Unit 4: Make Rover Move! — Activity 1: Your First Rover Program

In this first lesson for Unit 4, you will learn about working in the Python Editor to write a program that makes the TI-Innovator Rover move.

Objectives:

- Use the TI-84 Plus CE Python Editor
- Access the Rover (**rv**) submenus
- Control the TI-Innovator™ Rover through the TI-Innovator™ Hub
- Make the TI-Innovator™ Rover move **forward**, **backward**, **left**, and **right**

Getting Started

- Insert **import ti_rover as rv** from the **Types** submenu at the beginning of any program that utilizes the TI-Innovator Rover.
- The Rover commands are found by using the softkeys at the bottom of the screen.
- Some portion of the final instructions, such as numeric values and optional parameters, are entered as normal keypad characters or selected from another Rover menu.
- Most Rover commands leave the cursor inside the parentheses. This indicates that that there are more options to enter within the command. The TI-84 Plus CE requires that parentheses come in pairs.

Teacher Tip: Remember that you are programming the handheld to program the TI-Innovator Hub to *control* the TI-Innovator Rover. You are not programming the Rover directly.

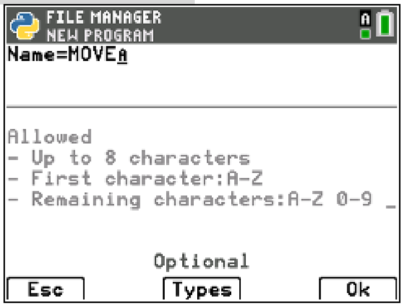
Your first Rover command imports all TI-Innovator commands and opens communication between the handheld, TI-Innovator Hub, and TI-Innovator Rover. **rv** is the name of the Rover device.

To create this statement:

1. Press the **Zoom** key to choose the Program Types
2. Then, select the **7: Rover**

This imports the following modules

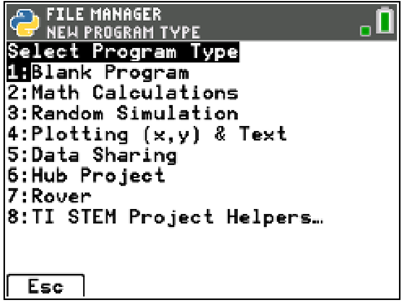
- from time import*
- from ti_system import*
- import ti_rover as rv*



```
FILE MANAGER
NEW PROGRAM
Name=MOVEA

Allowed
- Up to 8 characters
- First character:A-Z
- Remaining characters:A-Z 0-9

Optional
```



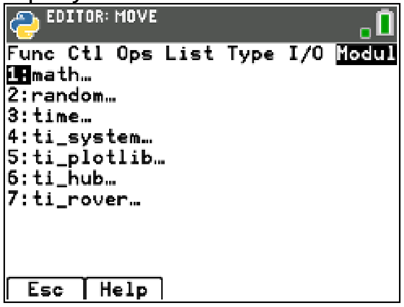
```
FILE MANAGER
NEW PROGRAM TYPE
Select Program Type
1:Blank Program
2:Math Calculations
3:Random Simulation
4:Plotting (x,y) & Text
5:Data Sharing
6:Hub Project
7:Rover
8:TI STEM Project Helpers…
```

Teacher Tip: The **import ti_rover as rv** command is required in all programs that utilize the TI-Innovator Rover. It connects the Rover to the TI-Innovator Hub just like other external devices. It also initializes the Rover's state (position and direction) and clears other Rover related data, such as distance travelled and headings.

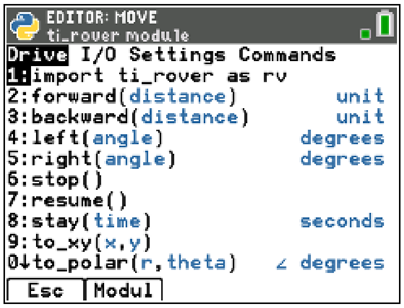
When running a program, if you hear a beep from the TI-Innovator Hub before (or while) you execute the program it means there is a connection error. Make sure the Rover is turned on and all cables are properly connected.

Driving the Rover

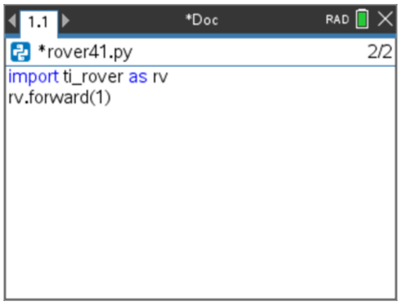
3. Press **Fns… > Modul > ti_rover** and then select the **Drive** menu as shown to the right.



