

TI-Nspire CXII-T CAS – Einstieg in die Programmierung mit Python

Dipl.Ing. Hans-Martin Hilbig



Teachers Teaching with Technology™

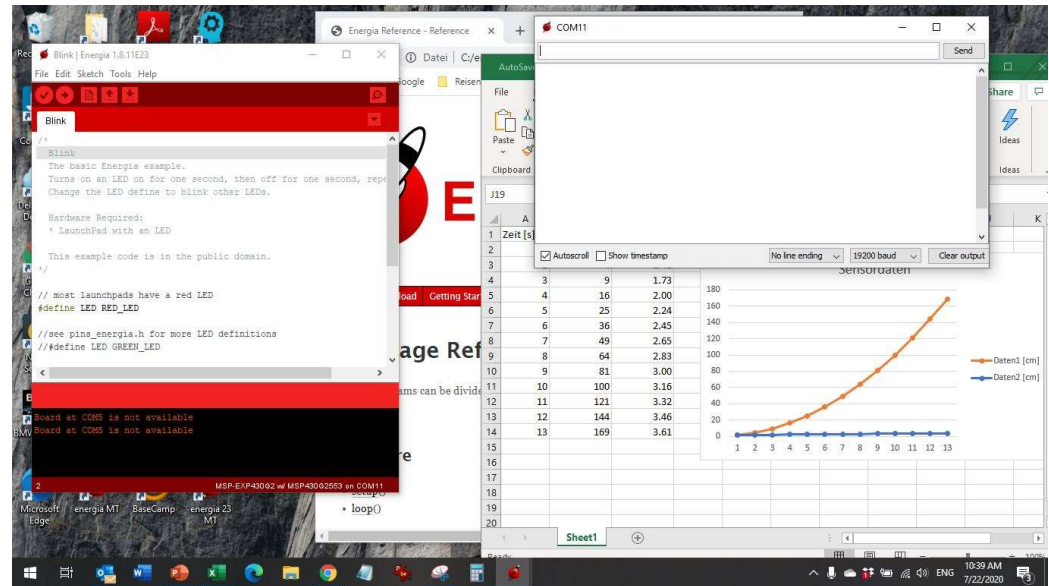
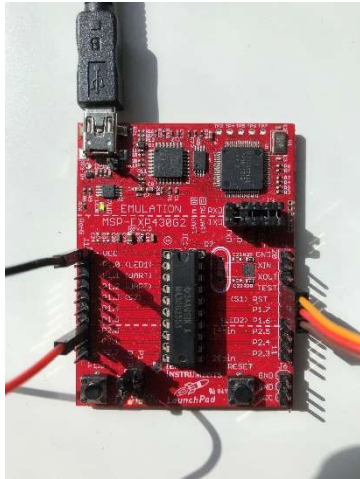
t3europe.eu

TI-Nspire CXII – Einstieg in Python: Agenda

- Lernsysteme
- Warum Python?
- Erste Schritte mit dem TI-Nspire-CXII in Python
- Lust auf mehr!
- Nützliche links
- Beispiele für Seminararbeiten mit Python



Einstieg in Python: Lernsysteme (1)



The screenshot displays a Windows desktop environment. In the foreground, a code editor window titled 'Blink | Energia 1.6.11E33' shows Python code for controlling an LED. The code includes comments and defines two LEDs: a red LED and a green LED. Below the code, a terminal window shows error messages: 'Board at COM5 is not available' and 'Board at COM5 is not available'. To the right, a window titled 'COM11' displays a line graph with two data series: 'Daten1 [cm]' (orange line) and 'Daten2 [cm]' (blue line). The x-axis is labeled 'Zeit [s]' and ranges from 1 to 13. The y-axis ranges from 0 to 180. The 'Daten1' series shows a non-linear increasing trend, while 'Daten2' remains constant at 0. A table of data points is visible in the background of the graph window.

Zeit [s]	Daten1 [cm]	Daten2 [cm]
3	9	1.73
4	16	2.00
5	25	2.24
6	36	2.45
7	49	2.65
8	64	2.83
9	81	3.00
10	100	3.16
11	121	3.32
12	144	3.46
13	169	3.61



Teachers Teaching with Technology™

HMH 7/22/20

t3europe.eu

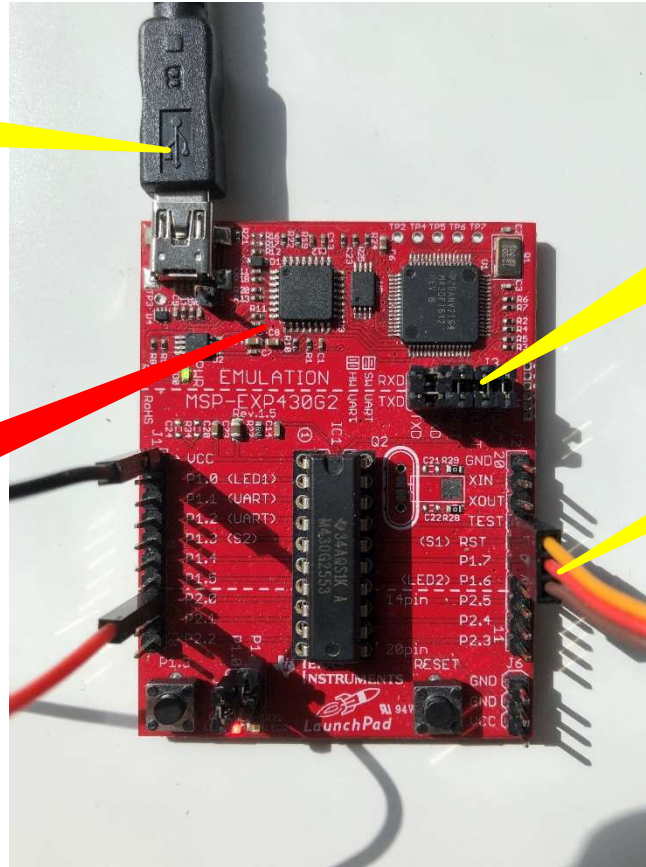
Einstieg in Python: Lernsysteme (1a)

Ist der USB-Stecker
im richtigen Port?

Hat jemand die
Jumper
umgesteckt?

Sind die Kabel auf
dem richtigen
Stecker?

