

Robotics 101

[2]: Navigation

Lesson Plan

Topic Focus

Students will investigate forward and backward motions using code.

- This challenge is a continued progress of the Robotics course, following Robotics 101 [1]: The Build lesson.
- The lesson starts out with an introduction to driving forward and backward, using positive and negative values in code. Students will also learn to change the distance traveled by testing and manipulating python code. The lesson progresses to more advanced tasks where students have the opportunity to apply the knowledge gained.
- The approximate total length in time for the *review, introduction and preparation* phases is 10 minutes. The *testing phases* should total approximately 20 minutes and the *exploration phases* should take no longer than 25 minutes combined. The teacher can determine the amount of time allocated for the mission phase, but it is expected to take 10 – 20 minutes.
- Extra time may be needed for debugging and fixing code.
- Interdisciplinary topics: distance, time, displacement, speed, integers, equations, metric measurements, trials, data collecting, modeling
- Before you start this challenge, make sure learners have read the Publisher's Plan, so they know how to record important achievements as they build their projects.

Description

This is the second in a series of lessons designed to get you going with your robotics kit in short increments. Last week you built the chassis of the rover, today we're going to take it for a drive.

Standards

CSTA: 2-CS-02, 2-CS-03, 2-DA-08, 2-DA-09, 2-AP-13, 2-AP-17, 2-AP-19, 3A-CS-02, 3A-DA-09, 3A-AP-17, 3A-AP-18
CC.MATH: 6.NS.C.5, 6.RP.3, 6.NS.5, 6.EE.9, 7.RP.1, 7.EE.3, 7.EE.4, 8.EE.5, 8.SP.3, N-Q.1, N-VM.1, A-SSE.4, A-REI.1, A-REI.2, F-LE.1, S-MD.7
NGSS: MS-PS2-2, HS-ETS1-2

Learning Goals

Domains	Experiences	Skills
<p>This project explores the following domains, or content areas:</p> <ul style="list-style-type: none">• Physical computing• Programs/ code scripts• Using robotics and code to control and model distances• Distance, time, and displacement• Metric measurements	<p>In completing this project, learners will experience the following:</p> <ul style="list-style-type: none">• Coding in Python• Importing from libraries• Variables and loops• Wiring components and controlling them with code• Collecting data and using it to debug errors• Positive and negative values to control direction	<p>In completing the project, learners will work on developing the following skills:</p> <ul style="list-style-type: none">• Problem solving• Iteration• Procedural thinking• Systems thinking

Challenge

Introduction

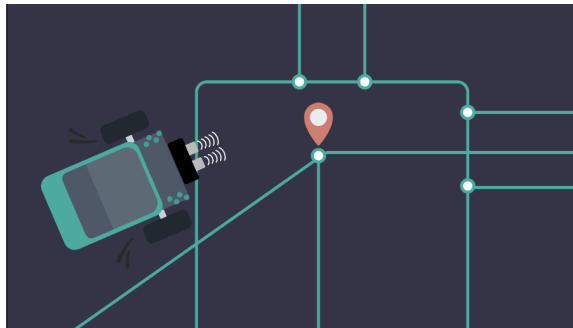
Time: TBD by Teacher

In the previous lesson...

In the previous lesson, you tackled the job of putting together the rover and you were able to observe a little of what it can do.

In this lesson...

Today we will start learning how to control the rover on our own, using code! You will test out sample code and then do some exploration to control what the robot does on your own.



Explanation: Teachers can invite students to complete this section as a class or small group discussion. It can also be completed by students individually. Questions can be modified or extended.

Tip: The complexity of the review and questioning is dependent upon the student age group. It is recommended that teachers use higher order questioning for older students.

