Unit 5: Rover's Sensors | **Skill Builder 1: Introducing Ranger**

In this lesson, you will learn to read the distance from the front of the TI-Innovator Rover to an obstacle. This lesson monitors that distance and displays information in two different ways. Another sensor is on the bottom of Rover and it can 'see' colors.

Objectives:

- Read Rover's Ranger measurement
- Display it on the screen using **print()**
- Display it on the screen using **text_at()**
- Determine the unit of measure

The two small cylinders on the front of Rover are not headlights. They are an Ultrasonic Ranger. A silent (to us) tone is sent out by one of the two sensors and the other one 'hears' the echo when the sound bounces off an obstacle. The internal software then calculates the distance to the obstacle using the speed of the sound and the time that it takes for the echo to return to the sensor: **D = V * T**

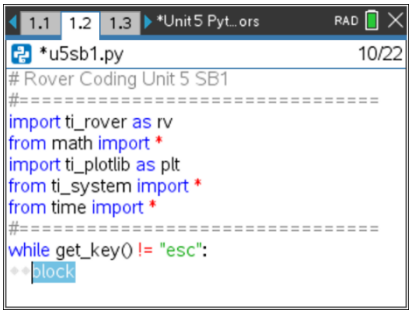
All that happens in the blink of an eye!

Teacher Tip: Sometimes referred to as a 'motion detector', an Ultrasonic Ranger is more of a 'rangefinder' that determines distance, not motion.

1. Begin a new Python Rover Coding project. For this lesson, use the convenient 'press **esc** to exit' loop:

```
while get_key() != "esc":
    block
```

You will find this command on several menus, but since you are using **Rover** commands, look in **menu > TI Rover > Commands**.



2. Add three statements in the **while** block to

   a) read the distance in front of Rover:

```
dist = rv.ranger_measurement()
```

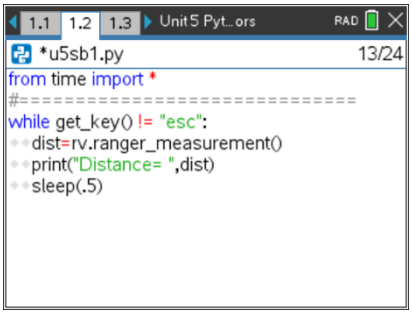
   (Find this on **menu > TI Rover > Inputs**.)

   b) print it on the screen:

```
print("Distance= ",dist)
```

   c) wait before reading the next distance:

```
sleep(.5)
```

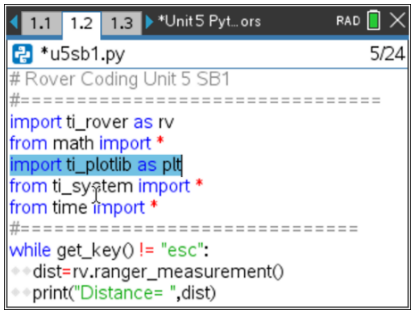


Run the program and move your hand toward and away from the front of Rover. Point Rover at a wall, the ceiling, and the floor. Watch the values that are displayed. Note: Rover does not move…yet.

Can you determine what distance unit (feet, meters, etc.) is being used?

Teacher Tip: The Ranger's distance unit is meters.

3. Rather than have the printed numbers scroll down the screen, improve the display of the distance values by using the **text_at()** function found in the **ti_plotlib** module. If you worked through the previous TI-Innovator™ Hub lessons (Units 1, 2, and 3), you have used that command before. But here you *must* provide the (temporary) name of the module due to the way it is imported in the Rover template:



import ti_plotlib as plt

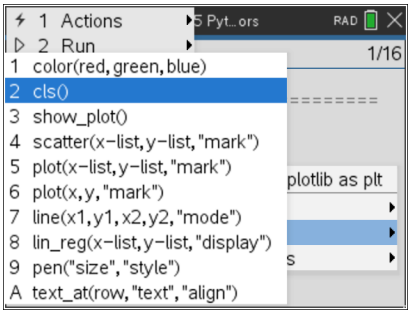
This *requires* that any functions that reside in the **ti_plotlib** module *must* be preceded by the 'alias' name **plt**. You will also use the **cls()** function in the module which clears the drawing screen.

Teacher Tip: The **ti_pltlib** module is used for plotting points, lines, or scatterplots of two lists in a special graphics 'canvas' within Python. It is not meant to replace the Graphs app of the TI-Nspire CX II.
New to Python? Here's some info about imports:

from xxx import *	use functions by name from the xxx module
import xxx	precede all functions in the module xxx with the name xxx.
Import xxx as yyy	precede all functions in the module xxx with the name yyy (the 'alias')

The two latter import methods help to document your code by indicating where the function is stored.

4. See the functions

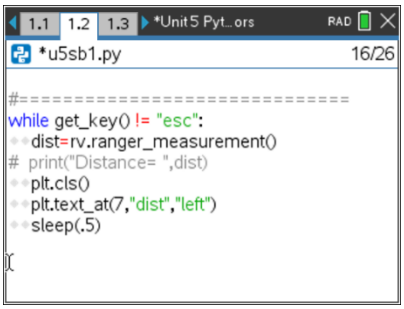


plt.cls()
plt.text_at(…)

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

Other functions found in this module are used for graphing and will be useful when you start using Rover's Coordinate System in the next unit.

5. Turn the **print()** statement into a #comment by pressing **ctrl+T** on the **print()** statement.



Add the two **plt** functions

plt.cls()
plt.text_at(7, str(dist), "left")

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

About **str(dist)** again: The **plt.text_at()** function can only display "text" which is either a literal string "in quotes" or a variable containing a string. **dist** is a variable containing a number, so it needs to be converted to a string using the **str()** function found on:

menu > Built Ins > Type

Row **7** is the vertical center line of the 13-line layout of this screen.
You can sample the distance faster by using a smaller **sleep()** value.
You have just created a digital tape measure!

In the next lesson, you will use this information to keep Rover from crashing into something.

Teacher Tip: It's okay to leave out the **cls()** function for less flicker.
Students can use either "center" or "right" justification, too, but then using **cls()** becomes more important since new data may not properly erase the old data.