Unit 5: Rover's Sensors — Skill Builder 2: Off the Wall

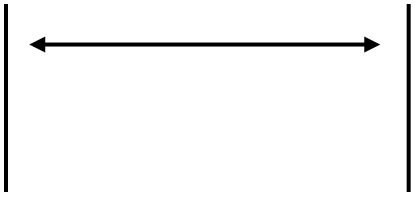
In this lesson, you will use the TI-Innovator Rover's Ultrasonic Ranger to avoid an obstacle.

Objectives:

- Read the distance to an object in front of Rover
- If the distance is 'too small', turn around and continue driving

Teacher Tip: In this project, students will encounter a fairly common issue when working with Rover: The TI-Nspire CX II works much faster than the Rover. In the LED lessons of the last unit, the TI-Nspire CX II had to wait until Rover finished a forward or a turn command before issuing the next instruction in the code.
The same problem arises here, but the effect (error) is more dramatic. The Rover struggles between moving, turning, and stopping. Adding another wait (in the right place) solves the problem.

As Rover is driving, your program can use the Ultrasonic Ranger to detect an obstacle ahead and interrupt Rover to take another route. In this project, the Rover will drive back and forth between two opposite walls.



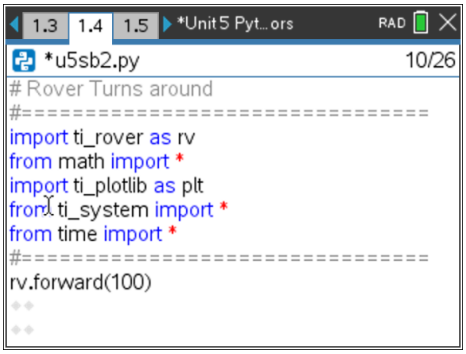
Teacher Tip: This project works well using hardcover books standing on end. Be sure the books are perpendicular to the Rover both vertically and horizontally. Otherwise, the Ranger will not 'see' the obstacle because the (silent) tone does not bounce (echo) directly back to the sensor.

1. Begin a new Python program using the Rover Coding template.

   Get Rover moving using:

   rv.forward (100)

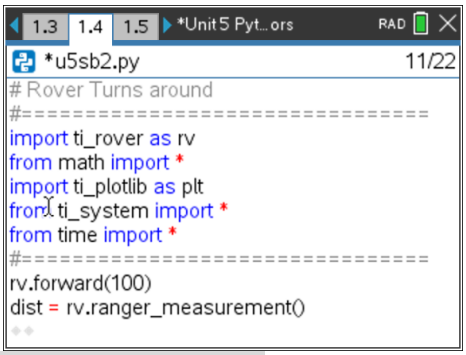
   Yes, that's 10 meters, or about 39 feet, but there will be an obstacle in the way.



2. Read the distance in front of Rover:

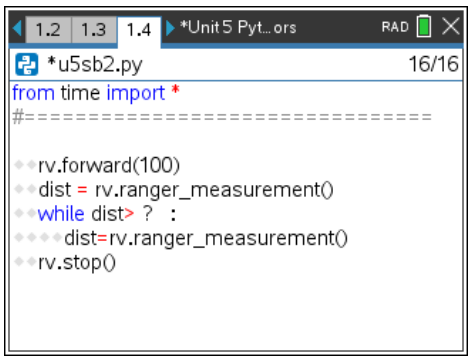
   dist = rv.ranger_measurement()

   As long as there's room to proceed, keep going. But if Rover is too close to an obstacle, turn around. How close is 'too close'? That is for you to determine.



Teacher Tip: Rover needs about 20 cm to turn around.

3. Add a **while** loop to continue monitoring the distance as long as the obstacle is 'far away':



while dist > ? :

(Replace the question mark (?) with a 'small' number.)

dist = rv.ranger_measurement()

For now, just stop Rover when the loop ends.

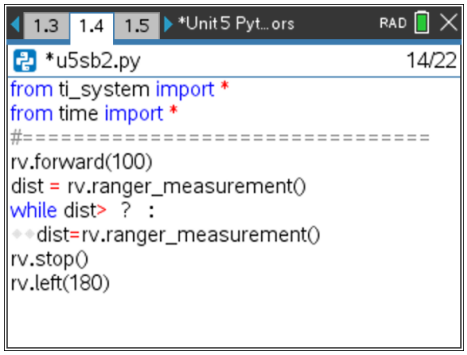
rv.stop()

rv.stop() is found on **menu > TI Rover > Drive**.