Preparation

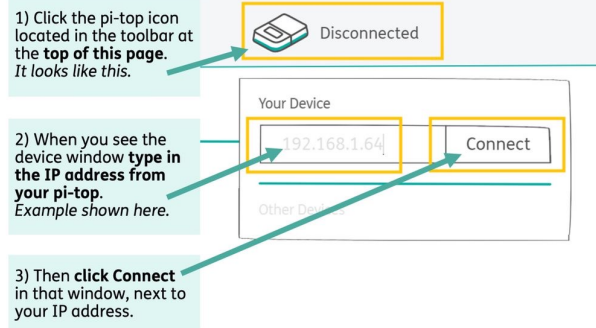
Time: 5 minutes

Connect the Rover to Further

If the pi-top[4] is not attached, please attach it to the expansion plate at this time.

When the pi-top [4] is connected to the chassis:

1. Power it up (if it's not already).
2. Onboard the pi-top [4] if you haven't already.
3. Click the blue, down arrow until you see the wifi screen (2 clicks). Stop here and wait for the IP address to be displayed.
4. Take a note of the IP address of the system. It's most likely something like 192.168.0.2
5. Click on the pi-top [4] icon above (next to the word "disconnected") and enter the IP address. Click "connect".
6. The state should change from "disconnected" to "connected". We can now send code to the pi-top [4].



Explanation: This section covers connecting the pi-top to Further to be able to run and test the code provided throughout the lesson.

Testing 1

Time: 5 - 10 minutes

In this phase you will learn how to control the rover's forward motion.

Test the Code:

Follow the instructions to run the sample code.

The code you will test in this section is written to move your rover forward.

- At this time place your rover on the floor. If your teacher designates a certain area please bring your rover there.
- In the student editor window you will see a .py file tab labeled, 1forward.py
- Check to make sure that this tab is highlighted green. Any code that will run when the play button is pressed is the code written in the file marked green.
- Click the PLAY button at the top of the student editor window, to run the 1forward.py sample code.

This code makes it so that the Rover moves forward 1/4 of a second.

```
#//////////////////// MOVE FORWARD //////////////////////
```

```
from pitop import Pitop
from pitop.robotics.drive_controller import
DriveController
from time import sleep
```

```
robot = Pitop()
# Notice that the objects in our code, match the
physical object itself (the robot)
drive = DriveController(left_motor_port="M3",
right_motor_port="M0")
robot.add_component(drive)
```

```
robot.drive.forward(0.5)
sleep(0.25)
```


```
robot.drive.stop()
```

Explanation: Students are given directions to set the rover in a specified area and to run the sample code. The purpose is to observe the outcome of a forward command.

Tips: It is recommended that teachers designate a specific area that is free from items/ furniture for testing. If possible, allow for multiple testing areas so that more students can test at the same time. Teachers can mark a “starting” line in the space(s) so that students can observe and measure distance.

If the rover reaches an obstacle or the end of the area, please have students stop the code or pick up the rover so that it is not damaged. If this situation occurs, the value contained inside of the sleep code () should be lowered.

Code Breakdown
<p>Review the graphic on the right to understand more about the code you ran.</p> <p>Review the information about each line in the graphic on the right. The first section of code imports the necessary modules of code from our libraries. The next section of code deals with the rover! And the last section of code is the while loop. The code contained inside of it will run until the program is stopped.</p>
Question:
How far did the rover travel?
Python Principles: Variables
More information and exercises regarding variables.

<div><p>The first line imports the Pitop file code from the pitop library created by our engineers. The second line imports the DriveController function from the robotics library so that the motion codes can be used. The third line will allow us to use the sleep function in python's time module so that we can specify time.</p><pre>from pitop import Pitop from pitop.robotics.drive_controller import DriveController from time import sleep robot = Pitop() drive = DriveController(left_motor_port="M3", right_motor_port="M0") robot.add_component(drive) robot.drive.forward(0.5) sleep(0.25)</pre><p>This line specifies the ports that the motors are plugged into on the expansion plate.</p><p>This line tells the computer to move the rover forward at 0.5 m/s.</p><p>The sleep code is used to tell the computer how long to do something. It will wait or keep running code for a specified number of seconds.</p></div>	<div><p>VARIABLES</p></div>
---	--

