



Unit 6: Rover's Coordinates

Skill Builder 2: The Distance Formula

In this lesson, you will use the distance formula from coordinate Geometry to calculate the distance between two points and compare the Rover's measured distance against the calculated distance.

**You will need a meter stick or metric tape measure for this lesson.

Objectives:

- Move to two different points
- Use a marker to draw the segment
- Use a function to compute the distance between two points and display the distance
- Measure the distance between the points
- Compute the error in the measurement versus the computation

Recall the "Distance Formula", which is based on the Pythagorean Theorem:

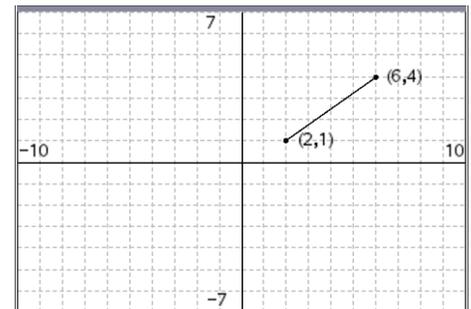
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Based on the image to the right, this becomes the Python statement:

$$d = \text{sqrt}((6 - 2)**2 + (4 - 1)**2)$$

This evaluates to: **d = 5**

Can you find a 3-4-5 right triangle in the image?



1. Start a new Python Rover Coding project.

Define a function called **dist** which takes four arguments (two pairs of coordinates) and will return the distance between the two points.

The **def function()** template is found on <Fns...>

The body of the function consists of one calculation:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

and the **return** statement: **return d**

return is also found on <Fns...>

Make sure these two statements are indented the same amount.

Note: there are two options for evaluating a square root:

$$() ** 0.5$$

sqrt() (must use **from math import *** for this function)

```

EDITOR: RVCOORDB
PROGRAM LINE 0006
# Rover
from time import *
from ti_system import *
import ti_rover as rv
def dist(x1,y1,x2,y2):
    d=
    return d
  
```



- Below the function (after the **return** statement), begin the main program. Be sure your code is no longer indented. Write four **input()** statements (using copy and paste) to enter the coordinates of the two points. Create simple prompts for the inputs and use the **float()** function to convert the input result from a string to a decimal value. Only one of these four statements is shown to the right. We are using the variable **a** to store the first x-coordinate. Use **b**, **c**, and **d** for the other three coordinates.

```

EDITOR: RVCOORDB
PROGRAM LINE 0010
# Rover
from time import *
from ti_system import *
import ti_rover as rv
def dist(x1,y1,x2,y2):
    d=
    return d

a=float(input("x1= ?"))_

```

- After the four **input()** statements, make Rover drive to the first point:

rv.to_xy(a,b)

Pause there while you *insert a marker* in the Rover’s marker holder to draw a line segment. Then continue driving to the second point. A good pause technique is:

print(“insert marker”)

disp_wait()

When you run the program and see ‘insert marker’ wait for Rover to stop and insert a marker into Rover and then press the **[clear]** key on the calculator.

```

EDITOR: RVCOORDB
PROGRAM LINE 0018
a=float(input("x1= ?"))
b=float(input("y1= ?"))
c=float(input("x2= ?"))
d=float(input("y2= ?"))

rv.to_xy(a,b)
print("insert marker")
disp_wait()

```

- Have Rover drive to the second point (making a line segment) and then use the **dist** function to determine the **calculated distance** between the two points:

cd = dist(a, b, c, d)

*we use the variable **cd** for ‘calculated distance’*

*Notice that the letter ‘d’ is used as a variable in two different ways: in the main program, it represents the second point’s y-coordinate but in the **dist()** function it is used to store and return the value of the calculated distance. These two variables do not conflict with each other because they have a different ‘scope’: the part of the program where the variable ‘lives’ (or: is valid).*

```

EDITOR: RVCOORDB
PROGRAM LINE 0019
a=float(input("x1= ?"))
b=float(input("y1= ?"))
c=float(input("x2= ?"))
d=float(input("y2= ?"))

rv.to_xy(a,b)
print("insert marker")
disp_wait()
rv.to_xy(c,d)
cd=dist(a,b,c,d)

```



10 Minutes of Code – Python

TI-84 PLUS CE PYTHON WITH THE TI-INNOVATOR™ ROVER

- Use a ruler or tape measure to determine the length of the segment that Rover made.

Add an `input()` statement to your program so that you can enter the measured distance, `md`.

```

EDITOR: RVCOORDB
PROGRAM LINE 0021
c=float(input("x2= "))
d=float(input("y2= "))

rv.to_xy(a,b)
print("insert marker")
disp_wait()
rv.to_xy(c,d)
cd=dist(a,b,c,d)
md=float(input("measured distance
e= "))

```

- Add two `print()` statements to display the **two distance values**.
How does the measured distance compare to the calculated distance?

```

EDITOR: RVCOORDB
PROGRAM LINE 0023
rv.to_xy(a,b)
print("insert marker")
disp_wait()
rv.to_xy(c,d)
cd=dist(a,b,c,d)
md=float(input("measured distance
e= "))
print("calculated= ",cd)
print("measured = ",md)

```

- Calculate the percent error using the formula
 $(\text{measured} - \text{calculated}) / \text{calculated} * 100$
and print the error.

```

EDITOR: RVCOORDB
PROGRAM LINE 0026
disp_wait()
rv.to_xy(c,d)
cd=dist(a,b,c,d)
md=float(input("measured distance
e= "))
print("calculated= ",cd)
print("measured = ",md)

e=(md-cd)/cd*100
print("error= ",e)

```