User Guide

# About

The purpose of this app is to collect data from an input Excel sheet (`input.xlsx`), and copy the information into a template Excel sheet (`template.xlsx`) using different column names, which are mapped in a configuration JSON file (`col\_config.json`), and save it as a new file (`output.xlsx`).

# Requirements

These libraries are required: `pandas==2.0.3` and `openpyxl==3.0.10`.

If you are using Conda, you can use the `environment.yml` file, which will set the name of the environment to `pks\_env`. (See [here](https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#creating-an-environment-from-an-environment-yml-file) for more information.) Create the environment with `conda env create -f environment.yml`, activate the new environment with `conda activate pks\_env`, and then run the Python file, following this format: `python pks.py input.xlsx template.xlsx output.xlsx col\_config.json`. To install Conda, see [here](https://conda.io/projects/conda/en/latest/user-guide/install/index.html).

# Definitions

`input.xlsx`: The input workbook that you want to take information from. This file’s data must be on a single sheet.

`template.xlsx`: The template file of the desired output format. This file must be a single sheet.

`output.xlsx`: The output filename that you want to use. Retaining the extension is recommended.

`col\_config.json`: The file that contains the column mappings from input to template. For more information, refer to the section [Column Configuration JSON](#_Column_Configuration_JSON).

# Usage Instructions

## Quick Overview

For all methods, make sure the input, template, and column configuration files are all in the same directory as `pks.py`. Otherwise, you must include the file path in the file string as well. (For example, “C:\Users\PKS\My Template.xlsx” instead of just “C:\Users\PKS\My Template.xlsx”.)

### Method 1: Command Line

You can use either the full input style or a simple input style.

|  |  |
| --- | --- |
| Full Input Form | Simple Input Form |
| `python pks.py input.xlsx template.xlsx output.xlsx col\_config.json` | `python pks.py` |

As you can see, the difference between the two modes is in the number of arguments.

Full Input:

Input: `python pks.py input.xlsx template.xlsx output.xlsx col\_config.json`

Example: `python pks.py "PKS Sample.xlsx" "New Firm Onboarding BAP Tracker Template.xlsx" new\_output.xlsx col\_config.json`

Spaces count as moving on to the next input element, so if there are spaces in your file names, enclose the entire file name in quotation marks.

Simple Input:

This will run in the “default file names” version. See the [Modifying File Names for Default File Names Version](#_Modifying_File_Names) section below for changing the names or paths for the files.

### Method 2: Visual Studio Code

Open `pks.py` in Visual Studio Code and run the program.

Running it simply as is will use the “default file names” version. See the [Modifying File Names for Default File Names Version](#_Modifying_File_Names) section below for changing the names or paths for the files.

You can change the command line arguments in Visual Studio code to have a similar functionality to the full input command line version, but I will not be teaching you that here.

### Modifying File Names for Default File Names Version

To change the file names or file paths, modify the strings in the function `get\_default\_files()` in `pks.py`, which starts on line 10.

A screenshot of a computer program

Description automatically generated  
Change *only* the file/directory path names, in yellow box above.

For example, if you wanted to change the input Excel file, you might change `"PKS Sample.xlsx"` to `"Actual PKS File.xlsx"`.

Regardless of whether there are whitespaces or not in your file/directory path, you must enclose them in quotation marks. Using single quotes is also acceptable, but you must keep it the same for each variable.

## Column Configuration JSON

Example, from October 2023 request:

|  |
| --- |
| {  "col\_keys": {  "Payout Advisor": "IAR",  "Client: Account Name": [  "Last Name",  "First Name"  ],  "Statement State": "State",  "Account Name": "Source/Carrier",  "Financial Account Number": "Contract / Policy #",  "Product Type": "Product Type",  "Current Account Value": "Most Recent Contract Value"  },  "col\_start": {  "IAR": "A9",  "Last Name": "B9",  "First Name": "C9",  "State": "F9",  "Source/Carrier": "G9",  "Contract / Policy #": "I9",  "Product Type": "J9",  "Most Recent Contract Value": "R9"  }  } |

A green and white diagram

Description automatically generated with medium confidence

If you want to understand JSON’s more, please look it up somewhere else. Basically, the structure used here involves key-value pairs.

