プログラミング演習３（Python）⑦ 音声処理モジュール Pyo

音声処理モジュール Pyo を使うと、音声信号の生成、フィルター・遅延・合成などの波形の処理が簡単にできる。なお、今回の演習も Google Colab ではなく、ローカル環境で行う。

・PYO のインストール方法
コマンドプロンプト等のコマンドラインから
pip install pyo
もしくは
py -m pip install pyo と入力して Enter を押す。
http://ajaxsoundstudio.com/pyodoc/download.html
には、Python3.8 がインストールされている場合は
py -3.8 -m pip install --user pyo
と入力すると書いてあるが、上記の書き方でインストールできるならそれでいい。。

他に、何か必要なライブラリがある場合はものがある場合は、
pip install ライブラリ名
と入力する
例えば、GUI ライブラリの Wxpython が必要だというメッセージが出た時は、
pip install wxPython
と入力して Enter キーを押す。
（実行結果）

```
C:\Users\[redacted]\Desktop\py>pip install wxPython
Collecting wxPython
  Downloading wxPython-4.2.2-cp39-cp39-win_amd64.whl.metadata (3.1 kB)
Collecting six (from wxPython)
  Downloading six-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
Collecting numpy (from wxPython)
  Downloading numpy-2.0.2-cp39-cp39-win_amd64.whl.metadata (59 kB)
Downloading wxPython-4.2.2-cp39-cp39-win_amd64.whl (16.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.7/16.7 MB 15.7 MB/s eta 0:00:00
Downloading numpy-2.0.2-cp39-cp39-win_amd64.whl (15.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.9/15.9 MB 15.4 MB/s eta 0:00:00
Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, numpy, wxPython
Successfully installed numpy-2.0.2 six-1.16.0 wxPython-4.2.2

C:\Users\[redacted]\Desktop\py>
```

ヴァージョンが古いというメッセージが気になるときは、
py -m pip install -upgrade pip
と入力して Enter キーを押して、最新のヴァージョンにする。

py7-1.py　正弦波の音を鳴らす

```
001  from pyo import *
002  s = Server().boot()                  # PYO のサーバを立ち上げる
003  s.amp = 0.1                          # 20 dB ゲインを下げる
004  a = Sine().out()                     # 正弦波を生成し、out()で出力する
005  a.ctrl(title="SIN Wave")             # パラメータ編集スライダー
006  sp = Scope(a)                        # 波形表示
007  sp.setGain(0.5)                      # 縦軸の倍率　0.5 => -6dB
008  sp.poll(1)                           # start ボタンを押すと波形が表示される
009  s.gui(locals())                      # GUI で終了処理などができるようにする
```