<p>Explanation: Students will review the parts of code written in the script. Attention should be focused on all of the sections, but careful review of the forward command is important.</p> <p>Tip: If the students are working through the lesson individually then they would review this information on their own. The teacher may decide to pace the class and discuss this information in whole group.</p> <p>This information could be used on an assessment, that would be given at the end of the lesson.</p>	<p>Teachers may want to have students use the same value so that they have one correct answer. This distance is important for the next section.</p> <p>This is a mini lesson in Further that teachers can incorporate in class or assign to specific students. It expands upon the Python concept – variables – and is ideal for students that have no foundational knowledge of the topic. It also can be assigned to provide remediation on the topic.</p>
---	--

**Larger diagrams are included at the end of this teacher guide.*

Exploration 1		Time: 5 - 7 minutes
These are small tasks to modify code or add your own code and explore things you have learned in the testing phase.	<p>Sample Answer Key –</p> <pre>#////////////////////// MORE DISTANCE //////////////////////// from pitop import Pitop from pitop.robotics.drive_controller import DriveController from time import sleep robot = Pitop() drive = DriveController(left_motor_port="M3", right_motor_port="M0") robot.add_component(drive) robot.drive.forward(0.5) sleep(0.5) robot.drive.stop()</pre>	<p>Explanation: Students will manipulate the code that was introduced in the last phase. The goal is to explore how the code can be changed to double the distance that the rover travels.</p> <p>Tip: Student answers may be different than what the answer key shows, but the value marked in yellow is what students should manipulate. It is recommended that teachers mark or determine a starting line for the students to use when testing.</p>
<p>Explore to extend!</p> <p>Practice Exercise 1</p> <p>In the program on the right, try changing the code so that the rover travels forward at double the distance.</p>		
<p>Question:</p> <p>How did you change the code so that the distance was doubled?</p>	<p>Sample Answer:</p> <p>I discovered that when the value written inside of the sleep function is multiplied by 2 then the distance travelled is doubled.</p>	<p>Answers may vary but should be focused on multiplying the sleep code value by 2. A possible answer is provided to the left of this box.</p>
<p>Question:</p> <p>How far did the rover travel this time?</p>		<p>Answers may vary depending on the amount of time passed in the sleep code; however, it should be double the distance travelled during testing.</p>
<p>Explore to extend!</p> <p>Practice Exercise 2</p> <p>Change the code so that you can explore how to move the rover various distances. This will prepare you for future exercises and lessons.</p>	<p>Sample Answer Key –</p> <pre>#////////////////////// MORE DISTANCE //////////////////////// from pitop import Pitop from pitop.robotics.drive_controller import DriveController from time import sleep robot = Pitop() drive = DriveController(left_motor_port="M3", right_motor_port="M0") robot.add_component(drive) robot.drive.forward(0.5) sleep(0.5) robot.drive.stop()</pre>	<p>Explanation: Students will manipulate and run test trials to see what happens when the value marked in yellow is changed.</p> <p>Tip: Teachers may want to limit the amount of trials/ tests the students should do, depending on time constraints.</p>

Testing 2

Time: 5-10 minutes

In this phase you will learn how to take the rover in the reverse direction.

Reversing the Rover

Now that you know how to move the rover forward at different distances, we can look at another way to manipulate the Rover's motion.

Follow the directions below to complete the exercise. This program or script will look similar to the previous phase, but we need to make one small change.

- Look at the line in the block below to see if you can spot the change.
- Add the line to the 2reverse.py script in the appropriate place (noted with a green comment).

```
robot.drive.forward(-0.5)
```

```
##### REVERSE #####

from pitop import Pitop
from pitop.robotics.drive_controller import DriveController
from time import sleep

robot = Pitop()
# Notice that the objects in our code, match the physical
object itself (the robot)
drive = DriveController(left_motor_port="M3",
right_motor_port="M0")
robot.add_component(drive)

#add the line to reverse
sleep(0.25)

robot.drive.stop()
```

Students will learn how to move the rover in a reverse direction using code

Explanation: They are instructed to add the line of code provided so that the rover moves in the reverse direction.

