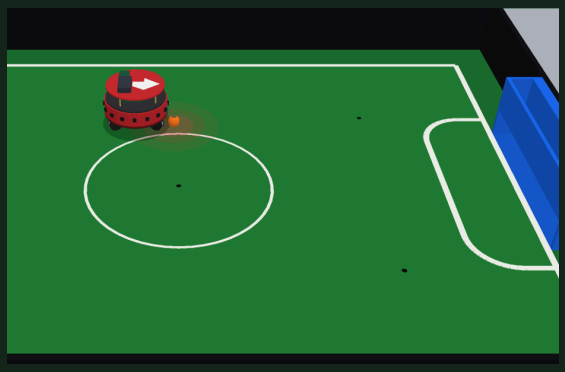
mytekDev

Robot Soccer Lab: Get Behind It and Stay In Bounds, Procedures with Parameters

Grades 6-8 · 45 minutes

Computer science



Slides, rubric and answer key | Print and go | No student accounts | Chromebook friendly

What's included

- A one-page overview: objectives, essential question, vocabulary, standards and assessment
- A 45-minute lesson sequence: warm-up, direct instruction, guided practice, independent practice, closure
- Slides for the board: a PowerPoint file that also opens in Google Slides
- A student worksheet with a QR code and tables for five offsets and five speeds
- A 3-question exit ticket, two to a page, with an answer key
- A rubric: 3 criteria, 4 levels, scored out of 12
- Differentiation for English learners, students with IEPs or 504 plans, and students who need more support or more challenge
- Cross-curricular extensions, and notes for teaching it in grades 4 and 5 and in high school
- Standards: CSTA 2017

Robot Soccer Lab: Get Behind It and Stay In Bounds, Procedures with Parameters

Grades 6-8 · 45 minutes · Computer science

Level 4 of Robot Soccer Lab wants three goals from three spots with no own goals. The starter program is card one from Level 1, written in Python as a procedure, go_round(offset), called as go_round(60). It swings wide and runs into the wall. Students change two lines so the offset grows with the ball's angle, never more than the parameter, and the robot scores all three. Then they test the parameter: 30, 45, 60, 90 and 120. Level 5 puts the balls just over the white line. A second procedure, chase(speed), drives straight at them and out of bounds, until students add the line sensors: when the robot sees white, it backs away the way it came.

At a glance

ESSENTIAL QUESTION

How does one procedure with a parameter make a robot easier to change and test?

OBJECTIVES

Students will be able to:

- explain what a procedure and its parameter do in a program
- change a procedure so its offset grows with the ball's angle
- test a parameter at several values and record each result
- use a sensor reading to back away from a line
- call the same procedure with different arguments for different jobs

VOCABULARY

Procedure, Parameter, Argument, Offset, Line sensor, Out of bounds

STANDARDS

2-AP-12, 2-AP-14, 2-AP-17

TIME AND MATERIALS

45 minutes, grades 6-8.

- One computer or Chromebook per student
- A projector
- The worksheet and exit ticket, one copy each per student
- The rubric, one copy per student if you grade the work

PREP

Make copies of the worksheet and exit ticket. Before class, open mytekdev.com/try/robotsoccerlab?lesson=get-behind-it-and-stay-in-bounds-procedures&ref=tpt-robotsoccerlab-2 once on a student computer, which checks that your school's filter allows the app, and follow

ASSESSMENT

- Formative: the questions under What to look for, asked while students work, and the exit ticket (answer key included).
- Summative: the worksheet, scored with the rubric.

TECHNOLOGY AND PRIVACY

Runs in any browser, Chromebooks included, with nothing to install. Students need no account and no email: the link works signed out, and nothing they do is stored on our server unless they choose to save.

the short guide yourself: about ten minutes.