Hat jemand einen
Kugelschreiber auf
die Platine fallen
gelassen?



Teachers Teaching with Technology™
HMH 7/22/20

t3europe.eu

Einstieg in Python: Lernsysteme (1b)

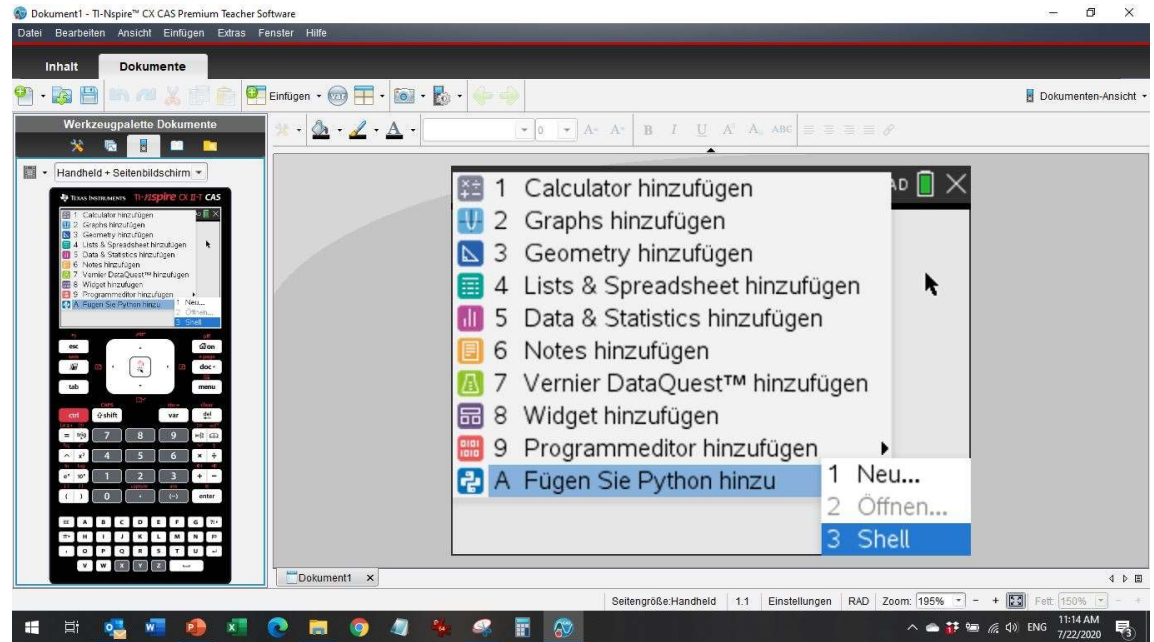
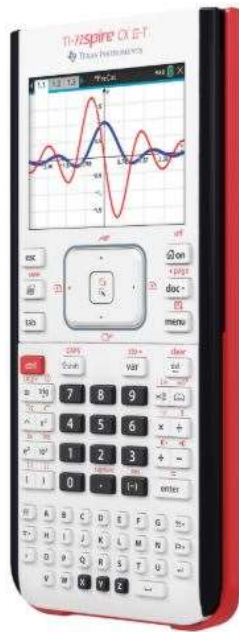
The screenshot shows the Energia IDE interface with several windows and callouts:

- Das Compiler-Fenster**: Points to the main code editor window.
- Fehlermeldung?**: Points to the console window showing error messages: "Board at COM5 is not available".
- Befehlsliste**: Points to the "COM11" window, which is the command list.
- E/A-Fenster**: Points to the "Serial" window, which is the E/A (Input/Output) window.
- Grafikfenster**: Points to the "Sensorwerte" window, which displays a line graph and a data table.
- Tabellenkalkulation**: Points to the data table in the "Sensorwerte" window.
- Schreibt da noch jemand nebenbei e-mails?**: Points to the taskbar at the bottom of the screen.

Zeit [s]			
3	9	1.73	
4	16	2.00	
5	25	2.24	
6	36	2.45	
7	49	2.65	
8	64	2.83	
9	81	3.00	
10	100	3.16	
11	121	3.32	
12	144	3.46	
13	169	3.61	



Einstieg in Python: Lernsysteme (2)



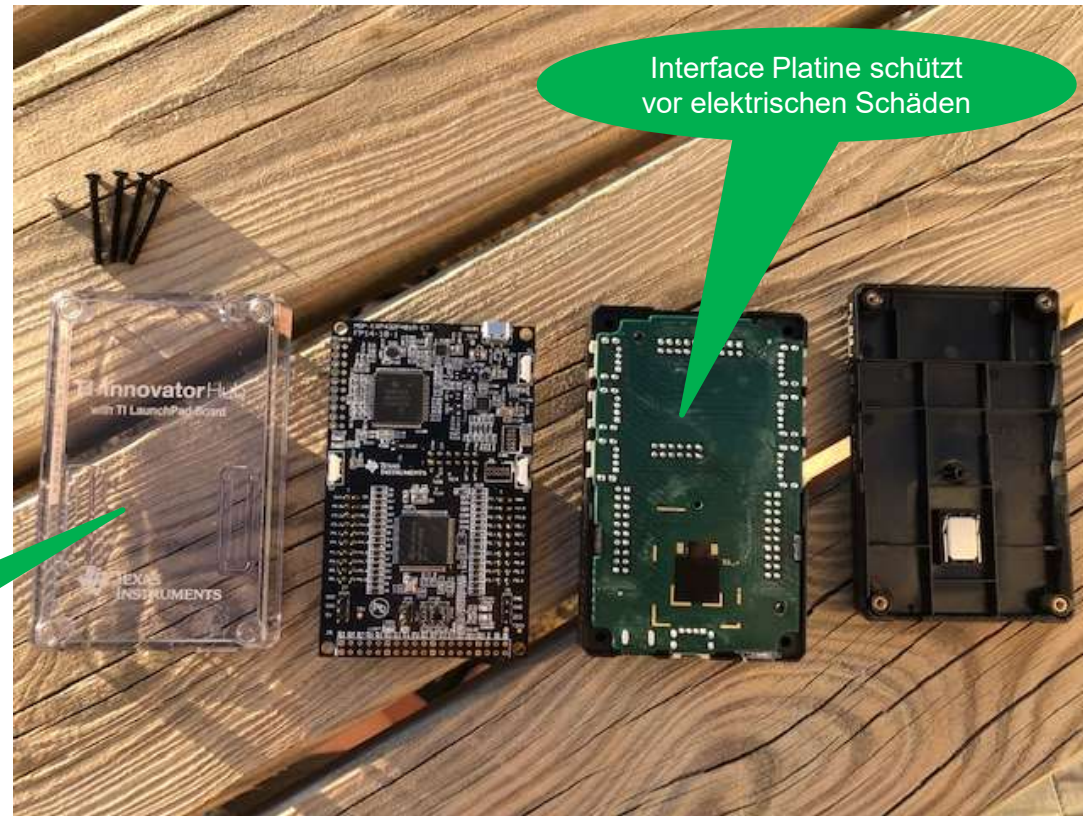
Teachers Teaching with Technology™
HMH 7/22/20

t3europe.eu

Einstieg in Python: Lernsysteme (2a)



