Unit 4: Driving Features | **Skill Builder 1: Make It Move!**

In this lesson, you will begin to operate the TI-Innovator Rover robotic vehicle by making 'Rover' move and turn.

Objectives:

- Make Rover move
- Make Rover drive in a pattern

Python's TI-Innovator **Rover** commands all begin with **rv.** This syntax is a result of the way that the **ti_rover** module is imported into your programs (**import ti_rover as rv**). The same is true of the **ti_plotlib** module. This technique is called 'aliasing' the module name (replacing it with a *shorter* name).

When you write a program that operates Rover, you are controlling the vehicle through easy-to-use commands but there's a lot going on 'behind the scenes' just like other TI-Innovator™ Hub commands.

Be sure your TI-84 Plus CE Python is connected to Rover and that Rover's power is turned on.

Teacher Tip: New to Python? Check out 'aliasing module Python' online.

1. Begin a **<New>** Python program and select the **Rover** template from the **<Type>** menu when entering the name of the program. Ours is ROVER1. (turn [alpha-lock] off to type the digit 1 in the program name). Note the last import statement: **import ti_rover as rv**. This syntax means that all functions selected from this module will begin with the prefix '**rv.**'

Teacher Tip: Remember that it is also possible to simply type statements into the Python editor but when controlling the Rover, **rv.** is still required and is a possible source of errors.

2. To make Rover go forward use **<math> ti_rover…** to get this statement from the Drive menu shown:

   rv.forward(distance)

   Notice that the menu does not include **rv.** in front of **forward** but **rv.** is pasted into your program. This is done to save space on the menu.

   The distance argument is measured in 'grid units' and you will be able to determine what that means in a moment. Use the number **1** for the argument and run the program. If Rover moves… hooray!

   But… just how *far* did Rover move? Measure the distance that Rover moved.

Teacher Tip: Rover's default unit is 10 cm or 1 decimeter or 0.1 m.
Distance units are in meters and angle units are in degrees.
If you *delete* the distance prompt, Rover moves 1 meter.
If Rover does not travel 10cm then, to set the unit, add this statement to the code before moving forward: **rv.grid_m_unit(0.1)** which is found on **[math] ti_rover… Commands**

3. Try the four basic drive commands: **forward**, **backward**, **left**, and **right**.

4. To drive a square:
   Erase the drive commands you entered (or make them comments using **[2nd] [3]** for the # symbol as shown).
   Use a **for** loop to make Rover drive in a square pattern.
   Begin with:

   for i in range(size):

   ♦♦

   which is found on **<Fns…> Ctl**

   Use an appropriate size (the number of sides of a square) to make a square and add just two statements in the (indented) loop block. Try it yourself and run the program.

5. Did you write something like this?

   ♦♦**rv.forward(1)**
   ♦♦**rv.right(90)**

   If Rover moves in a square pattern… Congratulations!

   Can you make a larger square? A different shape? Try adding a marker to Rover to draw the patterns.

Teacher Tip: There are other drive commands further down the **Drive** menu that allow further options. These options offer many customized drive statements related to distance, time, and rate using units such as meters, grid units, wheel revolutions (for forward and backward), and degrees, radians or gradians (for angles). These options are not covered in this course but make interesting exploration opportunities. See the online e-guide for Python Rover coding for more information.