Unit 6: Rover座標

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。

Skill Builder 2: 距離の公式

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。

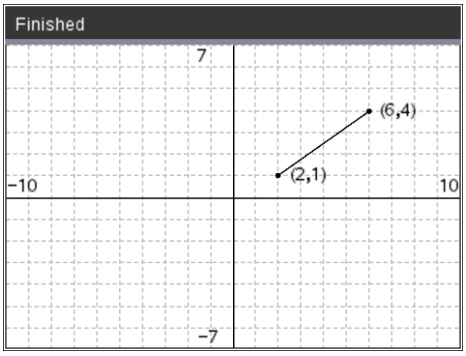
$$d=\sqrt{(x2-x1)^2+(y2-y1)^2}$$

次の式は，右図に基づくPythonステートメントです。

d = 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。



1. 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。

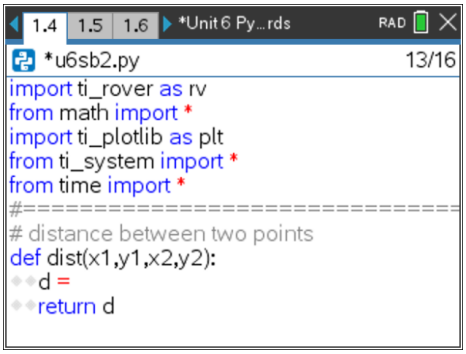
   関数の本体は，次の計算で構成されます。

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

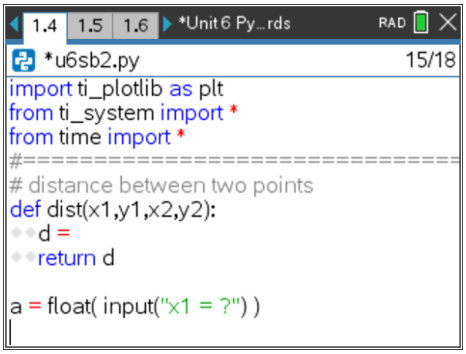
   そして，**return**ステートメントです：**return d**

   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。



2. 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。

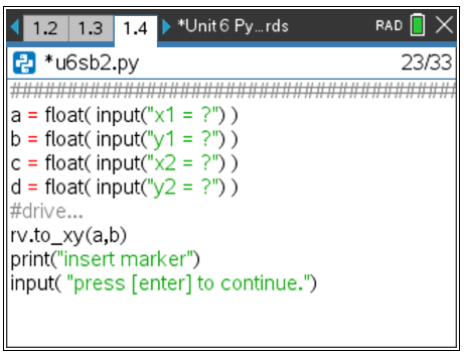


3. 4つの**input()**ステートメントの後，Roverを最初の点までドライブさせます。そこで一時停止します。Roverのマーカーホルダーに線分を描くマーカーを挿入します。つぎに，2番目の点までドライブを続けます。適切な一時停止ステートメントは，次のとおりです。

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

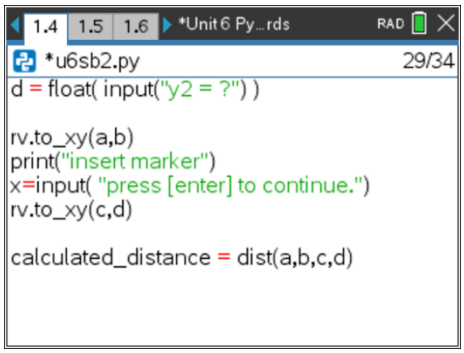
   input(「続行するには[enter]を押してください。」)

   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。



4. 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)

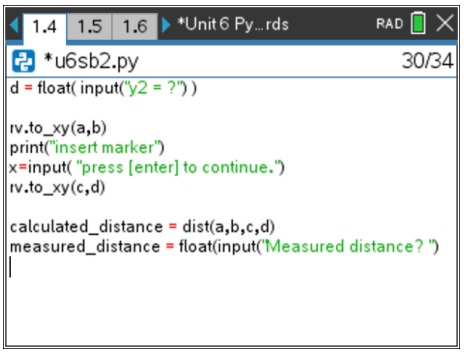


5. 定規または巻尺を使って，Roverが作成した線分の長さを決定します。

   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。



6. 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。