pi-top

Robotics 101

Scope and Sequence

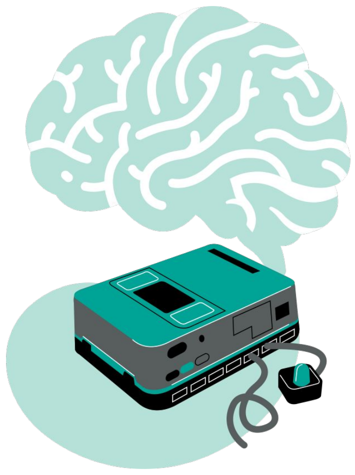
Summary

Robotics 101 is an introductory robotics course featuring a raspberry pi rover that students build then control with Python code. The course is thematic, focusing on robotics in space and life on mars. It blends computer science with engineering and robotics based practices so that students experience comprehensive interdisciplinary learning.

The series of lessons acts as foundational course for the robotics kit, and ultimately progress into more advanced robotics and computer science topics. Students will learn to program a robot, applying autonomous operation and machine learning concepts to fulfill real world tasks.

The lessons introduce students to basic function and algorithm design in conjunction with constructing readable programs. The course can be completed without the optional python principles mini lessons; however, for those new to Python concepts the course is enhanced with the mini lessons.

Lessons:



The build

Estimated time: 120-180 minutes

Description	This lesson focuses on building the pi-top rover model out of the pi-top robotics kit. They will use step by step written and visual instructions to complete the build. To finish out the lesson, students will test the rover by running a sample code program.
Objectives	**Students will learn:** • How to build a raspberry pi controlled controlled robot, combining hardware and software. • Ways to connect a device to a web browser using an IP address, or other digital address. • How to run and test a sample code program.
Guided Questions	How can hardware and software be combined to collect and exchange data? How is intelligent behavior conveyed through computers and robotics?
Outcomes	Students will build a raspberry pi controlled rover-like robot. They will use visual and written instructions to complete the design. Students will connect the pi-top to Further and test a sample code program. (Note: If the program is successful the rover was built properly.)
General Concepts	Engineering, robotics, rover, process, motion, encoder motors
CS Concepts	Hardware and software, IP address
CSTA Level 2 Standards	Design projects that combine hardware and software components to collect and exchange data. **2-CS-02**
	Systematically identify and fix problems with computing devices and their components. **2-CS-03**
CSTA Level 3 Standards	Identify and describe hardware. **CSTA.3B.CD2**
	Explain the notion of intelligent behavior through computer modeling and robotics. **CSTA.3B.CD5**
Other Standards	ISTE: 1c, 1d ISTE - Group Setting: 7b, 7c

Navigation

75 - 90 minutes

Description	The lesson focuses on learning and applying code to move the rover in both, forward and backward directions. Through investigation, students will realize how sleep commands can control the distance a robot travels. They will also investigate what happens when negative integers replace positive values in drive commands. Note: If teachers have electronics kits, then it is suggested that students complete the electronics kit extension at the end of the lesson.
Objectives	**Students will learn:** • To control a robot with Python text- based code to move in two directions (forward & reverse). • To use positive and negative integers to control direction. • How sleep commands are related to time and can control distance a robot travels.
Guided Questions	How is a rover's motion controlled through code? How can the use of negative integer values change the motion of the rover? Discuss the pros and cons of using sleep commands to control distance traveled.
Outcomes	Students will create and use code to move the rover forward and to reverse it. They will complete a specific forward and backward motion drill exercise using code to execute the task. They will develop a custom program with the goal of using the rover to push objects forward and backward in a space. Iterations are important because students will need to test and refine their code multiple times to create working programs.
General Concepts	Motion, forward, reverse, distance, displacement, metric system, integers, equations, modeling, ratios, mathematics, time, speed, integers, equations, metric measurements, trials, data collection, modeling
CS Concepts	Commands, sleep, methods
CSTA Level 2 Standards	Design projects that combine hardware and software components to collect and exchange data. **2-CS-02**
	Systematically identify and fix problems with computing devices and their components. **2-CS-03**
	Collect data using computational tools and transform the data to make it more useful and reliable. **2-DA-08**
	Refine computational models based on the data they have generated. **2-DA-09**

Continued next page...

