example2

May 7, 2018

1 What is a parameter dictionary?

In the main recipes a variable "p" is used often. This is a instance of a custom dictionary called a parameter dictionary or "ParamDict" for short.

We use this to store all constants and keep track (as a developer) of where constants were defined or modified.

Below we explore "p" in more detail.

First we begin as if we were in cal_DARK_spirou, importing all modules required to run the "set up" procedure.

```
In [21]: # imports as in cal_DARK_spirou.py
         from __future__ import division

         from SpirouDRS import spirouConfig
         from SpirouDRS import spirouCore
         from SpirouDRS import spirouStartup

         # ========================================
         # Define variables
         # ========================================
         # Name of program
         __NAME__ = 'cal_DARK_spirou.py'
         # Get version and author
         __version__ = spirouConfig.Constants.VERSION()
         __author__ = spirouConfig.Constants.AUTHORS()
         __date__ = spirouConfig.Constants.LATEST_EDIT()
         __release__ = spirouConfig.Constants.RELEASE()
         # Get Logging function
         WLOG = spirouCore.wlog
         # Get plotting functions
         sPlt = spirouCore.sPlt
```

The next section is generally inside the "main" function of a recipe, here we keep it outside so we can show the features easily. The setup of normal routines uses three functions "Begin", "LoadArguments" and "InitialFileSetup". We will go through each using cal_DARK_spirou.py as an example.

1.1 The begin function

This is the first function that will be run in any recipe "main" function. It loads all the primary constants (from the config.py file) as well as displaying the startup title and paths defined in the config.py file.

```
In [2]: # Begin function
        p = spirouStartup.Begin()
```

```
12:05:18.0 -   || ****************************************
12:05:18.0 -   || * SPIROU @(#) Geneva Observatory (0.1.029)
12:05:18.0 -   || ****************************************
12:05:18.0 -   ||(dir_data_raw)      DRS_DATA_RAW=/scratch/Projects/spirou_py3/data/raw
12:05:18.0 -   ||(dir_data_reduc)    DRS_DATA_REDUC=/scratch/Projects/spirou_py3/data/reduced
12:05:18.0 -   ||(dir_calib_db)      DRS_CALIB_DB=/scratch/Projects/spirou_py3/data/calibDB
12:05:18.0 -   ||(dir_data_msg)      DRS_DATA_MSG=/scratch/Projects/spirou_py3/data/msg
12:05:18.0 -   ||(print_level)       PRINT_LEVEL=all         %(error/warning/info/all)
12:05:18.0 -   ||(log_level)         LOG_LEVEL=all         %(error/warning/info/all)
12:05:18.0 -   ||(plot_graph)        DRS_PLOT=1            %(def/undef/trigger)
12:05:18.0 -   ||(used_date)         DRS_USED_DATE=undefined
12:05:18.0 -   ||(working_dir)       DRS_DATA_WORKING=/scratch/Projects/spirou_py3/data/tmp
12:05:18.0 -   ||                    DRS_INTERACTIVE is not set, running on-line mode
12:05:18.0 -   ||                    DRS_DEBUG is set, debug mode level:1
```

Taking a look at what is contained in "p" at this point is shown below. It should be representative of the constants defined in constants.py where type casting to floats/ints and booleans has been done where possible.

```
In [3]: p
```

```
Out[3]: {'COLOURED_LOG': True,
         'DRS_CALIB_DB': '/scratch/Projects/spirou_py3/data/calibDB',
         'DRS_CONFIG': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/',
         'DRS_DATA_MSG': '/scratch/Projects/spirou_py3/data/msg',
         'DRS_DATA_RAW': '/scratch/Projects/spirou_py3/data/raw',
         'DRS_DATA_REDUC': '/scratch/Projects/spirou_py3/data/reduced',
         'DRS_DATA_WORKING': '/scratch/Projects/spirou_py3/data/tmp',
         'DRS_DEBUG': 1,
         'DRS_INTERACTIVE': 0,
         'DRS_MAN': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/man/',
         'DRS_NAME': 'SPIROU',
         'DRS_PLOT': 1,
         'DRS_ROOT': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/',
         'DRS_USED_DATE': 'undefined',
         'DRS_VERSION': '0.1.029',
         'ICDP_NAME': 'constants_SPIROU.py',
         'LOG_LEVEL': 'all',
         'PRINT_LEVEL': 'all',
```

```
 'SPECIAL_NAME': 'special_config_SPIROU.py',
 'TDATA': '/scratch/Projects/spirou_py3/data/'}
```

