_werte[i]) + math.sin(20*x_werte[i]) ]
# Richte die Leinwand ein.
stddraw.setXscale(-math.pi/2, math.pi/2)
stddraw.setYscale(-2, 2)
# Male Striche zwischen aufeinanderfolgenden Punkten.
for i in range(n-1):
  stddraw.line(x_werte[i], y_werte[i], x_werte[i+1], y_werte[i+1])
# Zeige das Bild an.
stddraw.show()
```

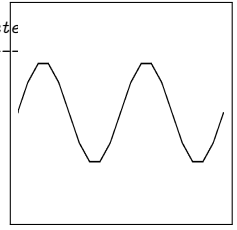


python3 functiongraph.py 21

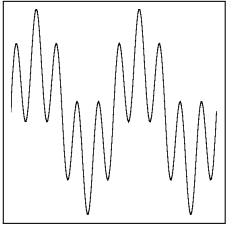


python3 functiongraph.py 20

```
#-------------------------------------------------------------------------------------------------------------
# functiongraph.py (so ähnlich wie im Buch)
#-------------------------------------------------------------------------------------------------------------
# Der Graph der Funktion f(x) = sin(4x)+sin(20x) wird im Intervall -pi/2..pi/2 gemalt.
# Von der Kommandozeile wird die Sampling-Dichte n (d.h. die Anzahl der zu berechnenden Punkte
#-------------------------------------------------------------------------------------------------------------
import math, sys, stddraw
# Lies n von der Kommandozeile
# und erzeuge Arrays für die x-Werte und die y-Werte des Graph der Funktion f.
n = int(sys.argv[1])
x_werte = []
y_werte = []
# Berechne für n x-Werte, die gleichmäßig über das Intervall -pi/2..pi/2 verteilt sind,
# jeweils den zugehörigen y-Wert f(x)=sin(4x)+sin(20x).
# Die x_werte sind -pi/2, -pi/2+1*pi/(n-1), -pi/2+2*pi/(n-1),..., -pi/2+(n-1)*pi/(n-1) .
# Speichere den x-Wert in x_werte und den zugehörigen y-Wert in y_werte.
# Am Ende gilt f(x_werte[i])==y_werte[i] für i in range(n).
for i in range(n):
    x_werte += [ -math.pi/2 + i * math.pi/(n-1) ]
    y_werte += [ math.sin(4*x_werte[i]) + math.sin(20*x_werte[i]) ]
# Richte die Leinwand ein.
stddraw.setXscale(-math.pi/2, math.pi/2)
stddraw.setYscale(-2, 2)
# Male Striche zwischen aufeinanderfolgenden Punkten.
for i in range(n-1):
  stddraw.line(x_werte[i], y_werte[i], x_werte[i+1], y_werte[i+1])
# Zeige das Bild an.
stddraw.show()
```

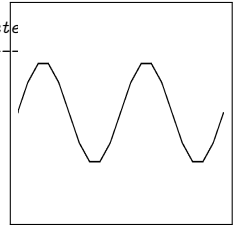


python3 functiongraph.py 21

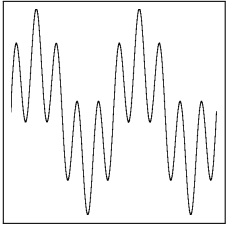


python3 functiongraph.py 200

