Quantitative Economics with Python
ECON-UA 370
Assignment #2

Due: 16th February (Beginning of Class)

Answer each of the questions below. Please provide clear explanations for your reasoning and include any code that you write to support your answer.

1. Write two functions (max) and (min) that take a collection of values and returns the maximum and minimum values respectively. You are **not** allowed to use the inbuilt python `max` or `min` functions. But you can use them to check your functions for accuracy.

$$\max([1,2,3]) = 3$$
$$\min([1,2,3]) = 1$$

a. Write each function using Python. Use comments to explain your code.
b. Does this work over different data types? What happens if you pass your function the following data `[1,'a',2,3,4]`
c. How might you make your function more robust?

2. Write Python functions that compute the **mean**, **median**, **range**, and **variance** for a collection of values. State your assumptions and write your solution using the basic python data structures, conditional logic statements, and functions that we have discussed in class. You may **not** use any libraries such as **numpy** to develop your solution. You may however use them to compare and test your function. You may also use `math.sqrt()`. Use comments to explain your code.

You may find the following code using **numpy** useful to test your answers.

```
>>> import numpy as np
>>> #-Mean-#
>>> data = [3,6,1,3,6,8,9,2,2,2,1]
>>> np.mean(data)
3.9090909090909092
>>> #-Median-#
>>> np.median(data)
3.0
>>> #-Range-#
>>> np.max(data) - np.min(data)
8
>>> #-Variance-#
>>> np.var(data)
7.3553719008264471
```

3. Fibonacci numbers are integers that follow the sequence:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...$$

The first two numbers in this Fibonacci sequence are F_1=1 and F_2=1, with subsequent values being the sum of the previous two values.

a. Express this sequence using mathematical notation.
b. Write a Python function that uses **iteration** to compute a Fibonacci number and/or sequence given some integer n. State your assumptions and note below how the function should behave:

```
>>> fib(4)
3
>>> fib(4, show_sequence=True)
1, 1, 2, [3]
```

c. Write a Python function that uses **recursion** to compute a Fibonacci number and/or sequence given some integer n. The output of the function should be the same as in part b.

4. Computing *approximate* Square Roots using Newton's Method.

Given a real valued function with one variable x

$$x : f(x) = 0$$

one can use the following formula to obtain an approximation when solving for the roots of the function $f(x)$.

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

The process can be repeated to produce an improved approximation:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

So all you need is the function, its first derivative, an initial guess, and some iteration to obtain a solution. For further reading checkout: `https://en.wikipedia.org/wiki/Newton%27s_method#Description`

a. Use Newton's method to analytically solve for an approximation of the square root function $\sqrt{x}$. For $r \in \mathbb{R}^+$, consider

$$x^2 = r$$

Let

$$f(x) := x^2 - r$$

and use Newton's method to solve the equation $f(x) = 0$.

b. Use the solution from part (a) to write a Python function that computationally solves for the square root of any given number $r \in \mathbb{R}^+$. **Hint:** Think about precision. You may **not** use `math.sqrt()`.

Your function should behave something like:

```
>>> square_root(4)
2
>>> square_root(14)
3.7416
>>> square_root(1.4)
1.1832
```