



Unit 2: TI-Innovator™ Hubのforループ

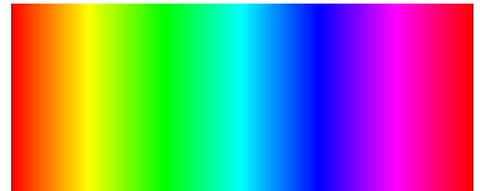
Skill Builder 2: 色をループ

このレッスンでは、いくつかのforループを使ってカラーLEDでさまざまな色を作成する混色について学習します。

目標

- **range()**関数でforループを使用
- カラーLEDで可能な多くの色を制御
- 長いプログラムのコーディング簡略化のためコピー/貼り付け/編集を使用

TI-Innovator HubのカラーLEDを使うと、虹のすべての色(それ以上)が可能です。このレッスンでは、forループを使ってカラーLEDをさまざまな色に変更するプログラムを作成します。さまざまなループオプションを使って練習することができます。



1. 新規のPython Hub Projectを開始します。

このプログラムは、LEDのカラーチャンネルを一度に1つずつ徐々に増減して、いくつかの色を混ぜ合わせます。

step(ステップ)値と**delay**(遅延)値を入力する2つの入カステートメントを記述します。**step**値は**int()**にすることができますが、**delay**は0から1(10進数の値)の間である場合があるので、**float()**にする必要があります。

stepは色の値の間のスペースであり、**delay**は各色のステップの時間遅延です。

```

1.3 1.4 1.5 *Unit 2 Py...ops RAD 11/36
*u2sb2.py
#=====
from ti_hub import *
from math import *
from time import sleep
from ti_plotlib import text_at,cls
from ti_system import get_key
#=====
step=int(input("Step? "))
delay=float(input("Delay? "))

```

2. menu > Built-ins > Control (メニュー>組み込み>制御)を押して、メニューのfor index in...セクションを確認します(右図参照)。

これらのステートメントはすべて**range()**関数を使いますが、引数が異なります。3つの中で最も用途が広いのは、次のステートメントです。

```

4 for index in range(size):
5 for index in range(start, stop):
6 for index in range(start, stop, step):

```

for index in range(start, stop, step):

Teacher Tip: **range()**を使うforループは、ループをカウントして(数えて)います。**step**は正の数または負の数のいずれかになります。**stop**(停止値)はループ内で処理されませんが、最終的に到達する必要があります。そうしないと、ループが終了しません(例: for i in range(10,5,2)の場合、10, 12, 14, 16, 18, ...が生成されます)。

3. ステートメントを選択します。

for index in range(start, stop, step):

indexを変数iに変更します。**start**には0、**stop**には256、**step**には**step**を入力します。(これは入カステートメントで使われる変数であるため、stepという単語を入力する必要があります。)

これにより、ループの各ステップに**step**を追加することにより、ループを0から最大255まで移動させます。たとえば、**step**が10の場合、iの値は次のようになります。

0, 10, 20, 30, ... まで?

```

1.2 1.3 1.4 *Unit 2 Py...ops RAD 12/57
*u2sb2.py
from ti_hub import *
from math import *
from time import sleep
from ti_plotlib import text_at,cls
from ti_system import get_key
#=====
step=int(input("Step? "))
delay=float(input("Delay? "))
# increase red
for i in range(0,256,step):
*block

```



4. ループblockの使用

color.rgb(i, 0, 0) (赤のみ)

これは、**menu > TI Hub > Hub Built in Devices > Color Output** (メニュー>TI Hub>Hub内蔵デバイス>カラー出力)にあります。

ctrl+Rを押してプログラムを実行します。LEDがすぐに点灯することに注意してください。ゆっくり点灯するはずです。**delay**(遅延)変数を使って、プロセスを少し遅くする必要があります。次のステップに進む前に、自分で試してみてください。

```
*u2sb2.py 12/62
from math import *
from time import sleep
from tiplotlib import text_at,cls
from ti_system import get_key
#=====
step=int(input("Step? "))
delay=float(input("Delay? "))
# increase red
for i in range(0,256,step):
    color.rgb(i,0,0)
**
```

