Generating Custom Word Documents From Templates Using Python

Vanessa Ekwegh Follow

Apr 9 · 5 min read

Photo by Florian Klauer on Unsplash

A few weeks ago, I started an exciting yet challenging project. The idea was to generate notification email attachments (word document or PDF) containing responses from a form owner's respondents using a custom word document template provided by the form owner.

Now that I am done with it, I would like to share a guide to implementing a feature like this and how to avoid some pitfalls I discovered in the process. At the end of this article, you will be able to generate your own custom Word documents from templates, and even add new stuff to existing documents.

Why would I want to Generate Custom Word Documents using Python?

I know, I know. Why go through the trouble of writing code to generate custom word documents when you can just create the document yourself?

Imagine if you have to type out transcripts for hundreds or thousands of students. What if you have to write generic reports based on feedback from parents or your customers, or add your company's watermark to each page of a document (or hundreds of documents)? Not looking so easy anymore, is it?

Having an application that generates these documents using templates would make things a lot easier and quicker for you. It's also a fun project to embark on if you are training to be a software developer.

What do I Need to Get Started?

You would need:

1. Basic knowledge of Python. I used Python 2.7 in this article.
2. Python installed on your machine. If you haven't installed Python before, read this for instructions.
3. Python web application. If you don't have one, you could use a framework (Flask or Django) to create one. I used the Flask framework because it's a lightweight framework and we don't need much. You can learn how to create a simple Flask application here.

4. A Package manager for Python. I would recommend using pip. You can find instructions on how to install pip here.

Do I need any External Libraries?

Yes, you need two:

1. python-docx. This is a very powerful library used for creating word documents with basically all the elements you need—images, header, footer, page breaks, etc. While this library is great for creating docx files, it's not very good at modifying them. We need this library in order to set the dimensions of any image we might add to the document, as you will see later on.
2. python-docx-template. This library is designed to generate documents using word documents as jinja templates. It makes slotting in custom values into a word document a lot easier. You can find more information in this article on library documentation here.

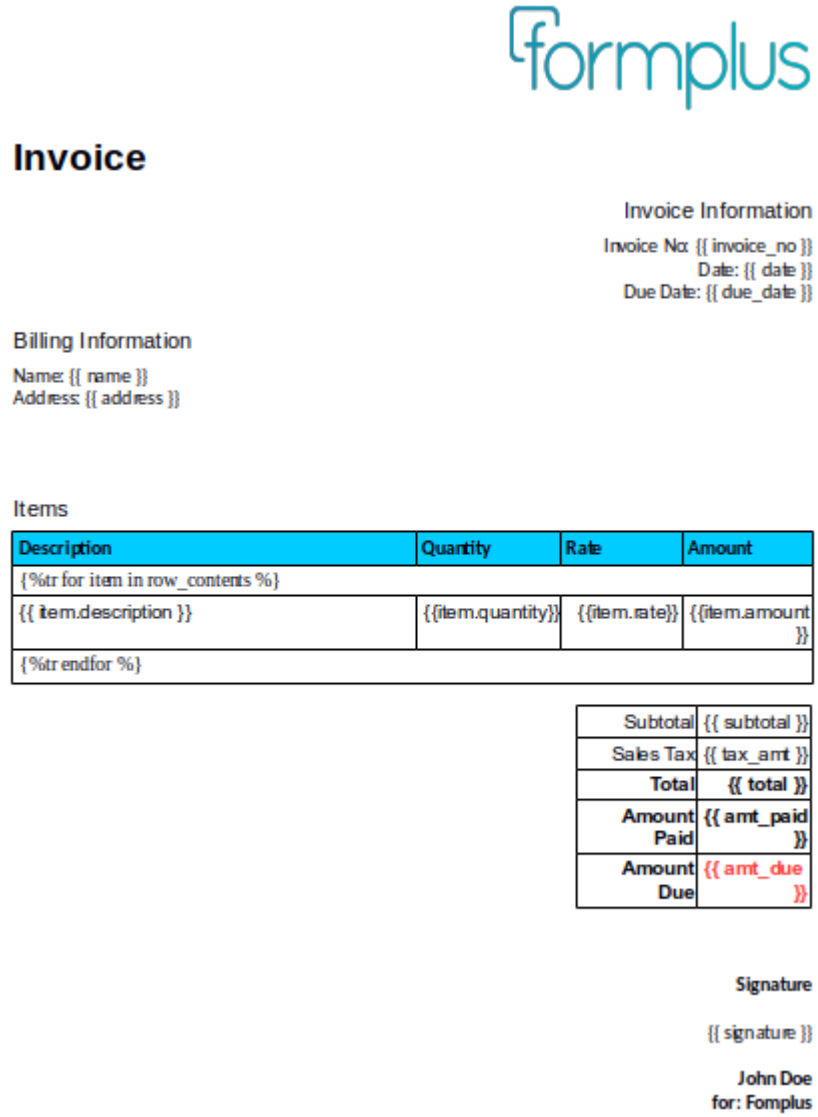
In order to install both dependencies, use the command prompt to navigate to the directory where you want to save your code and run the following commands:

```
pip install docx

pip install docxtpl
```

Creating a Word Document Template

Let's say we want to design an invoice generation system. We would need a Word Document as a template. I created a sample template. Here's what it looks like:



Invoice

Invoice Information

Invoice No: {{ invoice_no }}
Date: {{ date }}
Due Date: {{ due_date }}

Billing Information

Name: {{ name }}
Address: {{ address }}

Items

Description	Quantity	Rate	Amount
{%tr for item in row_contents %}			
{{ item.description }}	{{item.quantity}}	{{item.rate}}	{{item.amount }}
{%tr endfor %}			

