Setting Up a Project with Virtual Environment, PyBuilder, and PyCharm

## Abstract

Our goal in this article is to setup a tool set that builds component “libraries” which can be deployed to the *Databricks* version of PySpark. We introduce *PyBuilder*, a Maven-like Python open source build tool, which should work well for Java programmers building Python components. We also use *PyCharm* community edition, a popular free Python IDE.

This article is a part of a series of articles discussing Python dependency management practices for the Java programmer (see article: <http://www.tbd.com>.) A Windows 10 development environment is used for the examples, but nix environments are well documented and almost identical.

This article includes creating a virtual environment with utility ***venv***, and augments the project using *PyBuilder*. Finally, we wrap the new project with the *PyCharm* Python IDE. Documentation resources are included in the reference section at the end of the\is article.

. We cover these steps of project creation:

* Create a Python project directory ***myapppy***.
* Create a Virtual Environment in ***myapppy*** (for Python interpreter-based dependency isolation.)
* Create a PyBuilder project for ***myapppy***.
* Create a PyCharm Community Edition IDE project for ***myapppy***.
* Add Python Source and Test files for sample deployed components.
* Create deployable components using *PyBuilder*.
* Deploy the newly created components locally and verify.
* Deploy the newly created components to the Databricks Community Edition and verify.

## Create the *myapppy* Project Directory

Our first step is to create a package directory for our test project, that we will name ***myapppy.*** We create a *Virtual Environment* for our ***myapppy*** project as well:

D:\\Dependencies\myapppy>***python -m venv venv***

D:\\Dependencies\myapppy>***tree venv***

D:\\DEPENDENCIES\MYAPPPY\VENV *(abbreviated content)*

├───Include

├───Lib

│ └───site-packages

│ ├───pip

│ ├───pip-19.2.3.dist-info

│ ├───pkg\_resources

│ ├───setuptools

│ ├───setuptools-41.2.0.dist-info

└───Scripts

Next, we setup a *PyBuilder* instance for our project by using the script ***loadPyBuilder.cmd***. After running the load script from directory ***myapppy***, the results are:

Directory of D:\\Dependencies\myapppy\venv\Scripts

10/21/2019 04:53 PM <DIR> .

10/21/2019 04:53 PM <DIR> ..

10/21/2019 04:48 PM 2,345 activate

10/21/2019 04:48 PM 1,022 activate.bat

10/21/2019 04:48 PM 1,553 Activate.ps1

10/21/2019 04:48 PM 368 deactivate.bat

10/21/2019 04:48 PM 98,235 easy\_install-3.7.exe

10/21/2019 04:48 PM 98,235 easy\_install.exe

10/21/2019 04:53 PM 103,342 pip.exe

10/21/2019 04:53 PM 103,342 pip3.7.exe

10/21/2019 04:53 PM 103,342 pip3.exe

10/21/2019 04:48 PM 886 pyb

10/21/2019 04:48 PM 98,217 pyb\_.exe

10/21/2019 04:48 PM 98,210 pytail.exe

10/21/2019 04:47 PM 522,768 python.exe

10/21/2019 04:47 PM 522,256 pythonw.exe

10/21/2019 04:48 PM 98,213 wheel.exe

We are now ready to install the *PyBuilder* dependencies, but first we add a ***builder.py*** bootstrap file obtained from the *PyBuilder* project’s GitHub repository (see references.) This special build file “bootstraps” the ***PyBuilder*** installation. We are now able to create a *PyBuilder* environment for our project using steps recorded in file ***loadPyBuilder.cmd***:

D:\\Dependencies\myapppy>***installDependenciesPyBuilder.cmd > installDependenciesPyBuilder.log***

***PyBuilder*** now has dependencies and has added a utility (***pygmentize.exe***).

We now create our directories and basic PyBuilder infrastructure:

D:\\Dependencies\myapppy>***del build.py***

D:\\Dependencies\myapppy>***venv\Scripts\activate.bat***

(venv) \d:\\dev-topics-dependencies\Dependencies\myapppy>pyb\_ --start-project

Project name (default: 'myapppy') :

Source directory (default: 'src/main/python') :

Docs directory (default: 'docs') :

Unittest directory (default: 'src/unittest/python') :

Scripts directory (default: 'src/main/scripts') :

Use plugin python.flake8 (Y/n)? (default: 'y') :

Use plugin python.coverage (Y/n)? (default: 'y') :

Use plugin python.distutils (Y/n)? (default: 'y') :

Created 'setup.py'.

