



Unit 6: Rover座標

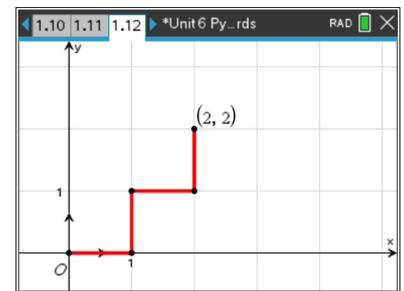
Application: ランダムウォーク

この応用では、Roverが原点(0, 0)から点(2, 2)までグリッド(格子)上を移動する頻度(北と東へのみランダムに回転させる)を確認できます。また、その点に到達する実験的および理論的な確率を調べます。

目標

- 乱数を使って次に進む方向を決定
- ゴール(2,2)に到達するかどうかを判断
- 画面に座標をプロット
- カラーLEDを使って成功または失敗を報告
- 実験を複数回繰り返し、Roverが目標に到達する頻度を決定
- 目標に到達する理論的な確率を調べる

Roverは原点Oからスタートします。各格子点で、Roverは一度に1グリッド単位ずつ、ランダムに東(右)または北(上)に移動できます。Roverがゴール(2, 2)に到達するために取ることができる異なるルートがいくつかあります。1つは、右図で赤/太字で強調表示されています。



Roverはいくつの異なる方法で目標を達成できますか。各動きがランダムである場合、Roverがゴールを逃す可能性はありますか。

Roverが目標を達成する確率はどれくらいですか。

この問題のシミュレーションを作成して、Roverが目標に到達した回数を追跡し、成功した試行の割合を決定します。

失敗について考えてみましょう。Roverが目標を達成できなかったことはどうやってわかりますか。

1. 新規のPython Rover Codingプロジェクトを開始します。

TI PlotLib > Setup(TI PlotLib>セットアップ)メニューを使って、ルートに沿ったRoverの位置を表示できるようにplt画面を設定します。

```
plt.window(-1, 4, -1, 4)
plt.grid(1, 1, "dashed")
plt.axes("on")
```

```
*u6App.py 8/42
from math import *
import ti_plottlib as plt
from ti_system import *
from time import *
#-----
plt.window(-1,4,-1,4)
plt.grid(1,1,"dashed")
plt.axes("on")
```

2. 試行(trial)回数の変数を設定します : **trials = 10**

成功(success)を数える変数を設定します : **successes = 0**

ゴールの点には2つの変数を設定します : **px, py = 2, 2** (これは有効です!)

forループを使って試行を実行します : **for i in range(trials):**

試行を開始します。

ループblockで、Roverの位置に2つの変数を使います : **rx, ry = 0, 0**

```
*u6App.py 13/51
trials=10
successes = 0
px, py = 2, 2
for i in range(trials):
    rx, ry = 0, 0
    ..
    ..
    ..
```



3. グラフ画面にその点をプロットします : `plt.plot(rx,ry,"o")`

```

1.8 1.9 1.10 *Unit 6 Py...rds RAD 18/44
u6App.py
#-----
plt.window(-1,4,-1,4)
plt.grid(1,1,"dashed")
plt.axes("on")
trials = int(input("# of trials? "))
successes = 0
px, py = 2, 2
for i in range(trials):
    rx, ry = 0, 0
    plt.plot(x,y,"o")

```

4. Roverがゴールの点にいない限り継続するwhileループを作成します。
ローバーは失敗しません。どのような状態が失敗になるかを定めることを考えましょう。
この条件は、演習として残します。

```

1.10 1.11 1.12 *Unit 6 Py...rds RAD 25/44
u6App.py
for i in range(trials):
    rx, ry = 0, 0
    while (rx,ry)!= (px,py) and ____? ____
    .....
    .....
    .....
    .....
    .....
    .....
    .....

```

5. whileブロックで、`randint(0, 1)`を使って、東(0°)または北(90°)のどちらに移動するかを決定します。これは、次のステートメントで実現されます。

`dir = randint(0, 1) * 90`

`randint()`ステートメントを使うには、`from random import*`をインポートする必要があります。Rover Codingテンプレートには含まれていません。

```

1.10 1.11 1.12 *Unit 6 Py...rds RAD 21/45
u6App.py
trials=10
successes = 0
px, py = 2, 2
for i in range(trials):
    rx, ry = 0, 0
    while (rx,ry)!= (px,py)
        dir=randint(0, 1) * 90
    .....
    .....
    .....

