Schweizerischer Tag über Mathematik und Unterricht
Kantonsschule Wettingen, 13. September 2017

Von BASIC und TI-81 zu Python und SageMath:

25 Jahre Programmieren im Grundlagenfach Mathematik

Moritz Adelmeyer

Kantonale Maturitätsschule für Erwachsene Zürich
moritz.adelmeyer@kme.ch

Ablauf

Präsentation (ca. 40 min)

- Rückblick auf 25 Jahre Programmieren im Grundlagenfach Mathematik mit Konstanten und Variablen betreffend …
  - … Ziele und Inhalte
  - … Sprache, Umgebung und Kontext
  - … Erfolge und Misserfolge
- Einblick in aktuelle Unterrichtsunterlage mit Python als Sprache und Zufallsvorgängen als Kontext
- Ausblick auf SageMath als Alternative oder Ergänzung zu Taschenrechnern und als Umgebung für Python

Diskussion (ca. 20 min)

- .....

Hintergrund

Herbstsemester 1992/93

Erste Anstellung als Mathematiklehrer, hier an der Kantonsschule Wettingen
Mit erster Klasse Anschaffung des ersten verfügbaren Grafikrechners TI-81
Erste Programmierversuche in dieser Klasse

Bis Heute

Programmierblock im Grundlagenfach Mathematik mit jeder gymnasialen Klasse

Herbstsemester 2015/16

Weiterbildungsurlaub, Besuch der ETH-Veranstaltung "Informatik im gymnasialen Mathematikunterricht", Umstieg auf Python

Herbstsemester 2016/17

Inhaltliche Überarbeitung und didaktische Ausarbeitung der Unterrichtsunterlage "Grundkurs Programmierung"

Konstanten

Konstante Ziele

- Algorithmisches Denken fördern, Algorithmen zum Leben erwecken
- Grundrüstzeug im Programmieren mitgeben (allen Gymnasiasten und Gymnasiastinnen, als Allgemeinbildung und Hochschulvorbereitung)

Konstante Inhalte

- Algorithmen in natürlicher und formaler Sprache beschreiben, durch Flussdiagramme und Werteprotokolle veranschaulichen
- Grundelemente der Programmierung vermitteln (Anweisungen zur Verarbeitung von Zahlen, je eine Art von Verzweigung und Schleife, ein Typ von Liste)

Beispiel

Schriftliche Maturprüfung Grundlagenfach Mathematik
Klasse HH7a, KME Zürich, Januar 2017

Aufgabe 3: Wahrscheinlichkeitsrechnung (4 + 4 = 8 Punkte)

Ein Würfel ist auf vier Seiten rot und auf den übrigen zwei Seiten blau gefärbt. Der Würfel wird so viele Male geworfen, bis

a) sowohl Rot als auch Blau aufgetreten sind,
b) Rot zum zweiten Mal aufgetreten ist.

Wie gross ist jeweils die Wahrscheinlichkeit, dass man n-mal werfen muss?

- Berechne jeweils die Wahrscheinlichkeit für n = 2, n = 3 und n = 4.
- Stelle jeweils eine Formel auf, welche die Wahrscheinlichkeit für jedes n angibt.

Aufgabe 4: Programmierung (4 Punkte)

Schreibe ein Python-Programm, welches den Zufallsvorgang aus Aufgabe 3 b) simuliert.

- Eingabe des Programms: Keine.
- Ausgabe des Programms: Nach jedem Wurf die aufgetretene Farbe und zum Schluss die Anzahl getätigter Würfe.

Das Programm beginnt wie folgt:

```
from random import *      # importiert das Zufallszahlenmodul
r = 0                     # zaehlt das Auftreten von Rot
```

Zwei Studierendenlösungen

```
w = 0 . # zaehlt Anzahl Würfe
while  r != 2 :                    # rot sind die zahlen 1-4,
       a1 = randint (1,6)          # blau 5-6
       w = w+1
       if  a1 <= 4:
           msgDlg ("Du hast rot gewürfelt")
           r = r+1
       else :
           msgDlg ("Du hast blau gewürfelt. Würfle
                    nochmals.")

msgDlg ("Du hast", w, "würfe gebraucht um zwei
         mal rot zu erhalten")
```

```
w=0      b=1   # ... blau
         r<2
         x=randint(0,1)   # würfeln rot o. blau
         while:
          x == 0
          msgDlg("rot gewürfelt)
          r=r+1
          w=w+1 *
            if: r<2
              msgDlg(w,"mal geworfen")

              else:
              msgDlg("blau geworfen")
* else:
  msgDlg("blau gewürfelt)
```

