Pseudo-code

Trying to remember all of the exact syntax of various python statements can often get in the way of figuring out how to set up a program. Knowing whether or not to use a for loop is a separate problem from knowing the exact syntax of a for loop. Pseudo-code uses a mixture of code and plain English to represent a program in a way that avoids any concerns about syntax, but still makes it clear how the program is designed to work.

As an example, lets consider the function from the first lab that took in an array of numbers and returned the sum of these numbers. The exact python code was:

```
def sum_func(array_like):
    out_sum = 0
    for i in array_like:
        out_sum += i
    return out_sum
```

Converting to pseudo-code this becomes:

```
sum_func(an array of numbers)
    Set initial sum to 0
    For each element in input array of numbers:
        Add element to sum
    Return sum
```

Pseudo-code uses plain English to describe what the different parts of the function are meant to do. Pseudo-code doesn't worry about exact variable names, or exact variable declarations, but does include the for loop and the return statement. From this pseudo-code you can see that you will need a variable that contains the sum, and you need to know the syntax for looping over the elements in a list.

Now lets consider an example of going in the other direction (converting from pseudo code to actual code). In the first lab you wrote a function that would go through a list of students and their grades and sort the students as either proficient or struggling. The pseudo-code version of this algorithm looks like:

```
For student and grade in class list
    If grade is greater than average grade
        Add student to proficient student list
    Else
        Add student to struggling student list
```

This pseudo-code makes it clear that we need to loop over every student in the list, and compare the grade of that student to the average grade for the class. Based on this comparison the student is either added to the proficient student list of the struggling student list.

From this pseudo-code you can imagine how to construct the actual piece of code. You will need lists for proficient students and struggling students, and you need to be able to add new entries to each of these lists. You will need to know the average grade for the class, as well as have a class list that contains names and grades (these may be defined earlier in the code). You will also need to figure out how to loop over the class list in a way that gives you access to both the grade and the student name. The exact python code for completing this task was:

```
proficient_students = []
struggling_students = []
for i in class_grades:
    if float(i[1]) > mean_score:
        proficient_students.append(i)
    else:
        struggling_students.append(i)
```

Where `class_grades` was a dictionary containing the names and grades for all of the students and `mean_score` was the average score. In this way you can use pseudo-code to lay out the basic structure of a program, and start to ask questions about how it will work (e.g., what if the array of numbers from the first example was actual a list of numbers as strings?), without worrying about syntax.

There are no formal rules for pseudo-code, or for how detailed to be when writing pseudo-code. Being too general isn't helpful (replacing the second example with `sort the students based on their grade` is not very helpful) but include too much detail and you end up just writing the function. Remember that pseudo-code is best viewed as a tool to help you think about how a piece of code is structured, without worrying about the exact syntax. Include as much, or as little, detail as is helpful for you to understand how the code operates.

Q: The following set of code (from the third lab) reads in and plots the positions of galaxies in Abell 0958.

```
abell_data = np.loadtxt('Abell0958.csv',delimiter=',',skiprows=2)
ra = abell_data[:,1]
dec = abell_data[:,2]

plt.plot(ra,dec,marker='o',color='k',linestyle='')
plt.xlabel('Right Ascension')
plt.ylabel('Declination')
plt.title('Abell 0958')
plt.axvline(np.mean(ra),alpha=.5)
plt.axhline(np.mean(dec),alpha=.5)
```

In the space below, re-write this code as pseudo-code:

Q: The following set of code (also from the third lab), reads in parallaxes from a file and calculates the average distance based on these parallaxes

```
def calc_avg_dist(file):
    data = np.loadtxt(file,delimiter=‘,’,skiprows=1,dtype=‘str’)
    parallax = []
    for x in data[:,5]:
        parallax.append(float(x))
    parallax = np.array(parallax)
    distance = 1000/parallax
    return np.mean(distance)
```

In the space below, re-write this code as pseudo-code:

Q: Write pseudo-code to take a series of images of a science target in different filters. The inputs for this code will be the science target, a list of the filters, a list of the exposure times for each filter, and a list of the number of images to take in each filter. The code will move the telescope to the target, checking to make sure it is above the horizon, and then command the camera to take the requested images.