CSTA Level 2 Standards	Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. **2-AP-13**
	Incorporate existing code, media, and libraries into original programs, and give attribution. **2-AP-16**
	Systematically test and refine programs using a range of test cases. **2-AP-17**
	Document programs in order to make them easier to follow, test, and debug. **2-AP-19**
CSTA Level 3 Standards	Compare levels of abstraction and interactions between application software, system software, and hardware layers. **3A-CS-02**
	Translate between different bit representations of real-world phenomena, such as characters, numbers, and images. **3A-DA-09**
	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects. **3A-AP-17**
	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs. **3A-AP-18**
	Explain the notion of intelligent behavior through computer modeling and robotics. **CSTA.3B.CD5**
Other standards	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 ISTE: 1c, 1d, 4a, 4c, 5a, 5b, 5c,

Optional Mini Lesson
Python Principles - Variables

Rover Race

150 - 180 minutes

Description	In this lesson, students will build upon using code to program motion and learn how to manipulate the rover's speed with a parameter. The lesson is themed around comparisons and science concepts relating to the Mars Rover, Perseverance. Note: If teachers have electronics kits, then it is suggested that students complete the electronics kit extension at the end of the lesson.
Objectives	**Students will learn:** • The basics of incorporating speed commands and value substitution with a variable. • To document programs in order to make them easier to follow, test, and debug. • To translate between different bit representations of real-world phenomena and code. • To manipulate outcomes, influenced by observing the cause & effect of changing values in code. • How computer science concepts - loops, comparisons, random, and range - apply to robotics. • To conduct trials, collect data, & create a scientific graph as a means to present the information. • To give a presentation or have a class discussion about how to vary the speed and why it is beneficial to do so.
Guided Questions	How can speed be declared without using a numerical value? What happens when you keep the speed factor the same but change the sleep time? Describe how to modify the code so that the robot reverses at similar speeds to a forward motion. The rover weighs 3.18 kilograms and Perseverance, the Mars rover, weighs about 45 kilograms. How do you think weight affects speed? Nasa's Mars Rover drives with a top speed of 0.016 kilometers. Calculate the speed of your rover and compare it to that of Perseverance's speed.
Outcomes	Students will learn to multiple ways to manipulate the rover's speed through code. They will start by using a value inside of a speed factor parameter, then transition to using a variable, and also apply the Python principles of random and range to declare the speed. Students will create a custom program in which the rover meets distance specifications traveling at different speeds. Inside of the share phase, they are given the opportunity to apply knowledge gained throughout the lesson by creating a custom challenge for a peer. The second task is to give a presentation or participate in a class discussion regarding how speeds can vary by the code used, and the benefits of using varying speeds. The thematic connection is linked to the Mars rover called Perseverance.
General Concepts	Motion, speed, distance, displacement, weight, metric system, integers, mathematics, calculations, equations, modeling, NASA, mars rover

Continued next page...

CS Concepts	Loops, importing, variables, sleep, built-in methods, random, range, booleans, comparisons, logical operators, characters (i), nesting
CSTA Level 2 Standards	Design projects that combine hardware and software components to collect and exchange data. **2-CS-02**
	Systematically identify and fix problems with computing devices and their components. **2-CS-03**
	Collect data using computational tools and transform the data to make it more useful and reliable. **2-DA-08**
	Refine computational models based on the data they have generated. 2-DA-09
	Create clearly named variables that represent different data types and perform operations on their values. **2-AP-11**
	Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. **2-AP-13**
	Create procedures with parameters to organize code and make it easier to reuse. **2-AP-14**
	Incorporate existing code, media, and libraries into original programs, and give attribution. 2-AP-16
	Document programs in order to make them easier to follow, test, and debug. **2-AP-19**
CSTA Level 3 Standards	Translate between different bit representations of real-world phenomena, such as characters, numbers, and images. **3A-DA-09**
	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects. **3A-AP-17**
	Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs. **3A-AP-23**
	Use data analysis to enhance understanding of complex natural and human systems. **CSTA.3B.CT5**
	Use models and simulations to help formulate, refine, and test scientific hypotheses. **CSTA.3B.CT8**
	Explain the notion of intelligent behavior through computer modeling and robotics. **CSTA.3B.CD5**
	Analyze data and identify patterns through modeling and simulation. **CSTA.3B.CT9**
	Analyze the beneficial and harmful effects of computing innovations. **CSTA.3B.CI2**
Other Standards	CS: CPP.L2-03, CPP.L2-09, CPP.L2-05 CC.MATH: 6.EE.B.6, 6.NS.C.8, REI.D.10 NGSS: MS-PS2-2, HS-PS2-1, HS-ETS1-2 ELA: RST.6-8.3 ISTE: 1c, 1d, 3d, 4c, 5b, 6c

