Lab 1

Coding Style
Write a documentation block at the top of each program unit. 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 or MobaXterm (Windows) or ssh -Y (Mac Terminal with XQuartz installed). Start a terminal. Either load the geany module and start it, or use any editor you prefer. Note you will need to type vim rather than vi to use features like syntax coloring.

To use Geany type
module load geany

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.

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*. Put the values into an array variable x. Use variables for the starting and ending values of x and the number of values. Use an array operation to fill a variable y.

Write a function to evaluate f(x) for any given real (scalar) value of x and call it each time through your loop

Make sure the function satisfies the requirement for an elemental function.
Eliminate the loop evaluation by changing your function to an elemental.

Project 3.

Download the CPI.csv file. 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). Once again, do not assume you know in advance how many lines the file contains.

Check that you have enough command line input. Stop with a message 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. 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 anything you know (Excel, Python, Matlab, etc.).

Project 5.

Write a program to compute the day of the week for any date of the Gregorian calendar.
Here is the formula:
W=(C+Y+L+M+D) mod 7
Y is the last two digits of the actual year and D is the actual day.

You need to obtain the value of C from the following rule for the years:
If year is in the 1400s, 1800s, 2200s, C=2
If year is in the 1500s, 1900s, 2300s, C=0
If year is in the 1600s, 2000s, 2400s, C=5
If year is in the 1700s, 2100s, 2500s, C=4
Months are numbered from 1 in the usual way, but (from January) M is 0, 3, 3, 6, 1, 4, 6, 2, 5, 0, 3, 5
The only tricky part of this algorithm is L, the number of *leap days* that have occurred since the beginning of the century of the given date. To obtain this:
1. Integer divide the last two digits of the year by 4 to obtain the number of "ordinary" leap years in the century up to that year, not counting the century year itself if applicable.
2. Obtain the remainder of the two digits and 4. If it is not a century year and the remainder is 0 the year is a leap year, otherwise it is not. If the year itself is a century year see Step 3.
3. If the century (1400, 1500, etc.) was evenly divisible by 400 then the century year is a leap year, otherwise it is not. Thus 2000 was a leap year but 1900 was not. So add 1 for centuries divisible by 400 and 0 otherwise.
4. If your date is January 1-February 29 of a leap year, subtract 1.
Try to devise a method to obtain the last two digits on your own.
Print the day of the week as a word (Monday, Tuesday, etc.). Remember that Sunday is the first day of the week and it will be counted as 0 in this algorithm.
Hint: use an appropriate data structure to store the names of the days and the values of M.
Test your program first with your own birth date. Then test with the following dates:
Today's date
December 25, 1642 (Note: this is Newton's birth date in the Julian calendar, but use it as a Gregorian date)
October 12, 1492
January 20, 2000
December 11, 2117

Extra Challenge for Experts

"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).