Alles geschützt in einem Gehäuse



-  jumper
-  Kurzschluß
-  Offene Bauteile

Einstieg in Python: Lernsysteme (2b)

The screenshot shows the TI-Nspire CX CAS Premium Teacher Software interface. A yellow callout bubble on the left says "Alles in einer Anwendung". A yellow callout bubble points to the "Befehlsliste" (command list) on the right, which includes options like "1 Calculator hinzufügen", "2 Graphs hinzufügen", "3 Geometry hinzufügen", "4 Lists & Spreadsheet hinzufügen", "5 Data & Statistics hinzufügen", "6 Notes hinzufügen", "7 Vernier DataQuest™ hinzufügen", "8 Widget hinzufügen", "9 Programmierer hinzufügen", and "A Fügen Sie Python hinzu". A yellow callout bubble points to the "Python" option, which has a sub-menu with "1 Neu...", "2 Öffnen...", and "3 Shell". A yellow callout bubble points to the "Shell" option. A yellow callout bubble points to the "Tabellenkalkulation" (spreadsheet) option. A yellow callout bubble points to the "Grafikfenster" (graph window) option. A yellow callout bubble points to the "E/A-Fenster" (input/output window) option. On the right side, there are two red prohibition signs with the text "Internet" and "Cloud" next to them.

Schreibt da noch jemand nebenbei e-mails?



Einstieg in Python*: Warum Python? (1)

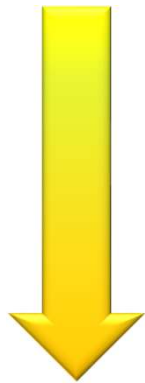
Eigenschaft	Python	Arduino	TI-Basic
Open Source	✓	✓	x
Bibliotheken	✓	✓	x
Portabel	✓	(✓)	x
Interpreter	✓	x	✓
Objekt orientiert	✓	✓	(x)
Hardware-einbettbar	✓	✓	x
erweiterbar	✓	✓	x
Vielseitige Anwendungen	✓	x	x
Hub-Unterprogramme	✓	✓	x



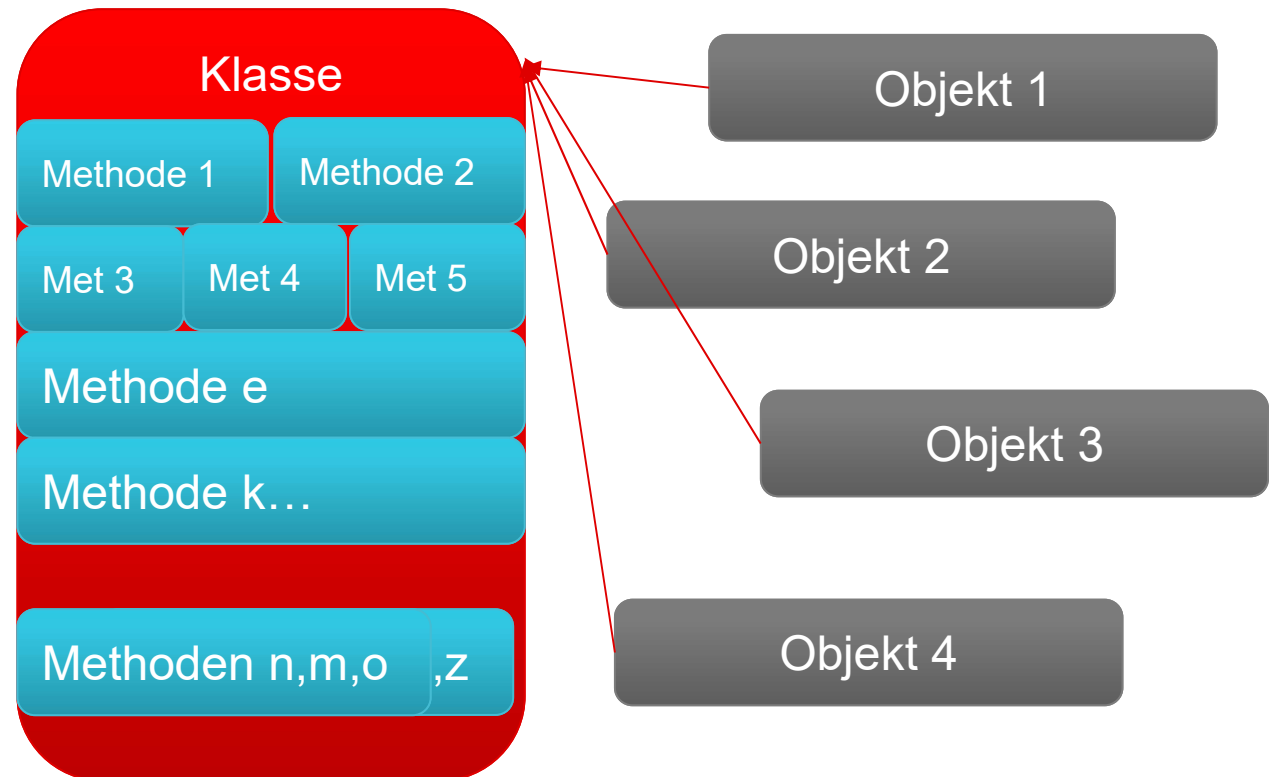
Einstieg in Python: Warum Python? (2)

Das Schichtenmodell:

Abstrakt, universell:



