Suppose you want to implement an echo, then a continuous-time equation would be:

y(t) = b0 x(t) + G x(t - T)

where T is the delay of the echo in seconds.

To implement this as a discrete-time difference equation, we would use

y(n) = b0 x(n) + G x(n - N)

where N depends on the sampling rate:

N = T * SAMPLING RATE.

If the sampling rate is Fs = 16000 samples/sec and the delay is T = 0.1 sec

then the delay in samples is

N = 1600 samples.

To implement this, we need to store a lot of past signal values!

And we would need to spend a lot of time updating

the past signal values (see wave_filter_python.py) which has commands

```
# Delays
x4 = x3
x3 = x2
x2 = x1
x1 = x0
y4 = y3
y3 = y2
y2 = y1
y1 = y0
```

To implement this for N = 1600, we need to update the values of 1600 variables
each time we compute a new output sample. We would like to avoid that.

A better way to implement this is to use a 'circular buffer'.

Demo programs for circular buffer.

echo_via_append.py
- a simple way to implement a delay

echo_via_circular_buffer.py
- a better way to implement a delay

- - - - Notes - - - -

How can we create a list of all zeros in Python?

Note how to concatenate lists:

```
>>> [3, 4, 5] + [10, 11, 12]
[3, 4, 5, 10, 11, 12]

>>> 3 * [5, 6, 7]
[5, 6, 7, 5, 6, 7, 5, 6, 7]
```

So, to create a list of all zeros:

```
>>> 10 * [0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

or use 'list comprehension':

```
>>> [ 0 for i in range(5) ]
[0, 0, 0, 0, 0]

>>> [ i**2 for i in range(5) ]
[0, 1, 4, 9, 16]
```

In the append approach, we manipulate the buffer this way:

```
>>> buffer = 5 * [0]
>>> buffer
[0, 0, 0, 0, 0]

>>> buffer.append(8)
>>> buffer
[0, 0, 0, 0, 0, 8]

>>> del buffer[0]
>>> buffer
[0, 0, 0, 0, 8]
```

About the 'del' command:
https://docs.python.org/3/tutorial/datastructures.html#the-del-statement

```
>>> buffer.append(5)
>>> del buffer[0]
>>> buffer
[0, 0, 0, 8, 5]
```

Question:

In the program echo_via_circular_buffer
change the line

```
   buffer[k] = x0
```

to

```
   buffer[k] = y0
```

Also with lower G_delay (0.6) and different delay_sec = 0.02

How does this change the effect of the program?
What is the transfer function of the system implemented by this change?
What is the impulse response of the system implemented by this change?

What happens when gain of the delay is greater than 1.0 ? G_delay > 1.0