The file must contain two keys at the first level: “col\_keys” and “col\_start” *exactly*. The “col\_keys” value must be a dictionary containing header column names from the input Excel that map to column names from the template Excel. The “col\_start” has column names from the template Excel that map to cell locations of the corresponding *header*, written in typical Excel cell coordinate convention, such as “A1”. Do not use any absolute references, as it will only accept cells in the form of one to three characters followed by any number string, without whitespaces. As of October 2023, the maximum number of letters for the column coordinate is three. The package used to write to the Excel file has a limit to the number of rows (about one million), but it is not checked early on, and it may terminate the program if you reach this number of rows or it may result in unexpected results. Avoid using special characters or encodings.

Example template:

|  |
| --- |
| {  "col\_keys": {  "InCol1": "OutCol1",  "InCol2": "OutCol2"  },  "col\_start": {  "OutCol1": "Header1Location",  "OutCol2": "Header2Location"  }  } |

In “col\_keys”, the string to the left of colon should exist as a header in `input.xlsx`, and the string to the right of the colon should exist in the `template.xlsx`. Each side of the colon should contain exactly one string, enclosed by quotation marks. There is one exception to this rule, which is how you control the name splitter.

To use the name splitter, the name column from `input.xlsx` must contain the word “name”, and it must map to a list of exactly two strings, “Last Name” and “First Name”. To indicate a list, use brackets. Furthermore, this can be the only key-value pair that has a value that is a list. Read more about it below in the section [Potential Problems and Related Notes](#_Potential_Problems_and).

## Name-Splitting

There is one input-output column mapping that has a very specific hard-coded implementation: the account name column. As of October 2023, this mapping was provided as `"Client: Account Name": ["Last Name", "First Name"]`.

Rules for name-splitting:

* There must be exactly one input column that maps to a list of exactly two output columns, and there must be exactly one such mapping. You cannot, for example have `"Address": `["Zip", "State"]` in addition to `"Client: Account Name": ["Last Name", "First Name"]` in the same file. In fact, such a mapping will not resolve because there is no function to handle it.
* The two output columns must be in a list format (denoted by brackets).
* The input column name must contain “name”, and the output column names must be exactly “Last Name” and “First Name”.
  + For example, if instead of using the mapping,  
    `"Client: Account Name": ["Last Name", "First Name"]`  
    you could use  
    `"accnt\_name": ["Last Name", "First Name"]`  
    if this is the only mapping that has a list instead of a single output column. The input column *contains* “name”. The output column strings are in a list and are exactly “Last Name” and “First Name”.
    - Other valid examples:  
      `"Client name": ["Last Name", "First Name"]`  
      `"Name": ["First Name", "Last Name"]`
  + Please be aware that the header names will follow standard capitalization conventions, regardless of what you put in the `col\_config.json` file. In other words, only the first letter of every term is capitalized. This is to guarantee that the order of the parser will always put the last element of the inputted full name in the “Last Name” column and the first element in the “First Name” column. Note that this means that the output template file and column configuration file must have the same capitalization scheme for the “Last Name” and “First Name” columns.
  + Please be aware that the names will assume the format of “First <any number of middle> Last”, separating each component of the name by whitespaces, and following standard capitalization conventions.
    - Examples:

|  |  |  |
| --- | --- | --- |
| Input Name | Last Name | First Name |
| First Last | Last | First |
| Susan T Smith | Smith | Susan |
| A B C D | D | A |
| susan tess smith | Smith | Susan |
| Mandy DeVries | Devries | Mandy |
| Firstname Lastname Accidentalterm | Accidentalterm | Firstname |

In short, the rules basically mean, to use the name-splitter, you must have some key with “name” in the word mapping to a list of the form `["Last Name", "First Name"]` or `["First Name", "Last Name"]`, and there can only be one usage of a name-splitter per file.

Empty names in the input Excel sheet will be saved as “None” in both the Last Name and First Name columns of the output Excel.

To make a different name splitter or to change the order/functionality, you must change the source code.

# Troubleshooting

If you receive `JSONDecodeError`, check that your `col\_config.json` file is correctly formatted. For example, you can use this website (<https://jsonlint.com/>) to check your JSON input, or you can search for a different validator, using a search query such as “JSON validator”.

If you receive `UnicodeDecodeError`, check that you passed the correct JSON file, and that the file is correctly formatted.

If you receive `FileNotFoundError`, check the names of all your files.

If you receive `ValueError`, you may have provided a bad location (coordinate) or a bad column name. Furthermore, worksheets cannot contain more than 1,048,576 rows. I don’t know if it can contain 1,048,576 rows either.

