Computer Science 135 (Spring 2017)
Duane A. Bailey
Laboratory 4
Presenting Data (due 11pm, Sunday)

Objective. To learn how to set up a personalized environment for data presentation.

This week we will experiment with a couple of approaches to presenting data. The first is to use the module `matplotlib.pyplot` to present our data as a graph. The second is to use the Python Image Library (`PIL`) to build images of data.

Before we get started, however, we'll have to install both of these tools in our own directories. This is a simple process that you must do once for each environment you use. Once the utilities have been installed, you have the option to *activate* them, or not.

Installing new modules.
Few Python environments have *all* the modules you might want to eventually have installed. This week we'll use the *virtual environment* system to allow us to experiment with adding new modules. First, we set up and *activate* the environment, then we install the new software. Here are the steps:

1. Before we can do this, we need to see what the `virtualenv` command is called. Type `virtu` and then hit TAB. The computer will complete the command. On our computers, it is `virtualenv-3.4`, but in other environments it may be something else. I'll assume it's `virtualenv-3.4`

2. Go to your home directory and create the environment in your `cs135` folder (the $ is a prompt):

   ```
   $ cd ~
   $ virtualenv-3.4 --system-site-packages -p python3 cs135
   Running virtualenv with interpreter...
   Using base prefix...
   New python executable...cs135/bin/python3
   Also creating executable in...cs135/bin/python
   Installing setuptools, pip, wheel...done.
   ```

   The phrase `--system-site-packages` indicates that you want to build on the environment you already have set up on your machine. The `-p python3` says "if I type `python`, I mean `python3`"; a nice feature, but not required.

3. We now go into our `cs135` folder and *activate* this new environment:

   ```
   $ cd cs135
   $ source bin/activate
   (cs135)$ python --version
   Python 3.4.5
   (cs135)$
   ```

   If the activation worked, you'll see your usual prompt, prefixed with the name of the folder that contains your virtual environment. In addition, `python` now refers to Python version 3.4.5. The

source command is simply saying "execute the commands found in the activate script in the bin subdirectory." Those commands changed your prompt, and told the shell that the bin subdirectory contains all the new modules.

Once an environment is activated, you can de-activate it (if you want to) by typing deactivate. Don't do that now, or everything will go back to normal, with python refering to version 2 of Python and not 3.

4. New modules are installed with pip (silly: "Pip Installs Python"). First, install the library module, matplotlib:

```
(cs135)$ pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-2.0.0...
    100% |################################| 49.7MB 7.6kB/s
... output omitted ...
Installing collected packages:...matplotlib
Successfully installed...matplotlib-2.0.0...
(cs135)$
```

   Now, we'll force matplotlib to build the fonts you might need for this machine:

```
(cs135)$ mkdir ~/.matplotlib
(cs135)$ echo 'backend: Agg' > ~/.matplotlib/matplotlibrc
(cs135)$ python
>>> import matplotlib.pyplot as plt
...Matplotlib is building.... This may take a moment.
>>> quit()
```

   The matplotlib environment is now installed.

5. Next, install the python image library enviornment:

```
(cs135)$ pip install pillow
Collecting pillow
 ....
Successfully installed pillow....
```

   If you're successful, you'll be able to use these modules for any new code you write, as long as you activate the environment at the beginning of each new session (*i.e.*. whenever you log in, open a Terminal window, *etc.*).

Now we'll get to work on displaying some data.

The remainder of the lab will generate a number of visuals. Let's grab some starter files by cloning the lab4 repository (use your CS id instead of 18xyz):

```
(cs135)$ cd ~/cs135
(cs135)$ git clone ssh://18xyz@gala.cs.williams.edu/~cs135/18xyz/lab4.git lab4
```

First, we'll play with `matplotlib.pyplot`. I'll show you how to draw some simple graphs, but I encourage you to learn more by investigating the `matplotlib` tutorial at

```
http://matplotlib.org/users/pyplot_tutorial.html
```

Task 1: Baseball.
In the `lab4` directory, you'll find `cubs16.csv`. This is a record of every game the Chicago Cubs played in 2016, stored as a CSV file. It's a story of glee that I grabbed from `baseball-reference.com` (go to Teams, Chicgo Cubs, click on Schedule, and collect the data in CSV form). If you're another kind of fan, you can grab data from your favorite team.

Column 0 (the first column, `Gm#`) is the game number: it marches from 1 to 162. Column 6 (the 7th column, `W/L`) is ether `W` or `L`. Notice that this file has a "header" every 50 games or so, which should be skipped. These headers can be recognized when column 6 is not exactly equal to `W` or `L`. We want to stream through this file and, for each game determine the "number of games over .500": it's simply the number of wins less the number of losses. We'll compute a list of 162 values called, say `gamesAbove`.

Given that, you can then use `matplotlib` to plot the trend over a season:

```
# at the top:
import matplotlib.pyplot as plt
... compute games above ...
n = len(gamesAbove)
plt.title("The 2016 Chicago Cubs Season")
plt.xlabel('Game number')
plt.ylabel('Games above .500')
plt.plot(range(1,n+1), gamesAbove, 'r-')
plt.axis([1,n, -25,25])
plt.savefig('baseball.pdf')
```

Please commit and push the graph `baseball.pdf`. Make sure your README is updated so that we can see which team you selected.

Task 2: Trends in Wallstreet?
We would like you to investigate some other time-oriented data of your choice and present the data in whatever way you desire. This visual should be submitted as `trend.pdf`. Please add some notes in README describing the origin of the dataset you used to generate `trend.pdf` and a sentence or two describing what's being displayed. Minor extra credit will be given if you demonstrate mastery of some new skills in your `matplotlib`-based plot.

Task 3: Wolfram's Rule 110 automata.
In this task we have an array of booleans that represent life in 100 row houses along a street. Each year houses are either occupied or not. The rule for determining whether a house (at the "center" of our attention) is occupied is based on whether there are neighbors to the left or right:

Left	Center	Right	Result
empty	empty	empty	empty
empty	empty	occupied	occupied
empty	occupied	empty	occupied
empty	occupied	occupied	occupied
occupied	empty	empty	empty
occupied	empty	occupied	occupied
occupied	occupied	empty	occupied
occupied	occupied	occupied	empty

I'd like you to create a 100×100 (or larger) image using PIL where the pixel at row `r` and column `c` represents whether house `c` is occupied in year `r`. So, for example, if `hood` is a list of 100 boolean values (*i.e.*. `hood[c]` is `True` when house `c` is occupied), we can write a row of blue-or-white pixels at row `r` with the following snippet of code:

```
from PIL import Image, ImageDraw
def createImage(width, height):
    """Create a width x height RGB PIL image, initially all white."""
    return Image.new("RGB", (width, height), (255,255,255))
def drawPoint(image, x, y, color):
    """Draw a color (RGB) pixel at x,y in image."""
    ImageDraw.Draw(image).point((x,y), color)
def saveImage(image, filename):
    """Write image to a PNG file."""
    image.save(filename, "PNG")
img = createImage(100,100)
hood = [ ... ]
for row in range(100):
   for col in range(100):
       drawPoint(img, col,row, (0,0,255) if hood[col] else (255,255,255))
img.save("wolfram.png")
```

Construct an image, `wolfram.png`, that is generated from observing—over 100 years—a neighborhood of 100 houses. Initially they are unoccupied, except for the center 20 which are *randomly* occupied.

Update the README file to include a description of what happens, qualitatively, when each of the following initial conditions are true:

1. The neighborhood is initially empty.
2. Every house is initially occupied.
3. Some houses near the center are initially occupied. The image `wolfram.png` corresponds to this case.

$\star$