（説明）

1行　PYO モジュールのすべてをインポートする

2行　PYO モジュールのサーバーを立ち上げる

3行　サーバーのゲインを下げる

4行　正弦波を生成し、out（）で出力して、a に代入する

5行　a の内容をスライダーでコントロールできるようにする。タイトルを SIN Wave とする

6行　ａの波形をｓｐに代入する

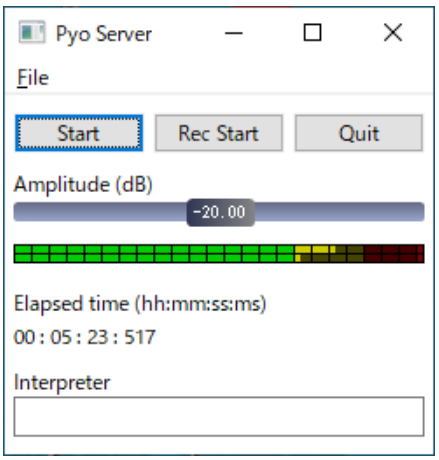
7行　波形のゲインを０．５にする

8行　start ボタンを押すと波形が表示される

9行　波形を表示させる

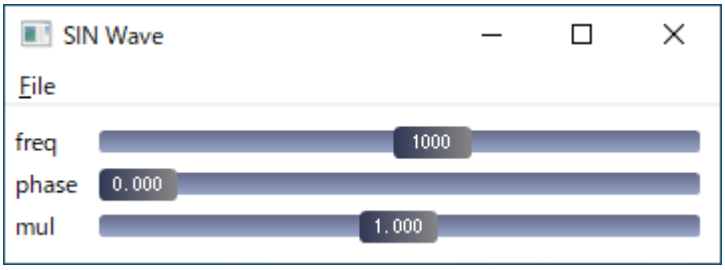
（実行結果）

Ｐｙｏサーバーの Start ボタンを押すと正弦波が発信する



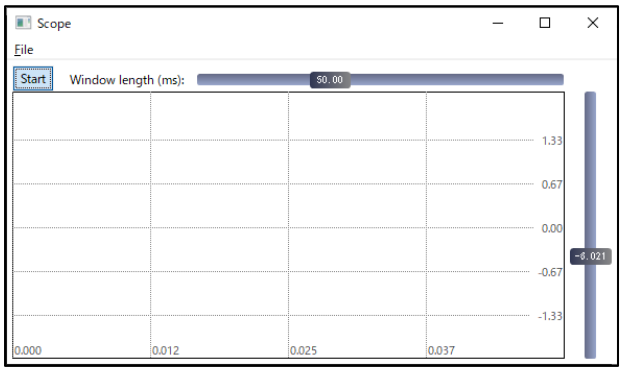
・正弦波のパラメータの設定

スライダを操作して周波数や振幅などが変えられる

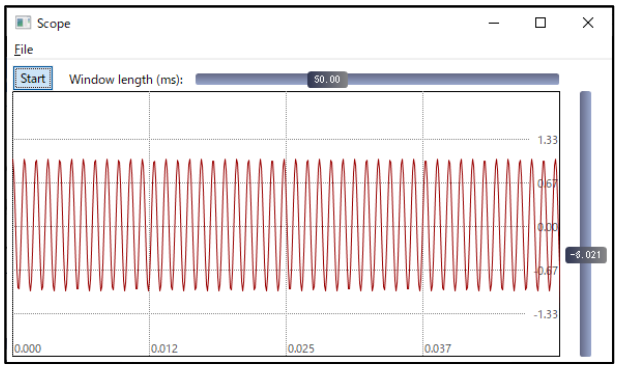


波形をＳｃｏｐｅで表示させる

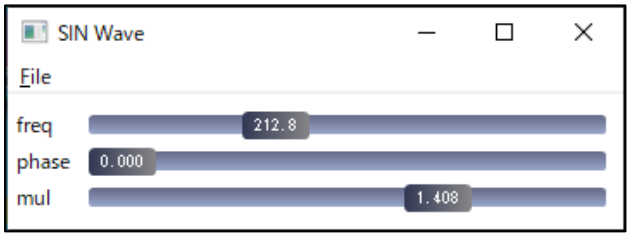
Start ボタンを押す前

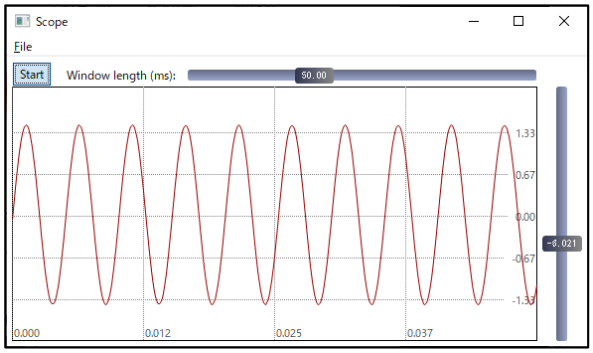


押した後



波形を表示させた後、スライダーの設定を変えると波形が変わる





py7-2.py　ＭＩＤＩキーボードを作成して音を鳴らし、周波数分布（スペクトラム）表示をする

```
001  From pyo import *                  # pyo モジュールをすべて読み込む
002  s = Server().boot(); s.start()  #   「サーバ」を作成・開始
003  notes = Notein( poly=10, scale=0, first=0, last=127, channel=0, mul=1 )
004  freqs = MToF( notes["pitch"] )  #  周波数や強度などの設定
005  amps = Port( notes["velocity"], risetime=0.005, falltime=0.5, mul=1 )
006  notes.keyboard()                   # 画面上で操作できる「キーボード」を生成する
007  osc = Sine( freq=freqs, mul=amps)  # 所定の周波数・強度の値で正弦波生成
008  out = osc.mix(1).mix(2).out()    # 出力を１系統にまとめた上で、左右に複製
009  sp = Scope(out)                    # 波形表示
010  spectrum = Spectrum(out)           # 周波数分布表示
```

```
011  sp.setGain(1)
012  sp.poll(1)
013  s.gui( locals() )
```

