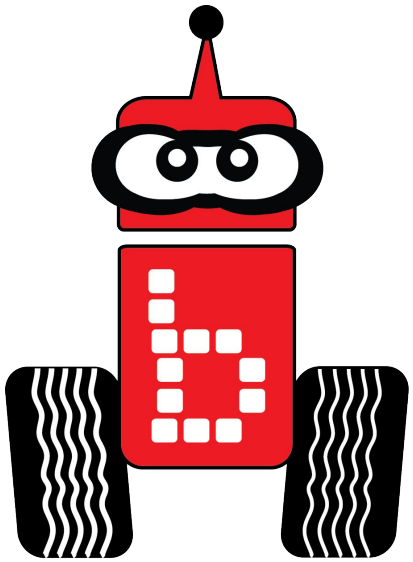


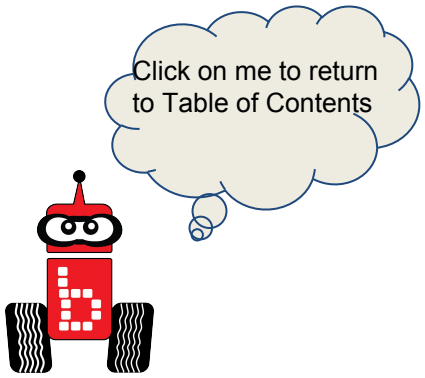
# Creating Functions Using Void

- **Key Concepts:**
  - **Students will familiarize themselves with the concept of functions and learn to write their own.**
- **Pacing:**
  - **Over several class periods**

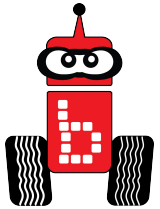
KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Table of Contents





# Introduction to Creating Your Own Function



## Goals

- Students will create their own functions

## Preparation

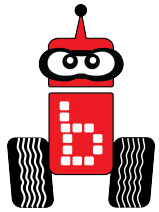
- Students must have a good understanding of functions and how to move the robot.

## Activities:

- Activity 1: Function
- Activity 2: Introduction to Creating Your Own Function
- Activity 3: Naming Your New Function
- Looking at Functions
- How do you Create a Functions
- 3 Components of a Function
- Activity 4: Function
- Activity 5: Right Turn
- Activity 6: Make a Square
- Activity 7: Make a Rectangle
- Activity 8: Make a Curvy Line
- Activity 9: Servos

KISS INSTITUTE FOR PRACTICAL ROBOTICS

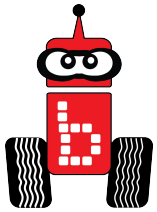
# Functions
# Activity 1



1. Write down as many functions as you know so far. Find the commonalities.
2. Use the strategy Fold the Line to discuss what you discovered.
3. Proceed to the next slide to find answer.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Functions
# Activity 1 continued



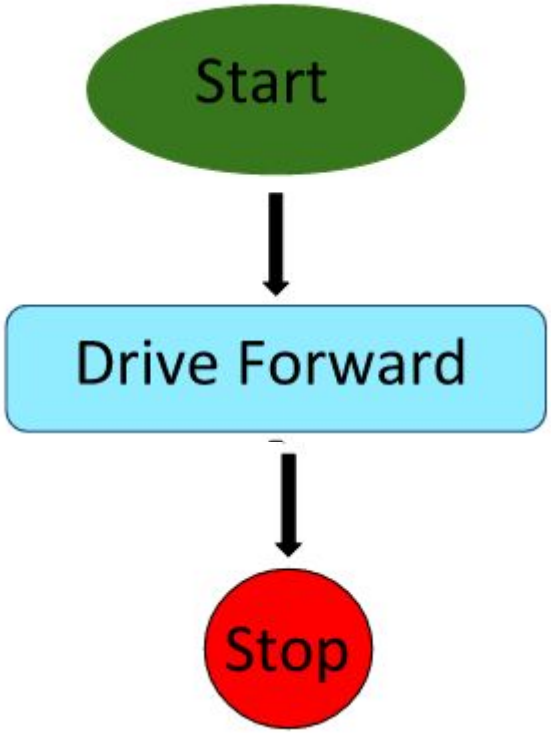
```
motor (0,100)
msleep (3000)
ao()
set_servo_position(3,1245)
enable_servos()
digital(8)
```

Did you discover these commonalities?