Variablen

Variable Sprache

- Von BASIC und Pascal zu Java und Python

Variable Umgebung

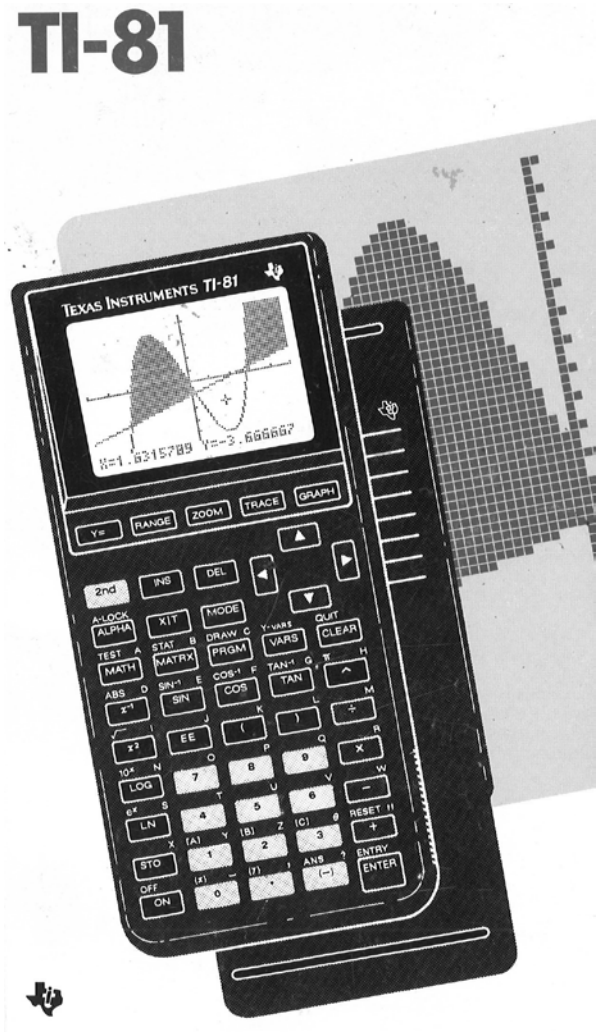
- Taschenrechner, Entwicklungsumgebungen, Mathematikprogramme

Variabler Kontext

- Stochastik, Numerik, Graphik

Beispiel

Kantonsschule Wettingen
Schuljahre 1992/93 und 1993/94

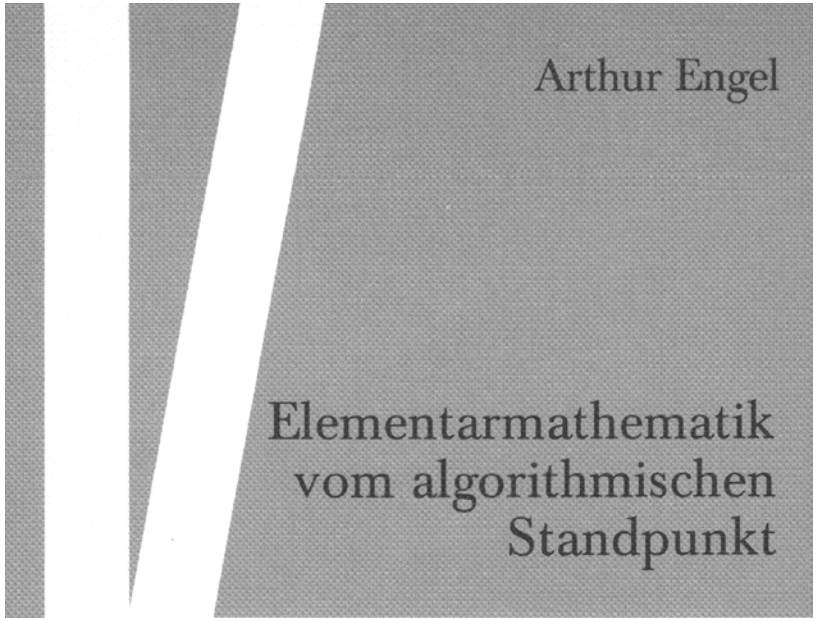



```
Prgm1:HERON
:Disp "A="
:Input A
:(1+A)/2→X
:Disp "X="
:Lbl 1
:Disp X
:X→Y
:(Y+A/Y)/2→X
:Pause
:If X<Y
:Goto 1
```

```
Prgm2:BILLARD
:ClrDraw
:1→Xmin
:96→Xmax
:0→Xscl
:1→Ymin
:64→Ymax
:0→Yscl
:Int(Rand*96+1)→X
:Int(Rand*64+1)→Y
:1→U
:1→V
:Lbl 1
:PT-On(X,Y)
:If X=96
:-1→U
:If X=1
:1→U
:If Y=64
:-1→V
:If Y=1
:1→V
:X+U→X
:Y+V→Y
:Goto 1
```