Note we can also look at where each key was defined:

```
In [4]: p.sources

Out[4]: {'COLOURED_LOG': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'DRS_CALIB_DB': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'DRS_CONFIG': 'spirouConfig.py.check_params()',
 'DRS_DATA_MSG': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'DRS_DATA_RAW': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'DRS_DATA_REDUC': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'DRS_DATA_WORKING': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'DRS_DEBUG': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'DRS_INTERACTIVE': 'spirouConfig.py.check_params()',
 'DRS_MAN': 'spirouConfig.py.check_params()',
 'DRS_NAME': 'spirouConfig.Constants',
 'DRS_PLOT': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'DRS_ROOT': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'DRS_USED_DATE': 'spirouConfig.py.check_params()',
 'DRS_VERSION': 'spirouConfig.Constants',
 'ICDP_NAME': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'LOG_LEVEL': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'PRINT_LEVEL': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'SPECIAL_NAME': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py',
 'TDATA': '/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/config.py'}
```

1.2 LoadArguments

This function is the second argument to run. It deals with loading of the arguments both from the recipes main() arguments and from run time (i.e. after the file name when called from python/ipython or a console).

By default the user will either have to define arguements from the command line or in the function call to the recipes "main" function.

i.e. for cal_DARK_spirou.py one could call the following from the command line: >> cal_DARK_Spirou.py 20170710 dark_dark

or in python:

cal_DARK_spirou.main(night_name=nightname, files=filenames)

```
In [5]: # we will assume we are inside the cal_DARK_Spirou.main function and have called
        #   the arguments correctly:
        night_name = '20170710'
        files = ['dark_dark02d406.fits']

        # Notebook only code
        import sys
        sys.argv = ['cal_DARK_spirou.py']
```

we can then call the LoadArguments function

```
In [6]: p = spirouStartup.LoadArguments(p, night_name, files)
```

```
12:05:27.0 -   |cal_DARK_spirou|Now running : cal_DARK_spirou on file(s):
12:05:27.0 -   |cal_DARK_spirou|On directory /scratch/Projects/spirou_py3/data/raw/20170710
12:05:27.0 -   |cal_DARK_spirou|ICDP_NAME loaded from: /scratch/Projects/spirou_py3/spirou_py3/I
```

Taking a look a "p" now we can see it has added the "run time" arguements and additional constants from the constants_SPIROU.py file. Note below we only print the key names.

```
In [7]: print(list(p.keys()))
```

```
['DRS_PLOT', 'DRS_DEBUG', 'TDATA', 'DRS_ROOT', 'DRS_DATA_RAW', 'DRS_DATA_REDUC', 'DRS_CALIB_DB',
```

Sources are set for all variables, some examples are shown below. Note that strings are evaluated (i.e. "IC_CCDX_RED_LOW = 2028" value is a int not a string.

```
In [8]: keys = ['DRS_NAME', 'IC_CCDX_RED_LOW', 'IC_CCDX_BLUE_LOW', 'KW_VERSION']

        for key in keys:
            args = [key, p[key], type(p[key]), p.sources[key]]
            print('\n - key = "{0}"\n\tvalue={1}\n\ttype={2}\n\tsource={3}'.format(*args))
```

```
 - key = "DRS_NAME"
        value=SPIROU
        type=<class 'str'>
        source=spirouConfig.Constants

 - key = "IC_CCDX_RED_LOW"
        value=2028
        type=<class 'int'>
        source=/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/constants_SPIROU.py

 - key = "IC_CCDX_BLUE_LOW"
        value=1848
        type=<class 'int'>
        source=/scratch/Projects/spirou_py3/spirou_py3/INTROOT/config/constants_SPIROU.py

 - key = "KW_VERSION"
        value=['VERSION', 'SPIROU_0.1.029', 'DRS version']
        type=<class 'list'>
        source=spirouKeywords.py
```