（説明）

２行　s.start()で初めから音が出るようにしている。ただし、ここではキーボードで入力をしないと音が出ない。

３～５行　MIDI のチャンネルの設定などをして、４，５行で周波数や強度などの設定をする

６行　MIDI キーボードを作成する

７行　所定の周波数などの値を、キーボードなどから得た値から得られるようにする。

８行　出力を１系統にまとめて、左右に複製する

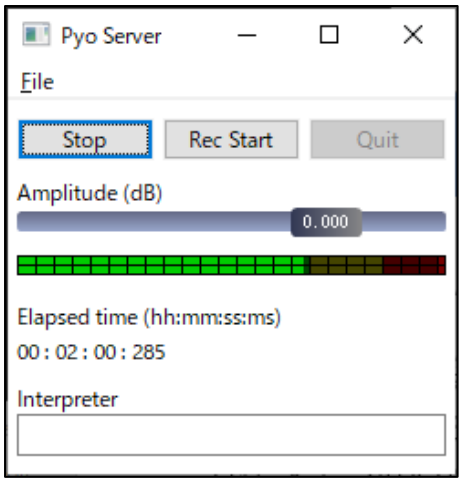
９行　out の波形を表示する

１０行　out の周波数分布を表示する

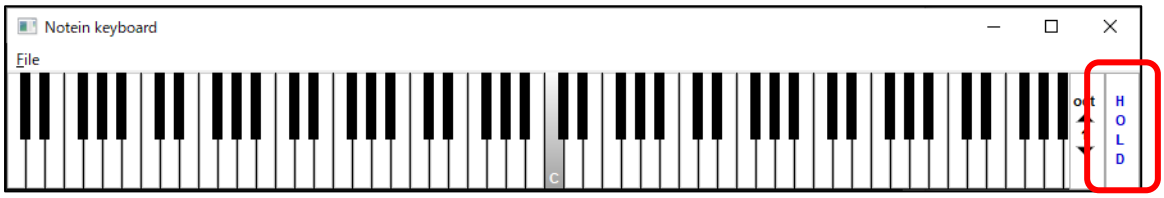
１１行　スコープの入力の設定

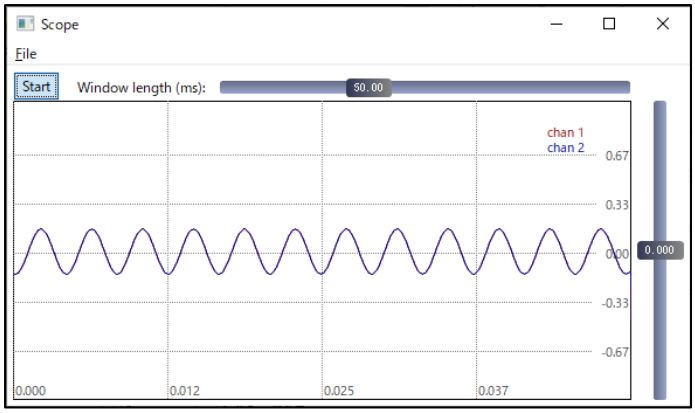
（実行結果）

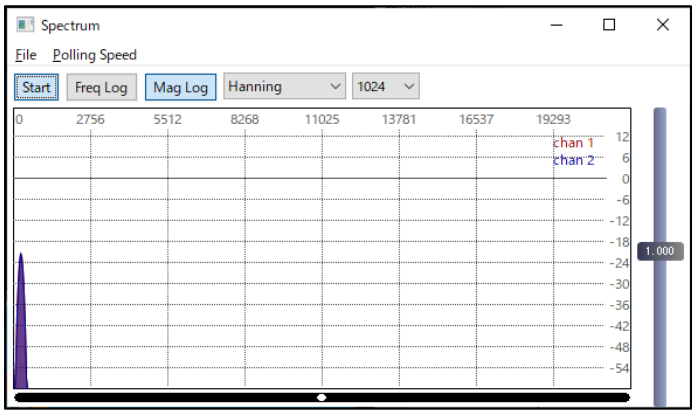
１行目に s.start()が入っているので、Start ボタンが押された状態になる。



初めの設定では「HOLD」が ON になっているので、押しっぱなしの状態になる。「HOLD」をクリックして解除した後、ボタンを離せば、時間と共に減衰する。




Scope
File
Start
Window length (ms):
50.00
chan 1
chan 2
0.67
0.33
0.00
-0.33
-0.67
0.000
0.000
0.012
0.025
0.037


Spectrum
File
Polling Speed
Start
Freq Log
Mag Log
Hanning
1024
0
2756
5512
8268
11025
13781
16537
19293
chan 1
chan 2
12
6
0
-6
-12
-18
-24
-30
-36
-42
-48
-54
1.000

py7-3.py　コーラス、ディレイ、リバーブを組み込む

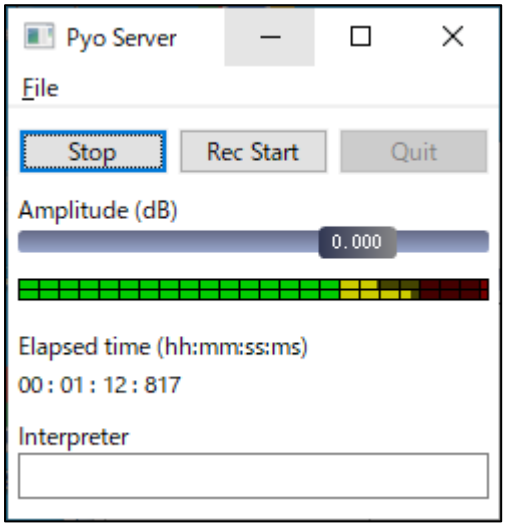
```