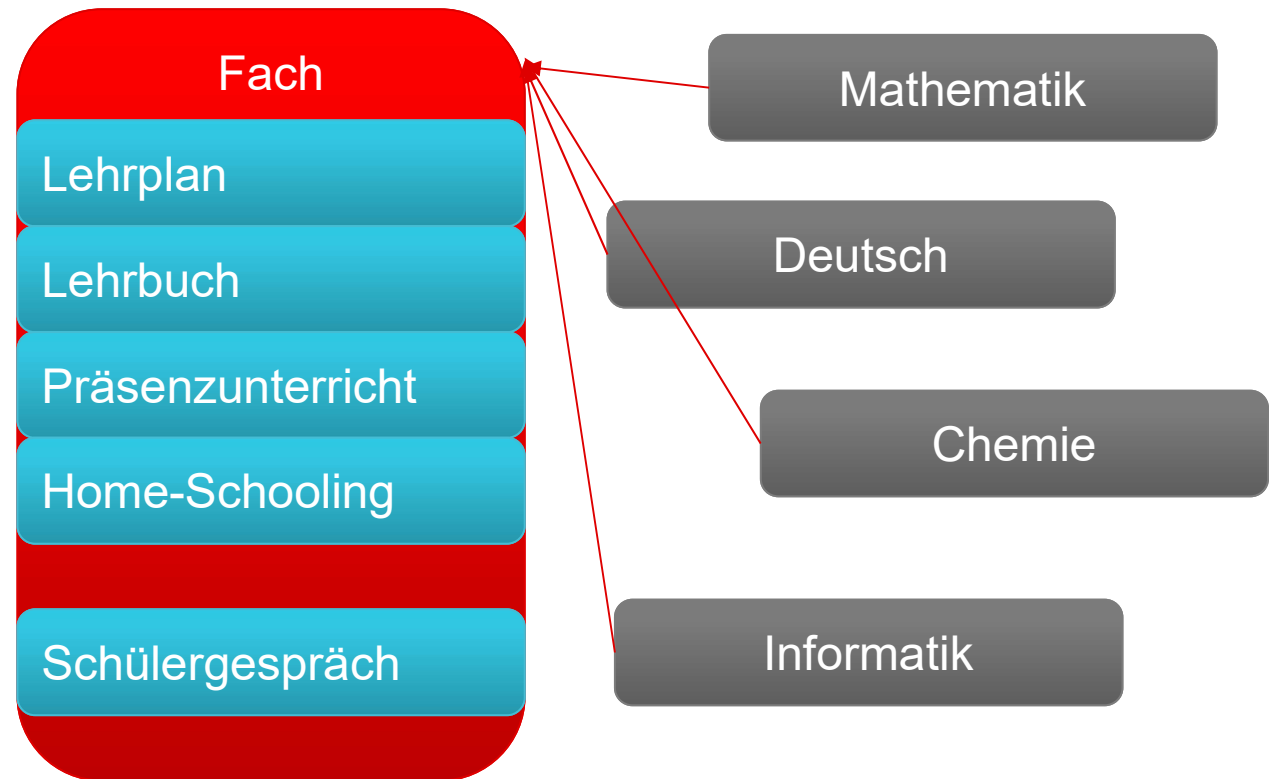
Hardware-nah:



Einstieg in Python: Warum Python? (2a)

Beispiel:

Die allgemeine
Hochschulreife



Erste Schritte mit dem TI-Nspire CXII in Python

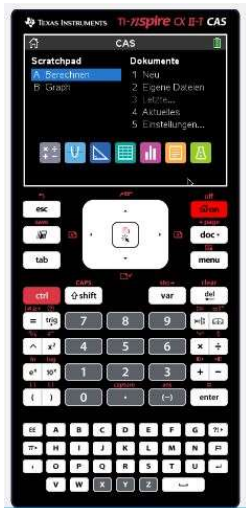
- Die shell – Willkommen auf dem Spielplatz!
- Variablen sind Objekte in Python!?
- Mehrere Zeilen in der Shell ausführen? Kein Problem!
- Starkes Stück! Listen in Python
- Wie spät ist es bitte? Das Modul “time”



Die shell – Willkommen auf dem Spielplatz!

