DATA_PROCESSOR DOCUMENTATION

DESCRIPTION [PLEASE READ BEFORE USE!]:

- The scripts in the **Data_Processor** allow you to convert files containing the same kind of data which come in different formats into one consistent format, which may be provided as an excel file or as a csv file (**THE SCRIPT WILL ONLY WORK WITH .CSV .XLSX AND .XSL FILE FORMATS**)
- All code in this project is written in **Python 2.7**
- To use the scripts, you would need to have a template file for which the data contained in the files which have different formats will be moved into. [**NOTE THAT THIS TEMPLATE FILE MUST BE OF EXCEL FORMAT.**]
- The scripts also provide a feature referred to as "***NORMALIZING***". This feature works on two types of data namely "**TIME AND DATE**" data and "**LONGITUDE AND LATITUDE**" data.
- **NORMALIZING** basically means the scripts can convert the fields to be normalized into different formats. For example: **TIME AND DATE** fields can be split into separate fields (**date, time, day, hours, minutes, unix_timestamp, am/pm**) and can be merged into one field (**datetime**). "**LONGITUDE AND LATITUDE**" data can be converted into **decimal degrees** or into **degrees minutes seconds** and the **north/south** and/or **east/west** fields can be filled in based on the values provided from the original file as well. (eg. If you have a **date** and a **time** column, the scripts can generate the **datetime** value and the **unix_timestamp** and fill it into the template, and so on…
- The "**NORMALIZATION**" **feature has a lot of limitations, which can be found in the LIMITATIONS section of this document.**
- The use of these scripts have a number of limitations which will be specified in the **LIMITATIONS** section and also in the **IMPORTANT NOTES** section which are at the end of this documentation.

HOW TO USE:

Before Use, it is recommended that you see the IMPORTANT NOTES section at the end of this document.

To use the scripts you would need to run them from the command line or using your desired IDE. I suggest downloading the community version of PyCharm and using that for the purpose of running these scripts. The following link is a video on how to setup PyCharm and run your scripts with it https://www.youtube.com/watch?v=JLfd9LOdu_U
You can use the scripts by running one of the following:

- **CommandLine_Handler.py**
- **Main.py**
- **Main_DataLoader.py**
- **Main_Normalizer.py**

CommandLine_Handler.py:

- **This script will need you to supply at least command-line 2 arguments;**
  - **The first argument** needs to be the full file path to the input data. It could be the path to a file which you wish to load or a folder containing multiple files which you wish to convert into the format of the template which you have created.
  - **The second argument** needs to be the full file path to the template file which the input will be loaded into. **This must be a single file and must me an excel file ('.xls' or '.xlsx' frmat).**

  NOTE: The next 2 arguments are optional

  - **The third argument** needs to be the full file path to the .JSON map file for the input file(s) which you are loading in the template
  - **The fourth argument** needs to be a number which represents what row in the template file contains the column headers (**NOTE: IF THIS IS NOT SUPPLIED IT DEFAULTS TO 1**)
- This script will copy the data from an input file and load it into a copy of the supplied template file. It will ask the user questions to know where the fields which are in the template are located in the input file(s).
- If a directory is provided as the input path argument, it will ask the user to confirm if all the files are of the same format or not.
- If all the input files are of the same format, it will only question the user on where to find the data once, else it will question the user on where the find the values for each file.

- The script will also **normalize** the data. **The normalization feature has a lot of limitations, which can be found in the LIMITATIONS section of this document**
- This script will make a directory (folder) called '**outputs**' in the directory where the template is located. This is where the files generated from running the scripts will be stored.
- Inside the '**outputs**' directory, there will be two directories ('**loaded**' and '**Normalized**').

    - ✓ The **loaded** directory will contain filled versions of the template, for each file that was data was gotten from with the naming convention ('***input filename***'-***'template filename'***.'***template-file-extension'***).
    - ✓ The **Normalized** directory will contain normalized versions of the template, for each file that data was gotten from with the naming convention ***(_normalized***-'***input filename'***-***'template filename'*****.csv**)

NOTE: When running this script it will ask you to supply the format for time, datetime and date fields, if this is not known by the user, it can be skipped by the user and the script will infer the format which it does correctly at least 95% of the time.

Main.py:

- **This script will need you to supply at least 2 fields to the Loader_Config.ini file;**

    (This is an alternative if you wish to avoid using command line arguments.)

    - **INPUT_FILE_PATH.** This field is in the **[INPUT_INFO]** section of the **Loader_Config.ini file,** and here you would need to supply the path to the file containing the data which you would like to load into the template **(simply copy the path to the file you want to load and paste it right after the '=' sign.)**
    - **TEMPLATE_PATH**. This field is in the **[TEMPLATE_INFO**] section of the **Loader_Config.ini file,** and here you need the supply the path to the Template which you wish to transfer the data from the input file into **(simply copy the path to the file you want to load and paste it right after the '=' sign.)**

IMPORTANT NOTE: Additionally you may supply the INPUT_FILE_MAP, the TEMPLATE_MAP_PATH and/or the TEMPLATE_HEADER_ROW if need be. [If the TEMPLATE_HEADER_ROW is not 'one' (1) it is necessary that you change it to the right number, but for the other two fields, the script will generate them for you if they do not exist already]

- This script will copy the data from an input file and load it into the supplied path for a template file. It will ask the user questions to know where the fields which are in the template are located in the input file.