1. All functions have **names**
2. All functions have a set of parentheses (the "**argument list**")
   - Some functions need more information ("**arguments**"), such as `motor(0,100)` and `msleep(3000)`
   - Some do not need more information, such as `ao()` and `enable_servos()`

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Introduction to Creating Your Own Function
## Activity 2- Continued

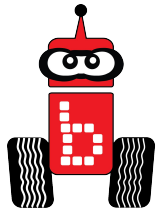



1. Write down the code that will drive the robot forward.

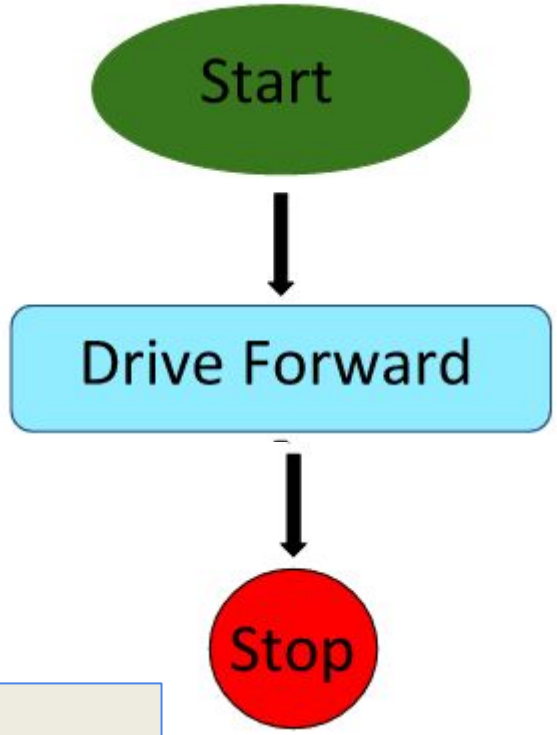
2. Proceed to the next slide.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Introduction to Creating Your Own Function
## Activity 2- Continued



```
int main ()
{
    printf("Hello, World!\n");

    motor (0,100); // move forward
    motor (3,100);
    msleep (1000);

    ao():
    return 0;
}
```

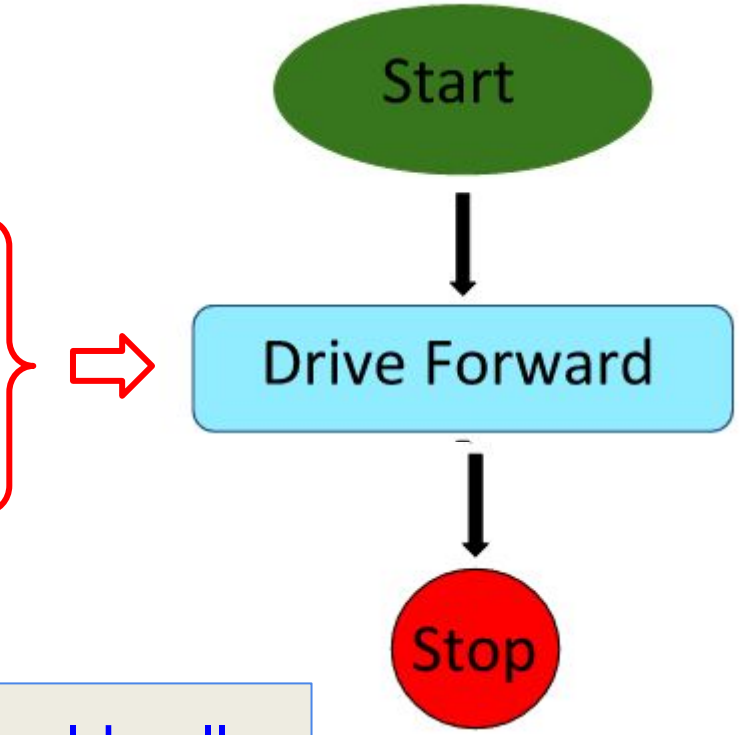



Wouldn't it be great to be able to make our code look like our flow chart!
Proceed to the next slide.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

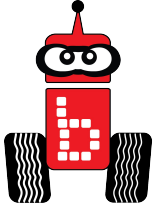
# Introduction to Creating Your Own Function
## Activity 2- Continued

```
int main ()
{
    printf("Hello, World!\n");

    motor (0,100); // move forward
    motor (3,100);
    msleep (1000);

    ao():
    return 0;
}
```




This section of code is what we would call "Drive Forward"
Proceed to the next slide.

