Programming HW #2
Due: 23:59 June 25

Please use the provided starter file for exercise 0. The provided notes on computing a centroid may be useful for completing exercises 1 and 2.

Aaron and the TAs will be in the labs starting at 20:30 June 25 to go over your solutions. For exercises 2 and 3, we will ask you to demonstrate your code using the provided test image.

Starter files are available here: https://homes.cs.washington.edu/~awb/hw2_starter.zip

0. Fruit Ninja Academy

Congratulations! The time has come for your final exam at the Fruit Ninja Academy. The array of fruits you will use to demonstrate your slicing prowess has already been assembled in the starter code. You will now face a series of slicing challenges. Extended the provided code to accomplish the following tasks, printing out the result of each task. The expected number of commands (not necessarily including printing) and expected output are given for each task.

a. Extract the bottom right element in one command.
```
Tomato
```

b. Extract the inner 2x2 square in one command.
```
[['Grapefruit' 'Kumquat']
 ['Orange' 'Tangerine']]
```

c. Extract the first and third rows in one command.
```
[['Apple' 'Banana' 'Blueberry' 'Cherry']
 ['Nectarine' 'Orange' 'Tangerine' 'Pomegranate']]
```

d. Extract the inner 2x2 square flipped vertically and horizontally in one command.
```
[['Tangerine' 'Orange']
 ['Kumquat' 'Grapefruit']]
```

e. Swap the first and last columns in three commands. Hint: make a copy of an array using the `copy()` method.
```
[['Cherry' 'Banana' 'Blueberry' 'Apple']
 ['Mango' 'Grapefruit' 'Kumquat' 'Coconut']
 ['Pomegranate' 'Orange' 'Tangerine' 'Nectarine']
 ['Tomato' 'Raspberry' 'Strawberry' 'Lemon']]
```

f. Replace every element with the string `"SLICED!"` in one command.
```
[['SLICED!' 'SLICED!' 'SLICED!' 'SLICED!']
 ['SLICED!' 'SLICED!' 'SLICED!' 'SLICED!']
 ['SLICED!' 'SLICED!' 'SLICED!' 'SLICED!']
 ['SLICED!' 'SLICED!' 'SLICED!' 'SLICED!']]
```

1. Centroid Hand Calculation

Below is a table of the intensity of a set of 25 pixels derived from an image of 1959Lick taken on June 30,2011.

0	33	21	33	8
0	56	51	53	26
23	120	149	73	18
55	101	116	50	16
11	78	26	2	10

Clearly the center pixel does not correspond to the center of the star! Where is it?

- Obtain the centroid of the above data and report the uncertainty. You may choose to use a calculator to do this or use an Excel spreadsheet.

Assuming that the center coordinate is (2,2) and (0,0) is in the upper left, the centroid should be calculated as x = 1.824 and y = 2.191 with an uncertainty (standard deviation of the mean) of 0.0311 for x and 0.0328 for y.

2. Centroid Calculation

The overall goal of this exercise is to write a program that takes as input one of your asteroid images and returns the centroid of the asteroid.

a. Write a program in python that finds the centroid of the data given in the above table. Check your result with what you obtained from your hand calculation.

b. Now you will write a function to find the centroid of an object in an actual image. The function should take as parameters the **filename** of a FITS image, the **x-coordinate** of the asteroid, the **y-coordinate** of the asteroid, and a pixel **radius** to use in computer the centroid (four parameters in all). This function should:

- Reads in the FITS image with the **filename** given
- Performs a weighted average of the pixel coordinates centered on the given **x-y-coordinates**. These pixels should include the specified pixel as well as **radius** pixels in each direction. For example, a **radius** of 1 would result in a centroid of nine pixels centered on the **x-y-coordinates**.
- Compute the uncertainty of the centroid in terms of the standard deviation of the mean in x and y
- Return the x and y of the computed centroid and the x and y uncertainty (four values in all)

You will use this function as part of your data reduction. It will be your primary way to determine the uncertainty in your centroids. For the provided sampleimage.fits, the asteroid 1951 Lick is located at 351,154. On this data, your function should return values very close to these:

Centroid: 350.9958 153.9956
Uncertainty (std. dev. of the mean) in x,y: 0.005254018 0.005249733

3. Differential Photometry

Write a Python function to compute the magnitude of the asteroid in an image using a star of known magnitude. This function should take as parameters the **filename** of a FITS image, the **x- and y-coordinates of a relatively isolated star**, the **width of the star** in pixels, the **magnitude** of that star, the **x- and y-coordinates of the asteroid**, the **width of the asteroid** in pixels, and the **x- and y-coordinates of a nearby blank portion of the image** (ten parameters in all). This function should:

a. Extract a box (2D array) of the **width of the star** centered on the star's **x-y-coordinates** from the image. Sum the pixel counts of all pixels inside the box. This count is 'star+sky.'
b. Extract a 3-by-3 box of blank sky centered at the given **x-y-coordinates** for blank sky. Determine the average pixel count of all pixels inside that box. This is 'avgSky.'
c. We'll refer to the pixel counts for just the star as the 'Signal,' which is given by

$$\text{Signal} = \text{'star+sky'} - (\text{avgSky} * N_{ap})$$

where N_{ap} is the number of pixels in the 'star+sky' aperture.

The general formula for determining a magnitude is:

$$\text{mag} = -2.5 * \log_{10}(\text{Signal}) + \text{const}$$

(It's really the flux of the star that is inserted into the magnitude formula, but here we assume that pixel count is directly proportional to flux and the constant of proportionality is absorbed into the constant term.) Since we know the magnitude of the star, we can determine the constant value.
d. Repeat the process for your asteroid in the image. Now that you know the constant, you can calculate its magnitude.
e. *[optional]* Instead of using a square aperture, it is more accurate to use a circular aperture around the star. Change your function so that instead of widths, it takes **parameters for radii of data values around the star and asteroid** as well as a **parameter for the width of a concentric annulus for background determination** (instead of a location of blank sky). There should be a few pixel gap between the aperture and the inner radius of the annulus. Use one of the methods below.
    i. *The pixel center must be within the aperture to be counted*
    ii. *The entire pixel must be within the aperture to be counted (need to test all four corners of the pixel)*
    iii. *The entire pixel must be outside the aperture to be rejected (need to test all four corners of the pixel)*
    iv. *Fractional-pixel method (more accurate): for pixels on the edge of the aperture, determine what percentage of the pixel area falls within the aperture. That same percentage of the pixel count will contribute to the 'star+sky' count and N_{ap} will cease to be an integer value.*

For the provided sampleimage.fits, the asteroid 1951 Lick is 3 pixels wide and is located at 351,154. The star at 173,342 is 5 pixels wide with a magnitude 15.26. The star at 355,285 is five pixels wide with a magnitude 16.11. Note that the widths given here are the total width of the object in pixels, not the radius of the object.

The final output of your program should be close to (using a 3-by-3 box centered at 200,200 for the blank sky):

Asteroid magnitude: 18.5008 (using first star to predict asteroid)
Asteroid magnitude: 18.7534 (with second star to predict asteroid)