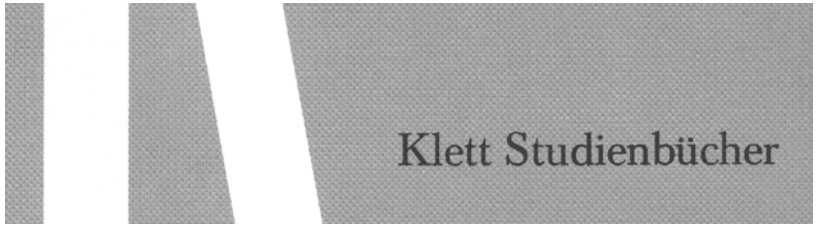
Vorbilder

1992



Vorwort

Um die Jahrhundertwende hat F. Klein eine Reform des Mathematikunterrichts eingeleitet. Die Reformbewegung adoptierte das Schlagwort „funktionales Denken". Der Funktionsbegriff sollte als Leitbegriff den ganzen Stoff durchdringen. Durch die weite Verbreitung der Computer und Taschenrechner ist die Zeit reif geworden für die nächste Reform unter dem Schlagwort „algorithmisches Denken". Der Begriff des Algorithmus sollte als Leitbegriff für die Schulmathematik dienen. Wir müssen den gesamten Schulstoff vom algorithmischen Standpunkt neu durchdenken.



9. Fig. 1.15 zeigt ein interessantes Programm. Eingabe ist eine reelle Zahl X. Ausgabe ist die Zahl Y = f (X). A und B sind Hilfsvariable. Bestimme die Funktion f.

