For question 1 Watch a video on YoutTube about the DX 21 and make an explicit connection to any of the concepts we have covered in the course. I watched a video called “Yamaha DX21 Basics - Intro & Special Features” from the resources page on the github. The Yamaha DX21 synthesizer was a significant advancement in music technology during the 1980s, as it was the first affordable synthesizer to utilize Yamaha's FM synthesis technology and MIDI communication. One interesting fact about the Yamaha DX21 synthesizer is that it was the first synthesizer to use the newly-developed Yamaha FB01 sound generator chip, which allowed for a more affordable implementation of the FM synthesis technology used in the DX7. The DX21’s affordability and versatility helped to democratize electronic music production, making it accessible to a wider range of musicians and producers. The DX21's impact on music technology is still felt today, as it paved the way for future advancements in synthesis and MIDI communication.

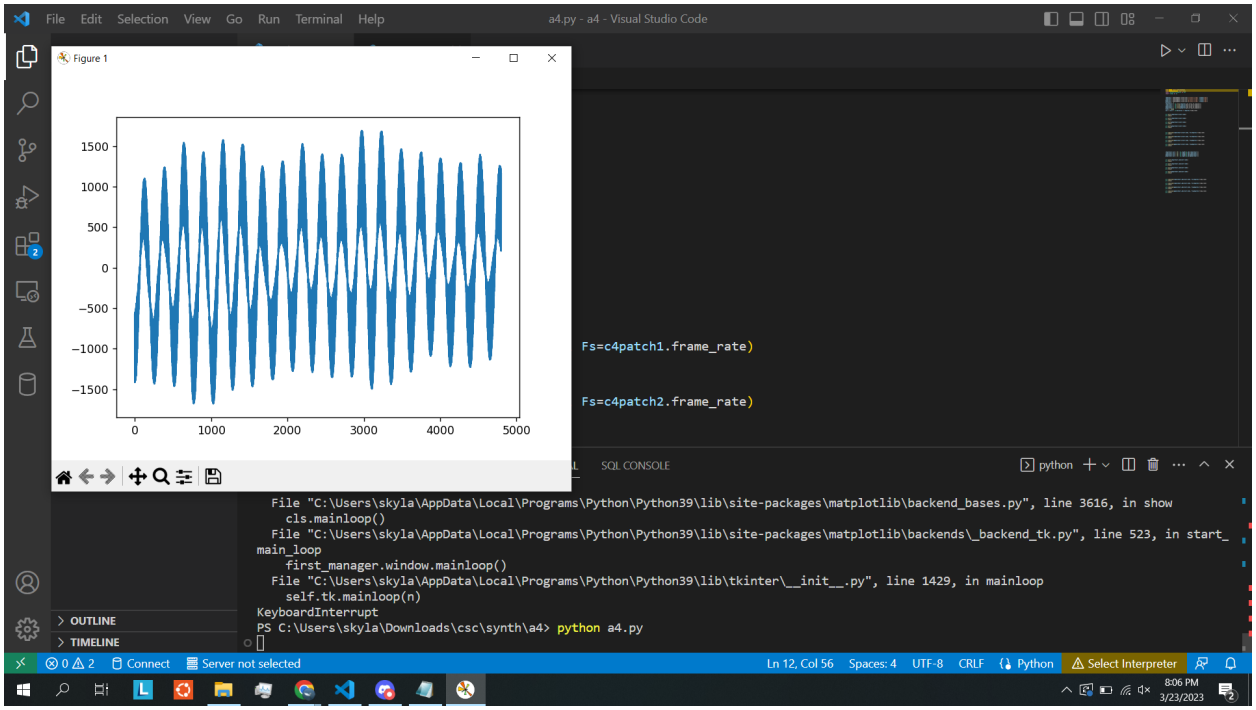
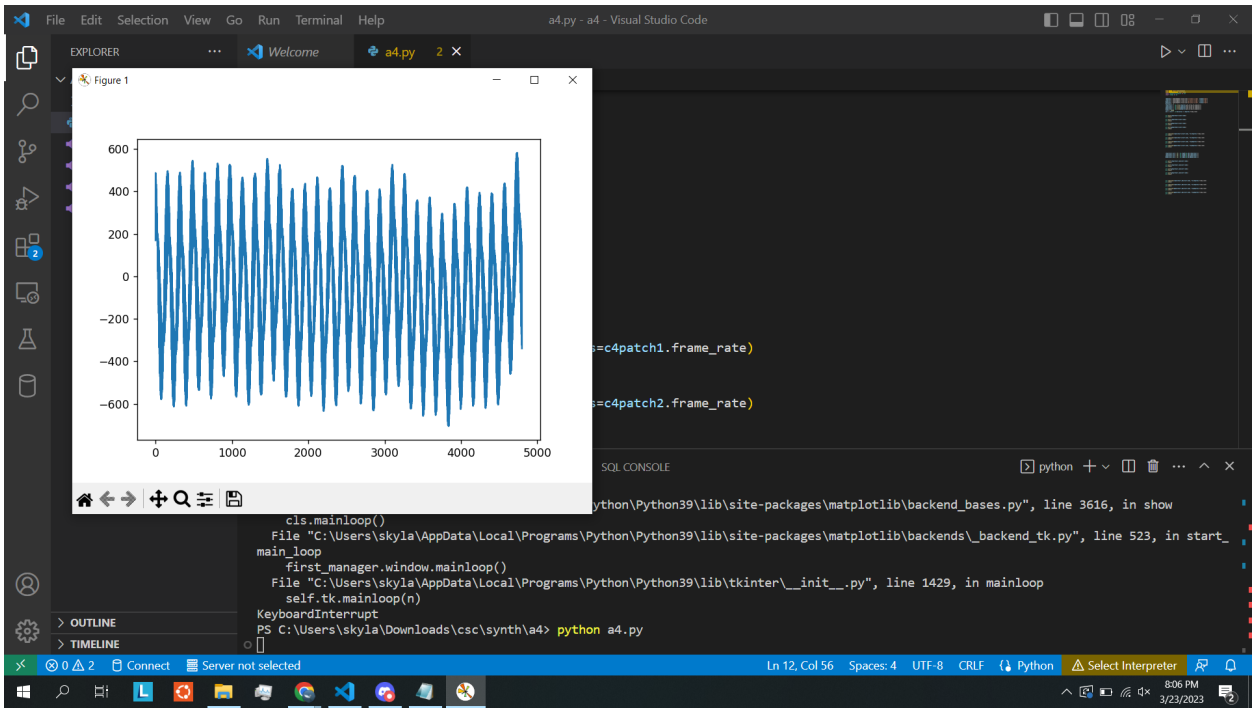
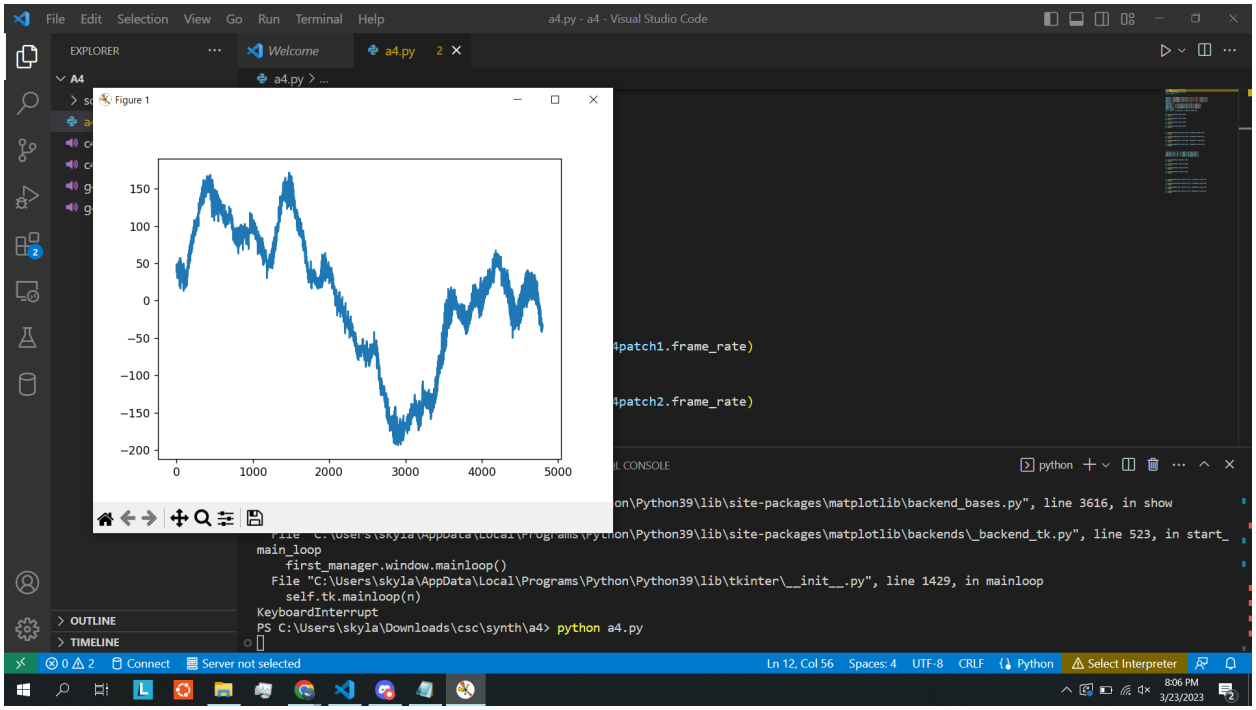
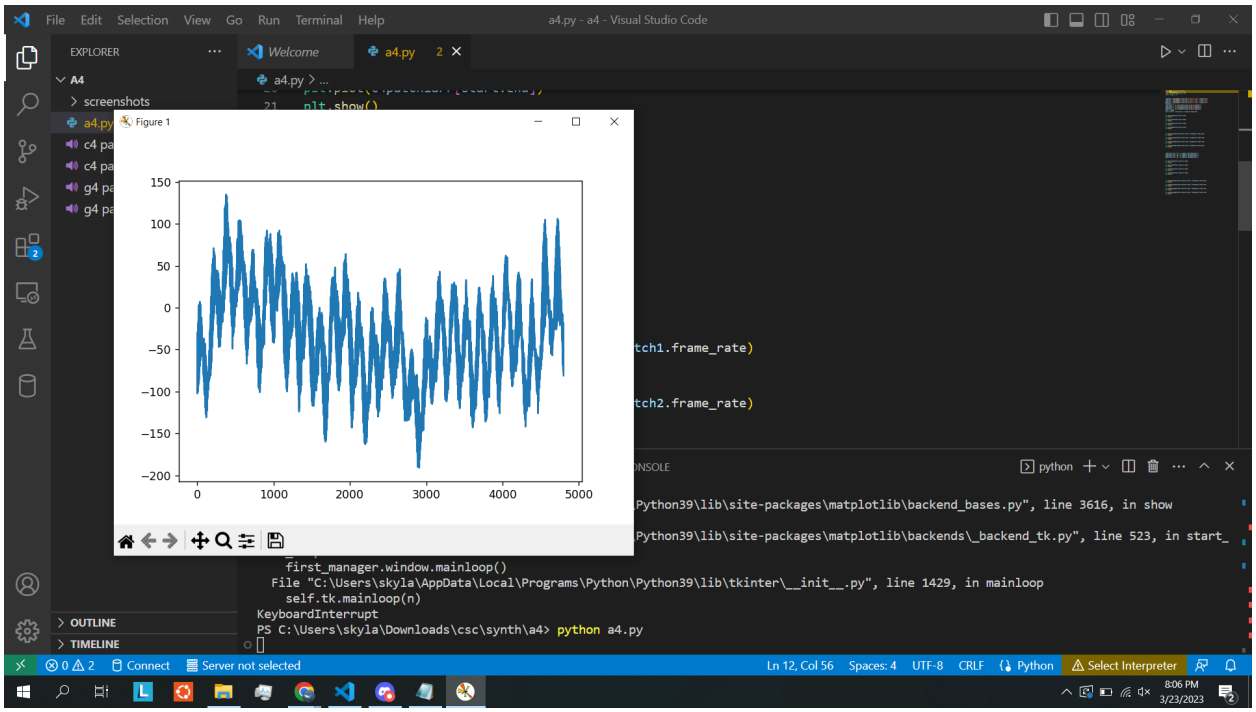
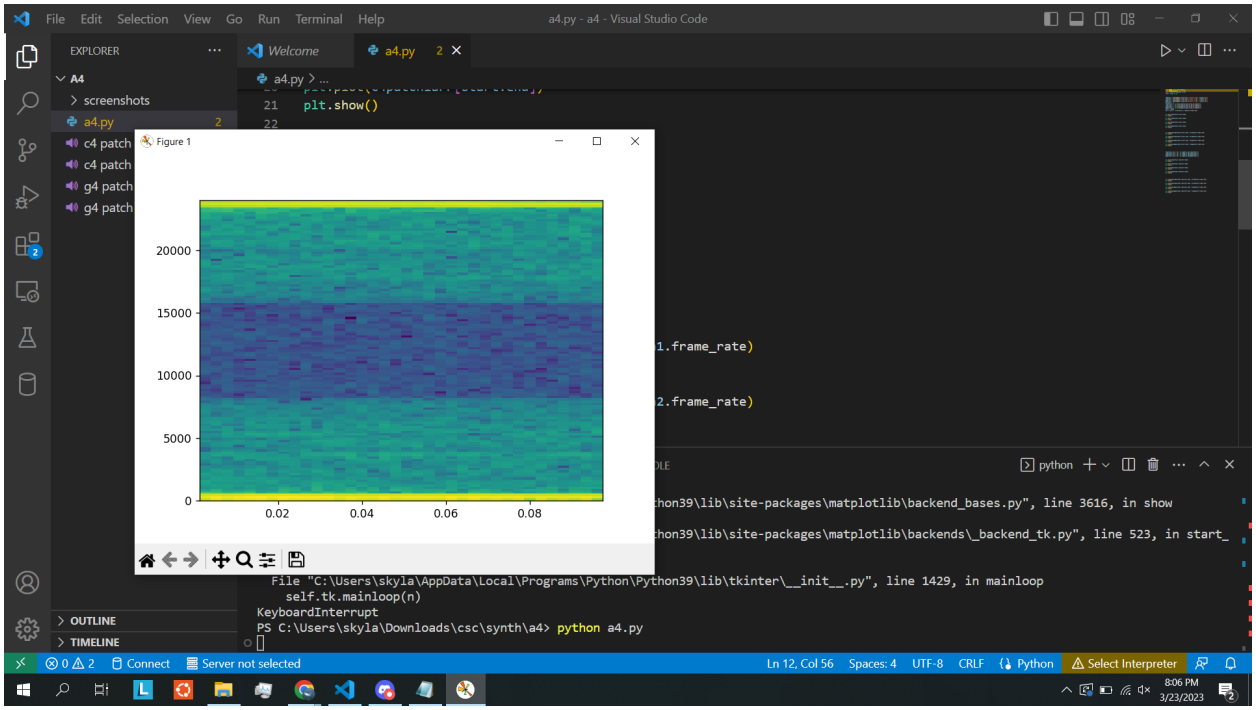


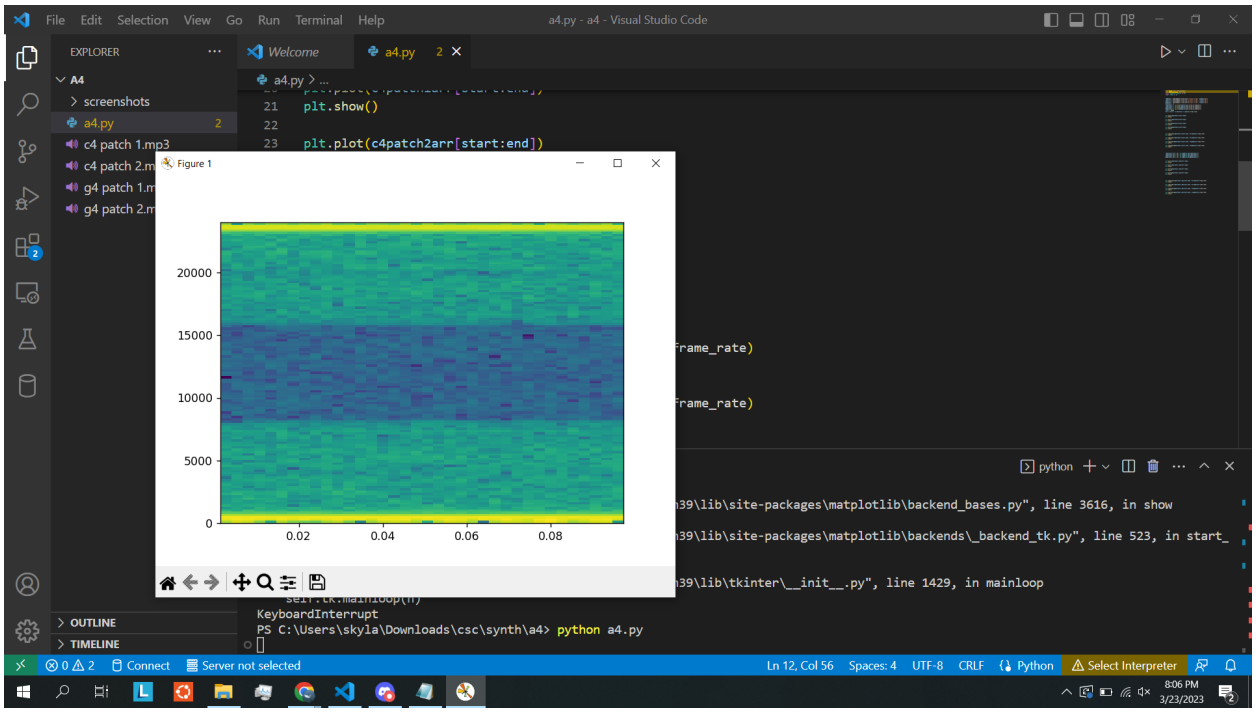
Figure 1
1500
1000
500
0
−500
−1000
−1500
0
1000
2000
3000
4000
5000
Fs=c4patch1.frame_rate)
Fs=c4patch2.frame_rate)
SQL CONSOLE
python
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\backend_bases.py", line 3616, in show
cls.mainloop()
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\backends_backend_tk.py", line 523, in start_
main_loop
first_manager.window.mainloop()
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\tkinter__init__.py", line 1429, in mainloop
self.tk.mainloop(n)
KeyboardInterrupt
PS C:\Users\skyla\Downloads\csc\synth\a4> python a4.py
OUTLINE
TIMELINE
Ln 12, Col 56
Spaces: 4
UTF-8
CRLF
Python
Select Interpreter
8:06 PM
3/23/2023


File
Edit
Selection
View
Go
Run
Terminal
Help
a4.py - a4 - Visual Studio Code
EXPLORER
Welcome
a4.py
2
Figure 1
600
400
200
0
−200
−400
−600
0
1000
2000
3000
4000
5000
SQL CONSOLE
python
cls.mainloop()
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\backends_backend_tk.py", line 523, in start_
main_loop
first_manager.window.mainloop()
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\tkinter__init__.py", line 1429, in mainloop
self.tk.mainloop(n)
KeyboardInterrupt
PS C:\Users\skyla\Downloads\csc\synth\a4> python a4.py
OUTLINE
TIMELINE
Connect
Server not selected
Ln 12, Col 56
Spaces: 4
UTF-8
CRLF
Python
Select Interpreter
8:06 PM
3/23/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
Welcome
a4.py 2
A4
Figure 1
150
100
50
0
−50
−100
−150
−200
0
1000
2000
3000
4000
5000
4patch1.frame_rate)
4patch2.frame_rate)
CONSOLE
python
on\Python39\lib\site-packages\matplotlib\backend_bases.py", line 3616, in show
\Python39\lib\site-packages\matplotlib\backends_backend_tk.py", line 523, in start_
main_loop
first_manager.window.mainloop()
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\tkinter__init__.py", line 1429, in mainloop
self.tk.mainloop(n)
KeyboardInterrupt
PS C:\Users\skyla\Downloads\csc\synth\a4> python a4.py
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
Welcome
a4.py 2
A4
screenshots
a4.py
Figure 1
150
100
50
0
−50
−100
−150
−200
0
1000
2000
3000
4000
5000
tch1.frame_rate)
tch2.frame_rate)
CONSOLE
python
Python39\lib\site-packages\matplotlib\backend_bases.py", line 3616, in show
Python39\lib\site-packages\matplotlib\backends_backend_tk.py", line 523, in start_
first_manager.window.mainloop()
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\tkinter__init__.py", line 1429, in mainloop
self.tk.mainloop(n)
KeyboardInterrupt
PS C:\Users\skyla\Downloads\csc\synth\a4> python a4.py
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch
c4 patch
g4 patch
g4 patch
Welcome
a4.py 2
a4.py > ...
21 plt.show()
22
Figure 1
20000
15000
10000
5000
0
0.02
0.04
0.06
0.08
1.frame_rate)
2.frame_rate)
python
thon39\lib\site-packages\matplotlib\backend_bases.py", line 3616, in show
thon39\lib\site-packages\matplotlib\backends_backend_tk.py", line 523, in start_
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\tkinter__init__.py", line 1429, in mainloop
self.tk.mainloop(n)
KeyboardInterrupt
PS C:\Users\skyla\Downloads\csc\synth\a4> python a4.py
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch 1.mp3
Welcome
a4.py 2
a4.py > ...
21 plt.show()
22
23 plt.plot(c4patch2arr[start:end])
Figure 1
20000
15000
10000
5000
0
0.02
0.04
0.06
0.08
frame_rate)
frame_rate)
python
39\lib\site-packages\matplotlib\backend_bases.py", line 3616, in show
39\lib\site-packages\matplotlib\backends_backend_tk.py", line 523, in start_
39\lib\tkinter__init__.py", line 1429, in mainloop
KeyboardInterrupt
PS C:\Users\skyla\Downloads\csc\synth\a4> python a4.py
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023

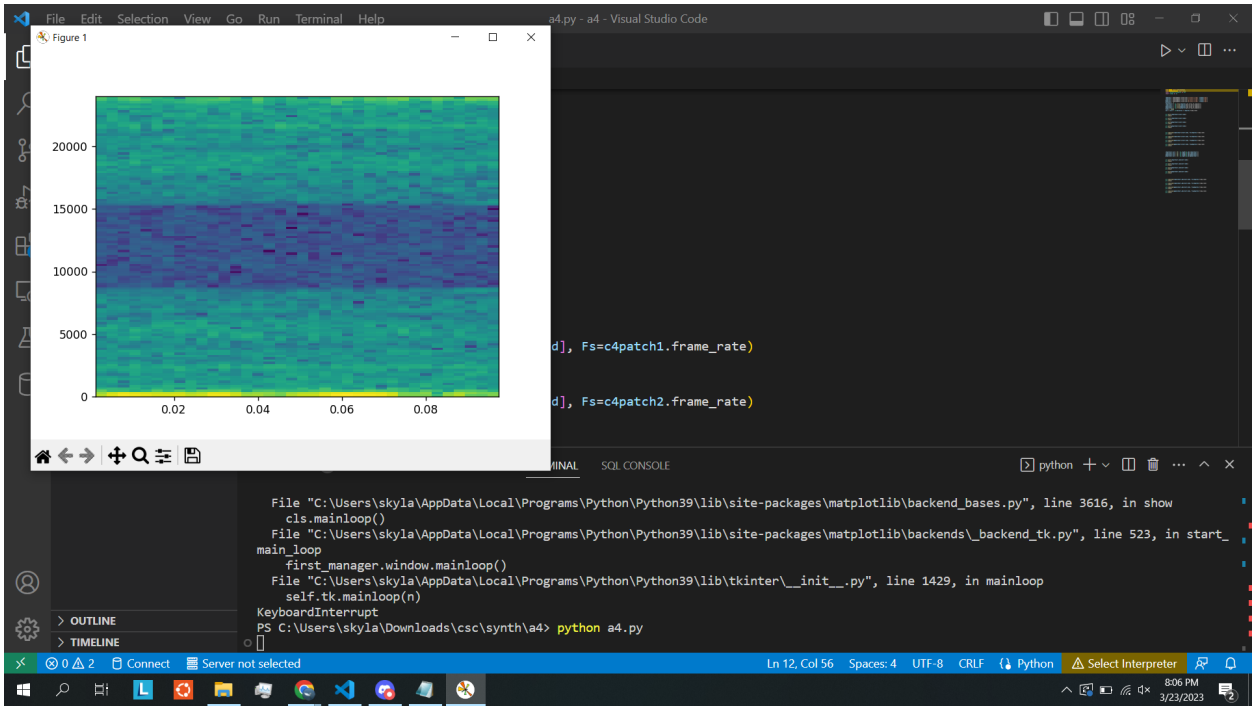


Figure 1
20000
15000
10000
5000
0
0.02
0.04
0.06
0.08
File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
d], Fs=c4patch1.frame_rate)
d], Fs=c4patch2.frame_rate)
SQL CONSOLE
python
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\backend_bases.py", line 3616, in show
cls.mainloop()
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\backends_backend_tk.py", line 523, in start_
main_loop
first_manager.window.mainloop()
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\tkinter__init__.py", line 1429, in mainloop
self.tk.mainloop(n)
KeyboardInterrupt
PS C:\Users\skyla\Downloads\csc\synth\a4> python a4.py
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023

