実例1：マトリクスLEDの制御

鈴木 量三朗

リスト1　Pythonでフォント・データを生成する

```
# CQ の文字
DATA_SRC = (
    ' **     ',
    '*       ',
    '*       ',
    ' **  ** ',
    '    *  *',
    '    *  *',
    '     ** ',
    '       *'
)

delim = 'FONT_DATA = ('
for data_row in DATA_SRC:
  print(f'{delim}0b', end='')
  for data in data_row:
    v = '1' if data == '*' else '0'
    print(f'{v}', end='')
  delim = ', \n\t'
print(')')
```

基礎編第4章で8×8のドット・マトリクスLEDを制御しています．これをPythonでも扱ってみます．

準備 1：表示するフォントのデータはPythonで作る

ここではマトリクスLEDに文字を表示します．その表示するフォント・データはPythonを使って生成します（**リスト1**）

生成されたデータである`data.py`（**リスト2**）はPythonプログラムの形になっており，他のPythonプログラムから`import`で読み込むことができます．

フォント・データがこのように1つであれば，手で編集した方が効率が良いでしょう．しかし，今回筆者は最終的に複数のフォント・データを作りました．その際にはLinuxのコマンドである`banner`コマンドを利用しました．そこからさらに幾つか手を加えてPythonのデータにしました．

また，Tang Nano 9Kのサンプル`svo_term.v`に含まれているフォント･データを利用するのもよいでしょう．

回路 1：マトリクスLEDを行と列に分けて制御

マトリクスLEDを制御するインターフェースは行（`row`）と列（`col`）を基本としています．ある瞬間に特定の行のスイッチだけを入れ，さらに列をグラウンドにすることでLEDが光ります．その瞬間は他の行は全て消灯していますが，高速に行を切り替えることで，あたかも複数のLEDを同時に光らせているかのように見せています．

リスト2　Pythonで生成したフォント・データ

```
FONT_DATA = (0b01100000,
             0b10000000,
             0b10000000,
             0b01100110,
             0b00001001,
             0b00001001,
             0b00000110,
             0b00000001)
```

リスト3　マトリクスLEDを行と列で制御するための初期化処理

```
  def __init__(self, interval):
    self.row = Port(bit8, 'out')
    self.col = Port(bit8, 'out')

    self.interval = interval
    self.append_worker(self.main)
```

`row`と`col`のインターフェースを含めた初期化処理をPolyphonyで書くと**リスト3**のようになります．

回路 2：行と列にデータを設定する処理

中心となる処理を`main`のワーカに構築します．アルゴリズムは基礎編第4章と同じなので，そのtopモジュールを参考にPythonのプログラムに変更していきます．

`main`の処理は`row_cnt`で示される行のフォント・データを`col`に設定し，さらに対応する`row`をONにします．

これで“CQ”の文字がマトリクスLEDに表示されます．フォント・データは先ほど作ったものを`import`して利用しています．

`_wait`で待つ時間は27000回のループなので，1/1000秒のオーダです．この時間を長くするとマトリクスLEDが上から順に，行ごとに点灯していく様子を見られます（**リスト4**）．

表示できるフォントを増やす

フォント・データを次の4つに増やしてみます