Skripte/Funktionen mit Python Stand: 14.5.12

Wenn man Python-Code wiederholt ausführen will, z.B. Laden eines Datensatzes, plotten usw. dann ist das ein Script.
Das Script schreibt man in einem Texteditor und speichert es unter dem Namen '*scriptname.py*'.

Inhatl von '*stest.py*':

```
from pylab import *
d=loadtxt('line.dat')
plot(d[:,0],d[:,1],'*')
show()
```

Nach dem Starten von Ipython wird dieses Skript ausgeführt

> %run stest

Es erscheint der Plot, zudem ist das Array d im sogenannten 'interactive namespace' vorhanden und nutzbar.

>whos

```
In [3]: whos
Variable   Type       Data/Info
------------------------------
d          ndarray    40x2: 80 elems, type `float64`, 640 bytes
```

Wahlweise könnte man/frau auch python direkt mit dem Script aufrufen (Linux), auch dann wird das Script ausgeführt, eine Graphik erscheint.
python stest.py

Python-Code, der eine bestimmte Aufgabe ausführt, und den man/frau öfter braucht, sollte man in Funktionen auf der Festplatte speichern, z.B. eine Geraden-Funktion:

```
# -*- coding: utf-8 -*-
# Gerade:
# x  laufvariable
# m  Steigung
# b  Achsenabschnitt

def line(x,m,b):
        return x*m+b
```

Kommentare!

Funktionen-Definition

Rückgabewert

Speichere mit dem Namen '*line.py*'. Eine Funktion muss mit
> import line

in ipython bekannt gemacht werden. Eine Datei wird allerdings nicht als Funktion, sondern als Modul angesehen, d.h. Die Datei kann mehr als eine Funktion enthalten.

> ??line
zeigt Informationen über das geladene Modul.

```
# -*- coding: utf-8 -*-
# Gerade:
# x  laufvariable
# m  Steigung
# b  Achsenabschnitt

def line(x,m,b):
        return x*m+b
```

Das Modul wird im *'interactive namespace*' angezeigt:

```
In [8]: whos
Variable   Type      Data/Info
------------------------------
line       module    <module 'line' from 'line.py'>
```

Es werden aber keine Variablen, die innerhalb der Funktion benutzt werden, angezeigt (Unterschied zum Skript!)

```
In [9]: y=line.line(1,2,3)

In [10]: whos
Variable   Type      Data/Info
------------------------------
line       module    <module 'line' from 'line.py'>
y          int       -395
```

Wenn verlangt, aber die 'Return'- Rückgabewerte:

Die Funktionen des Moduls müssen mit Modulname.Funktion aufgerufen werden.

```
In [4]: line.line(1,2,3)
Out[4]: 5
```

Wenn an der Funktion line etwas geändert wurde, muss das Modu mit
> reload(line)
neu geladen werden.

Wenn das nicht geht, kann mit
> reset
der gesamte Interactive Namespace gelöscht werden!

```
In [11]: reset
Once deleted, variables cannot be recovered. Proceed (y/[n])?  y

In [12]: whos
Interactive namespace is empty.
```

Testen Sie das verhalten, wenn mehrere Funktionen in die Datei 'line.dat' geschrieben werden!