Orbit

135 minutes

Description	In the previous lesson you learned how to control the robot's speed by giving it more power, or increasing the RPMs. In this lesson you will learn two different ways to make the robot turn left and turn right. Note: If teachers have electronics kits, then it is suggested that students complete the electronics kit extension at the end of the lesson.
Objectives	**Students will learn:** • How to turn the rover by coding the wheel motors to spin in opposite directions - clockwise and counter-clockwise. When motors are on opposite sides of the chassis, one is essentially backwards or flipped in comparison to the other. • How to turn the rover by programming the device Iso that power is given to one motor at a time. • To debug intentional errors written in a program to meet a specific outcome. • To create programs that will execute certain behaviors, such as moving in a complete circle and completing a square. • To program the rover so that it moves around an environment, carefully considering obstacles, walls, openings, etc.
Guided Questions	After investigating, describe how motor control affects how a robot turns. How can turn and drive commands be used to program the rover to make a "U" turn? What are rotations per minute (rpms) and how do they affect rover performance? What is the best way to program the rover to make turns? What should be addressed when creating a program in which the rover responds to and moves around the environment that it is in?
Outcomes	The lesson starts out with the basics of executing left and right turns. Students will learn to pre-form a turn when two motors move in reverse of each other. They will learn another method of turning, which is by giving power to only one motor. Some of the culminating tasks include: "Your rover's journey is being broadcast back home, create the code to make the rover do a spinning dance and impress the viewers! Let's pretend your rover needs to get around the space station, or navigate around it while exploring outside."
General Concepts	Motion, turning, rotations per minute, speed, integers, trials, data collection, distance, displacement, direction, metric system, mathematics, time, trials, modeling
CS Concepts	While true loops, for loops, rpms, methods, debugging, program refinement, commands, built-in methods, random and range, booleans & comparisons, importing

Continued next page...

CSTA Level 2 Standards	Design projects that combine hardware and software components to collect and exchange data. **2-CS-02**
	Systematically identify and fix problems with computing devices and their components. **2-CS-03**
	Collect data using computational tools and transform the data to make it more useful and reliable. **2-DA-08**
	Refine computational models based on the data they have generated. **2-DA-09**
	Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. **2-AP-13**
	Incorporate existing code, media, and libraries into original programs, and give attribution. **2-AP-16**
	Systematically test and refine programs using a range of test cases. **2-AP-17**
	Document programs in order to make them easier to follow, test, and debug. **2-AP-19**
CSTA Level 3 Standards	Translate between different bit representations of real-world phenomena, such as characters, numbers, and images. **3A-DA-09**
	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects. **3A-AP-17**
	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs. **3A-AP-18**
	Compare and contrast fundamental data structures and their uses. **3B-AP-12**
	Use models and simulations to help formulate, refine, and test scientific hypotheses. **CSTA.3B.CT8**
	Explain the notion of intelligent behavior through computer modeling and robotics. **CSTA.3B.CD5**
Other Standards	CS: CPP.L2-09, CPP.L2-05 CC.MATH: 6.NS.5, 7.RP.A.2 NGSS: MS-ETS1-2, MS-ETS1-4 CC.ELA: RST.6-8.3 ISTE: 1c, 1d, 3c, 5b

Optional Mini Lesson
Python Principles:

- The Import System
- Built-In Methods
- Random and Ranges
- Booleans & Comparisons

