Introduction to Generating PDFs

Depending on what your automation does, you might want to generate a PDF report at the end, which lets you decide exactly how you want your information to look like.

There's a few tools in Python that let you generate PDFs with the content that you want. Here, we'll learn about one of them: ***ReportLab***. ReportLab has a **lot** of different features for creating PDF documents. We'll cover just the basics here, and give you pointers for more information at the end.

For our examples, we'll be mostly using the high-level classes and methods in the ***Page Layout and Typography Using Scripts (PLATYPUS)*** part of the ReportLab module.

Let's say that I have an awesome collection of fruit, and I want to create a PDF report of all the different kinds of fruit I have! I can easily represent the different kinds of fruit and how much of each I have with a Python dictionary. It might look something like this:

```
fruit = {
  "elderberries": 1,
  "figs": 1,
  "apples": 2,
  "durians": 3,
  "bananas": 5,
  "cherries": 8,
  "grapes": 13
}

```

Now let's take this information and turn it into a report that we can show off! We're going to use the **SimpleDocTemplate** class to build our PDF.

```
>>> from reportlab.platypus import SimpleDocTemplate
>>> report = SimpleDocTemplate("/tmp/report.pdf")
```

The **report** object that we just created will end up generating a PDF using the filename **/tmp/report.pdf**. Now, let's add some content to it! We'll create a title, some text in paragraphs, and some charts and images. For that, we're going to use what reportlab calls ***Flowables***. Flowables are sort of like chunks of a document that reportlab can arrange to make a complete report. Let's import some Flowable classes.

```
>>> from reportlab.platypus import Paragraph, Spacer, Table, Image
```

Each of these items (**Paragraph**, **Spacer**, **Table**, and **Image**) are classes that build individual elements in the final document. We have to tell reportlab what ***style*** we want each part of the document to have, so let's import some more things from the module to describe style.

```
>>> from reportlab.lib.styles import getSampleStyleSheet
>>> styles = getSampleStyleSheet()
```

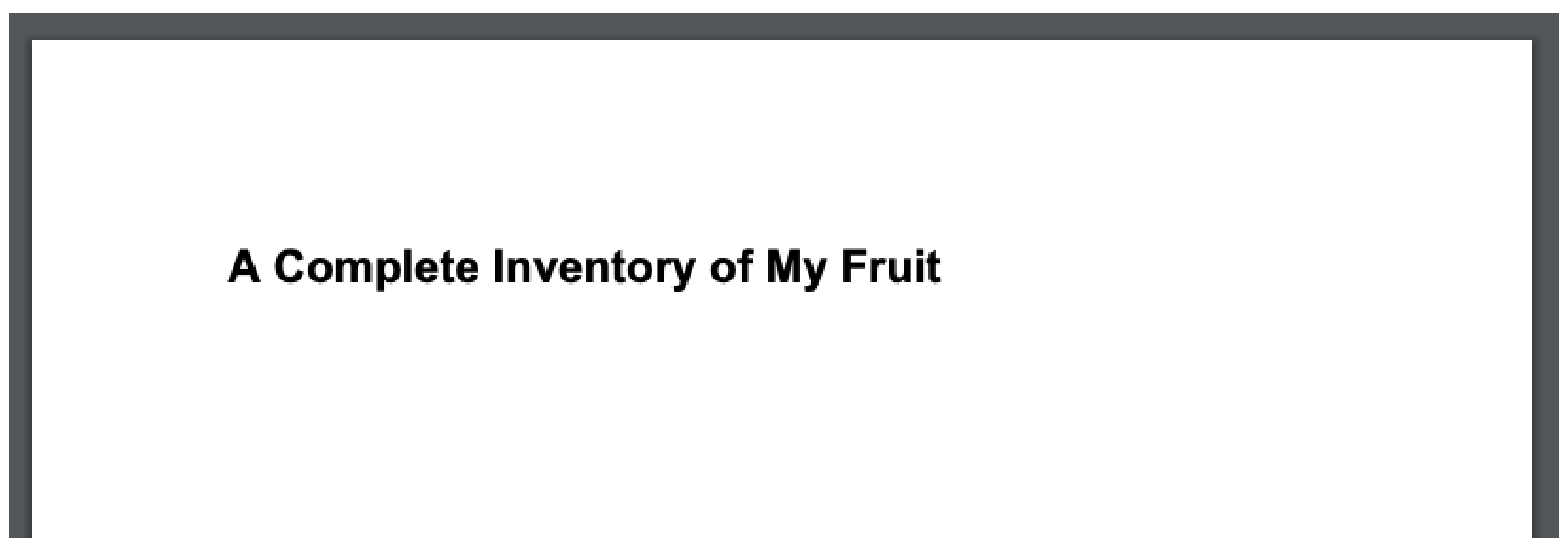
You can make a style all of your own, but we'll use the default provided by the module for these examples. The **styles** object now contains a default "sample" style. It's like a dictionary of different style settings. If you've ever written HTML, the style settings will look familiar. For example **h1** represents the style for the first level of headers. Alright, we're finally ready to give this report a title!

```
>>> report_title = Paragraph("A Complete Inventory of My Fruit", styles["h1"])
```

Let's take a look at what this will look like. We can build the PDF now by using the **build()** method of our report. It takes a list of Flowable elements, and generates a PDF with them.

```
>>> report.build([report_title])
```

Okay, now let's take a look at the PDF:


A Complete Inventory of My Fruit

It's not much, but it's a start!

Up next, we'll look into an interesting Flowable for our reports: Tables.

Mark as completed

Like Dislike Report an issue