Before you begin

- Install PyQt5 using conda
- Grab gui,py, slider.py, and random-graph.py from files
  - Check that they all run
- Take a look at the code, and see if you can figure out what the various pieces of it do. For the remainder of this lab, you should build on this code, since it contains all of the boilerplate code that you'll need.

Learning Objectives:

You should be able to do the following:

- Create a simple GUI using PyQt5
- Know how to set callbacks
- Know how the event manager communicates with your code

Thoughts, come back after you have read the problems:

- QSliders only deal with integers. You're going to have to do some simple math to make them work with floating point numbers, like in the example.
- Your interface doesn't have to look exactly like the ones we give you. It should have the same number of sliders, labeled in the same way, and the same buttons, but the layout and details can differ. We're really looking for the same functionality, rather than the exact same look.
- For the sine wave graph, you should utilize your SliderDisplay class to make things easy. Remember that SliderDisplay itself is a type of widget, and therefore instances of this widget can be added to a layout in the same way you'd add any other widget like a label or a slider.

Grading Checkpoints [16 points]

Check the footnotes for associated rubric items * . **Assignment Grading Checkpoints (12 points total + 1 extra credit)**

- [2 points] In gui.py, added a label and a slider (1 point), with the slider changing the label on movement (1 point).
- [5 points] SliderDisplay meets the specified criterion (each 1 point):
  - Sets and displays low/high bounds correctly
  - Correct number of ticks
  - Properly-interpolated values
  - Name and unit displayed
  - Maximum 3 decimals displayed
- [5 points] Sine wave graph meets the specified criterion (each 1 point):
  - Sliders with proper values shown on left
  - Graph updates when clicked
  - Sine wave in graph looks smooth with correct values from sliders
  - Graph has a title corresponding to the user input in the left area
  - Graph is consistently plotted in the range [0, 5] x [-5, 5]

*Indicates that the autograder will tell you if this problem is correct on our set of test cases and assign you credit.

- [1 points] **Extra credit:** Sine wave graph updates automatically when any of the sliders is moved.

Standard Grading Checkpoints (4 points total)

- [2 points] Code passes PEP8 checks with 10 errors max * .
- [2 points]Code passes TA-reviewed style checks for cleanliness, layout, readability, etc.

For the code check, we are mainly focused on your usage of encapsulation and proper, "non-hacky" ways to do things like set callbacks, update label/slider values, etc.

Problem 1 Edit gui.py

First, we'll get you started with adding basic elements to an interface. Download `gui.py` and look through it to understand what's going on. Then, do the following:

- Add a label above the quit button which initially displays the value zero.
- Below that, add a horizontal slider with a range from 0 to 10. When you drag the slider, it should change the value of the label.

Problem 2 Slider

Look at `slider.py`. This file has a class called `SliderDisplay`. By default, the Slider class inside of PyQt only has support for integer values. Our goal here is to implement a slider which can work on arbitrary ranges, like shown below:

Implement the SliderDisplay class in the file to create a slider which takes in a name ("Magnitude" in the above) and ranges from low to high with the corresponding number of ticks. It should also take in a string units (defaults to an empty string) which is displayed alongside the numeric reading (dB in the above example). You'll need to take care of the following:

- Saving down the inputs passed by the user as attributes
- Creating the display elements, adding them to the layout, and attaching any necessary callbacks to them

- Computing a correct value based on a linear interpolation and updating the label afterwards

Note that the `value()` method should only return a numeric value. If you want the code to update the label with the corresponding value, you should do that in a different method which calls value(). You should also make sure that the label displays a consistent number of decimal points (3 is fine) so that the size of the window doesn't jump all over the place if the label evaluates to a number like 3.100000000001.

Problem 3 Sine wave graph

For this problem, refer to the provided file `random_graph.py`, which plots 10 random values on a graph. Copy the code into a new file called `sine_graph.py`.

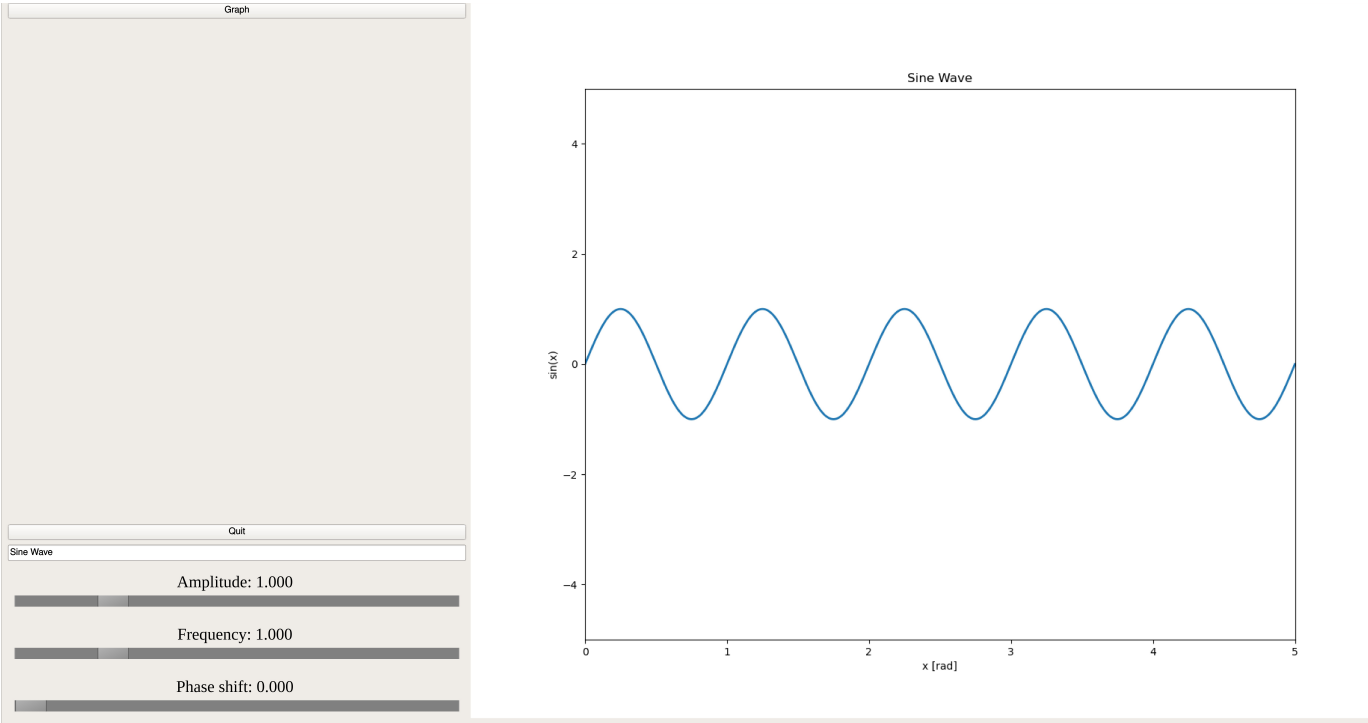
Create a GUI like the one shown below which plots a sine wave of the form

$$A\sin(2\omega\pi(x-\phi))$$

with the ranges on the parameter sliders as follows:

- Amplitude: 0 to 5
- Frequency: 0 to 5
- Phase shift: 0 to 1

It should also have a text box to enter the title of the plot shown, and a button that, when clicked, updates the graph. Regardless of the values chosen for the parameters, the plot should always be shown with axis ranges of `x=[0, 5]` and `y=[-5, 5]`.



Extra credit: Make it so that the graph automatically redraws as you move the 3 sliders, so you can continuously see the effect of changing the amplitude/frequency/phase shift.