```
#-------------------------------------------------------------------------------------------------------
# functiongraph.py (so ähnlich wie im Buch)
#-------------------------------------------------------------------------------------------------------
# Der Graph der Funktion f(x) = sin(4x)+sin(20x) wird im Intervall -pi/2..pi/2 gemalt.
# Von der Kommandozeile wird die Sampling-Dichte n (d.h. die Anzahl der zu berechnenden Punkte
#-------------------------------------------------------------------------------------------------------
import math, sys, stddraw
# Lies n von der Kommandozeile
# und erzeuge Arrays für die x-Werte und die y-Werte des Graph der Funktion f.
n = int(sys.argv[1])
x_werte = []
y_werte = []
# Berechne für n x-Werte, die gleichmäßig über das Intervall -pi/2..pi/2 verteilt sind,
# jeweils den zugehörigen y-Wert f(x)=sin(4x)+sin(20x).
# Die x_werte sind -pi/2, -pi/2+1*pi/(n-1), -pi/2+2*pi/(n-1),..., -pi/2+(n-1)*pi/(n-1) .
# Speichere den x-Wert in x_werte und den zugehörigen y-Wert in y_werte.
# Am Ende gilt f(x_werte[i])==y_werte[i] für i in range(n).
for i in range(n):
    x_werte += [ -math.pi/2 + i * math.pi/(n-1) ]
    y_werte += [ math.sin(4*x_werte[i]) + math.sin(20*x_werte[i]) ]
# Richte die Leinwand ein.
stddraw.setXscale(-math.pi/2, math.pi/2)
stddraw.setYscale(-2, 2)
# Male Striche zwischen aufeinanderfolgenden Punkten.
for i in range(n-1):
  stddraw.line(x_werte[i], y_werte[i], x_werte[i+1], y_werte[i+1])
# Zeige das Bild an.
stddraw.show()
```



python3 functiongraph.py 21



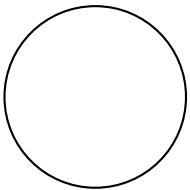
python3 functiongraph.py 512

8 Weitere `stddraw`-Funktionen: (1) Kreise malen

`stddraw.circle(x, y, r)` male einen Kreis

mit Mittelpunkt (x, y) und Radius r

```
import stddraw
x = 2
y = 3
r = 4
stddraw.setXscale(0, 6)
stddraw.setYscale(0, 6)

stddraw.circle(x, y, r)
stddraw.show()
```

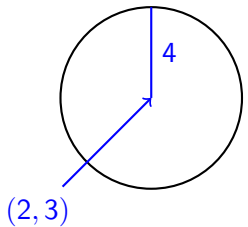


8 Weitere `stddraw`-Funktionen: (1) Kreise malen

`stddraw.circle(x, y, r)` male einen Kreis

mit Mittelpunkt (x, y) und Radius r

```
import stddraw
x = 2
y = 3
r = 4
stddraw.setXscale(0, 6)
stddraw.setYscale(0, 6)

stddraw.circle(x, y, r)
stddraw.show()
```

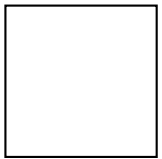


(2) Quadrate malen

`stddraw.square(x, y, r)` male ein Quadrat

mit Mittelpunkt (x, y) und Radius r

```
import stddraw
x = 2
y = 3
r = 4
stddraw.setXscale(0, 6)
stddraw.setYscale(0, 6)

stddraw.square(x, y, r)
stddraw.show()
```

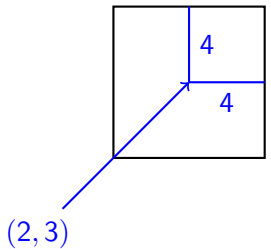


(2) Quadrate malen

`stddraw.square(x, y, r)` male ein Quadrat

mit Mittelpunkt (x, y) und Radius r

```
import stddraw
x = 2
y = 3
r = 4
stddraw.setXscale(0, 6)
stddraw.setYscale(0, 6)

stddraw.square(x, y, r)
stddraw.show()
```



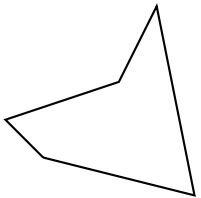
(3) Vielecke malen

`stddraw.polygon(x, y)` male ein Vieleck mit den Ecken

(`x[0]`, `y[0]`), (`x[1]`, `y[1]`), ...

(`x` und `y` sind Arrays)