3 Schritte bis zum Spielplatz:

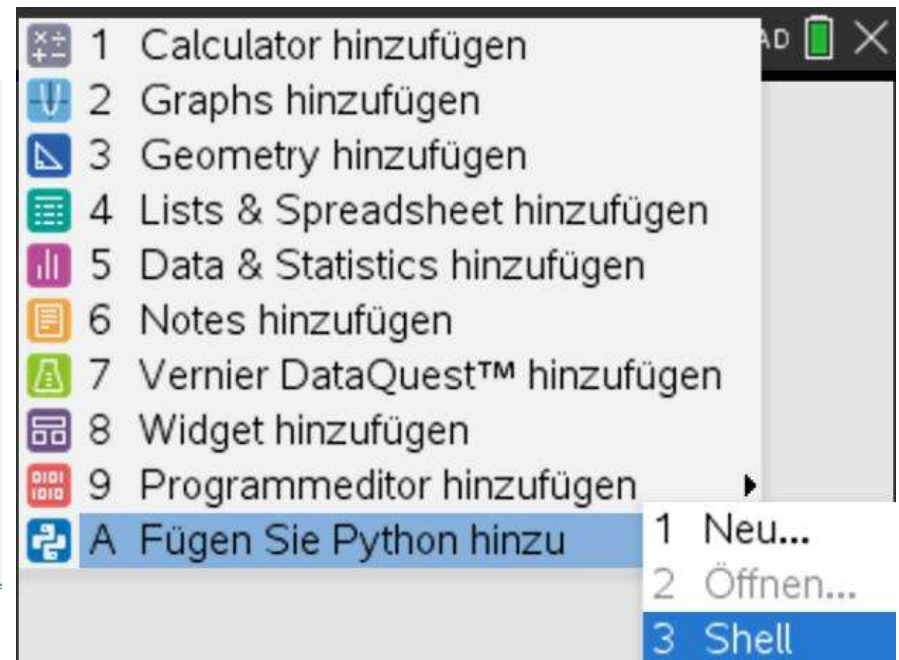
1. Einschalten



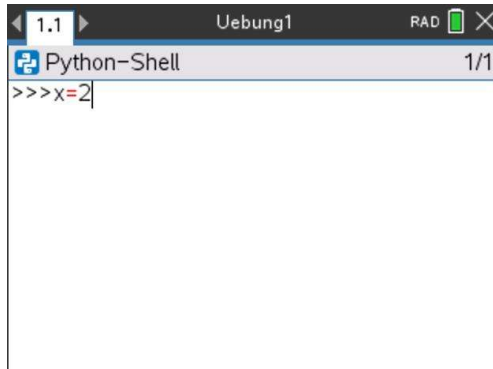
2. Neues Dokument



3. A-3 Shell

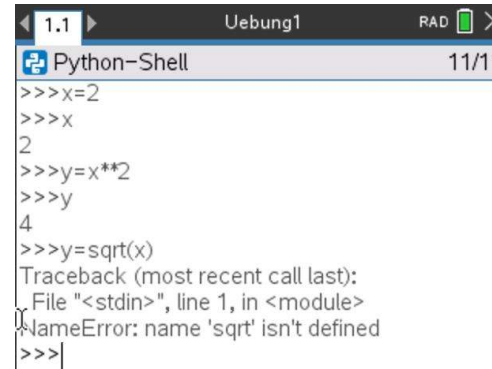


Variablen sind Objekte in Python? (1)



```
1.1 Uebung1 RAD 1/1
Python-Shell
>>>x=2
```

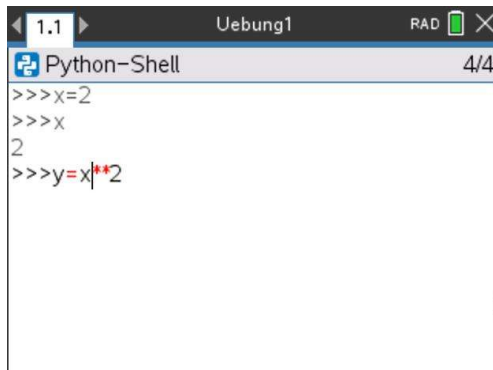
Operatoren werden von der Shell in rot dargestellt



```
1.1 Uebung1 RAD 11/11
Python-Shell
>>>x=2
>>>x
2
>>>y=x**2
>>>y
4
>>>y=sqrt(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sqrt' isn't defined
>>>|
```

Die 'Quadratwurzel-Taste' hat sqrt() erzeugt, aber die Funktion ist nicht bekannt!?

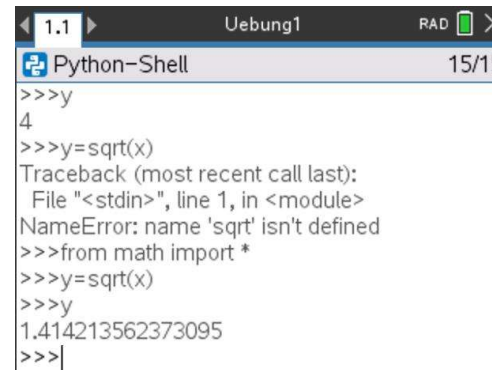
Viele Funktionen von Python befinden sich in eigenen Modulen (Bibliotheken)...



```
1.1 Uebung1 RAD 4/4
Python-Shell
>>>x=2
>>>x
2
>>>y=x**2
```

Man kann sich den Variablen-Inhalt direkt anzeigen lassen, auch ohne 'print()' Funktion

Die Tasten des TI-Nspire erzeugen automatisch die richtige Funktion – hier die Taste x^2



```
1.1 Uebung1 RAD 15/15
Python-Shell
>>>y
4
>>>y=sqrt(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sqrt' isn't defined
>>>from math import *
>>>y=sqrt(x)
>>>y
1.414213562373095
>>>|
```

... wie die sqrt() Funktion sich im module 'math' befindet und erst importiert werden muss..

Dann klappt's!



Variablen sind Objekte in Python? (2)

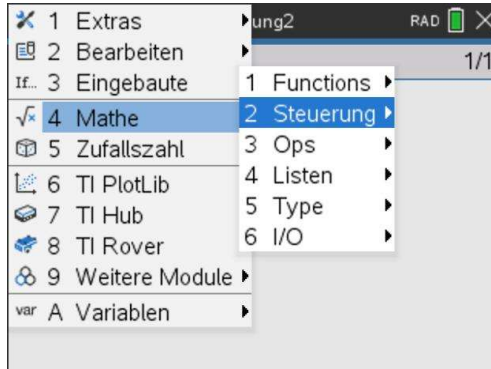
```
1.1 Uebung1 RAD 8/8
Python-Shell
>>>type(x)
<class 'int'>
>>>type(y)
<class 'float'>
>>>y=2*x
>>>type(y)
<class 'int'>
>>>|
```

x ist eine Variable der Klasse 'Int', y der Klasse 'float' (Gleitkomma). Sobald aber der Inhalt von y wieder ganzzahlig wird, wechselt y automatisch wieder zurück zu 'Int'.

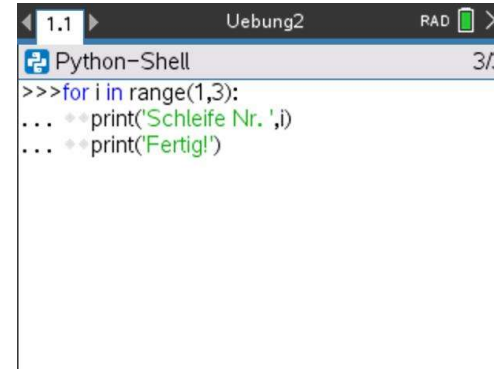
```
1 Letztes Programm wiederholen (Ctrl+R)
2 Zu Python-Editor wechseln
If... 3 Starten...
4 Protokoll löschen
5 Shell erneut initialisieren
6 dir()
7 Von PROGRAMM import *...
8 TI Rover
9 Weitere Module
var A Variablen
1.4 142 13302373093
>>>|
```

Man kann das Protokoll der Shell löschen, der Variableninhalt bleibt dabei erhalten. Wenn man auch die geladenen Module und die Variablen löschen möchte, benutzt man den Befehl (5), Shell erneut initialisieren

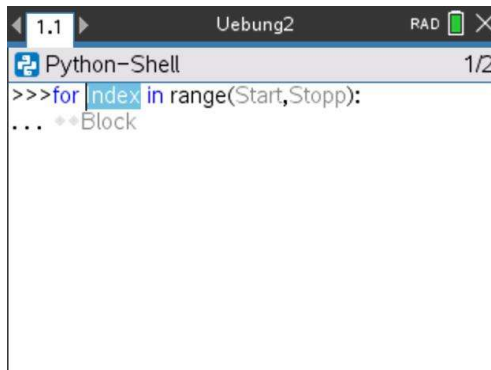
Mehrere Zeilen in der Shell ausführen? Kein Problem!



