



Informatik I

Debugging

G. Zachmann
 Clausthal University, Germany
zach@in.tu-clausthal.de



Der erste Computer-Bug



Photo # NH 96566-KN First Computer "Bug", 1945

9/2

9/9

0800 Anchan started

1000 " stopped - anchan ✓

1300 (032) MP-MC ~~1.50476415~~ 4.615925059(-2)

033 PRO = 2.130476415

cover 2.130676415

Relays 6-2 in 033 failed special speed test in relay .. 11.00 test.

1100 Started Cosine Tape (Sine check)

1525 Started Multi Adder Test.

1545  Relay #70 Panel F (moth) in relay.

1700 First actual case of bug being found. Anchan started.

1700 closed down.

Handwritten note in orange: Bug 2145 2049 3372



Grace Hopper
 Admiral, US Navy



Folgen von Programmfehlern



- 1962 führte ein fehlender Bindestrich in einem Fortran-Programm zum Verlust der Venus-Sonde Mariner 1, welche über 80 Millionen US-Dollar gekostet hatte.
- Zwischen 1985 und 1987 gab es mehrere Unfälle mit dem medizinischen Bestrahlungsgerät Therac-25. Infolge einer Überdosis, die durch fehlerhafte Programmierung und fehlende Sicherungsmaßnahmen verursacht wurde, mussten Organe entfernt werden, ein Patient verstarb drei Wochen nach der Bestrahlung.
- 1999: NASA-Sonde *Climate Orbiter* verpasst den Landeanflug auf den Mars, weil die Programmierer das falsche Maßsystem verwendeten – Yard statt Meter. Die NASA verlor dadurch die mehrere hundert Millionen Dollar teure Sonde.



- 1996: Ariane-5 explodierte
 - Grund: Konvertierung einer zu großen Zahl von 64-bit float nach 16-bit int
- 1994: "Pentium-Bug"
 - Inkorrekte Division in manchen Fällen wegen unvollständiger Lookup-Tabellen
- Orkan "Lothar": keine Vorwarnung durch Wetterdienst
 - Grund: ignoriertes Ausreiser
- Millennium-Bridge in London:
 - Falsche Annahmen über Fußgänger-Verhalten → fehlerhafte Simulation



Programmfehler (oder Bugs)

- Ursache:
 - Programmierfehler
 - Fehler in der Laufzeitumgebung
 - Unvollständigkeit, Mehrdeutigkeiten, Fehler in der Spezifikation
- Eine Regel besagt: Ab einer bestimmten Größe enthalten Computerprogramme auch immer Programmfehler
- Arten von Programmfehlern:
 - Syntaxfehler
 - Laufzeitfehler
 - Designfehler

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Debugging 5

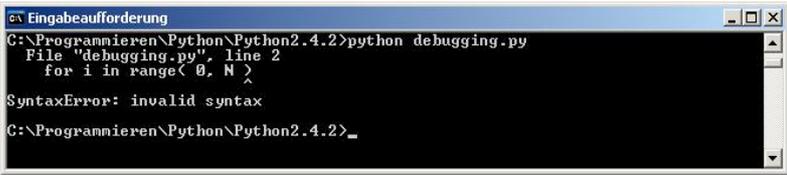
Syntaxfehler

- Syntaxfehler sind Verstöße gegen grammatische Regeln der Programmiersprache

```
N = 100
for i in range( 0, N )
    while (N % i) == 0:
        print i
        N = N / i
```

Doppelpunkt vergessen

- Solche Fehlern werden beim Versuch des Programmstarts von Python erkannt und gemeldet.



Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Debugging 6



Laufzeitfehler (Semantische Fehler)



- Programme mit Syntaxfehlern lassen sich nicht starten.
- Semantische Fehler sind dagegen Fehler, die die Bedeutung des Programmcodes betreffen

```
a = "String"  
print a[10]
```

a besteht nur aus
6 Zeichen

- Solche Fehler können das Programm zum Absturz bringen, das Programm friert ein (Endlosschleifen) oder verhält sich merkwürdig

```
cx Eingabeaufforderung  
C:\Programmieren\Python\Python2.4.2>python debugging2.py  
Traceback (most recent call last):  
  File "debugging2.py", line 2, in ?  
    print a[10]  
IndexError: string index out of range  
C:\Programmieren\Python\Python2.4.2>
```



Designfehler



- Designfehler sind Fehler im Grundkonzept einer Software
- Fehlerarten:
 - Fehler in der Anforderungsdefinition / Spezifikation
 - Das Programm tut nicht das, was der Kunde wollte
 - beruhen meist auf mangelnder Kenntnis des Fachgebietes, für das die Software entwickelt wird
 - Fehler im Softwaredesign
 - Das Programm läßt sich später nicht mehr verstehen / warten / verbessern
 - beruhen oft auf mangelnder Erfahrung des Softwareentwicklers



