CSC108H1 Lab

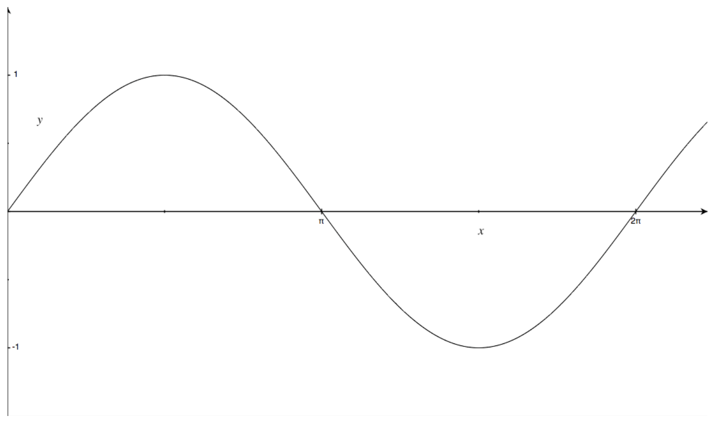
1 Objectives

- Introduce how sounds are synthesized and manipulated digitally.
- Use `while` or `for` loops to create notes with different frequencies.
- Use `while` loops to practice string processing.
- Use `while` loops to echo a sound.

2 Synthesizing Notes

Sounds are waves of air pressure. When a microphone records sound it takes a measure of the pressure in front of the microphone and returns it as a value. These values are called samples and can be either positive for increases in pressure or negative for decreases. The more often it takes samples the closer it can approximate the shape of the wave. This is called the sampling rate and is in units of samples per second. Through this process an analog signal (the wave) is converted into a digital signal (a list of values). When you play a sound file the reverse is done. Your computer's speakers convert the digital signal into an analog one that you can hear!

Recall from trigonometry the sine function, f(x) = sin(x):



If this was a wave of air pressure it would be the simplest sound in the world. As you'll remember this function has a cycle (repeats itself) of 2 pi. For this lab you will be synthesizing notes by create sine waves with different frequencies. First, some terminology and an explanation of their effect on the sound we hear.

frequency

For a cyclic process frequency is the measure of repeated cycles per unit time. The SI unit is Hz (herz), where 1 Hz is one cycle per second. A simple sine wave sound would have a frequency of 2 * pi Hz. When we hear sounds with different frequencies we hear them as having higher or lower *pitches*. The higher the pitch the more cycles per second. Following is a table of musical notes and their corresponding frequencies (in Hz) starting at middle C.

C	264
D	297
E	330
F	352
G	396
A	440
B	494

period

A period is one cycle in a repeated event and is defined as the reciprocal of the frequency. The SI unit is seconds per cycle. A sine wave sound would have a period of 1.0 / (2 * pi) seconds per cycle.

amplitude

The amplitude of a wave is the distance between zero and the top or bottom of a cycle. In general, the amplitude is the most important factor in our perception of the *volume* of a sound. The greater the amplitude the louder the sound. For digital sounds there is a maximum amplitude, which depends on the encoding used for each sample. Since sounds in `media.py` have a default encoding of 16 signed bits there is a maximum of 32767 as a pressure value and a minimum of -32768.

Your goal in this section is to create a function, which takes in a frequency, an amplitude, and a number of samples and returns a Sound object for that note. For example, `create_note(440, 6000, 11025)` would return the note A that is half a second long (at a sampling rate of 22050). Its highest pressure value would be around 6000 and its lowest -6000.

In order to change the amplitude of a sine wave the function must be multiplied by that amplitude. For example, f(x) = sin(x) * 5, would have an amplitude of 5. When it comes to changing the length of the period it is done inside the sine function. For example f(x) = sin(2 * pi * x / 10) would have a period of 10. To change the frequency in a Sound object you'll have to determine the size of the period in samples, or the samples per period. You'll need both the sampling rate and the desired frequency to calculate this. Use the `math` module's `sin()` function and `pi` variable to calculate the individual values of the sin curve as you iterate across the Sound.

Good luck! Ask your TA if you have questions about the mathematics or the programming.

3 Making Songs

Now that you've got a whole octave at your disposal it's time to make some noise. We've provided you with a mystery song:

```
mystery = "CCGGAA2GFFEEDD2CGGFFEEDGGFFEEDCCGGAA2GFFEEDD2C"
```

It's your job to write a function that takes a string like `mystery` and returns the song that it represents. However, there are some complications. Some of the notes in `mystery` are played for two counts (the notes are preceded by a 2). Your function will need to handle this. Here are a few hints:

- Create a helper function that takes a character and returns the right note. To make this faster you might want to define some global variables for each of the notes.
- Use a `while` loops to step through the string. What do you do when you reach a number?
- You might want to append a tenth of a second of silence between notes to make them sound more distinct.

Show your TA the song when you've finished.

4 Echo echo echo echo echo echo echo

Sound mixing involves manipulating the values of the samples to change the quality of the sound. In this section you'll create a function that takes a sound and a delay and returns that sound with an echo. For example `echo(snd, 22050)` would return `snd` with an echo delayed by one second.

Echos are produced by adding a copy of the sound wave offset by the delay with a decreased amplitude. Adding waves together is very easy with computers. All you need to do is each corresponding samples' values together.

Write the the function `echo()`. Hint: Make a copy of the sound to use as a source, and use a while loop to iterate across the target adding the appropriate values of the source.

Show your TA the echoed sound.

5 Final thoughts

The sounds we produced in this lab sound very artificial. This is because they have only a single frequency. When instruments produce notes many frequencies can be heard. The way notes are synthesized for specific instruments is by creating waves that have all the appropriate frequencies for that instrument.