example8

May 7, 2018

1 Using the logging (WLOG)

Printing and logging is controlled via the `WLOG` function (`SpirouDRS.spirouCore.spirouLog.logger` aliased to `WLOG`)

The first thing that is needed is the import and alias to the logger function (`WLOG`):

```
In [3]: from SpirouDRS import spirouCore

        # Get Logging function
        WLOG = spirouCore.wlog
```

Then we can use the logger as follows:
`WLOG(level, program, message)`
and produces the following entry:

`` `HH:MM:SS.s - char | program | message` ``

where:

- level: (string) is the type of log message we are producing by default it can be one of the following "all", "info", 'warning", "error" or "" (a blank string). It also sets the "char" in the printed entry. This chooses the colour of the text
- program: (string) is the "program" message that is printed/logged (i.e. the recipe that is being run or any other custom string that is required)
- message: (string) this is the message part of the print statement/log statement.

Below we show each of these in use:

```
In [5]: # set up a program name
        program = "test program"
```

1.0.1 General/info message:

```
In [7]: WLOG('', program, 'This is a general message')
        WLOG('all', program, 'This is a general message')
        WLOG('info', program, 'This is an info message')
```

```
11:22:16.0 -   |test program|This is a general message
11:22:16.0 -   |test program|This is a general message
11:22:16.0 - * |test program|This is an info message
```

1.0.2 Warning message:

```
In [9]: WLOG('warning', program, 'This is a warning messagge')

11:23:07.0 - @ |test program|This is a warning messagge
```

1.0.3 Error message:

An error message (by default) has an added feature. An error will automatically stop the recipe. Therefore using a error logging should only be used when the recipes cannot continue afterwards.

In the DRS (in none interactive mode) this exit will be silent. If `DEBUG=1` the error will be handled and the user will be able to debug from the point of exit (unlike uncatch python errors).

```
In [12]: WLOG('error', program, 'This is an error message')

11:28:45.0 - ! |test program|This is an error message



        An exception has occurred, use %tb to see the full traceback.


        SystemExit: 1
```

1.1 Customising the logger

There are several ways these default settings can be changed/modified. They are: - changing the levels - changing the levels that exit python - changing which levels write to the standard output (the screen) - changing which levels write to the log file - changing the colours of the levels (when coloured text is enabled)

These are shown below and are changable in `SpirouDRS.spirouConfig.spirouConst.py`

```
In [14]: # noinspection PyPep8Naming
         def LOG_TRIG_KEYS():
             """
             The log trigger key characters to use in log. Keys must be the same as
             spirouConst.WRITE_LEVELS()

             i.e.

             if the following is defined:
             >> trig_key[error] = '!'
             and the following log is used:
             >> WLOG('error', 'program', 'message')
             the output is:
             >> print("TIMESTAMP - ! |program|message")
```

```
    :return trig_key: dictionary, contains all the trigger keys and the
                      characters/strings to use in logging. Keys must be the
                      same as spirouConst.WRITE_LEVELS()
    """
    # The trigger character to display for each
    trig_key = dict(all=' ', error='!', warning='@', info='*', graph='~')
    return trig_key


# noinspection PyPep8Naming
def WRITE_LEVEL():
    """
    The write levels. Keys must be the same as spirouConst.LOG_TRIG_KEYS()

    The write levels define which levels are logged and printed (based on
    constants "PRINT_LEVEL" and "LOG_LEVEL" in the primary config file

    i.e. if
    >> PRINT_LEVEL = 'warning'
    then no level with a numerical value less than
    >> write_level['warning']
    will be printed to the screen

    similarly if
    >> LOG_LEVEL = 'error'
    then no level with a numerical value less than
    >> write_level['error']
    will be printed to the log file

    :return write_level: dictionary, contains the keys and numerical levels
                         of each trigger level. Keys must be the same as
                         spirouConst.LOG_TRIG_KEYS()
    """
    write_level = dict(error=3, warning=2, info=1, graph=0, all=0)
    return write_level


# noinspection PyPep8Naming
def LOG_CAUGHT_WARNINGS():
    """
    Defines a master switch, whether to report warnings that are caught in

    >> with warnings.catch_warnings(record=True) as w:
    >>     code_that_may_gen_warnings

    :return warn: bool, if True reports warnings, if False does not
    """
```

```
    # Define whether we warn
    warn = True
    return warn


# noinspection PyPep8Naming
def COLOUREDLEVELS():
    """
    Defines the colours if using coloured log.
    Allowed colour strings are found here:
            see here:
            http://ozzmaker.com/add-colour-to-text-in-python/
            or in spirouConst.bcolors (colour class):
                HEADER, OKBLUE, OKGREEN, WARNING, FAIL,
                BOLD, UNDERLINE

    :return clevels: dictionary, containing all the keys identical to
                     LOG_TRIG_KEYS or WRITE_LEVEL, values must be strings
                     that prodive colour information to python print statement
                     see here:
                         http://ozzmaker.com/add-colour-to-text-in-python/
    """
    # reference:
    # http://ozzmaker.com/add-colour-to-text-in-python/
    clevels = dict(error=BColors.FAIL,  # red
                   warning=BColors.WARNING,  # yellow
                   info=BColors.OKGREEN,  # green
                   graph=BColors.OKBLUE,  # green
                   all=BColors.OKGREEN)  # green
    return clevels


# defines the colours
class BColors:
    HEADER = '\033[95;1m'
    OKBLUE = '\033[94;1m'
    OKGREEN = '\033[92;1m'
    WARNING = '\033[93;1m'
    FAIL = '\033[91;1m'
    ENDC = '\033[0;0m'
    BOLD = '\033[1m'
    UNDERLINE = '\033[4m'
```