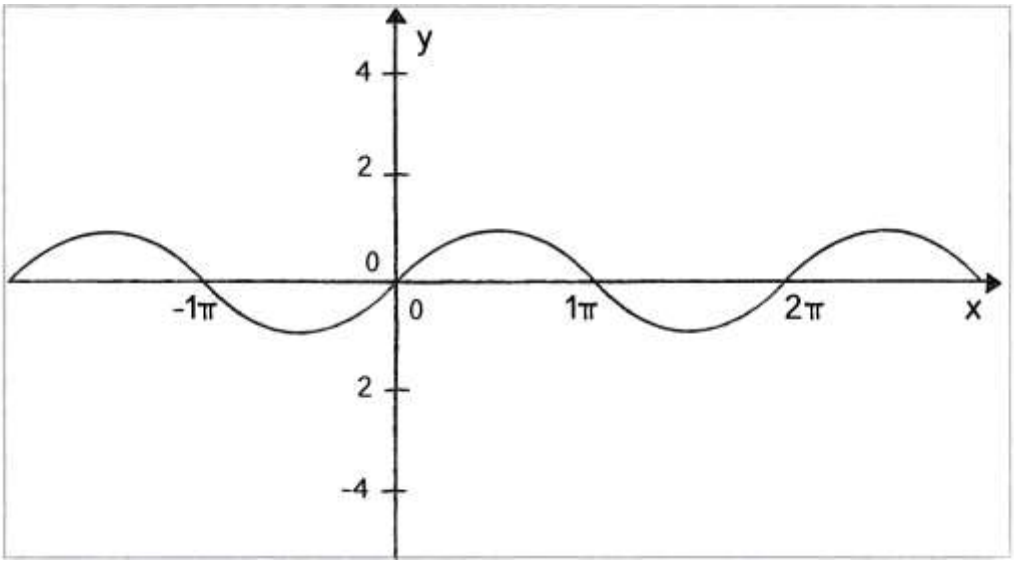
Oscillation amplitude and period

Are you amazed yet? In the angular motion section, we saw some pretty great uses of tangent (for finding the angle of a vector) and sine and cosine (for converting from polar to Cartesian coordinates). We could stop right here and be satisfied. But we're not going to. This is only the beginning. What sine and cosine can do for you goes beyond mathematical formulas and right triangles.

Let's take a look at a graph of the sine function, where *y = sine(x)*



You'll notice that the output of the sine function is a smooth curve alternating between –1 and 1. This type of a behaviour is known as ***oscillation***, a periodic movement between two points. Plucking a guitar string, swinging a pendulum, bouncing on a pogo stick—these are all examples of oscillating motion.

And so we happily discover that we can simulate oscillation in a ProcessingJS program by assigning the output of the sine function to an object's location. Note that this will follow the same methodology we applied to Perlin noise in the noise section.

Let's begin with a really basic scenario. We want a circle to oscillate from the left side to the right side of our canvas.

This is what is known as ***simple harmonic motion*** (or, to be fancier, “the periodic sinusoidal oscillation of an object”). It’s going to be a simple program to write, but before we get into the code, let’s familiarize ourselves with some of the terminology of oscillation (and waves).

Simple harmonic motion can be expressed as any location (in our case, the `x` location) as a function of time, with the following two elements:

- ***Amplitude***: The distance from the centre of motion to either extreme
- ***Period***: The amount of time it takes for one complete cycle of motion

Looking at the graph of sine embedded above, we can see that the amplitude is 1 and the period is `TWO_PI`; the output of sine never rises above 1 or below -1; and every `TWO_PI` radians (or 360 degrees) the wave pattern repeats.

Now, in the ProcessingJS world we live in, what is amplitude and what is period? Amplitude can be measured rather easily in pixels. In the case of a window 200 pixels wide, we would oscillate from the centre 100 pixels to the right and 100 pixels to the left. Therefore:

```
// Our amplitude measured in pixels

var amplitude = 100;
```

Period is the amount of time it takes for one cycle, but what is time in our ProcessingJS world? I mean, certainly we could say we want the circle to oscillate every three seconds. And we could track the milliseconds elapsed in our program (using `millis()`) and come up with an elaborate algorithm for oscillating an object according to real-world time.

We have another option, however: we can use the fact that ProcessingJS programs have a notion of "frames", and that by default, a program attempts to run 30 "frames per second." ProcessingJS gives us the `frameCount` variable to find out what frame we're currently on, and

the `frameRate()` function to change the preferred frames per second. 30 FPS is the default frame rate, as that's a good rate to produce a smooth animation to trick the human brain, but it can sometimes be helpful to slow down that frame rate, like when debugging.

We can thus decide to base our period on number of frames elapsed, as we've seen its closely related to real world time- we can say that the oscillating motion should repeat every 30 frames, or 50 frames, or 1000 frames, etc.

```
// Our period is measured in frames (our unit of time for animation)
var period = 120;
```

Once we have the amplitude and period, it's time to write a formula to calculate x as a function of time, which we will substitute the current frame count for.

```
var x = amplitude * sin(TWO_PI * frameCount / period);
```

Let's dissect the formula a bit more and try to understand each component. The first is probably the easiest. Whatever comes out of the sine function we multiply by amplitude. We know that sine will oscillate between -1 and 1. If we take that value and multiply it by amplitude then we'll get the desired result: a value oscillating between -amplitude and amplitude. (Note: this is also a place where we could use ProcessingJS's `map()` function to map the output of sine to a custom range.)

Now, let's look at what is inside the sine function:

TWO_PI * frameCount / period

What's going on here? Let's start with what we know. We know that sine will repeat every 2*PI radians—i.e. it will start at 0 and repeat at 2*PI, 4*PI, 6*PI, etc. If the period is 120 frames, then we want the oscillating

motion to repeat when the `frameCount` is at 120 frames, 240 frames, 360 frames, etc. `frameCount`is really the only variable; it starts at 0 and counts upward. Let's take a look at what the formula yields with those values.

frameCount	frameCount / period	TWO_PI * frameCount / period
0	0	0
60	0.5	PI
120	1	TWO_PI
240	2	2 * TWO_PI (or 4* PI)
etc.		

`frameCount` divided by period tells us how many cycles we've completed—are we halfway through the first cycle? Have we completed two cycles? By multiplying that number by `TWO_PI`, we get the result we want, since `TWO_PI` is the number of radians required for one sine to complete one cycle.

Wrapping this all up, here's the program that oscillates the x location of a circle with an amplitude of 100 pixels and a period of 120 frames.

```
angleMode = "radians";


var period = 120;    // length of one period in frames

var amplitude = 100;  // in pixels

stroke(0, 0, 0);

strokeWeight(2);

fill(127);


draw = function() {

  background(255);
```

```
  // Calculating horizontal location according to
  //  formula for simple harmonic motion
  var x = amplitude * sin(TWO_PI * frameCount / period);
  pushMatrix();
  translate(width/2, height/2);
  line(0, 0, x, 0);
  ellipse(x, 0, 48, 48);
  popMatrix();
};
```

It's also worth mentioning the term ***frequency***: the number of cycles per time unit. Frequency is equal to 1 divided by period. If the period is 120 frames, then only 1/120th of a cycle is completed in one frame, and so frequency = 1/120 cycles/frame. In the above example, we simply chose to define the rate of oscillation in terms of period and therefore did not need a variable for frequency.

Note that we worked through all of that using the sine function (`sin()` in ProcessingJS), but the same ideas apply to using the cosine function. The period is the same for both, and the main difference is just whether the beginning amplitude starts at 1 or at 0.