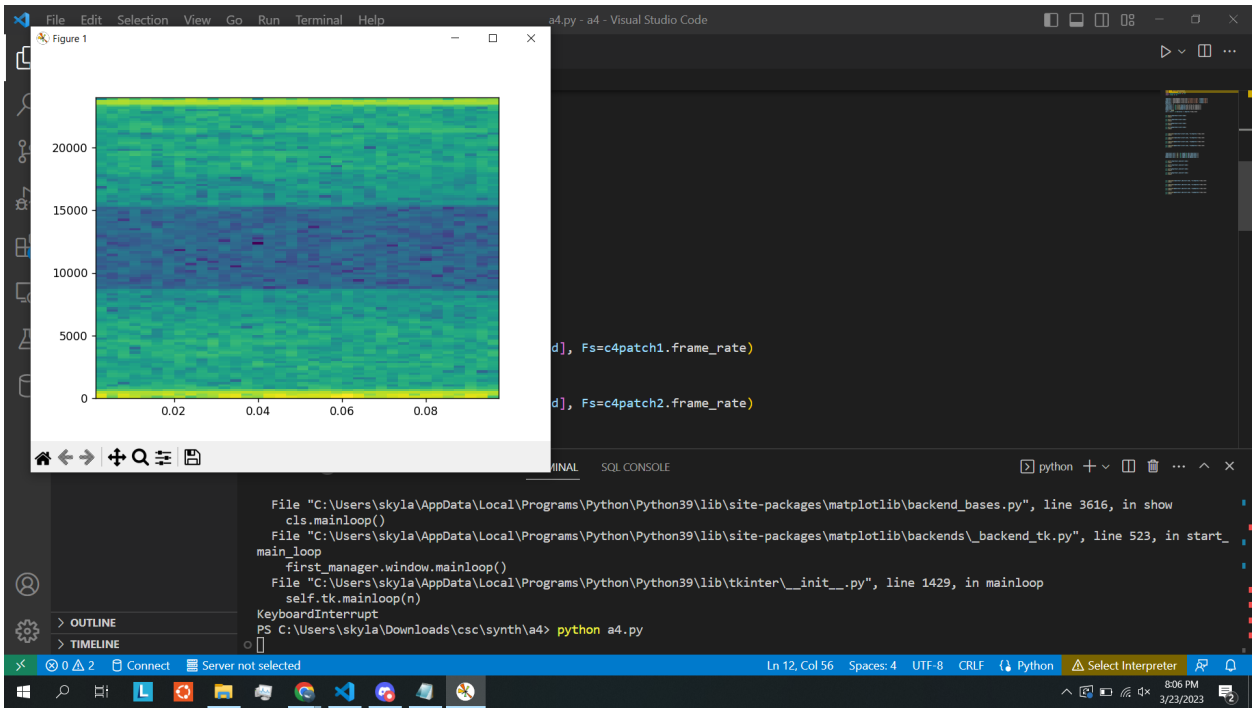
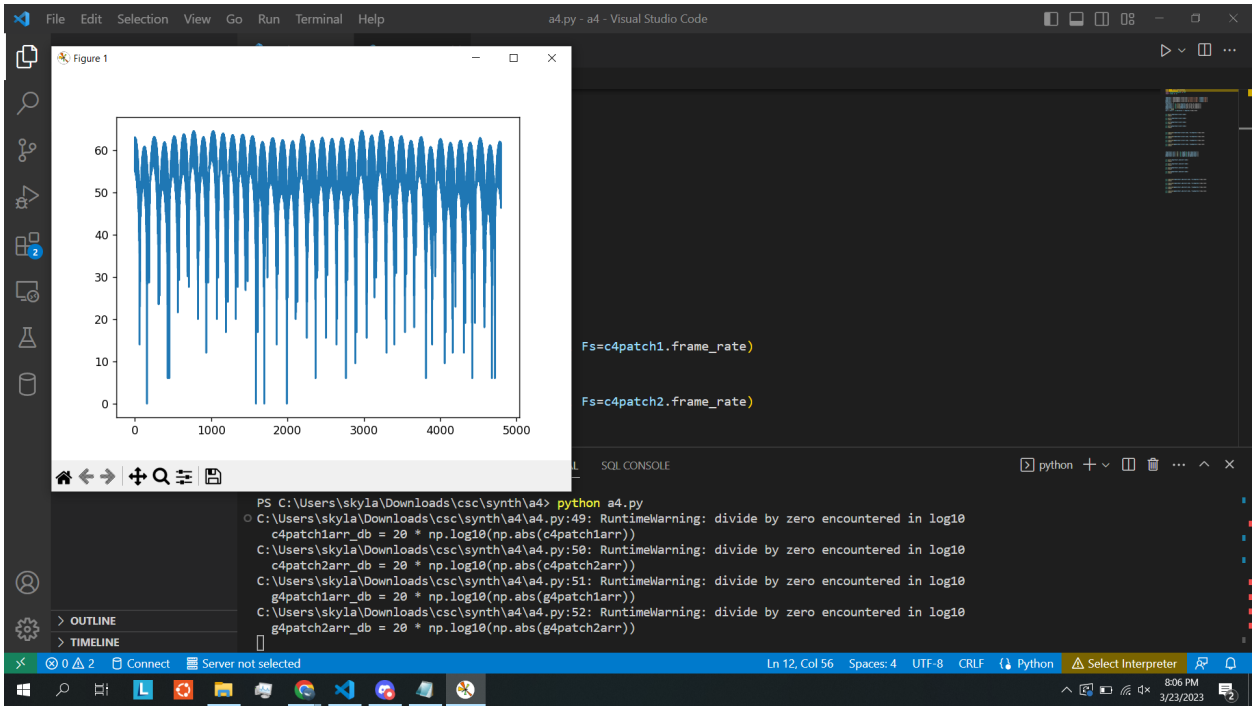
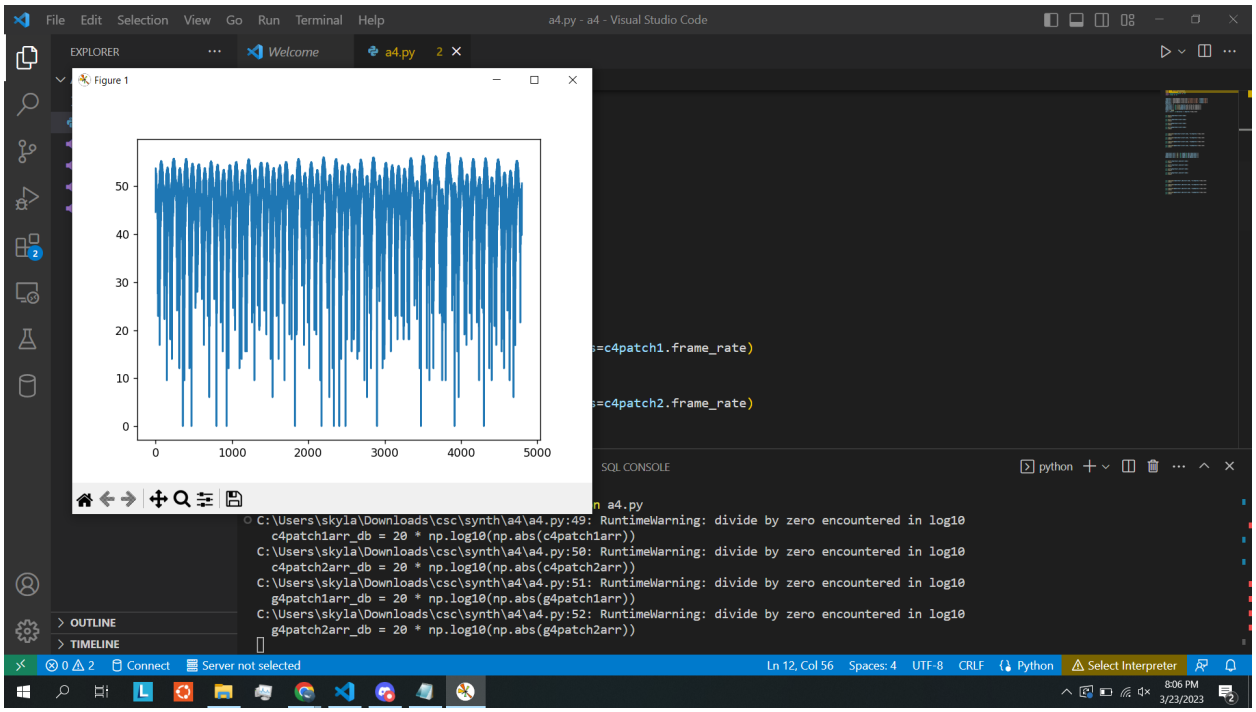
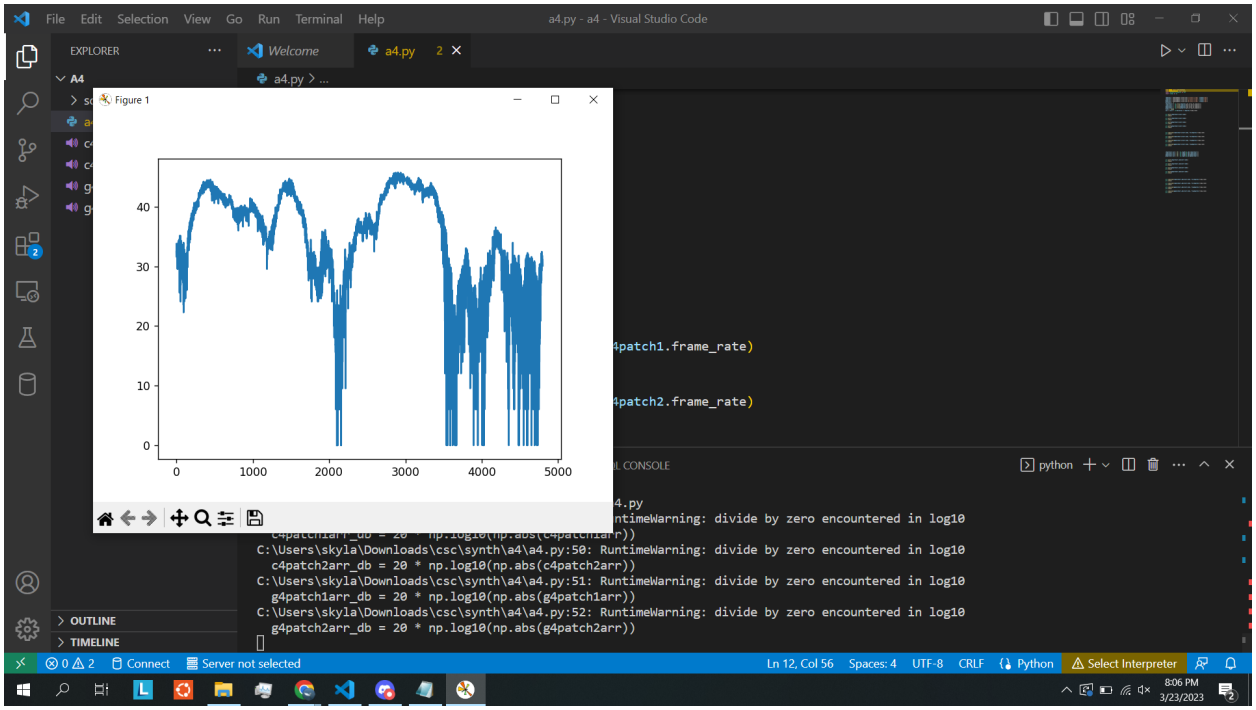
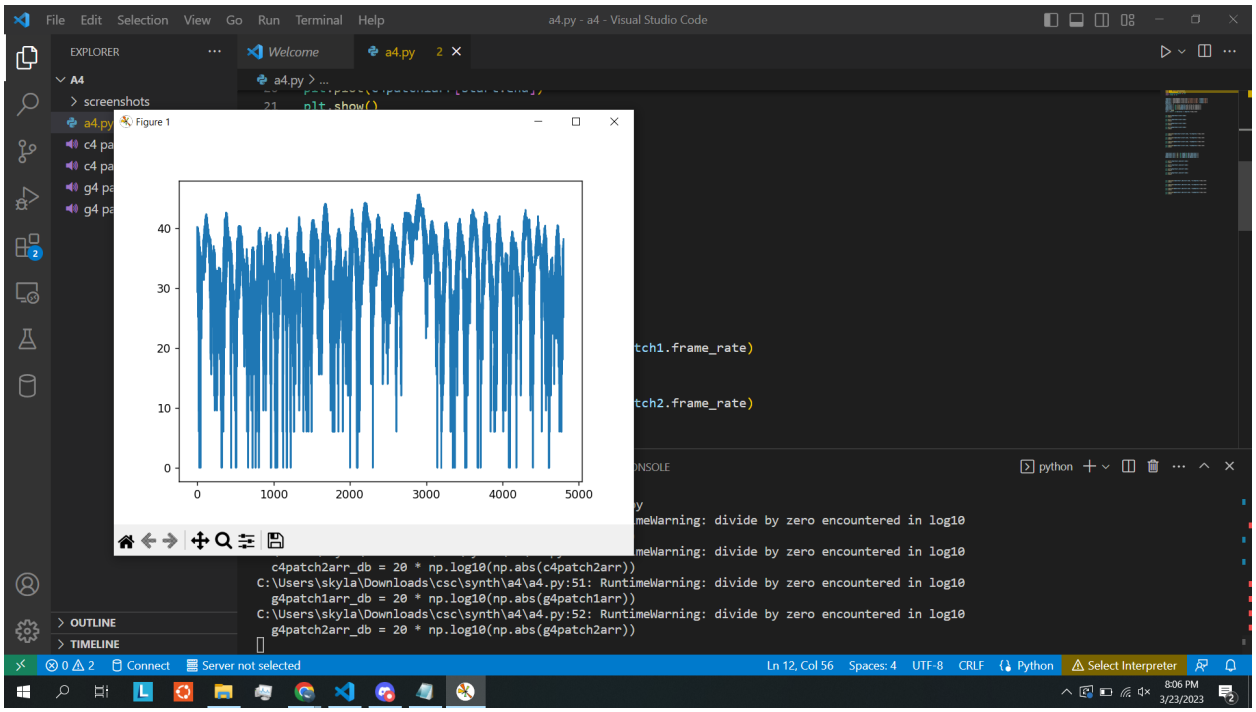


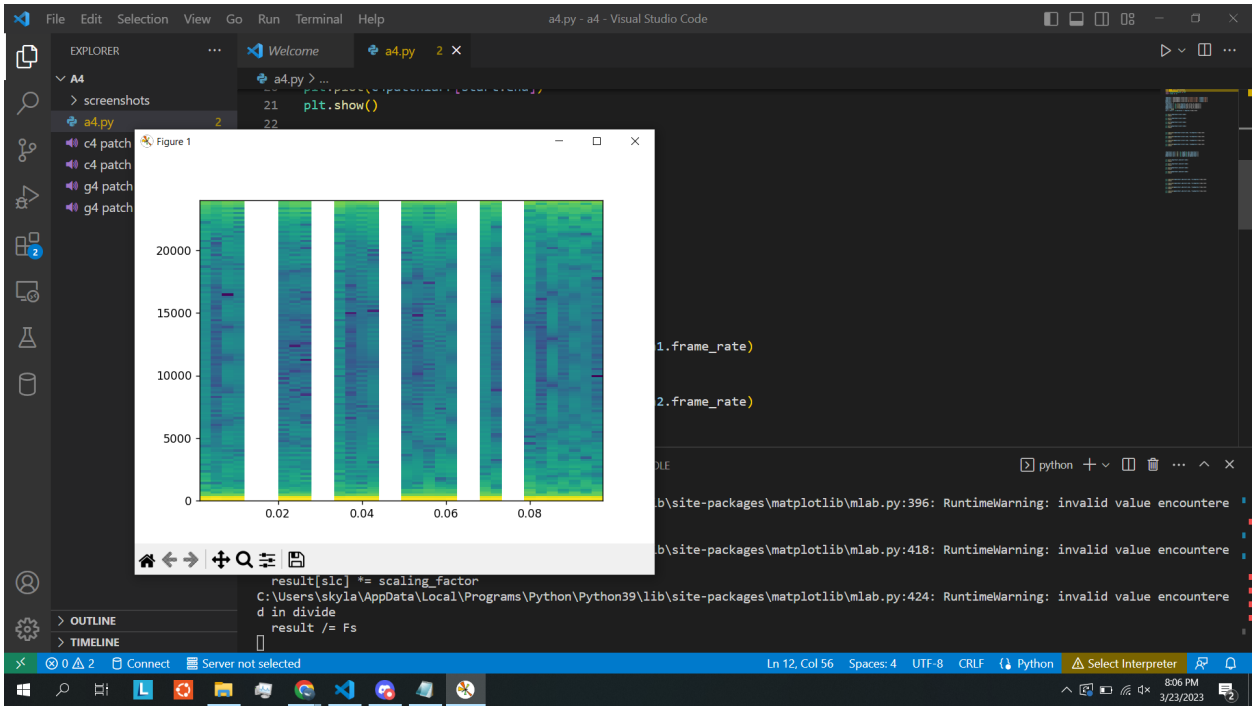
Figure 1
20000
15000
10000
5000
0
0.02
0.04
0.06
0.08
File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
d], Fs=c4patch1.frame_rate)
d], Fs=c4patch2.frame_rate)
SQL CONSOLE
python
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\backend_bases.py", line 3616, in show
cls.mainloop()
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\backends_backend_tk.py", line 523, in start_
main_loop
first_manager.window.mainloop()
File "C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\tkinter__init__.py", line 1429, in mainloop
self.tk.mainloop(n)
KeyboardInterrupt
PS C:\Users\skyla\Downloads\csc\synth\a4> python a4.py
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023

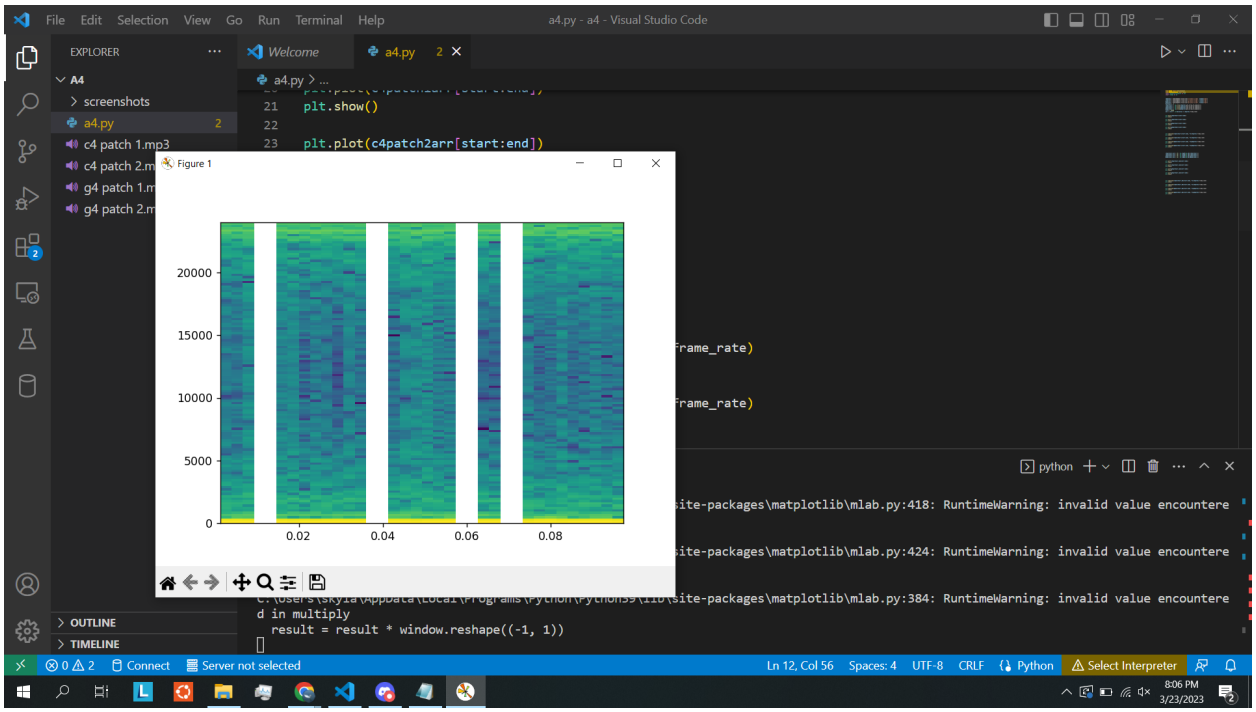

File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
Figure 1
Fs=c4patch1.frame_rate)
Fs=c4patch2.frame_rate)
SQL CONSOLE
python
PS C:\Users\skyla\Downloads\csc\synth\a4> python a4.py
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:49: RuntimeWarning: divide by zero encountered in log10
c4patch1arr_db = 20 * np.log10(np.abs(c4patch1arr))
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:50: RuntimeWarning: divide by zero encountered in log10
c4patch2arr_db = 20 * np.log10(np.abs(c4patch2arr))
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:51: RuntimeWarning: divide by zero encountered in log10
g4patch1arr_db = 20 * np.log10(np.abs(g4patch1arr))
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:52: RuntimeWarning: divide by zero encountered in log10
g4patch2arr_db = 20 * np.log10(np.abs(g4patch2arr))
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
Welcome
a4.py 2
Figure 1
=c4patch1.frame_rate)
=c4patch2.frame_rate)
SQL CONSOLE
python
n a4.py
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:49: RuntimeWarning: divide by zero encountered in log10
c4patch1arr_db = 20 * np.log10(np.abs(c4patch1arr))
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:50: RuntimeWarning: divide by zero encountered in log10
c4patch2arr_db = 20 * np.log10(np.abs(c4patch2arr))
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:51: RuntimeWarning: divide by zero encountered in log10
g4patch1arr_db = 20 * np.log10(np.abs(g4patch1arr))
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:52: RuntimeWarning: divide by zero encountered in log10
g4patch2arr_db = 20 * np.log10(np.abs(g4patch2arr))
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
Welcome
a4.py 2
A4
Figure 1
4patch1.frame_rate)
4patch2.frame_rate)
CONSOLE
python
4.py
ntimeWarning: divide by zero encountered in log10
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:50: RuntimeWarning: divide by zero encountered in log10
c4patch2arr_db = 20 * np.log10(np.abs(c4patch2arr))
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:51: RuntimeWarning: divide by zero encountered in log10
g4patch1arr_db = 20 * np.log10(np.abs(g4patch1arr))
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:52: RuntimeWarning: divide by zero encountered in log10
g4patch2arr_db = 20 * np.log10(np.abs(g4patch2arr))
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
Welcome
a4.py 2
A4
screenshots
a4.py
Figure 1
tch1.frame_rate)
tch2.frame_rate)
CONSOLE
python
meWarning: divide by zero encountered in log10
meWarning: divide by zero encountered in log10
c4patch2arr_db = 20 * np.log10(np.abs(c4patch2arr))
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:51: RuntimeWarning: divide by zero encountered in log10
g4patch1arr_db = 20 * np.log10(np.abs(g4patch1arr))
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:52: RuntimeWarning: divide by zero encountered in log10
g4patch2arr_db = 20 * np.log10(np.abs(g4patch2arr))
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
Welcome
a4.py 2
A4
screenshots
a4.py 2
c4 patch
c4 patch
g4 patch
g4 patch
Figure 1
20000
15000
10000
5000
0
0.02
0.04
0.06
0.08
plt.show()
1.frame_rate)
2.frame_rate)
python
b\site-packages\matplotlib\mlab.py:396: RuntimeWarning: invalid value encountere
b\site-packages\matplotlib\mlab.py:418: RuntimeWarning: invalid value encountere
result[slc] *= scaling_factor
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:424: RuntimeWarning: invalid value encountere
d in divide
result /= Fs
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
Welcome
a4.py 2
A4
screenshots
a4.py 2
c4 patch 1.mp3
c4 patch 2.m
g4 patch 1.m
g4 patch 2.m
Figure 1
20000
15000
10000
5000
0
0.02
0.04
0.06
0.08
plt.show()
plt.plot(c4patch2arr[start:end])
frame_rate)
frame_rate)
python
site-packages\matplotlib\mlab.py:418: RuntimeWarning: invalid value encountere
site-packages\matplotlib\mlab.py:424: RuntimeWarning: invalid value encountere
site-packages\matplotlib\mlab.py:384: RuntimeWarning: invalid value encountere
d in multiply
result = result * window.reshape((-1, 1))
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023

