0.0.1 Q1c(i) Amplitude

First, let's take a look at amplitude. A negative amplitude is equivalent to a positive one with a phase shift of π (since $\sin(\theta - \pi) = -\sin(\theta)$), so we can restrict ourselves to the case when $A \geq 0$. Let's listen to a few different amplitude settings.

```
In [13]: pause = np.zeros(int(fs * 0.5))

         volume_progression = np.concatenate([np.hstack((pure_tone_gen("C", fs, duration=1.5, amplitude=
                                              for ampl in np.linspace(0, 1, 11)])
         ipd.Audio(data=volume_progression, rate=fs)
```

```
Out[13]: <IPython.lib.display.Audio object>
```

Q: How many of the different amplitude settings could you tell apart? Did they all sound the same? Or, at the other extreme, could you tell almost all of them apart? A rough estimate is fine.

By "tell apart", we mean the **volume level should be distinct**, not that the music sounds fundamentally different.

A: Probably max 7 of them. The higher amplitude (louder) ones are much harder to tell apart than the low amplitude (quieter) ones.

0.0.2 Q1c(ii) Phase

Now, let's try the phase. We did 11 different amplitude settings, so let's do 11 different phase settings. While the amplitude can be any real number (theoretically, at least — we don't want to blow out our speakers or eardrums!), phase always wraps around every 2π, so we'll pick 11 uniformly spaced values from this range.

```
In [14]: pause = np.zeros(int(fs * 0.5))

         volume_progression = np.concatenate([np.hstack((pure_tone_gen("C", fs, duration=1.5, amplitude=
                                              for phase in np.linspace(0, 2 * np.pi, 11)])
         ipd.Audio(data=volume_progression, rate=fs)
```

```
Out[14]: <IPython.lib.display.Audio object>
```

(Problem 1(c)(ii): Phase) **Q:** How many of the different phase settings could you tell apart? Did they all sound the same? Or, at the other extreme, could you tell almost all of them apart? A rough estimate is fine.

A: They all sound identical.

0.0.3 Q1c(iii): Amplitude vs. Phase

Q: Based on the experiments you just performed, which do you conclude that the human ear is more sensitive to: amplitude, or phase?

Note: If all settings sounded the same for *both* amplitude and phase, then try using headphones or an external speaker (or turn up your volume).

A: Clearly, since I could tell apart some of the amplitudes, we are much more sensitive to amplitude changes than phase.

0.1 Q2b(i): Get Rect

The length L rectangular signal (sometimes also called the "rect" for short, or, alternatively, the "boxcar" signal) is defined as

$$r(n) = \begin{cases} 1 & n = 0, 1, 2, ..., L-1 \\ 0 & \text{otherwise} \end{cases}$$

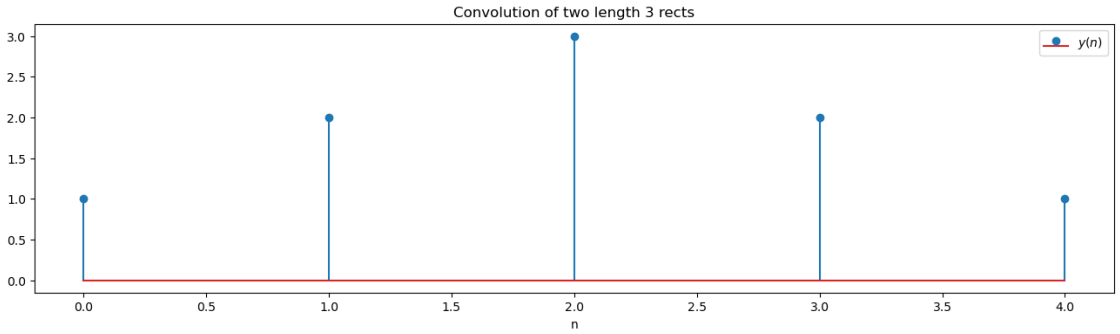
Some alternate definitions of the rect will normalize it (so that each nonzero point of the signal has height $1/L$), and some will center it around zero (although this can only be done when L is odd, so that there is a center point and an equal number of nonzero signal points on each side of $n = 0$).

