SWE/TBP Career Fair Directory & Registration

Automatic Directory and Table of Companies Generator Guide

# Preface

The Directory has long been a staple of the Career Fair. Samples exist back to the mid-90s—showing the development of the Directory since that time. As the number of companies has grown, the challenge of presenting the data to attendees in a usable, coherent manner has created a heavy burden on chairs and has led to problematic transcription errors, stemming from a variety of causes. In 2013 it was set about to create a simple tool that would take the registration data and assemble it into a usable directory, with minimal human intervention needed. To build on that the tool was expanded to further create the binder covers and a [html table](http://umcareerfair.org/companies) of company data. What follows is a user guide, with minimal technical guide interleaved. This should obviously be a living document that adapts to the growth of the tool over the years.

# Revision History

|  |  |  |
| --- | --- | --- |
| Date | Author | Notes |
| 20 February 2014 | Mike Hand | Document Created. |

# Prerequisites

Before getting started there are a couple of provisos—a couple of quid pro quos. Or at least there are some software packages you’ll need.

## Required Software

1. Python 2.7. Python is an open source computer scripting language–kinda like Matlab, except less sucky. This can be downloaded from: <http://www.python.org/download/releases/2.7/>
2. LaTeX, particularly XeLaTeX. LaTeX (pronounced Luh-tech) is an open source typesetting suite designed to streamline the creation of technical documents, books, etc. It’s main reason for use here is that it allows programmatic creation of documents, which is kind of the point here. This will depend slightly on what operating system you’re using, for Windows I’d recommend the MikTeX distribution (<http://miktex.org/2.9/setup>). For Mac, I’m not sure, but make sure that it supports XeLaTeX.

## Setting up the Directory

For better or worse, the script is somewhat naïve and expects that the working directory be configured in a particular manner. This is shown in Figure 1. Note in particular that any directory accessed by the script has no whitespace in the name.

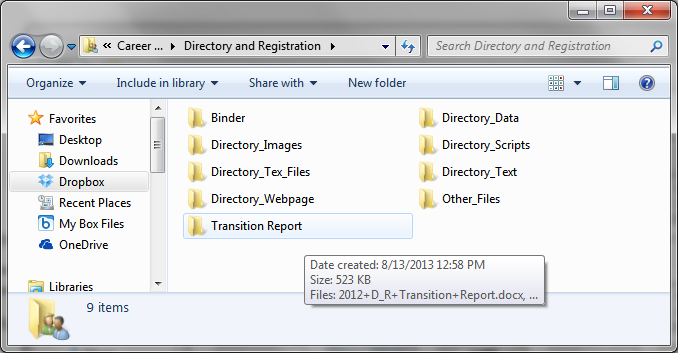


Figure 1: Directory Structure

## Necessary files

A lot of data goes into the directory, and not all of that is ready until quite late in the process. As such, it is recommended to continue using the each piece of information, data, image, etc. from the year prior until it is replaced. There’s no need to hold off on checking things until the last sponsor letter is ready—just use the ones from last year.

### Directory Data Folder

Contains spreadsheets of data (csv formatted). Excel format will not work, csv is the simple, unmarked up way of handling spreadsheet data.

1. administration.csv: This file contains the names and titles of those in the college administration being thanked.
2. chairs.csv: This file contains the names and committees of each chair on Career Fair. Note the formatting as it is slightly non-standard.
3. companiesYYYY.csv: This file contains the company registration data from which basically everything is assembled. The program is somewhat robust to formatting here, as it checks the header row to locate what information is where. It’s still a program however, and if it doesn’t know that the company name is abbreviated “comp\_nm” (why would it?) it won’t find that information, so don’t change up the header labels if it can be avoided; and if you have to, make it sensible.
4. directors.csv: Contains the names, society affiliation and signature file (no extension) of each director.
5. ecrc.csv: Contains the people in the ECRC that are being thanked
6. facilities.csv: Contains the people in facilities that are being thanked.

Each of these file names can be changed, but you’ll have to tell the program at run-time what file it is. It’s easiest to just use these file names, apart from the companies list.

### Directory Text Folder

This folder contains text files with the welcome letter from the directors as well as each of the sponsor letters. Note that the files are text format (not Word). By default, it assumes 3 Sponsor letters, since the file names use the current year, you’ll have to specify these at run time (or tweak the code).

### Directory Tex Files Folder

This folder will contain the tex files output by running the program. These include “Directory\_M\_DD.tex”, “BinderCover.tex”, and a lot of process files (.log,.aux, .synctex.gz, .out, and others). There are also a number of static files that are stored here.

