Lab 3

Coding standards: use appropriate names for your variables, functions, etc. Write docstrings as always.

For Beginners

Project 1.

Write a program that evaluates the function

$$f(x) = \frac{1}{\pi(1 + x^2)}$$

For 401 values of x equally spaced between -4.0 and 4.0 *inclusive*. Use lists and one or two for loops to accomplish this. Use matplotlib to plot the function. Use variables for the starting and ending values of x and the number of values.

Project 2.

Change Project 1 to use NumPy arrays but continue to fill the arrays using loops. Write a function to evaluate f(x) for any given real value of x and call it each time through your loop. Plot as before.

Project 3

Change Project 2 to use NumPy built-ins like arange or linspace to generate the values for x. Use your function as a ufunc; that is, pass it the entire x array and obtain the entire result array. This should enable you to eliminate all the for loops. Plot as before.

For Intermediates

Project 4.

Write a Christmas Bird Count program as in Project 6 from yesterday as follows:
1. Change lists to NumPy arrays.
2. Use try/except to verify that the specified file exists before you open it.
3. Use the NumPy built-in function loadtxt to read the data, then use NumPy max, mean, std, and argmax to compute the quantities obtained manually yesterday.
4. Use Matplotlib to plot number versus year.

You do not need to write any functions of your own for this project, just use the built-in NumPy and Matplotlib functions.

Project 5.

Repeat Project 4 but use Pandas to read the data into a dataframe. Use Pandas to obtain the statistics as in Project 4. Use Pandas to plot the number versus year. After studying the tutorial referenced in the notes, create a two-dimensional NumPy array from the dataframe columns then extract a smaller 2-d array containing the year and number data. Plot this using appropriate slicing.

For Experts

Project 6.

A. Download the file wm1deg.grid from the Collab site for this assignment (Resources/Data). This is a topographic dataset for the entire Earth at a resolution of 1 deg by 1 deg. Each row is a line of latitude; the entries in the row are the elevations at each longitude along that value of latitude. Elevations are represented as deviations from sea level (plus or minus) in meters, to the nearest meter (i.e. no fractional parts). The starting latitude is 89.5 N (positive) and the starting longitude is 0.5 E. Remember that latitudes run from -90 to 90 while longitudes run either -180 to 180 or 0 to 360.

Your program should read the name of the topographic dataset from the command line. Do not hardcode it or ask for user input.

1. Use loadtxt to read in the data and use Matplotlib to make a contour plot of it. The standard for maps is that north is at the top of the page and east is to the right. Does your picture conform to this? You will need to make one-dimensional arrays for lat and long that are of the appropriate size and whose values you compute from the starting points and the resolution. You can use these as the x and y arrays for your contour plots.

Find the locations of the maximum and minimum elevations. Plot these points on your final contour plot. Are they where you think they should be?

2. write a function to compute the total surface area of the Earth that satisfies some criterion. The element of area of the surface of a sphere is given by $dA=R^2\cos(\theta)d\theta d\phi$. (Cosine rather than sine because we measure latitudes from the equator and not the pole.) The radius of the Earth is 6378.1 km. Theta θ is latitude and phi ϕ is longitude. To compute the area, we sum up all the (approximate) area

elements. The $d\theta$ is given by the latitudinal resolution converted to radians and the $d\phi$ by the longitudinal resolution in radians. Remember that latitude must be in radians when you compute the cosine. NumPy has a function for this purpose, as does the math module. This function should take as its arguments the latitude array, the latitude resolution, the longitude resolution, and a mask array that describes the criterion. You can check your code by ignoring topographic information and making sure you get 4π (for R=1). Watch your units.

Use this function to compute the fraction of the surface area of the Earth that is ocean.

3. Write another function to compute the total volume that satisfies some criterion specified by a mask array. This is very similar to computing surface area but now we must use topographical information as well for the third dimension, which is the length of the column (positive, please). The volume element to be summed is $dV=zdA$.

Use this function to compute the volume of the ocean. You may find that your fraction won't be all that accurate but it shouldn't be wacky, so be sure to sanity-check your result. (It's surprisingly close considering the coarse resolution.) Print your results with some explanation of what each number is. Watch your units.

You should put your subprograms that are related to geographical manipulations or computations into a module called geo.py. Functions related to reading and displaying data may go into your main program or you may create a module for that.

B. Use Basemap to add coastline figures to your contour plot of the topography.

C. If you still have time, write a function that extracts a specified contiguous region from the data. NumPy has functions to help with this, such as nonzero. As an application, the boundaries of North America are, very roughly, 12 degrees N to 66.5 degrees N and 130 degrees W to 60 degrees W. Don't forget to convert the longitude to whatever numerical system you are using. Pull out the data for this rectangle and make a contour map of it. Add geographical and political boundaries with Basemap.