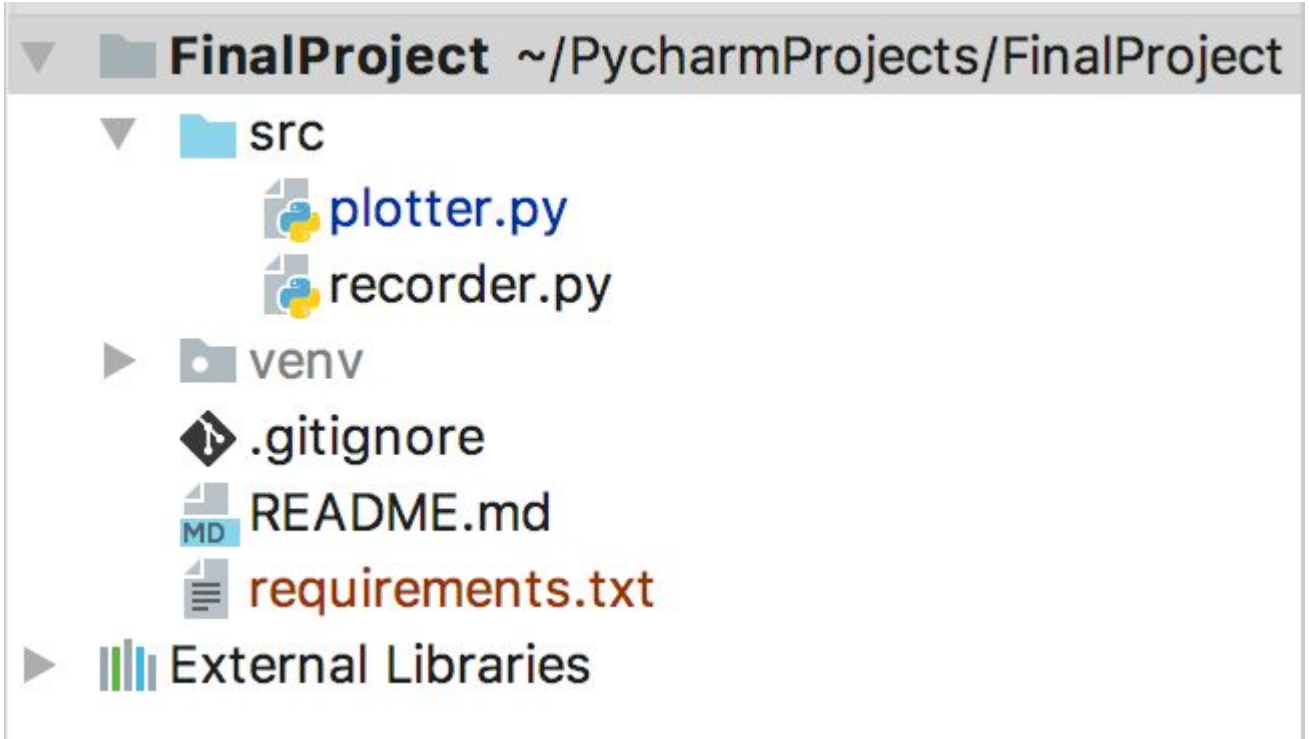
Audio Visualization

Project Goal

- Explore audio I/O
  - PyAudio
- Explore data visualization
  - PyQtGraph
- Explore Python front-end development
  - PyQt5

Project Structure

- Python 3.6
- Git



Recorder

```
import threading
import pyaudio
import atexit
import numpy


# On creation, records audio through the microphone
class Recorder(object):
```

Recorder

```
def __init__(self, rate=4000, chunksize=1024):
   # Initialize audio variables and pyaudio
   self.rate = rate
   self.chunksize = chunksize
   self.pyaudio_instance = pyaudio.PyAudio()
   self.stream = self.pyaudio_instance.open(format=pyaudio.paInt16,
                                            channels=1,
                                            rate=self.rate,
                                            input=True,
                                            frames_per_buffer=self.chunksize,
                                            stream_callback=self.new_frame)
   self.frames = []

   # Initialize multithreading routines
   self.lock = threading.Lock()
   self.stop = False
   atexit.register(self.close)
```

Recorder

```
# Callback method when a new audio frame is recorded
def new_frame(self, data, frame_count, time_info, status):
   data = numpy.fromstring(data, 'int16')
   with self.lock:
       self.frames.append(data)
       if self.stop:
           return None, pyaudio.paComplete
   return None, pyaudio.paContinue
```

Recorder

```
# Return filled audio frames, and re-initialize audio frames
def get_frames(self):
   with self.lock:
       frames = self.frames
       self.frames = []
       return frames
```

Recorder

```
def start(self):
   self.stream.start_stream()

# Stop recording, close the stream, terminate pyaudio
def close(self):
   with self.lock:
       self.stop = True
   self.stream.close()
   self.pyaudio_instance.terminate()
```

Plotter

```
from collections import deque
from pyqtgraph.Qt import QtGui, QtCore
import pyqtgraph
from recorder import Recorder
import audioop


# Plots sound graph from microphone
class Plotter(QtGui.QWidget):
    RMS_QUEUE_SIZE = 100  # RMS queue size in timer intervals
    TIMER_INTERVAL = 100  # Timer interval in milliseconds
```

Plotter

```
def __init__(self, width, height):
    QtGui.QWidget.__init__(self)

    # Initialize the user interface
    graphics_window = pyqtgraph.GraphicsWindow()
    layout = QtGui.QGridLayout()
    self.setLayout(layout)
    self.resize(width, height)
    self.setWindowTitle('PyAudio Demo')

    # Create plots
    pyqtgraph.setConfigOptions(antialias=True)
    oscilloscope_widget = pyqtgraph.PlotWidget()
    oscilloscope_curve = oscilloscope_widget.plot(pen='y')
    oscilloscope_widget.setXRange(0,1050)  # TODO Extract constants when measurement units are identified
    oscilloscope_widget.setYRange(-50000,50000)
    self.oscilloscope_curve = oscilloscope_curve

    volume_widget = pyqtgraph.PlotWidget()
    volume_curve = volume_widget.plot(pen='y')
    volume_widget.setYRange(0, 16000)
    volume_widget.setXRange(0, Plotter.RMS_QUEUE_SIZE)
    self.volume_curve = volume_curve
```

Plotter

```
# Create other widgets
layout.addWidget(oscilloscope_widget, 0, 0)
layout.addWidget(volume_widget,0, 1)

self.show()
timer = QtCore.QTimer()
timer.timeout.connect(self.handle_new_data)
timer.start(Plotter.TIMER_INTERVAL)
self.timer = timer

# Initialize the recorder
recorder = Recorder()
recorder.start()
self.recorder = recorder
self.rms_deque = deque(maxlen=Plotter.RMS_QUEUE_SIZE)
```

Plotter

```
# Handle callback data from recorder
def handle_new_data(self):
   frames = self.recorder.get_frames()
   if len(frames) > 0:
       current_frame = frames[-1]
       rms = audioop.rms(current_frame,2)
       self.rms_deque.append(rms)
       self.oscilloscope_curve.setData(current_frame)
       self.volume_curve.setData(list(self.rms_deque))
```

Plotter

```
# Start Qt event loop unless running in interactive mode or using pyside.
if __name__ == '__main__':
   import sys
   if (sys.flags.interactive != 1) or not hasattr(QtCore, 'PYQT_VERSION'):
       app = QtGui.QApplication(sys.argv)
       screen_rect = app.desktop().screenGeometry()
       window = Plotter(screen_rect.width(), screen_rect.height())
       sys.exit(app.exec_())
```