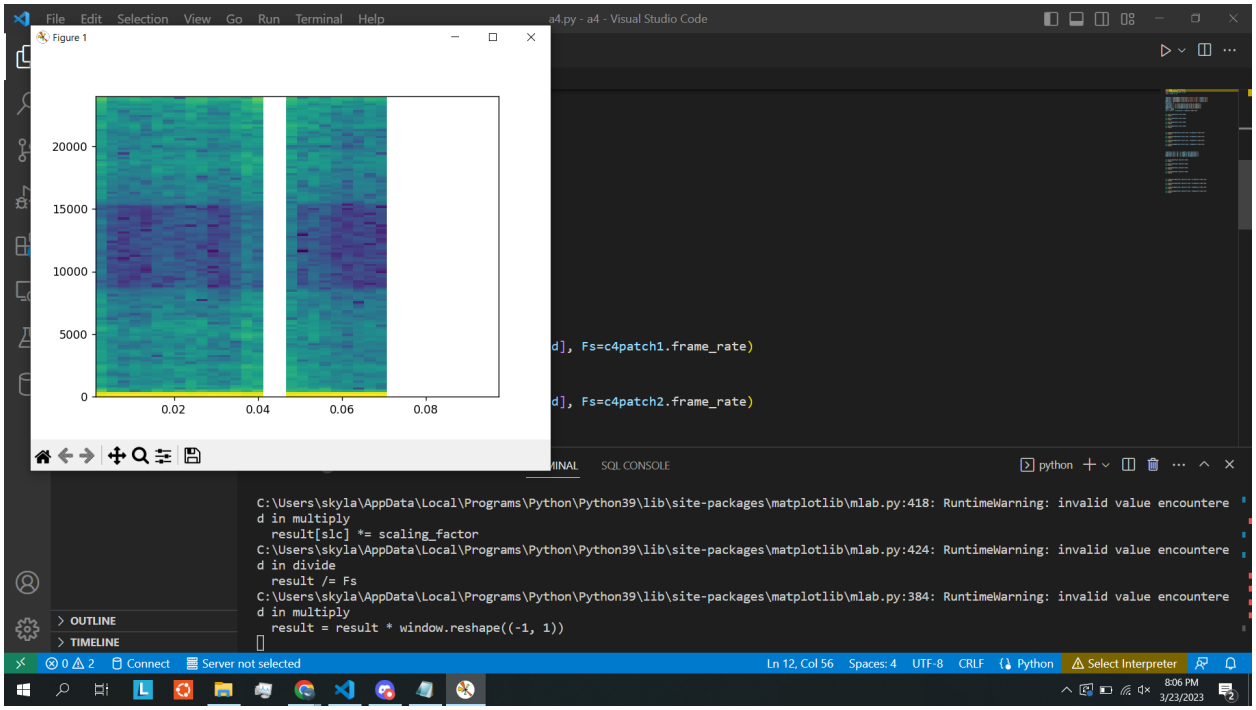


Figure 1
20000
15000
10000
5000
0
0.02
0.04
0.06
0.08
File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
d], Fs=c4patch1.frame_rate)
d], Fs=c4patch2.frame_rate)
SQL CONSOLE
python
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:418: RuntimeWarning: invalid value encountere
d in multiply
result[slc] *= scaling_factor
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:424: RuntimeWarning: invalid value encountere
d in divide
result /= Fs
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:384: RuntimeWarning: invalid value encountere
d in multiply
result = result * window.reshape((-1, 1))
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023

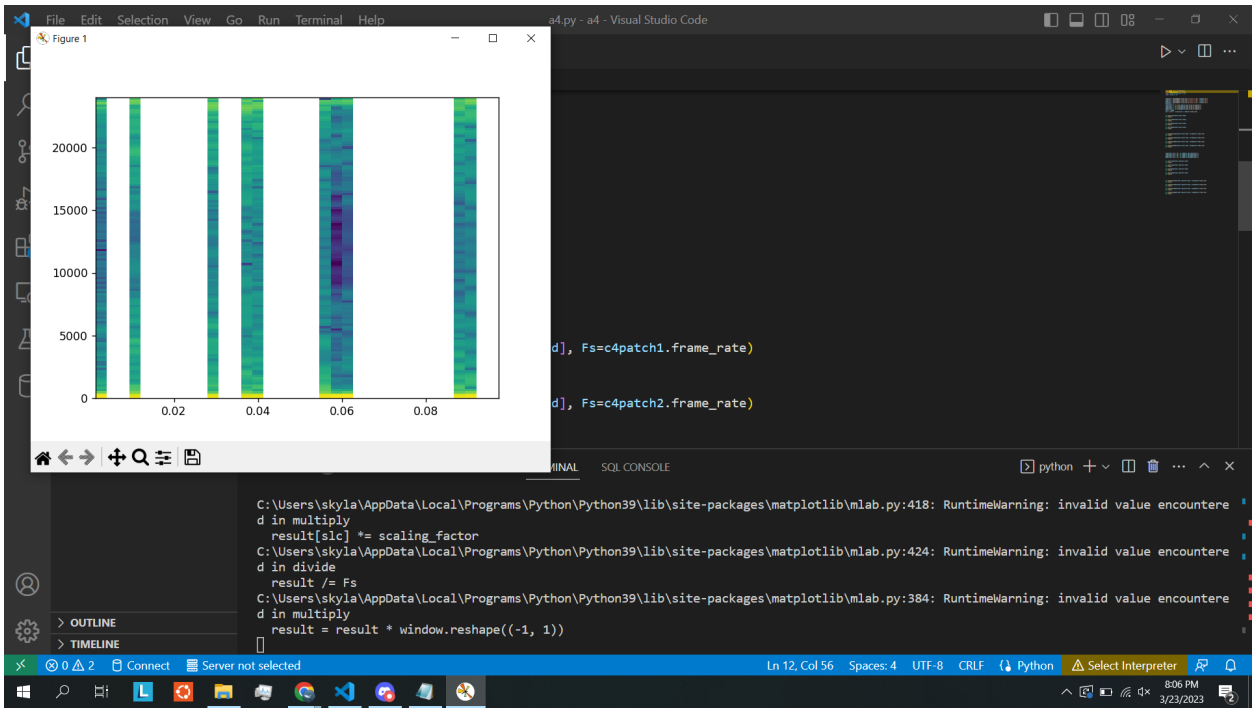
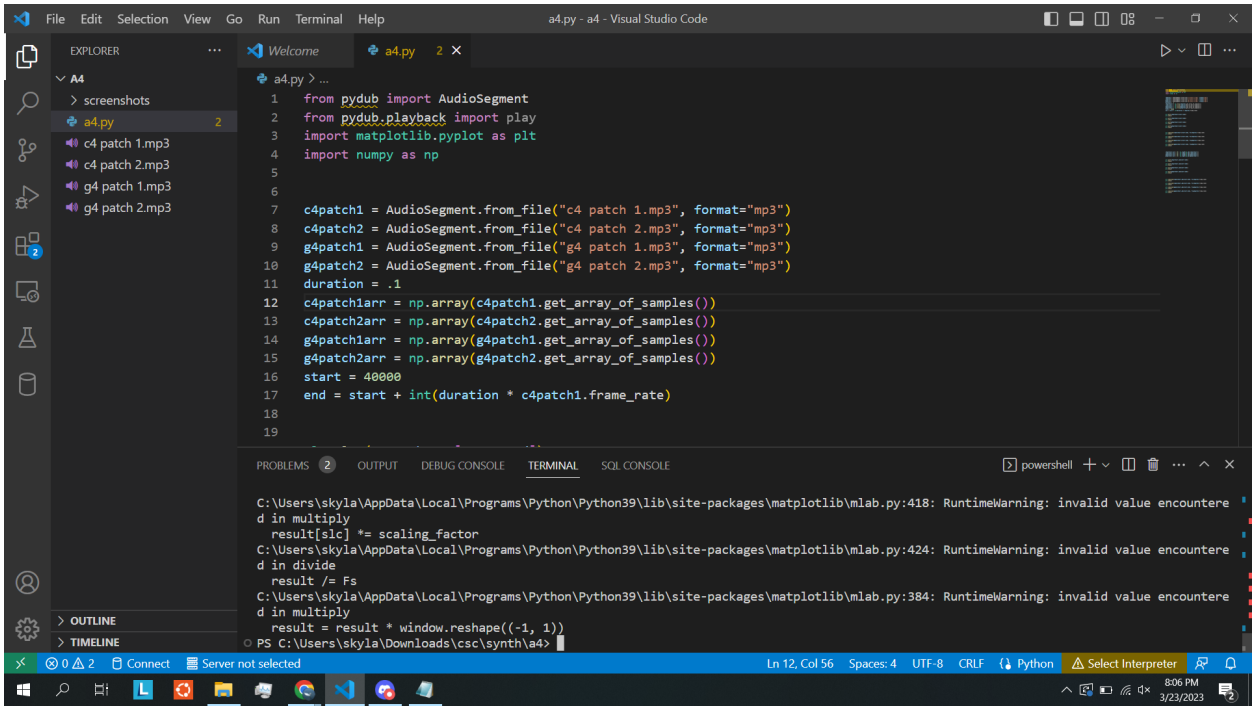
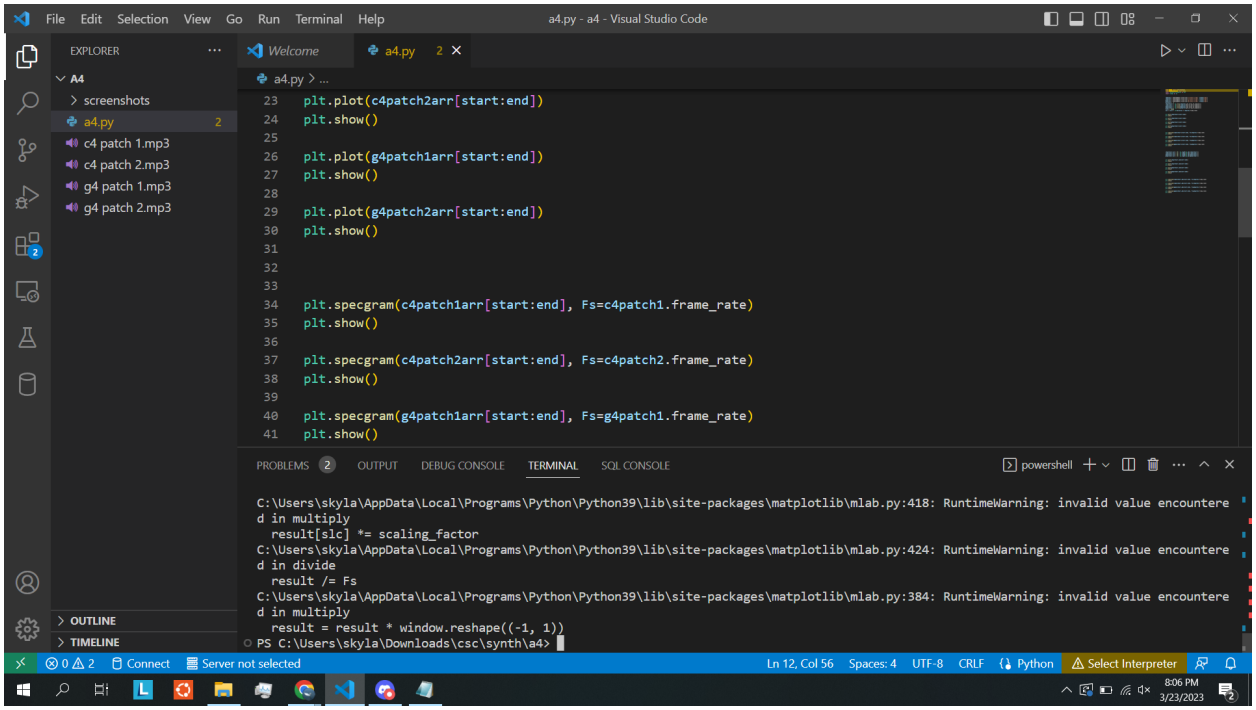
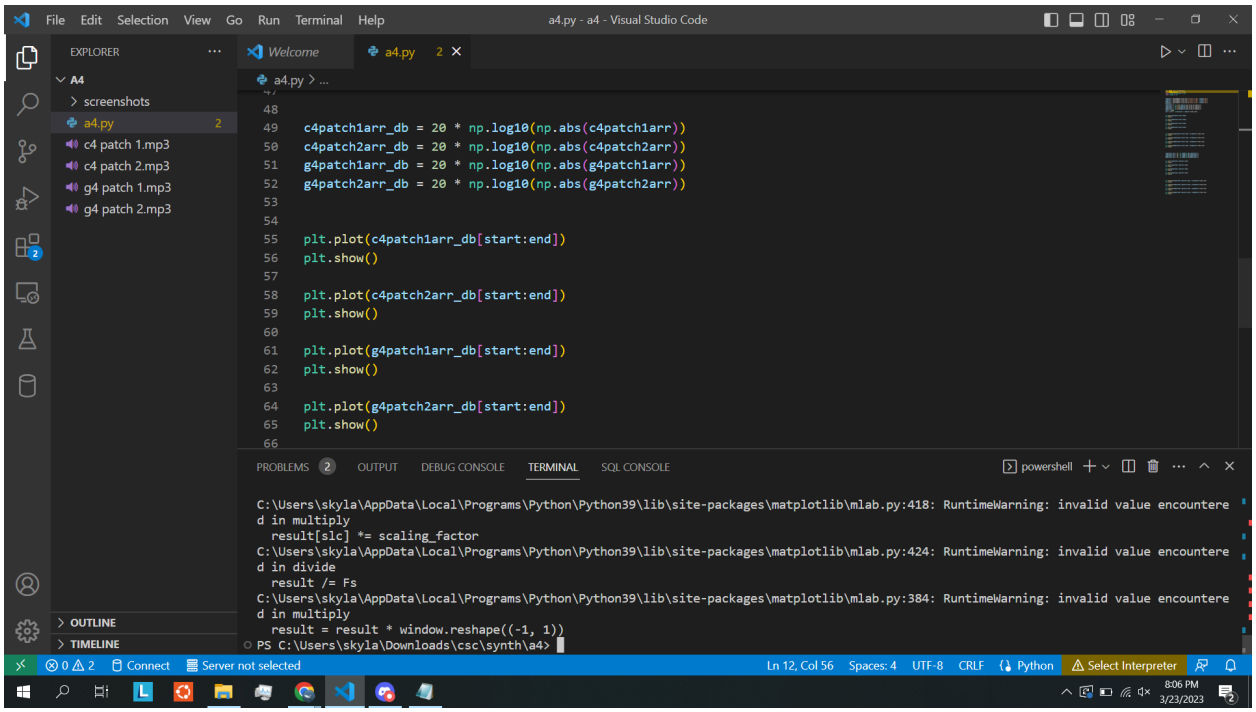


Figure 1
20000
15000
10000
5000
0
0.02
0.04
0.06
0.08
File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
d], Fs=c4patch1.frame_rate)
d], Fs=c4patch2.frame_rate)
SQL CONSOLE
python
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:418: RuntimeWarning: invalid value encountere
d in multiply
result[slc] *= scaling_factor
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:424: RuntimeWarning: invalid value encountere
d in divide
result /= Fs
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:384: RuntimeWarning: invalid value encountere
d in multiply
result = result * window.reshape((-1, 1))
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
Welcome
a4.py 2
a4.py > ...
1 from pydub import AudioSegment
2 from pydub.playback import play
3 import matplotlib.pyplot as plt
4 import numpy as np
5
6
7 c4patch1 = AudioSegment.from_file("c4 patch 1.mp3", format="mp3")
8 c4patch2 = AudioSegment.from_file("c4 patch 2.mp3", format="mp3")
9 g4patch1 = AudioSegment.from_file("g4 patch 1.mp3", format="mp3")
10 g4patch2 = AudioSegment.from_file("g4 patch 2.mp3", format="mp3")
11 duration = .1
12 c4patch1arr = np.array(c4patch1.get_array_of_samples())
13 c4patch2arr = np.array(c4patch2.get_array_of_samples())
14 g4patch1arr = np.array(g4patch1.get_array_of_samples())
15 g4patch2arr = np.array(g4patch2.get_array_of_samples())
16 start = 40000
17 end = start + int(duration * c4patch1.frame_rate)
18
19
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
powershell
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:418: RuntimeWarning: invalid value encountere
d in multiply
result[slc] *= scaling_factor
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:424: RuntimeWarning: invalid value encountere
d in divide
result /= Fs
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:384: RuntimeWarning: invalid value encountere
d in multiply
result = result * window.reshape((-1, 1))
PS C:\Users\skyla\Downloads\csc\synth\a4>
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python Select Interpreter
8:06 PM
3/23/2023



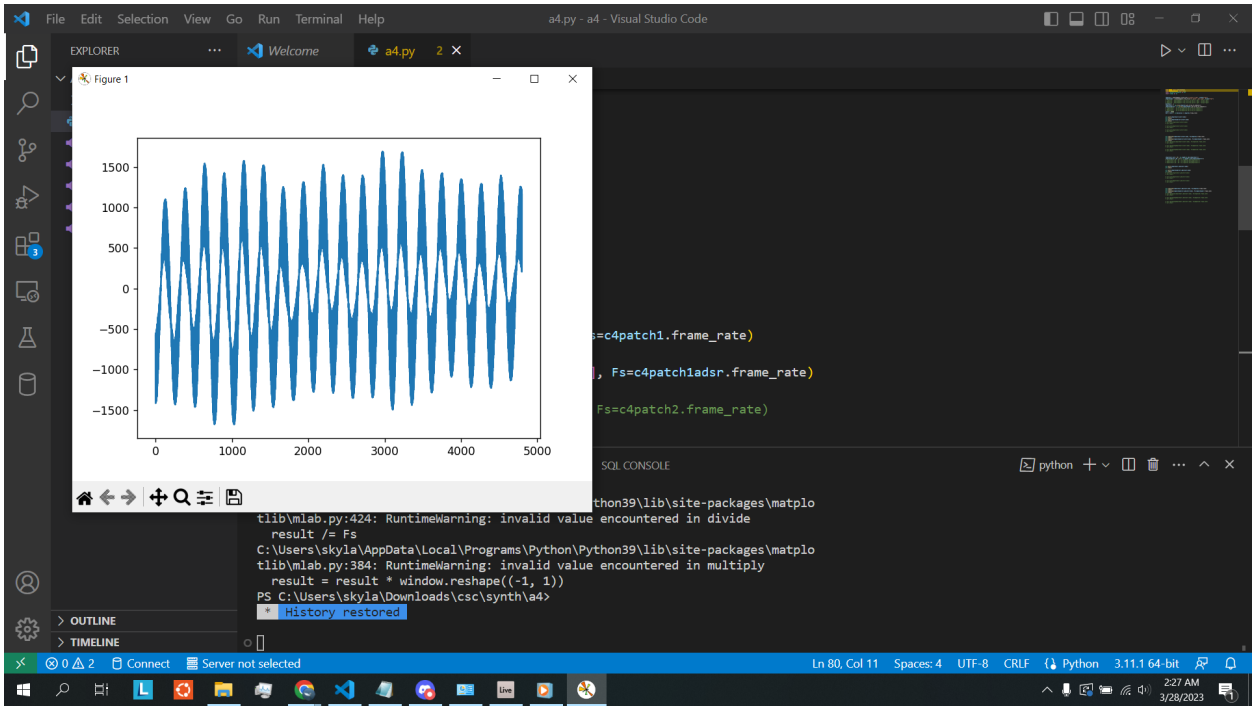


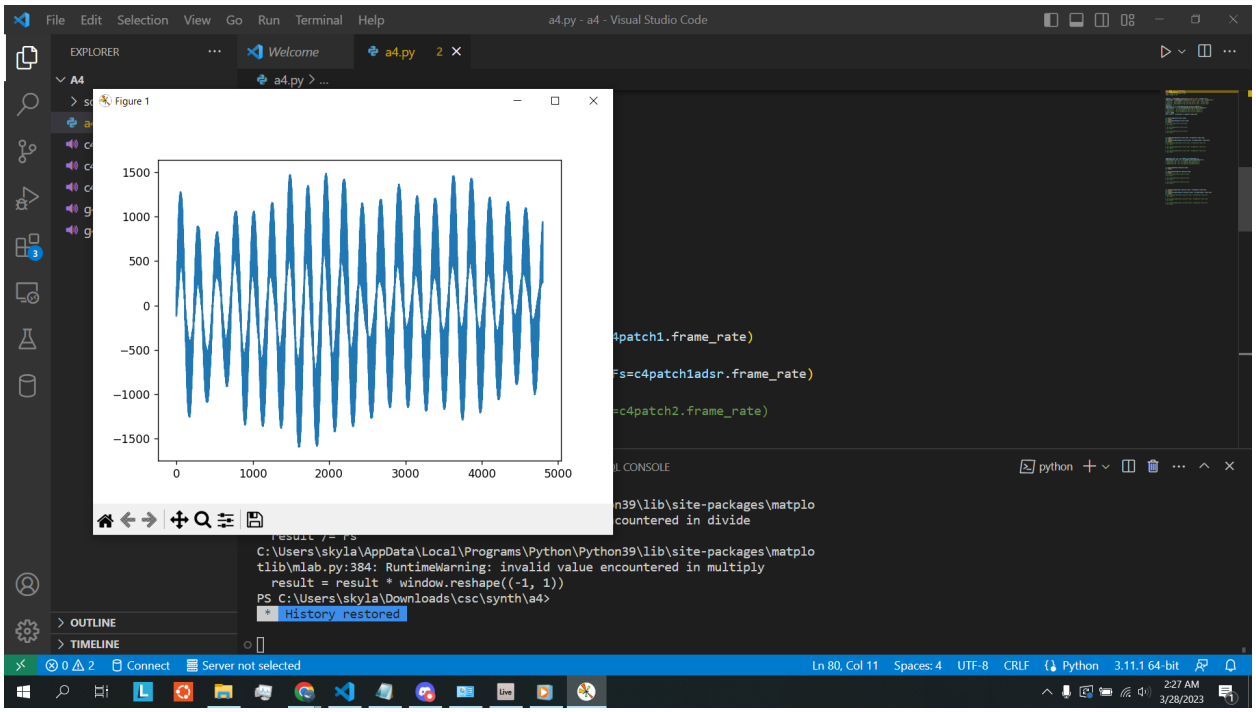
For question 2 I had to record four notes. two of C4 and the other two of G4. The reason I have two of each note is because they are from two different patches of the Yamaha DX21. I used the Yamaha DX21 that is in ECS602. I have to graph the time domain plots and magnitude spectrograms of these “four” notes in linear and decibel amplitudes. I used a very short duration that corresponds to a few pitch cycles of the notes. The graph screenshots show a cycle of C4 note 1, C4 note 2, G4 note 1, and then G4 note 2. Using this cycle you can see, for each note, the time domain plots for linear amplitude, then magnitude spectrograms for linear amplitude,

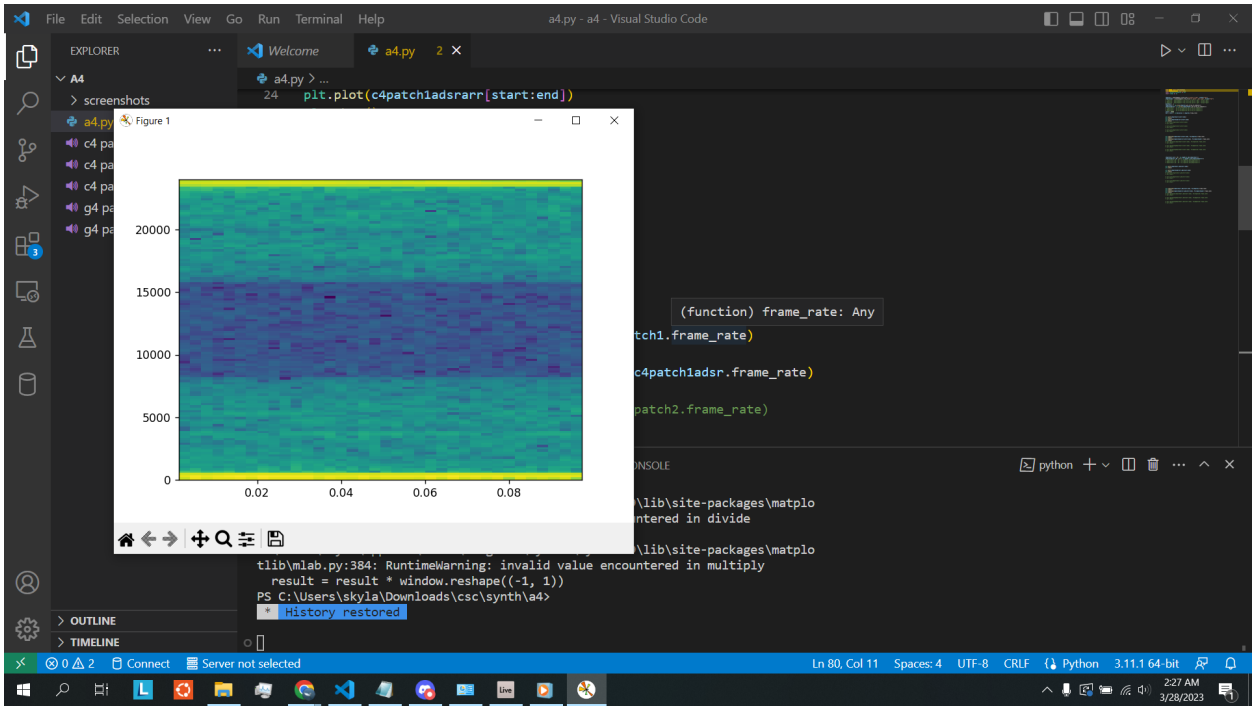
then the time domain plots in decibel amplitude, and finally the magnitude spectrograms in decibel amplitude. In terms of the code shown on the other screenshots, the process was to load the mp3 files using pydub and define the short duration of each note in seconds. The short duration I chose to correspond with a few pitch cycles of the notes was .1 seconds. Then after I convert the mp3 files to numpy arrays I can get a starting and ending spot for the sample of the duration I set earlier. After that I can plot the time domain signals using linear amplitude. I can also plot the magnitude spectrograms for linear amplitude as well using ".specgram". Then after converting the amplitude to dB I can plot the time domain signals and magnitude spectrograms using dB amplitude and the same process of utilizing ".plot" and ".specgram".