# Naming Your New Function
# Activity 3



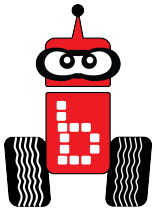
1. Have the students read and discuss the following slides:

   Looking at Functions
   How Do You Create a Function?
      Step 1: Function Prototype
      Step 2: Function Definition
      Step 3: Function Call
   Example of a Function
   Function Definition
   Function Call
   3 Components of Function Prototype
   Identifying the 3 Components of Functions

2. Use the strategy chant it, sing it, rap it to share out the information.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Looking at Functions



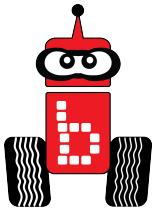
Computer scientist use conventions when naming/creating a function that they will use all the time:

- Make the name of your new function intuitive. Example: Drive forward means you want to drive forward for one second.
  - You must put an underscore for any blank space.

  drive forward = drive_forward()
  - It can not start with a number.
  - It can not have the same name as another function in the library.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Looking at Functions



**Functions** are helpful if you repeat actions multiple times:

- driving straight forward → `drive_forward();`
- making a 90° left turn → `turn_left_90();`
- making a 180° turn → `turn_around();`
- lifting an arm up → `lift_arm();`
- closing a claw → `close_claw();`
- doing a barrel roll → `barrel_roll();`
- driving straight forward → `straight();`

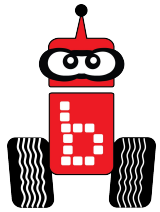
**We made these up… and that's the point!**

**You can write your own functions to do whatever you want!**

Proceed to the next slide

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# How Do You Create a Function?
# Step 1: Function Prototype



1. Decide what you are going to name your function (**function name**) that will move your robot forward (it should be intuitive); for example:

```
drive_forward or df
```

2. You will use the **function name** to create a **function prototype**
   - The list of function prototypes is similar to a list of vocabulary words

This is the "**function prototype**"

```
void drive_forward();
```

A **function prototype** has a semicolon at the end!

**Function Name**

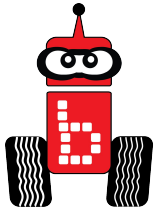
or

```
void df ();
```

Proceed to the next slide

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# How Do You Create a Function?
# Step 1: Function Prototype



3. You type your **function prototypes** on the line *after* `#include <kipr/botball.h>` and *before* `int main()`
   - The list of function prototypes is similar to a list of vocabulary words

This is the "**function prototype**"

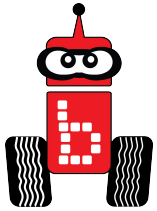
```
#include <kipr/botball.h>

void drive_forward();

int main()
{
  printf("Hello, World\n");
  return 0;
}
```

A **function prototype** has a semicolon at the end!

Proceed to the next slide

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# How Do You Create a Function?
# Step 1: Function Prototype



4. You type your **function prototypes** on the line *after* `#include <kipr/botball.h>` and *before* `int main()`
   - The list of function prototypes is similar to a list of vocabulary words

**What is `void`?**

Use `void` in your function prototype if you are ***commanding*** the robot to do something.

**Example:**

```
void turn_right();
```

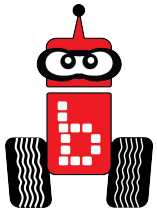
```
#include <kipr/botball.h>

void drive_forward();

int main()
{
  printf("Hello, World\n");
  return 0;
}
```

Proceed to the next slide

Click here for:
Additional Data Types- Advanced learning

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# How Do You Create a Function?
# Step 2: Function Definition



5. The **function definition** (i.e., what the robot will actually do) of the new function goes after the last "**}**"
   - The function definition is similar to the definition of a vocabulary word

Proceed to the next slide

**Example:**

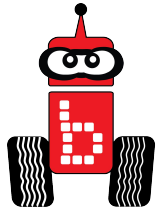
```
#include <kipr/botball.h>

void drive_forward();

int main()
{
  printf("Hello, World\n");
  return 0;
}

void drive_forward()
{
  motor(0,100); // forward
  motor(3,100);
  msleep(1000);

  ao();
}
```

This is the **function definition**.

A **function definition** does not have a semicolon at the end.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# How Do You Create a Function?
# Step 3: Function Call



6. You *use* the new function name (function call) you created in your block of code between the two curly braces; in place of the several line of code that you would have written before you created your new function.