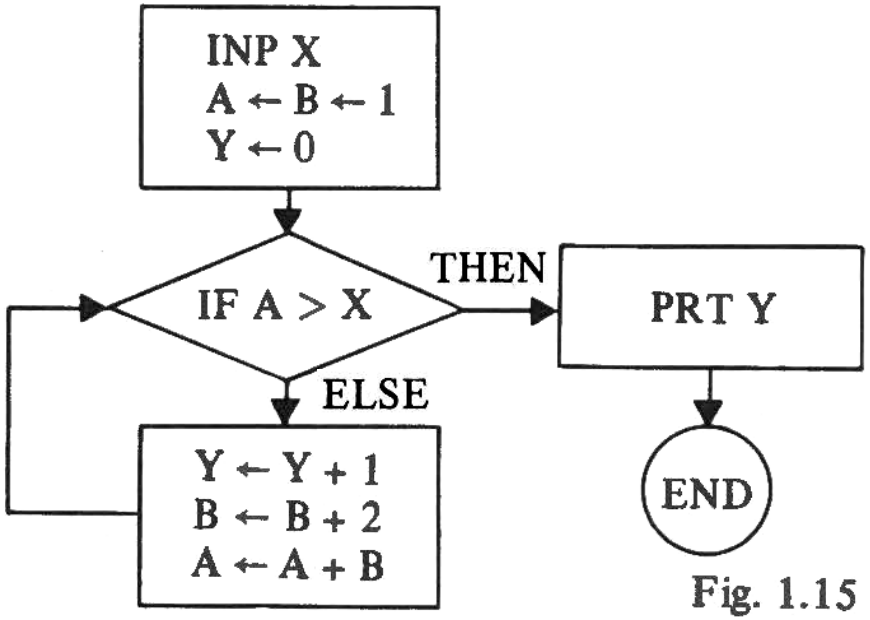
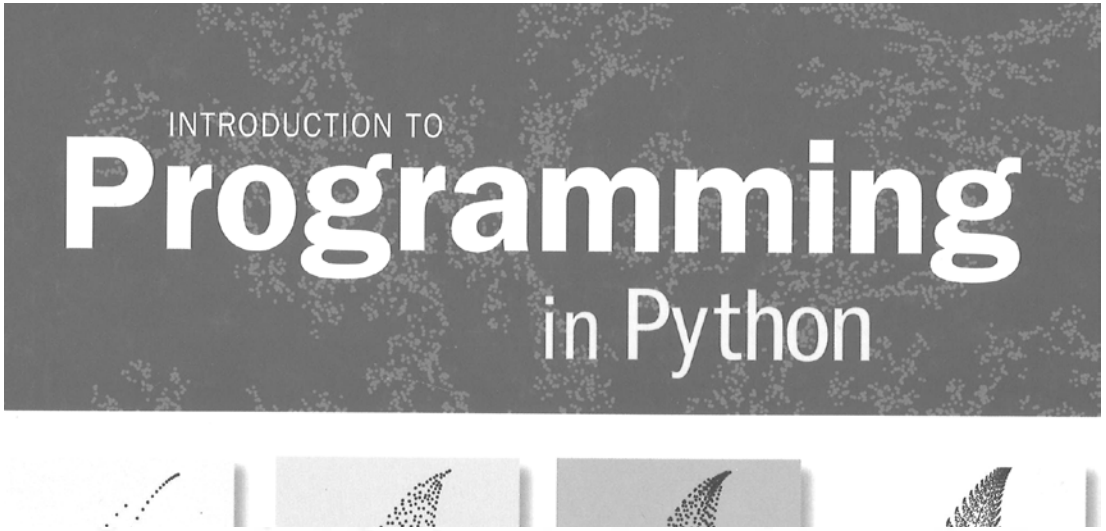


Fig. 1.15

2017



THE BASIS FOR EDUCATION IN THE last millennium was "reading, writing, and arithmetic"; now it is reading, writing, and *computing*. Learning to program is an essential part of the education of every student in the sciences and engineering. Beyond direct applications, it is the first step in understanding the nature of computer science's undeniable impact on the modern world.

Robert Sedgewick • Kevin Wayne • Robert Dondero

The book is organized around four stages of learning to program: basic elements, functions, object-oriented programming, and algorithms .

divisorpattern.py

```
import sys
import stdio

n = int(sys.argv[1])

for i in range(1, n+1):
    # Write the ith line.
    for j in range(1, n+1):
        # Write the jth entry in the ith line
        if (i % j == 0) or (j % i == 0):
            stdio.write('* ')
        else:
            stdio.write('  ')
    stdio.writeln(i)
```

```
% python divisorpattern.py 3
* * *  1
* *    2
*   *  3
```

