HOMEWORK 0: PYTHON BOOTCAMP

Due electronically February 9 at 11:40 am

1. This assignment exercises your Python programming skills on a set of problems that do not have physical significance, but employ computational concepts that we will use throughout our study of computational materials science. Make sure you download all assignment files before beginning. You will need:

- **env_cross_platform.yaml**: conda environment we will be using in the class
- **HW0.py**: python file where you will implement solutions to the challenges
- **helper.py**: contains an extra function that will be used in the assignment.
- **data.npy**: contains data for one of the questions.

(a) Set up software environment for use in this class. After downloading Anaconda (or Miniconda if you prefer) and the assignment files, configure a conda environment using the **env_cross_platform.yaml** file. This ensures all package versions are consistent. Once your environment is installed, activate the environment and run "`conda env export > my_env.yml`," Turn in the resulting text file **my_env.yml**.

The file **HW0.py** contains instructions for the programming problems in parts (b) – (d). There are 3 problems, and each one has its own function to implement. The arguments are already listed, and the steps to implement and outputs are described in the docstring of each function. Turn in your completed **HW0.py** file with your solutions filled in; there is no need to turn in the input or output files you used.

(b) Implement function **q1** in **HW0.py**. This question presents tasks that involve reading from/writing to files, basic array manipulation, working with random number generators, and loops. Note: With Python, it is good practice to implement as many operations with vectorized functions instead of loops, when possible. If you do this, you will notice considerable speedup of your code on large data sets.

(c) Implement function **q2** in **HW0.py**. Here, you write a short pattern generator to practice importing Python modules, loops and conditional statements, and data visualization. Note: there is a function in **helper.py** that is needed to implement this. You should import **helper.py** directly, *do not* simply copy/paste the function into the code.

(d) Implement function **q3** in **HW0.py**. By writing a simple finite difference routine to compute a numerical derivative, you will work with functions as objects, basic numerical computations, and plotting.

```
Grading: 4 points per problem:
0 = no attempt or code errors out
1 = code runs and produces some output, but majority is incorrect or missing
2 = code runs and produces all required outputs, but many are incorrect
3 = code runs, produces all required outputs, and most are correct
4 = code runs, produces all required outputs, and [almost] all are correct
```