Tip: If students paste the code on the right of the comment line, then Python will recognize it as part of the comment. Students can replace the comment with the reverse command or press enter/return to produce a blank line under the comment and then add the command.

Test the Code

Follow the instructions to run the sample code.

Once you have made the changes, click the PLAY button at the top of the student code editor window.

```
Sample Answer Key
##### REVERSE #####

from pitop import Pitop
from pitop.robotics.drive_controller import DriveController
from time import sleep

robot = Pitop()
# Notice that the objects in our code, match the physical
object itself (the robot)
drive = DriveController(left_motor_port="M3",
right_motor_port="M0")
robot.add_component(drive)

#add the line to reverse
robot.drive.forward(-0.5)
sleep(0.25)

robot.drive.stop()
```

Explanation: Students are given directions for running the code to test and observe how direction can be reversed using a negative speed value.

Tip: Like the first testing phase the sleep code time may need to be adjusted. If the rover reaches an obstacle or the end of the area, direct students to stop the code or pick up the rover so that it is not damaged. If this situation occurs, the value contained inside of the sleep code () should be lowered.

Exploration 2

Time: 5-15 minutes

These are small tasks to modify code or add your own code to explore things you have learned in the previous phase.

Explore motions!

Practice Exercise 1

In the 2reverse.py file, modify or add to the code so that the rover goes forward, then backward, then forward and backward again.

The stop command may be a necessary additional if the rover doesn't stop when intended.

Sample Answer Key

```
#////////// EXPLORE MOTIONS //////////
```

```
from pitop import Pitop
from pitop.robotics.drive_controller import DriveController
from time import sleep
```

```
robot = Pitop()
# Notice that the objects in our code, match the physical
object itself (the robot)
drive = DriveController(left_motor_port="M3",
right_motor_port="M0")
robot.add_component(drive)
```

```
robot.drive.forward(0.5)
sleep(0.25)
robot.drive.forward(-0.5)
sleep(0.25)
robot.drive.forward(0.5)
sleep(0.25)
robot.drive.forward(-0.5)
sleep(0.25)
```

Explanation: Students will manipulate the code that was introduced in the lesson thus far. The goal is to explore how to move the rover in a cycle between forward and backward motions.

Tip: The sleep time may need to be modified depending on the physical space used.

Explore the past and present!

Practice Exercise 2

Write a program so that your robot begins at a starting line, goes forward 0.6 meters (2 feet), returns to the starting line, goes forward 1.2 meters (4 feet), then returns to the starting line, and stops.

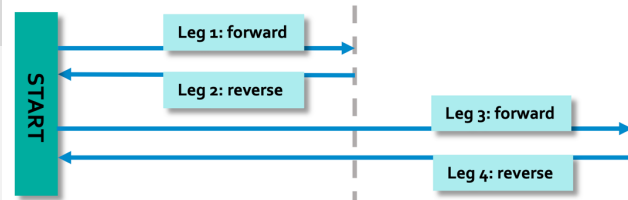
If you need to see a visual of this exercise, see the diagram in the section below.

Explanation: Students will complete a specific forward and backward motion drill exercise. The diagram is located on the next page of the teacher guide. A larger graphic is provided at the end of this document.

Tip: It is recommended that teachers mark where the robot should start. To make the exercise easier teachers can mark the 0.6 meter and 1.2 meter distances. Possible materials to use: mild adhesive tape, electrical tape, sticky notes, etc. But the material should lay flat on the floor so that the robot does not interfere with the marks.

Sample Diagram for Exercise 2

Review the diagram on the right side for a visual of the motions the exercise.