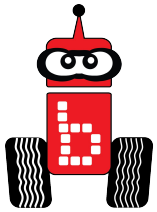
**Example:**

```
#include <kipr/botball.h>

void drive_forward();

int main()
{
  printf("Hello, World\n");
  drive_forward();
  return 0;
}

void drive_forward()
{
  motor(0,100); // forward
  motor(3,100);
  msleep(1000);

  ao();
}
```

This is the "**function call**".

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# How Do You Create a Function?
# 3 Components of a Function



These are 3 components of a function and should be created in this order:

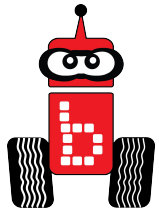
1. Function prototype
   a. Create a **function prototype** on the line *after* `#include <kipr/botball.h>` and *before* `int main()`.

2. *Function definition*
   a. *Define* the new function (i.e., specify what the robot will actually do). The **function definition** goes after the last "`}`" in your program.

3. *Function Call*
   a. *Call* (use) the new function by typing the **function name** within a block of code.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

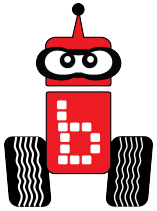
# My First Function
# Activity 4



1. Write a new function for driving your robot.
2. Remember to include the 3 components.
   a. Make sure the name is intuitive
3. Run your program.
4. Discuss with your partner's how this can be helpful.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Turn Right Function
# Activity 5



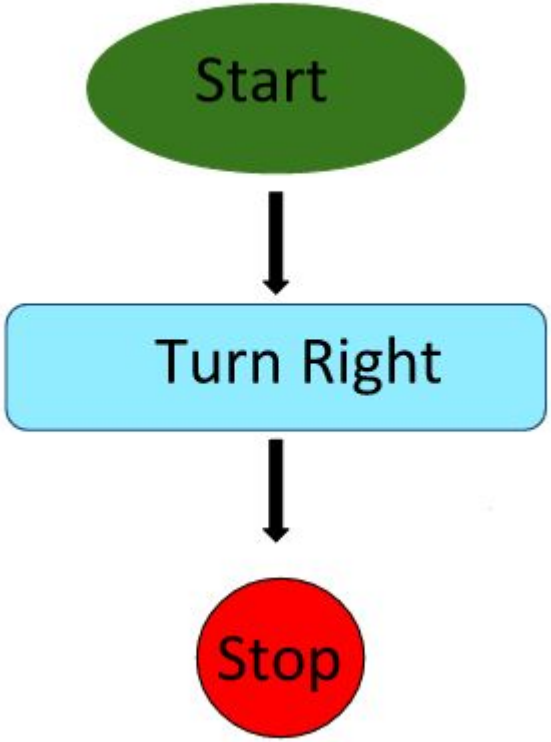
Goal: To create and write a function for your robot to make a right turn

1. Create a flowchart to make the robot turn right.
2. Share your flowchart with your partner.
   - Do they look the same?

Proceed to the next slide.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Turn Right: Activity 5 Continued



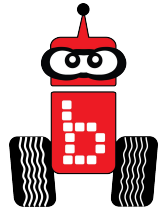

What would the code for this flowchart look like?

3. Write the code for a right turn.

*Proceed to the next slide

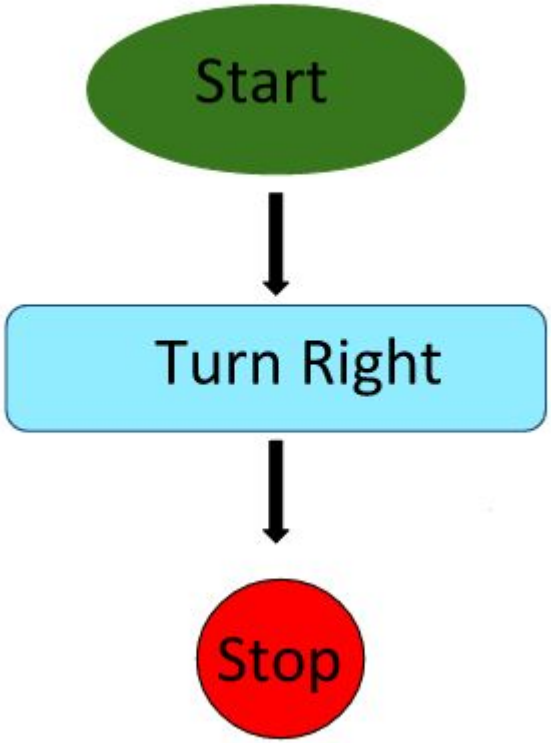
KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Turn Right: Activity 5 Continued



