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:

$$d=\sqrt{(x2-x1)^2+(y2-y1)^2}$$

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

d = 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:

   $$d=\sqrt{(x2-x1)^2+(y2-y1)^2}$$

   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)

Teacher Tip: To use one inch rather than 10 centimeters per grid unit, set Rover's unit to:
rv.grid_m_unit(0.0254)
To use the **sqrt( )** function add **from math import *** to the top of your code and then get the **sqrt( )** function from **[math] math…**

2. 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.

Teacher Tip: Using different variables for the *actual* parameters **(a, b, c, d)** reinforces the idea that these identifiers are really different from the *formal* parameters **(x1, x2, y1, y2)** used in defining the function.

3. 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.

4. Have Rover drive to the second point (making a line segment) and then use the dist function to determine the **c**alculated **d**istance 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).*

Teacher Tip: The long variable name is used for clarity. The students will measure the length of the segment using a meter stick or a metric tape measure and compare it to the **cd.**

5. 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 distanc
     e= ?"))

Fns… | a A # | Tools | Run | Files
```

6. 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 distanc
     e= ?"))
print("calculated= ",cd)
print("measured  = ",md)
_
Fns… | a A # | Tools | Run | Files
```

7. Calculate the percent error using the formula

   (measured - calculated*)* / 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 distanc
     e= ?"))
print("calculated= ",cd)
print("measured  = ",md)

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

Fns… | a A # | Tools | Run | Files
```

Teacher Tip: If you have a set of Rovers, try this program on each one to determine which Rover is the 'most accurate'.