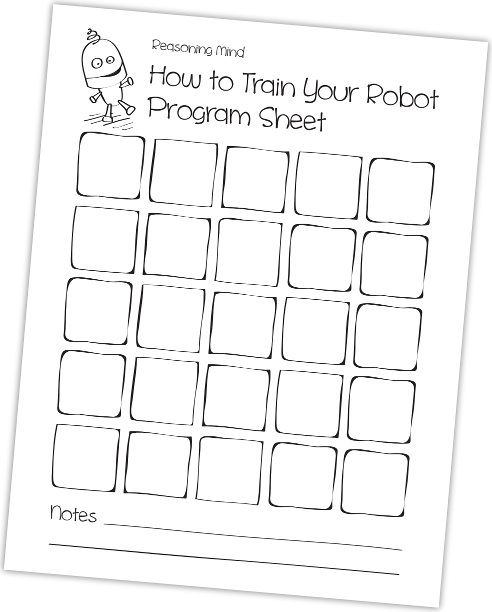
Reasoning Mind

How to Train Your Robot

Congratulations on the purchase of your brand new MVK-3000. You undoubtedly have many questions about the functionality of your robot, and it is our goal to answer them for you in this manual.

We're guessing you didn't order a robot to replace your recently broken coat rack. You want your robot to do things, and in order to do that you'll have to learn what to tell it so that it will carry out your wishes.



Reasoning Mind

How to Train Your Robot Program Sheet

Notes

To the right you can see an example of a program sheet. The purpose of a program sheet is to provide you with a place to put instructions for your robot.

Your robot will read whatever you put on the sheet and execute the commands to the best of its ability. It reads the sheet just like you would, left to right and top down.

Let's learn a few commands. The first few commands you should know are ones that will make your robot walk around.

↑ Forward

This command is known as **Forward**. It moves your robot forward one station, if it is at all possible.

↱ Right

This command is known as **Right**. It makes your robot turn 90 degrees to the right.

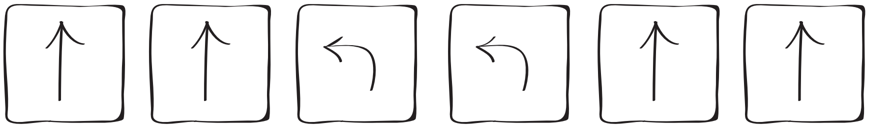
↰ Left

This command is known as **Left**. It makes your robot turn 90 degrees to the left.

This is a good place to pause and think about the things you can do with just these commands. Turn the page and decipher the programs on it.

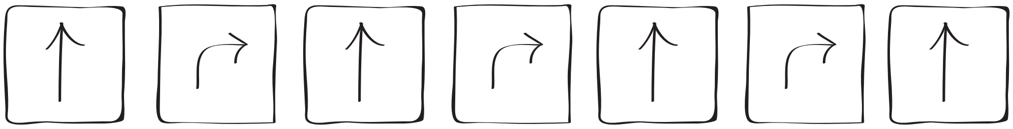
Program 1.

What will this program do?



Program 2.

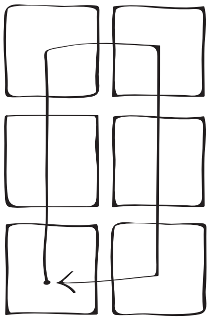
How about this one?

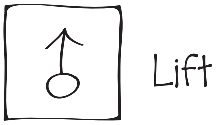


You might guess that you can't do too many things with your robot with only these three commands. You're right, but it's best to master the basics before you try something more complicated.

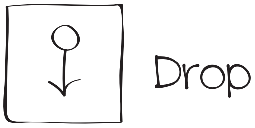
Exercise A.

Try writing a program that would have your robot walk a rectangle on a 2 by 3 grid of stations, shown to the right. Your robot starts in the lower left corner, facing up. This program will be long, but soon we'll have commands to shorten it up.





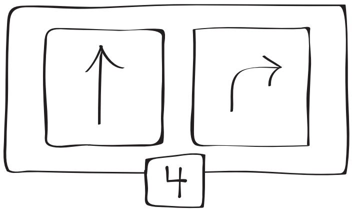
This command is known as **Lift**. If there are any objects at the same station as your robot, it will pick one up.



This command is known as **Drop**. If your robot is holding any objects, it will drop one at its current station.