Other types of errors may occur, and they usually will have some additional notes that will help you figure out what was found to be incorrect.

|  |  |
| --- | --- |
| Message | Issue |
| The correct number of arguments was not passed. | The command requires exactly zero or exactly four arguments. |
| Given more than one file sharing the same name. | The input command contains an output\_file name that matches another passed file name. |
| Please verify the top level of <col\_config.json> contains two dictionaries, "col\_keys" and "col\_start". | The app could not find the two dictionary keys at the first level of the JSON file.  Refer to the [Column Configuration JSON](#_Column_Configuration_JSON) section above for more details. |
| Please verify the top level of <col\_config.json> contains the keys "col\_keys" and "col\_start". | The app could not find the dictionary keys of names exactly “col\_keys” and “col\_start”.  Refer to the [Column Configuration JSON](#_Column_Configuration_JSON) section above for more details. |
| Please verify col\_key's output columns match col\_start's keys. | The output column names of col\_keys must match the keys of col\_start.  Refer to the [Column Configuration JSON](#_Column_Configuration_JSON) section above for more details. |
| Location string <invalid coordinate> does not follow Excel coordinate convention. | The displayed coordinate provided in the value for some col\_start was found to be invalid. These coordinates must adhere to a letter-digit format, with one to three letters followed by at least one digit, without any spaces. Do not use any absolute indicators (the $ in Excel).  Refer to the [Column Configuration JSON](#_Column_Configuration_JSON) section above for more details. |
| Please verify the columns in the column configuration file exist in the input excel sheet. | The column keys (col\_keys) must exist as header columns in the input Excel sheet. |
| More than one key from col\_key was found to map to a list. | Your `col\_config.json` file contains more than one instance of a key mapping to a list.  Refer to the [Column Configuration JSON](#_Column_Configuration_JSON) section above for more details. |
| Two output columns are required for name splitting. Found <number>. | The number of items in your list of output values for your name column in col\_keys was not equal to two. They must be in one of the two forms: `["Last Name", "First Name"]` or `["First Name", "Last Name"]`.  Refer to the [Name-Splitting](#_Name-Splitting) section above for more details. |
| <Input name column> is not valid for name-splitting. | The key in col\_keys that maps to a list does not contain the word “name” in it.  Refer to the [Name-Splitting](#_Name-Splitting) section above for more details. |
| Name splitting output columns do not match "Last Name" and "First Name". | The output template columns indicated by the name col\_key and list format are not named “Last Name” and “First Name”. The list must be in one of the two forms: `["Last Name", "First Name"]` or `["First Name", "Last Name"]`. |
| <Column name> does not match <worksheet data>. Please check your template file. | The column name does not match the string found in the worksheet using the given cell address. |

# Warnings, Issues, and Considerations

Warnings

* Input/Output Files
  + There are little to no file input/output safety measures. Be careful about overwriting files. Consider backing up all files in a separate location.
* Strict Usability
  + Generic column headers can be mapped flexibly, but input columns need to exist in the input Excel, and template columns need to exist in the template Excel. Incorrectly using the `col\_config.json` may result in undesired effects. The number of template columns must match the number of template locations. The template column names must match the columns indicated by the coordinates in the actual template file.
* Special Characters
  + Do not use fancy character encodings or very special characters. It might be possible to use special characters for the input spreadsheet’s data because these are just being copied, but the `col\_config.json` file and column headers rely on exact string matches. I would suggest avoiding special characters completely because they may encode differently.
* Bad Cell Coordinates
  + The first validation on the cells ensures that the coordinate leniently adheres to typical Excel cell referencing conventions, such as “AA5”. This will only check if there are one to three alphabetical characters in the beginning of the string, followed by at least one digit. There is no check on the digit or combination of alphabetical characters and digits, so an incorrect or impossible coordinate can pass. If this occurs, you may receive a `ValueError`.

Issues or Future Considerations

* The `col\_config.json` file is probably a little overcomplicated. The locations may have been included in “col\_keys” section, so that less rewriting was required.
* Be careful about using large file sizes.
* Using the name-splitting function means that the split header column names must be exactly “Last Name” and “First Name” (case-sensitive). A possible fix to this would be to explicitly return the column names that are meant to be last name and first name column names and save these into variables.
* A mode for real-time user input for file names.
* A mode for working with multiple inputs, one template, and multiple outputs.