This initial run of ***PyBuilder*** creates the inital ***setup.py*** and ***build.py*** files, along with the ***src, target*** and ***docs*** directories. The newly created ***build.py*** should look something like this:

from pybuilder.core import use\_plugin, init

use\_plugin("python.core")

use\_plugin("python.unittest")

use\_plugin("python.install\_dependencies")

use\_plugin("python.flake8")

use\_plugin("python.coverage")

use\_plugin("python.distutils")

name = "myapppy"

default\_task = "publish"

@init

def set\_properties(project):

pass

We can do a ***PyBuilder*** “verify” (Maven “test”) run on the no-source file environment, and we get something like this:

(venv) D:\\Dependencies\myapppy>***pyb\_ verify***

PyBuilder version 0.12.0.dev20190116131423

Build started at 2019-10-21 17:24:08

------------------------------------------------------------

[INFO] Building myapppy version 1.0.dev0

[INFO] Executing build in \D:\\Dependencies\myapppy

[INFO] Going to execute task verify

Package(s) not found: coverage, flake8, pypandoc, twine, unittest-xml-reporting

[INFO] Installing plugin dependency coverage

[INFO] Installing plugin dependency flake8

[INFO] Installing plugin dependency pypandoc

[INFO] Installing plugin dependency twine

[INFO] Installing plugin dependency unittest-xml-reporting

[INFO] Running unit tests

[WARN] Not forking for <function do\_run\_tests at 0x000002B29AF87948> due to Windows incompatibilities (see #184). Measurements (coverage, etc.) might be biased.

[INFO] Executing unit tests from Python modules in \D:\\dependencies\myapppy\src\unittest\python

[WARN] No unit tests executed.

[INFO] All unit tests passed.

[INFO] Building distribution in \D:\\dependencies\myapppy\target\dist\myapppy-1.0.dev0

[INFO] Copying scripts to \D:\\dependencies\myapppy\target\dist\myapppy-1.0.dev0\scripts

[INFO] Writing setup.py as \D:\\dependencies\myapppy\target\dist\myapppy-1.0.dev0\setup.py

[INFO] Collecting coverage information

[WARN] coverage\_branch\_threshold\_warn is 0 and branch coverage will not be checked

[WARN] coverage\_branch\_partial\_threshold\_warn is 0 and partial branch coverage will not be checked

[WARN] Not forking for <function do\_coverage at 0x000002B29AFAF438> due to Windows incompatibilities (see #184). Measurements (coverage, etc.) might be biased.

[INFO] Running unit tests

[INFO] Executing unit tests from Python modules in \D:\\dependencies\myapppy\src\unittest\python

[WARN] No unit tests executed.

[INFO] All unit tests passed.

Coverage.py warning: No data was collected. (no-data-collected)

[INFO] Overall coverage is 100%

[INFO] Overall coverage branch coverage is 100%

[INFO] Overall coverage partial branch coverage is 100%

------------------------------------------------------------

BUILD FAILED - No data to report.

------------------------------------------------------------

Build finished at 2019-10-21 17:24:30

Build took 21 seconds (21780 ms)

We have created this directory structure for *PyBuilder*:

D:\\Dependencies\myapppy>dir & tree docs & tree src & tree target

Directory of D:\\Dependencies\myapppy

10/22/2019 02:45 PM <DIR> .

10/22/2019 02:45 PM <DIR> ..