4. Select **forward(*distance*)** from the **Drive** menu.



Notice that the **rv.forward()** command that is pasted into the program has the insertion cursor inside the parentheses with helper text of distance and pop-up text of *grid units*. This helps the programmer know what parameters the command requires.



```
import ti_rover as rv
rv.forward(1)
```

5. We add the number **1** here:

   rv.forward(1)

6. In the Python Editor, press **trace** softkey to run the program. This command performs the 'Check Syntax & Save' operation, switches to a Python Shell application, and runs the program. Be sure that there is about one foot of free space in front of the Rover.

The Rover should move forward. But how far? Study the movement carefully and determine what **rv.forward(1)** means.

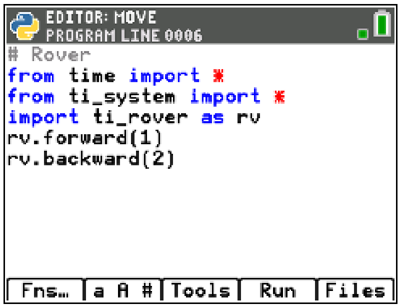
The Shell application displays **>>>** when the program ends. Notice that the program actually ends before the Rover finishes moving. The handheld and the TI-Innovator Hub work at different rates.

Teacher Tip: rv.forward(1) moves the Rover forward 10 cm. This unit value can be customized.

Driving Backwards

7. Edit the program, and add the statement **rv.backward(1)** below the **rv.forward()** command by pressing **Fns… > Modul> ti_rover** > **backward**.

8. Add the number **1** in the parentheses.



```
# Rover
from time import *
from ti_system import *
import ti_rover as rv
rv.forward(1)
rv.backward(2)
```

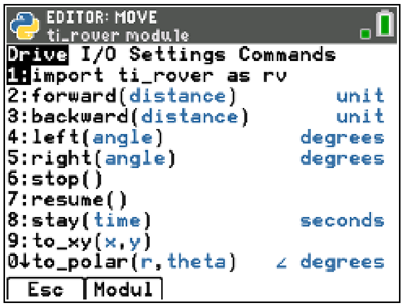
14. Run the program again (**trace**).

This time, the Rover should move forward a bit and then back to its original position. If it does, congratulations! You made the Rover move.

Teacher Tip: Syntax errors are caught by the 'Check Syntax and Save' process. If a program generates a runtime error, then an error message will display in the Shell application identifying a line in the program. This might or might not be the actual place that caused the error.
If there's an error in the TI-Innovator Hub code, then the TI-Innovator Hub will beep unexpectedly. (The color LED also blinks but it's hidden from view when installed in the Rover.)
The third error that can occur is with the Rover itself; incorrect instructions will result in unexpected behavior. Be careful that the Rover does not drive off the table.

Turning

The next two commands in the **Drive** menu are **rv.left()** and **rv.right()**.



```
1:import ti_rover as rv
2:forward(distance)          unit
3:backward(distance)         unit
4:left(angle)             degrees
5:right(angle)            degrees
6:stop()
7:resume()
8:stay(time)              seconds
9:to_xy(x,y)
0↓to_polar(r,theta)    ∠ degrees
```

15. Add these two commands to your program. Notice the in-line prompt requesting the number of degrees. Type 90 and run the program again.
    rv.left(90)
    rv.right(90)

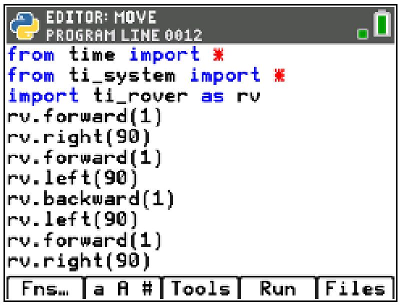
What do these instructions do?

Teacher Tip: rv.left(90) and **rv.right(90)** cause the TI Rover to turn 90 degrees left or right, respectively, from its current heading. Note that distance and turning angles are heavily dependent on the type of surface on which the Rover is travelling. The center of the turn angle is halfway between the bases of the wheels.

Two consecutive **rv.forward(1)** commands will cause rover to move, pause, and then move again.

Making it Travel

Study the program to the right, and predict what the Rover will do and where it will end up when the program ends.



```
from time import *
from ti_system import *
import ti_rover as rv
rv.forward(1)
rv.right(90)
rv.forward(1)
rv.left(90)
rv.backward(1)
rv.left(90)
rv.forward(1)
rv.right(90)
```

16. Enter these commands into your handheld, and run the program.

Did your program do what you expected? Can you make a program *with only these commands* that causes the Rover to make a *rectangular* pattern?