Leveraging Python for Extending the Capability of MS Software Applications

Stacey Simonoff, Don Kuehl, Yongdong Wang
Cerno Bioscience, Las Vegas, NV USA

MP 391

Introduction

MS Instrument and software vendors have long used various embedded language extensions to customize and/or automate MS software applications. While useful for simple automation, reporting and calculation tasks, they fall short of the heavy lifting commonly required for efficient data processing of the large and complex data sets created by MS.

The Open Source Python programming language[1] is the most popular scientific programming language in the world. It provides a gateway into industrial strength, computationally intensive applications such as AI, statistical modeling, computational chemistry, and graphics. Companies such as Google and NASA have even adopted it for production environments. Python is relatively easy to learn as an interpreted language but can also be compiled for greater performance. We will describe, compare, and contrast Python integration for MS software applications.

Method

The software used for this project was Cerno's MassWorks (Host) which provides various calibration and data analysis features for MS. The underlying foundation for MS specific computational functions is a collection of C++ DLLs. The libraries were wrapped in a Python layer to allow fast execution from Python. Any data being analyzed in the Host can be processed by a custom Python application (PyApp) by accessing a simple pop-up menu which transfers the data and associated information to the PyApp for custom processing. From the PyApp, the extensive Python library of graphics and computational tools such as MatPlotLib, NumPy, SciPy are accessible, as well as libraries exposed by a custom Python API (see Figure 1). The system as a whole has been coined "MassLab", providing a quick and easy laboratory for experimenting with specialized MS analysis.

Results and Discussion

MS data is opened from the Host for accurate mass calibration and analysis. Since this software is vendor agnostic and can directly open MS data from most major MS vendors directly, it provides a good platform for generic MS data manipulation. Once the MS data is opened, a PyApp can be called from a simple pop-up menu and the Host passes the data pointers and associated methods to the PyApp for processing/viewing/reporting. Once the data is passed to the PyApp, the data can viewed and processed using all available tools from Python or the Cerno libraries without being restricted by the Host architecture. While PyApps can be interactive, they can also provide automated report generation, LIMS integration, plus gateways to other applications. The processed data can also be passed back to the Host for further processing, saving, or report generation.

While this method of "App" processing is not as well integrated into the main software application as a more traditional "Macro" language, it is far more flexible and robust. Since the PyApp runs independently of the Host, it cannot unduly impact the Host with unexpected errors or crashes. In addition, the PyApp is not dependent on the more limited library of functionality of the Host and can take advantage of the huge number of scientific libraries available to the Python community. Finally, since the PyApp processes the data outside the Host, it does not impact any regulatory or compliance issues which may be required of the Host.

To illustrate the power of this approach, a number of PyApps were created. In one example (not shown), the PyApp directly calls the NIST search API to perform EI or MS/MS searches against commercial libraries, collates the results, and produces reports. Figure 2 illustrates a PyApp that performs deconvolution of coeluting peaks for either GC or LC/MS data. The resulting pure component spectra can be passed back to MassWorks for further analysis. The deconvolution is an advanced two-step algorithm[2] that is part of the Cerno library while the graphics and UI are Python.

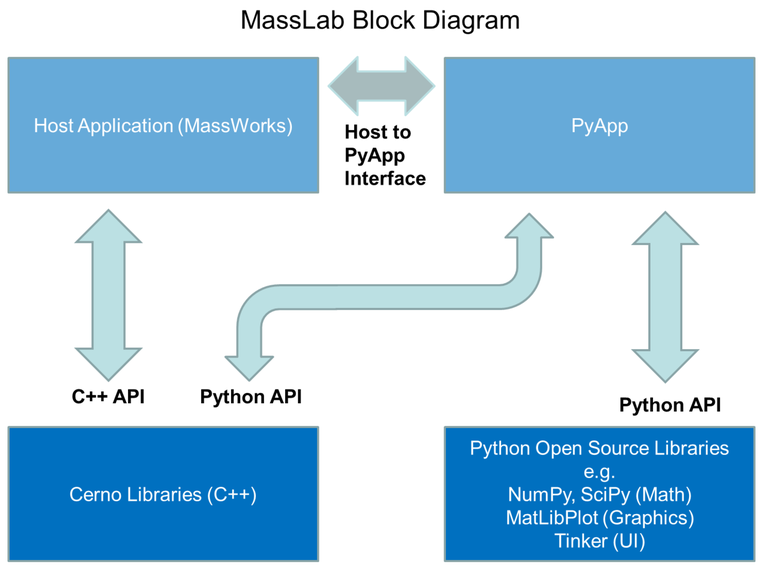

Figure 1. MassLab block diagram illustrates the dual C++/Python API which allows both the MassWorks app and PyApp's to access Cerno libraries. The PyApp (called from MassWorks) can also utilize the rich Python libraries.