5. sleep(delay)ステートメントを追加しましたか。

これで、LEDは徐々に赤くなります。つぎに、別の**for**ループを記述して、赤色LEDに徐々に緑色を加えます。

赤の**for**ループをコピーしてループの下に貼り付け、テキストを編集して緑のチャンネルのみを制御できます。

インデント(字下げ)に注意してください。

赤チャンネルがつねに完全に点灯していることを確認してください。

```
*u2sb2.py 6/62
from tiplotlib import text_at,cls
from ti_system import get_key
#=====
step=int(input("Step? "))
delay=float(input("Delay? "))
# increase red
for i in range(0,256,step):
    color.rgb(i,0,0)
    sleep(delay)
**
```

Teacher Tip: ループ変数*i*を再利用してもかまいません。

コピー&ペースト? コンピュータの場合と同じように:

Shift+矢印 キストを選択します。

Ctrl+c でコピーしてから、

Ctrl+v で貼り付けます。

6. 緑色のループは次のとおりです。

for i in range(0, 256, step):
color.rgb(255, i, 0)
sleep(delay)

赤のチャンネルが255に設定され、ループ変数が緑の位置にあることに注意してください。

プログラムを再実行してください。プログラムの最後に今、何色が見えますか。

```
*u2sb2.py 14/52
from ti_system import get_key
#=====
step=int(input("Step? "))
delay=float(input("Delay? "))
# increase red
for i in range(0,256,step):
    color.rgb(i,0,0)
    sleep(delay)
for i in range(0,256,step):
    color.rgb(255,i,0)
    sleep(delay)
**
```

7. ここで、トリッキーな部分について...forループの方向を変更して、赤みを徐々に取り除きます。

for i in range(255, 0, -step):

このループは*i*=255で始まり、ループインデックス*i*はステップ値だけ減少します。

```
*u2sb2.py 22/53
for i in range(0,256,step):
    color.rgb(i,0,0)
    sleep(delay)
for i in range(0,256,step):
    color.rgb(255,i,0)
    sleep(delay)
for i in range(255,0,-step):
    color.rgb(i,255,0)
    sleep(delay)
**
```



Teacher Tip: ループで値0に近づきますが、0使いません。ステップによっては、0に達する直前に停止します。インデックスが停止値と等しいか通過すると、ループが終了することに注意してください。

8. 各forステートメントの前に# commentsを追加して、それが何をするのかをより明確に説明します。物事を整理しておく役立ち場合があります。コメントは#記号で始まり(ctrl+Tを押す)、プログラムの実行時には無視されます。

このプロジェクトを完了するには、次の場所にループを追加します。

- 青を増やす
- 緑を減らし、青は維持する
- 赤を増やす(再び)
- 青を減らすが、赤は維持する
- 赤を減らしてゆっくりとオフにする

このプロセスでは、赤-緑、緑-青、青-赤の3つの色のペアすべてが混合されます。その過程で、黄色、シアン、マゼンタも表示されます。

プログラムの終了時に、LEDはオフになっているはずですが。

作業内容を保存することを忘れないでください。

```

delay=float(input("Delay? "))
# increase red
for i in range(0,256,step):
    color.rgb(i,0,0)
    sleep(delay)
#increase green
for i in range(0,256,step):
    color.rgb(255,i,0)
    sleep(delay)
#decrease red but keep green
for i in range(255,0,-step):

```

Teacher Tip: 最終的なコードはかなり長いのですが、それを入力することはコピー/貼り付け/編集によって容易になります。

```

step=int(input("Step? "))
delay=float(input("Delay? "))
# increase red
for i in range(0,256,step):
    color.rgb(i,0,0)
    sleep(delay)
# increase green
for i in range(0,256,step):
    color.rgb(255,i,0)
    sleep(delay)
# decrease red
for i in range(255,0,-step):
    color.rgb(i,255,0)
    sleep(delay)
# increase blue
for i in range(0,256,step):
    color.rgb(0,255,i)
    sleep(delay)
# decrease green
for i in range(255,0,-step):
    color.rgb(0,i,255)
    sleep(delay)
# increase red

```



```
for i in range(0,256,step):
    color.rgb(i,0,255)
    sleep(delay)
# decrease blue
for i in range(255,0,-step):
    color.rgb(255,0,i)
    sleep(delay)
# decrease red then all off
for i in range(255,0,-step):
    color.rgb(i,0,0)
    sleep(delay)
color.off()
```