1. AuxiliaryStuff.tex: Defines some of the default style things. **Go in and change the year to the present** (ctrl+F to make sure you find all of them). Other than that, you shouldn’t need to make any modifications.
2. FontPage.tex: This is the tex file that defines the front page of the binder. You don’t need to worry about most of this, just **go in and change the year and dates. Further, change the logo file**, the one currently in there is named “christie\_vector\_no\_text”. The image file referred to should be in a vector (.pdf, or .eps) format and be located in the Directory Images folder.
3. MapAndShit.tex: This is the file which contains the code for the page that has the map and…shit… on it. At any event, unless you change the list of majors, buildings, or hiring policies, you shouldn’t even need to open this.

### Directory Webpage Folder

This contains auxiliary files used to present the company data table. In particular there are 2 CSS files (bootstrap.css and bookswap.css) which define the presentation, and 4 Javascript files (bootstrap.min.js, jquery.js, sort\_filter\_table.js, and sticky-header.js) which enable certain types of interactivity (sorting/filtering the table and sticking the table header to the top of the screen, for instance). Nothing needs to change here, so let’s move on.

### Directory Images Folder

This file contains the images used in generating the directory, as well as a repository of company logos for use should any company be difficult to acquire a high-res logo from. The naming scheme can be specified at runtime but should also be reasonably apparent from the files present (gold, platinum, diamond sponsor logo files as well as the SWE and TBP and CF logos, and the Director signature files are present, among others).

# Actually Running the Tool

Python, like C++ if you remember ENGR 101, is run from the command line. On Mac or Linux this will be much like it was in 101. On Windows, things are unsurprisingly different. First go ahead and open the command prompt (cmd on Windows).

## How to get help

Navigate to the directory you’ve set up and drop into the scripts directory (cd is the command to change directories). Then run “python CF\_SpreadsheetReader.py –help”. (**Note:** you may need to modify your path variables to do this. If you’re not sure what that means, you should ask a CS major or just replace the word “python” with the full path to where the python executable lives—likely C:\Python27\python.exe). This will pull up the help menu and will explain what the different runtime options are and how to use them. In all honesty as you’ll be running this quite a bit, it’s actually easier to just go in and modify the code. We’ll get to that later. Figure 2 shows the output resulting from running the help command. Note that most of the options are just specifying the file names we’ve previously covered. It also notes what the default values are for each so that you have an idea of what you’ll need to modify. Finally it notes how to actually go about specifying those options “—option1 <value> --option2 <value>”. Let’s assume you’ve got these all specified, the next step is to, well, actually run the thing.

## The part where you actually run it

Assuming the code is in its default state running the program will do 4 things.

1. Generate the .tex file containing all of the directory data.
2. Compiling said .tex file into a pdf that can be used.
3. Generating a .tex file for the binder covers (this one doesn’t compile by default as it’s a much simpler document)
4. Generates the html file containing the company “spreadsheet”

To actually run the code, type “python CF\_SpreadsheetReader.py” and click enter. It will take a while as there is a bunch of data, but it should generate everything you need. The most expensive part of the operation is generating the pdf from the .tex files. When it finishes, it will copy the final version of the pdf out into the main level of the D&R directory with the name “Current\_Directory.pdf”. It will leave the BinderCover.tex and the html files in-place.

That’s it. Assuming there were no issues, you have a fully functional directory. If things look amiss, the Appendices have some information on how to modify the code (both Python and LaTeX) to address issues that may arise. Good Luck!



Figure 2: Help menu

# Appendix A: Modifying the Code

Sometimes there are some problems that arise that it is necessary, or just easier to go and tweak the code to get the behavior you want. The first case is when you modify the code to actually generate the documents. The second is to modify the output to address things that really cannot be handled programmatically and need to be manually tweaked to produce a sane looking document.

## Appendix A.1 Modifying Python

The first main reason to modify the Python code is that you want to change what the default values are for file names, sponsor names, etc. and you’d prefer not to have to do this every time you run the program. If you’re comfortable with Python (it’s actually not that difficult, and totally worth learning) open up the program “CF\_SpreadsheetReader.py” and scroll to near the bottom (around line 670). Inside the main function you’ll find the runtime argument/option definitions and will note the default values specified as well. Change these to whatever you need them to be and you won’t have to specify them at run time.

The other main reason for modifying the Python code is if, say, the directory is stable but you want to keep updating the webpage or the binder covers. In this case you can comment out (#) the lines that result in the compilation of the directory (around line 790 you’ll find 4 lines that do sequentially the 4 things noted in the section on running the tool).

## Appendix A.2 Modifying LaTeX

This is a bit trickier, and should only be done once you’re reasonably certain that the underlying data is stable. The reason is that anything you change manually in the LaTeX code (apart from the parts in the static files noted previously) will be blown away the next time you run the program. The only things that should really be handled here are matters of scaling and spacing that didn’t work well when it ran. If you’re not familiar with LaTeX and how to fix scaling or spacing, it’s a bit beyond the scope here, but almost any grad student could help you out. That failing, google “how to resize images in latex” or “how to add (remove) vertical space in latex”. Stackoverflow is a wonderful resource.