Lesson sequence

MINUTES	STAGE	WHAT YOU DO	WHAT STUDENTS DO
0 to 5	**Warm-up** Hook	Show the title, essential question and Today I can slides (about 1 minute). Then the warm-up: the recipe with a number in brackets.	Answer the warm-up on the worksheet.
5 to 13	**Direct instruction (I do)** Model it	Project Level 4 and open the Python tab. Read go_round aloud: def, the parameter offset, the clock turned into an angle from -150 to 180, and the two moves that add or take away offset. Point at the last lines: go_round(60) is the call, 60 the argument. Press play on the Simulator tab: the robot runs into the wall. Show the card one slide, then the min slide. **It's working when:** Work two examples on the board: ball at 1 o'clock, angle 30, min(30, 60) is 30, the robot moves at 60. Ball at 3 o'clock, angle 90, min(90, 60) is 60, the robot moves at 150.	Fill in the vocabulary on the worksheet.
13 to 22	**Guided practice (We do)** Change the procedure	Students open the link or scan the QR code. A short guide runs Level 4 into the wall, then has them change the two moves in go_round. Walk the room until every student has three goals, then they test go_round(30), (45), (90) and (120). **Watch for:** Students who change the call to go_round(min(...)). The change goes inside the procedure, where angle is known.	Follow the guide. Fill in the offset table.
22 to 37	**Independent practice (You do)** Level 5: stay in bounds	Students press Next: Level 5. They change the loop to call chase(50) instead of go_round(60), and play: out at the wall. Then they add the line check before it: if robot.on_line(), move at robot.line_direction() + 180, speed 60, wait 0.2, else chase(50). Then test chase at 30, 60, 70 and 90. Each level starts in real time. Speed up and Slower, under the field, change the speed; the round play button starts the level again from the beginning, with the program as it is now. Walk the room with the What to look for questions. **If they ask:** "Replace the move and wait with robot.stop(). What happens, and why?" Stop or back away **Watch for:** Students who keep go_round at Level 5. The balls are over the line, so going round them takes the robot over it; it passes on some runs and goes out on others.	Write the line check, fill in the speed table, and answer the questions.
37 to 45	**Closure** Why a parameter? And exit ticket	Show the closure slide and take answers. Hand out the exit ticket.	Discuss, then answer the exit ticket.

Before you teach: a ten-minute walkthrough

Open mytekdev.com/try/robotsoccerlab?lesson=get-behind-it-and-stay-in-bounds-procedures&ref=tpt-robotsoccerlab-2. A short guide opens over Level 4, Get behind it, with the program in the Python tab. The field is in the middle: green carpet, white lines, the yellow goal (the robot's own) and the blue goal it attacks. The red robot is round, with an arrow on top for its front, and the orange ball glows so the robot can find it. The box on the field says the level's goal and, after each run, what happened and why. The readings beside it show the ball clock, the heading, the speed and how far away the ball is.

1. **The program.** Two procedures. go_round(offset) turns the ball clock into an angle from -150 to 180 (clock 7 to 11 come out negative, to the left), then moves at that angle plus offset, or minus it on the left. It is card one from Level 1. chase(speed) moves straight at the ball, card two from Level 2. The loop at the bottom calls go_round(60).
2. **The change.** angle + offset becomes angle + min(angle, offset), and angle - offset becomes angle - min(-angle, offset). The offset now grows with the ball's angle and never goes over the parameter: a ball at 1 o'clock gets 30, a ball at 2 o'clock or more gets 60.
3. **Level 5.** Next: Level 5 keeps the program. The starter robot already has line sensors. robot.on_line() is True when a sensor under the robot sees white, and robot.line_direction() says which way the white is, so line_direction() + 180 is the way back.
4. **Speed.** Each level starts in real time. Speed up and Slower, under the field, change the speed; the round play button starts the level again from the beginning, with the program as it is now. Each press of play takes the next seed for the receivers' small noise, so times can differ by a few hundredths of a second between runs.
5. **Numbers to expect.** Level 4: the starter goes out at the wall at 3.4 to 4.0 s. With the change: go_round(30) out at 1.7 s; 45 out at 2.7 s; 60 three goals in about 13.6 s; 90 three goals in about 15.0 s; 120 out at about 9.8 s. Level 5, chase(50) with no line check: out at 1.9 s, and at every other speed too. With the line check: chase(30) in about 11.6 s, 50 in about 9.9 s, 60 in about 8.4 s, 70 in about 8.9 s, 90 out. With robot.stop() instead of backing away, it sits on the line until time is up, or at 60 and 90 runs out.

> Stopping on the line looks safe and fails: the robot stops with a sensor on the white, so on_line() stays True and it never moves again, or at higher speeds it is already over before it stops. Backing away the way it came is the fix. A good question to ask.

Background for the teacher

A procedure (a function, in Python) gives a set of steps a name; a parameter lets each call give it a different value. Real robot programs are built this way, so one tested piece, get behind the ball, can be reused by a striker, a kicker and a goalie, and a fix goes in once.

The field's white line marks out of bounds. In the rules the app is modeled on, a robot that touches the wall or goes wholly into a penalty area sits out for a minute, so teams put line sensors under their robots and turn back when

they see white. The field and rules are modeled on the RoboCupJunior Soccer rules; the app is not affiliated with or endorsed by the RoboCup Federation.

