File Archiving Tool

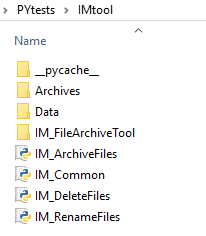
**Purpose:** the purpose of these pythons is multifold:

1. They are being utilized with a cloud-based data-import utility which requires a constant file name. If source-systems provide data-files with dynamic names (i.e. with date suffixes), then these must first be converted to a static name.
2. To archive data-files for reuse at later date
3. To delete expired archived files

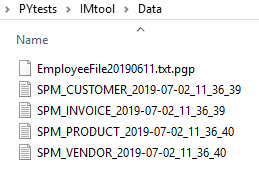
**Process:**

1. Rename data-files (to achieve a constant file name)
2. Decrypt data-files (done by a system process, no python required)
3. Import data-files (done by a system process, no python required)
4. Encrypt data-files (done by a system process, no python required)
5. Archive data-files (move to Archive directory, with same name as step1)
6. Remove expired archived data-files

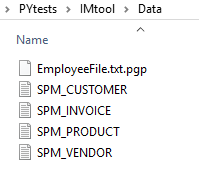
**Setup**:



**STEP01- Before IM\_RenameFiles is run:**



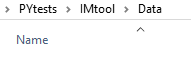
**STEP02- After IM\_RenameFiles.py is run. This looks good.**

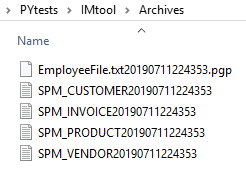


**STEP03- After IM\_ArchiveFiles.py is run. The files are correctly relocated.**

**But two problems arise: 1) my mechanism for applying current date suffix is not good. These files should actually take on the name before they were initially renamed. Consider the following use-case. The client wishes to rerun may-5 files. If they move those from the archive to the data-folder, then my mechanism will now convert them to Jul-11 files, which is not correct. Therefore, my current mechanism is not right. It should archive the files as the same name it came in as.**

**2) look at the first file, the date suffix went between .txt and .pgp. This will not be a problem if we rename the file back to its original name.**





These files should be archived with the same name as in Step01

**STEP04- After IM\_DeleteFiles.py is run:**

