Data Analysis in Geophysics (CERI 7104/8104)
Homework 1 – Due 9/22/17

This homework involves programming in Python. There are two problems, each of which requires you to fill in some code. I have defined the functions to work with a test function, so that you and I can both check if your code is working correctly. I have provided some test cases for each problem; I will use these test cases, plus additional tests, when grading your homework. Points will also be awarded for code clarity, robustness, and documentation.

1. Write a Python function `newton_poly(p, x0, tol, nmax)` to implement Newton's Method to find a root of a polynomial of arbitrary order. The function should take a list representation of a polynomial `p`, an initial guess (float) for the root `x0`, a tolerance `tol` (float) that determines when the algorithm has converged, and a maximum number of iterations `nmax` (integer). The function should return the root and the number of iterations required to converge (recall that you can return multiple values from a function if you separate them with commas in the `return` statement).

   Newton's Method can be described as follows:

   (a) Evaluate the polynomial f using the guess x_i. If the absolute value of the polynomial is less than the tolerance, the guess is good enough and you can exit the loop.

   (b) If the polynomial value is larger than the tolerance, update the guess using $x_{i+1} = x_i - f(x_i)/f'(x_i)$. If the derivative $f'(x_i) = 0$, raise an exception (see below). If the absolute value of $x_{i+1} - x_i$ is less than the tolerance, then the guess is good enough and you can exit.

   (c) Repeat using x_{i+1} as the new guess.

   In your implementation, you will define two additional functions `poly_eval` and `poly_der` to evaluate the polynomial and its derivative. These functions should take a list representation of a polynomial and a float representing the point at which the polynomial or derivative is evaluated, and return a float for the value or derivative. For the sake of consistency, the list representation of your polynomial should contain the coefficients for each term with the largest order first. Thus, for example, the polynomial $f(x) = 3x^3 + x - 4$ is represented by the list `p = [3., 0., 1., -4.]`.

   If the function reaches the maximum number of iterations without converging, or a zero derivative, your function should throw an error message using `raise ValueError`. Python uses `raise` to denote exceptions (also known as errors). There are many types of built-in exceptions in Python, and `ValueError` is appropriate for these cases.

All functions must contain documentation strings describing the purpose of the function, and your code must be adequately commented. Also, you should put assertions into each function where they might be useful.

Please use the included file "hw1_newton.py" that includes skeleton code and code to run the tests for you. You will need to fill in code for all three functions and add documentation and appropriate assertion statements. I have provided four test cases that you can use to verify that your code is working correctly. I will use these tests, in addition to four additional tests not included here, in grading your assignment (with additional points for code clarity, robustness, and adequate documentation). To run the tests on your code, simply run the file as a script, and it will print the test results to the screen. You are also welcome to modify the code to run additional tests (there is no need to remove any additional tests that you add).

2. Write a Python function `catalog_hist(catalog, nbins, minmag)` to make a histogram of an earthquake catalog. The catalog will be represented by a list, with each list entry being another list containing the event time (written in decimal years, so an event occurring at midnight on January 1, 2017, would be 2017.0) and the magnitude. The function should have three inputs: the list representation of the catalog (required), the number of bins `nbins` (integer, default value of 20), and the minimum magnitude `minmag` to use in the analysis (float, default is `None`, meaning that all events should be included).

   Your code needs to determine the start times of each bin, and the number of events with magnitudes above the threshhold in that time bin. The start of the first bin should be midnight on January 1 of the first year represented in the catalog and the end of the last bin should be midnight on January 1 following the latest year represented in the catalog. You can assume the events in the catalog are sorted in time. The function should return two lists, one that is a list of floats of length `nbins` indicating the start time of each bin, and a list of integers of length `nbins` indicating the number of events in each bin.

   I have provided "hw1_catalog.py" containing skeleton code and some tests that will be run if you execute the file as a script. If you wish to do additional tests, I also have provided a function to read a catalog from a text file in one of two formats: (1) a file that is already in the desired format, and (2) a file that includes the full date and time in a `datetime` string, a format used in the ANSS catalog and available as a module in Python. You should make sure your program works on the catalogs that I have provided: "newmadridcatalog.txt" is a catalog for the New Madrid Seismic Zone, and "pagercatalog.txt" is a 100 year global earthquake catalog, and I have provided some additional code for testing your results. Again, the provided tests will count towards your grade, and I will run additional tests not included here.