Test your program by pointing Rover towards a 'wall' (less than 40 feet away) and run the program. Rover should stop before hitting the 'wall'.

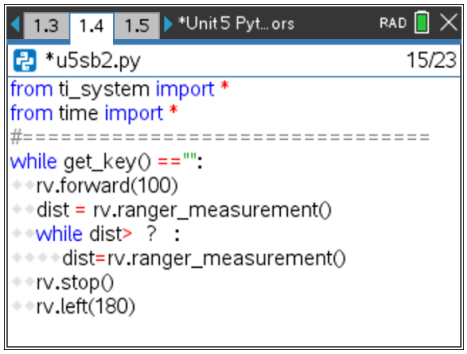
> **Teacher Tip:** Your hand works well as an obstacle, too.
> **rv.stop()** *immediately* stops Rover from driving and clears all commands in the drive queue.
>
> The principle above: Get Rover moving, and then 'monitor' the distance. The drive command is queued, but the **measurement()** is not.

4. When Rover encounters the obstacle, the Rover should turn around. After Rover stops, have the Rover turn around using:



rv.left(180) or **rv.right(180)**

5. After turning around, go in the opposite direction (**forward**) until a key is pressed. Remember that Rover is moving, so it might be tricky to press the **esc** key.



Place the statement

while get_key() != "esc":

at the beginning of your code.

Edit the statement so that *any key* can be used to stop the program:

while get_key() == "":

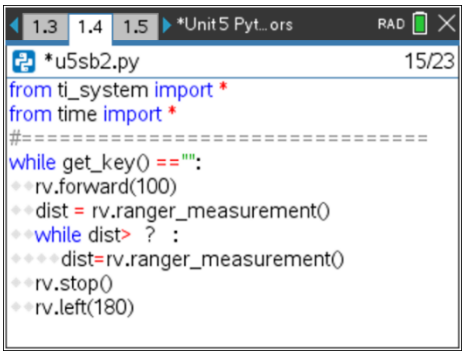
(== stands for "equals", "" stands for 'nothing')

Indent all the statements below this **while** statement to make them the **while block**. The easiest way to indent a group of statements is to select all those statements with **shift+down arrow**, and then press the **tab** key.

Teacher Tip: This is the point where the code will not work properly. When the **left(180)** command is processed, control immediately goes back to the top of the loop and Rover is still too close to the wall and will not actually move because the program runs faster than Rover can follow the instructions. You will hear the motors struggling. This is a good example of a runtime error. All the code is correct as far as the computer is concerned, but the physical effect is undesirable (wrong).

6. Run the program now. Rover approaches the wall, stops, and turns around (if there's enough room). Wait - something is wrong. Rover is stuck!

   Turning around takes Rover a second or two, but your program proceeds back to the top of the loop immediately, tells Rover to go forward, then detects the wall again and stops. Rover turns, goes forward, and stops. Rover is overwhelmed because the commands are all coming in almost at once. You will hear the motors struggle, but Rover does not move. Everything is happening too fast and Rover will have a nervous breakdown if something is not done… and soon!

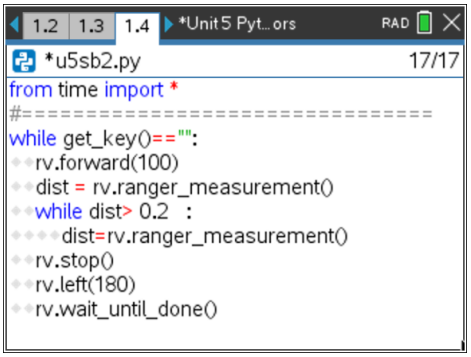


7. Press any key to terminate the program.
   Rover might still struggle for a while. If so, just unplug Rover from the handheld, turn the power switch off, and then reconnect and turn on again.

   To fix the issue in your program, make the TI-Nspire CX II *wait until* Rover has *finished* turning around by adding the statement:

   rv.wait_until_done()

   Place this statement just below the **rv.left(180)** statement.

   Run the program again and Rover will happily travel between two opposite walls.

   Rover needs about **20 cm** to turn around. The Rover's aft is longer than the bow. **while dist > 0.2:** handles this well.