Now we're getting somewhere! With the ability to pick up and put down objects, you can have your room cleaned by your MVK-3000 in no time!

Let's also add the idea of a **loop**. A loop is a way to make your robot repeat a sequence of commands a specific number of times. To make a loop, draw a box around the



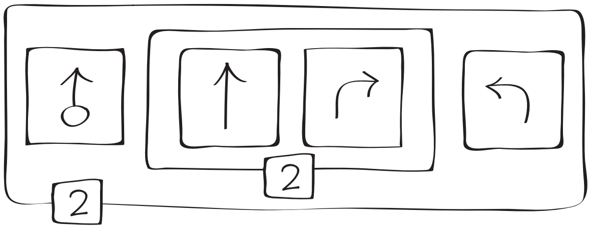
commands you want your robot to repeat, and write the number of times to you want them repeated underneath.

Program 3.

This program uses a loop to recreate a familiar program. Which one is it? There is *one* difference in the outcome of this program and the other one. Can you explain what the difference is?

Exercise B.

Make your rectangle program shorter by using a loop.



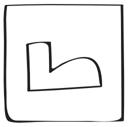
You can also put loops in your loops. This means you can make some very complicated programs.

Exercise C.

Your robot starts in the lower left corner of the 3 by 3 station grid, facing up and executes the program on the right. Where does your robot wind up, what direction is it facing at the end, and what has it done in the meanwhile?

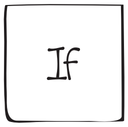
Still, it might seem like your robot can only perform the most basic of tasks. True, it could put away the dishes, but you'd have to tell it what to do for each and every dish in the dishwasher **and** you'd have to write a different program each time since there would likely be a different number of dishes in each load. Here is where **sensors** and **conditionals** come to the rescue.

Here's a sensor and conditional to start you off.

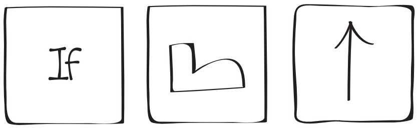


Feet

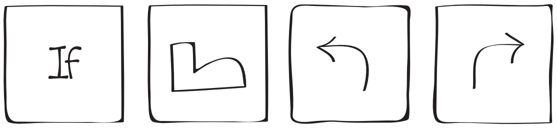
This sensor is known as **Feet**.
Your robot will let you know if it can walk forward.



This conditional is known as **If**. **If** does nothing by itself, but when a sensor follows it, magic happens. It's best explained by an example.



In order the commands are **If**, **Feet**, and **Forward**. **If** needs the next command to be a sensor. **If** executes the sensor, and if it succeeds then the command after the sensor is run. Otherwise, it is skipped and the program continues. This program says “If I can walk forward, do so.”



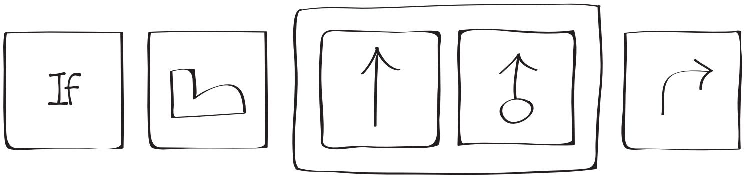
Program 4.

What will the program do if your robot is facing a wall?

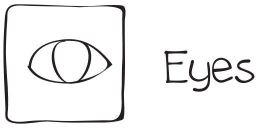
Now we can talk about **groups**. A group written like a loop, but without a number underneath. The important thing about grouping commands is that it allows your robot to treat the group as if it is one single command.

Exercise D.

What does this program do?



Here’s another sensor for you to play with:

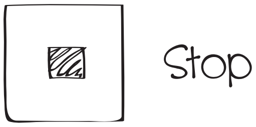


This sensor is known as **Eyes**. Your robot will report if it sees an object at its current station.

Exercise E.

Write a program which walks the robot in a small square, only trying to pick up objects if there are any at its feet.

These are the sensors in our standard MVK-3000. If you wish to install additional sensors, please make sure to contact us at our corporate address. Additional sensors can be installed at the price of $300 for one, or $3000 for two. Each additional one adds one zero to the end of the price. If you wish to have no additional sensors, please send us $30.



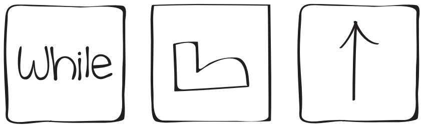
This command is known as **Stop**. Your robot will shut off, no longer executing any commands immediately. It is a way to bail from a program once you are happy with the results.