```

6. Roverを正確な角度(`dir`)に向けて1単位前進させます。
(右図の記述は不完全です。)

プロット速度を制御するため`rv.wait_until_done()`ステートメントを追加します。

```

1.10 1.11 1.12 *Unit 6 Py...rds RAD 23/45
u6App.py
while (rx,ry)!= (px,py) and
    dir=randint(0, 1) * 90
    rv.to_angle( )
    rv.forward( )
    .....
    .....
    .....
    .....

```

7. Roverの位置変数`rx`と`ry`を更新します。

Roverが東に移動した場合(`dir == 0`),
`rx`に1を足します。
そうでないとき,
`ry`に1を足します。

`plt.plot(rx,ry,"o")` (図には示されていません)を使って、画面内のRoverの位置をプロットします。

```

1.8 1.9 1.10 *Unit 6 Py...rds RAD 30/50
u6App.py
while (rx,ry)!= (px,py) and rx<3 and ry<3:
    dir=randint(0, 1) * 90
    rv.to_angle(dir)
    rv.forward(1)
    if dir==0:
        block
    else:
        block
    .....
    .....
    .....

```

8. **while**ループが終了したら、Roverが成功したかどうかを判断します。

if (rx, ry) == (px, py):

変数**successes**に1を加えて、成功をカウントします。

“SUCCESS!” または “FAIL”を印刷します。

Roverが成功した場合はカラーLEDを**GREEN** (緑)で点灯します。それ以外はカラーLEDを**RED** (赤)で点灯します。

つぎに、次のように入力します(表示されていません)。

- Roverが(0, 0)に戻り、東向き(角度0°)になるための2つのステートメント
- LEDをオフにするステートメント
- plt.cls()**で始まるプロット画面を再描画し、3つの**Setup**ステートメントを再度使うステートメント

9. **for**ループが終了した後(すべての試行が終了したとき)、実験結果を印刷します。

- successes**(成功)の総数
- 成功の割合(**successes / trials *100**)

右図は、10回の試行サンプルです。パーセンテージは、成功の実験的確率 (**experimental probability**)です。

10. **理論的確率(theoretical probability)**

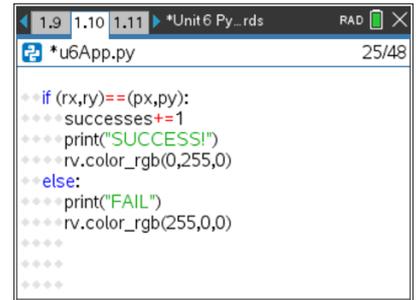
(0, 0)から(2, 2)へは、6つの成功ルートがあります。

x=3またはy=3のいずれかへは、10個の失敗ルートがあります(図の1と4を参照)。理論上の確率は次のとおりです。

$$\frac{\text{success}}{\text{success} + \text{fail}} = \frac{6}{6+10} = \frac{3}{8}$$

成功率は37.5%です。

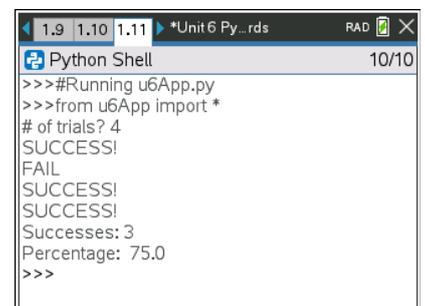
あなたの実験はどれくらい近づきましたか。より正確な実験的確率 (**experimental probability**)を得るためには何をすべきですか。



```

1.9 1.10 1.11 *Unit6 Py...rds RAD 25/48
u6App.py
if (rx,ry)==(px,py):
    successes+=1
    print("SUCCESS!")
    rv.color_rgb(0,255,0)
else:
    print("FAIL")
    rv.color_rgb(255,0,0)

```



```

1.9 1.10 1.11 *Unit6 Py...rds RAD 10/10
Python Shell
>>>#Running u6App.py
>>>from u6App import *
# of trials? 4
SUCCESS!
FAIL
SUCCESS!
SUCCESS!
Successes: 3
Percentage: 75.0
>>>

```