```
import stddraw

x = [1,0,3,4,5]
y = [2,3,4,6,1]
stddraw.setXscale(0, 6)
stddraw.setYscale(0, 6)

stddraw.polygon(x,y)
stddraw.show()
```



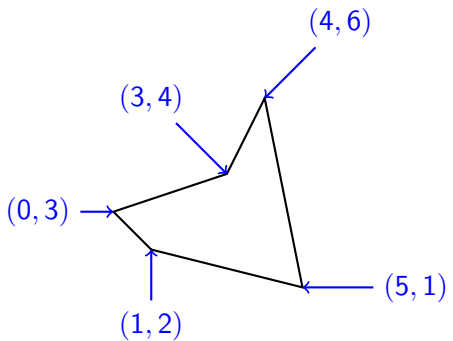
(3) Vielecke malen

`stddraw.polygon(x, y)` male ein Vieleck mit den Ecken

(`x[0]`, `y[0]`), (`x[1]`, `y[1]`), ...

(`x` und `y` sind Arrays)

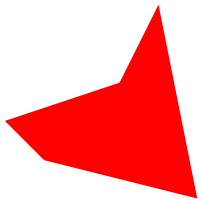
```
import stddraw

x = [1,0,3,4,5]
y = [2,3,4,6,1]
stddraw.setXscale(0, 6)
stddraw.setYscale(0, 6)

stddraw.polygon(x,y)
stddraw.show()
```



Alle Formen können auch gefüllt gemalt werden mittels
`stddraw.filledCircle(x,y,r)`, `stddraw.filledSquare(x,y,r)`,
`stddraw.filledPolygon(x,y)`.

`stddraw.filledPolygon(x, y)` male ein ausgefülltes Vieleck mit den Ecken
(`x[0]`, `y[0]`), (`x[1]`, `y[1]`), ... (`x` und `y` sind Arrays)

```
import stddraw

x = [1,0,3,4,5]
y = [2,3,4,6,1]
stddraw.setXscale(0, 6)
stddraw.setYscale(0, 6)
stddraw.setPenColor(stddraw.RED)

stddraw.filledPolygon(x,y)
stddraw.show()
```



Sonstige `stddraw`-Funktionen

Male ein Rechteck:

`stddraw.rectangle(x,y, w,h)`	male ein Rechteck mit Punkt `(x,y)` unten links, Breite `w` und Höhe `h`
`stddraw.filledRectangle(x,y, w,h)`	male ein gefülltes Rechteck ...

Schreibe Text:

`stddraw.text(x,y, s)`	schreibe String `s` zentriert an Punkt `(x,y)`
`stddraw.setFontSize(n)`	setze die Buchstabengröße auf `n` (Standard 12)
`stddraw.setFontFamily(f)`	setze den Zeichensatz auf `f` (Standard helvetica)

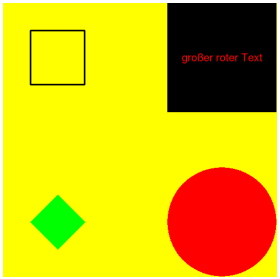
Lösche die Leinwand:

`stddraw.clear(c)`	lösche die Leinwand und färbe jedes Pixel mit Farbe `c`

Speichere das Bild:

`stddraw.save(name)`	speichere das Bild in Datei `name` (endet mit `.jpg` oder `.png`)

```
import stddraw

stddraw.clear(stddraw.YELLOW)

stddraw.square(0.2, 0.8, 0.1)

stddraw.filledSquare(0.8, 0.8, 0.2)

stddraw.setPenColor(stddraw.GREEN)
xd = [0.1, 0.2, 0.3, 0.2]
yd = [0.2, 0.3, 0.2, 0.1]
stddraw.filledPolygon(xd, yd)

stddraw.setPenColor(stddraw.RED)
stddraw.filledCircle(0.8, 0.2, 0.2)

stddraw.setFontSize(20)
stddraw.text(0.8,0.8, 'großer roter Text')