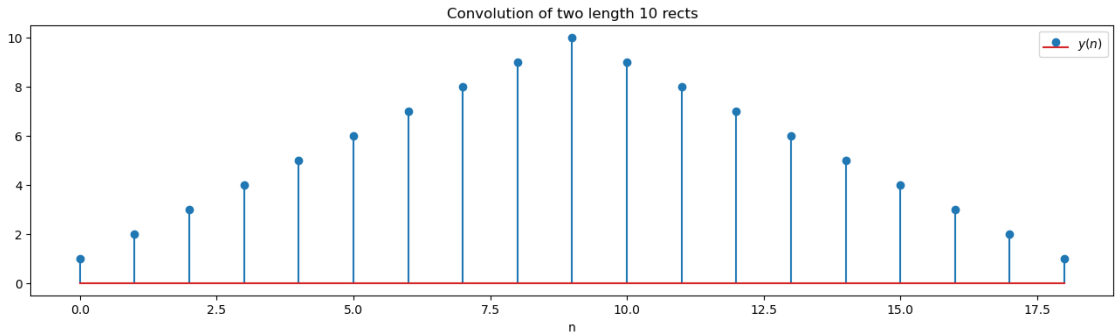
Generate three plots 1. Convolve a length 3 rect with a length 3 rect and plot it. Use "full" mode. Title it "Convolution of two length 3 rects".

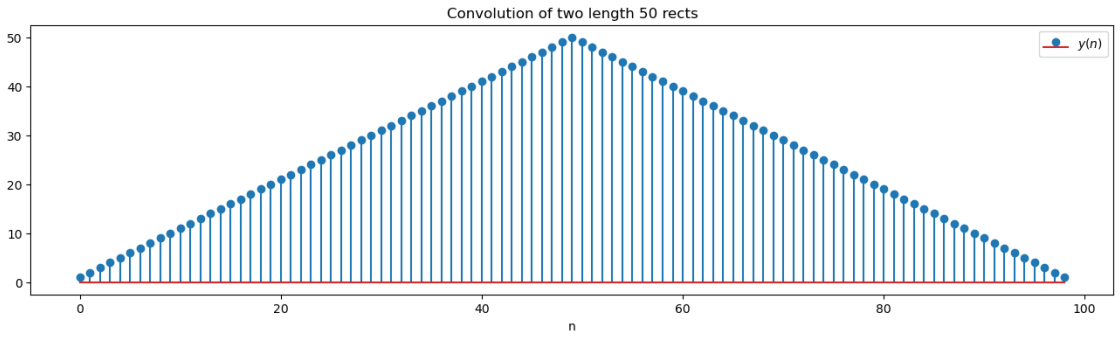
2. Convolve a length 10 rect with a length 10 rect and plot it, again in "full" mode. Title it "Convolution of two length 10 rects".
3. Convolve a length 50 rect with a length 50 rect and plot it, again in "full" mode. Title it "Convolution of two length 50 rects".

```
In [39]: q1 = [1 for _ in range(3)]
         q2 = [1 for _ in range(10)]
         q3 = [1 for _ in range(50)]

         p1 = np.convolve(q1, q1, mode="full")
         p2 = np.convolve(q2, q2, mode="full")
         p3 = np.convolve(q3, q3, mode="full")

         x1 = np.arange(0, 5)
         x2 = np.arange(0, 19)
         x3 = np.arange(0, 99)

         plt.figure(figsize=(16, 4))
         plt.stem(x1, p1, label='$y(n)$')
         plt.xlabel("n")
         plt.title("Convolution of two length 3 rects")
         plt.legend()
         plt.show()

         plt.figure(figsize=(16, 4))
         plt.stem(x2, p2, label='$y(n)$')
         plt.xlabel("n")
         plt.title("Convolution of two length 10 rects")
         plt.legend()
         plt.show()
```

