Intro to Python **Math 342**

This is a brief introduction to the Python programming language. We will cover:

- How to define variables.
- How to use for and while-loops.
- How to create functions (both regular and lambda functions).
- How to import libraries.

The Babylonian Algorithm

The Babylonian algorithm is a way to compute the square root of a number a. It is a recursive algorithm. You start with a guess x_0 that should be close to the answer. Then each step, you calculate a new guess x_{n+1} based on the previous guess x_n. The formula is

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right).$$

1. To assign values to variables in Python, like to set a equal to 5, just type `a=5`.

2. There are two ways to create functions in Python, lambda functions and regular functions. Lambda functions are for simple functions that can be defined in one line. Here's how to input the Babylonian algorithm function as a lambda function.

   ```
   f = lambda x: 1/2*(x+a/x)
   ```

   Enter this function and then compute `f(2)` and `f(f(2))`.

3. Let $x = 5$, and then repeatedly calculate $x = f(x)$. Each time, use the command `print(x)` to check the new value of x. Notice, we don't worry about the subscripts on the x-variable.

4. To automate the process above, we can use a for-loop. Here is an example for-loop that adds the numbers from 0 to 9.

   ```
   total = 0
   for i in range(10):
       total = total + i
       print(i)
   print(total)
   ```

   Other languages use curly braces or keywords (like `end` in Matlab) to indicate which code is repeated in the loop. Python uses indentation instead. Most style guides recommend using four spaces to indent to make it clear which lines are inside the loop. Write a for-loop to repeat the Babylonian algorithm recursion `x=f(x)` ten times (starting with $x = 2$). What is the value of x at the end? What is the value at each step?

5. Python has a built-in square root function, but to use it you need to load the standard math library. Type this command to load the math library and compute the square root of 5.

```
import math
print(math.sqrt(5))
```

6. Python also has while-loops that repeat until some condition is no longer satisfied. We want to stop using the Babylonian algorithm when the new value $f(x)$ and the old value x are really close. We can check this using the absolute value function to find the distance between x and $f(x)$. Complete the following while-loop to run the Babylonian algorithm automatically.

```
while abs(f(x)-x) > 10**(-6):
    # What code should you put here to repeatedly update the value of x?
```

7. Now let's create a new function that can compute the square root of any positive number a, to any desired precision. To do this, use the following code:

```
def findSquareRoot(a,precision):
    # Insert your code here.  I recommend using a while-loop.
    # The last line of your function should return the answer.
    return x
```

Test your function by computing the following:

- `findSquareRoot(5,10**(-6))`
- `findSquareRoot(100,10**(-6))`