Assignment 1

In this assignment, you will develop a tool that visualizes a schedule. Your script will take in a file that contains data about when specific jobs run and will output an image that shows that schedule.

Part 1: Connecting to the server and setting up your workspace.

1. Connect to server. Use whatever tool you use to connect to `classroom.cs.unc.edu`, but instead connect to `comp590f18.cs.unc.edu`.
2. Check that you are in your home directory now.

```
pwd
```

This should return `/home/<yourCSlogin>`.
3. Make a new directory called HW1.

```
mkdir HW1
```

4. Copy the graphics library in to that directory:

```
cd HW1
cp /home/shared/graphics.py .
```

5. Copy the sample data input (`data1.json` and `data2.json`) to your directory:

```
cp /home/shared/data* .
```

6. Copy the template to your directory and rename it `test.py`:

```
cp /home/shared/template1.py test.py
```

Part 2: The assignment.

Input: A `.json` file. It contains information about the tasks and each job execution.
Output: A `.ps` file. This will be an image which, by the end of Section B of this assignment, will represent the schedule described by the job executions in the JSON file.

A - Checking the basics

1. Run the script.

```
python test.py data1.json
```

This runs the script `test.py` and passes in the argument `data1.json`. It should produce an image called `schedule.ps`. At this point, the data in `data1.json` is not actually used in producing any part of the image.
2. Convert the `.ps` file to a `.pdf`.

```
ps2pdf schedule.ps <pdfName>
```

If you leave `<pdfName>` blank, the default name will replace the `.ps` with a `.pdf`, so the new file would be `schedule.pdf`.

3. Crop the .pdf, because the ps2pdf command defaults to producing a 8.5”x11” pdf.

```
    pdfcrop schedule.pdf <newPdfName>
```

If you leave `<newPdfName>` blank, the default will be `schedule-crop.pdf`.
4. Compare the cropped image to `schedule.pdf` in the shared folder; they should match. Do this by copying both images to your laptop or whatever machine you are working on.

B - Testing the waters

5. Modify `test.py` to do the following:

- Draw a line from the center of the canvas to the upper right hand corner.
- Draw a small, colored rectangle somewhere on the canvas.
- Print out the maximum time (`max_t`) from the JSON file by using a `print` statement.
- Write your name in text somewhere on the canvas.
- Remove the code to generate the original circle and line.

To complete the above, look at the example line and circle in the original program. Additionally, look at `graphics.py`, the library that provides an easy way to create a circle, rectangle, polygon, etc. Toward the end of that file, the function `test()` is defined and has examples of common objects. Note that the coordinate (0, 0) is in the upper left corner of the canvas. Use your prefered search engine to look up “Tkinter colors” to see a list of pre-named colors or use a tool like https://htmlcolorcodes.com/color-picker/ to find the RGB values that correspond to the color(s) you would like to use. For further references, see below. Save the image that you produce as `PartB.pdf`.

C - Crafting a visualization tool

1. Copy your code from Part A to a new file, called `visualizer.py`.

```
    cp test.py visualizer.py
```

2. Draw one line per task in the JSON file passed in. Your visualizer should handle task sets with 1-6 tasks. Each task specified in the JSON file with a task number (`num`), its phase (`phase`), its computation time (`c`), and its relative deadline (`deadline`).
3. Fill in the function to draw an up arrow at a specific location. Feel free to change the function definition to specify height, width, etc.
4. Determine the number of pixels corresponding to one unit of time given the maximum time of the schedule, stored as `max_t` in the JSON file.
5. Add arrows to each task’s line to indicate job releases and deadlines.
6. For each execution in the list, draw a rectangle of the appropriate length (based on your chosen time scale) on the correct task’s line. Color this rectangle based on the task; all tasks should have a different color. Each job execution is listed in the JSON file with the corresponding task number (`taskNum`), start of that chunk of execution (`start`), and end of that chunk of execution (`end`).
7. Run your visualizer with `data1.json` and `data2.json` as input and generate `Schedule1.pdf` and `Schedule2.pdf`, respectively.

Part 3: Submitting your assignment.

Your assignment should be in `/home/<yourCSlogin>/HW1`. I will collect homework from here and check the last time that each file was modified. This assignment is due on Sept. 5, 2018 at 9:05am EST. If the files have been edited after that time, I will assume that you have chosen to use one or more late days. If you would like to continue tweaking your solution, do so in a different folder. Make sure that the following files are in `/home/<yourCSlogin>/HW1`, especially if you were working in a different directory or on a different machine:

- `test.py`
- `PartB.pdf`
- `visualizer.py`
- `Schedule1.pdf`
- `Schedule2.pdf`

Your schedule visualizer will also be tested with different inputs.

Appendix: Additional resources

RGB Color Picker: https://htmlcolorcodes.com/color-picker/
The Python Tutorial - we are using version 2.7: https://docs.python.org/2.7/tutorial/index.html
What is JSON?: https://www.pythonforbeginners.com/json/what-is-json