Robot soccer has led to real companies: Raffaello D'Andrea, who coached Cornell's robot soccer team, went on to co-found Kiva Systems, whose warehouse robots Amazon bought in 2012. Robotics, mechanical, electrical and software engineers all work on robots like these.

What to look for

Ask these while students work. A student who has understood the lesson can answer them without help:

- In go_round(60), what is the parameter and what is the argument?
- The ball is at 1 o'clock. Which way does the robot move before the change, and after?
- Which arguments to go_round worked, and which did not?
- Why does stopping on the line not work?
- Why is line_direction() + 180 the way back?

> The second question is the one that matters. A student who says "before, 30 + 60 is 90, straight sideways; after, 30 + 30 is 60, so it moves less wide" can trace a procedure with its parameter. A student who says "it works now" has the result but not the idea.

Cross-curricular extensions

- Math: make a table of the ball's angle and min(angle, 60) for every hour on the clock, and graph it.
- Science: look up how a line sensor works: an infrared light and a sensor that reads how much comes back from the floor.
- Careers: find a robot that follows or avoids lines at work, such as a warehouse robot, and what it uses to see them.
- Computer science: find a procedure with a parameter in something you use, such as a volume control, and say what its argument is.

Vocabulary

Procedure	A named set of steps the program can run whenever it needs them. In Python it starts with def.
Parameter	A name in a procedure's first line, offset or speed, that gets a value each time the procedure runs.
Argument	The value given to a parameter when the procedure is called: the 60 in go_round(60).
Offset	The extra angle the robot aims past the ball, so it goes round it.
Line sensor	A sensor under the robot that sees the white line at the edge of the field.
Out of bounds	Touching the wall or going wholly into a penalty area. In a real match the robot sits out for a minute.

Differentiation

ENGLISH LEARNERS

- Keep the vocabulary slide up, and point at the robot, the ball or the box on the field as you say each word: the picture carries the meaning.
- Give a sentence frame on a card: go_round(____) aims at most ____ degrees past the ball.
- Label a printout of the program: procedure, parameter, call, argument, in two colors.

STUDENTS WITH IEPS OR 504 PLANS

- Pair students, one at the keyboard and one recording results.
- Give the two changed lines on a card, to copy exactly.
- Where the IEP allows a modified assignment, score the first two rubric rows only.

STUDENTS WHO NEED MORE SUPPORT

- Make the change to go_round together on the projector.
- Give the line check as four lines to type in, with the indents marked.
- The box on the field says why every run failed, out at the wall or wholly in a penalty area.

STUDENTS READY FOR MORE

- Write a third procedure, back_away(seconds), with the two line-check lines inside it, and call back_away(0.2). Try 0.5 and 1.
- Give chase a second parameter for the angle offset, so one procedure does both jobs.
- Open Level 6, Kick: add robot.kick() when robot.has_ball() is true, and call go_round(60) the rest of the time.

Adapting for other grades

Younger students: In grades 4 and 5, make the go_round change together on the board and let pairs test the five arguments; stop before Level 5. Score the first two rubric rows, total 8, and 6 meets the objectives.

Older students: In high school, start from an empty program and have students write go_round and chase themselves from the Level 1 and 2 cards, then a back_away(seconds) procedure. Level 9 then runs a striker and a goalie as two procedures on two robots.

Student worksheet

Name: _____ Date: _____

Open the app: scan the QR code, or go to mytekdev.com/try/robotsoccerlab?lesson=get-behind-it-and-stay-in-bounds-procedures

Essential question: How does one procedure with a parameter make a robot easier to change and test?

Warm-up: What does the number in "add sugar(2 spoons)" let you do? _____

1. **Vocabulary.** Procedure, parameter, argument, offset, line sensor, out of bounds: write each in your own words.

2. **Level 4 offsets.** Changed go_round, then go_round(____): 30: _____. 45: _____. 60: _____. 90: _____. 120: _____.

3. **Level 5 speeds.** With the line check, chase(____): 30: _____. 50: _____. 60: _____. 70: _____. 90: _____.

4. **Stop instead.** With robot.stop() instead of backing away: _____

1. Why does go_round need min to score from every spot?

2. Why is a parameter better than writing the steps again for each number?

Exit ticket

Name: _____

1. In chase(50), 50 is: (A) the parameter (B) the argument (C) the procedure's name

2. min(30, 60) is: (A) 30 (B) 60 (C) 90

3. A friend's robot stops when it sees the white line, and then never moves again. Use what you learned today to say what to change, and why.

