Lab 7

We are now switching language to a compiled language, Fortran. For the most part you will be redoing projects you have already done. This will enable you to concentrate on writing the language and practicing coding rather than working out the algorithms.

Coding Style
Instead of a documentation string, we have a documentation block at the top. For subprograms it should include a short description of input and output parameters.

Always use implicit none. Always choose good variable names.

Log in to Rivanna (the HPC system at UVa) with FastX. Start a terminal. Either load the geany module and start it, or use any editor you prefer.

Project 1.

Go through the notes and type in the complete examples. Save each one under some appropriate name ending in `.f90`. Compile and run the programs. Make a few changes to see what happens.

Project 2.

Translate as many of your projects from Lab 1 into Fortran as you can. Do not feel pressure to rush through the projects; it is better to do a good job on a few than a sloppy job on several. Especially if you are beginner, work slowly and carefully.

For Python projects that used lists, you can use a large array instead. It is possible to mimic Python lists by resizing arrays (NumPy does similar things) but that requires facility with allocating and deallocating. Just be sure to set up an array of sufficient size.

Project 3.

Download the CPI.csv file from Resources/Data. This file contains the annual Consumer Price Index for all years for which it is available, which I laboriously compiled by running the online calculator at the Bureau of Labor Statistics 101 times. The actual baseline is 1980 but I set my baseline at the beginning of the data, in the year 1913.

Write a program that will read from the command line the name of a file. Read this file into your program. Request from the user a year (use non-advancing IO so it will behave similarly to raw_input in Python). Once again, do not assume you know in advance how many lines the file contains. You should be able to reuse most of the subprogram you wrote in Project 2 to count the number of lines in the file.

Check that you have enough command line input. You are more limited in exception handling in Fortran than in Python, but stop if you don't have enough command line input. Use the inquire statement to check that the file exists before you attempt to open it. Similar to Python you can add a message to stop, e.g. `stop "Invalid data".`

The ratio of any two years is an estimate of the change in the cost of living. Compute the change in the cost of living from the year you specify to 2013. Print it out neatly with some informative text. Do not print more than 2 decimal places since this is all the data contain. Test your program using the year 1954.

In 1954 a color television cost $1295. From your result how much would that be in 2013 dollars?

Project 4.

The derivative of the CPI will give us a crude estimate of the inflation rate.
The CPI data I have provided is annual so let us use the formula

$$I = \frac{CPI(yr+1) - CPI(yr)}{12}$$

Notice that you always have one less year of inflation than of CPIs. Using code from Project 2, compute the inflation array and write it to a file inflation.csv

Plot the CPI and the inflation using Python.

Project 5.

"So you think you know Fortran"

I was given a program that has code to generate a format string dynamically. In particular, it can print items that might be arrays, with the repeat value generated automatically. For example, given n1=5, n2=1, n3=3 and a pattern
'(#e15.8,#i5,#f8.2)'
the result would be
'(5e15.8,1i5,3f8.2)'

However, it didn't work if any of the n1, n2, n3 variables were two digits. The algorithm was convoluted and hard to understand. It did work for two digits if I used two hash marks, e.g. ##e15.8, but that requires hand-editing the several output files to find every place I wanted to write out more than 9 array elements.

The author of the original code didn't use any character/string functions other than substrings. I am convinced that this would be implemented more generally with better use of strings. Your task is to come up with a way to do this. Be sure to test your program. If you have time and are particularly clever, come up with a way to handle a 0 (i.e. I want to skip printing something).