Lastly, here's a new conditional:

This conditional is known as **While**. **While** is a lot like an **If** that repeats the next command until the sensor fails. It checks the sensor at the end of the command, or group of commands.

Program 5.

What do you think this program does?



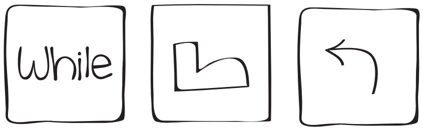
Exercise F.

Write a program which makes your robot walk the perimeter of the 3 by 3 grid. For this program you *must* use **While**. *Hint:* Don't forget loops, they'll make your program much smaller.

This conditional is known as **Until**. **Until** is the opposite of **While**. **Until** does the next command (or group of commands) and stops once the sensor succeeds.

Program 6.

What does this program do?



Exercise G.

What happens if the robot can already walk forward?

Exercise H.

Write a program which has a robot walk forward stopping only once it can pick something up.

Tasks

Now the real fun begins. Here's a list of tasks for you to accomplish with your new robot. Each time, try to make your program as short as possible.

Task 1.

Your robot is on the 3 by 3 grid, standing on A facing B. As long as your robot can move forward, it will pick something up, turn right, and step forward.

Bonus: Make it so that once the robot picks up everything it eventually stops.

Task 2.

Your robot is on the 1 by 8 grid, standing on A facing B. Two objects are placed at random, but separate stations in front of it. Write a program which will make it walk past the first object, and pick up the second.

Task 3.

Your robot is on the 1 by 8 grid, standing on A facing B. Write a program which will let it patrol the grid (walk from corner to corner) indefinitely.

Bonus: Your robot will start at a random station facing a random direction.

Bonus: Make your robot pirouette (do a full turn) on every square.

Task 4.

Your robot is on the 1 by 8 grid, standing on A facing B. You want your robot to move all the objects from A to H. Here, your robot can only hold one object at a time, and you don’t know how many objects there are in advance.

Task 5.

You have two robots which start at opposite ends of a 1 by 8 grid. They need to find each other and shut off in the same station. Each robot will have its own program. *Note:* Robots count as objects for the **Eyes** sensor.

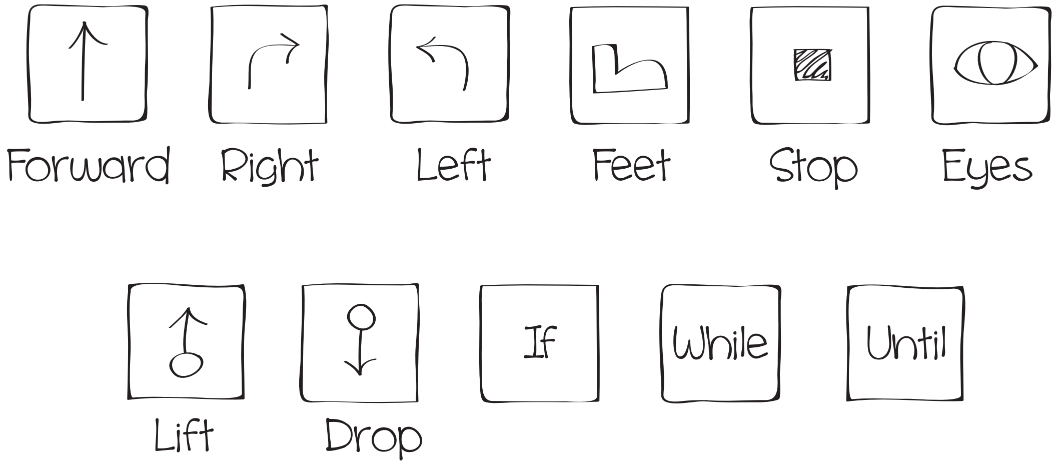
Bonus: There is now a third (decoy) robot in a random station.

Bonus: Your robots must not shut off in the same station as the decoy.

Task 6.

Your robot is on a 3 by 3 grid, starting in the lower left corner facing up. It needs to end in the upper left corner. There will be walls blocking the way, but you don’t know where. If there is a wall between two stations, then you can not go from one to the other. Write a program which gets your robot to the other side safely.

Commands


Forward
Right
Left
Feet
Stop
Eyes
Lift
Drop
If
While
Until