Question A1.4 (expected):

Write a function `generate_triangle` that takes as argument a MIDI note number and duration (floating point and in seconds) and creates a triangle wave signal with the corresponding frequency and duration. A triangle wave can be created by summing sinusoidal signals with appropriate frequencies (https://en.wikipedia.org/wiki/Triangle_wave). The function should also take as input the number of sinusoids used for the approximation. The triangle wave signal returned should be at audio rate.

You can use the follwing formulate to convert MIDI note numbers to frequencies and the order of arguments for the function and an example of calling it are provided below in comments. There is also code in the comments to show how by adding additional harmonics we get closer to a true triangle wave.

In [13]:
```
def midi_to_freq(pitch):
    return 440*(2**((pitch-69)/12.0))
```

In [14]:
```
def generate_triangle(pitch, nharmonics, dur=1.0, amp=1.0, sr=44100):
    def get_sin(A0,f0,P0,N,fs):
        t=np.arange(0,N/fs,1/fs)
        return A0*np.sin(2*np.pi*f0*t+P0)
    b=midi_to_freq(pitch)
    a=np.arange(1,b,2)
    PanSin=[]
    for i in range(nharmonics):
        f0=a[i]*b
        xt=get_sin(amp,f0,np.pi/2,dur*sr,sr)/(a[i]**2)
        PanSin.append(xt)
    triangle_signal=sum(PanSin)
    return triangle_signal
```

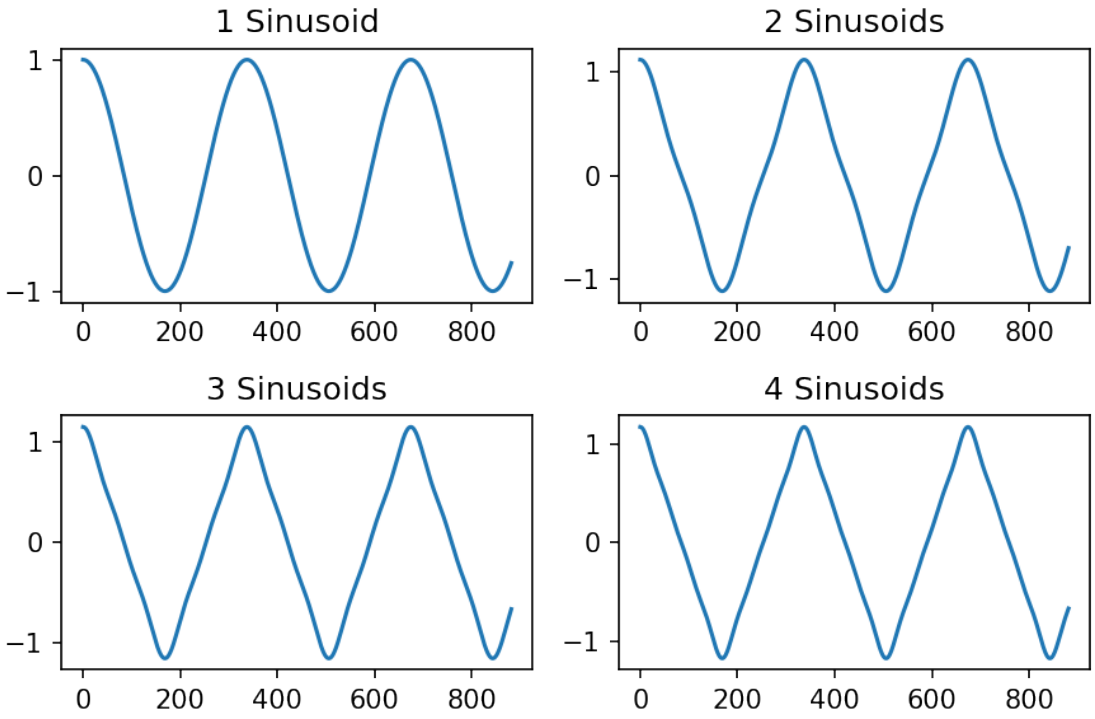
In [15]:
```
# code to check how your function works

signal = generate_triangle(48, 25)
ipd.Audio(signal, rate=44100)


x1 = generate_triangle(48, 1, 0.02)
x2 = generate_triangle(48, 2, 0.02)
x3 = generate_triangle(48, 3, 0.02)
x4 = generate_triangle(48, 4, 0.02)

fig, axs = plt.subplots(2,2, dpi=150)
axs[0,0].plot(x1)
axs[0,0].set_title('1 Sinusoid')
axs[0,1].plot(x2)
axs[0,1].set_title('2 Sinusoids')
axs[1,0].plot(x3)
axs[1,0].set_title('3 Sinusoids')
axs[1,1].plot(x4)
axs[1,1].set_title('4 Sinusoids')
```

```
fig.tight_layout()
```



Question A1.5 (advanced):

This question requires significantly more work than the other questions and also is more open ended.

The goal in this question is to create a simple synthesizer that can be used to play music pieces. You might have noticed that when we create pitched sounds either using single sinusoids or the triangular wave generator they start somewhat abruptly. The solution to this problem is to apply an amplitude envelope to the generated sounds (this was also done in the THX sound example we explored earlier in the class).

Read about signal envelopes: https://en.wikipedia.org/wiki/Envelope_(music) and implement an ADSR envelope generator that can be multiplied with an audio signal to provide smoother attacks and releases. The ADSR envelope genrator should take as input the duration and have the amplitude target points and corresponding times be expressed as percentages of 1.0 for the amplitude and of the duration for the points.

Show how you can render a simple melody initially using non-enveloped signals and then using enveloped signals. Provide both audio examples and plots showing how your ADSR works.

Implement a way to provide a music score and render it using your ADSR-enveloped notes using the triangle wave generator. There are different ways to go about this and it is up to you how you encode the information. Here are some suggestions:

1. Come up with your own format
2. Use SKINI https://ccrma.stanford.edu/software/stk/skini.html
3. Use MIDI files and the mido Python library to parse them
4. The Music21 tiny notation format https://web.mit.edu/music21/doc/usersGuide/usersGuide_16_tinyNotation.html

Type your answer here, replacing this text.

Question A1.G (CSC575/advanced):

This question only needs to be answered by graduate students who are registered in CSC575 and for these students it will be graded instead of question A1.1 which is optional for them.

Write a function that detects whether an audio recording of a single note is a single sinusoid, a sum of sinusoids approximation of a triangle wave, or a sum of sinusoids approximation of a square wave. You can assume you know the frequecy and the phase of the single note and that at least 3 harmonics are used for the approximations. Base your solution on the method of measuring amplitude by taking the dot-product with a sinusoidal basis function. Extend your solution to work without the assumption that the basis function and the note have the same phase using a pair of sine and cosine basis funcation to estimate the amplitude.

```
In [16]: # Your great detector of waveform type
```