Vermeidung von Fehlern



- Je früher in einem Entwicklungsprozeß ein Fehler auftritt und je später er entdeckt wird, desto aufwendiger ist es, ihn zu beheben.
- Designfehler lassen sich oft durch gute Planung und Wahl eines geeigneten Softwaredesigns weitestgehend vermeiden
- Syntaktische Fehler erkennt meist der Compiler oder der Interpreter
- Beim Auffinden semantischer Fehler helfen (für bestimmte Fälle / Klassen von Fehlern) spezielle Programme



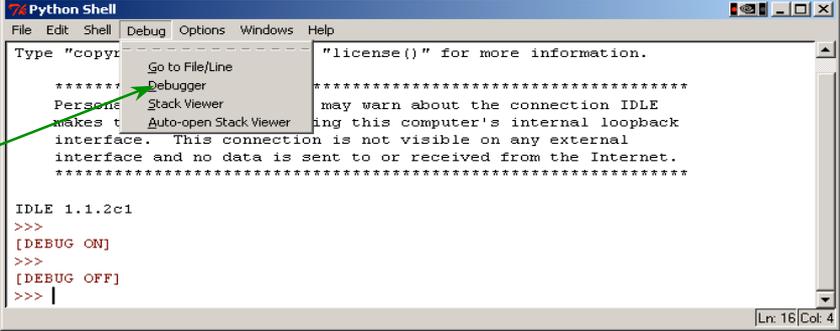
Debugger



- Nicht immer sind semantische Fehler so einfach zu finden wie in dem Beispiel oben
- Deswegen hilft ein spezielles Programm, der sogenannte Debugger, dem Programmierer bei der Fehlersuche
- Ein Debugger ermöglicht die Ablaufverfolgung des Programms in einzelnen Schritten
- Mit Python wird ein Debugger mitgeliefert

Python-Debugger

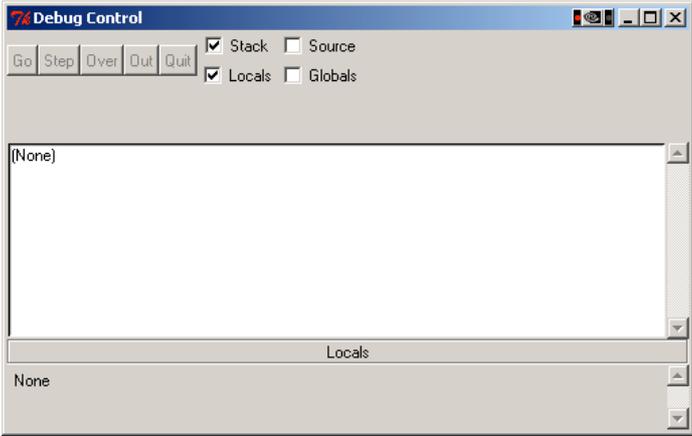
- Die Python-IDE Idle starten
- Das Menu „Debugger“ wählen
- Auf Menüpunkt „Debugger“ drücken



Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Debugging 11

Python-Debugger

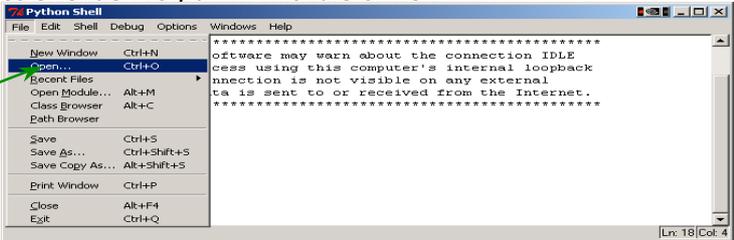
- Das Debugger-Fenster wird geöffnet



Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Debugging 12

Python-Debugger

- Das betreffende Programm mit Idle öffnen



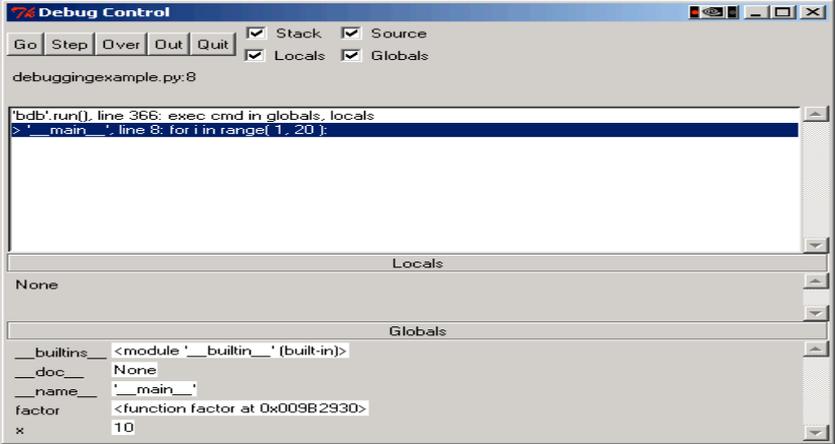
- Und starten



Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Debugging 13

Python-Debugger

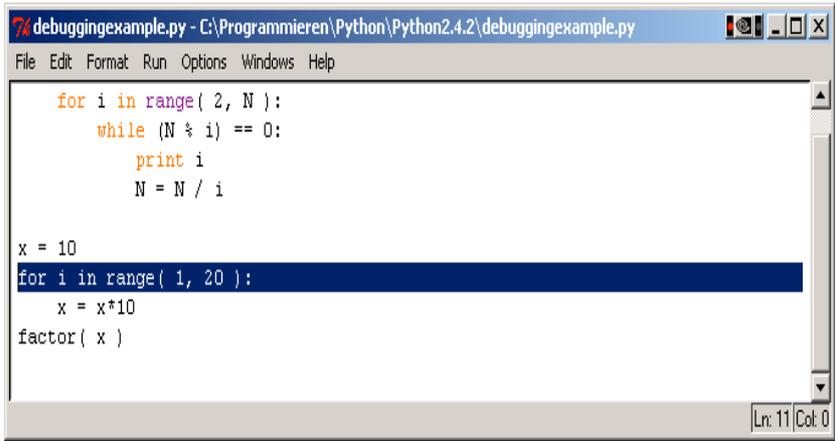
- Im Debugger-Fenster sieht man den Stack, sowie die Werte der lokalen und globalen Variablen



Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Debugging 14

Python-Debugger

- Im Sourcecode-Fenster wird die aktuelle Position, an der der Debugger das Programm gerade hält, angezeigt



```

debuggingexample.py - C:\Programmieren\Python\Python2.4.2\debuggingexample.py
File Edit Format Run Options Windows Help

    for i in range( 2, N ):
        while (N % i) == 0:
            print i
            N = N / i

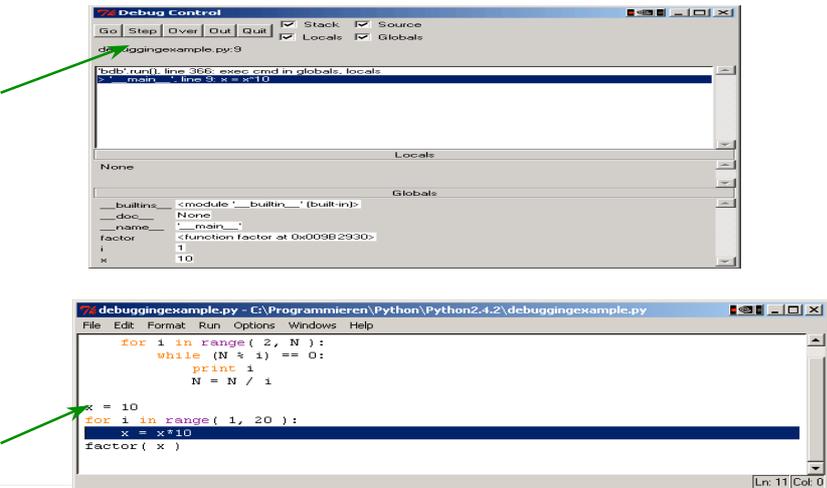
x = 10
for i in range( 1, 20 ):
    x = x*10
factor( x )
Ln: 11 | Col: 0

```

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Debugging 15

Python-Debugger

- Mit dem Step-Button kann man das Programm schrittweise weiter ausführen



Debug Control

Go Step Over Out Quit Stack Source Locals Globals

debuggingexample.py:9

bdb> run(), line 365: execs cmd in globals, locals
 bdb> main -> line 9: x = x*10

Locals

None

Globals

__builtin__ <module '__builtin__' (built-in)>
 __doc__ None
 __name__ '__main__'
 factor <function factor at 0x009B2930>
 i 1
 x 10

```

debuggingexample.py - C:\Programmieren\Python\Python2.4.2\debuggingexample.py
File Edit Format Run Options Windows Help

    for i in range( 2, N ):
        while (N % i) == 0:
            print i
            N = N / i

x = 10
for i in range( 1, 20 ):
    x = x*10
factor( x )
Ln: 11 | Col: 0

```

Prof. Dr. G. Zachmann Informatik 1 - WS 05/06 Debugging 16



Python-Debugger



- Wenn man nur an bestimmten Stellen des Source interessiert ist kann man per Rechtsklick im Sourcecode-Fenster einen **Breakpoint** setzen und mittles **Go**-Taste im Debugger-Fenster bis zu dieser Stelle laufen lassen

The screenshot shows a Python IDE window titled 'debuggingexample.py - C:\Programmieren\Python\Python2.4.2\debuggingexample.py'. The menu bar includes 'File', 'Edit', 'Format', 'Run', 'Options', 'Windows', and 'Help'. The code editor contains the following Python code:

```
for i in range( 2, N ):
    % i) == 0:
    i
    N = N / i

x = 10
for i in range( 1, 20 ):
    x = x*10
factor( x )
```

A context menu is open over the first line of the first loop, with 'Set Breakpoint' selected. The status bar at the bottom right of the editor shows 'Ln: 2 | Col: 1'.