Unit 1: PythonによるTI-Innovator™ Hubのスタート

Application: 信号機

この応用では，信号機をシミュレーションするプログラムを作成します。信号と音の両方(視覚障害者用)です。

目標

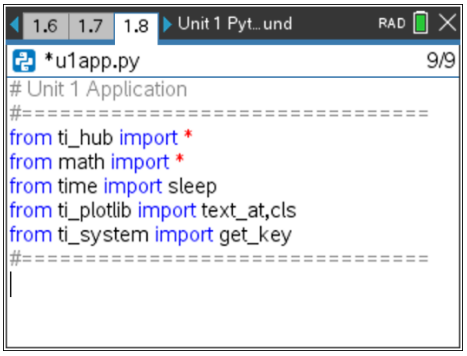
- 信号機の赤, 黄, 緑状態のタイミング制御
- 光, 色, 音を1つのプロジェクトに
- **while**ループを使って**esc**が押されるまで手順を繰り返す

信号機には，赤，黄，緑の3つの電球(またはLED式)があり，色覚異常の人は点灯している電球の位置だけで電球の状態を知ることができます。

道路の脇には，目の見えない歩行者に安全に通りを横断できることを知らせる音声信号もあります。信号機の一部として音を出します。

1. 信号が赤，黄，緑の秒数を入力するプログラムを作成します。信号が赤の場合，カラーLEDと赤のLEDの両方が点灯している必要があります。信号が黄色または緑色の場合は，カラーLEDのみが点灯している必要があります。信号が緑色のときは，安全であることを示す音声信号もあるはずです。

   新規のPython Hub Projectから始めます。私たちのものはu1appと呼ばれています。

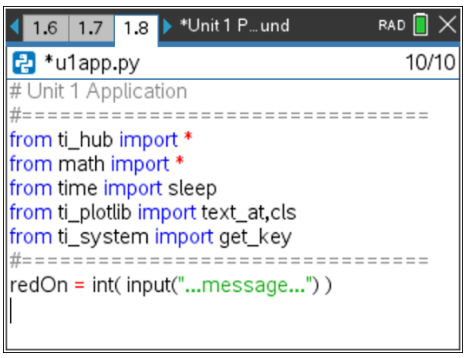


Teacher Tip: このプロジェクトでは，**while get_key()!=**'esc'ループを導入して，**esc**キーが押されたときにプログラムを終了します。これは，キーを押すだけで終了するプログラムを作成する最も簡単な方法です。

2. ライトが赤，黄，緑になる時間(秒単位)の3つの**input()**ステートメントを追加します。簡単にするために，整数のみを使うため，**input()**関数の前後で**int()**関数を使って，入力された文字列を数値に変換することを忘れないでください。

   スクリーンショットは，1つのサンプルステートメントのみを示しています。

   redOnの大文字のOに注意してください。Pythonでは大文字と小文字が区別されるため，この変数のすべての呼び出しはまったく同じ方法で記述する必要があります。



3. 赤信号から始めます。**color** LEDを赤にし，**light**(赤色LED)もオンにします。

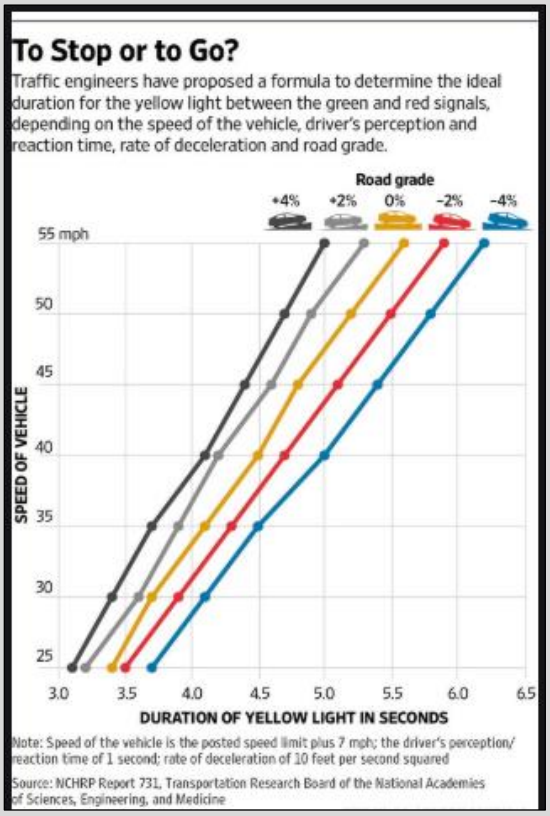
   次のステートメントにより**sleep()**関数を使って，**redOn**秒数の間プログラムを一時停止します。

   sleep(redOn)

   colorが黄色に変わったら，必ず**light**ライトを消してください。

```
1.6 1.7 1.8  *Doc  RAD
*u1app.py  15/15
from time import sleep
from ti_plotlib import text_at,cls
from ti_system import get_key
#==============================
redOn=int(input("...message..."))
# another input
# another input
color.rgb(255,0,0)
light.on()
sleep(redOn)
light.off
```

Teacher Tip: 大文字はまれですが，Pythonプログラムでは許可されています。もう1つの規則は，アンダースコア文字red_onです。TI-Nspire グラフ電卓では，このアンダースコア(_)文字は，文字Gキーの右側の句読点キーにあります。大文字は**shift**キーを使って作成します。
黄色のライトはどのくらいの長さですか。速度と道路の勾配によって異なります。