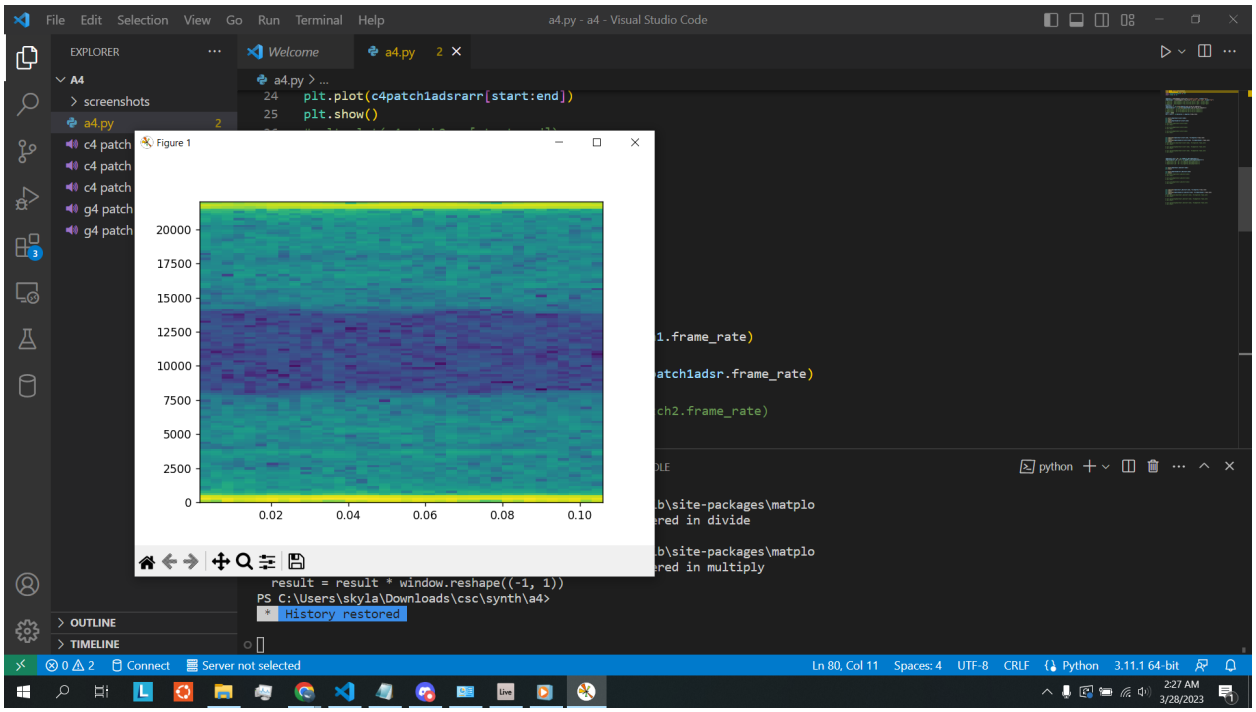
For question 3 I have to get the two FM algorithms that are used for each of my sound patches from the previous question where I used the DX21 synthesizer. To learn more about the FM algorithms I used the manual for the DX21 synthesizer and some discussion sites that can be found online for research. Of the DX21 synthesizer's available algorithms for FM synthesis, I found that I was using Algorithms 5 and 1. Algorithms for the DX21 synthesizer are typically referred to by their number designation in most documentation and discussions. They do not have specific names like some other synthesizers may give to their algorithms. Conversely I have found that some users may refer to what they do specifically, such as calling them the percussive algorithm or the pad algorithm. Each algorithm uses a different set of operator connections. Algorithm 1 consists of two operators: Operator 1 acts as the modulator, and

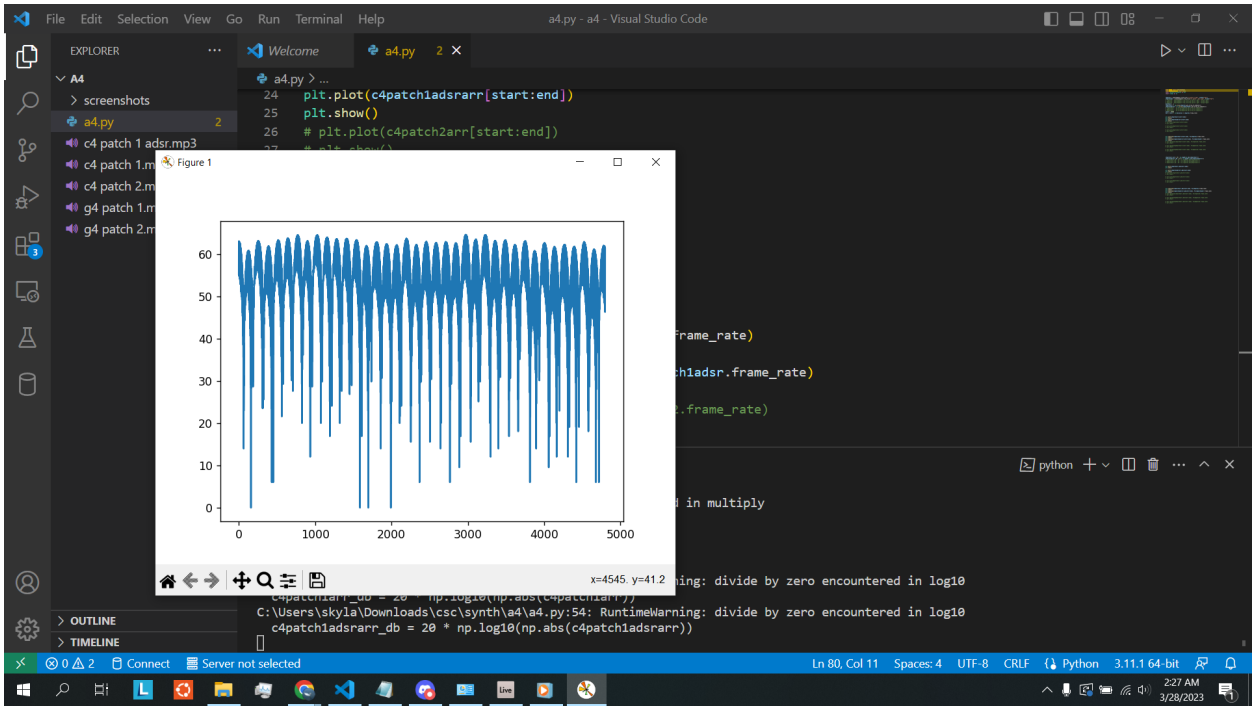
Operator 2 acts as the carrier. In this algorithm, the output of Operator 1 is fed into the frequency modulation input of Operator 2. Algorithm 5, on the other hand, is a more complex algorithm that uses four operators. Operators 1 and 2 act as modulators, while Operators 3 and 4 act as carriers. In this algorithm, the output of Operator 1 is fed into the frequency modulation input of Operator 2, which in turn is fed into the frequency modulation input of Operator 3. The output of Operator 2 is also fed into the frequency modulation input of Operator 4. The manual for the DX21 synthesizer and the discussion sites I browsed were very helpful for getting the information I needed and then clarifying it so it was more digestible.

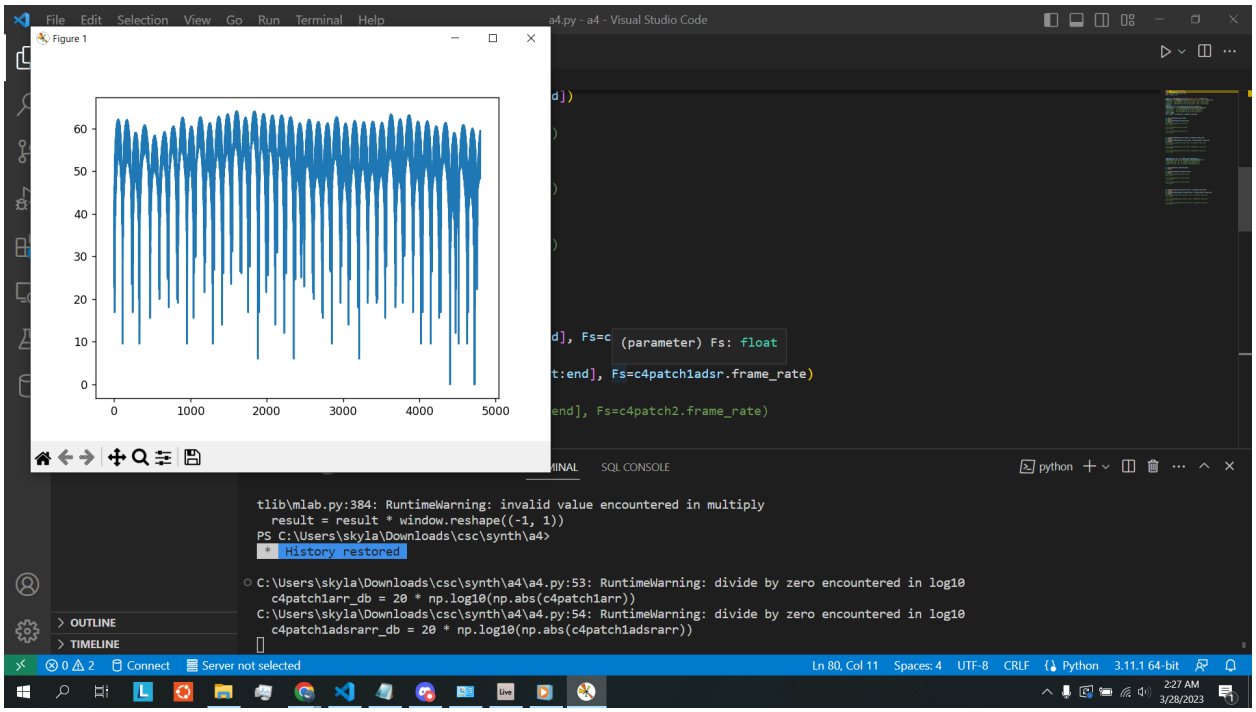

File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
Welcome
a4.py 2
Figure 1
1500
1000
500
0
−500
−1000
−1500
0
1000
2000
3000
4000
5000
=c4patch1.frame_rate)
, Fs=c4patch1adsr.frame_rate)
Fs=c4patch2.frame_rate)
SQL CONSOLE
python
thon39\lib\site-packages\matplo
tlib\mlab.py:424: RuntimeWarning: invalid value encountered in divide
result /= Fs
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplo
tlib\mlab.py:384: RuntimeWarning: invalid value encountered in multiply
result = result * window.reshape((-1, 1))
PS C:\Users\skyla\Downloads\csc\synth\a4>
History restored
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 80, Col 11 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:27 AM
3/28/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
Welcome
a4.py 2
A4
a4.py > ...
Figure 1
1500
1000
500
0
−500
−1000
−1500
0
1000
2000
3000
4000
5000
4patch1.frame_rate)
Fs=c4patch1adsr.frame_rate)
=c4patch2.frame_rate)
L CONSOLE
python
n39\lib\site-packages\matplo
countered in divide
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplo
tlib\mlab.py:384: RuntimeWarning: invalid value encountered in multiply
result = result * window.reshape((-1, 1))
PS C:\Users\skyla\Downloads\csc\synth\a4>
History restored
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 80, Col 11 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:27 AM
3/28/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
Welcome
a4.py 2
A4
screenshots
a4.py
a4.py > ...
24 plt.plot(c4patch1adsrarr[start:end])
Figure 1
20000
15000
10000
5000
0
0.02
0.04
0.06
0.08
(function) frame_rate: Any
tch1.frame_rate)
c4patch1adsr.frame_rate)
patch2.frame_rate)
python
\lib\site-packages\matplo
ntered in divide
\lib\site-packages\matplo
tlib\mlab.py:384: RuntimeWarning: invalid value encountered in multiply
result = result * window.reshape((-1, 1))
PS C:\Users\skyla\Downloads\csc\synth\a4>
History restored
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 80, Col 11 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:27 AM
3/28/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
Welcome
a4.py 2
A4
screenshots
a4.py 2
a4.py > ...
24 plt.plot(c4patch1adsrarr[start:end])
25 plt.show()
Figure 1
20000
17500
15000
12500
10000
7500
5000
2500
0
0.02
0.04
0.06
0.08
0.10
1.frame_rate)
atch1adsr.frame_rate)
ch2.frame_rate)
python
b\site-packages\matplo
ered in divide
b\site-packages\matplo
ered in multiply
result = result * window.reshape((-1, 1))
PS C:\Users\skyla\Downloads\csc\synth\a4>
History restored
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 80, Col 11 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:27 AM
3/28/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch 1 adsr.mp3
Welcome
a4.py 2
plt.plot(c4patch1adsrarr[start:end])
plt.show()
# plt.plot(c4patch2arr[start:end])
Figure 1
x=4545. y=41.2
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:54: RuntimeWarning: divide by zero encountered in log10
c4patch1adsrarr_db = 20 * np.log10(np.abs(c4patch1adsrarr))
OUTLINE
TIMELINE
Ln 80, Col 11 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:27 AM
3/28/2023


File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
Figure 1
(parameter) Fs: float
SQL CONSOLE
tlib\mlab.py:384: RuntimeWarning: invalid value encountered in multiply
result = result * window.reshape((-1, 1))
PS C:\Users\skyla\Downloads\csc\synth\a4>
History restored
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:53: RuntimeWarning: divide by zero encountered in log10
c4patch1arr_db = 20 * np.log10(np.abs(c4patch1arr))
C:\Users\skyla\Downloads\csc\synth\a4\a4.py:54: RuntimeWarning: divide by zero encountered in log10
c4patch1adsrarr_db = 20 * np.log10(np.abs(c4patch1adsrarr))
OUTLINE
TIMELINE
Ln 80, Col 11 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:27 AM
3/28/2023

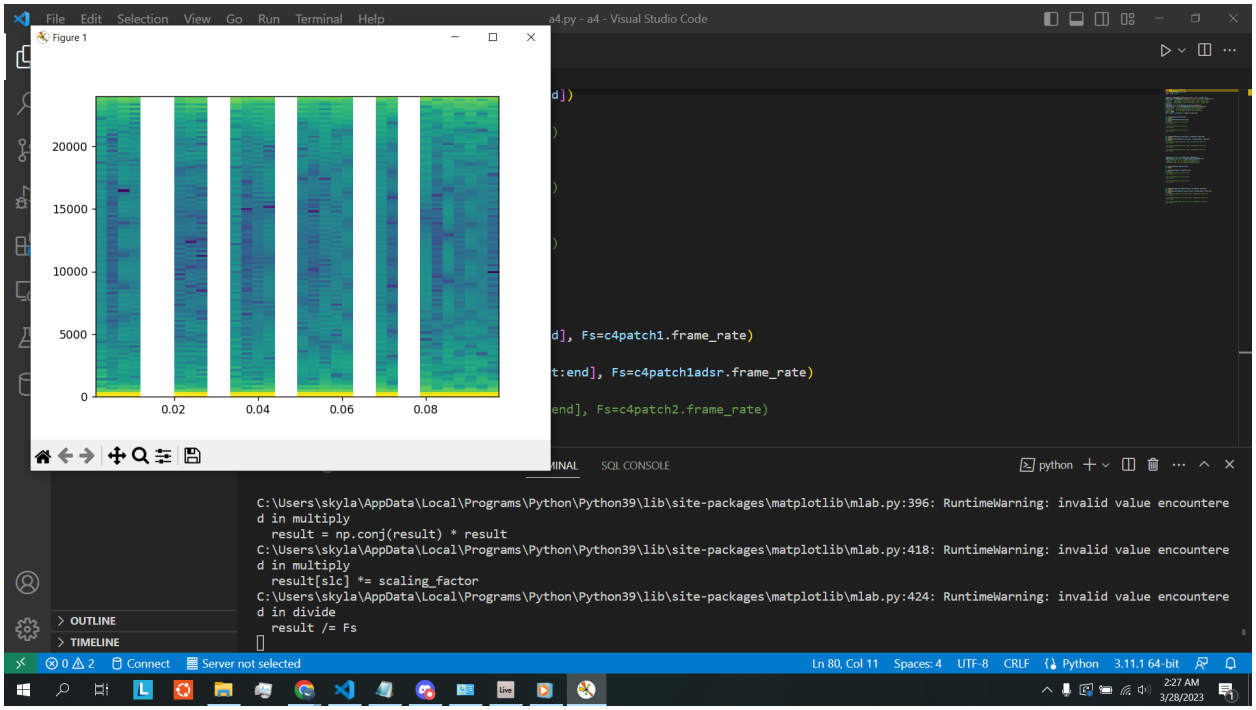


Figure 1
20000
15000
10000
5000
0
0.02
0.04
0.06
0.08
File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
d])
)
)
)
d], Fs=c4patch1.frame_rate)
t:end], Fs=c4patch1adsr.frame_rate)
end], Fs=c4patch2.frame_rate)
SQL CONSOLE
python
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:396: RuntimeWarning: invalid value encountere
d in multiply
result = np.conj(result) * result
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:418: RuntimeWarning: invalid value encountere
d in multiply
result[slc] *= scaling_factor
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:424: RuntimeWarning: invalid value encountere
d in divide
result /= Fs
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 80, Col 11 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:27 AM
3/28/2023

