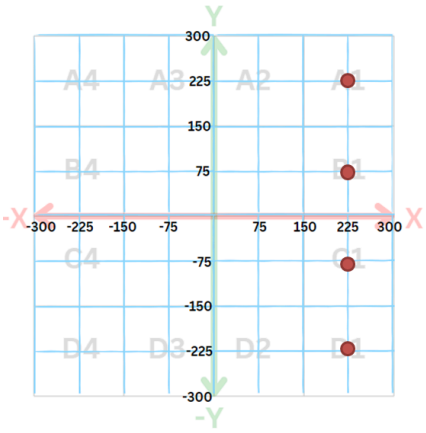
Lesson 11 Functions: Jobs and Subjobs

Vocabulary

Function- a sequence of commands that can be reused together later in a program.
Call- a request in the program to run a function.
Job- a unit of work or unit of execution.
Subjob- a component of a job, sometimes called a step.

Direct Instruction

Engage with ORA Motion Types



Open the **EngageL11Code** or Share Code: 8osbry6, and have it ready to run on a connected ORA. Set up the Engage activity by placing a wooden cylinder, on end, in the center of A1, B1, C1, and D1.

Before you run the code, tell the students, as they watch the code being executed, to think about how the series of commands could be "chunked up" into a set of commands. Run the code a few times, resetting the cylinders between runs. Direct students to the Engage section of the Student Activity Sheet and have them break down the code they observed into a few different tasks. After they try to chunk the code into tasks, show the Engage code on the display so students can see the code consisting of only 7 blocks and the functions they use.

Next, show the Lesson 11 video.

Explore the Purpose of Functions

Open up the **ExploreL11Code** and display it so students can view it. Alternatively, have students open share code: **osee2nj** in the Ozobot Editor to view the code. The code has four functions that have already been defined.

For each function, students should decide on a name for each and give a brief description of what the function code does. Direct students to the Explore section of the Student Activity Sheet and have them fill out the chart based on their answers.

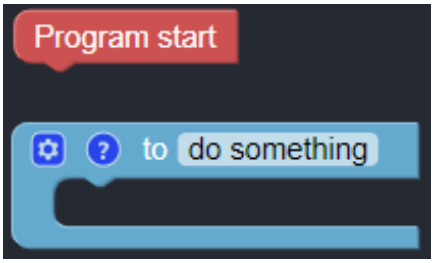
Explain Functions in Ozobot Editor

As students saw in the Engage section, whole tasks (or jobs) can be broken down into subtasks (or subjobs). Ask the students to think of an everyday task that could be broken down into subtasks/subjobs. Get a few responses from the students. One example could be making a peanut butter and jelly sandwich. "Make a PB&J" is the overall task/job, and "get bread" or "apply jelly" are subtasks/subjobs.

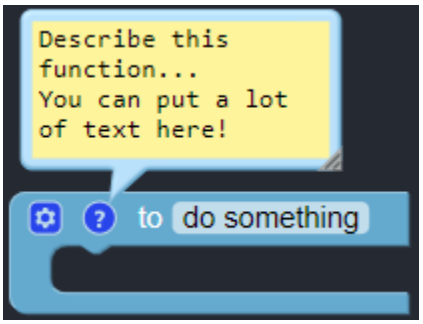
Tell students that in programming, functions allow us to repeat a code sequence without having to write the code over again. This allows us to simplify a

complex action by using only one or a few commands. As they saw in the Engage section, the long, complex series of tasks was reduced to seven blocks of code!

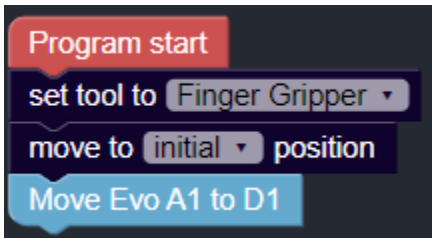
Open up the Ozobot Editor and go to Level 4 or 5. Click on the Functions tab on the bottom of the left-hand menu. At the top is the **to do something** block. **Do something** is a placeholder for the function's name.

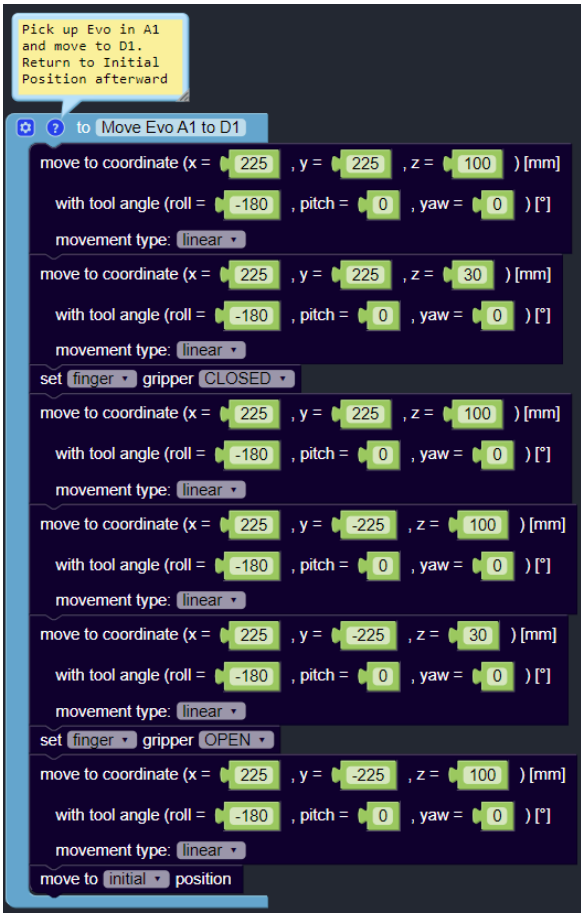


When the **to do something** block is added to the workspace, it does not connect with the **Program Start** block or any other block because a function is built within the **to do something** block.



