**Making Python modules Accessible to Python Command Line Interpreter and Jupyter:**

From a command prompt, you can temporarily (just for the current session) access a python module you created using the following commands:

import sys

sys.path.append(r'C:\code\my-library-folder')

import my-python-module as myFun

This also works in a Jupyter notebook cell. The r at the start of “r'C:\code\my-library-folder'” brings it in as raw text without escaping the ‘\’.

note that in the above:

* your module might be named “my-python-module.py” but you drop the “.py”
* this syntax only works for the current python session on the command prompt or console
* you can use any syntax of the import statement you want using your module name (without the “.py”)
* In example above, “as myFun” renames this to something shorter for convenience. Now we can access functions from the module as in: “myFun.first\_function\_in\_module()”

**Making it Accessible All the Time:**

For command line interpreters:

So how do we make it accessible all the time without having to put in the SYS path statement?

On Windows, you add or modify the user environment variable: %PYTHONPATH%

On my Win7 64 bit machine, this is done with:

Start Menu 🡪 Control Panel 🡪 System and Security 🡪 System 🡪 Advanced System Settings 🡪

[Environment Variables] button 🡪 User Variables

Then if the “PYTHONPATH” variable is in the window, select it and use [edit] button to change it

Otherwise use [new] button to create it. Name = “PYTHONPATH”, Value = <your path>

For more than one path, separate each one with a “;”

For Jupyter:

We need to add a line to the Jupyter profile. There is a link to a detailed write-up under references. Walking through this content, the following was tested on my Win7 machine:

ipython profile create 🡺 creates the profile file

ipython profile locate 🡪 tells you location of .ipython folder

Then in ipython folder, the file is created in <ipython folder>\default\_profile

Then, uncomment out this line in the file: ipython\_config.py

c.InteractiveShellApp.exec\_lines = []

edit this line so it reads like this (substituting the desired file path into it):

c.InteractiveShellApp.exec\_lines = [import sys; sys.path.append(r”<your-path-here>”)’]

For more than one path, repeat the ‘sys.path.append …’ line and separate each new line from the previous with “;”

**References:**

Creating simple Python modules: <https://www.ibiblio.org/g2swap/byteofpython/read/making-modules.html>

Making them accessible to Python command line interpreter: <http://stackoverflow.com/questions/7472436/add-a-directory-to-python-sys-path-so-that-its-included-each-time-i-use-python>

Making them accessible to Jupyter / iPython (permanently):

<https://www.lucypark.kr/blog/2013/02/10/when-python-imports-and-ipython-does-not/>