CSCI 2202 COMPUTER MODELLING FOR SCIENTISTS LAB 4
LISTS, STRINGS & FLOATS

1. Write a program that asks the user to input three numbers, and prints the largest and smallest values. *e.g.* For `input: 6.7, 74, 9` the output is: `The largest number input is 74 and the smallest 6.7`

2. A leap year is a year (1582 and after) divisible by 4, *unless* it is a century year. A century year is a leap year **if** it is divisible by 400.
   Hence, 1900 is not a leap year, while 2000 is.
   Write a program `isLeapYear.py` that takes a year as input and prints if the year input is or is not a leap year.
   `input: 2020 output: 2020 is a leap year`
   `input: 1900 output: 1900 is not a leap year`

3. In the lecture notes, Henon's algorithm for finding the square root was outlined. It is reproduced below. Write a program to implement Henon's algorithm to find the square root of a number, that asks the user to input a number x to find the square root of. Floats should not be compared exactly (think why) instead, you should produce an answer correct to within `1.e-5` (this is *close enough*). *i.e.* stop the loop when $abs(g * g - x) < 1.e - 5$
   Once the rogram is working, compute the square roots of $xList = [10, 20, \dots 90]$. Compare the values to the values obtained using `math.sqrt()`. Henon's Algorithm to find the square root y of a number x:

   (a) Start with a guess: g
   (b) Test: Is $g * g$ close (*enough*) to x?
   (c) If YES then DONE. Report: $y = g$
   (d) Else update guess: $g_{new} = \frac{1}{2}\left(g + \frac{x}{g}\right)$
   (e) $g = g_{new}$
   (f) goto 2.

4. The value of π is equal to the following infinite series:

$$\pi = 4 \cdot \left(\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots\right)$$

   (a) While we cannot compute the entire infinite series, we can get an approximation to the value by using the first n terms. Allow the user to input n. Name you program `piSeries.py` Experiment with $n = 10,\ 100,\ 1000,\ 10000$.

(b) As you can see, the value gets closer to the actual value of π as the number of terms increases. Make a copy of the program you made fro the exercise above and save it as `piTolerance.py`. Modify the program so that you keep adding terms such that the computation with n terms and the computation with $n+1$ differs by less than $10e-5$ (1 part in 10000). Your program should print out the estimate of π and the number of terms used to obtain the value.