**P717\_Python\_README.txt**

Python programs provided AS-IS with no guarantee of correctness, fit for purpose or warranty of any kind. These programs were used during development of the IOGP p717 format specification, user guide and sample data.

Bert Kampes, 2018-08-10

**Programs**

* P717writer reads a P72000 file and outputs a rudimentary p717. Not fully correct. Intended to help development of the sample data files. Intended to be interactive where the end user is asked to provide EPSG code and axes order for example (which are not part of P72000).
* P717reader reads a P717 file and creates a human readable report. It verifies records it reads against the expected format (not 100%). It is intended to be helpful to spot format issues and could be developed into a more advanced checker.

**Source files:**

* P717writer.py: Main program. It has a basic P72000 reader internally and reads in variables into P717 objects, which are then written out to P717 file.
* P717reader.py: Main program. It loops over the input file twice. First to read and define all mandatory objects, second to parse all optional rows. Data is organized in objects, the Poslog being the central entity that is linked to Survey, Wellbore, etc. It then writes a report for each survey.
  + Pretty stable but needs an update and better looking output with real input data. First focus on a p72000top717 converter so these files can be made.
* P717records.py: Format spec + helper functions such as “is\_record” to check a record against spec. This is a useful file to start with as it contains all definitions.
  + Records.py is stable, unless of course the spec changes.
* P717classes.py: contains the class definitions, e.g., for a WELL, a SURVEY, etc. Each class contains “set” functions to parse the csv rows that were read in the calling program and “writer” functions to write to file. Additionally classes have helper functions.
  + Classes.py is pretty stable but needs some refactoring and demo/pretty printers added (needs to be updated obviously if spec changes)
* P717epsg.py: Basic sample how to get WKT from EPSG over the internet. Intended to be included as a way to dump the CRS definition as comment records into P717 if a full definition is not available.

**Input test data:**

* Data1.csv is an example input data set in p717 format. (actually not included anymore – outdated)… to be created again…

**Outputs:**

* P7writer
  + screen output: DOS prompt
  + debug log file: p717writer.log
  + p717 file. This file can be input in the p717reader.
* P7reader
  + screen output: DOS prompt
  + debug log file: p717reader.log
  + survey reports: e.g. “p717reader-survey1\_2018-01-07.txt”: txt file human readable report for each survey in the input p717 file.

