Complete the following two assignments, which are intended to give you some practice working with Python code. At the end of each assignment you will find instructions on what deliverables need to be uploaded to iLearn for grading.

Python code assignment 1:

Define the following variables:

1. an integer variable named qubit1 and assign it a value of 12
2. a boolean variable named boo and assign it a value of FALSE
3. a string called Bell and assign it a value of “Collapse to 1”; add a comment to your code indicating that this is a piece of text
4. a string called BellPair and assign it a value of “I also collapse to 1”
5. Define a list called my\_list containing (in order) the first string, the integer, the second string, and the Boolean
6. Print the second string from the list you just created without mentioning the name of the string (hint: you can access the string by its location in the list).
7. Add the number 137 to the end of the list
8. Print the final list in order, with each list item on a separate line (not all on one line of text)
9. Calculate and print out the difference between the last item on the list and the cosine of zero (hint: you may need to import a math package into your program)

**Deliverables:** Submit your Python program and its output to iLearn (you may use screen shots as needed).

Python Code Assignment 2:

This exercise should give you some idea of the reason why reducing a problem from exponential time complexity to quadratic or linear complexity represents such a significant time savings. We’ll look at variation on the chess board problem discussed in this week’s lecture. Consider the interactive Jupyter code example in your text book, section 0.1, on how to loop over a range of numbers (the 14th from the start of the section). The following example is given in the book:

For j in range(5):  
 print(j)

Modify this code to perform the following calculations, then run each calculation using interactive mode in your text book (or a similar Python tool of your choosing).

1. Print a list of all j values from 1 to 64. Note that you’ll have to modify the given code, which starts at j=0 by default. This result is an example of a linear progression of values.
2. Print a list of 64 values, starting with j=1 and doubling the value of j for each successive iteration
3. Print a list of 64 values, starting with j=1 and squaring the value of j for each successive iteration. This result is an example of a quadratic progression of values.
4. Print a list of 64 values, starting with j=1 and raising 2 to the power j for each successive iteration. Note that the Python code for A raised to the N power is A\*\*N. This result is an example of an exponential progression of values.
5. Compare the last value on each of the above four lists.   
   How many times larger is the 64th value of the exponential progression than the quadratic progression?   
   How many times large is the 64th value of the exponential progression than the linear progression?

**Deliverables**: Submit a copy of your code for items 1-4 and a screen shot of the results for steps 1-4 to iLearn. Submit your answers for the two questions in item 5 to iLearn.