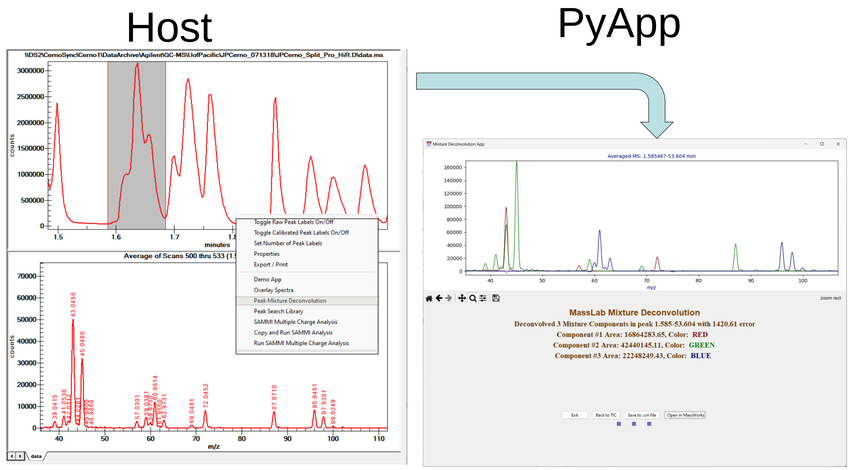

Figure 2. Chromatographic peak deconvolution is easily added to the host software using a deconvolution PyApp. Simply select the GC or LC/MS peak and choose the PyApp to launch from the right click menu. Interactive deconvolution shows the end results and the pure component spectra.

It is also possible to use PyApps to automate processing in the Host (MassWorks). For example, MassWorks provides powerful calibration tools to enable formula ID on unit resolution mass spectrometers, but it is generally a manual peak by peak procedure. Since PyApps have access to the same computational libraries, the automated identification, processing, and reporting could be performed for all peaks GC or LC run.

In another example, a completely new approach to large molecule multiply charge deconvolution was implemented which leverages the extensive NumPy and SciPy libraries (Figure 3). This type of advanced processing would essentially be impossible in conventional "Macro" based approaches or would at least be intolerably slow.

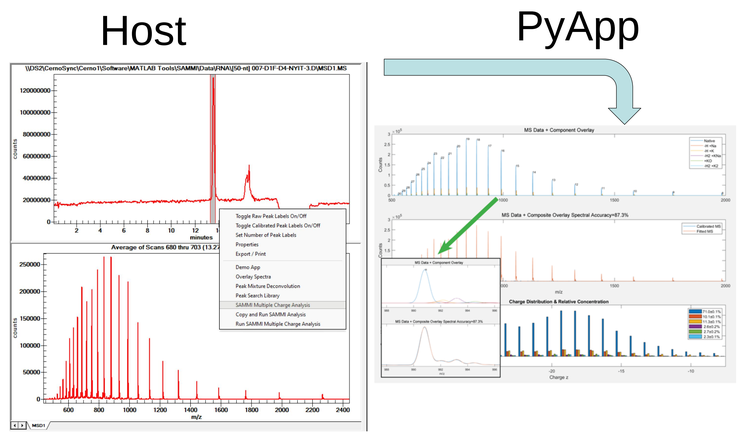

Figure 3. PyApp's can be extremely powerful processing applications, not just simple macros. In this example, an entirely new method for the deconvoluton of multiply charged large molecules was developed in Python (SAMMI[3]) and is added as a powerful extension to MassWorks.

Conclusion

The MassLab system of integrating "PyApps" into the existing Host software provides a powerful platform not only for automation and report generation, but also for adding completely new applications. The dual interface of the Cerno libraries enables the PyApp access to the full power of the underlying Cerno computational engine. The PyApp also has access to additional Python libraries to provide the "Heavy Lifting" required for computationally intensive apps, as well as powerful graphics and UI capabilities. The simple Host-to-PyApp interface allows calling PyApps to process existing data and allows the processed data to be passed back to the Host for archiving or further processing.

References

1. Python Official Website: https://www.python.org/
2. Y. Wang and S. Simonoff. ASMS 2018 TP816.
3. Y. Wang and S. Simonoff. ASMS 2023 TP524.