Applications of Jupyter notebooks in METplus

## Training

Examples of using the METcalcpy and METdatadb tools/libraries

* readily import the modules of interest and start writing code to experiment on how to use

Sharing code between organizations:

Already incorporated code snippets into METplus from jupyter notebook provided by Zach Lawrence NOAA/PSL for zonal and meridional means plotting

Commit jupyter .ipynb files (small, text files in markup language) to Github repository, where it can be rendered (but not run/not interactive)

\*NOTE:\* Not as useful for scripts, such as the code in the METplus wrappers and METplotpy repositories. since the code in those repositories were designed to be run from command line. However, there are some modules in each repository which can be imported and used by other code. But the main purpose of these repositories is to provide scripts that can be run directly.

Explanation of terms:

===============

Scripts:

Python file intended to be run directly

Module: Python file intended to be imported into scripts or other modules

Package: collection of related modules that work together to provide functionality

Library: term used to indicate a bundle of code to provide a wide range of functionality.

example: Matplotlib is a plotting library and used to generate plots. Likewise with Unidata METpy which, from its “About” section on GitHub, is a “…collection of tools written in Python for reading visualizing and performing calculations with weather data”.

**Simply sharing the jupyter notebook:**

Commit .ipynb file to public Github repository. The notebook can be rendered but is not interactive (user cannot run any cells or make any modifications).

**Minimal set up for users:**

Docker containers of jupyter notebooks so users don’t need to recreate Python working environments. Disadvantage: need to have Docker client and instruct users to run a few docker command line instructions. Docker client must be installed as root and licensing is now required.

Work around is to utilize podman, which lends itself better to HPC environments and has minor modifications to use existing docker images.

**Minimal set up for users and host:**

Github has Binder, where you point to the location of your jupyter notebook file in your Github repository, along with a requirements.txt file to indicate which Python packages are required. Behind the scenes, a docker container is created, allowing the user to simply open up a browser, click on the Launch binder badge in the README portion of the jupyter notebook’s GitHub repository. And the user can interact by running cells, etc. Downsides, often issues with pip installation of some packages, some of which are beyond the control of the owner of the jupyter notebook.