10/21/2019 05:17 PM 339 build.py

10/21/2019 05:17 PM <DIR> docs

10/21/2019 05:07 PM 1,394 installDependenciesPyBuilder.cmd

10/21/2019 05:09 PM 2,176 installDependenciesPyBuilder.log

10/21/2019 04:42 PM 1,057 loadPyBuilder.cmd

10/21/2019 04:53 PM 83,384 loadPyBuilder.log

10/21/2019 05:17 PM 2,527 setup.py

10/21/2019 05:17 PM <DIR> src

10/21/2019 05:24 PM <DIR> target

10/21/2019 04:48 PM <DIR> venv

D:\\DEPENDENCIES\MYAPPPY\SRC

├───main

│ ├───python

│ └───scripts

└───unittest

└───python

D:\\DEPENDENCIES\MYAPPPY\TARGET

├───dist

│ └───myapppy-1.0.dev0

│ └───scripts

├───logs

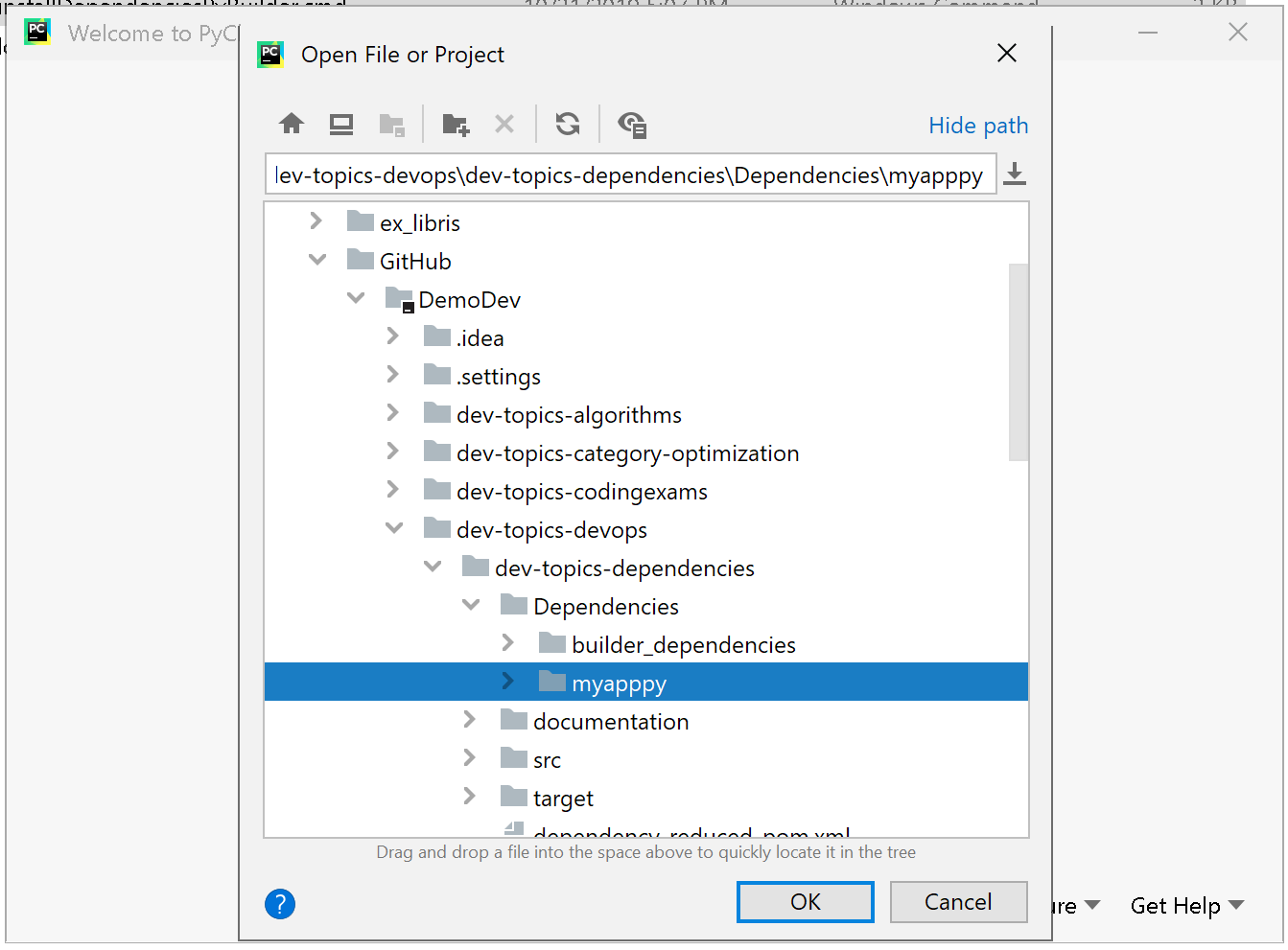
│ └───install\_dependencies

└───reports

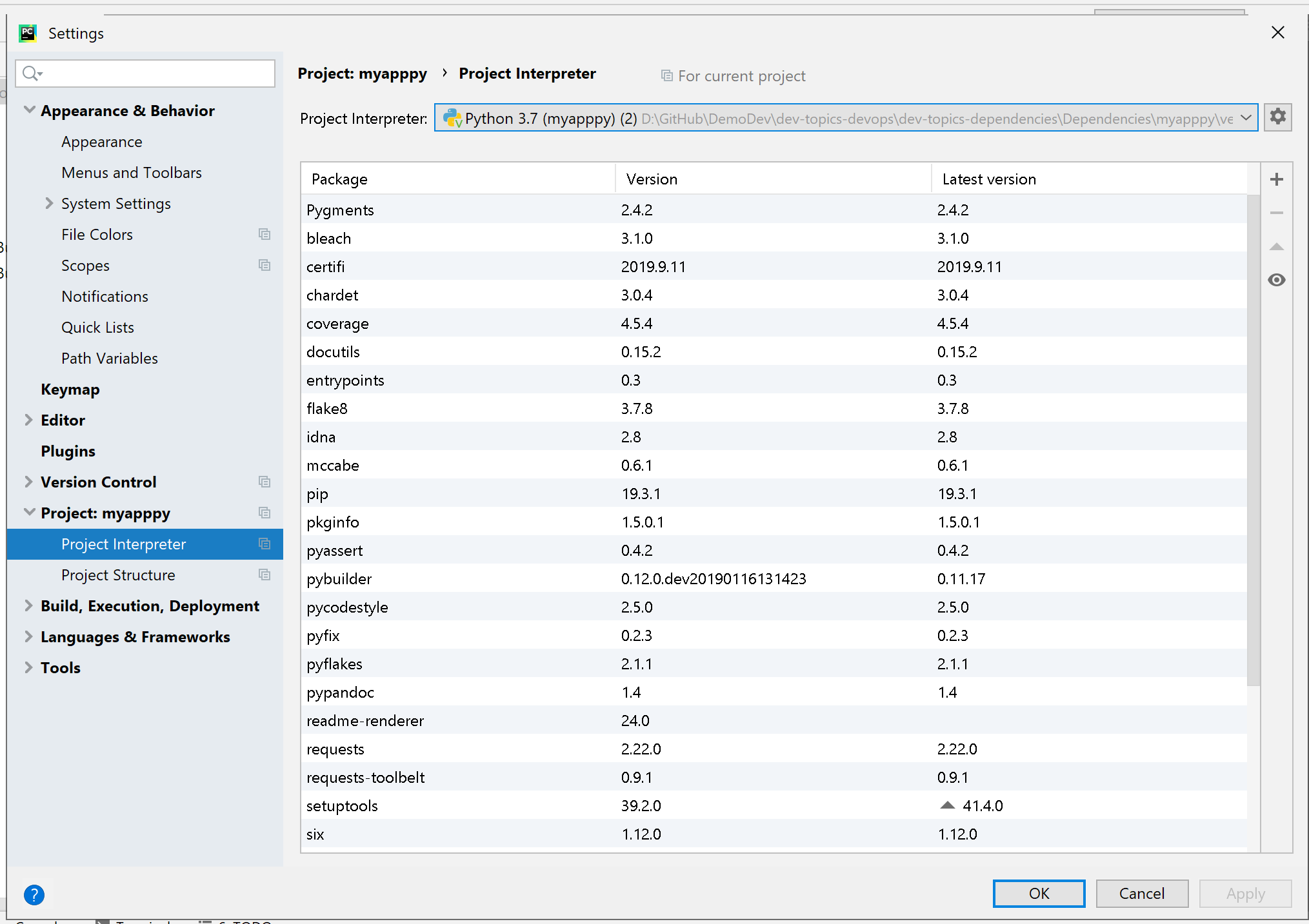
## Create the *PyBuilder* Project

Create a *PyCharm Community Edition* Project over the *PyBuilder* Structure using the IDE.