Explanation: This is the diagram provided to students for exercise 2.

**Larger diagrams are included at the end of this teacher guide.*

Missions

Time: 10 - 20 minutes

You are now ready to complete your own rover missions! At this time, you will complete two missions to showcase what you have learned.

Push the Rock!**Mission 1**

Crush up some pieces of paper to form 2 "rocks". Or if you have 2 plastic cups laying around, small empty boxes, or very light objects those can be used as a substitution.

Place the robot on the ground, then place a "rock" in front of it and one behind it. Write code in the 3mission1.py file to see how far apart you can push them apart with the rover, in your given space.

Reflection Question:

What are 2 successes that you encountered in this lesson?

Reflection Question:

What are 2 challenges that you encountered during this lesson, and how did you solve them?

Explanation: Students will use what they have learned in the lesson to complete a mission task without any specific code provided to them.

Tip: Teachers can direct students to use any code provided in the lesson or code they created to complete the mission.

Students will need to test and refine their code multiple times to create a working program; therefore, the time allotted for the mission may need to be extended.

Students will reflect on the lesson and detail the successes they had. Answers will vary.

Students will reflect on the lesson and detail the problems/issues they had and describe how they solved them. Answers will vary.

Electronics Kit Extension

Time: 10 - 20 minutes

The following sections require the use of the Electronics Kit. If you have the kit or are interested in purchasing it see the information below.

What is the Electronics Kit?

The image on the right shows what is inside of the kit. It includes different sensors, electrical components, and wires.

Don't have the kit?

Find out more here: <https://www.pi-top.com/bundles/electronics-kit>



Note: This phase of the lesson is appropriate for those with the electronics kit. If you do not have it, we suggest that you remove this phase of the lesson for students.

If you are interested in purchasing the kit, information to do so is provided on the left.

Beep Beep!

Mission 2

If you have a buzzer component, add it to your device and modify/ write code in the 4mission2.py file, so that the buzzer indicates when the rover is going in reverse, or backing up.

- A digital port, indicated with a D, must be used.
- The code uses digital port D0; however, if you want to use another port you modify that port number in the code.

Note: Sample code has been provided for you so that you can see what code is necessary for the buzzer.

```
#//////////////////// BUZZER REVERSE //////////////////////////////////////
```

```
from pitop import Pitop
from pitop.robotics.drive_controller import DriveController
from time import sleep
from pitop.pma import Buzzer #import the code from an
                              external file to use the buzzer
```

```
robot = Pitop()
# Notice that the objects in our code, match the physical
object itself (the robot)
drive = DriveController(left_motor_port="M3",
                        right_motor_port="M0")
robot.add_component(drive)
```

```
#----- Variables -----
buzzer = Buzzer("D0") #buzzer variable to declare
                        which port it is plugged into
```

```
while True:
```

```
    buzzer.on()
    sleep(0.5)
    buzzer.off()
```

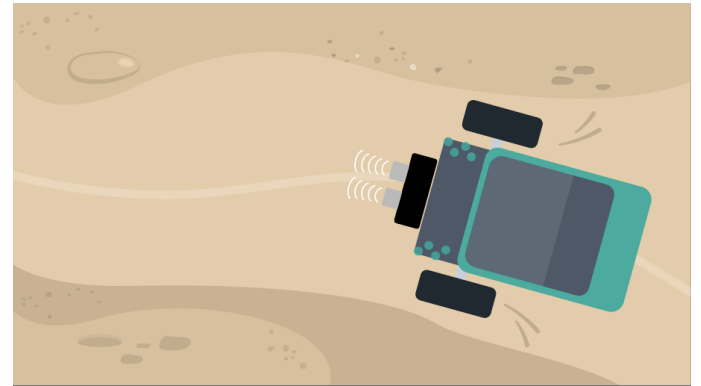
Explanation: A electronics kit is needed for this mission. If your class only has the robotics kit then you may want to delete this mission in advance.