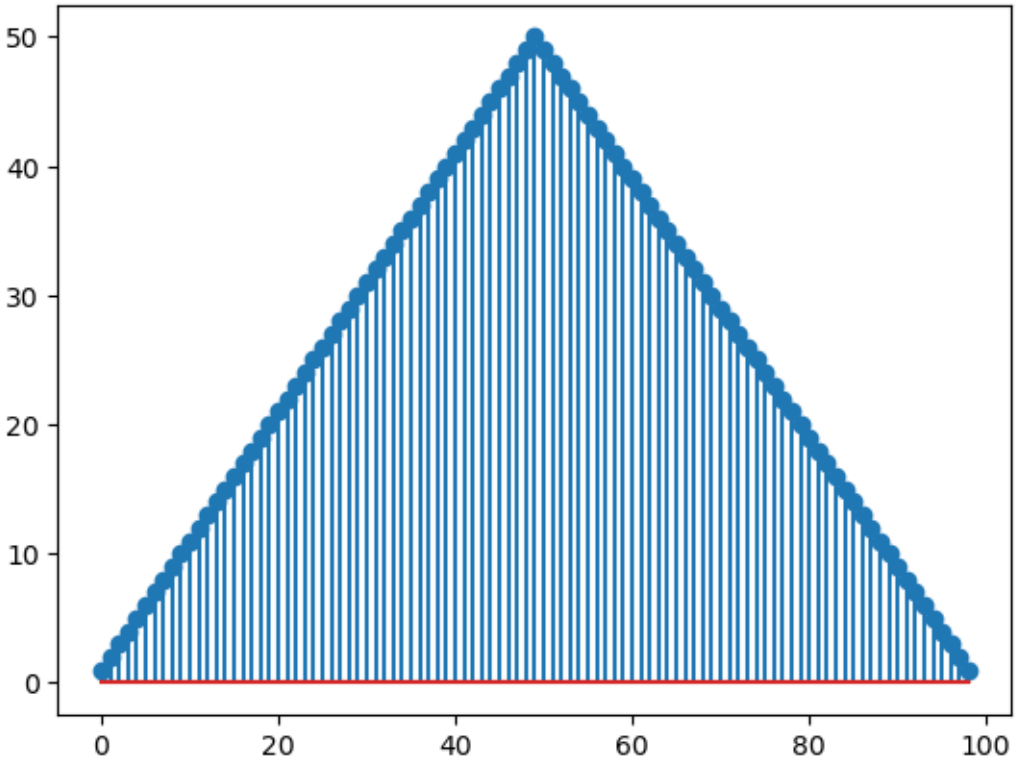
```
plt.figure(figsize=(16, 4))
plt.stem(x3, p3, label='$y(n)$')
plt.xlabel("n")
plt.title("Convolution of two length 50 rects")
plt.legend()
plt.show()
```







Out[39]: <StemContainer object of 3 artists>



Q: In general, what shape is the convolution of two rects *of the same length*? A one-word answer is fine.

A: Triangle

0.2 Q2b(ii): Get Rect Again

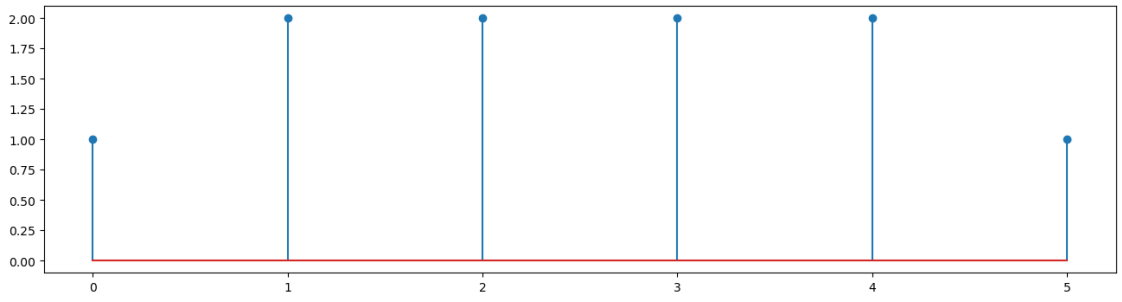
Now, we'll convolve rects that have *different* lengths and see what happens.