001  from pyo import *                  # pyo モジュールをすべて読み込む
002  s = Server().boot(); s.start()  #  音声処理を実行する「サーバ」を作成・開始
003  notes = Notein( poly=10, scale=0, first=0, last=127, channel=0, mul=1 )
004  freqs = MToF( notes["pitch"] )
005  amps = Port( notes["velocity"], risetime=0.005, falltime=0.5, mul=1 )
006  notes.keyboard()  # 画面上で操作できる「キーボード」を生成する
007  osc = Sine( freq=freqs, mul=amps)# 所定周波数・強度で正弦波生成
008  out = osc.mix(1).mix(2).out()  # 出力を１系統にまとめた上で、左右に複製
009  chorus = Chorus(out,depth=[1.5,1.6]).out(); chorus.ctrl()
010  delay = Delay(chorus,delay=[.15,.16]).out(); delay.ctrl()
011  reverb=WGVerb(delay, feedback=[.7,.7]).out(); reverb.ctrl()
012  sp = Scope(out)                    # 波形表示
013  spectrum = Spectrum(out)           # 周波数分布表示
014  sp2 = Scope(reverb)                # 出力波形を表示
015  spectrum2 = Spectrum(reverb)    # 周波数分布を表示
016  sp.setGain(1)
017  sp.poll(1)
018  sp2.setGain(1)
019  sp2.poll(1)
020  s.gui( locals() )
```