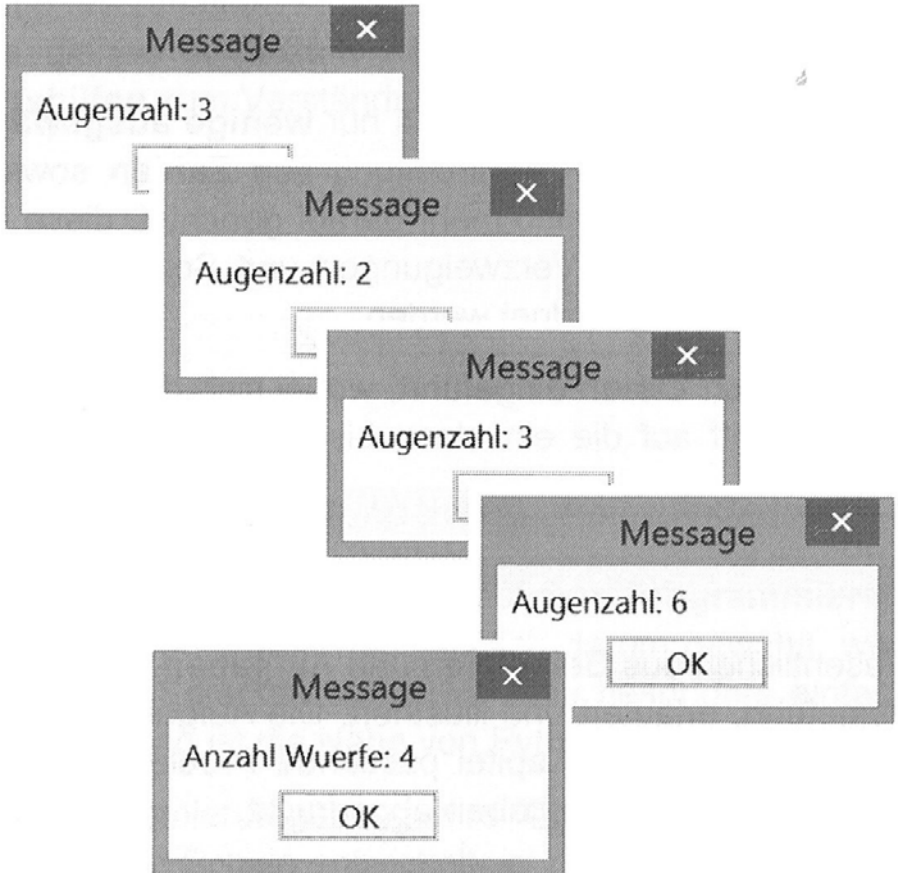
i	*j*	*i* % *j*	*j* % *i*	*output*
1	1	0	0	*
1	2	1	0	*
1	3	1	0	*
				1
2	1	0	1	*
2	2	0	0	*
2	3	2	1	
				2
3	1	0	1	*
3	2	1	2	
3	3	0	0	*
				3

Grundkurs Programmieren

Für den Einsatz im Mathematikunterricht unter Verwendung von Python mit Fokus auf Zufallsvorgänge

Moritz Adelmeyer / Dezember 2016

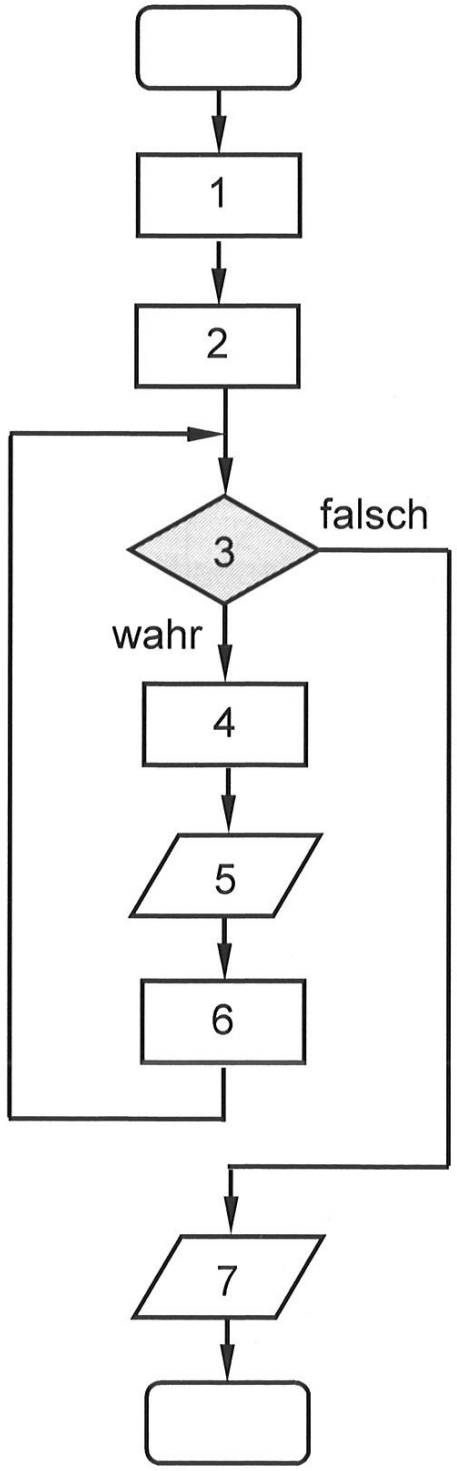
```
from random import *
a = 0
w = 0
while a != 6 :
    a = randint(1,6)
    msgDlg("Augenzahl:", a)
    w = w + 1
msgDlg("Anzahl Wuerfe:", w)
```