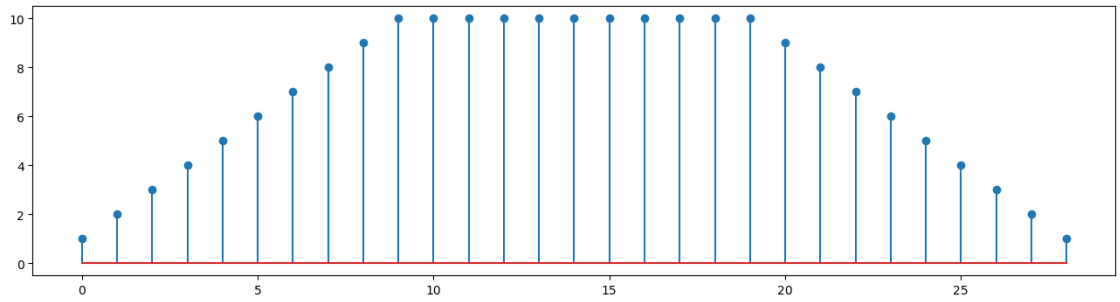
Create the following plots: 1. Convolve a length 2 rect with a length 5 rect. 2. Convolve a length 10 and length 20 rect. 3. Convolve a length 60 and length 20 rect.

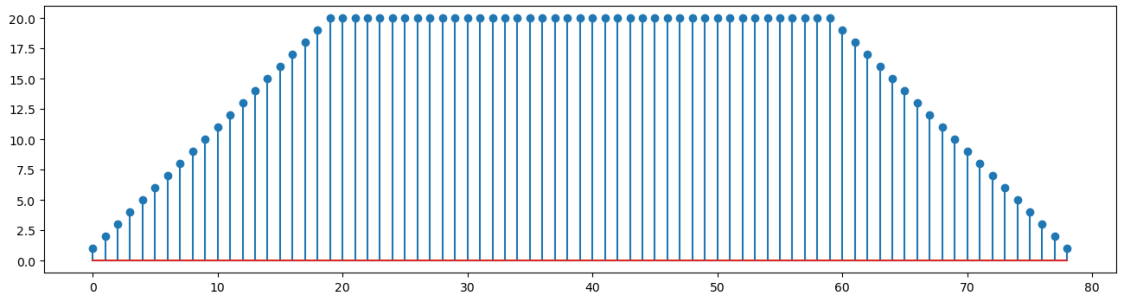
For this part, perform all convolutions in "full" mode, so you don't have to worry about signal values being cut out and can focus on the results. As a consequence, it's perfectly fine to not zero-pad any of your signals pre-convolution. Remember, "full" mode doesn't cut anything out. For example, if you were asked to convolve a length 2 rect with a length 4 rect, it's fine to use `np.array([1, 1])` and `np.array([1, 1, 1, 1])`, respectively, as the NumPy array representations of your signals.

For all parts of this question, plot the convolution result, and give your plot a reasonable title.

```
In [42]: p1 = np.convolve([1 for _ in range(2)], [1 for _ in range(5)], mode="full")
         p2 = np.convolve([1 for _ in range(10)], [1 for _ in range(20)], mode="full")
         p3 = np.convolve([1 for _ in range(60)], [1 for _ in range(20)], mode="full")

         plt.figure(figsize=(16, 4))
         plt.stem(p1)
         plt.show()

         plt.figure(figsize=(16, 4))
         plt.stem(p2)
         plt.show()

         plt.figure(figsize=(16, 4))
         plt.stem(p3)
         plt.show()
```







Q: In general, the convolution of two rects of different lengths is a trapezoid, which you should see from your plots. We'll define the *length* of the trapezoid's "top" as the number of points for which it attains its maximum value. For example, the length of a triangle's top is 1, since it peaks at one data point and slopes downward on either side of the peak.

Which trapezoid would you expect to have a longer top: one obtained by convolving a length 10 rect and length 20 rect, or one obtained by convolving a length 15 rect and length 20 rect? Briefly explain why.

A: I would expect that the former would have a longer top, becuase the top results from one signal being entirely overlaid onto the other. There are more instances of this happening between a length 10 and 20 rect than a length 15 and 20 rect.