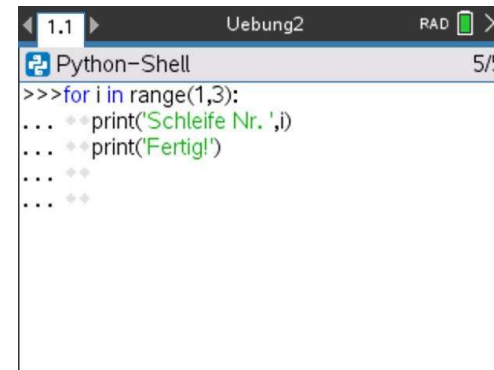
Man beginnt mit einer Steuerungs-Funktion, z.B. einer For() Schleife



Das Programm soll dreimal die Schleife durchlaufen, die Schleifennummer anzeigen und zum Schluß das Programmende melden.



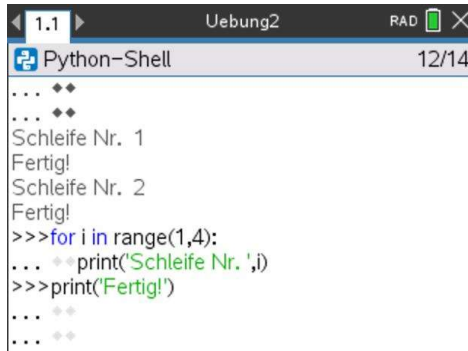
Die Shell sorgt automatisch für die richtige Formatierung und gibt Ausfüll-Tipps



Am Ende des gewünschten Programms drückt man 2x <Enter> und beim 3. <Enter> drücken startet das Programm...!



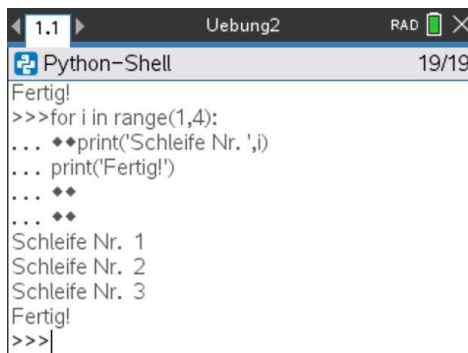
Mehrere Zeilen in der Shell ausführen? Kein Problem!



```
1.1 Uebung2 RAD 12/14
Python-Shell
... ♦♦
... ♦♦
Schleife Nr. 1
Fertig!
Schleife Nr. 2
Fertig!
>>>for i in range(1,4):
... ♦♦print("Schleife Nr. ",i)
>>>print("Fertig!")
... ♦♦
... ♦♦
```

Das Programm schreibt bei jedem Schleifendurchlauf 'Fertig!'. ??!!... Ich habe vergessen, die <print('Fertig!')> Zeile wieder zwei Tabulatoren herauszurücken, um das Schleifenende zu signalisieren!

Anders als in Basic oder Arduino wird eine For() Schleife bei einem Stop-Wert von n nur (n-1) mal durchlaufen. Das ist gewöhnungsbedürftig.



```
1.1 Uebung2 RAD 19/19
Python-Shell
Fertig!
>>>for i in range(1,4):
... ♦♦print("Schleife Nr. ",i)
... print("Fertig!")
... ♦♦
... ♦♦
Schleife Nr. 1
Schleife Nr. 2
Schleife Nr. 3
Fertig!
>>>|
```

Jetzt sieht es besser aus!



Starkes Stück! Listen in Python

```
1.1 Uebung3 RAD 9/9
Python-Shell
>>>l1=[8,7,6,5,4,3,2,1]
>>>l1
[8, 7, 6, 5, 4, 3, 2, 1]
>>>type(l1)
<class 'list'>
>>>l2=['Meine','Zahlenreihe']
>>>type(l2)
<class 'list'>
>>>|
```

Eine Liste mit Zahlen,
Eine zweite mit Strings

```
1.1 Uebung3 RAD 9/9
Python-Shell
>>>x=5
>>>y=3
>>>l4=[x+y,x-y,x*y,x/y]
>>>l4
[8, 2, 15, 1.6666666666666667]
>>>x=2
>>>l4
[8, 2, 15, 1.6666666666666667]
>>>|
```

Eine Liste kann auch aus
Formeln oder Funktionen
bestehen. Jedoch
werden die Formeln bei
Aufruf der Liste in Werte
verwandelt...

```
1.1 Uebung3 RAD 9/9
Python-Shell
>>>sorted(l1)
[1, 2, 3, 4, 5, 6, 7, 8]
>>>l3=sorted(l1)+l2
>>>l3
[1, 2, 3, 4, 5, 6, 7, 8, 'Meine', 'Zahlenreihe']
>>>l3=l2+sorted(l1)
>>>l3
['Meine', 'Zahlenreihe', 1, 2, 3, 4, 5, 6, 7, 8]
>>>|
```

Man kann Zahlen einer
Liste sortieren

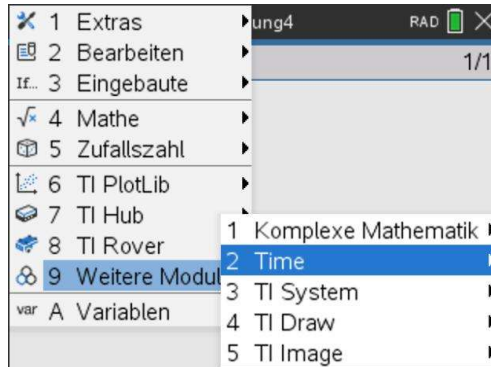
Und zu einer neuen,
gemischten Liste
kombinieren

```
1.1 Uebung3 RAD 12/12
Python-Shell
>>>y=3
>>>l4=[x+y,x-y,x*y,x/y]
>>>l4
[8, 2, 15, 1.6666666666666667]
>>>x=2
>>>l4
[8, 2, 15, 1.6666666666666667]
>>>l4=[x+y,x-y,x*y,x/y]
>>>l4
[5, -1, 6, 0.6666666666666666]
>>>|
```

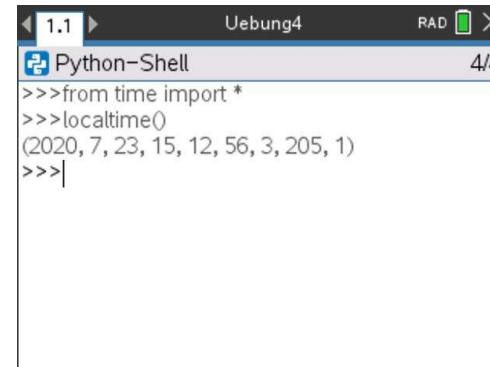
... und die Liste muss zur
Aktualisierung neu
aufgerufen werden!