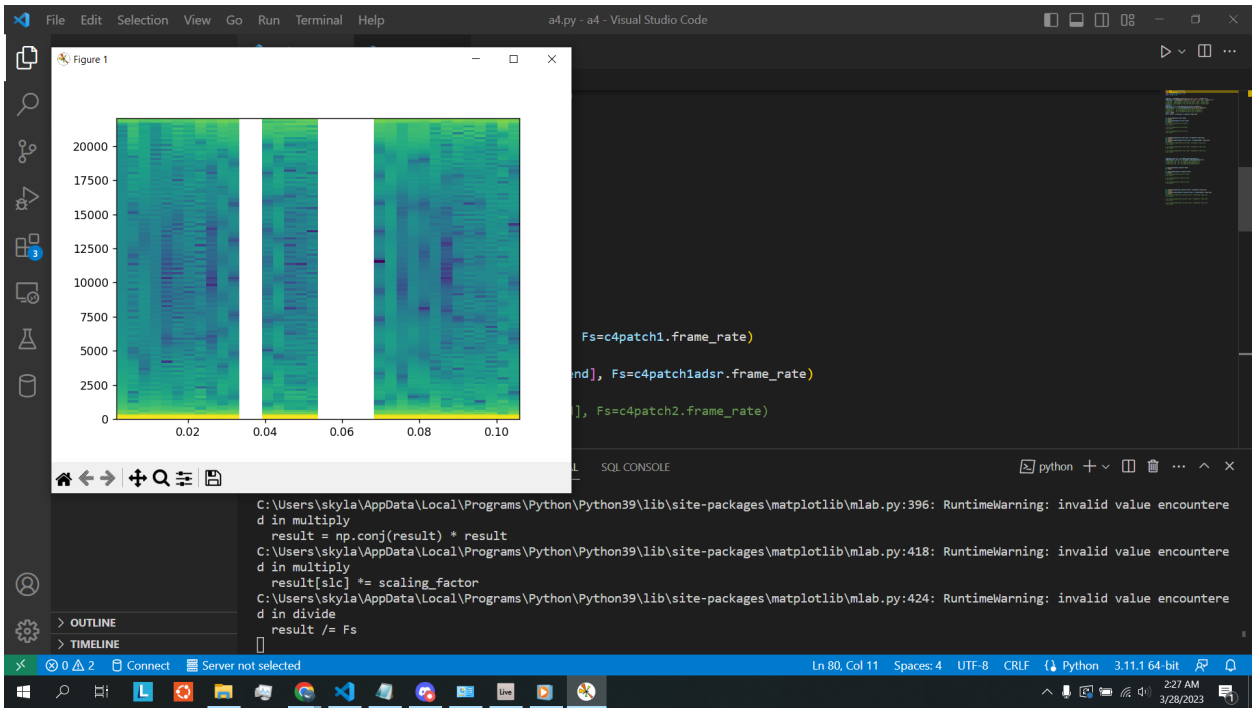
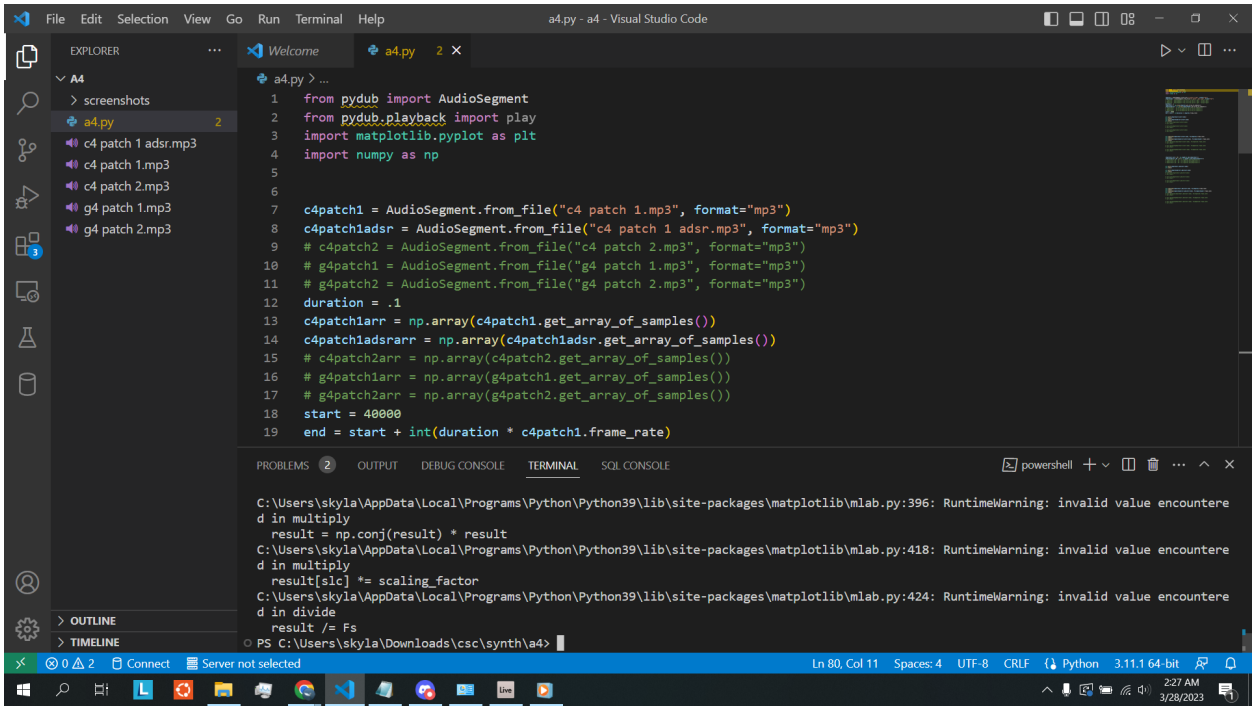
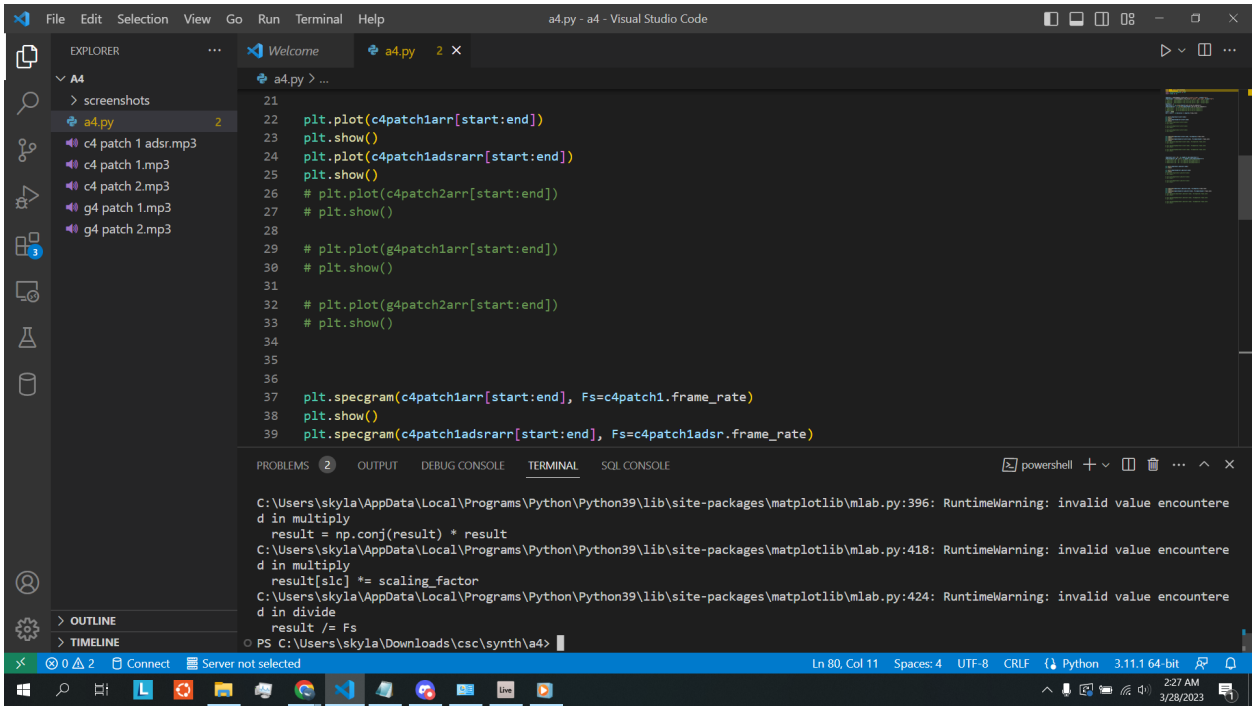
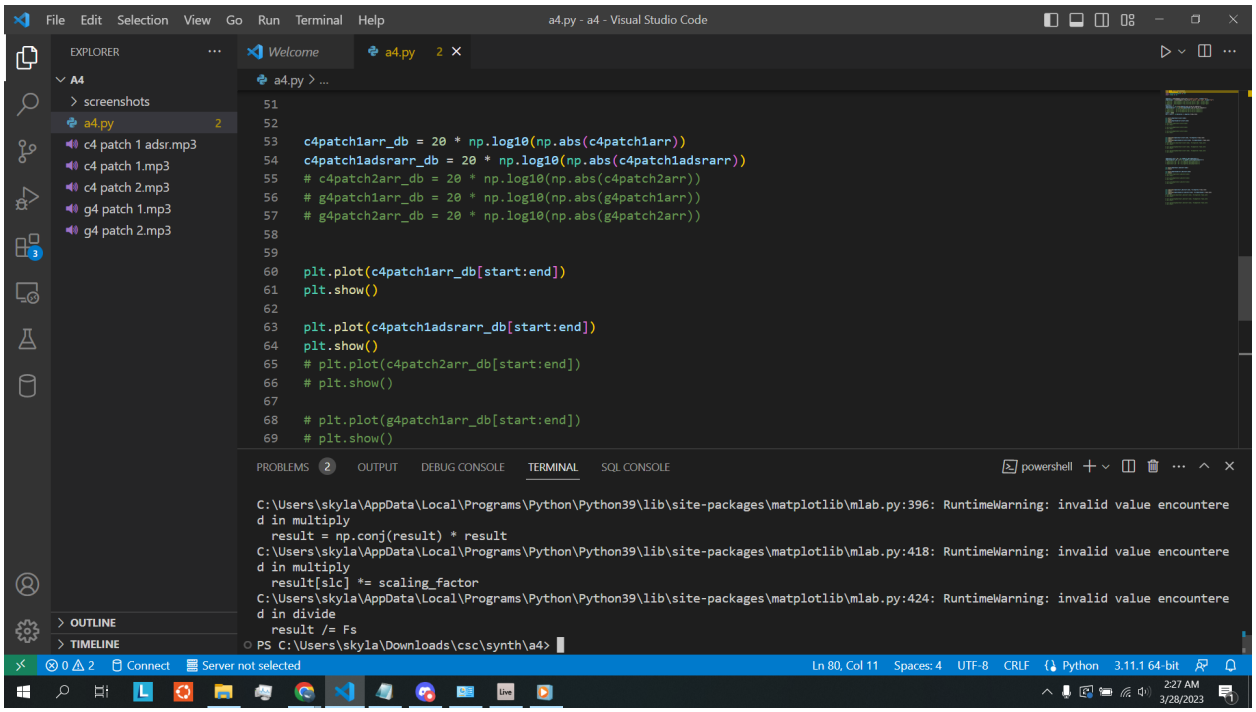


Figure 1
20000
17500
15000
12500
10000
7500
5000
2500
0
0.02
0.04
0.06
0.08
0.10
File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
Fs=c4patch1.frame_rate)
nd], Fs=c4patch1adsr.frame_rate)
], Fs=c4patch2.frame_rate)
SQL CONSOLE
python
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:396: RuntimeWarning: invalid value encountere
d in multiply
result = np.conj(result) * result
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:418: RuntimeWarning: invalid value encountere
d in multiply
result[slc] *= scaling_factor
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:424: RuntimeWarning: invalid value encountere
d in divide
result /= Fs
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 80, Col 11 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:27 AM
3/28/2023

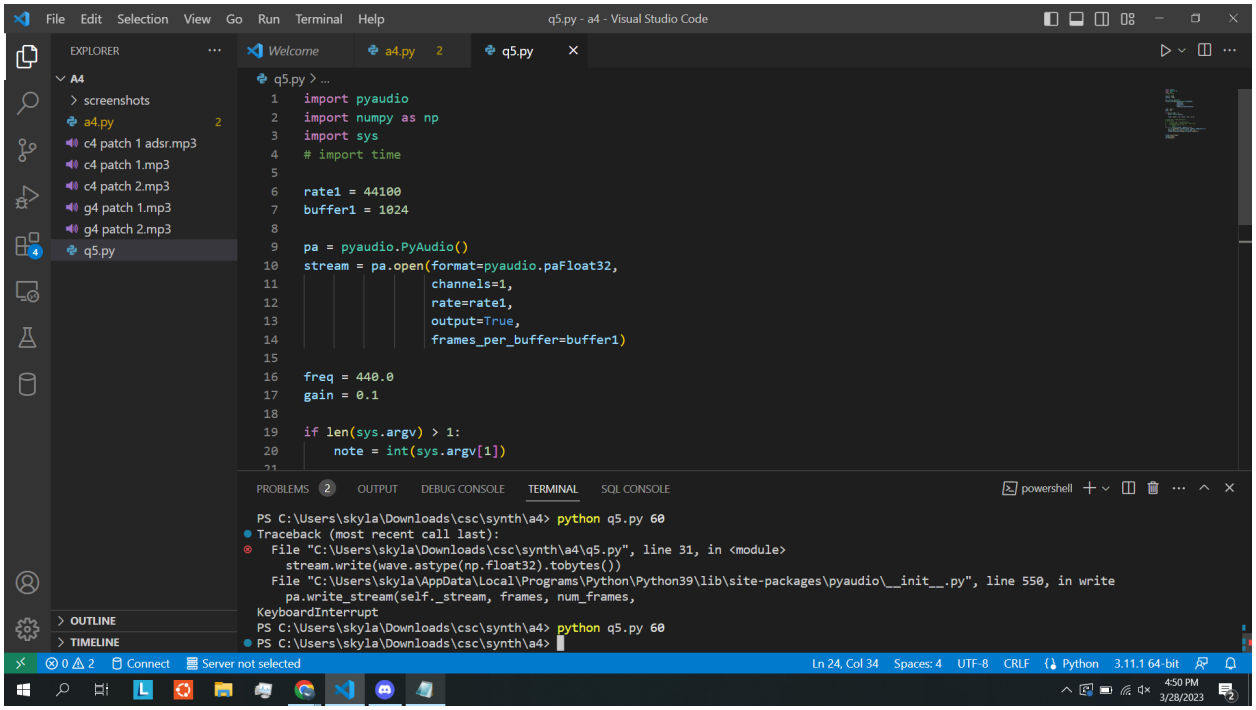

File Edit Selection View Go Run Terminal Help
a4.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch 1 adsr.mp3
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
Welcome
a4.py 2
a4.py > ...
1 from pydub import AudioSegment
2 from pydub.playback import play
3 import matplotlib.pyplot as plt
4 import numpy as np
5
6
7 c4patch1 = AudioSegment.from_file("c4 patch 1.mp3", format="mp3")
8 c4patch1adsr = AudioSegment.from_file("c4 patch 1 adsr.mp3", format="mp3")
9 # c4patch2 = AudioSegment.from_file("c4 patch 2.mp3", format="mp3")
10 # g4patch1 = AudioSegment.from_file("g4 patch 1.mp3", format="mp3")
11 # g4patch2 = AudioSegment.from_file("g4 patch 2.mp3", format="mp3")
12 duration = .1
13 c4patch1arr = np.array(c4patch1.get_array_of_samples())
14 c4patch1adsrarr = np.array(c4patch1adsr.get_array_of_samples())
15 # c4patch2arr = np.array(c4patch2.get_array_of_samples())
16 # g4patch1arr = np.array(g4patch1.get_array_of_samples())
17 # g4patch2arr = np.array(g4patch2.get_array_of_samples())
18 start = 40000
19 end = start + int(duration * c4patch1.frame_rate)
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
powershell
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:396: RuntimeWarning: invalid value encountered in multiply
result = np.conj(result) * result
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:418: RuntimeWarning: invalid value encountered in multiply
result[slc] *= scaling_factor
C:\Users\skyla\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\mlab.py:424: RuntimeWarning: invalid value encountered in divide
result /= Fs
PS C:\Users\skyla\Downloads\csc\synth\a4>
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 80, Col 11 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:27 AM
3/28/2023

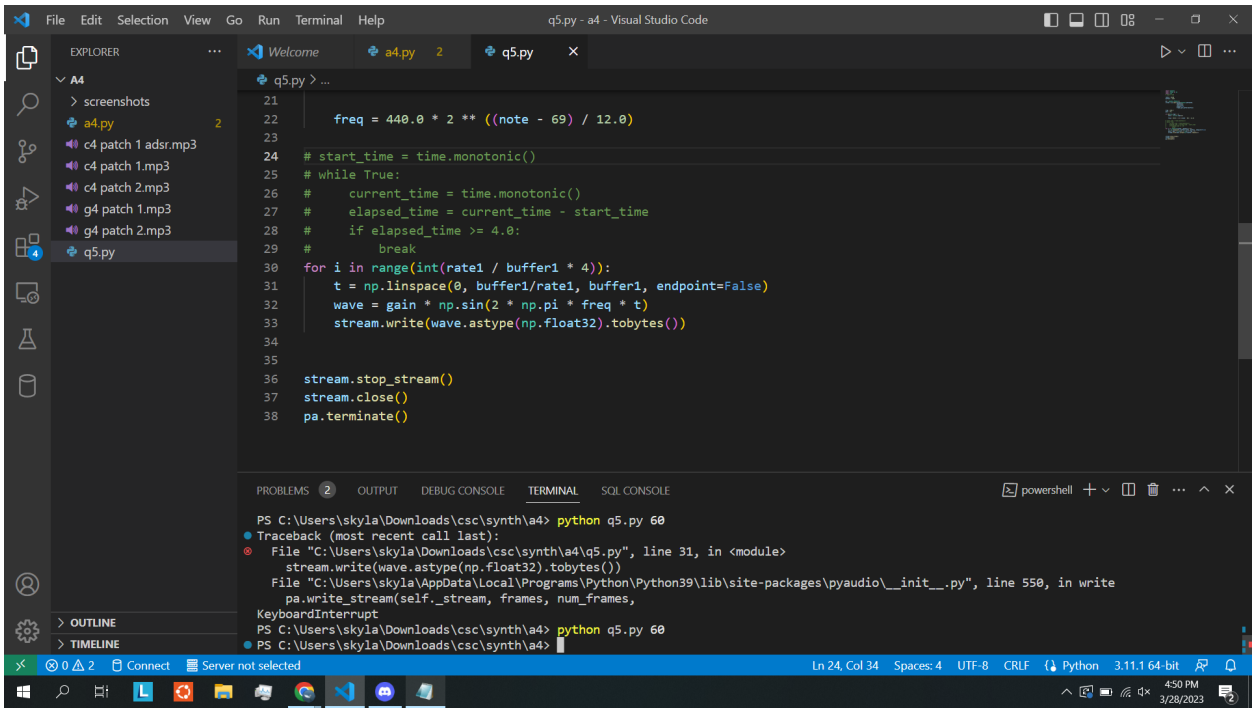




For question 4 I used one of the patches that I had used previously. I kept the algorithm the same but changed the adsr to have a much lower sustain. Using the code from question 2 I can show the time-domain waveform and magnitude spectrum before the changes and after the changes. I used two C4 note sounds from the first patch but with different adsr. I used a very short duration that corresponds to a few pitch cycles of the notes. The graph screenshots show a back and forth cycle of the former C4 note 1 and the C4 note 2 which has a changed adsr. With this cycle you can see, for each note, the time domain plots for linear amplitude, then

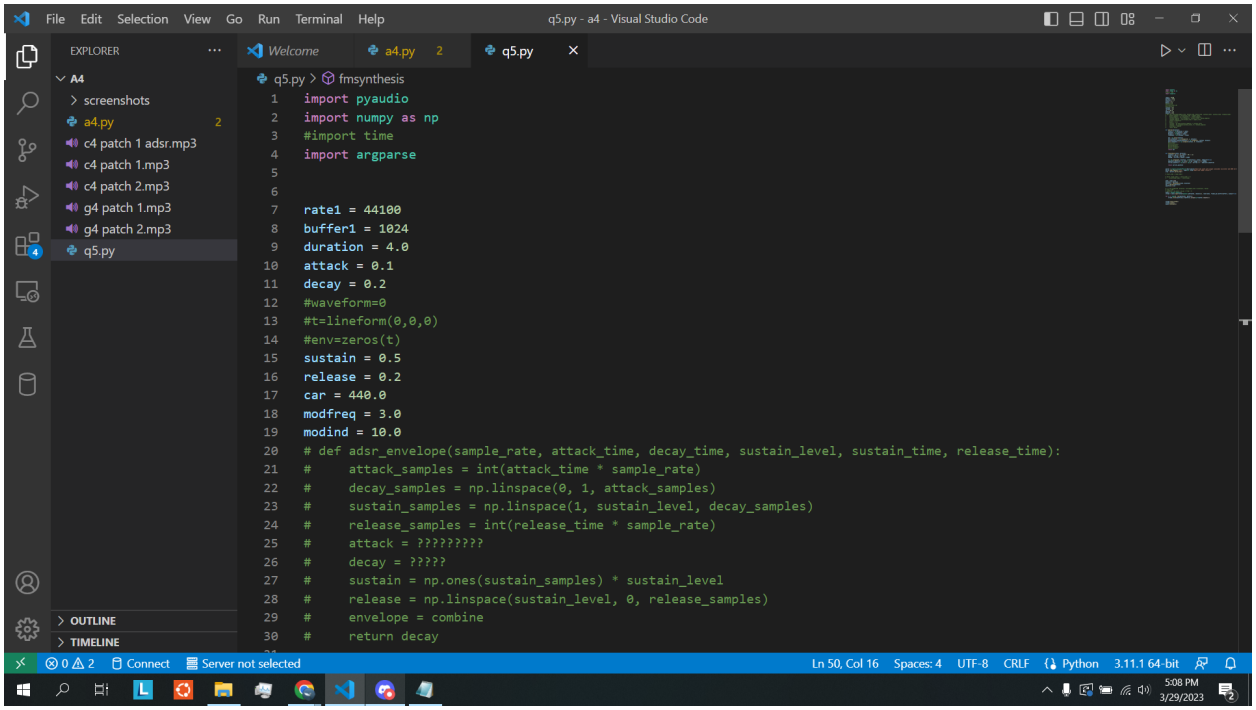
magnitude spectrograms for linear amplitude, then the time domain plots in decibel amplitude, and finally the magnitude spectrograms in decibel amplitude. In terms of the code shown on the other screenshots, the process was to load the mp3 files using pydub and define the short duration of each note in seconds. The short duration I chose to correspond with a few pitch cycles of the notes was .1 seconds. Then after I convert the mp3 files to numpy arrays I can get a starting and ending spot for the sample of the duration I set earlier. After that I can plot the time domain signals using linear amplitude. I can also plot the magnitude spectrograms for linear amplitude as well using ".specgram". Then after converting the amplitude to dB I can plot the time domain signals and magnitude spectrograms using dB amplitude and the same process of utilizing ".plot" and ".specgram".

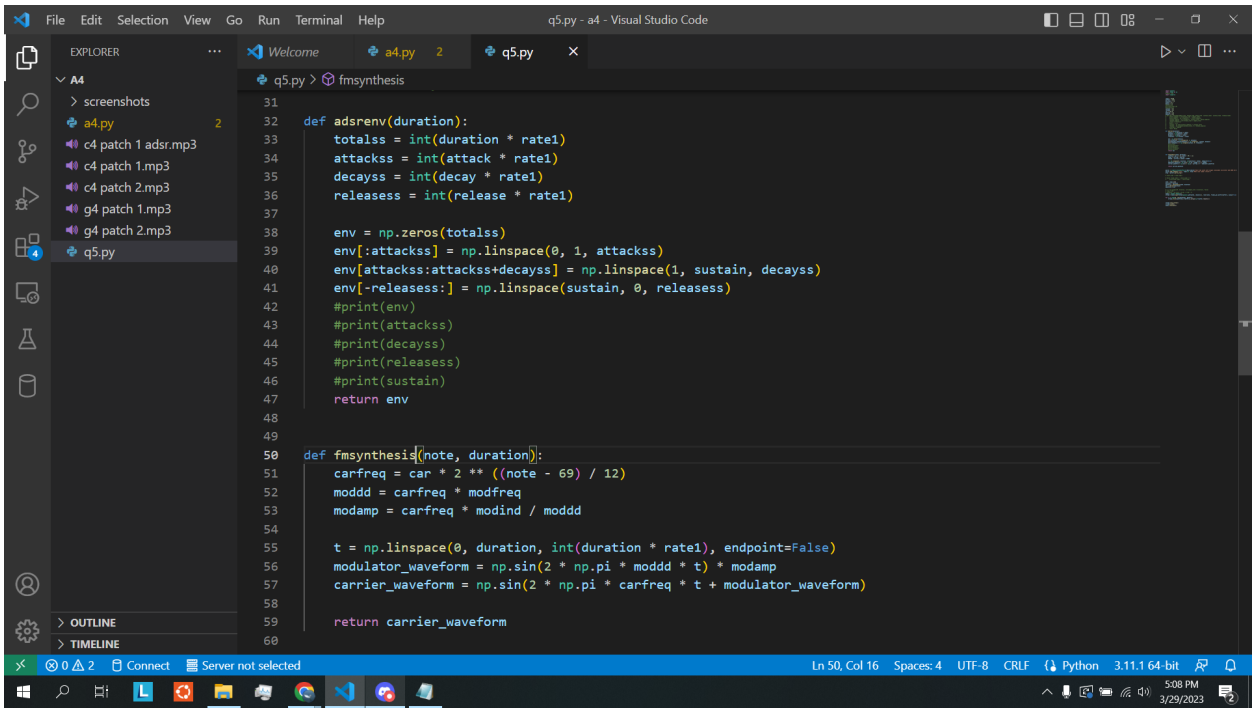


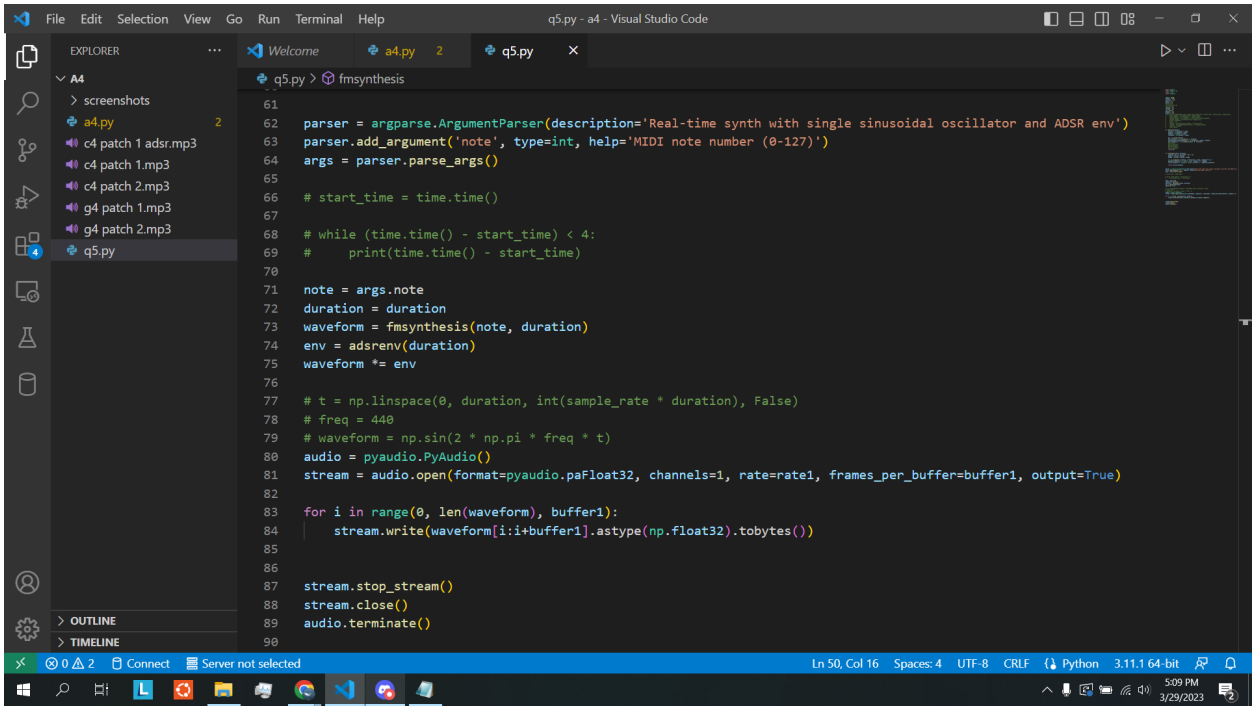


For question 5 I had to create a real-time synth in python. The synth has a single sinusoidal oscillator and takes midi note numbers from the command line as input. Based on the code shown in the code, the process of creating this real-time synth goes as follows. Set up PyAudio stream parameters, the PyAudio stream, and the synth parameters. Initialize the starting frequency and the gain of oscillator. Once all this is complete we can check for command line input and calculate frequency based on MIDI note number. Using this input we generate audio

