# Ref step code

Versions:

* Python 2.7
* PyVisa 1.8
* Python(x,y) 2.7.10.0 which includes:
  + wxPython 2.8.12.1-1

NI VISA:

* NIVISA needs to be downloaded, and then the IEEE488 plugin too, both files are very large. They depend on the .NET studios for the C++ libraries. Version used was “NI-488.2 for Windows, version 15.0”

Basic description of the code:

* RefStep.py – the main program
* noname.py – the GUI as generated with wxFormBuilder (note version 3.5, usually incompatible with other versions)
* analysis.py – the analysis object
* graph\_data.py – a graph that presents measurement results
* gpib\_data.py – the main thread of IEEE488 actions, reads through table and controls the instruments according to the table. Instrument objects return strings that get sent to the try\_command function.
* gpib\_inst.py – an instrument class for all instruments, returns strings that need to be evaluated/executed by the gpib\_data thread.
* stuff.py – methods and classes for threads launched by the GUI
* pywxgrideditmixin.py – an enhancement for wxGrid that facilitates cut/paste by region
* tables.py looks after putting Excel data into wxGrids
* GridMaker.py is a class for printers to grid, it checks the grid size prior to printing and updates it if necessary. Allows for printing of a whole column or row.

Initially, a user is expected to upload a dictionary file to the program. This file will be used for communicating with the instruments. Here a data table can be uploaded too, containing the information for the IEE thread to step through. This file is not necessary; it can be generated by the program using the “generate table” button. This operation is handled within the main thread of the program, and relies on the ranges of the instruments in order to generate an appropriate ref-step table. The table can be used to calibrate either the ranges of the DVM or of Source X. These ranges can be untrue to the ranges of the instrument, as long as each specified range is enveloped by a real instrument range (or else overload and out of range errors will occur).

The run button will start the worker thread reading through the table with an optional safety check routine.

## The worker thread

The current worker thread does safety checks (are instruments wired correctly and are they responding?) then goes into reading the table for instructions. Each instruction is sent to the instrument class, and instruments return a three element list with a success status, a string to print to the log files and a result (if this was a reading). The “com” function calls an abort on the string if the commands were not successful.

Also in the thread, there are several points in which it needs to wait, those are made into loops that continuously check if the thread wants to abort until the wait time is up. This is so that the thread can be aborted without waiting for it to finish its pause.

The thread saves a log file of all commands as a text document, and an excel file containing a copy of the control table with each individual data point from each measurement printed to the right.

## Data analysis

“Analyse data” is executed from the main thread. It creates an analysis.Analyser object that requires the name of the raw data file to be analysed.

The analysis object reads the table and saves the entire thing as an array. It then attempts the analysis for both the DVM and Source X.

* Reading from the start, it identifies the first mid point and finish points of each sequence, by looking for three repeated settings on source X.
* The midpoint and start point are used to infer the end point, so the ascent and decent must be symmetric.
* For start and end points, all readings between are turned into GTC objects using the number of readings, standard deviation and mean.
* Such a list of GTC object is sent to be split up into data applicable to the meter ratios, and data for source X (of the first four data points, each instrument has one which they do not use in the ratio formula).
* Computes linearity and gain ratios for the ascending and descending sections for both the meter and the source X, using the same data, in separate functions.
* These ratios are printed to the results sheet in the same workbook
* The row below that which was previously the last row becomes the new start row for the next set of measures and the process repeats.

It computes the gain ratio and linearity ratios using the GTC. It is possible to move the entire computation to a separate thread, instead of an object within the main thread. However the computations are quite fast so no noticeable freeze time is induced.

## Generate table

A table can be generated based on the ranges of the instruments, this is done using GridMaker.py. It initially computes the ref step, as a tenth (or otherwise) of the difference between the top of a particular range and the bottom of the range. In most cases the bottom of the range is zero.

## Things to possibly add or change:

* Temperature and relative humidity can be recorded, they could be given as a key word argument to the instrument class and if no query exists for them, the thread skips the command.

## Bugs (that I know of)

Plot stops after many data points for some reason. Memory leak? Couldn’t track it down with the basic task manager, and don’t really know what happens when so many graphs get repeatedly made, are they saved somewhere in the memory?

After a reset, the plot does not erase previous data.

When a large table is loaded or created, the wx table creates its own secondary scroll bar. When you mouse over that extra scroll bar the entire computer screen turns black. This is some wx magic that I couldn’t solve, perhaps its due to a cap on the size of the scroll windows or something.