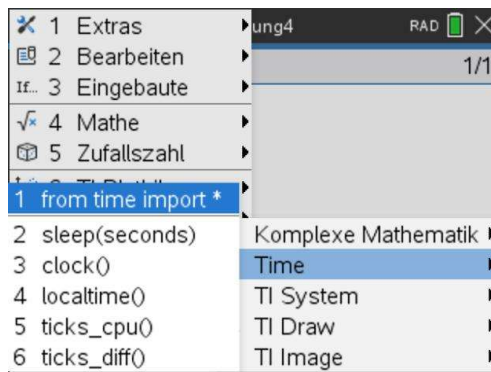
Wie spät ist es bitte? Das Modul "time"



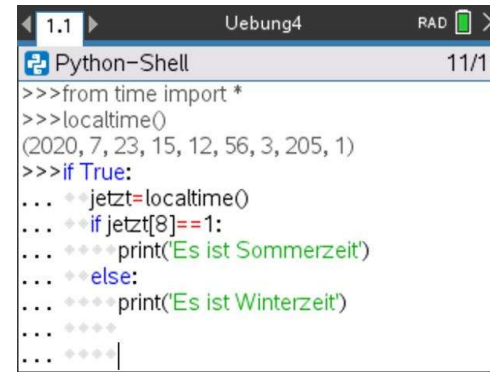
Unter <9> findet man weitere Module, u.a. auch das Modul 'time'.



localtime() zeigt den Kalender, Uhrzeit, Wochentag, Tag des Jahres und Sommer/Winterzeit an



Das Modul 'time' enthält wichtige Funktionen zum Thema Zeit(-messen)



Ein kleines Programm zur Prüfung, ob gerade Sommerzeit ist



Wie spät ist es bitte? Das Modul "time"

```
1.1 | Uebung4 | RAD | Python-Shell | 13/13
(2020, 7, 23, 15, 12, 56, 3, 205, 1)
>>>if True:
... ♦♦jetzt=localtime()
... ♦♦if jetzt[8]==1:
... ♦♦♦♦print('Es ist Sommerzeit')
... ♦♦else:
... ♦♦♦♦print('Es ist Winterzeit')
... ♦♦♦♦
... ♦♦♦♦
... ♦♦♦♦
Es ist Sommerzeit
>>>|
```

Im Oktober 2020 wird ein T³ Europe Seminar zum Thema 'time' Modul angeboten!

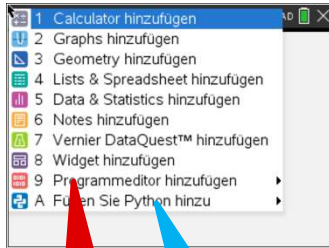


Einstieg in Python: Lust auf mehr!

- <menu> – einmal durch die Speisekarte
- Die Grafikoberfläche des TI-Nspire-CXII – Beispiel Mandelbrot
- Willkommen in der Open Source Welt! – Python Turtle
- Nicht genug TI-RGB Arrays in der Klasse? Kein Problem!



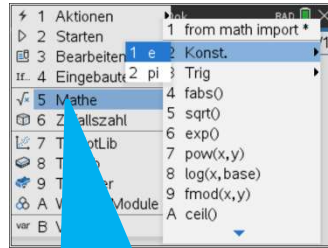
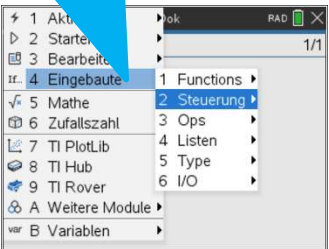
<menu> – einmal durch die Speisekarte



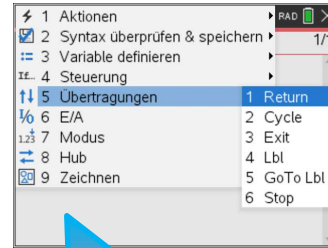
TI-Basic

Python

Python-Grundfunktionen

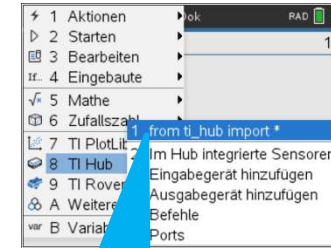
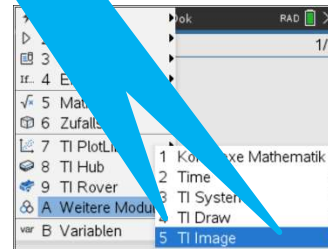
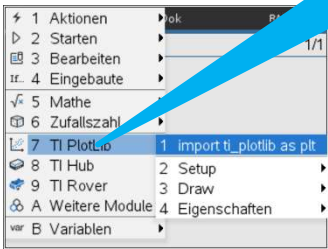


Funktionen aus dem Mathe Modul



Kontextabhängiges Menu (Basic/Python)