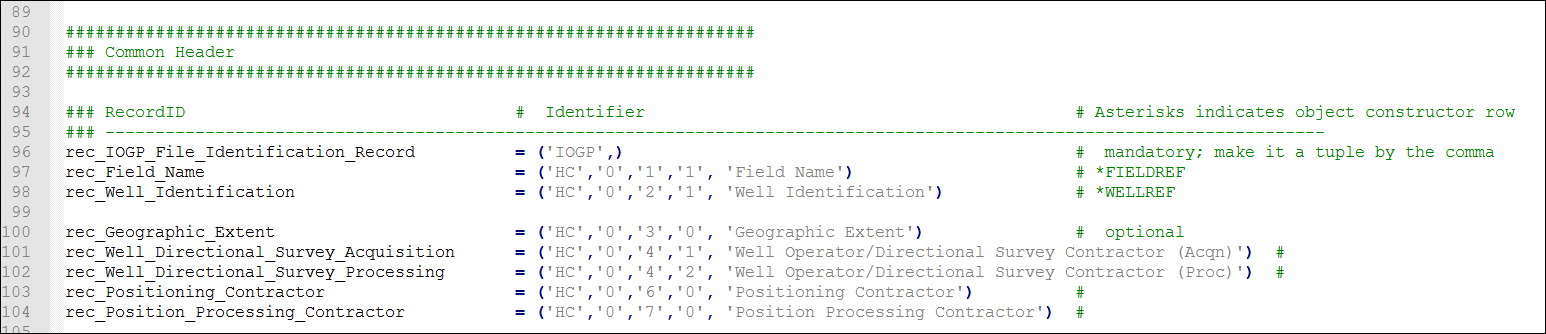
**Remaining Work Includes:**

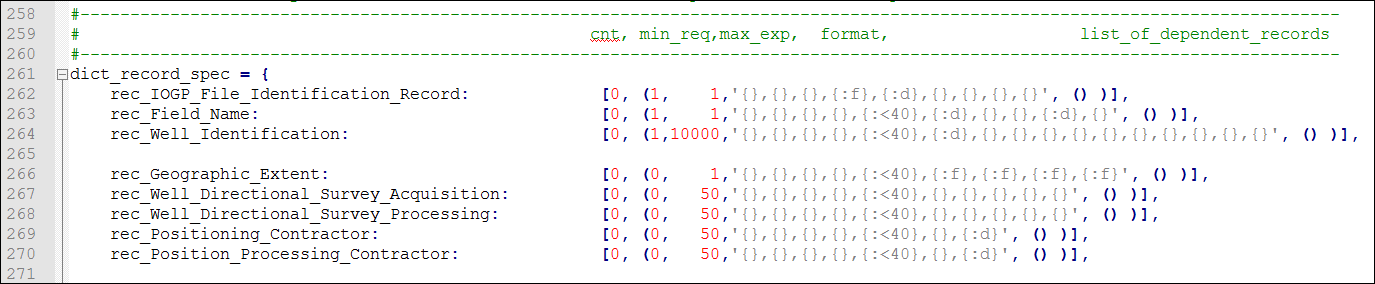
* Create main() with proper argument passing from command line
* Create demo (unit test) functions that writes a basic p717 file from stored strings internally. This requires to create set\_rec\_\* (the current constructor rows) which are called in the constructor if a row is passed as argument. Then a “demo” function that sets the class functions and calls the write\_rec\_\*() functions to output a basic file.
* Improve the p717reader.py to output a better looking report. Current known issue it uses SURVEY as central object and needs to iterate over POSLOG dict objects.
* This is a task that can be done in relative isolation from the other work on the library.
* Create P72000 reader stand alone.
* Create/update p72000top717 writer.py: The writer program should create empty (ish) objects for a single survey. Then populate them with the p72000 variables (i.e., fill the relevant variables in the p717 classes). Then call the writer class functions in correct order to output (or write a p717 writer that checks if records are filled and write them out; and call that function…)

**Installation:**

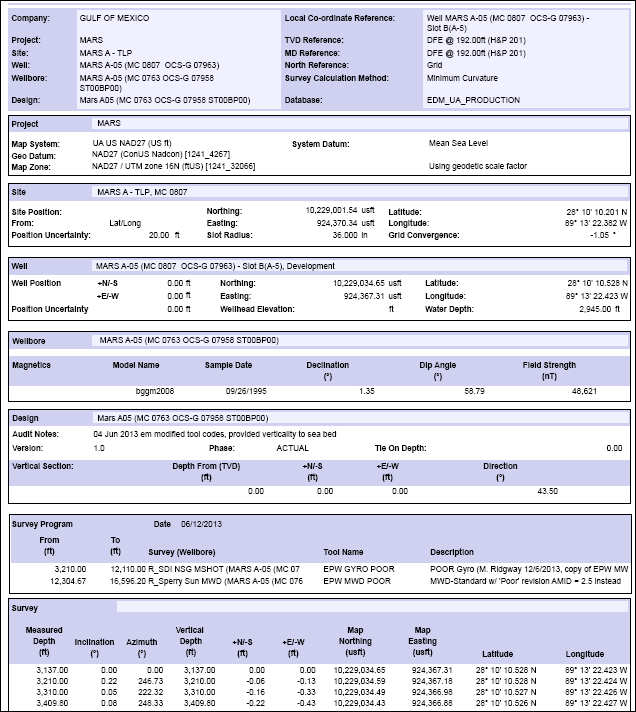
* Notepad++ editor was used during development.
* Install python 3 on windows (www.python.org)
* In Windows, open a CMD (DOS) prompt. Change directories to the folder and type
  + python p717writer.py data1.p72000 (outputs data1.p72000.p717)
  + python p717reader.py data1.p7200.p717 (outputs to screen and report to a .txt file)

Note: file p717records.py contains the format spec, e.g.,

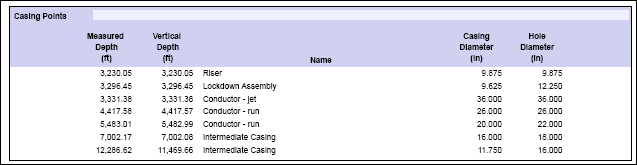




Example report structure for p717reader.py



…



…

