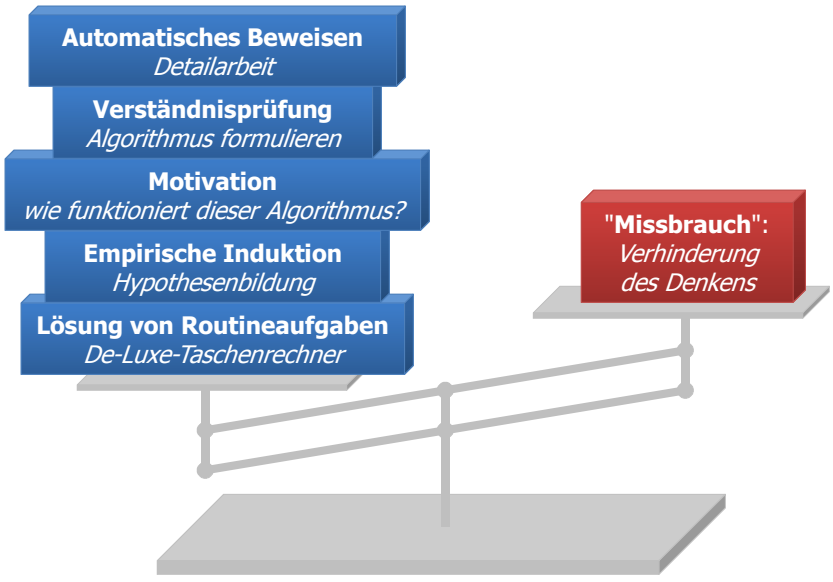
Prof. Dr. Hartmut Ring
Universität Siegen

http://mfi.math.uni-siegen.de
www.mathGUI.de

Kurzeinführung in mathGUIde und Python

Wintersemester 2010/2011

Einsatz von Computeralgebrasystemen



Computeralgebrasysteme: Auswahl

Kommerzielle Systeme

universell
teuer
black box

Derive 6.1
Studenten : 83 €

Maple 14
Student Edition: 100 €

Mathematica 7
3.800 €, Student Ed.: 153 €

Open Source

oft spezialisiert
kostenlos
white box

mathGUIde
www.mathGUI.de

Sage
Download: www.sagemath.org
Web-Notebook: www.sagenb.de.vu

mathGUIde

- ist ein **Präsentationsprogramm mit integriertem Mathematik-System**,
- fördert die Freude am kreativen mathematischen Experimentieren,
- hilft bei der Kontrolle des Verständnisses mathematischer Algorithmen,
- ermöglicht direkte Implementierungen ohne technischen Ballast,
- erlaubt durch Einblick in die mitgelieferten, didaktisch optimierten Quellen das schnelle Verständnis mathematischer Algorithmen,
- fördert das experimentelle Durchdringen des Stoffs der Lehrveranstaltungen „Diskrete Mathematik für Informatiker„ und „Lineare Algebra für Informatiker“,
- ist Freeware und wird mit allen Quellen verbreitet,
- baut auf der weit verbreiteten Skriptsprache Python 3.1 auf und ist daher leicht zu erlernen und zu erweitern,
- besteht aus einem Python-Paket (mathguide.py) und einer einfach zu bedienenden grafischen Benutzeroberfläche (GUI),
- kommt mit einer Online-Referenz aller Funktionen und Klassen.
- Bei der Implementierung der Algorithmen hat Lesbarkeit und Verständlichkeit Vorrang vor Effizienz.
- mathGUIde wurde mit der portablen GUI-Klassenbibliothek Qt (open source license) in C++ entwickelt und läuft auf allen üblichen Plattformen (Windows, Mac OS, Linux etc.).
- mathGUIde ist **speziell auf die Vorlesungen „Diskrete Mathematik für Informatiker“ und „Lineare Algebra für Informatiker“ zugeschnitten**, und die Bedienung ist in wenigen Minuten erlernt.

www.mathGUI.de

Sprachvarianten

5

Maple `isprime(7);`

```
f := proc(n)
  if n = 1 then 1;
  else n * f(n-1);
  end if;
end proc;
```

Mathematica `PrimeQ[7]`

```
f[1] = 1
f[n_] := n f[n-1]
```

mathGUIde `isPrime(7)`

```
def f(n):
  if n == 1:
    return 1
  else:
    return n * f(n-1)
```

Java `// nicht vorhanden`

```
int f(int n) {
  if (n == 1)
    return 1;
  else
    return n * f(n-1);
}
```

Python: Visuelle Syntax

6

Beispiel: **Fibonacci**-Zahlen (jede Zahl ist Summe ihrer beiden Vorgänger)

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
fib(n)	**0**	**1**	**1**	**2**	**3**	**5**	**8**	**13**	**21**	**34**	**55**	**89**	**144**	**233**	**377**	**610**	**987**	**1597** ...

Java

```
public class Fib {
    static int fib(int n) {
        int a, b, a0, i;
        if (n < 2)
            return n;
        else {
            a = 0;
            b = 1;
            for (i=2; i<=n; i++) {
                a0 = a;
                a = b;
                b = a0 + b;
            }
            return b;
        }
    }
    public static void main(String[] args) {
       System.out.println(fib(5));
    }
}
```

Python

```
def fib(n):
    if n < 2:
        return n
    else:
        a = 0
        b = 1
        for i in range(2, n+1):
            a0 = a
            a = b
            b = a0 + b
        return b

print(fib(5))
```

Einfacher mit Simultanzuweisung:

```
a, b = b, a+b
```

Wichtige Python-Datentypen

7

Strings

Wahlweise einfache oder doppelte **Anführungszeichen**

```
'Wer hat den "Faust" geschrieben?'
"don't worry"
```

Ganze Zahlen

beliebig groß

```
n = 237468372462834682346823648237 6 ** 7
n = 237468372462834682346823648237 6 ^ 7
```
*

Listen

polymorphe Aufzählung

```
l = ["a", 2, [3]]
```

```
list(range(5))      → [0,1,2,3,4]
list(range(2,5))    → [2,3,4]
list(range(5,2))    → []
list(fromTo(2,4))*  → [2,3,4]
list(fromTo(4,2))*  → []
```

* nur in mathGUIde

for-Schleifen

8

Iteration über Aufzählungen

```
for i in fromTo(2,4):
    print(i*i)
```
*

```
for i in [2, 3, 5, 7]:
    print(i, "ist eine einstellige Primzahl")
```

```
for i in [2, 3, 5, 7]:
    print("{0} ist eine einstellige Primzahl".format(i))
```

```
for i in fromTo(2,4):
    print("2 * {0} = {1}".format(i, 2*i))
```
*

```
for i in fromTo(1,100):
    if isPrime(i):
        print("{0} ist eine Primzahl".format(i))
```
*

* nur in mathGUIde

„List Comprehensions“ 9

$\{2i \mid i \in \{1,2,3\}\}$

```
[ 2*i for i in [1,2,3] ]
```

[2, 4, 6]

```
[fibonacci(i) for i in fromTo(1,100)]

[prime(i) for i in fromTo(1,10)]

[i*i for i in fromTo(1,10)]

sum([i*i for i in fromTo(1,10)])
```

$\{10n \mid n \in \{0,\ldots,9\}, n \bmod 5 = 0\}$

[10, 20, 30, 40, 60, 70, 80, 90]

```
[10*n for n in range(10) if n % 5 != 0]
```


Eingebaute Python-Datentypen 10