Origin Tagging Test

What is this?

At Origin, we take legal template documents called Final Terms Pro Formas and make them machine-readable so that users can automate the drafting of their Final Terms (FTs) on our platform. We do this by converting the Final Terms Pro Forma into a docx Jinja template that we can feed data into to generate Final Terms (FTs). We need help to convert many of these Final Terms Pro Formas into Jinja templates and this is a test to see if we could work together on this.

The Task

The task is to tag up (ie. convert the Final Terms Pro Forma into a full Jinja template) the Pro Forma for an issuer called Munifin. You will do so by using the pro-forma, `Munifin FT Template.docx`. This template contains a sort of skeleton of the FTs where the structure has been given, along with some instructions on filling it in. This is used by lawyers to assist in the drafting and is a good starting point for you. To help you accomplish this, here are the resources we've provided * **Examples of the Munifin FTs that have been drafted** `Munifin Final Terms Examples` : these FTs are the final outputs and you can use them if you're unsure about what the Pro Forma is looking for in different scenarios and to sanity check your work * **Example of tagged Final Terms Pro Formas for other issuers** `Example Jinja Tagged Final Terms` : we've provided copies of full Jinja templates for similar documents. These will be useful as you can lean on them for a lot of the structure and the logic expected * **CSV file of our tag library** `Tag Test/tag_library.csv` : This file contains a list of all the variables (tags) that we've used up to this point. You won't need to use them all in your FT task but this will give you an idea of what tags are avaiable to use * **Test Jsons/docx outputs** `Tag Test/Testcases` : Here we have 3 sets of jsons and their corresponding expected output. Below, you'll find instructions on how you can use these to test your progress * **Jinja documentation**: Here is the doc page for Jinja but it is a well-documented templating language so you should be able to find plenty of resources on it

Things to note

- You'll notice that we always make our logic explicit. ie. for a boolean `x`, `{% if x %}` and `{% if x == True%}` are equivalent. However, we always go with the latter. This is because we parse the templates to decide what fields exactly our users will need to provide based where the fields appear and what they are dependent on. We make the logic explicit because it makes that part easier.
- Following point one, when you have two fields and one depends on the other, say `x` and `i_depend_on_x`. If you add an if statement like `{% if x == True and i_depend_on_x == True%}`, the parse will interpret both fields as being required at the same level. However, that's generally not what we mean as we want the dependent field to be required iff `x` is true. The solution to that is to nest the if statements. ie. `{% if x == True %}{% if i_depend_on_x == True%}{%endif%}{%endif%}`
- we've created a few extra filters for our internal use that you'll see being used in the tagged example like `decimal_to_string` that aren't part of the basic jinja package. When you encounter some of these, either work around it somehow or let us know in the debrief and we can discuss.

Testing your templates

We've provided you with a script that can render your templates so you can verify what you're doing as you go. First, how to install it, followed by how to use it.

Installation

Start by installing python onto your computer as the command line tool is written in python. To make sure that this has worked, open a terminal program and run:

```
python -V
```

You should see an output of `Python 3.x.x` with each `x` being a specific number depending on the version of python that you downloaded. Once python is installed, clone the repo using git and change your working directory to the project:

Create a virtual environment to install the project dependencies into and enter it. You may need to add a `3` to the end of

`python` if your version of python is 3 or later:

```
python -m venv venv
source venv/bin/activate
```

Now that the repo is downloaded, get the project dependencies with the following pip commands:

```
cd 'Tag Test'
pip install --upgrade pip
pip install -r requirements.txt
```

This should download all the requirements you need to run the command line tool. You are now ready to use the tool.

Usage

```
usage: python render_document.py [-h] json_input_file docx_template_file docx_output_file

A command line tool to create a termsheet using the Origin termsheet template with an Airbrush-compliar

positional arguments:
    json_input_file     The path to the json input file
    docx_template_file  is the path to the docx Jinja template
    docx_output_file    The path to create the docx termsheet at

optional arguments:
    -h, --help        show this help message and exit
```

To make it easy to test out your template, we have created a few example trades that you can try out. You can find these in the Examples folder. You will find both json input files and what a correct docx output file looks liks. This should also be helpful since it will give you an indication of what tags we're expecting to see in the Final Terms

Run the following command in your terminal to render a document so you can compare the output

```
python render_document.py <input_json_file> <docx_template_file> <path_to_output_file>
```