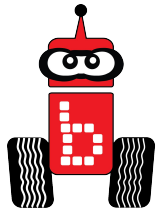
Example:

```
int main()
{
  printf("Hello, World!\n");

  motor(0,100); // turn right
  motor(3,-100);
  msleep(1000);

  ao();
  return 0;
}
```




KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Turn Right: Activity 5 Continued



```
int main()
{
  printf("Hello, World!\n");

  motor(0,100);
  motor(3,-100);
  msleep(1000);

  ao();
  return 0;
}
```

Turn Right

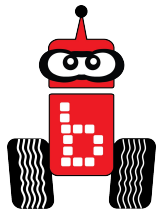
=

```
void turn_right(); // turn right for 1 sec

int main()
{
  printf("Hello, World!\n");

  turn_right();

  return 0;
}

void turn_right()
{
  motor(0,100); // right
  motor(3,-100);
  msleep(1000);

  ao();
}
```

4. Now write a function for that to turn right! This will let us have a right turn function in our library. Use a different name than the example.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Make a Square
# Activity 6



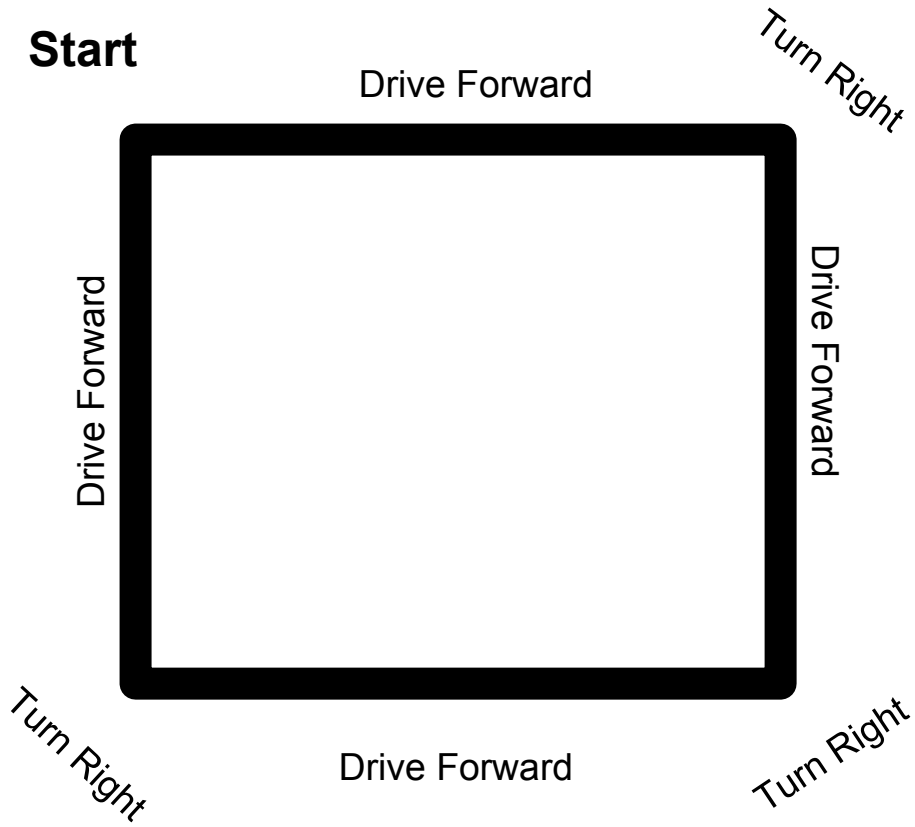
**Goal:** To create a flowchart and write functions to drive along the path of the perimeter of a square

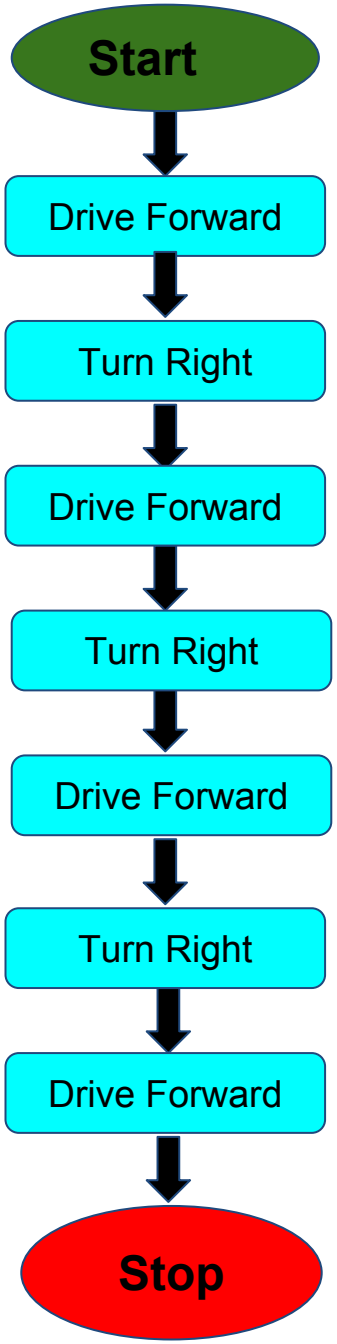
- Create a flowchart for this function
- Share your flowchart with your partner.
  - Do they look the same?