Tip: If students need more practice using the buzzer, direct them to complete the pi-top[4]: Component Tutorial on the buzzer.

Resources and Related Content

Here are some other projects or resources you may find useful. More Robotics lessons:

- Robotics 101 [1]: The Build
- Robotics 101 [3]: Rover Race
- Robotics 101 [4]: Orbit
- Robotics 101 [5]: Maneuvering the Terrain
- Robotics 101 [6]: Ultra Control
- Robotics 101 [7]: Rover Vision
- Robotics 101 [8]: Path Finder
- Robotics 101 [9]: Natural Evasion



The first line imports the Pitop file code from the pitop library created by our engineers. The second line imports the DriveController function from the robotics library so that the motion codes can be used. The third line will allow us to use the sleep function in python's time module so that we can specify time.

```
from pitop import Pitop
from pitop.robotics.drive_controller import DriveController
from time import sleep

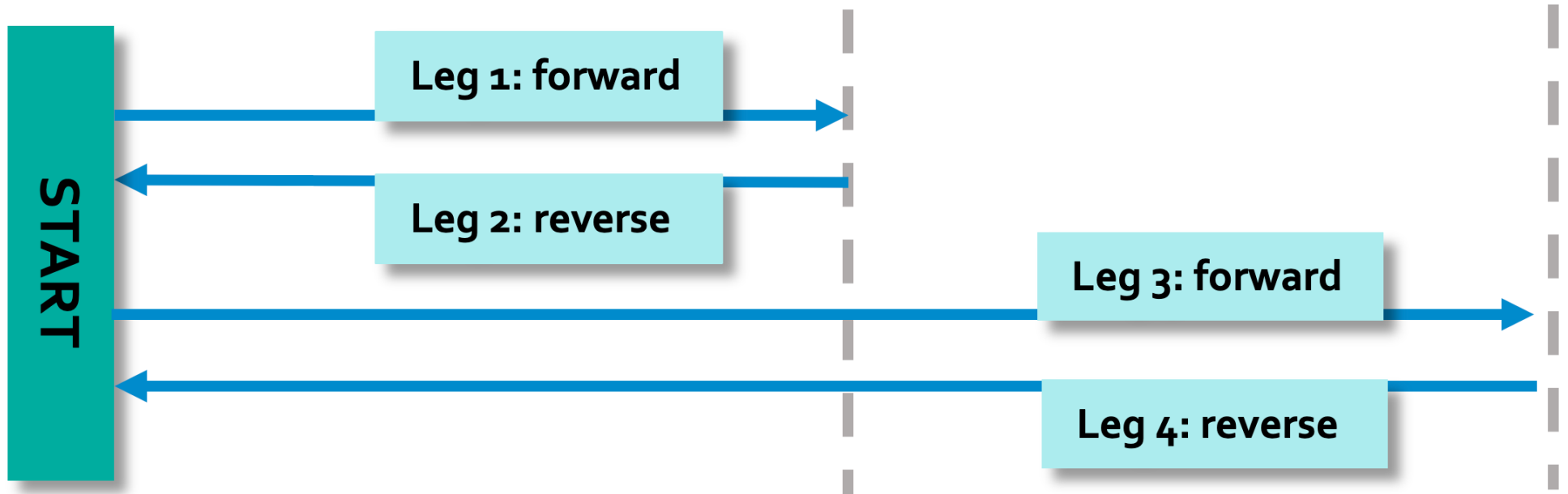
robot = Pitop()
drive = DriveController(left_motor_port="M3", right_motor_port="M0")
robot.add_component(drive)

robot.drive.forward(0.5)
sleep(0.25)
```

This line specifies the ports that the motors are plugged into on the expansion plate.

This line tells the computer to move the rover forward at 0.5 m/s.

The sleep code is used to tell the computer how long to do something. It will wait or keep running code for a specified number of seconds.



Exploration 2 Graphic