Motor Control Functions...

It's time to make Wink go! Controlling the motors is super easy. The motors are controlled by a single simple function called motors(). The function works like this...

```
motors(left,right);    //the motors() function needs a left motor speed and a right
                       //motor speed to work. The values can range from 0 to 255.
                       //Positive numbers make motors go forward.
                       //Negative numbers make motors go backward.
```

Let's look at a real example and see what happens.

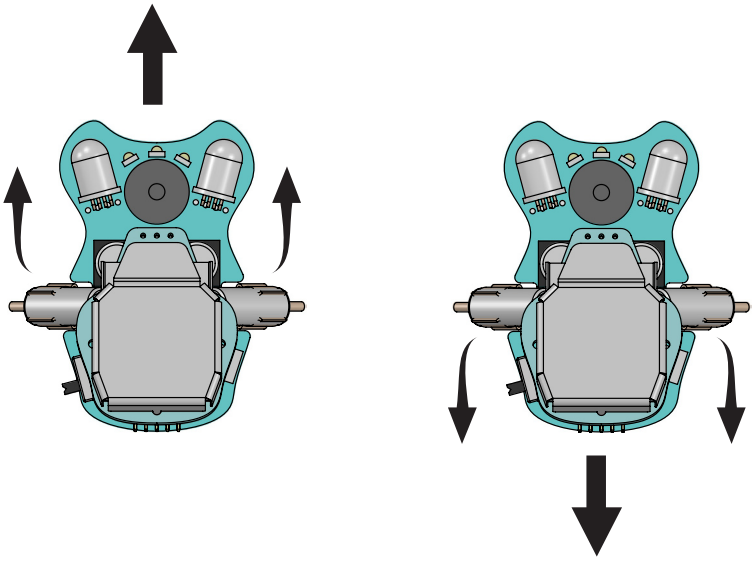
```
void loop(){
  motors(150,150);          //Both motors forward at 150 speed
  delay(1000);              //wait 1 second
  motors(-150,-150);        //Both motors backward at 150 speed
  delay(1000);              //wait 1 second then loop
}
```

Wink_Ch03Motors_Ex01

Both left and right motors run forward at speed 150 for one second, then they both go backward at speed 150 for one second. This loops over and over.

Now lets discuss what is happening. Wink moves whenever his motors turn. If the left and right both turn in the forward direction, then Wink will move forward.

If both motors turn in the reverse direction, then Wink will move backward.



Making Wink turn...

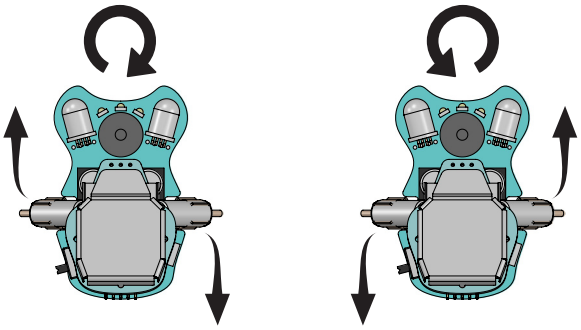
So we see what happens when both motors turn the same speed. Wink goes in a mostly straight direction. If we run the motors in opposite directions, Wink will spin in his own footprint.

```
void loop(){
  motors(250,-250);        //Spin to the right
  delay(3000);             //wait 3 seconds
  motors(-250,250);        //Spin to the left
  delay(3000);             //wait 3 seconds
}
```

Wink_Ch03Motors_Ex02

The positive left number moves the left side forward and the negative right number moves the right side backward.

The opposite is also true.



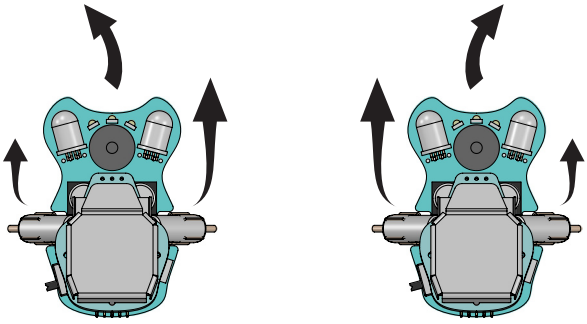
With this next example, be careful that Wink doesn't jump off your table. If we drive both motors the same direction, but at different speeds, Wink will drive forward in an arc movement. He can drive backward in an arc movement as well by using two different negative numbers.

```
void loop(){
  motors(150,100);         //Arc to the right
  delay(1000);             //wait 1 second
  motors(100,150);         //Arc to the left
  delay(1000);             //wait 1 second
}
```

Wink_Ch03Motors_Ex03

If both numbers are positive but different values, then one side will drive faster than the other, so Wink will drive in an arc.

Run this code and see what happens. Be careful wink doesn't jump off your table.



A few more motor functions...

We've included a few other simple functions you can also use to control the motors.

You can make Wink rotate in his own footprint by using spinRight() and spinLeft(). You will pass either of these functions a speed between 0 and 255 to set the spin speed. The Spin functions run the motors in opposite directions.

```
void loop(){
  spinRight(200);          //spin right at speed 200
  delay(2000);             //wait 2 seconds
  spinLeft(200);           //spin left at speed 200
  delay(2000);             //wait 2 seconds
}
```

Wink_Ch03Motors_Ex04

Use the spin functions to make Wink turn in his own footprint. This is done by running the motors in opposite directions.

You can also make Wink stop moving his motors using the function beStill(). This function doesn't require any values and causes Wink to immediately stop his motors.

```
void loop(){
  motors(150,150);         //both motors speed 150
  delay(1000);             //wait 1 second
  beStill();               //stop moving
  delay(3000);             //wait 3 seconds
}
```

Wink_Ch03Motors_Ex05

Use the beStill() function to make Wink stop moving.

Finally, you can use accelerateMotors() to smoothly accelerate from one speed to another speed. This will be useful in the drag race lesson later. This function causes Wink to begin spinning his motors at a certain speed, then transition smoothly to a different speed over a certain period of time. It accepts three values, which are the starting speed, the ending speed, and the period of time the acceleration will take. The time period is given in milliseconds, so a value of 1000 will take one second to go from the starting speed to the final speed. A short period results in fast acceleration, while a longer period results in more gradual acceleration. More on this in the drag race lesson.

```
void loop(){
  accelerateMotors(0,250,2000);  //go from 0 to 250 speed over
                                 //a period of 2 seconds
  beStill();                     //stop moving
  delay(3000);                   //wait 3 seconds
}
```

Wink_Ch03Motors_Ex06

Smooth acceleration from 0 to 250, over a period of 2 seconds (2000 milliseconds).

Wait 3 seconds before repeating.

1 SKILL LEVEL

A few more notes on motors...

If you do a bit of experimenting, you will likely realize that even if you set both motors to the same speed in the same direction, Wink doesn't travel in an exactly straight line. He will likely tend to curve off to one side or the other. This is because mechanical systems are never perfect. One motor will always produce a bit more power than the other, and one motor will always get a bit more traction on the surface - so one side will always tend to run a little stronger than the other. This can be adjusted in various ways as you'll see in upcoming chapters.