Budget Tables with Python and Latex

Jason Krasavage

October 12, 2018

1 Introduction and Motivation

For quite some time I have been using the whiteboard in my workspace to keep track of the various items that I plan to spend my money on. I simply write down what they are, and how much they cost. I then keep an updated note of the funds that are currently in my bank account, and what will be remaining if/when I buy every item on my list. This is simply because I like to know, at a quick glance, what the net worth of my bank account is.

After doing this for a while, I began to realize it was becoming a little bit of a hassle: erasing and rewriting every day or so. As I started to use LaTeX, I grew very fond of the tables you could produce with it. I wanted to, instead of using the whiteboard, use LaTeX to make the tables for my budget, and I did; however, this also became very cumbersome, dealing with the math by hand when my budget became large and complicated.

Then I finally decided to do the whole process programmatically, using Python and LaTeX in unison. I wrote a small Python 3 script that allows you to enter the items, their prices, the money in your bank account, and the desired color theme of the table. After running this script you are left with a .pdf built using LaTeX that is dumped in the same directory that the script was in.

2 Requirements

This script requires a number of components to run properly. If you have any trouble using it for some reason, feel free to contact me via the email in the README that is in the Github repository.

The components you need to run this script are:

- A Python 3 distribution
- A Perl 6 distribution (I use ActivePerl)
- A current installation of LaTeX (I use the MacTex distribution)
- The *PyLatex* module for Python 3
- *pdfCropMargins* from the Python Package Index

If you have these installed on your machine, then you should have no issues. The *PyLatex* module was built for Python 3 only, which is why this script will only work with Python 3. The PyLatex module actually uses Perl 6 to interface with LaTeX on your machine, which is why that must be installed. *PyLatex* is a module for Python 3 that generates and renders .tex files via Python code, it is available through the Python Package Index (*'pip3 install pylatex'*). pdfCropMargins is a command line utility that automatically crops pdf files based on certain, user selected, parameters.

3 Usage

Interaction with this piece of software starts in the Python 3 script. There are **4 variables** that are of interest to the user: items (list), prices (list), bank (int), and chosenColor (string).

items and *prices* are two lists who's indices **must correspond**. For example, the index #1 of items is 'September Paycheck', and index #1 of prices is the amount I am anticipating for that paycheck. Likewise, index #3 of items is my 'Food Budget' for the month, so index #3 of prices is the amount that I expect to spend on food this month. Note that for this entry, the price value is negative, because it is money than I expect to *spend* rather than earn.

The bank variable should be changed to the current balance of your bank account, and the chosenColor variable can be used to modify the color of the resulting table.

Available colors are as follows, and to use them correctly the color (string) variable needs to match these texts perfectly (for example, no space when you are using dark or light gray).

- red
- green
- blue
- cyan
- yellow
- magenta
- brown
- darkgray
- gray
- lightgray
- lime
- olive
- orange
- pink
- purple
- teal
- violet

After modifying all 4 variables, and running the script, a file called BudgetTable.pdf will be written to the same directory as the BudgetTable.py script. If you open this file, you will see the table in the upper left hand corner, occupying a fraction of the available space in the document.

The excess white space in the document makes it difficult to successfully embed the pdf somewhere it will be useful (on your desktop, in an email, in a text message, etc.). To work around this I have opted to use *pdfCropMargins*. To install this utility, type "pip install pdfCropMargins" into your terminal. After doing this, navigate to the directory in which the both the script and newly created table pdf reside. Then, execute this command: "pdf-crop-margins -p4 0 0 0 0 BudgetTable.pdf -o NewBudgetTable.pdf". This will crop the document to the exact size of the generated table, and write it to a *new* file: NewBudgetTable.pdf.

Now, the pdf is much easier to use and embed. For my personally use, I paste these pdf tables into "Stickies" that are on the desktop of my Mac Laptop, so that they are always easily visible to me. With a little more automation and thought, a user could write a small script on top of this to automatically update the end environment that the table will be shown, so that each time a new table is produced it is updated in the final location.