1. Open the IDE
2. Open the myapppy project (the myapppy directory)

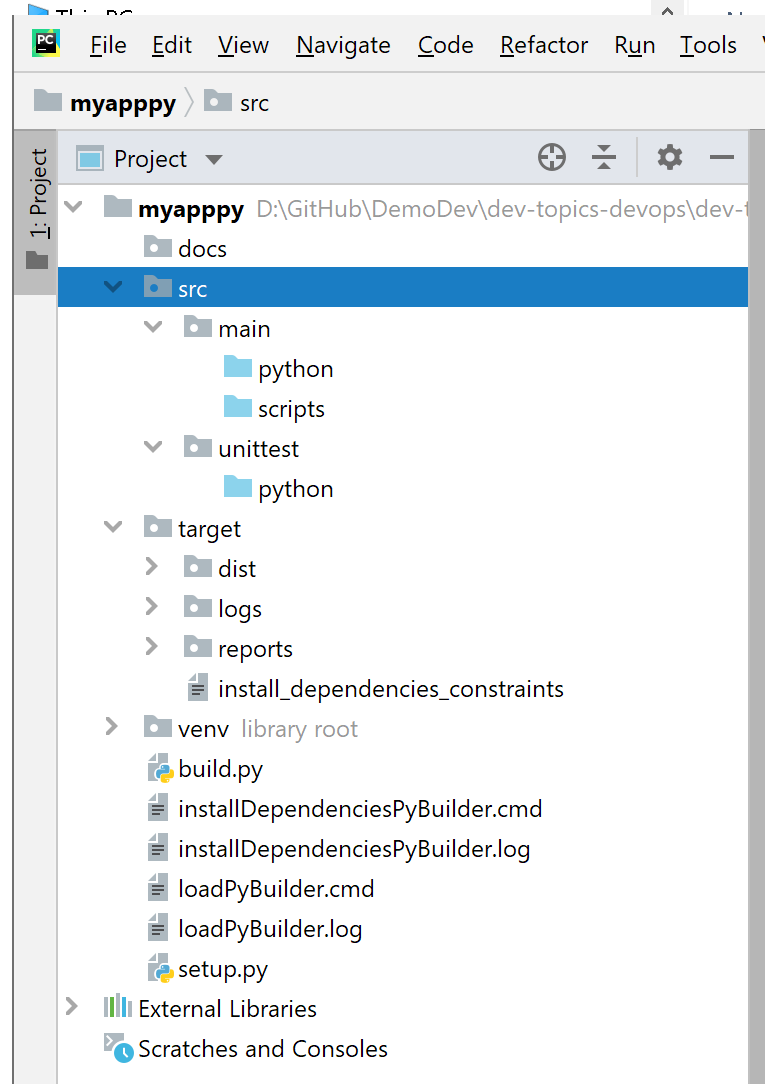


1. Select the virtual environment to associated with the project (***File>Settings>Project Interpreter>Show All>****{select venv}*)



1. Mark source code directories as “source root” (***highlight>right click>Mark as Sources Root***)

The required directories are show in blue in the diagram below. The source directories are ***src\main\python***, ***src\main\scripts***, ***unittest\python***.



Now synchronize, delete compiled Python files, and close the project.

## Add Python Source and Test files for sample deployed components

We can now add source files for functionality and unit tests.

# References - Resources

1. Virtualenv – used to create a controlled Python runtime environment: <https://pypi.python.org/pypi/virtualenv>).
2. Additional Virtualenv documentation: <https://virtualenv.pypa.io/en/latest/>).
3. Venv background: <https://realpython.com/python-virtual-environments-a-primer/>.

## PyBuilder Documentation

1. PyBuilder Documentation Home: <http://pybuilder.github.io/>.
2. PyBuilder GitHub repository: <https://github.com/pybuilder/pybuilder>.
3. PyBuilder ***build.py*** link in GitHub: <https://github.com/pybuilder/pybuilder/blob/master/build.py>.
4. PyBuilder tutorial (top level): <https://pybuilder.readthedocs.io/en/latest/walkthrough-new.html>.
5. PyBuilder tutorials: <http://pybuilder.github.io/documentation/tutorial.html#.XaJXGkZKiUk>.
6. PyBuilder PDF: <https://buildmedia.readthedocs.org/media/pdf/pybuilder/stable/pybuilder.pdf>.