stddraw.show()
```

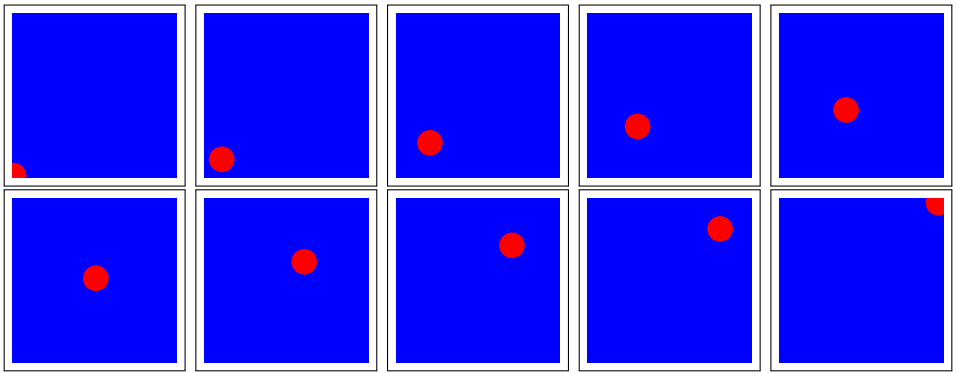


Bilder können mittels Klick auf die rechte Maustaste
oder mit `stddraw.save(dateiname)` gespeichert werden.

Die Dateinamenendung muss `.jpg` oder `.png` sein.

9 Bewegte Bilder

Zeigt man schnell hintereinander ähnliche Bilder, so sieht es aus, als ob sich etwas bewegt.



Programmier-Auftrag: lasse eine Kugel kreuz und quer über die Leinwand rollen (wie auf einem Billardtisch)

Die Kugel rollt immer im gleichen Tempo.

Wenn sie an eine Seite stößt, prallt sie ab und rollt weiter.

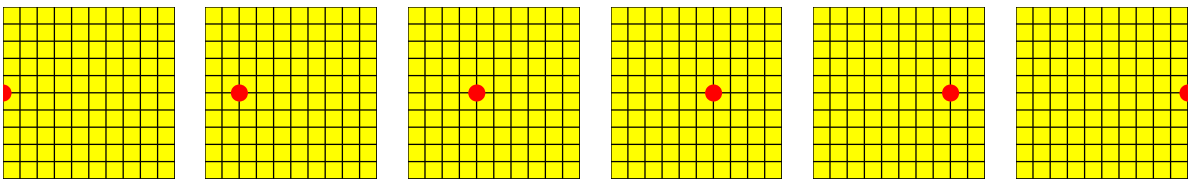
Schrittweise Ideen entwickeln

Wir wollen zuerst einfach mal eine Kugel rollen lassen, ohne dass sie abprallt.

Dazu wollen wir sie in der Mitte der Leinwand von links nach rechts rollen lassen.

Wir stellen uns die Leinwand mit x-Koordinaten $0 \ldots 1$ und y-Koordinaten $0 \ldots 1$ vor.
Die Ecke links unten hat die Koordinaten $(0, 0)$, die Ecke rechts oben hat die Koordinaten $(1, 1)$.
Die Mitte des linken Randes hat die Koordinaten $(0, 0.5)$.

Wenn die Kugel rollt, können wir uns folgende Momentaufnahmen vorstellen:



$(0.0, 0.5)$ $(0.2, 0.5)$ $(0.4, 0.5)$ $(0.6, 0.5)$ $(0.8, 0.5)$ $(1.0, 0.5)$

Die x-Koordinate der Position der Kugel wird von Bild zu Bild 0.2 größer.

Die grobe Struktur

Kugel rollt horizontal vom linken zum rechten Rand

1. Lies die Schrittweite der Bewegung der Kugel
   und die Dauer des Anzeigens eines Bildes ein.
2. Solange die Kugel nicht am rechten Rand angekommen ist:
   - **2.1** bewege sie einen Schritt nach rechts
   - **2.2** zeige das neue Bild an

Die grobe Struktur

Kugel rollt horizontal vom linken zum rechten Rand

1. Lies die Schrittweite der Bewegung der Kugel
   und die Dauer des Anzeigens eines Bildes ein.
2. Solange die Kugel nicht am rechten Rand angekommen ist:
   - **2.1** bewege sie einen Schritt nach rechts
   - **2.2** zeige das neue Bild an

Das Programm kugel_h1.py

```
#-------------------------------------------------------------------------------------------
# Eine Kugel rollt von links nach rechts durch das Bild.
#-------------------------------------------------------------------------------------------
import sys, stddraw

schrittweite = float(sys.argv[1])
anzeigedauer = int(sys.argv[2])
# Der Punkt (x_pos,y_pos) ist die aktuelle Position der Kugel.
x_pos = 0.0
y_pos = 0.5
# Die Kugel hat Radius 0.08 und ist rot.
RADIUS = 0.08
stddraw.setPenColor(stddraw.RED)