Bezug

Als PDF auf Anfrage per Mail

Inhalt



Konzeption

- Zielgruppe: Gymnasiastinnen und Gymnasiasten aller Profile
- Zielsetzung: Algorithmisches Denken, Grundrüstzeug im Programmieren
- Inhalt: Beschränkung auf wenige grundlegende Elemente
- Aufbau: Überblick, Beispiele, Aufgaben, Rückblick, Lösungen
- Vorstellungshilfen: Flussdiagramme, Werteprotokolle
- Sprache Python: Übersichtlichkeit, Verfügbarkeit, Universalität, Beliebtheit
- Umgebung TigerJython: Einfachheit
- Kontext Zufallsvorgänge: Bezug zum Mathematik-unterricht, Zugänglichkeit
- Zeitbedarf: Kapitel 1 – 5 ca. 24 Lektionen, Kapitel 6 – 7 ca. 10 Lektionen, zusätzlich Hausaufgaben

Zeile	a	w	a ≠ 6
1	0		
2		0	
3			wahr
4	3		
6		1	
3			wahr
4	2		
6		2	
3			wahr
4	3		
6		3	
3			wahr
4	6		
6		4	
3			falsch

SageMath

Merkmale

- Leistungsfähige Mathematiksoftware (konkurrenzfähig zu Mathematica, Maple & Co.)
- Frei verfügbar, installierbar auf allen Systemen
- Frei zugängliche Webversionen (Cloud Computing)
- Programmiert in Python (verwendet NumPy, SymPy & Co.) und kann mit Python programmiert werden

Websites

- Homepage: sagemath.org
- Einfache Webversion: sagecell.sagemath.org
- Volle Webversion: cloud.sagemath.org / cocalc.com (Collaborative Calculation)

Books

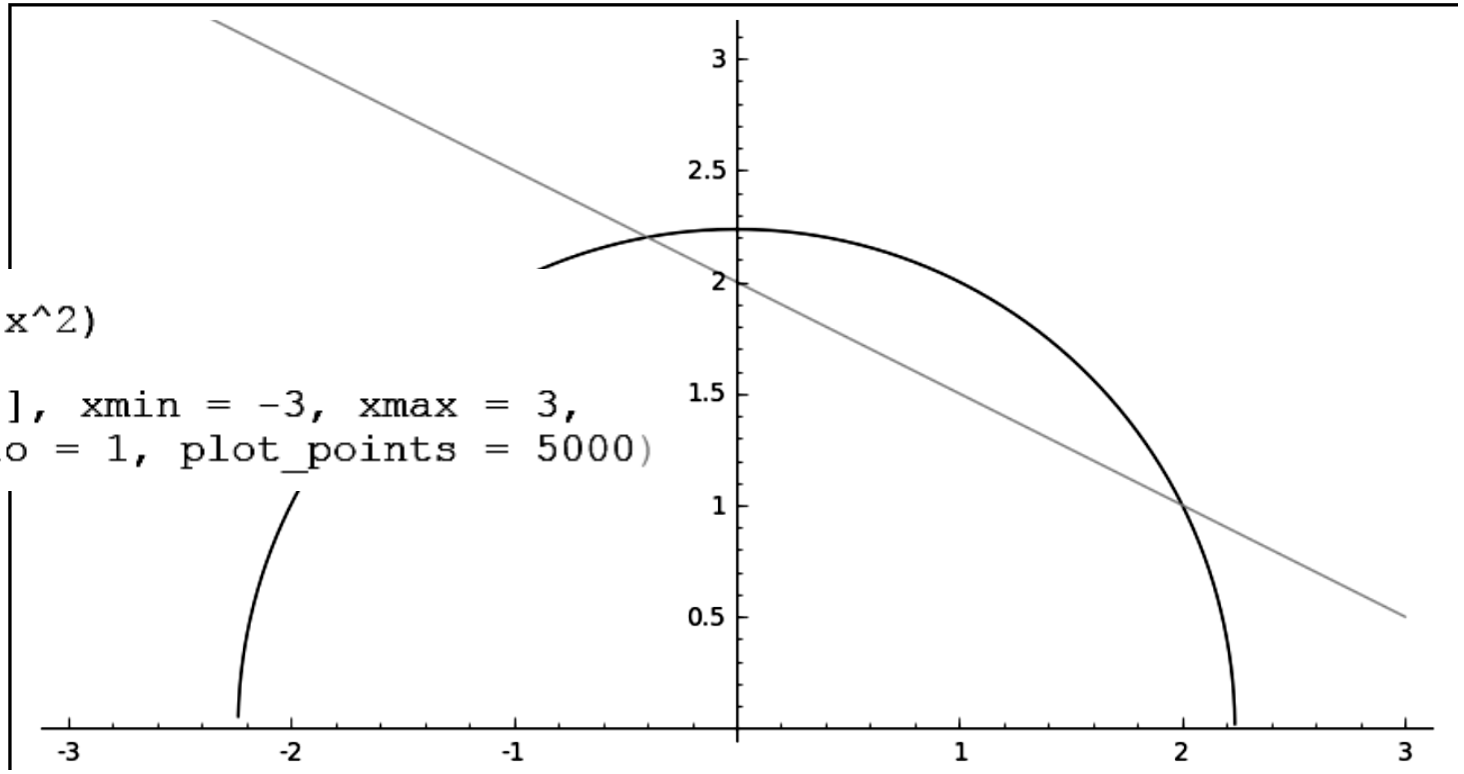
- Craig Finch: Sage Beginner's Guide. Packt Publishing, 2011
- Gregory Bard: Sage for Undergraduates. American Mathematical Society, 2015

```
1  var("x", "y")
2  solve([x^2 + y^2 == 5, x + 2*y == 4], [x, y])
```

