# CS 196: Data Hackerspace

Assignment 1: Data Parsing and NumPy

September 18, 2018

**Instructions:** This homework will be due electronically on Monday, ~~September 24th~~ at 7pm as a submission on Github. Please write your answers as a `.py` file (Jupyter notebooks will not be graded) and commit it to a repository called `data_hackerspace_homework`. Fill out the Google Form at https://goo.gl/forms/mjDt9hBOanQ2AbnF3 so we can associate your netID and your Github profile. Skeleton code is given as `hw1.py`. Do not edit anything outside of the given functions.

## 1 Parsing CSV Data

Fill in the function `histogram_times` to take in name of a CSV file in the same format as `airplane_crashes.csv` (in the archive) and return a Python list of how many crashes occurred at each hour of the day. Ignore rows where no time is given. For example, if you were given a file which contains only the following lines:

```
Date,Time,Location,Operator,Flight #,Route,Type,Registration,cn/In,Aboard,Fatalities,Ground,Summary
09/17/1908,17:00,"Virginia",Military - U.S. Army,,Demonstration,Wright Flyer III,,1,2,1,0,"Dur..."
07/12/1912,06:30,"AtlantiCity, New Jersey",Military - U.S. Navy,,Test flight,Dirigible,,,5,5,0,"Fir..."
08/06/1913,,"Victoria, British Columbia, Canada",Private,-,,Curtiss seaplane,,,1,1,0,"The f..."
09/09/1913,18:30,Over the North Sea,Military - German Navy,,,Zeppelin L-1 (airship),,,20,14,0,"The ai..."
10/17/1913,18:30,"Germany",Military - German Navy,,,Zeppelin L-2 (airship),,,30,30,0,"Hydr..."
```

your function should return the Python list

```
[0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,2,0,0,0,0,0]
```

## 2 Parsing JSON Data

In this section, you write functions to take in name of a JSON file in the same format as `pokedex.json` (in the archive)

### 2.1 Gotta Weigh 'Em All

Fill in the function `weigh_pokemons`, which takes in two parameters: a JSON file and an float representing a weight in kilograms. The function will return a list of pokemon names who weigh the same as the given input weight. Return empty list if no pokemons are found with the same weight.

If the input JSON file is `pokedex.json` weight is `10.0`, the function `weigh_pokemons` should return

```
["Kakuna", "Magikarp"]
```

### 2.2 Single-type Pokemon

Fill in the function `single_type_candy_count` which takes in one parameter: a JSON file. The function will return an integer which is the total count of single-type pokemons' candies. If the candy count for a pokemon is not defined, assume that the pokemon has no candies.

If the input JSON file is `pokedex.json`, the function `single_type_candy_count` should return

```
2712
```

## 3 NumPy

This section will test your ability to use NumPy for linear algebra and statistics.

## 3.1 Reflections and Projections

Fill in the function `reflections_and_projections`, which takes in a two-dimensional NumPy array with 2 rows and $n$ columns (think of these as multiple points in the plane) and does the following operations to each point in order:

1. Reflects the point over the line $y = 1$
2. Rotates the point $\frac{\pi}{2}$ radians around the origin
3. Projects the point onto the line $y = 3x$

Then, the function should return a $2 \times n$ NumPy array of the transformed points.
For your reference, the operation to rotate a point $(x, y)$ by $\theta$ radians in 2d space is given by

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

and the operation to project a point $(x, y)$ onto a line $y = mx$ is given by

$$\frac{1}{m^2 + 1} \begin{bmatrix} 1 & m \\ m & m^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

## 3.2 Normalization

Normalization is a preprocessing step to make sure that data are centered at the origin and that each component has unit variance. This has a variety of uses; for example, it is used in photography to fix poor contrast and lighting issues. Greyscale bitmap images are stored as two-dimensional arrays of integers from 0 (black) to 255 (white). Fill in the function `normalize`, which takes in a two-dimensional NumPy integer array of size $32 \times 32$ and returns a normalized version. For each pixel $p$, the following formula gives the corresponding output pixel $p'$, where max and min are the maximum and minimum brightness of pixels in the image, respectively:

$$p' = \frac{255}{\max - \min}(p - \min)$$

## 3.3 Normalization Part II (Extra Credit)

Notice that when using that linear formula for normalization, there are a lot of extreme values. Photos tend to look better with a well-defined midrange and only some shadows and highlights. This sigmoidal nonlinear normalization scheme uses an exponential function to map more pixels to mid-range intensities, and very few to either extreme. Fill in the function `sigmoid_normalize`, which takes in a two-dimensional NumPy integer array of size $32 \times 32$ and a variance parameter $a$, and returns a normalized version. For each pixel $p$, the following formula gives the corresponding output pixel $p'$:

$$p' = 255(1 + e^{-a^{-1}(p-128)})^{-1}$$

**Hint:** A normal Python function applied to a NumPy array simply applies that function to each element of the array.