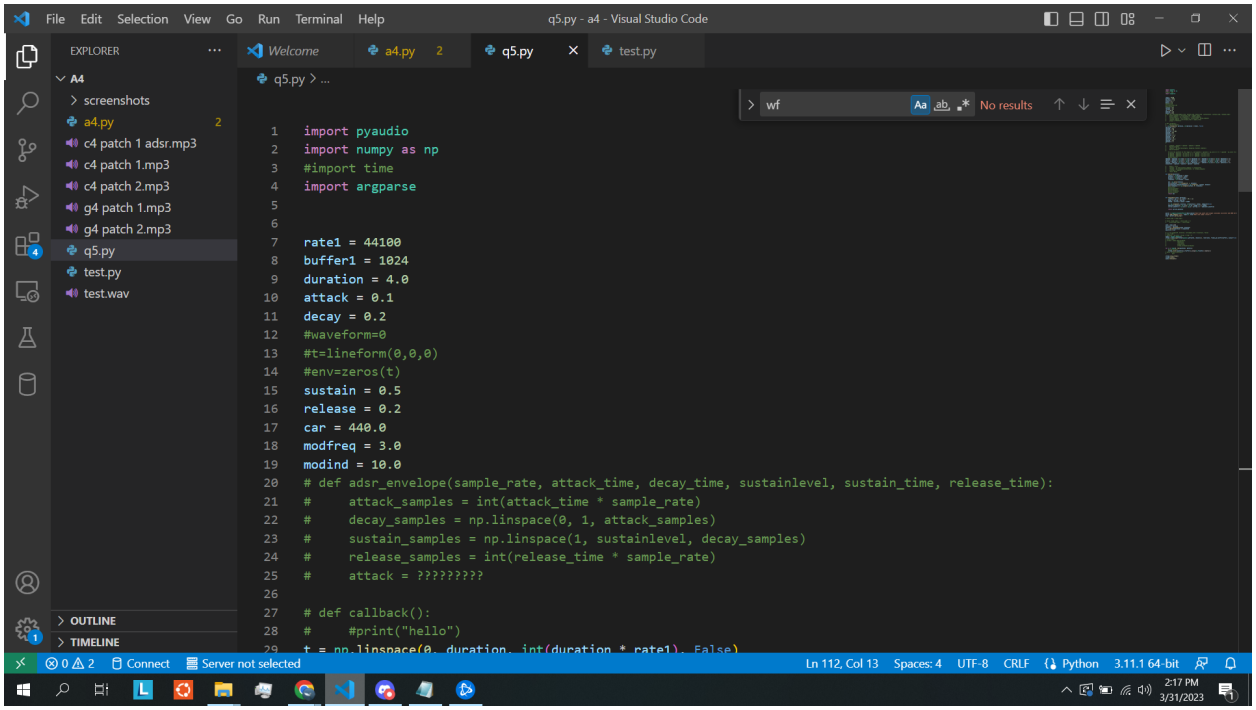
output for x number of seconds and once we are done generating that sound we close PyAudio stream and the program concludes.

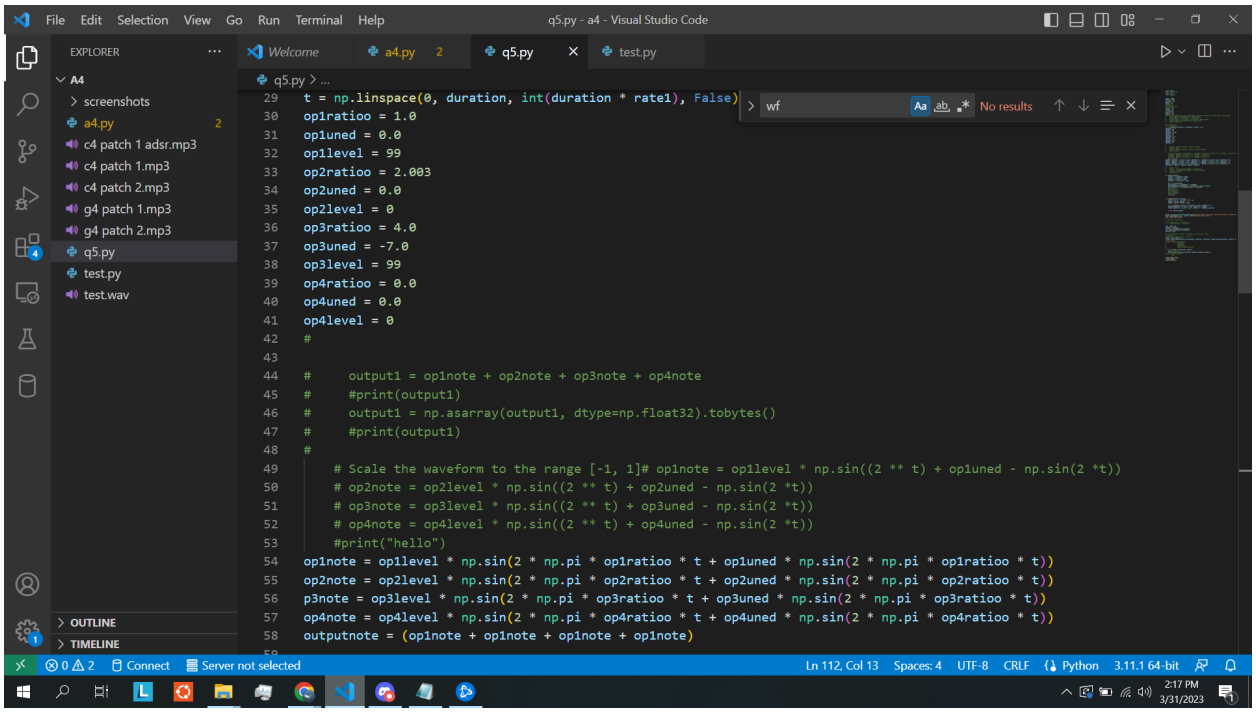

File Edit Selection View Go Run Terminal Help
q5.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch 1 adsr.mp3
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
q5.py
Welcome
a4.py 2
q5.py
q5.py > fmsynthesis
import pyaudio
import numpy as np
#import time
import argparse
rate1 = 44100
buffer1 = 1024
duration = 4.0
attack = 0.1
decay = 0.2
#waveform=0
#t=lineform(0,0,0)
#env=zeros(t)
sustain = 0.5
release = 0.2
car = 440.0
modfreq = 3.0
modind = 10.0
# def adsr_envelope(sample_rate, attack_time, decay_time, sustain_level, sustain_time, release_time):
# attack_samples = int(attack_time * sample_rate)
# decay_samples = np.linspace(0, 1, attack_samples)
# sustain_samples = np.linspace(1, sustain_level, decay_samples)
# release_samples = int(release_time * sample_rate)
# attack = ?????????
# decay = ?????
# sustain = np.ones(sustain_samples) * sustain_level
# release = np.linspace(sustain_level, 0, release_samples)
# envelope = combine
# return decay
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 50, Col 16 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
5:08 PM
3/29/2023

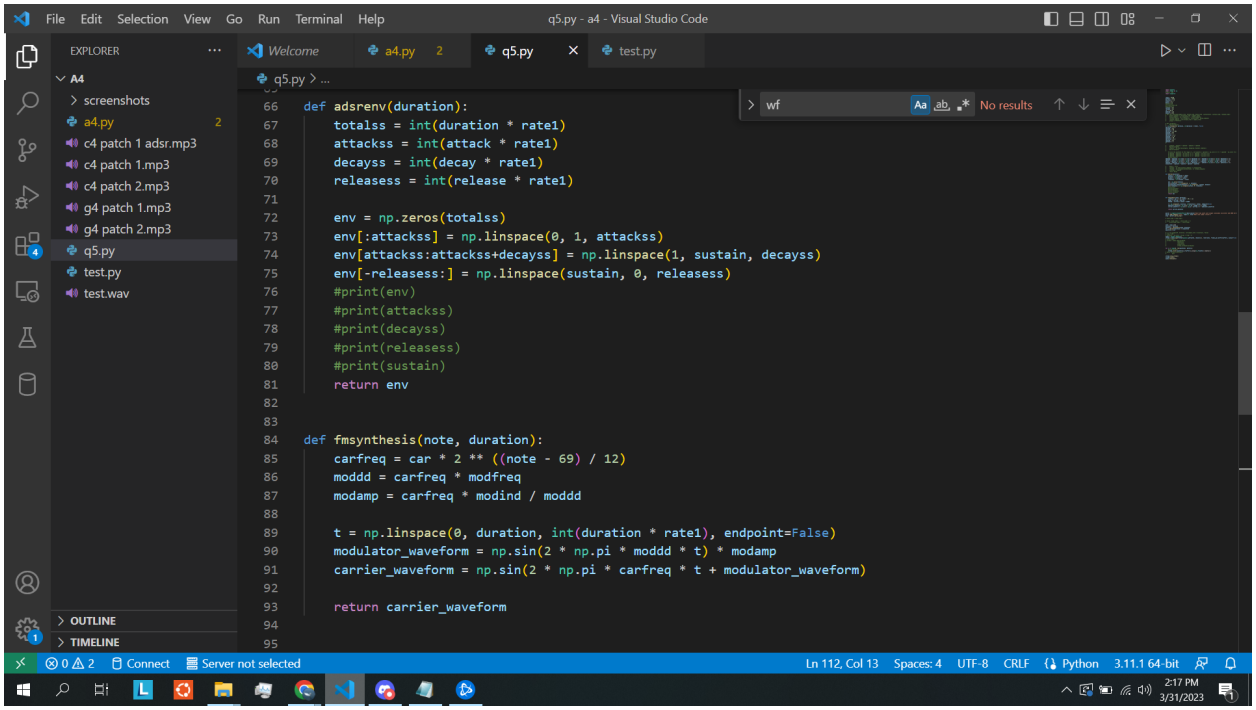

File Edit Selection View Go Run Terminal Help
q5.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch 1 adsr.mp3
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
q5.py
Welcome
a4.py 2
q5.py
q5.py > fmsynthesis
def adsrenv(duration):
totalss = int(duration * rate1)
attackss = int(attack * rate1)
decayss = int(decay * rate1)
releasess = int(release * rate1)
env = np.zeros(totalss)
env[:attackss] = np.linspace(0, 1, attackss)
env[attackss:attackss+decayss] = np.linspace(1, sustain, decayss)
env[-releasess:] = np.linspace(sustain, 0, releasess)
#print(env)
#print(attackss)
#print(decayss)
#print(releasess)
#print(sustain)
return env
def fmsynthesis(note, duration):
carfreq = car * 2 ** ((note - 69) / 12)
moddd = carfreq * modfreq
modamp = carfreq * modind / moddd
t = np.linspace(0, duration, int(duration * rate1), endpoint=False)
modulator_waveform = np.sin(2 * np.pi * moddd * t) * modamp
carrier_waveform = np.sin(2 * np.pi * carfreq * t + modulator_waveform)
return carrier_waveform
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 50, Col 16 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
5:08 PM
3/29/2023

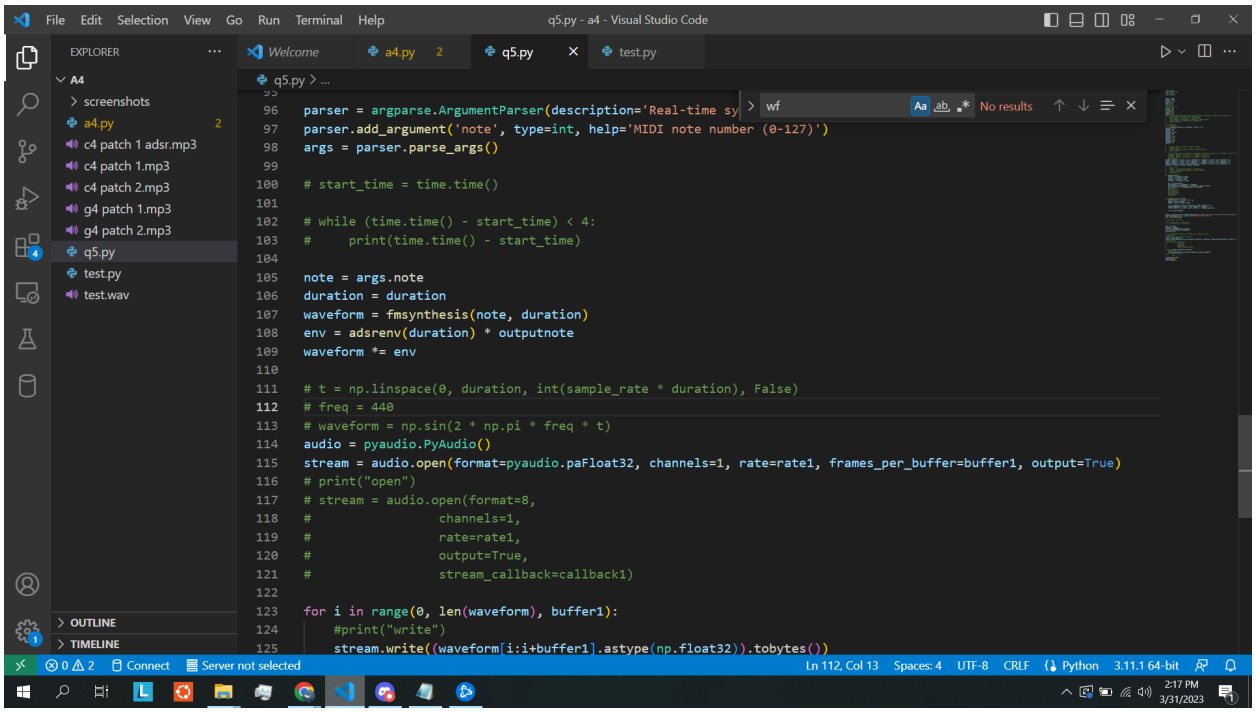


For question 6 I had to extend the synthesizer from the previous question to support an ADSR envelope and two-operator FM synthesis. In order to do this we first have to calculate the number of samples for each stage of the envelope using the predetermined time duration and the sample rate. Next, we generate arrays for each stage of the envelope using the numpy.linspace() function. In the case of the attack and release stages, we generate arrays that go from 0 to 1 and from 1 to 0. In the case of the decay stage, we generate an array that goes from 1 to the sustain level. Finally, in the case of the sustain stage, we generate an array of constant values equal to the sustain level. We then concatenate the four arrays into a single envelope array using the numpy.concatenate() function. Finally, we return the envelope array. When applied to an audio waveform, the ADSR envelope generated by this function will shape the amplitude of the waveform over time, producing a characteristic "pluck" or "punch" sound. For fm synthesis we first have to calculate the number of samples for the waveform using the provided duration and sample rate. We then create arrays for the carrier and modulator waveforms using the numpy.linspace() function. In this case, we're generating a wave for both the carrier and modulator waveforms. Next, we calculate the modulation index, which determines the amount of frequency modulation applied to the carrier waveform. We then apply the frequency modulation to the carrier waveform by multiplying it with the modulation index. Finally, we return the modulated carrier waveform.


File Edit Selection View Go Run Terminal Help
q5.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch 1 adsr.mp3
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
q5.py
test.py
test.wav
Welcome
a4.py 2
q5.py
test.py
wf
No results
import pyaudio
import numpy as np
#import time
import argparse
rate1 = 44100
buffer1 = 1024
duration = 4.0
attack = 0.1
decay = 0.2
#waveform=0
#t=lineform(0,0,0)
#env=zeros(t)
sustain = 0.5
release = 0.2
car = 440.0
modfreq = 3.0
modind = 10.0
# def adsr_envelope(sample_rate, attack_time, decay_time, sustainlevel, sustain_time, release_time):
# attack_samples = int(attack_time * sample_rate)
# decay_samples = np.linspace(0, 1, attack_samples)
# sustain_samples = np.linspace(1, sustainlevel, decay_samples)
# release_samples = int(release_time * sample_rate)
# attack = ?????????
# def callback():
# #print("hello")
t = np.linspace(0, duration, int(duration * rate1), False)
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 112, Col 13 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:17 PM
3/31/2023

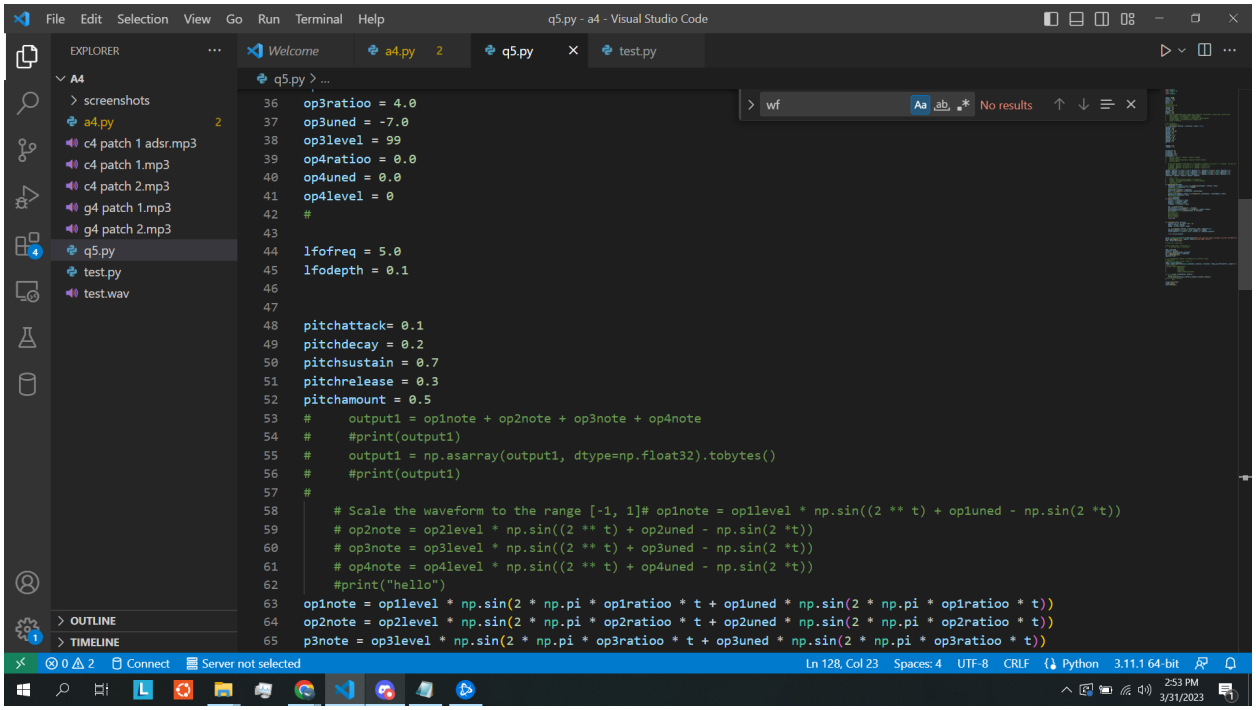

File Edit Selection View Go Run Terminal Help
q5.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch 1 adsr.mp3
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
q5.py
test.py
test.wav
Welcome
a4.py 2
q5.py
test.py
wf
No results
t = np.linspace(0, duration, int(duration * rate1), False)
op1ratioo = 1.0
op1uned = 0.0
op1level = 99
op2ratioo = 2.003
op2uned = 0.0
op2level = 0
op3ratioo = 4.0
op3uned = -7.0
op3level = 99
op4ratioo = 0.0
op4uned = 0.0
op4level = 0
# output1 = op1note + op2note + op3note + op4note
# #print(output1)
# output1 = np.asarray(output1, dtype=np.float32).tobytes()
# #print(output1)
# # Scale the waveform to the range [-1, 1]# op1note = op1level * np.sin((2 ** t) + op1uned - np.sin(2 *t))
# op2note = op2level * np.sin((2 ** t) + op2uned - np.sin(2 *t))
# op3note = op3level * np.sin((2 ** t) + op3uned - np.sin(2 *t))
# op4note = op4level * np.sin((2 ** t) + op4uned - np.sin(2 *t))
#print("hello")
op1note = op1level * np.sin(2 * np.pi * op1ratioo * t + op1uned * np.sin(2 * np.pi * op1ratioo * t))
op2note = op2level * np.sin(2 * np.pi * op2ratioo * t + op2uned * np.sin(2 * np.pi * op2ratioo * t))
p3note = op3level * np.sin(2 * np.pi * op3ratioo * t + op3uned * np.sin(2 * np.pi * op3ratioo * t))
op4note = op4level * np.sin(2 * np.pi * op4ratioo * t + op4uned * np.sin(2 * np.pi * op4ratioo * t))
outputnote = (op1note + op1note + op1note + op1note)
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 112, Col 13 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:17 PM
3/31/2023