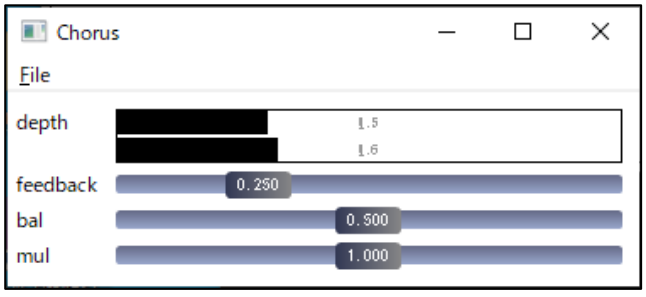
（解説）

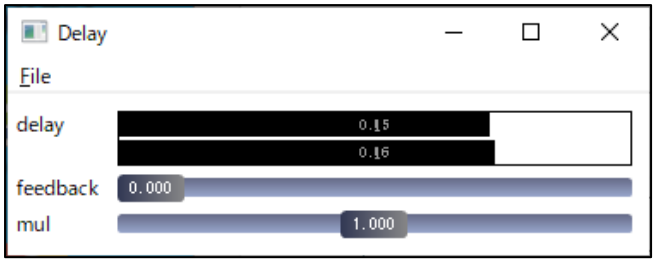
９行　コーラス　１０行　ディレイ　１１行　リバーブ　の機能を付けた

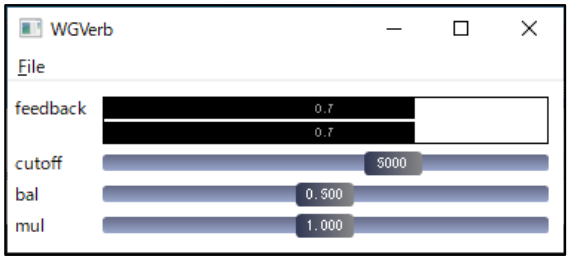
１８，１９行　効果を付けた波形を別のスコープで表示させた

（実行結果）

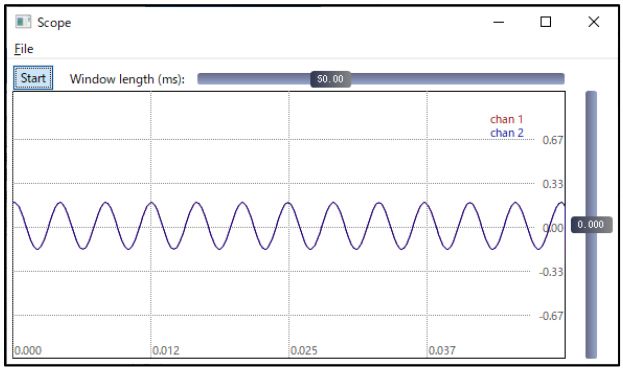


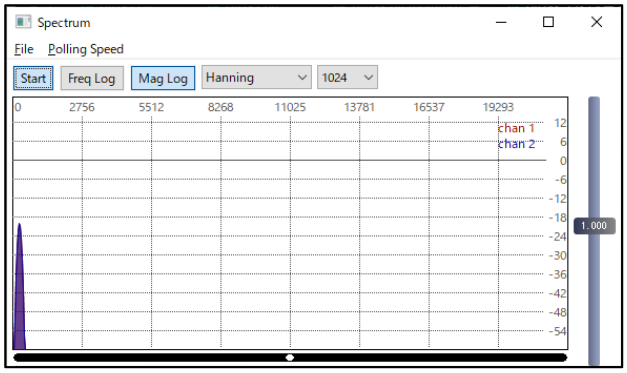




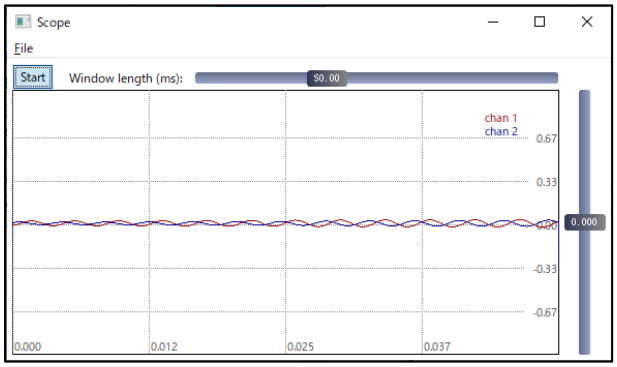


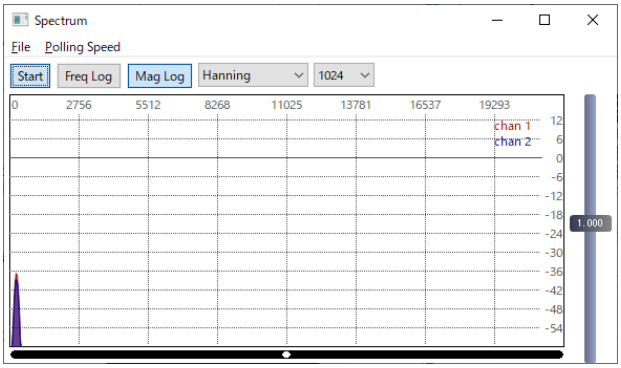
正弦波の波形と周波数分布





効果を入れた後の波形と周波数分布





py7-4.py　キーボードで作った正弦波に、マイクからの入力を合成する

```
001  from pyo import *                    # pyo モジュールをすべて読み込む
002  s = Server().boot(); s.start()  #  音声処理を実行する「サーバ」を作成・開始
003  notes = Notein( poly=10, scale=0, first=0, last=127, channel=0, mul=1 )
004  freqs = MToF( notes["pitch"] )
005  amps = Port( notes["velocity"], risetime=0.005, falltime=0.5, mul=1 )
006  notes.keyboard()  # 画面上で操作できる「キーボード」を生成する
007  osc = Sine( freq=freqs, mul=amps)# 所定周波数・強度で正弦波生成
008  mic = Input().play(); mic.ctrl()  # マイク入力
009  out = (osc+mic).mix(1).mix(2).out() #マイクの入力をまとめた上で、左右に複製
010  sp = Scope(out)# 波形表示
011  spectrum = Spectrum(out) #周波数分布表示
012  sp.setGain(0.5) #ゲインの調整
013  sp.poll(1)
014  s.gui( locals() )
```