Subtotal	{{ subtotal }}
Sales Tax	{{ tax_amt }}
Total	**{{ total }}**
Amount Paid	**{{ amt_paid }}**
Amount Due	{{ amt_due }}

Signature

{{ signature }}

John Doe
for: Fomplus

For the purpose of simplicity, I'll be using the following tags (there are more tags in the python-docx-template library documentation article):

Tag	Description	Usage
{{ var }}	Single variable.	{{ invoice_number }}
{%tc jinja2_tag %}	Table column. This tag can be used to implement for loops (e.g. adding multiple columns to a table) and conditional statements (e.g. display or hide a column based on the value of a variable) for table columns.	**For Loop:** {%tc for item in col_contents%} … your tags here {%tc endfor %} **Conditional Statement:** {%tc if invoice_number %} {%tc endif %}
{%tr jinja2_tag %}	Table row. This tag can be used to implement for loops (e.g. adding multiple rows to a table) and conditional statements (e.g. display or hide a row based on the value of a variable) for table row.	**For Loop:** {%tr for item in row_contents%} … your tags here {%tr endfor %} **Conditional Statement:** {%tr if invoice_number %} … your tags here {%tr endif %}

It is important to note that you should not put the same tag twice in a run, paragraph, table column or row. The library documentation explains better.

Generating Document From Word Template

Preparing Template Context

The context contains the information we would like to add to our document. It is a dictionary with key-value pairs, where the key is the name of the variable we included in our template and the value is what we would like to substitute the variable with. We are adding single variables, an array of elements for the for-loop and an inline image.

Single Variable

We would need the following variables for our invoice template:

Variable	Tag Name
Invoice Number	{{ invoice_no }}
Date	{{ date }}
Due Date	{{ due_date }}
Name of Customer	{{ name }}
Billing Address	{{ address }}
Subtotal	{{ subtotal }}
Sales Tax	{{ tax_amt }}
Total	{{ total }}
Amount Paid	{{ amt_paid }}
Amount Due	{{ amt_due }}

Our context should look like this:

```
{
    'invoice_no': 12345,
    'date': '30 Mar',
    'due_date': '30 Apr',
    'name': 'Jane Doe',
    'address': '123 Quiet Lane',
    'subtotal': 335,
    'tax_amt': 10,
    'total': 345
```

For Loops

The context for adding data to the Word document using for loops takes the form of an array of elements. For example, in our invoice, we would like to add all the items purchased to the table. We already have a for loop in the template that looks like this:

{%tr for item in row_contents %}			
{{ item.description }}	{{ item.quantity }}	{{ item.rate }}	{{ item.amount }}
{%tr endfor %}			

The variable row_contents will hold the array of elements we need to add to the table as rows.

If our customer bought:

1. 30 dozens of eggs at $5 for each dozen. The total amount is $150.
2. 10 bags of all-purpose flour at $15 for each bag. The total is $150.
3. 5 bags of granulated sugar at $7 each. The total amount is $35.

Our final context with all the required variables (except signature—we'll get to that later) should look like this:

```
{
    'invoice_no': 12345,
    'date': '30 Mar',
    'due_date': '30 Apr',
    'name': 'Jane Doe',
    'address': '123 Quiet Lane',
    'subtotal': 335,
    'tax_amt': 10,
    'total': 345,
    'amt_paid': 100,
    'amt_due': 245,
    'row_contents': [
        {
            'description': 'Eggs',
            'quantity': 30,
            'rate': 5,
            'amount': 150
        }, {
            'description': 'All Purpose Flour',
            'quantity': 10,
```

Signature

You could create a separate function that adds the signature to the context you already have. It could look like this:

```
from docx.shared import Cm
from docxtpl import DocxTemplate, InlineImage


def add_signature(template, context, signature):
    tmp = DocxTemplate(template)

    img_size = Cm(7)  # sets the size of the image
    sign = InlineImage(tmp, signature, img_size)
```

Please note that the *signature* argument could be either a file path or a string buffer that holds the signature image.

Bringing It Together—Generating the Word Document

The code for generating the word document can be found in this github repo.

Here's how the document should look like when downloaded:

formplus

Invoice

Invoice Information

Invoice No: 12345
Date: 30 Mar
Due Date: 30 Apr

Billing Information

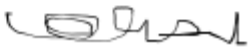
Name: Jane Doe
Address: 123 Quiet Lane

Items

Description	Quantity	Rate	Amount
Eggs	30	5	150
All Purpose Flour	10	15	150
Eggs	5	7	35

Subtotal	335
Sales Tax	10
Total	**345**
Amount Paid	**100**
Amount Due	245

Signature



John Doe
for: Formplus

Tips

1. You can open your Word document template using a file path, as shown in this article, or a string buffer containing the word document (in case you plan to get your Word template from a database). The latter eliminates the need for the template to be in the same directory as the Python script.

2. In order to add an inline images to your Word document, ensure that your images are in bmp, gif, jpeg, tiff or png file formats. The library only supports these file formats for images.

You can set the size of your inline image using Cm, Inches, Mm, Emu, Pts, and Twips. Just import them from the *docx.shared* module like this:

```
1  from docx.shared import Cm, Inches, Mm, Emu  # add other uni
```

Wrapping Up

This project is a great way to learn and practice coding using Python, components of Word documents and file processing. It also helps automate simple tasks, saving time and energy.

If you want to learn how to take a web page offline using service workers, this article on the Formplus blog is a great guide. Take a look and let me know how it goes.