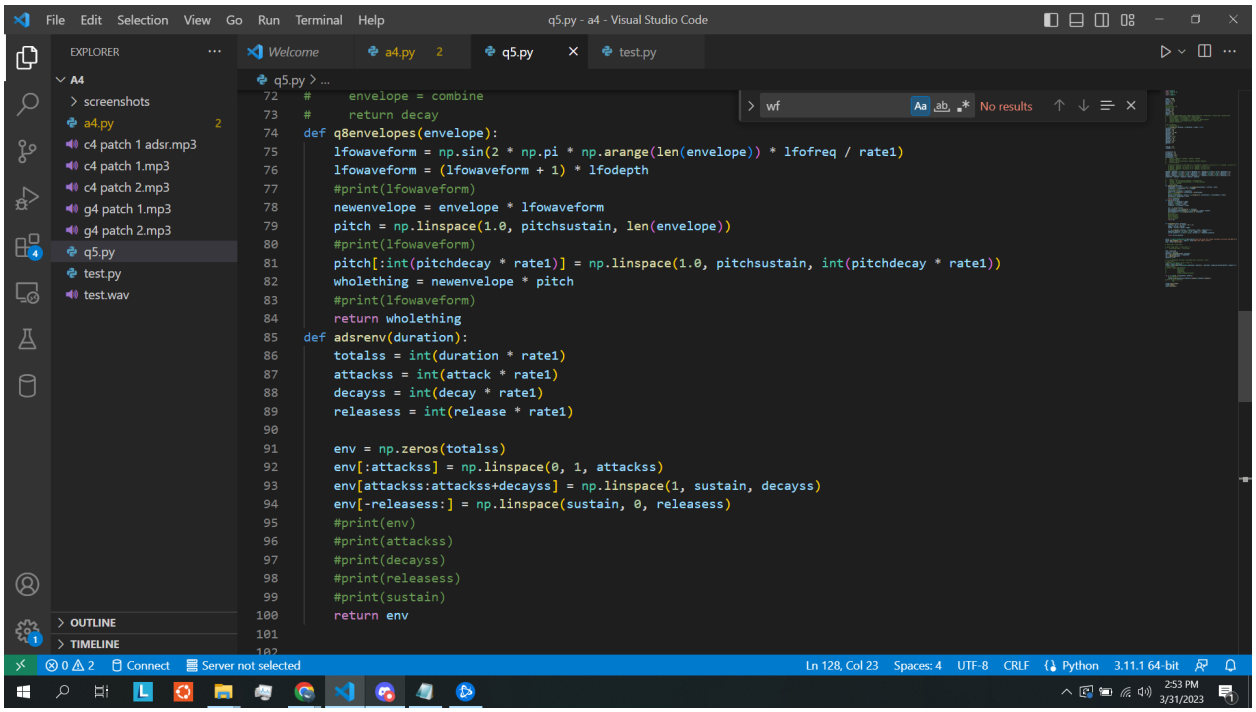


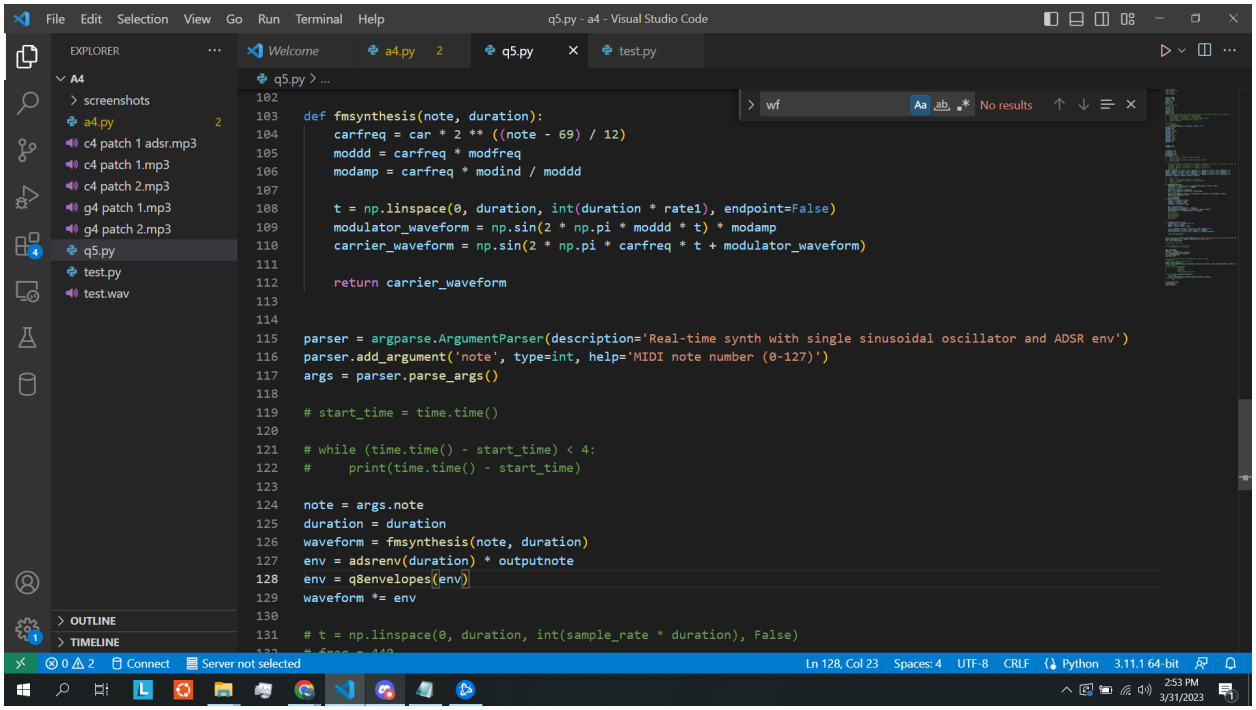


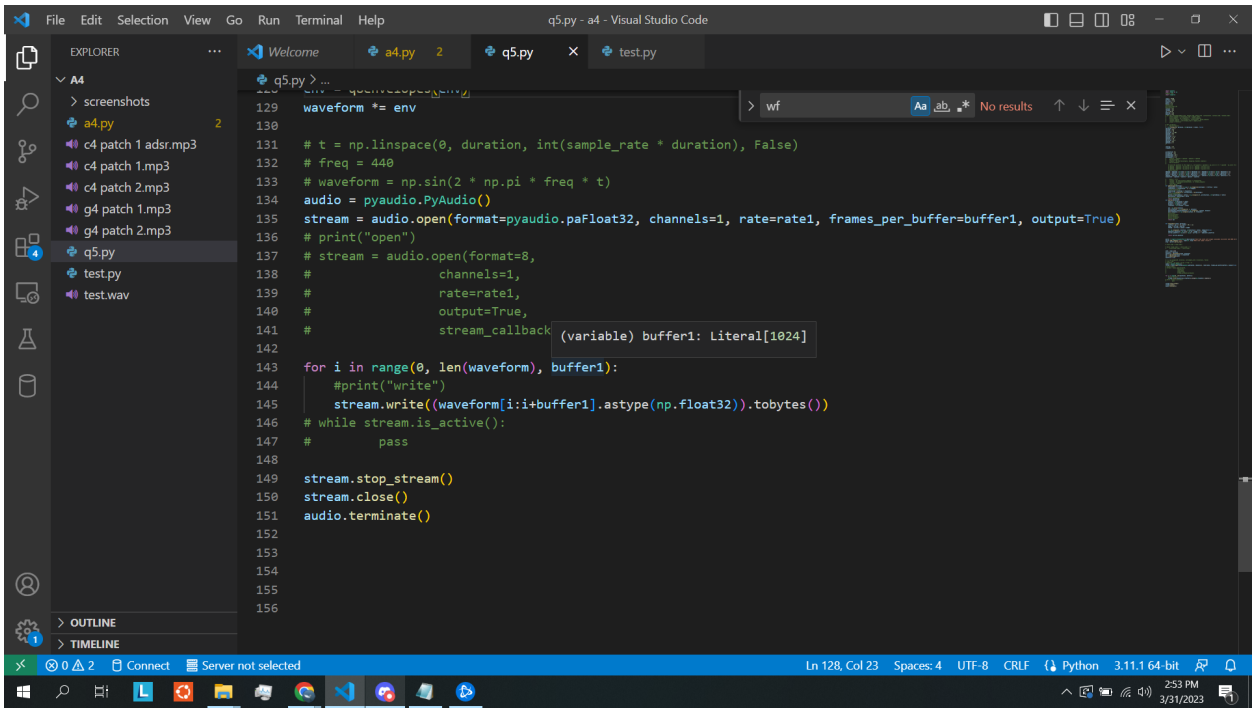
For question 7 we have to implement one of the algorithms and try to approximate one patch sound. I had to extend the implementation to 4 operators and also implement the frequency ratios and detune parameters but ignore the LFO and Pitch envelopes. I started by defining the parameters for each operator, making sure to include the frequency ratio, detune, and amplitude modulation parameters. The waves that are generated are modulated and the amplitude modulation is multiplied with the wave using a modulating waveform. Next, I combined the waves for each operator by summing the waves together. The amplitude of each operator is

multiplied by a scaling factor to avoid clipping and ensure that the output is within the range of -1.0 to 1.0. I then normalized the output to make sure the range is correct. After normalizing the output, the rest of the code runs making the calculated audio that has the adsr envelope and the patch run for the designated number of seconds. The patch that is being replicated in this program is a classic DX21 bass sound. While searching online this patch is characterized by a rich, warm tone with a fast attack and moderate decay. The sound is created using four operators, with the first two operators set to a 1:1 frequency ratio, and the third and fourth operators set to a 3:2 frequency ratio. The detune parameter is set to create a slight detuning effect, which adds depth to the sound. The sound is then shaped using a combination of envelope generators and amplitude modulation. In the original DX21 patch, the envelope generators are used to shape the pitch and amplitude of the sound. However, in this implementation, only the amplitude envelope generator is used to shape the amplitude of the sound.


File Edit Selection View Go Run Terminal Help
q5.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch 1 adsr.mp3
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
q5.py
test.py
test.wav
Welcome
a4.py 2
q5.py
test.py
wf
No results
op3ratioo = 4.0
op3uned = -7.0
op3level = 99
op4ratioo = 0.0
op4uned = 0.0
op4level = 0
lfofreq = 5.0
lfodepth = 0.1
pitchattack= 0.1
pitchdecay = 0.2
pitchsustain = 0.7
pitchrelease = 0.3
pitchamount = 0.5
output1 = op1note + op2note + op3note + op4note
#print(output1)
output1 = np.asarray(output1, dtype=np.float32).tobytes()
#print(output1)
Scale the waveform to the range [-1, 1]# op1note = op1level * np.sin((2 ** t) + op1uned - np.sin(2 *t))
op2note = op2level * np.sin((2 ** t) + op2uned - np.sin(2 *t))
op3note = op3level * np.sin((2 ** t) + op3uned - np.sin(2 *t))
op4note = op4level * np.sin((2 ** t) + op4uned - np.sin(2 *t))
#print("hello")
op1note = op1level * np.sin(2 * np.pi * op1ratioo * t + op1uned * np.sin(2 * np.pi * op1ratioo * t))
op2note = op2level * np.sin(2 * np.pi * op2ratioo * t + op2uned * np.sin(2 * np.pi * op2ratioo * t))
p3note = op3level * np.sin(2 * np.pi * op3ratioo * t + op3uned * np.sin(2 * np.pi * op3ratioo * t))
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 128, Col 23 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:53 PM 3/31/2023

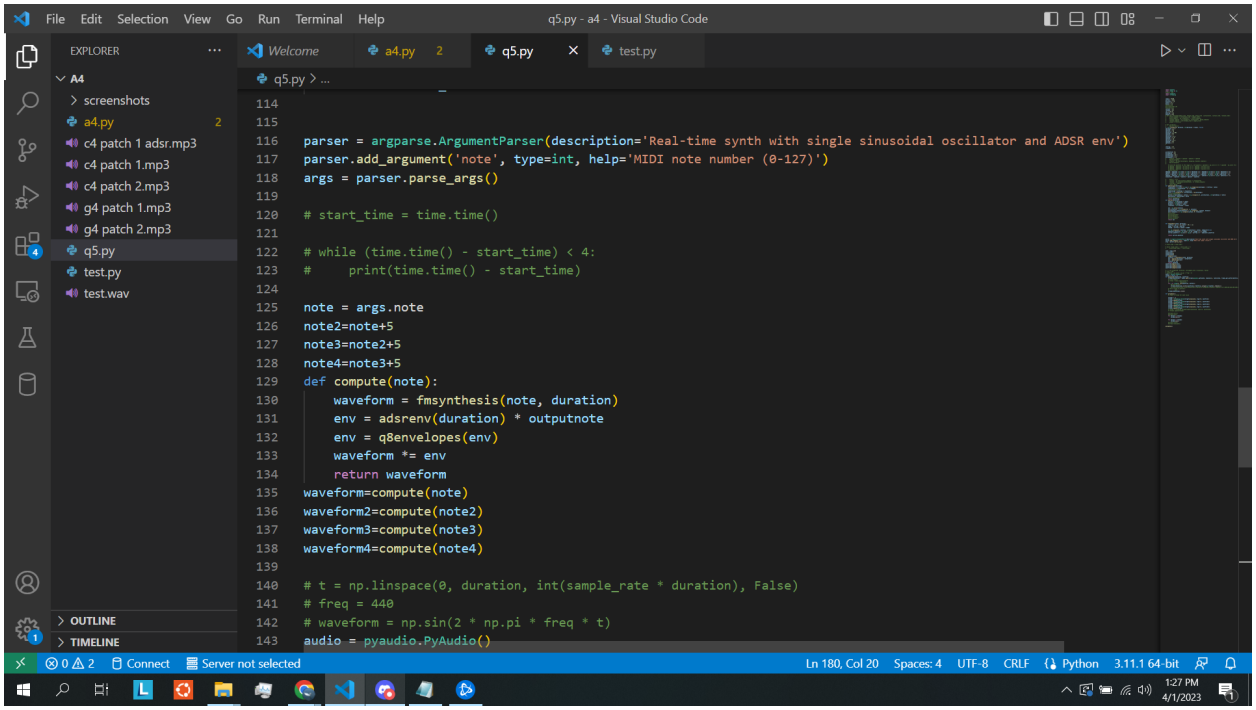

File Edit Selection View Go Run Terminal Help
q5.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch 1 adsr.mp3
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
q5.py
test.py
test.wav
Welcome
a4.py 2
q5.py
test.py
wf
No results
envelope = combine
return decay
def q8envelopes(envelope):
lfowaveform = np.sin(2 * np.pi * np.arange(len(envelope)) * lfofreq / rate1)
lfowaveform = (lfowaveform + 1) * lfodepth
#print(lfowaveform)
newenvelope = envelope * lfowaveform
pitch = np.linspace(1.0, pitchsustain, len(envelope))
#print(lfowaveform)
pitch[:int(pitchdecay * rate1)] = np.linspace(1.0, pitchsustain, int(pitchdecay * rate1))
wholething = newenvelope * pitch
#print(lfowaveform)
return wholething
def adsrenv(duration):
totalss = int(duration * rate1)
attackss = int(attack * rate1)
decayss = int(decay * rate1)
releasess = int(release * rate1)
env = np.zeros(totalss)
env[:attackss] = np.linspace(0, 1, attackss)
env[attackss:attackss+decayss] = np.linspace(1, sustain, decayss)
env[-releasess:] = np.linspace(sustain, 0, releasess)
#print(env)
#print(attackss)
#print(decayss)
#print(releasess)
#print(sustain)
return env
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 128, Col 23 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:53 PM 3/31/2023

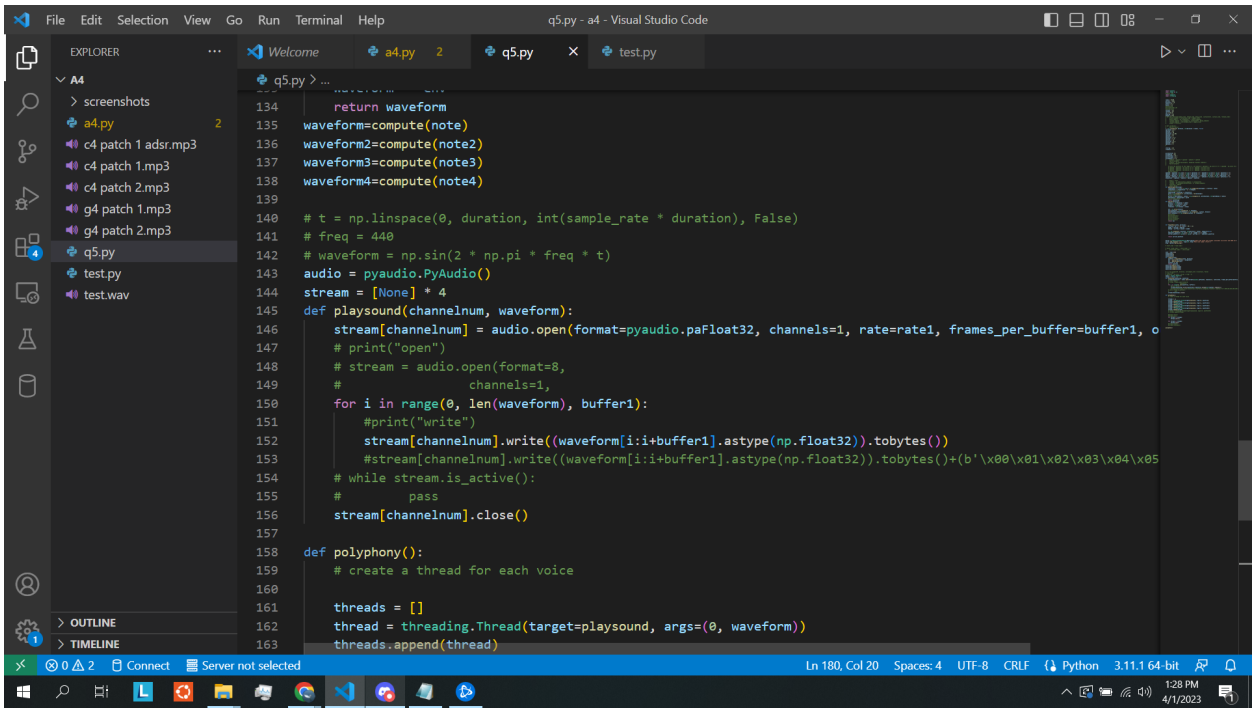


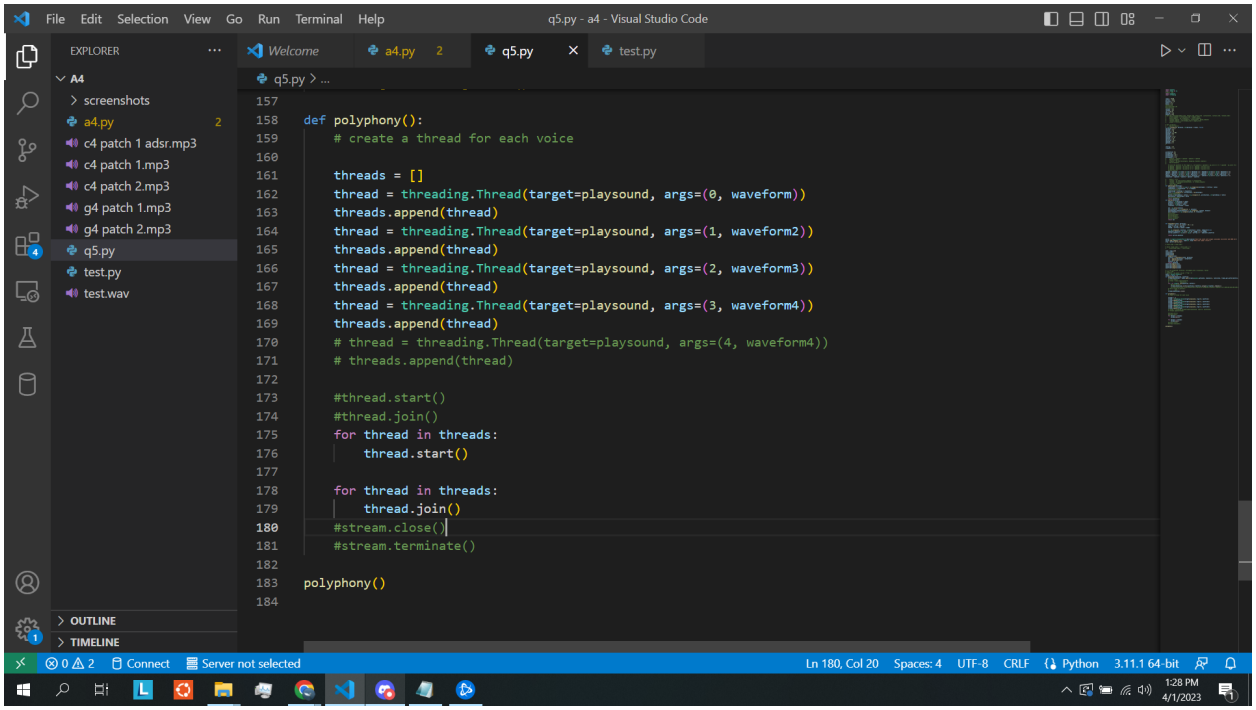


For question 8 I had to add the LFO and pitch envelopes to the implementation. The synthesizer is still mono-phonic, but can still approximate the existing patch.I added LFO and pitch envelopes to an existing envelope using a single function. The input parameters are the envelope that I am adding information to. The rest of the needed information is declared/initialized elsewhere. I first calculated the LFO waveform as a wave with the specified frequency and depth. I then multiplied this waveform with the input envelope to apply the LFO modulation. Next, I generate a pitch envelope as a linear ramp from 1.0 to the specified sustain

level. The first decay time seconds of the envelope are used for the ramp. The pitch envelope is then multiplied with the LFO-modulated envelope to apply the pitch modulation. Finally, the modified envelope with both LFO and pitch envelopes applied is returned. This function is useful for creating software synthesizers, sound effects, or other audio processing tasks, and can make the resulting sound more expressive and dynamic. It is used to create a similar sound to a specific patch from the Yamaha DX21 synthesizer. The replicated patch is talked about in the previous question report but LFO and Pitch Envelopes are powerful tools that can be used to add movement, variation, and complexity to a DX21 synthesizer patch. By experimenting with different settings and combinations, you can create a wide range of sounds that are more unique and expressive than what was possible with the code without these two elements

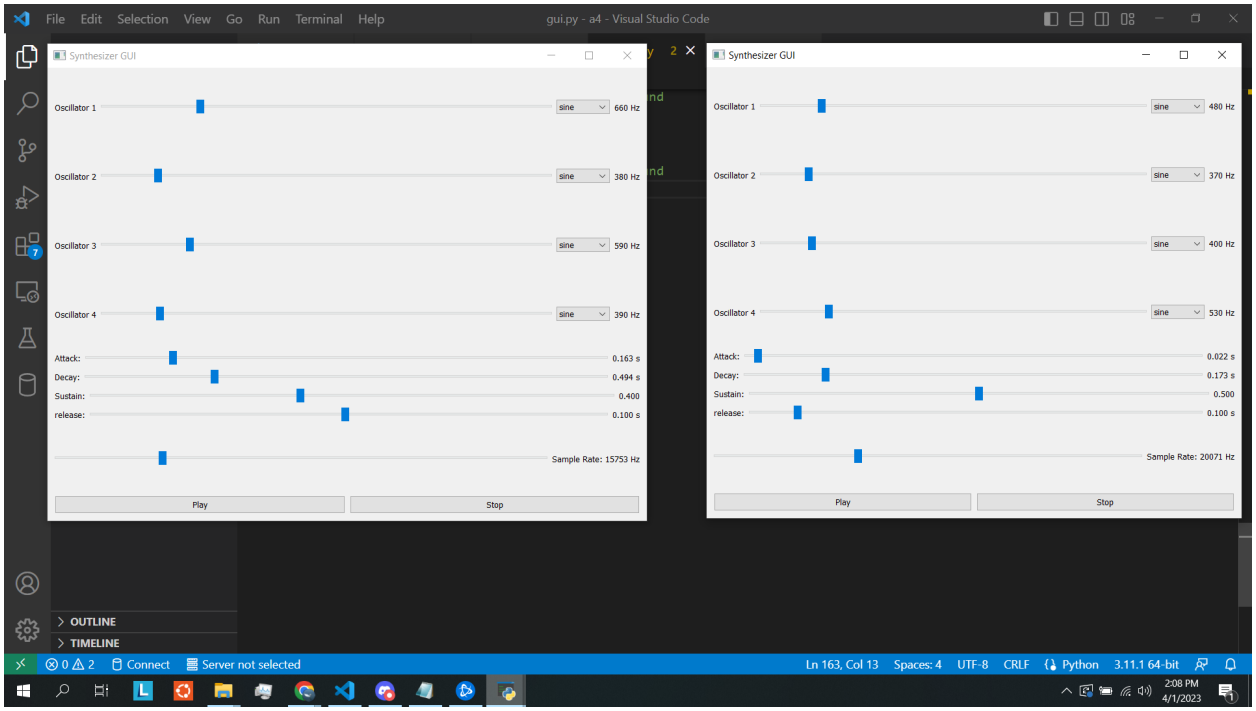

File Edit Selection View Go Run Terminal Help
q5.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch 1 adsr.mp3
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
q5.py
test.py
test.wav
OUTLINE
TIMELINE
Welcome
a4.py 2
q5.py
test.py
q5.py > ...
parser = argparse.ArgumentParser(description='Real-time synth with single sinusoidal oscillator and ADSR env')
parser.add_argument('note', type=int, help='MIDI note number (0-127)')
args = parser.parse_args()
# start_time = time.time()
# while (time.time() - start_time) < 4:
# print(time.time() - start_time)
note = args.note
note2=note+5
note3=note2+5
note4=note3+5
def compute(note):
waveform = fmsynthesis(note, duration)
env = adsrenv(duration) * outputnote
env = q8envelopes(env)
waveform *= env
return waveform
waveform=compute(note)
waveform2=compute(note2)
waveform3=compute(note3)
waveform4=compute(note4)
# t = np.linspace(0, duration, int(sample_rate * duration), False)
# freq = 440
# waveform = np.sin(2 * np.pi * freq * t)
audio = pyaudio.PyAudio()
0 2 Connect Server not selected
Ln 180, Col 20 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
1:27 PM
4/1/2023