Drei Grafik-Module



MINT Module

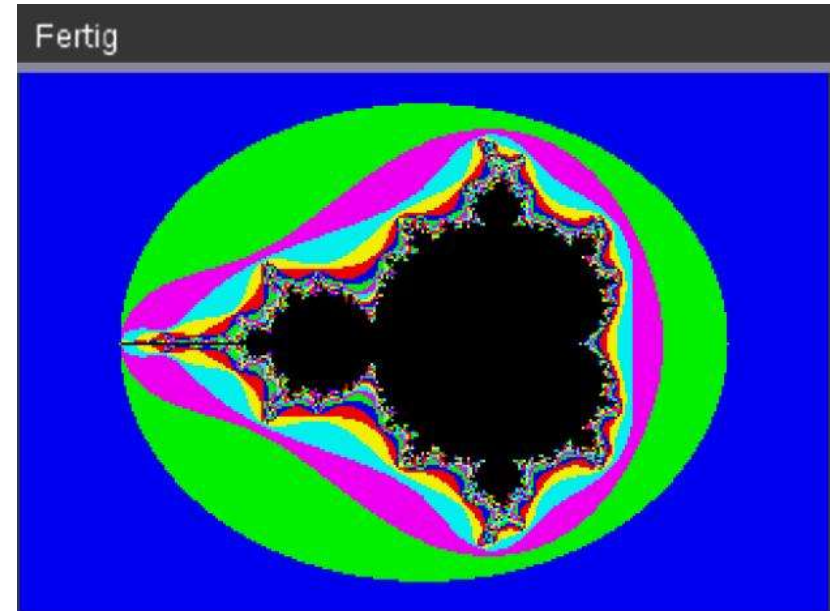


Die Grafikoberfläche – Beispiel Mandelbrot

```
mandel-v2 RAD 1/31
mandel.py
from cmath import *
from ti_plotlib import *
from ti_draw import plot_xy, use_buffer, paint_buf

def mycol(k):
    pal=[0,0,1,0,1,1,0,0]
    n=k%6
    color(240*pal[n],240*pal[n+1],240*pal[n+2])

def linspace(x1,x2,n):
    return [x1+k/n*(x2-x1) for k in range(0,n)]
```



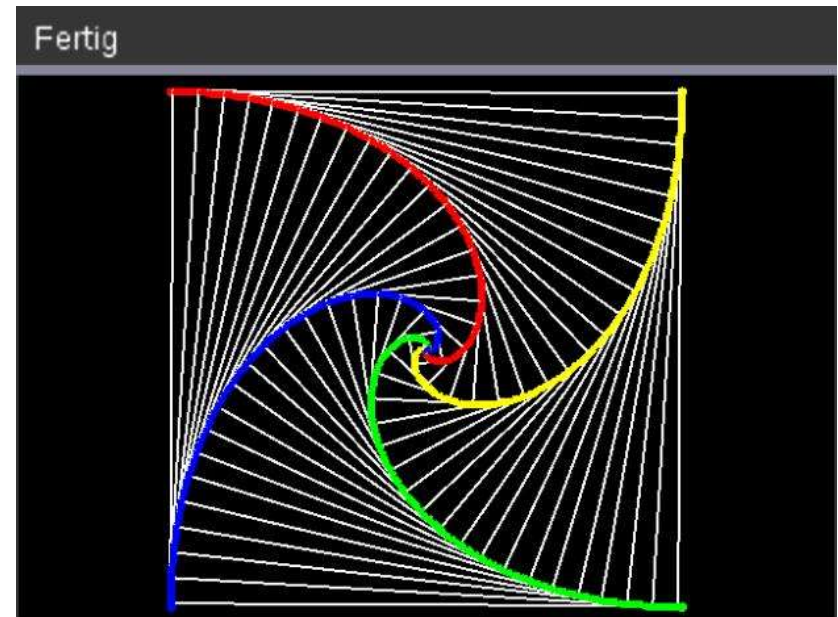
Code-Urheber: Didier Deses (T³ Ausbilder Belgien)

TI-Nspire CXII Grafikoberfläche (318 x 212 Pixel)



Willkommen in der Open Source Welt! – Turtle

```
1.1 1.2 1.3 turtles 4 a... are RAD X  
ants.py 1/35  
from cx_turtle2 import *  
from math import atan, pi  
# initial positions of ants determined by:  
p=100  
t1=turtle(-p,p,0)  
t1.color(255,0,0)  
t2=turtle(p,p,270)  
t2.color(255,255,0)  
t3=turtle(p,-p,180)  
t3.color(0,255,0)  
t4=turtle(-p,-p,90)
```



Code-Urheber: John Hanna (T³ Ausbilder USA)

Vier unabhängige Turtle-Objekte erzeugen die Grafik



Nicht genug TI-RGB Arrays in der Klasse? Kein Problem!

```
1.1 1.2 RGBArraySim RAD [X]  
rgbsim.py 13/29  
  
hub=True  
try:  
    from ti_hub import *  
    myrgb=rgb_array()  
except:  
    hub=False  
    from cx_rgbsim2 import *  
    myrgb=rgbsim()  
  
myrgb.set_all(0,255,0)
```



Je nachdem, ob der TI-Hub angeschlossen ist oder nicht, wird dem Objekt „myrgb“ entweder die Hub-Klasse oder die RGB-Simulator-Klasse zugewiesen

TI-Nspire CXII – Einstieg in Python: Seminararbeiten

- Portierung eines Open Source Python Spiels auf Nspire MicroPython
 - Beschreiben Sie Limitierungen bei der Portierung und wie Sie sie umgangen haben
- Portierung einer Open Source Arduino Bibliothek auf Nspire Python
 - Vergleichen Sie die Sprach-Eigenschaften von Arduino mit denen von Python
 - Beschreiben Sie Limitierungen bei der Portierung und wie Sie sie umgangen haben
- Entwicklung eines Rover Simulators auf Nspire CXII unter Verwendung der CXII Grafikoberfläche
 - Erstellen Sie ein funktionales und zeitliches Modell als Objektklasse, das die Rover Befehle 1:1 abbildet



TI-Nspire CXII – Einstieg in Python: Nützliche links

- T³ Europe – aufgezeichnete Webinare
 - Python on Nspire CXII (Speakers: Harshal Chhaya; Steve Debaugé; Michel Stella)
 - Why you should absolutely teach your student to code in Python (Abir Marina)

<https://www.t3europe.eu/en/t3-europe/webinars/on-demand>
- MicroPython documentation
<https://docs.micropython.org/en/latest/index.html>
- MicroPython Online-Nachschlagewerk mit Beispielen
<https://www.programiz.com/python-programming/first-program>
- Künstliche Intelligenz mit Python
https://www.tutorialspoint.com/artificial_intelligence_with_python/index.htm
- ... oder auch gerne jederzeit eine e-mail mit Fragen, Anregungen und Wünschen an: hm-hilbig@web.de