カリフォルニア州サンフランシスコで最も急な通りは，2.5%または約18°の勾配のフィルバートストリートです。

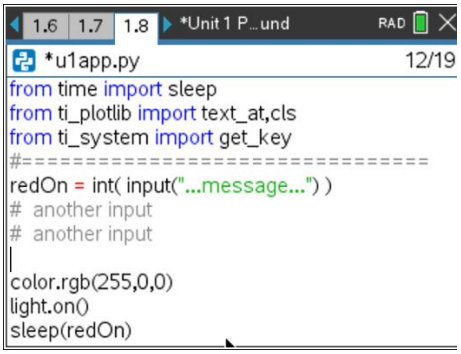
4. 黄色の信号の場合はLEDの色を黄色に変え，つぎに色の信号の場合は緑色に変えます。

   緑色のライトが点灯しているときに，スピーカーを使って音を出します(**.tone**または**.note**のいずれかを使います)。緑色のライトが点灯している間，サウンドはオンのままで，その後オフになります。

   プログラムを実行してテストします。TI-Innovator Hubのライトをご覧ください。

Teacher Tip: 次のセクションでは、whileループを紹介します。

5. プログラムが正常に機能した場合，プログラムは3色の1サイクル後に終了しました。プログラムが3色を繰り返し循環できるようにするには，コードにループを追加します。



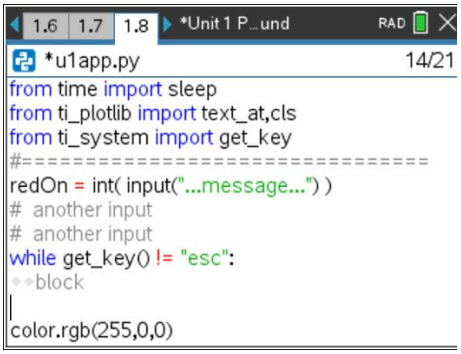
6. **loop**は，コードのブロックを繰り返し処理するプログラミングの制御構造です。Pythonループの1つのタイプは，**while** <condition>：loopです。<condition>はブール式であり，**True**または**False**のいずれかに評価されます。

例： **while x<10:** (コロン(:)が必要)
block (ブロックは意図的にインデント(字下げ))

xが10未満である限り，ブロック内のステートメントは繰り返し実行されます。xが10以上の場合，ループは終了し，処理はブロックの下の最初のステートメントに渡されます。

Teacher Tip: while構造内のステートメントは，同じ量だけインデント(字下げ)する必要があります これは，コードのブロックを決定するPythonの方法です。

7. 3つの入力ステートメントの下のプログラムに空白行を挿入します。



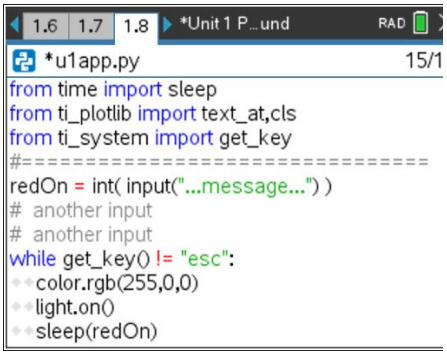
つぎに，ステートメントを取得します。

while get_key() != ”esc”:

これは，**menu > TI Hub > Commands(**メニュー>TI Hub>コマンド)から。
薄い灰色の単語blockは2つのスペースでインデントされており，これらのスペースには薄い灰色ののひし形◆のプレースホルダーがあります(実際には単なるスペースです)。

!=は，≠(ノットイコール)のPython記号です。

8. **while**ループは，escキーが押されるまで，ブロック内のすべてのステートメントを繰り返し実行します。ブロックはインデント(2つのスペース)によって定義されます。



block(インラインプロンプト(行内入力要請))という単語を削除してから，プログラムの残りの部分を2つのスペースでインデント(字下げ)します。各行の先頭に2つのスペースを入力するか，各行の先頭で**Tab**キーを押すか，あるいは次のショートカットが使えます。

whileステートメントの下にあるすべてのステートメントを選択し(**shift+下矢印**を使用)，**tab**キーを押します。これにより，選択した各行がインデントされます。

9. すべての色，光，音のステートメントが同じ数のスペース(2)でインデントされていることを確認します。これにより，すべてのlightコードが**while**ブロックになります。**注意**：不適切なインデントはエラーにつながる可能性があります。

   プログラムを再度実行します。プログラムを停止する準備ができたら，**esc**を押します。プログラムはすべてのブロックコードを通過する必要があり，緑色のライトが終了した後にのみループを停止するため，プログラムの停止に遅延が生じる可能性があります。

   プログラムが終了すると，TI-Innovator HubのLEDの状態はどうなりますか。 両方がオフになっていることをどのように確認しますか。

   Teacher Tip: escを押すと，カラーLEDが緑色になります。オフになっていることを確認するには，プログラムの下部にcolor.off()というステートメントを追加するだけです。ループの外側にある必要があります(インデントされていません)。

   考えられる解決策の1つ：

```
redOn = int( input("Red time? ") )
yellowOn = int( input("Yellow time? ") )
greenOn = int( input("Green time? ") )
while get_key() != "esc":
  color.rgb(255,0,0)
  light.on()
  sleep(redOn)
  light.off()
  color.rgb(255,255,0)
  sleep(yellowOn)
  color.rgb(0,255,0)
  sound.note("A4",greenOn)
  sleep(greenOn)
color.off()
```