# Solange die Kugel nicht am rechten Rand ist, wird schrittweite zu x_pos addiert,
# die Leinwand gelöscht und mit der Kugel an der neuen Position angezeigt.
while x_pos<=1.0:
    # Bestimme die nächste Position der Kugel.
    x_pos = x_pos + schrittweite
    # Lösche die Leinwand, male die Kugel auf ihrer neuen Position und zeige die (gelbe) Leinwand an.
    stddraw.clear(stddraw.YELLOW)
    stddraw.filledCircle(x_pos, y_pos, RADIUS)
    stddraw.show(anzeigedauer)

stddraw.show(1000)   # Zeige das Bild noch 1 Sekunde an.
```

```
python3 kugel_h1.py 0.01 100
```

lässt die Kugel langsam aber etwas ruckelig rollen.
Bei einer Anzeigedauer von 100 ms werden 10 Bilder pro Sekunde gezeigt.
Das ist zu wenig, um es nicht ruckelig wirken zu lassen.

```
python3 kugel_h1.py 0.001 10
```

lässt die Kugel genauso schnell aber nicht ruckelig.
Bei einer Anzeigedauer von 10 ms werden 100 Bilder pro Sekunde gezeigt.

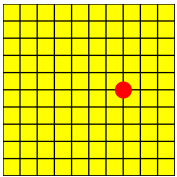
```
python3 kugel_h1.py 0.050 10
```

lässt die Kugel sehr schnell über die Leinwand fliegen.

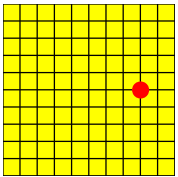
Nächster Ideenschritt: lasse die Kugel hin- und herprallen (auf der Horizontalen)

Bisher:
das Rollen der Kugel wird durch `schrittweite` gesteuert.
Aus der aktuellen Position `(x_pos,y_pos)` und `schrittweite`
wird die nächste Position `(x_pos+schrittweite, y_pos)` bestimmt.

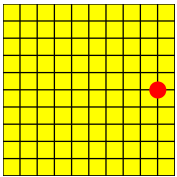
Was passiert, wenn die Kugel an den rechten Rand prallt?



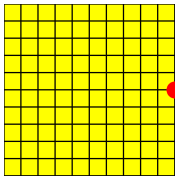
$(0.7, 0.5)$
`schrittweite`



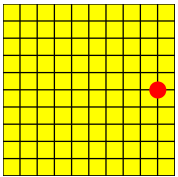
$(0.8, 0.5)$
0.1



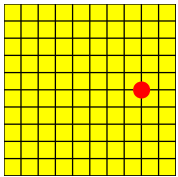
$(0.9, 0.5)$
0.1



$(1.0, 0.5)$
0.1



$(0.9, 0.5)$
-0.1



$(0.8, 0.5)$
-0.1

Das Vorzeichen von `schrittweite` ändert sich.
Wenn die Kugel an den linken Rand prallt, ändert sich das Vorzeichen wieder ...

Die grobe Struktur

Kugel prallt horizontal zwischen linkem und rechtem Rand hin und her

1. Lies die Schrittweite der Bewegung der Kugel
   und die Dauer des Anzeigens eines Bildes ein
2. wiederhole:
   - **2.1** falls die Kugel am linken oder rechten Rand angekommen ist,
     dann ändere das Vorzeichen der Schrittweite
   - **2.2** bewege die Kugel um eine Schrittweite auf der x-Achse
   - **2.3** zeige das neue Bild an

Die grobe Struktur

Kugel prallt horizontal zwischen linkem und rechtem Rand hin und her

1. Lies die Schrittweite der Bewegung der Kugel
und die Dauer des Anzeigens eines Bildes ein

2. wiederhole:

2.1 falls die Kugel am linken oder rechten Rand angekommen ist,
dann ändere das Vorzeichen der Schrittweite

2.2 bewege die Kugel um eine Schrittweite auf der x-Achse

2.3 zeige das neue Bild an

Das Programm kugel_h2.py

