



#### Unit 6: Rover座標

#### Skill Builder 2: 距離の公式

このレッスンでは、距離の公式を使って2点間の距離を計算し、Roverが測定した距離と計算した距離を比較します。このレッスンでは、定規または巻尺が必要になります。

#### 目標

- 異なる2点間を移動
- マーカーを使って線分を描画
- 関数を使って2点間の距離を計算し、表示
- 2点間の距離を測定
- 測定値と計算結果の誤差を計算

ピタゴラスの定理による距離の公式を思い出しましょう。

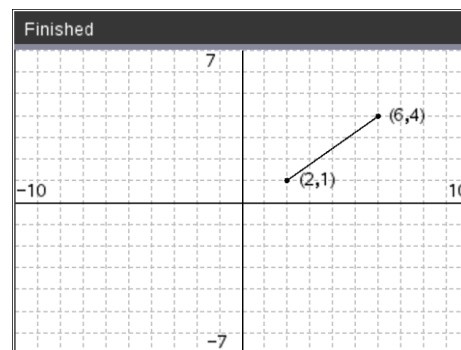
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

次の式は、右図に基づくPythonステートメントです。

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

これを計算すると  $d = 5$

右図に、3辺が3, 4, 5の直角三角形はありますか。



- 新規のPython Rover Codingプロジェクトを開始します。

4つの引数(2組の座標)を取り、2点間の距離を返すdistという関数を定義します。

**def function():**テンプレートは、**menu > Built-ins > Functions**(メニュー>組み込み>関数)にあります。

関数の本体は、次の計算で構成されます。

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

そして、**return**ステートメントです：**return d**

**return**は、**menu > Built-ins > Functions**(メニュー>組み込み>関数)にあります

これら2つのステートメントは同じ量だけインデント(字下げ)されています。

```

1.4 1.5 1.6 *Unit 6 Py...rds RAD
*u6sb2.py 13/16
import ti_rover as rv
from math import *
import ti_plotlib as plt
from ti_system import *
from time import *
=====
# distance between two points
def dist(x1,y1,x2,y2):
    d =
    return d
    
```

- 関数の下(**return**ステートメントの後)で、メインプログラムを開始します。コードがインデントされていないことを確認します。2つの点の座標を入力するため4つの**input()**ステートメントを記述します(コピー&ペーストを使います)。入力プロンプトを作成し、**float()**関数を使って入力結果を文字列から10進数値に変換します。4つのステートメントのうちの1つが右図に示されています。変数**a**を使って、最初のx座標を格納しています。他の3つの座標は**b, c, d**を使います。

```

1.4 1.5 1.6 *Unit 6 Py...rds RAD
*u6sb2.py 15/18
import ti_plotlib as plt
from ti_system import *
from time import *
=====
# distance between two points
def dist(x1,y1,x2,y2):
    d =
    return d
a = float( input("x1 = ?") )
    
```



- 4つのinput()ステートメントの後、Roverを最初の点までドライブさせます。そこで一時停止します。Roverのマーカーホルダーに線分を描くマーカーを挿入します。つぎに、2番目の点までドライブを続けます。適切な一時停止ステートメントは、次のとおりです。

**input(“press [enter] to continue.”)**

input(「続行するには[enter]を押してください。」)

この入力関数の結果は、何も入力されていないため、変数に値を割り当てません。

- つぎに、プログラムに2点の座標を使って距離関数を評価させます。

**calculated\_distance = dist(a, b, c, d)**

- 定規または巻尺を使って、Roverが作成した線分の長さを決定します。

input()ステートメントをプログラムに追加して、**measured\_distance**を入力できるようにします。

print()ステートメントを追加して、2つの距離変数を表示します。

測定された距離は、計算された距離とどのように比較されますか。

- 式を使って誤差(パーセント)を計算します。

**(measured - calculated) / calculated \* 100**

そして、エラーを出力します。

```

1.2 1.3 1.4 *Unit 6 Py...rds RAD 23/33
#u6sb2.py
#####
a = float( input("x1 = ?") )
b = float( input("y1 = ?") )
c = float( input("x2 = ?") )
d = float( input("y2 = ?") )
#drive...
rv.to_xy(a,b)
print("insert marker")
input( "press [enter] to continue.")

```

```

1.4 1.5 1.6 *Unit 6 Py...rds RAD 29/34
#u6sb2.py
d = float( input("y2 = ?") )

rv.to_xy(a,b)
print("insert marker")
x=input( "press [enter] to continue.")
rv.to_xy(c,d)

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

```

```

1.4 1.5 1.6 *Unit 6 Py...rds RAD 30/34
#u6sb2.py
d = float( input("y2 = ?") )

rv.to_xy(a,b)
print("insert marker")
x=input( "press [enter] to continue.")
rv.to_xy(c,d)

calculated_distance = dist(a,b,c,d)
measured_distance = float(input("Measured distance? "))
|

```