Family edition

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

Ages 11-14 · about 45 minutes, in one sitting

In Robot Soccer Lab your child changes a procedure in Python so a soccer robot scores from anywhere, then teaches it to stay inside the white line.

What you need

- A computer or Chromebook with a current browser.
- The link: mytekdev.com/try/robotsoccerlab?lesson=get-behind-it-and-stay-in-bounds-procedures&ref=fam-robotsoccerlab-2. No account, no email and nothing to install.

Before you start: try it once yourself

Open mytekdev.com/try/robotsoccerlab?lesson=get-behind-it-and-stay-in-bounds-procedures&ref=fam-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.

The ideas, in plain words

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.

Do it together

1. Open mytekdev.com/try/robotsoccerlab?lesson=get-behind-it-and-stay-in-bounds-procedures&ref=fam-robotsoccerlab-2 and press play.

   > **You'll know it worked when:** The robot swings wide round the ball and runs into the wall: out of bounds, in about 4 seconds.

2. Open `Python`. In go_round, change `angle + offset` to `angle + min(angle, offset)` and `angle - offset` to `angle - min(-angle, offset)`. Go back to `Simulator` and press play.

   > **If you get stuck:** Keep the spaces at the start of each line exactly as they were.
   >
   > **You'll know it worked when:** Three goals in about 13.6 seconds.

3. Change `go_round(60)` at the bottom to 30, then 90, then 120. Play each.

   > **You'll know it worked when:** 30 runs into the wall early, 90 scores all three in about 15 seconds, 120 goes out.

4. Put 60 back and press `Next: Level 5`. Change `go_round(60)` to `chase(50)` and press play.

   > **You'll know it worked when:** The robot drives straight at the first ball and over the line into the wall.

5. Above `chase(50)` add: `if robot.on_line():`, then indented `robot.move(robot.line_direction() + 180, 60)` and `robot.wait(0.2)`, then `else:` with `chase(50)` indented under it. Play.

> **If you get stuck:** Each line inside the if and the else starts 4 spaces further in.
>
> **You'll know it worked when:** The robot backs away each time it sees the line, and gets close to all three balls in about 10 seconds.

Talk about it

Ask these while you work together. A child who has understood it 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 child 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 child who says "it works now" has the result but not the idea.

For younger and older kids

Younger: For a younger child, stop after the three goals on Level 4, and try the different numbers in go_round.

Older: For an older child, write a third procedure, back_away(seconds), and try it with 0.2, 0.5 and 1.

Keep going

- 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.

Bring it into other subjects

- 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.

For your records

Ready to copy into an hour log, a portfolio or a transcript.

LESSON	Robot Soccer Lab: Get Behind It and Stay In Bounds, Procedures with Parameters
SUBJECTS	Computer science
TIME	45 minutes
WHAT WE DID	Changed a Python procedure so a simulated soccer robot gets behind the ball from anywhere, tested its parameter at five values, and used line sensors to keep the robot in bounds.
SKILLS PRACTICED	Procedures with parameters; Testing a program; Using sensors in conditions

Activity sheet (optional)

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?

Quick check

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.

Answers

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.

If your state or umbrella school asks, this lesson covers CSTA 2017 2-AP-12, CSTA 2017 2-AP-14, CSTA 2017 2-AP-17.

Terms of use

This lesson was made by mytekDev. You may print and copy it for the children you teach at home, and change it for them. You may not sell it, share it with other families, or post it publicly.

The app this lesson uses is free and runs in the browser, with nothing to install and no account needed. It is at mytekdev.com.

© 2026 mytekDev. All rights reserved.