

Unit 6: Coordinates with Rover

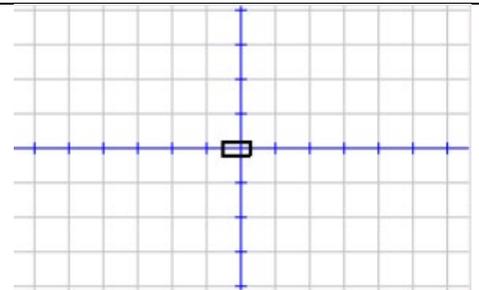
Skill Builder 1: Drive and Plot

In this lesson, you will learn about the Rover’s coordinate system and how to drive to a particular point represented by an ordered pair of numbers.

Objectives:

- Understand the Rover’s coordinate system and initial position and heading
- Make the Rover move to a certain point on the coordinate plane
- Plot Rover’s points on the TI-Nspire CX II screen

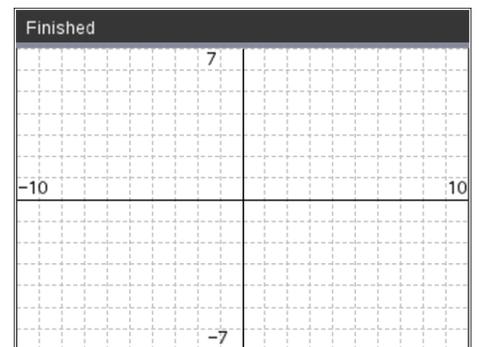
The Rover has a ‘built-in’ coordinate system just like a Cartesian graphing system. When you **import ti_rover as rv**, the Rover’s position on the coordinate grid is set to (0,0) and its heading is 0 degrees (pointing toward the positive x-axis, or ‘east’ on a map).



Teacher Tip: Rover also has a polar coordinate system!

In addition to Rover’s coordinate grid, we can use the on-screen coordinate system that is available from the **ti_plotlib** module which is also imported into the Rover Coding template for just this reason. If you use just these three statements from **menu > TI PlotLib > Setup**, you will get the screen to the right when you run:

```
plt.window(-10,10,-7,7)
plt.grid(1,1,"dashed")
plt.axes("on")
```



As we tell Rover to move around to points on the floor, we will also be plotting those points on the screen.

Teacher Tip: The Rover’s coordinate unit is the same as the driving unit: 10cm. The grid unit can be changed using **rv.grid_m_unit(scale_value)**. The default **scale_value** is 0.1 m/unit (10cm). To make the grid unit one inch use the **scale_value** 0.0254.

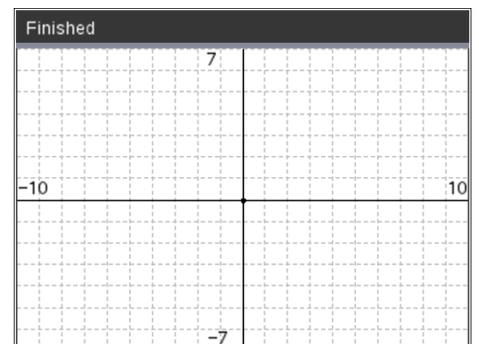
1. The first point to plot is Rover’s home position: (0, 0) using the statement

```
plt.plot(x,y,"mark")
```

found on **menu > TI PlotLib > Draw**.

Change the point (x, y) to (0, 0) and then select a “mark” to represent Rover from the provided list. If you run the program now, you will see a mark (we chose the dot ‘o’) at the origin in the center of the screen.

Now, whenever Rover drives to a point on the floor, also plot the point on the screen.