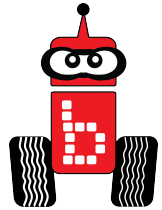
Proceed to the next slide to see if you flowchart looks similar.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Make a Square: Activity 6



Notice “Drive Forward” and “Turn Right” are repeated. Proceed to the next slide.

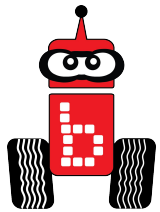
KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Make a Square: Activity 6 Continued

- Write the code to drive along the path of the perimeter of a square using the functions you have created.
  - How many functions do you need to create?
- Compile and run your code before proceeding onto the next slide.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Make a Square: Activity 6
# Answer Key



**Remember**
Semicolons *do* go at the end of **function prototypes**

```
void drive_forward(); // drive forward for 1 sec
void turn_right(); // turn right for 1 sec

int main()
{
  printf("Hello, World!\n");

  drive_forward(); // drive in a square
  turn_right();
  drive_forward();
  turn_right();
  drive_forward();
  turn_right();
  drive_forward();
  turn_right();

  return 0;
}

void drive_forward()
{
  motor(0,100); // drive forward
  motor(3,100);
  msleep(1000);

  ao();
}

void turn_right()
{
  motor(0,100); // turn right
  motor(3,-100);
  msleep(1000);

  ao();
}
```

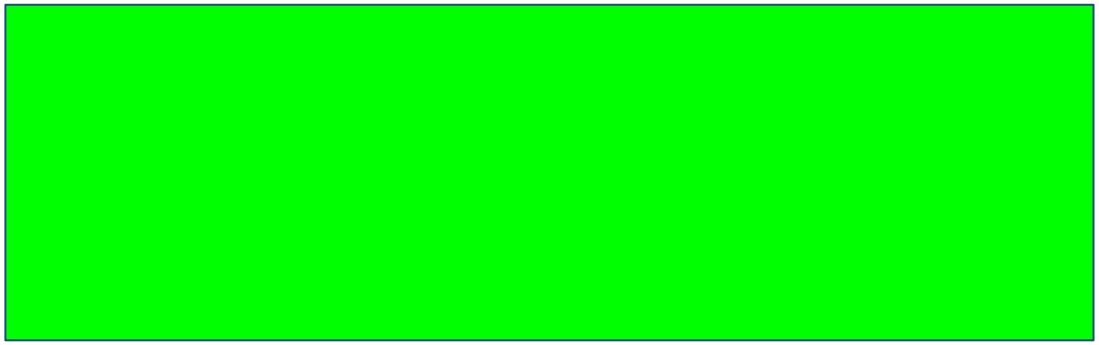
**Remember**
Semicolons *do not* go at the end of **function definitions**

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Make a Rectangle
# Activity 7

Goal: Student will write their own functions that will drive the robot in a rectangle.

1. Students should use code planning paper to write out their psuedocode or create flowchart.



KISS INSTITUTE FOR PRACTICAL ROBOTICS

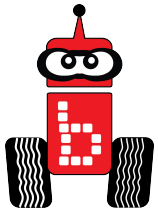
# Make a Curvy Line
# Activity 8

Goal: Student will write their own functions that will drive the robot in the path of a Curvy line.

1. Students should use code planning paper to write out their psuedocode or create flowchart.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Servo
# Activity 9



Goal: Student will write their own functions that will drive the robot in the path of a Curvy line.

1. Use the "Go Fetch" activity 9 from unit 9, except you must create your own functions.
2. Students should use code planning paper to write out their psuedocode or create flowchart.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

# Assessments and Rubrics

Suggestions: *Understanding* or *Group Collaboration* rubrics

KISS INSTITUTE FOR PRACTICAL ROBOTICS