On the **to do something** block, there is a light blue area to name your function and a blue question mark. This question mark button allows the user to add details about the function, how it behaves, when a user should call the function, or other relevant information.





To build your function, drag blocks into your **to do something** block. When you are done making your function, it is good practice to add a description for what action the function performs.

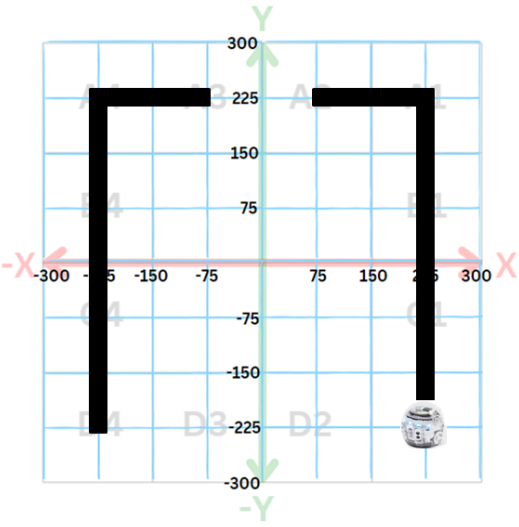
To "call" or use your newly created function, go to the Functions category again. At the bottom is a new block with the name of the newly created function.

Drag the new function block to the Program Start. If the function does not have the blocks to set the end tool or move to the initial position, add those before the function block.

Several functions can be created in the same program, so the user needs to pick and choose which functions they need to complete a task!

Elaborate: Create and Call Your Own Functions

In this challenge, students will define and call functions independently. Then, students will combine functions using concurrency to increase efficiency.



Setup: Place Evo in D1. Use Line Cutouts to make a path from D4 to A3 and then a path from A2 to D1 (8 cutout lines, see image, next page).

Task Requirements:

1. ORA must start in the Initial Position.
2. Students must create **functions** to:
    a. Program ORA to move Evo from D1 to D4.
    b. Program Evo to follow the line from D4 to A3 and begin to flash the top lights.
    c. Program ORA to move Evo from A3 to A2.
    d. Program Evo to follow the line from A2 to D1 and begin to play at least 4 notes.
3. Students must also use concurrency block(s) to perform all of the same actions but in a shorter time frame. (Times of function-only and using concurrency will be reported in the Evaluate Section).
4. ORA must finish in the Initial Position.

    Direct students to plan this code in the Elaborate section of their Student Activity Sheet. After creating the pseudocode and drawing the code, direct them to create this code on their devices in the Ozobot Editor. Ensure students are connected to your shared workspace.

    As students complete their code, they will test their code. Click on a given student's or student group's tab on the Ozobot Editor. Click "Run" to execute their code. Reset the Evo between each code run (if needed). Remember to follow the Safety Protocol established in Lesson 1.

    If students are waiting for the ORA, they could:

    - Answer Questions on the Student Activity Sheet
    - Complete the Evaluate section of the Activity Sheets.

Evaluate: Exit Ticket

Students will fill out and submit the Evaluate section of the Activity Sheets as their exit ticket.

Optional Extension: Packaging Plant Cobot

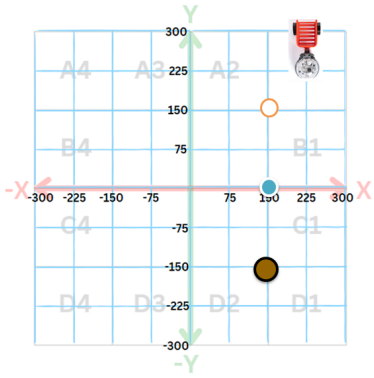
In this activity, students will assume the role of an Automation Engineer in an automated candy packaging plant. The automation engineer breaks a job into its subjobs and programs functions to be called to complete the objective.

The job of this lesson is to package candy for shipping. To do this, we will use an Evo Trailer to receive the packaging and move it around the warehouse facility.

Source: Ozobot

In this task, students will program Evo to move along a candy packaging assembly line as ORA places a cup in Evo's trailer in A1, a candy in Evo's trailer in B1, and a lid in C1. Evo will then take the packaged candy to the warehouse in D4

Setup: Place Evo in A1. Place a Dixie cup at the corner of A1/A2/B1, a candy at the corner of B1/C1, and a student-created 'lid' made of cardboard with a tab at the corner of C1/D1/D2.



Task Requirements:

1. ORA must start in the Initial Position.
2. Students will break the task into **functions** and complete the task using function blocks.
3. Evo must be programmed to move WITHOUT using "line follow" blocks
4. ORA must place a cup in Evo's trailer in A1, a candy in the cup when Evo is in B1, and place a lid on the cup when Evo is in C1.
5. ORA must return to the Initial Position when complete.
6. Students will use concurrency to decrease the amount of time it takes to load, package, and deliver the candy.