2 InitialFileSetup

This is used to set up files and check them for required prefixes, and loads the calibration database if it is reuiqred. Below we use the example of cal_loc_RAW_spirou.py where either "dark_flat" or "flat_dark" is required (and a set of different fiber parameter are required for each - i.e. "dark_flat" means we are using fiber "C" and "flat_dark" means we are using fibers "AB")

```
In [10]: # cal_loc_RAW requires some additional imports
         from __future__ import division

         from SpirouDRS import spirouImage
         from SpirouDRS import spirouLOCOR
         from SpirouDRS import spirouStartup

         # redefine arguments
         night_name = '20170710'
         files = ['flat_dark02f10.fits', 'flat_dark03f10.fits',
                  'flat_dark04f10.fits', 'flat_dark05f10.fits',
                  'flat_dark06f10.fits']

         # ----------------------------------------------------------------------
         # Set up
         # ----------------------------------------------------------------------
         # get parameters from config files/run time args/load paths + calibdb
         p = spirouStartup.Begin()
         p = spirouStartup.LoadArguments(p, night_name, files)
         # run specific start up
         params2add = dict()
         params2add['dark_flat'] = spirouLOCOR.FiberParams(p, 'C')
         params2add['flat_dark'] = spirouLOCOR.FiberParams(p, 'AB')
         p = spirouStartup.InitialFileSetup(p, kind='localisation',
                                            prefixes=['dark_flat', 'flat_dark'],
                                            add_to_p=params2add, calibdb=True)
```

```
12:05:37.0 -   || ****************************************
12:05:37.0 -   || * SPIROU @(#) Geneva Observatory (0.1.029)
12:05:37.0 -   || ****************************************
12:05:37.0 -   ||(dir_data_raw)      DRS_DATA_RAW=/scratch/Projects/spirou_py3/data/raw
12:05:37.0 -   ||(dir_data_reduc)    DRS_DATA_REDUC=/scratch/Projects/spirou_py3/data/reduced
12:05:37.0 -   ||(dir_calib_db)      DRS_CALIB_DB=/scratch/Projects/spirou_py3/data/calibDB
12:05:37.0 -   ||(dir_data_msg)      DRS_DATA_MSG=/scratch/Projects/spirou_py3/data/msg
12:05:37.0 -   ||(print_level)       PRINT_LEVEL=all         %(error/warning/info/all)
12:05:37.0 -   ||(log_level)         LOG_LEVEL=all         %(error/warning/info/all)
12:05:37.0 -   ||(plot_graph)        DRS_PLOT=1            %(def/undef/trigger)
12:05:37.0 -   ||(used_date)         DRS_USED_DATE=undefined
12:05:37.0 -   ||(working_dir)       DRS_DATA_WORKING=/scratch/Projects/spirou_py3/data/tmp
12:05:37.0 -   ||                    DRS_INTERACTIVE is not set, running on-line mode
12:05:37.0 -   ||                    DRS_DEBUG is set, debug mode level:1
```

```
12:05:37.0 -   |cal_DARK_spirou|Now running : cal_DARK_spirou on file(s):
12:05:37.0 -   |cal_DARK_spirou|On directory /scratch/Projects/spirou_py3/data/raw/20170710
12:05:37.0 -   |cal_DARK_spirou|ICDP_NAME loaded from: /scratch/Projects/spirou_py3/spirou_py3/I
12:05:37.0 - * |cal_DARK_spirou|Correct type of image for localisation (dark_flat or flat_dark)
12:05:38.0 -   |cal_DARK_spirou|Calibration file: 20170710_flat_flat02f10_badpixel.fits already
12:05:38.0 -   |cal_DARK_spirou|Calibration file: 20170710_flat_dark02f10_blaze_AB.fits already
12:05:38.0 -   |cal_DARK_spirou|Calibration file: 20170710_dark_flat02f10_blaze_C.fits already e
12:05:39.0 -   |cal_DARK_spirou|Calibration file: 20170710_dark_dark02d406.fits already exists -
12:05:39.0 -   |cal_DARK_spirou|Calibration file: 20170710_flat_dark02f10_flat_AB.fits already e
12:05:39.0 -   |cal_DARK_spirou|Calibration file: 20170710_dark_flat02f10_flat_C.fits already ex
12:05:39.0 -   |cal_DARK_spirou|Calibration file: 20170710_flat_dark02f10_loco_AB.fits already e
12:05:39.0 -   |cal_DARK_spirou|Calibration file: 20170710_dark_flat02f10_loco_C.fits already ex
12:05:39.0 -   |cal_DARK_spirou|Calibration file: 20170710_flat_dark02f10_order_profile_AB.fits
12:05:40.0 -   |cal_DARK_spirou|Calibration file: 20170710_dark_flat02f10_order_profile_C.fits a
12:05:40.0 -   |cal_DARK_spirou|Calibration file: 20170710_fp_fp02a203_tilt.fits already exists
12:05:40.0 -   |cal_DARK_spirou|Calibration file: spirou_wave_ini3.fits already exists - not cop
12:05:40.0 -   |cal_DARK_spirou|Calibration file: 2017-10-11_21-32-17_hcone_hcone02c406_wave_AB.
12:05:40.0 -   |cal_DARK_spirou|Calibration file: spirou_wave_ini3.fits already exists - not cop
12:05:40.0 -   |cal_DARK_spirou|Calibration file: 2017-10-11_21-32-17_hcone_hcone02c406_wave_C.f
```

The DRS is now set up and if "calibdb" was True then we have access to the full calibration database (using "fitsfilename" as the reference to sort between files)

```
In [12]: print('\nfitsfilename: {0}'.format(p['fitsfilename']))
         print('\narg_file_names: {0}'.format(p['arg_file_names']))
         print('\ncalib db: {0}'.format(p['calibdb']))
```

```
fitsfilename: /scratch/Projects/spirou_py3/data/raw/20170710/flat_dark02f10.fits

arg_file_names: ['flat_dark02f10.fits', 'flat_dark03f10.fits', 'flat_dark04f10.fits', 'flat_dark

calib db: {'BADPIX': ['20170710', '20170710_flat_flat02f10_badpixel.fits'], 'BLAZE_AB': ['201707
```