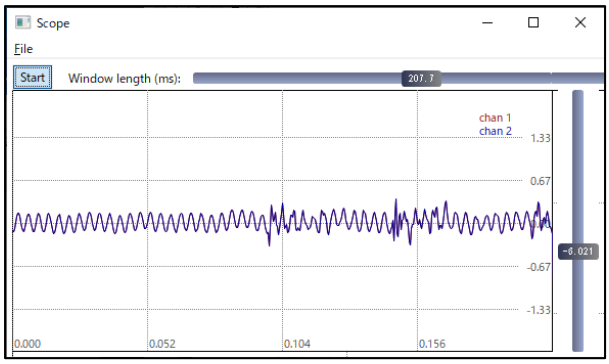
（解説）

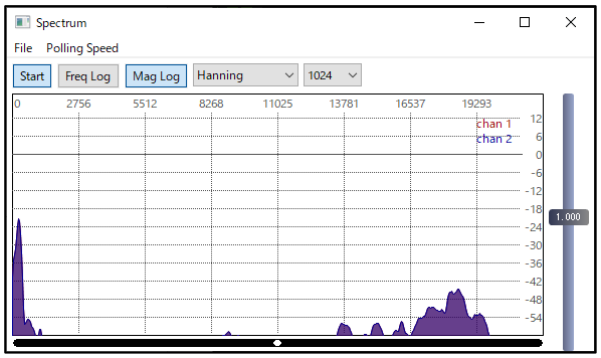
8行　マイク入力

9行　マイク入力とオシレータの波形（MIDI キーボードを操作して作った正弦波）と合成する

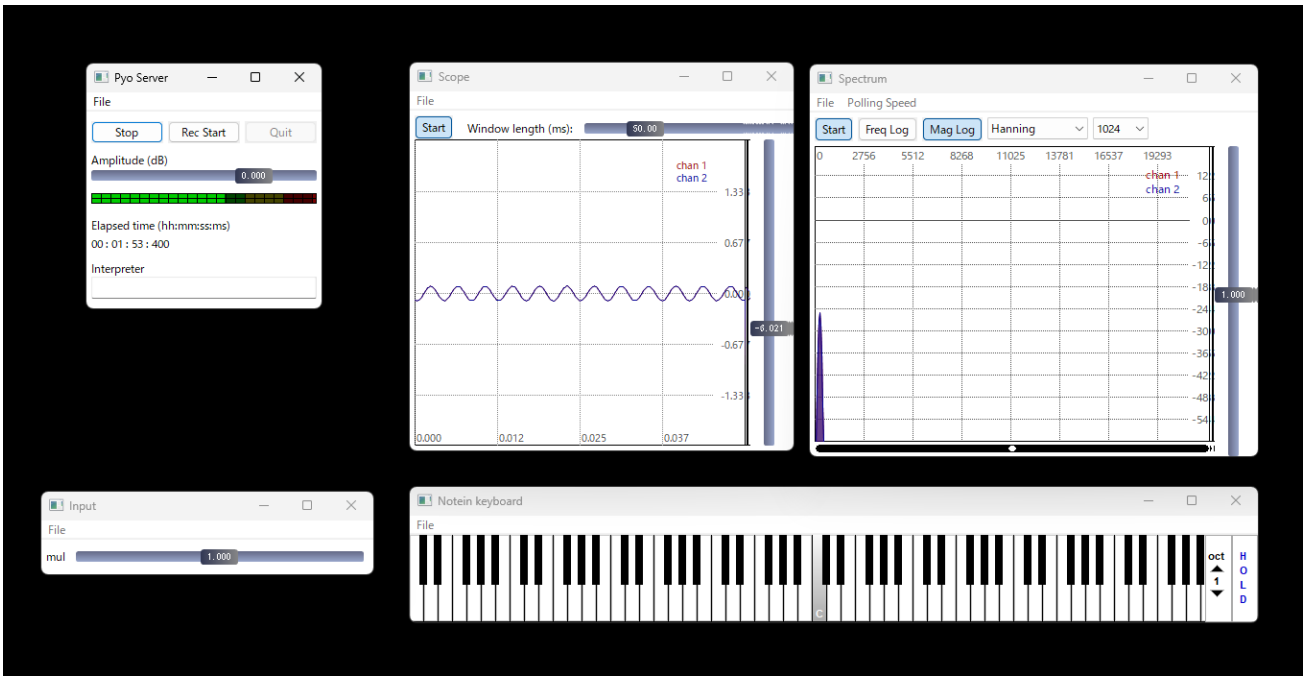
（実行結果）

マイクのそばで音を立てると、正弦波にノイズが入る





ディスプレイ全体



（参考図書）平林純 著「なんでも PYTHON プログラミング」（技術評論社）
（参考サイト）MIDI & AUDIO LAB 「Python モジュール PYO を使ってみる」
https://webmidiaudio.com/npage514.html