```
#-----------------------------------------------------------------------------------------------------
# Eine Kugel prallt zwischen dem linken und rechten Rand des Bildes hin und her.
#-----------------------------------------------------------------------------------------------------
import sys, stddraw
# Lies die Schrittweite, die die Kugel von Bild zu Bild macht, und die Anzeigedauer des Bildes ein.
schrittweite = float(sys.argv[1])
anzeigedauer = int(sys.argv[2])
# Der Punkt (x_pos,y_pos) ist die aktuelle Position der Kugel.
x_pos = 0.1
y_pos = 0.5
# Die Kugel hat Radius 0.08 und ist rot.
RADIUS = 0.08
stddraw.setPenColor(stddraw.RED)

# Die Kugel bewegt sich von Bild zu Bild schrittweite auf der x-Achse.
# Wenn der rechte oder der linke Rand erreicht wird, läuft die Kugel in die umgekehrte Richtung.
while True:
    # Bestimme die nächste Position der Kugel.
    if x_pos<=0 or x_pos>=1:
       schrittweite = -schrittweite
    x_pos = x_pos + schrittweite
    # Lösche die Leinwand, male die Kugel auf ihrer neuen Position und zeige die (gelbe) Leinwand an.
    stddraw.clear(stddraw.YELLOW)
    stddraw.filledCircle(x_pos, y_pos, RADIUS)
    stddraw.show(anzeigedauer)
```

Nächster Ideenschritt: die Kugel läuft in eine beliebige Richtung

Wir müssen das, was wir bisher nur für die x-Koordinate der Position der Kugel machen, auch für die y-Koordinaten machen.

Wir brauchen dann statt `schrittweite`

`x_schrittweite` für die Änderung der x-Koordinate in einem Schritt und
`y_schrittweite` für die Änderung der y-Koordinate in einem Schritt.

Wenn die Kugel an den rechten oder linken Rand prallt,
dann ändert sich das Vorzeichen von `x_schrittweite`.
Wenn die Kugel an den oberen oder unteren Rand prallt,
dann ändert sich das Vorzeichen von `y_schrittweite`.
Dadurch erreichen wir ein Abprallen mit „Einfallswinkel = Ausfallswinkel“.

Die Struktur des Programmes bleibt wie gehabt.

Das Programm kugel_3.py

```
import sys, stddraw, random
# Lies die Änderungen der Koordinaten und die Anzeigedauer jedes Bildes ein.
x_schrittweite = float(sys.argv[1])
y_schrittweite = float(sys.argv[2])
anzeigedauer = int(sys.argv[3])
# Der Punkt (x_pos,y_pos) ist die aktuelle Position der Kugel. Er wird zu Beginn zufällig gewählt.
x_pos = random.random()
y_pos = random.random()
# Leinwandgröße sowie Größe und Farbe der Kugel werden eingestellt.
stddraw.setCanvasSize(800,800)
stddraw.setPenColor(stddraw.RED)
RADIUS = 0.05
# Die Kugel wird auf der Leinwand bewegt.
while True:
    # Falls die Kugel an die rechte oder linke Seite prallt, dann ändere ihre x-Richtung.
    if x_pos>=1.0 or x_pos<=0.0:  x_schrittweite = -x_schrittweite
    # Falls die Kugel an die untere oder obere Seite prallt, dann ändere ihre y-Richtung.
    if y_pos>=1.0 or y_pos<=0.0:  y_schrittweite = -y_schrittweite
    # Bestimme die nächste Position der Kugel.
    x_pos = x_pos + x_schrittweite
    y_pos = y_pos + y_schrittweite
    # Lösche die Leinwand, male die Kugel auf ihrer neuen Position und zeige die Leinwand an.
    stddraw.clear(stddraw.YELLOW)
    stddraw.filledCircle(x_pos, y_pos, RADIUS)
    stddraw.show(anzeigedauer)
```

Zusammenfassung

standard draw

Wir haben gesehen, wie Bilder aus einfachen Elementen
(Punkte, Striche, Kreise, Rechtecke, Schrift)
gemalt und angezeigt werden können.

Mit diesen Mitteln lassen sich z.B. Messwerte
und einfache Animationen graphisch ansprechend darstellen.

Wir haben gesehen, wie man das Entwickeln eines Programmes
mit speziellen Fällen und einfachen Ideen beginnt
und die Fälle und Ideen schrittweise verallgemeinert.