2.1 Defining and updating variables using ParamDict

One will need to update and set variables in the parameter dictionary at various points in a recipe. When one does this one should update the source(s). This is shown below. It is also possible to update/add multiple variables with one source.

```
In [13]: # set new variable
         p['MY_NEW_VARIABLE'] = 3.141592
         p.set_source('MY_NEW_VARIABLE', 'myprogram.myfunction1()')

         # update current variable and append source
         p['FITSFILENAME'] = 'newfile2'
         p.append_source('FITSFILENAME', 'myprogram.myfunction2()')
```

```
        # add multiple variables with same source name
        p['VAR1'] = 1
        p['VAR2'] = 2
        p['VAR3'] = 3
        p.set_sources(['VAR1', 'VAR2', 'VAR3'], 'myprogram.myfunction3()')

        p.append_sources(['VAR1', 'VAR3'], 'myprogram.main()')
```

Below we show the output of each new key in "p"

```
In [14]: keys = ['MY_NEW_VARIABLE', 'FITSFILENAME', 'VAR1', 'VAR2', 'VAR3']

         for key in keys:
             args = [key, p[key], type(p[key]), p.sources[key]]
             print('\n - key = "{0}"\n\tvalue={1}\n\ttype={2}\n\tsource={3}'.format(*args))


 - key = "MY_NEW_VARIABLE"
       value=3.141592
       type=<class 'float'>
       source=myprogram.myfunction1()

 - key = "FITSFILENAME"
       value=newfile2
       type=<class 'str'>
       source=spirouStarup.py.get_call_arg_files_fitsfilename() myprogram.myfunction2() myprogr

 - key = "VAR1"
       value=1
       type=<class 'int'>
       source=myprogram.myfunction3() myprogram.main()

 - key = "VAR2"
       value=2
       type=<class 'int'>
       source=myprogram.myfunction3()

 - key = "VAR3"
       value=3
       type=<class 'int'>
       source=myprogram.myfunction3() myprogram.main()
```

2.2 Error and exception handling

Constants and variables is one large area where errors and exceptions can be caused, be it from incorrectly defined constants entered by the users in the confirugations files, configuration files missing, illegal characters or a developer trying to access a key that does not exist. The parameter

dictionary was designed with this in mind. It will use a special custom exception class (ConfigError) to guide the user/developer to a solution when an error/exception is generated, it is also designed to work with the WLOG logging function to provide easy feedback to the user.

```
In [15]: # try to access a undefined variable
         my_var = p['random']


         ---------------------------------------------------------------------------

         KeyError                                  Traceback (most recent call last)

         /scratch/Projects/spirou_py3/spirou_py3/INTROOT/SpirouDRS/spirouConfig/spirouConfig.py i
         145         try:
     --> 146             return super(ParamDict, self).__getitem__(key)
         147         except KeyError:


         KeyError: 'RANDOM'


     During handling of the above exception, another exception occurred:


         ConfigError                               Traceback (most recent call last)

         <ipython-input-15-9d5ebfbc5eff> in <module>()
           1 # try to access a undefined variable
     ----> 2 my_var = p['random']


         /scratch/Projects/spirou_py3/spirou_py3/INTROOT/SpirouDRS/spirouConfig/spirouConfig.py i
         148             emsg = ('Config Error: Parameter "{0}" not found in parameter '
         149                     'dictionary')
     --> 150             raise ConfigError(emsg.format(oldkey), level='error')
         151
         152     def __setitem__(self, key, value, source=None):


         ConfigError: level=error: Config Error: Parameter "random" not found in parameter dictio
```

This can be caught and piped cleanly to the logging system as below.
When a ConfigError is raise it has two attributes: "level" and "message"
(i.e. ConfigError as e --> e.level and e.message)
Note that the "level" of an exception is set to "error" and thus WLOG will exit.

```
In [22]: from SpirouDRS.spirouConfig import ConfigError
```

```
        # try to get a constant
        try:
            my_var = p['random']
        # catch the ConfigError
        except ConfigError as e:
            print("\nlevel = {0}".format(e.level))
            print("\nmessage = {0}".format(e.message))
            # using with log
            print("\n Log message:\n")
            WLOG(e.level, 'program', e.message)
```

```
level = error

message = Config Error: Parameter "random" not found in parameter dictionary

 Log message:

12:06:54.0 - ! |program|Config Error: Parameter "random" not found in parameter dictionary
        Error found and running in DEBUG mode



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


        SystemExit: 1



/scratch/bin/anaconda3/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2918: UserWa
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```