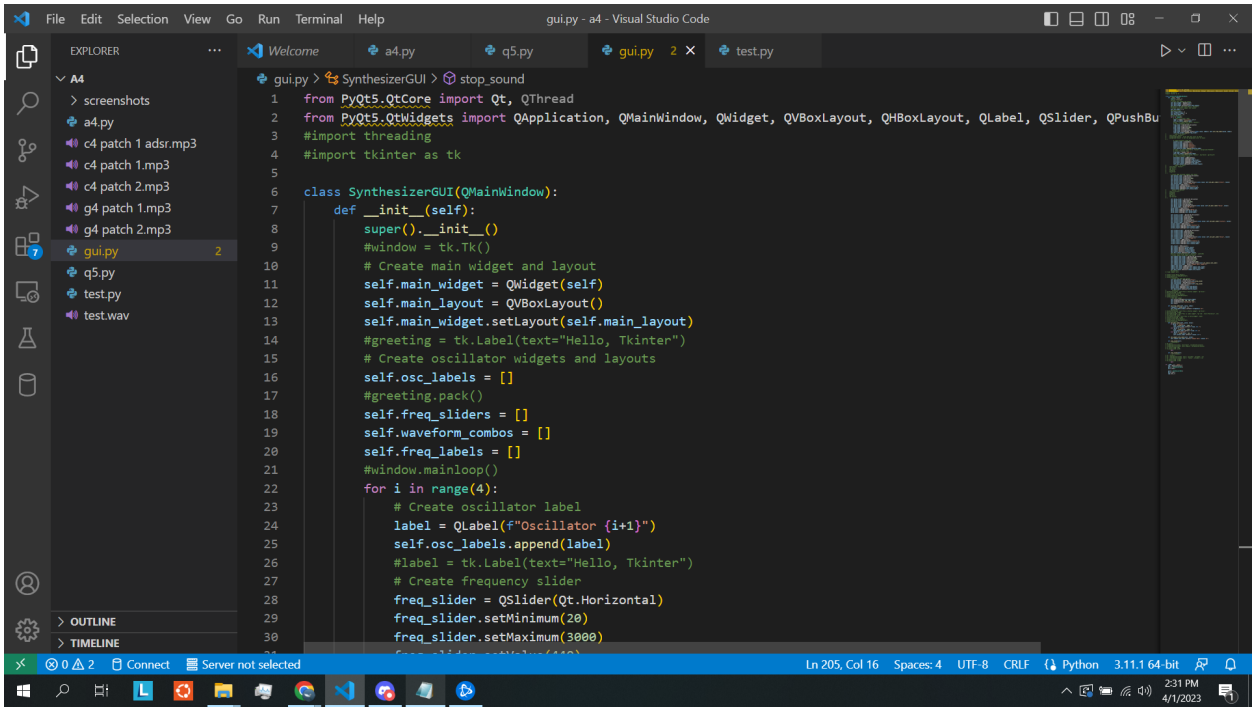

File Edit Selection View Go Run Terminal Help
q5.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py 2
c4 patch 1 adsr.mp3
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
q5.py
test.py
test.wav
OUTLINE
TIMELINE
Welcome
a4.py 2
q5.py
test.py
q5.py > ...
return waveform
waveform=compute(note)
waveform2=compute(note2)
waveform3=compute(note3)
waveform4=compute(note4)
# t = np.linspace(0, duration, int(sample_rate * duration), False)
# freq = 440
# waveform = np.sin(2 * np.pi * freq * t)
audio = pyaudio.PyAudio()
stream = [None] * 4
def playsound(channelnum, waveform):
stream[channelnum] = audio.open(format=pyaudio.paFloat32, channels=1, rate=rate1, frames_per_buffer=buffer1, o
# print("open")
# stream = audio.open(format=8,
# channels=1,
for i in range(0, len(waveform), buffer1):
#print("write")
stream[channelnum].write((waveform[i:i+buffer1].astype(np.float32)).tobytes())
#stream[channelnum].write((waveform[i:i+buffer1].astype(np.float32)).tobytes()+(b'\x00\x01\x02\x03\x04\x05
while stream.is_active():
pass
stream[channelnum].close()
def polyphony():
# create a thread for each voice
threads = []
thread = threading.Thread(target=playsound, args=(0, waveform))
threads.append(thread)
0 2 Connect Server not selected
Ln 180, Col 20 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
1:28 PM
4/1/2023

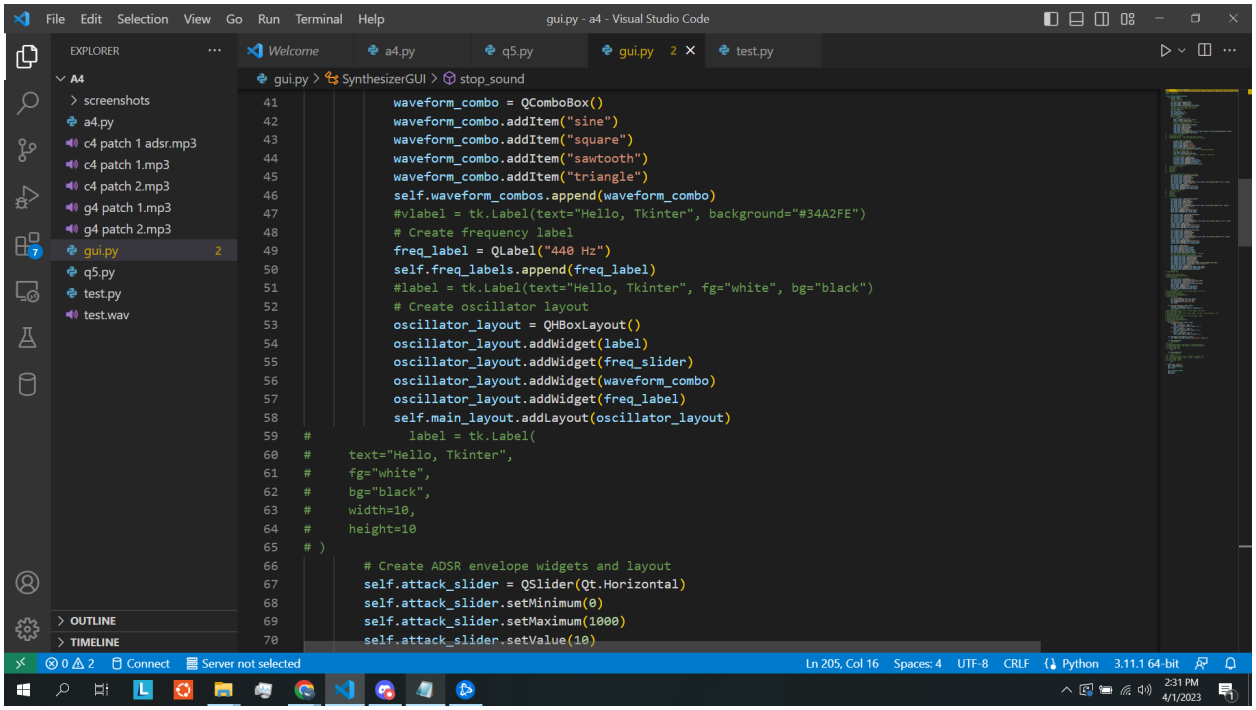


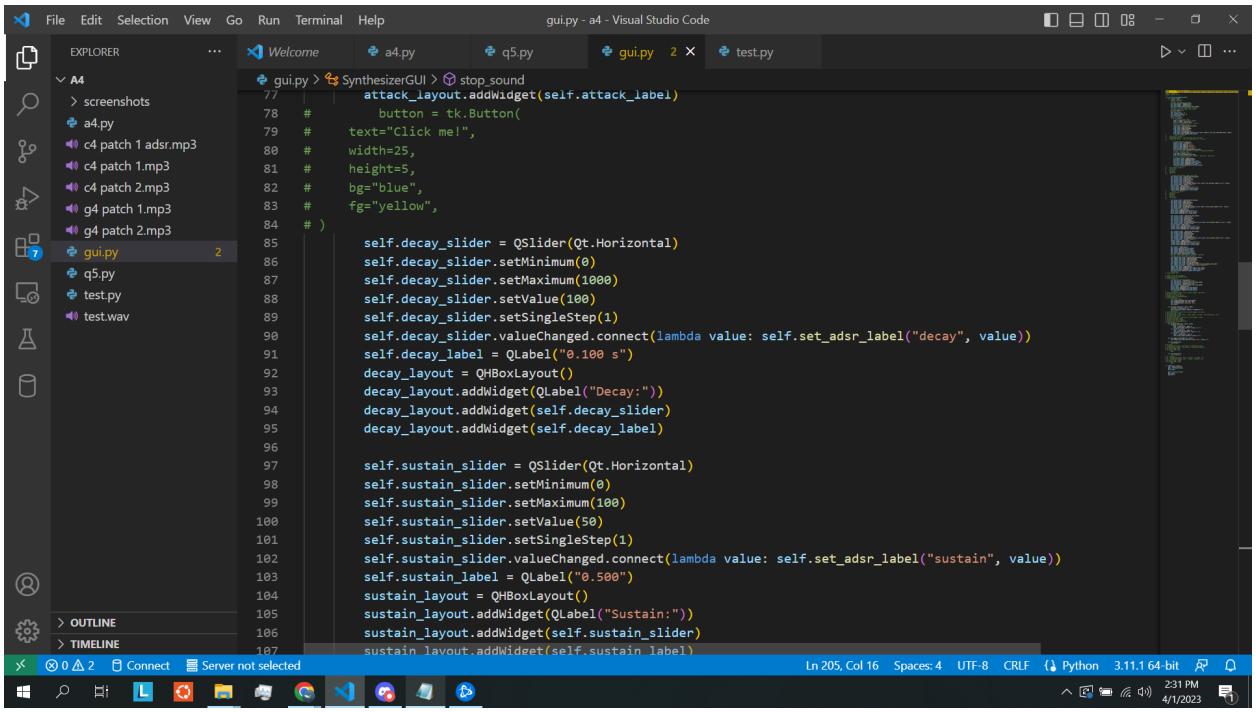
For question 9 we had to extend the synthesizer to support polyphony using voice allocation. In order to avoid conflicting sounds when using polyphony, each sound gets assigned a separate voice or channel. The process of doing that for this question is called voice allocation. I decided to implement polyphony and voice allocation through the use of threads. Threads are separate sequences of code that can run concurrently with other threads. By assigning each sound to a separate thread, we can ensure that multiple sounds can be played simultaneously without interfering with each other. First, I needed to create a thread for each sound that we wanted to play using the inputted note numbers. Each thread runs the function playsound() that plays the sound on a specific voice or channel. Next, we need to ensure that each thread is synchronized so that they do not interfere with each other. Finally, we need to ensure that the threads are properly cleaned up when they finish playing their sound. We use the join() method to wait for each thread to finish before exiting the program. I decided the number of voices to use by investigating how many voices I could have simultaneously without audio artifacts on my computer. Then once I hear a weird echoing noise then I know I've gone too far and have too many voices such that they can not play in unison/simultaneously.

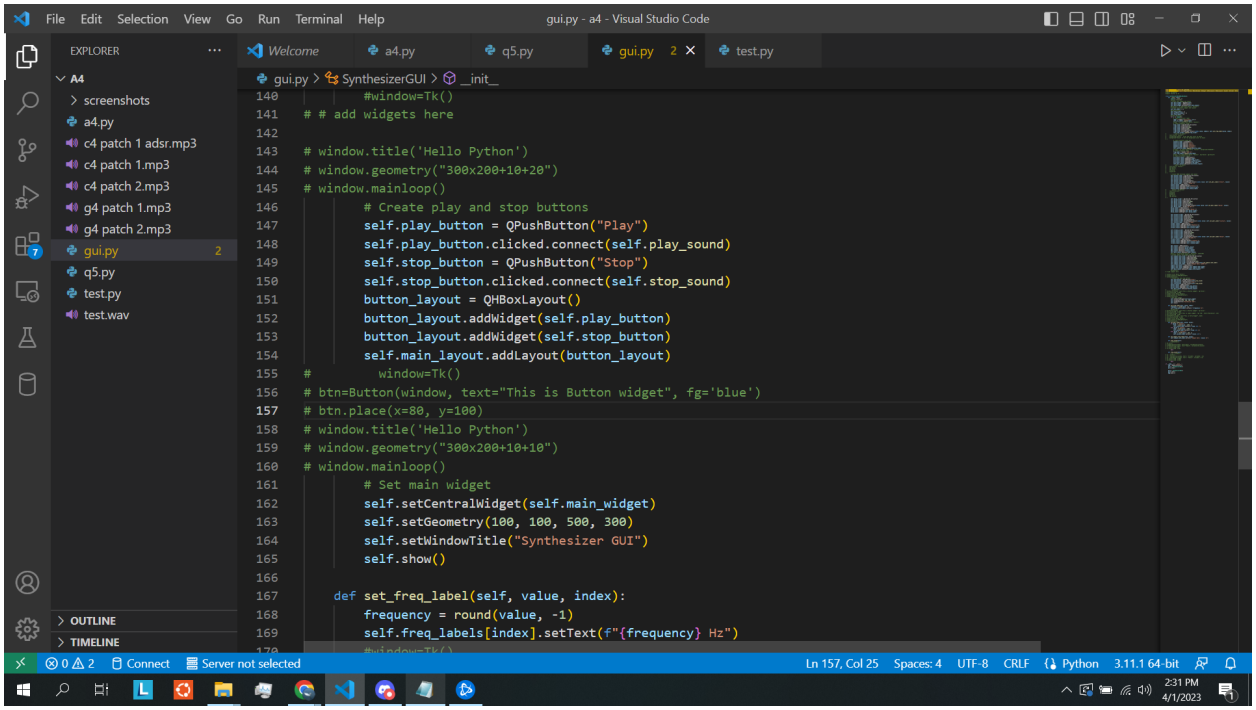
.

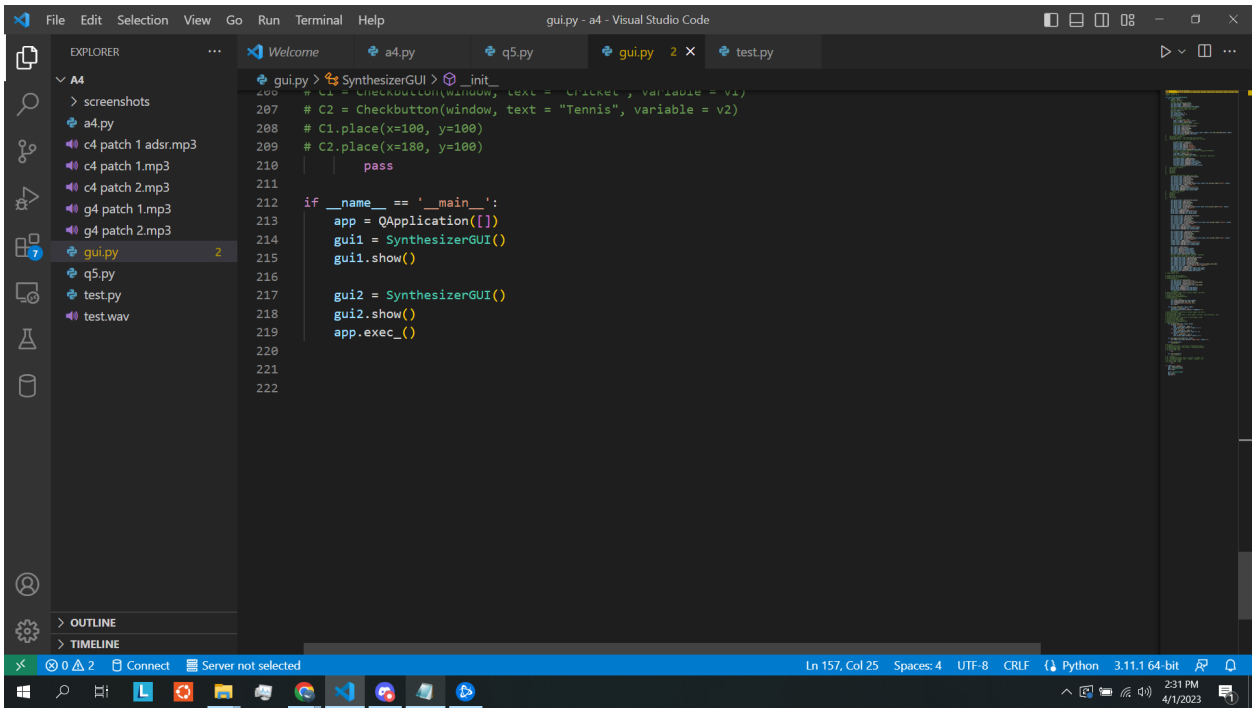

File Edit Selection View Go Run Terminal Help
gui.py - a4 - Visual Studio Code
Synthesizer GUI
Oscillator 1
sine
660 Hz
Oscillator 2
sine
380 Hz
Oscillator 3
sine
590 Hz
Oscillator 4
sine
390 Hz
Attack:
0.163 s
Decay:
0.494 s
Sustain:
0.400
release:
0.100 s
Sample Rate: 15753 Hz
Play
Stop
Synthesizer GUI
Oscillator 1
sine
480 Hz
Oscillator 2
sine
370 Hz
Oscillator 3
sine
400 Hz
Oscillator 4
sine
530 Hz
Attack:
0.022 s
Decay:
0.173 s
Sustain:
0.500
release:
0.100 s
Sample Rate: 20071 Hz
Play
Stop
OUTLINE
TIMELINE
Connect
Server not selected
Ln 163, Col 13
Spaces: 4
UTF-8
CRLF
Python
3.11.1 64-bit
2:08 PM
4/1/2023


File Edit Selection View Go Run Terminal Help
gui.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py
c4 patch 1 adsr.mp3
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
gui.py
q5.py
test.py
test.wav
Welcome
a4.py
q5.py
gui.py
test.py
gui.py > SynthesizerGUI > stop_sound
from PyQt5.QtCore import Qt, QThread
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QSlider, QPushBu
#import threading
#import tkinter as tk
class SynthesizerGUI(QMainWindow):
def __init__(self):
super().__init__()
#window = tk.Tk()
# Create main widget and layout
self.main_widget = QWidget(self)
self.main_layout = QVBoxLayout()
self.main_widget.setLayout(self.main_layout)
#greeting = tk.Label(text="Hello, Tkinter")
# Create oscillator widgets and layouts
self.osc_labels = []
#greeting.pack()
self.freq_sliders = []
self.waveform_combos = []
self.freq_labels = []
#window.mainloop()
for i in range(4):
# Create oscillator label
label = QLabel(f"Oscillator {i+1}")
self.osc_labels.append(label)
#label = tk.Label(text="Hello, Tkinter")
# Create frequency slider
freq_slider = QSlider(Qt.Horizontal)
freq_slider.setMinimum(20)
freq_slider.setMaximum(3000)
OUTLINE
TIMELINE
Connect
Server not selected
Ln 205, Col 16
Spaces: 4
UTF-8
CRLF
Python
3.11.1 64-bit
2:31 PM
4/1/2023


File Edit Selection View Go Run Terminal Help
gui.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py
c4 patch 1 adsr.mp3
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
gui.py 2
q5.py
test.py
test.wav
Welcome
a4.py
q5.py
gui.py 2
test.py
gui.py > SynthesizerGUI > stop_sound
waveform_combo = QComboBox()
waveform_combo.addItem("sine")
waveform_combo.addItem("square")
waveform_combo.addItem("sawtooth")
waveform_combo.addItem("triangle")
self.waveform_combos.append(waveform_combo)
#vlabel = tk.Label(text="Hello, Tkinter", background="#34A2FE")
# Create frequency label
freq_label = QLabel("440 Hz")
self.freq_labels.append(freq_label)
#label = tk.Label(text="Hello, Tkinter", fg="white", bg="black")
# Create oscillator layout
oscillator_layout = QHBoxLayout()
oscillator_layout.addWidget(label)
oscillator_layout.addWidget(freq_slider)
oscillator_layout.addWidget(waveform_combo)
oscillator_layout.addWidget(freq_label)
self.main_layout.addLayout(oscillator_layout)
# label = tk.Label(
# text="Hello, Tkinter",
# fg="white",
# bg="black",
# width=10,
# height=10
#)
# Create ADSR envelope widgets and layout
self.attack_slider = QSlider(Qt.Horizontal)
self.attack_slider.setMinimum(0)
self.attack_slider.setMaximum(1000)
self.attack_slider.setValue(10)
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 205, Col 16 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:31 PM 4/1/2023


File Edit Selection View Go Run Terminal Help
gui.py - a4 - Visual Studio Code
EXPLORER
A4
screenshots
a4.py
c4 patch 1 adsr.mp3
c4 patch 1.mp3
c4 patch 2.mp3
g4 patch 1.mp3
g4 patch 2.mp3
gui.py 2
q5.py
test.py
test.wav
Welcome
a4.py
q5.py
gui.py 2
test.py
gui.py > SynthesizerGUI > stop_sound
attack_layout.addWidget(self.attack_label)
# button = tk.Button(
# text="Click me!",
# width=25,
# height=5,
# bg="blue",
# fg="yellow",
#)
self.decay_slider = QSlider(Qt.Horizontal)
self.decay_slider.setMinimum(0)
self.decay_slider.setMaximum(1000)
self.decay_slider.setValue(100)
self.decay_slider.setSingleStep(1)
self.decay_slider.valueChanged.connect(lambda value: self.set_adsr_label("decay", value))
self.decay_label = QLabel("0.100 s")
decay_layout = QHBoxLayout()
decay_layout.addWidget(QLabel("Decay:"))
decay_layout.addWidget(self.decay_slider)
decay_layout.addWidget(self.decay_label)
self.sustain_slider = QSlider(Qt.Horizontal)
self.sustain_slider.setMinimum(0)
self.sustain_slider.setMaximum(100)
self.sustain_slider.setValue(50)
self.sustain_slider.setSingleStep(1)
self.sustain_slider.valueChanged.connect(lambda value: self.set_adsr_label("sustain", value))
self.sustain_label = QLabel("0.500")
sustain_layout = QHBoxLayout()
sustain_layout.addWidget(QLabel("Sustain:"))
sustain_layout.addWidget(self.sustain_slider)
sustain_layout.addWidget(self.sustain_label)
OUTLINE
TIMELINE
0 2 Connect Server not selected
Ln 205, Col 16 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit
2:31 PM 4/1/2023





For question 10 I had to add a graphical-user interface for the synthesizer. For this I used PyQt5. The code creates a SynthesizerGUI class that extends PyQt5's QMainWindow. In the constructor of the class, we set up the basic layout of the GUI with four oscillators, an ADSR envelope, and a sample rate slider. The oscillators are created using QComboBox widgets that allow the user to select between different waveforms. The GUI also has a play button that uses the sounddevice library. The sample rate slider in the GUI allows the user to adjust the sample rate. The code updates the sample rate value whenever the slider is moved, and regenerates

the waveform with the new sample rate value. This updating whenever the slider is moved is true for all the sliders. This SynthesizerGUI class provides a simple way for users to generate and play custom synthesized sounds with different waveform shapes, ADSR envelopes, and sample rates. I also made it such that you can have two instances of the gui open at the same time for polyphony, you can add more than two but just for showing that I can, I chose to show two. In terms of steps, I first had to Import necessary libraries and modules, including PyQt5 and sounddevice, then make/define a SynthesizerGUI class and in this class set up the basic layout of the GUI. This contains widgets for waveform shapes, ADSR envelope values, and the sample rate. Next was connecting the sound playing method to a QPushButton widget in the GUI so that the user can play or stop the generated sound whenever they would like. Lastly, creating a QApplication object and two SynthesizerGUI objects, and run the application using the app.exec_() method. This way I can get two gui windows.