- The script will also **normalize** the data. **The normalization feature has a lot of limitations, which can be found in the LIMITATIONS section of this document**
- This script will make a directory (folder) called ‘**outputs’** in the directory where the template is located. This is where the files generated from running the scripts will be stored.
- Inside the ‘**outputs**’ directory, there will be two directories (‘**loaded’** and ‘**Normalized**’).

    - ✓ The **loaded** directory will contain filled versions of the template, for each file that was data was gotten from with the naming convention (‘***input filename’***-***‘template filename’***.’***template-file-extension’***).
    - ✓ The **Normalized** directory will contain normalized versions of the template, for each file that data was gotten from with the naming convention ***(_normalized***-‘***input filename’***-***‘template filename’***.**csv**)

NOTE: When running this script it will ask you to supply the format for time, datetime and date fields, if this is not known by the user, it can be skipped by the user and the script will infer the format which it does correctly at least 95% of the time.

Main_DataLoader.py:

- **This script will need you to supply at least command-line 2 arguments;**
    - **The first argument** needs to be the full file path to the input data. It could be the path to a file which you wish to load or a folder containing multiple files which you wish to convert into the format of the template which you have created.
    - **The second argument** needs to be the full file path to the template file which the input will be loaded into.

    NOTE: The next 2 arguments are optional

    - **The third argument** needs to be the full file path to the .JSON map file for the input file(s) which you are loading in the template
    - **The fourth argument** needs to be a number which represents what row in the template file contains the column headers (**NOTE: IF THIS IS NOT SUPPLIED IT DEFAULTS TO 1**)
- This script will copy the data from an input file and load it into the supplied path for a template file. It will ask the user questions to know where the fields which are in the template are located in the input file(s).
- It will ask the user to confirm if all the files are of the same format or not.
- If all the input files are of the same format, it will only question the user on where to find the data once, else it will question the user on where the find the values for each file.

- This script will make a directory (folder) called ‘**outputs’** in the location of the template to which the data is being loaded to within that folder it will make a directory called ‘**loaded’**.

    - ✓ The **loaded** directory will contain filled versions of the template, for each file that was data was gotten from with the naming convention (‘***input filename’***-***‘template filename’***.’***template-file-extension’***).

NOTE: When running this script it will ask you to supply the format for time, datetime and date fields, if this is not known by the user, it can be skipped by the user and the script will infer the format which it does correctly at least 95% of the time

Main_Normalizer.py:

- **This script will need you to supply at least command-line 2 arguments;**
    - **The first argument** needs to be the full file path to the input data. It could be the path to a file which you wish to load or a folder containing multiple files which you wish to convert into the format of the template which you have created.
    - **The second argument** needs to be the row number that contains the header names in the file.
- This script will **NORMALIZE** the data from an input file and generate a normalized version of the file.
- It will ask the user to confirm if all the files are of the same format or not.
- If all the input files are of the same format, it will only question the user on where to find the data once, else it will question the user on where the find the values for each file.
- This script will make a directory (folder) called ‘**outputs’** in the location of the template to which the data is being loaded to within that folder it will make a directory called ‘**normalized’**.
    - ✓ The **normalized** directory will contain filled versions of the template, for each file that was data was gotten from with the naming convention (‘***input filename’***-***‘template filename’***.csv)

For more technical details on how this tool works. “in-code” documentation is provided as well as Markdown files describing the JSON files generated when running this tool.

LIMITATIONS:

- When loading data from a file to a template if you want the **date and/or time** fields to be treated as actual date or time values (**and formatted as such**), you would need to follow the naming conventions stated below in the for the scripts to work.
- The **NORMALIZING (and some of the data loading)** feature will only work if you follow certain naming conventions for the **date and time** fields and the **longitude and latitude** fields.
    - If you have a **datetime** field, it **MUST** be named **'datetime'**
    - If you have a **date** field, it must be named '**date**'
    - If you have a **time** field, it must be named **'time'**
    - If you have an **hour** field, it must be named **'hour'***
    - If you have a **minutes** field, it must be named **'minutes'***
    - If you have a **seconds** field, it must be named **'seconds'***
    - If you have a **day** field, it must be named **'day'***
    - If you have a **month** field, it must be named **'month'***
    - If you have a **year** field, it must be named '**year**'*
    - If you have a **longitude** or **latitude** value in **decimal degrees format**, it must be named **longitude_decdeg** or **latitude_decdeg**
    - If you have a **longitude** or **latitude** value in **degrees minutes seconds format**, it must be named **longitude_minsec** or **latitude_minsec [Remember that this must be formatted right that is ##°##'##" or ##°##' where ## represents a decimal number or an integer]**
    - If you have columns containing the **cardinal points** for the **longitude and/or latitude values**, the column header for the latitude cardinal point must be **N/S** and the column header for longitude cardinal points must be **E/W**

IMPORTANT NOTES:

FOR THE ABOVE FIELDS WITH LIMITED NAMESPACE:

- **THE CASING OF CHOICE DOES NOT HAVE TO BE THE SAME (eg longitude_decdeg could be Longitude_DEcDEg)**
- **THE UNDERSCORES (_) MAY BE REPLACED WITH SPACES (eg. longitude_minsec could be longitude minsec)**
- **The column name definitions which have * top right of them must only contain numbers (eg. the month field must only have numbers in the data field)**