10 Minutes of Code - Python

TI-NSPIRE™ CX II WITH THE TI-INNOVATOR™ ROVER

UNIT 6: SKILL BUILDER 1

TEACHER NOTES

- Depending on the space that you have, make Rover drive to a point in each of the four quadrants using `rv.to_xy()` found on **menu > TI Rover > Drive**.

```
rv.to_xy(1,1)
rv.to_xy(-1,1)
rv.to_xy(-1,-1)
rv.to_xy(1,-1)
```

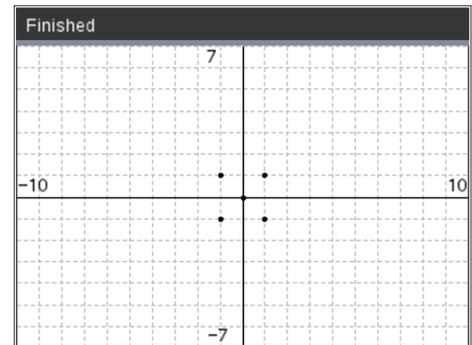
You do not have to use 1s and you don't even have to use the same numbers. But you do have to make sure that Rover visits all four quadrants.

Try your program now. Notice that Rover turns *directly* toward the position of the next point before moving there.

```
*u6sb1.py
#-----
# all four quadrants...
rv.to_xy(1,1)
rv.to_xy(-1,1)
rv.to_xy(-1,-1)
rv.to_xy(1,-1)
```

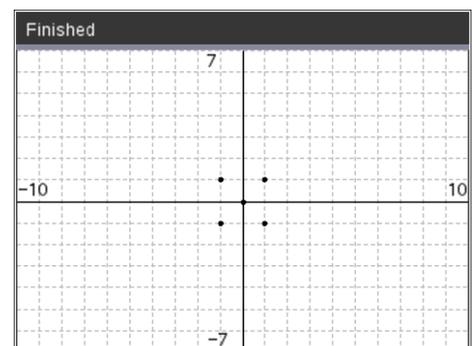
Teacher Tip: Rover might 'wiggle' a little when lining up for the next movement. This is Rover's gyroscope fine-tuning the heading before driving.

- Add `plt.plot(x,y,"mark")` statements to your program to plot the points on the screen *after* Rover reaches them. Try it now. Did the program perform as you expected?



Teacher Tip: Plotting on the screen is fast and easy using the `tiplotlib` tools.

- Answer: No! The points on the screen are plotted almost immediately when the program runs, and it takes Rover some time to drive to all four points. How do you sync the plotting with the driving?





10 Minutes of Code - Python

TI-NSPIRE™ CX II WITH THE TI-INNOVATOR™ ROVER

UNIT 6: SKILL BUILDER 1

TEACHER NOTES

5. Use the Rover function `rv.wait_until_done()` found on **menu > TI Rover > Commands** to pause the *program* while Rover is moving. You will need one of these functions for each driving point and order does matter. Try it now. Where will you place those *wait* functions?

```

1.2 1.3 1.4 *Unit 6 Py... rds RAD 12/28
*u6sb1.py
# all four quadrants...
rv.to_xy(1,1)
plt.plot(1,1,"o")
|
rv.to_xy(-1,1)
plt.plot(-1,1,"o")
|
rv.to_xy(-1,-1)
plt.plot(-1,-1,"o")
|
rv.to_xy(1,-1)
plt.plot(1,-1,"o")

```

Teacher Tip: The `wait_until_done()` functions belong after the `rv_to_xy()` and before the `plt.plot()` for each point:

```

1.2 1.3 1.4 *Unit 6 Py... rds RAD 20/32
*u6sb1.py
# all four quadrants...
rv.to_xy(1,1)
rv.wait_until_done()
plt.plot(1,1,"o")
|
rv.to_xy(-1,1)
rv.wait_until_done()
plt.plot(-1,1,"o")
|
rv.to_xy(-1,-1)
rv.wait_until_done()
plt.plot(-1,-1,"o")

```

6. At the end of your program, send Rover to the 'home' position, (0, 0). Use the statement

`rv.to_angle(0, "degrees")`

found on **menu > TI Rover > Drive** to have Rover point in the original direction ('east'). It's nice to put the toys away, right?

```

1.2 1.3 1.4 *Unit 6 Py... rds RAD 45/45
*u6sb1.py
#back to where she started ('home'):
rv.to_xy(0,0)
rv.to_angle(0,"degrees")

```