Robot Soccer Lab: Get Behind It and Stay In Bounds, Procedures with Parameters

Exit ticket

Name: _____

1. In chase(50), 50 is: (A) the parameter (B) the argument (C) the procedure's name

2. min(30, 60) is: (A) 30 (B) 60 (C) 90

3. A friend's robot stops when it sees the white line, and then never moves again. Use what you learned today to say what to change, and why.

Answer key: exit ticket

QUESTION	ANSWER	WHY
1	B	The argument is the value given in the call; speed is the parameter it fills.
2	A	min gives the smaller number.
3	Open answer	Back away the way it came, line_direction() + 180, for a moment: stopped on the line, the sensor still sees white, so the program keeps stopping.

Worksheet: what students should find

BLANK	WHAT THE APP SHOWS
Level 4 offsets	30: out at the wall. 45: out at the wall. 60: three goals, about 13.6 s. 90: three goals, about 15 s. 120: out.
Level 5 speeds	30: passes, about 11.6 s. 50: passes, about 9.9 s. 60: passes, about 8.4 s. 70: passes, about 8.9 s. 90: out.
Stop instead	It sits on the line until time is up (or at 60 and faster, goes out).
Question 1	A fixed 60 sends the robot wide when the ball is nearly in front; the offset should be no bigger than the ball's angle.
Question 2	One copy of the steps works for every number, and a fix goes in once.

> Each run uses the next seed for the sensors' small noise, so times can differ a little between runs; which tests pass and fail is the same on any computer.

Rubric

Score each row from 1 to 4.

Name: _____

CRITERIA	4 EXCEEDS	3 MEETS	2 APPROACHING	1 BEGINNING
Changing the procedure	Makes the change, scores three goals, and explains min with two worked examples.	Makes the change and scores three goals.	Changes one line, or changes the call instead.	No change.
Testing the parameter	Tests five arguments and five speeds, records each, and states the range that works.	Tests five arguments and records each.	Tests one or two.	Does not test.
Staying in bounds	Writes the line check, passes Level 5, and explains why stopping is not enough.	Writes the line check and passes Level 5.	Writes a line check that stops but does not pass.	No line check.

Total: ____ / 12. A score of 9 or more meets the objectives.

Standards

The lesson is built for CSTA 2-AP-14: students change a procedure with a parameter, then call it and a second procedure with different arguments. Testing those arguments and speeds is 2-AP-17, and the loop with a sensor condition and an else is 2-AP-12.

FRAMEWORK	CODE	GRADES	STANDARD	HOW THE LESSON MEETS IT
CSTA 2017	**2-AP-12**	6-8	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.	Students add an if and else inside the forever loop that reads the line sensor, with a wait inside the if.
CSTA 2017	**2-AP-14**	6-8	Create procedures with parameters to organize code and make it easier to reuse.	Students change go_round(offset), call it with five arguments, and call a second procedure, chase(speed), for Level 5.
CSTA 2017	**2-AP-17**	6-8	Systematically test and refine programs using a range of test cases.	Students test go_round at five arguments and chase at five speeds, record each result, and find the range that works.

Checked November 2026 against: CSTA K-12 Computer Science Standards, Revised 2017.

Optional: save student work in a free class

The whole lesson runs at the link on the worksheet, with no accounts and no sign-up.

To keep students' work, and see it all in one place, make a free class. It takes about ten minutes, once, and covers every lesson in this series: mytekdev.com/tools/class-setup

Students in a class open mytekdev.com/tools/robot-soccer-lab/lessons/get-behind-it-and-stay-in-bounds-procedures/start instead of the worksheet link. Each one gets their own copy of the starter, and the same link reopens it next time.

Terms of use

Thank you for downloading this lesson. The plan, the handouts and the art were made by mytekDev.

You may:

- print and copy it for your students
- change it for your own classes
- post it on your class website or learning management system
- share this file with other teachers, whole and unchanged, with this page included

You may not:

- sell it, or put it in a product you sell
- share a changed copy
- claim it as your own

The app this lesson uses is free and runs in the browser. Students need no account. It is at mytekdev.com.

Questions, or found a mistake? Ask on the Teachers Pay Teachers Q&A for this resource, or write to us at mytekdev.com/contact.

© 2026 mytekDev. All rights reserved.