A better print statement

Small demo

1. What is the point excactly

While writing Python Code one usually inserts print statements such as this:

```
print ("I am here, doing this")
```

to help finding bugs.

This is a bad idea since one has to remove these statements for the production version of the program. Or make them conditional with something like this:

```
if (debug_variable):
    print ("I am here, doing this")
```

The demo program **demo_printest_1.py** demonstrates the use of the **class MyPrint**. Depending on the commandline parameter more or less debug output is written to the console **AND** to the logfile.

So use these commandline parms:

- none
- -d: minimal debug output
- -D more debug output
- -A even more debug output

The demo program **demo_printest_2.py** also demonstrates the use of the class MyPrint.
The program contains a programing error in this statement: nummer[4]=23
An exception is raised and this fact is also recorded in the logfile.

How to use the better print statement::

Use this for output that you always want to see in the log and on the screen

```
mypri.myprint(DEBUG_LEVEL0,"This Pi's IP-Adress: {}".format(ipadr))
```

Use this for output that only appears with commandline parm -d

```
mypri.myprint(DEBUG_LEVEL1, "I am here")
```

Use this for output that only appears with commandline parm -D

```
mypri.myprint(DEBUG_LEVEL2, "program started")
```

Use this for output that only appears with commandline parm -A

```
mypri.myprint(DEBUG_LEVEL3, "program ended")
```

In other words:
Using commandline parm - A gives you everything, this is for detailed debugging
Using commandline parm - D gives you some details but not everything
Using commandline parm -d gives you just a little more than normal

This, of course, depends on the proper use of the myprint statements and their DEBUG_LEVEL.

Note: I always include class definition files in a folder **sub**. You will find the class definition in the file myprint.py in this folder.. This file contains actually two classes: MyLog and MyPrint which inherits from MyLog.

Note on logfiles:

Checkout the docu on logfiles here: The class **MyLog** implements the actual logging.
Logging in Python

A maximum of **three** logfiles will be written to the current folder. Since this is a so called **rotating log** the folder will never overflow, each logfile will grow to max 30 kB. See definition in the MyLog class.

Also checkout my webpages:

Foto Galleries

Projects Page

YouTube Channel

September 2020, Peter K. Boxler