Maneuvering the Terrain

300 minutes

Description	Students will construct a maze (to meet given specifications) and program the rover to successfully complete the maze. The lesson includes an introduction focusing on forming flow chart diagrams to break down large tasks into manageable components and to plan code programs. It walks students through how to code the beginning segments of the maze, which they ultimately finish on their own. Students will also learn how to use functions and input commands in code programs. As the lesson is completed students will work through iterations and data collection to debug errors and refine the user experience.
Objectives	Students will learn: • To combine code sections and use prior knowledge gain throughout the course to form a program the rover will execute to successfully complete a maze. • To use flow charts and code diagrams to plan and create a successful program. • How to create functions for code reusability and program refinement. • How to modify a program to accept input commands to control the rover. • To document and break down large tasks into smaller components to meet goals. • To Illustrate ways computing systems implement input, and output through hardware components.
Guided Questions	How can flow chart diagrams help programmers write a program for complex tasks? What are ways that a complex program can be simplified using code or programming related practices? Discuss the impact of input modifications on the functionality of the programs you have written. Discuss intended and unintended implications when modifying the program to use input commands (e.g., breaking other functionality).
Outcomes	This lesson provides the opportunity for students to apply all of the knowledge gained thus far in the course and complete a maze challenge. It starts with an introduction to flowcharts and diagramming code programs to break down large tasks into smaller and simpler tasks. Students will write code to execute actions that complete segments of a maze. A section on functions is incorporated to teach students code reusability, which leads to creating a program to complete the maze. The aim is to navigate an environment with the fewest possible iterations. Students will further refine their programs to include input commands which control the rover. Theme connection: maneuvering around the rocky terrain of Mars.

Continued next page...

General Concepts	Maze design, program development, iterations, distance, displacement, integers, metric system, refining programs, user experience, keyboard inputs, time, speed, trials, data collecting, modeling
CS Concepts	Flow charts, program development, iteration, functions, keyboard input, if statements, branching
CSTA Level 2 Standards	Design projects that combine hardware and software components to collect and exchange data. **2-CS-02**
	Systematically identify and fix problems with computing devices and their components. **2-CS-03**
	Collect data using computational tools and transform the data to make it more useful and reliable. **2-DA-08**
	Refine computational models based on the data they have generated. **2-DA-09**
	Use flowcharts and/or pseudocode to address complex problems as algorithms. **2-AP-10**
	Incorporate existing code, media, and libraries into original programs, and give attribution. **2-AP-16**
	Document programs in order to make them easier to follow, test, and debug. 2-AP-19
CSTA Level 3 Standards	Compare levels of abstraction and interactions between application software, system software, and hardware layers. **3A-CS-02**
	Translate between different bit representations of real-world phenomena, such as characters, numbers, and images. **3A-DA-09**
	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects. **3A-AP-17**
	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs. 3A-AP-18
	Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs. **3A-AP-23**
	Illustrate ways computing systems implement logic, input, and output through hardware components. **3B-CS-02**
	Illustrate the flow of execution of a recursive algorithm. **3B-AP-13**
	Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution. **3B-AP-15**

Continued next page...

CSTA Level 3 Standards	Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality). **3B-AP-22**
	Evaluate key qualities of a program through a process such as a code review. **3B-AP-23**
	Discuss the impact of modifications on the functionality of application programs. **CSTA.3B.CD1**
	Explain the notion of intelligent behavior through computer modeling and robotics. **CSTA.3B.CD5**
	Use models and simulations to help formulate, refine, and test scientific hypotheses. **CSTA.3B.CT8**
Other Standards	CS: CPP.L2-09, CPP.L2-05, L1:6.CT.6, L1:6.CT.5, L1:6.CT.1, L1:6.CPP.5 CC.MATH: 6.SP.B.5b, 6.RP.A.3d, HSS-ID.A, HSG-GMD.B NGSS: MS-ETS1, HS-ETS1-2, SEP-5-HS6, SEP-5-HS4, SEP-1-HS4 CC.ELA: RST.6-8.3, RST.11-12.3 ISTE: 1a, 1c, 1d, 4a, 4b, 4c, 4d, 5b, 5c, 5d, 6c

Optional Mini Lesson
Python Principles:

- Flow Charts
- Functions
- Input
- If Statements
- Branching