Evaluate

```
[[x == 2, y == 1], [x == (-2/5), y == (11/5)]]
```

```
3  k(x) = sqrt(5 - x^2)
4  g(x) = 2 - x/2
5  plot([k(x), g(x)], xmin = -3, xmax = 3,
6       aspect_ratio = 1, plot_points = 5000)
```



Schlusspunkt

Eindimensionaler, symmetrischer Random Walk

Wie gross ist die Wahrscheinlichkeit, nach n Schritten **erstmals** wieder am Startort zu sein?

```
 1 from random import *
 2
 3 dtot = 1000000              # Totale Anzahl Durchgaenge festlegen
 4 nmax = 25                   # Maximale Anzahl Schritte pro Durchgang festlegen
 5
 6 # Resultateliste bereitstellen
 7
 8 r = range(nmax + 1)         # r[n] = Anzahl Rueckkehren nach n Schritten
 9 n = 1
10 while (n <= nmax):
11     r[n] = 0
12     n = n + 1
13
14 # Alle Durchgaenge ausfuehren
15
16 d = 0                       # Zaehler fuer Durchgaenge bereitstellen
17
18 while (d < dtot):
19
20     # Einen Random Walk ausfuehren
21
22     n = 0                   # Zaehler fuer Schritte bereitstellen
23     x = 0                   # Startort einnehmen
```

```
24
25      while ((x != 0 and n < nmax) or n == 0):
26
27          z = random()        # Zufallszahl zwischen 0 und
28
29          # Naechsten Schritt machen
30
31          if (z < 0.5):
32              x = x + 1       # Nach rechts gehen
33          else:
34              x = x - 1       # Nach links gehen
35
36          n = n + 1           # Zaehler fuer Schritte aktua
37
38      # Resultateliste aktualisieren
39
40      if x == 0:
41          r[n] = r[n] + 1
42
43      d = d + 1                # Zaehler fuer Durchgaenge a
44
45  # Resultate als relative Haeufigkeiten ausgeben
46
47  n = 1
48  while (n <= nmax):
49      print "n =", n, ":", r[n]*1.0/dtot
50      n = n + 1
```

```
n = 1 : 0.0
n = 2 : 0.499677
n = 3 : 0.0
n = 4 : 0.12519
n = 5 : 0.0
n = 6 : 0.062629
n = 7 : 0.0
n = 8 : 0.039246
n = 9 : 0.0
n = 10 : 0.027221
n = 11 : 0.0
n = 12 : 0.020413
n = 13 : 0.0
n = 14 : 0.015927
n = 15 : 0.0
n = 16 : 0.013099
n = 17 : 0.0
n = 18 : 0.010997
n = 19 : 0.0
n = 20 : 0.009177
n = 21 : 0.0
n = 22 : 0.008238
n = 23 : 0.0
n = 24 : 0.00695
n = 25 : 0.0
```