# データ科学基礎演習B（8）

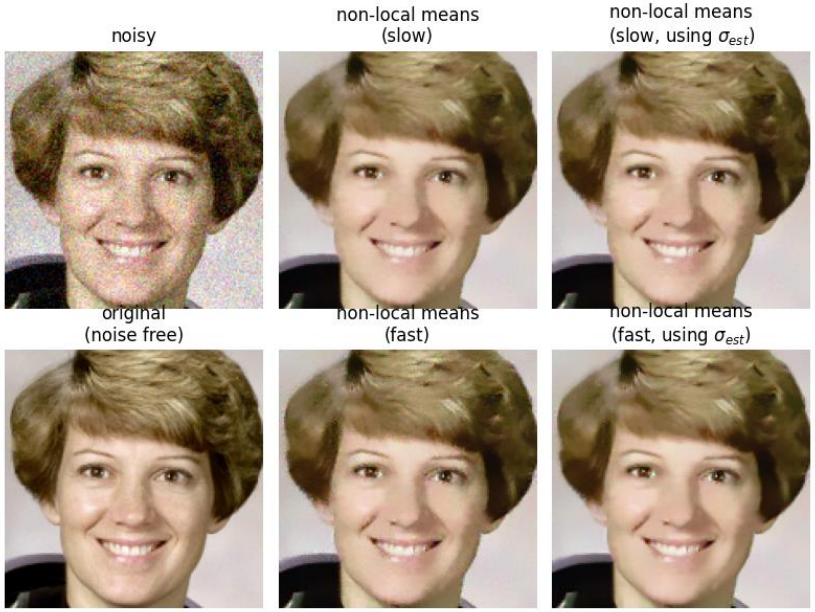
データ科学科目部会

# 画像の入出力

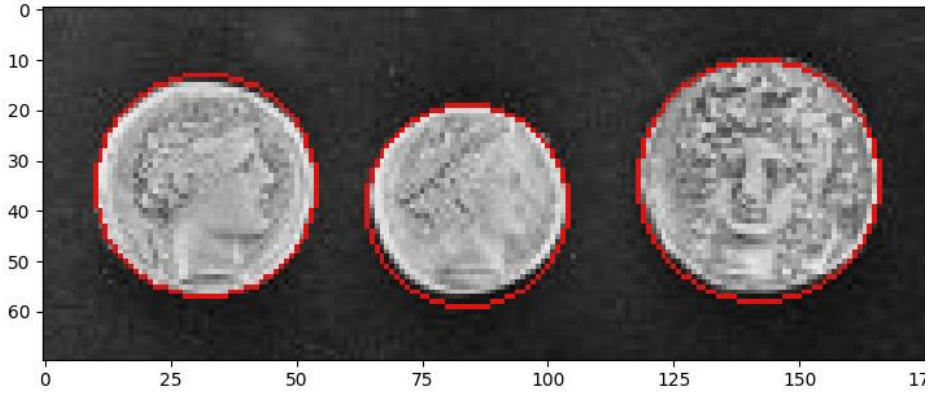
scikit-image に触れてみよう

# scikit-image（https://scikit-image.org/）とは？

- 様々な画像処理手法を網羅したライブラリ
  - 画像入出力，可視化，フィルタリング，特徴抽出，etc.
- Numpy配列を使って画像処理ができる
  - scikit-learn と相性が良い画像処理ライブラリ




ノイズ除去の例




円検出の例

※図はsckit-imageホームページより引用

# scikit-imageの入出力機能をインポート

- scikit-imageのモジュール名は skimage
- scikit-imageは非常に大きなモジュールのため，必要な機能のみをインポートしよう

- 入出力機能をインポート
  – skimage モジュールから io をインポート

```
from skimage import io
```

# scikit-imageを用いた画像の入出力

- 覚える関数は2つ
  - imread 関数

ファイル（URL）から画像を読み込む

    - jpeg，png，gif 等の様々な画像フォーマットに対応
  - imshow 関数

画像の変数の中身を可視化

    - scikit-imageのimreadで読み込んだ画像を表示

# 画像の読み込み(imread 関数)

- 画像のURLを指定して画像を読み込む

```
img = io.imread(画像URL)
```

画像データを格納した
numpy配列

# 画像の表示(imshow 関数)

- 画像を格納した変数を指定して画像を表示

```
io.imshow(画像を格納した変数)
```

画像データを格納した
numpy配列

# 画像の読み込みと表示

- 下記URLの画像を読み込んでみよう

```
https://bit.ly/39pAu0V
```

```
[1]  1  from skimage import io

[2]  1  img = io.imread('https://bit.ly/39pAu0V')

[3]  1  io.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f401ca6a390>
```

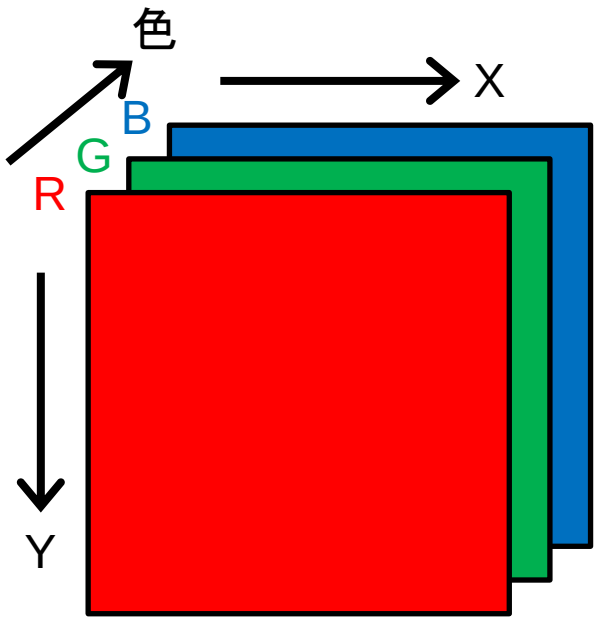

Eileen M. Collins†

※女性初のスペースシャトルパイロット
（画像はNASA Great Images databaseより引用）

# 画像データの中身とアクセス方法

```
from skimage import io
img = io.imread('https://bit.ly/39pAu0V')
```

img[Y座標，X座標，色チャンネル]




```
[1]  1  from skimage import io

[2]  1  img = io.imread('https://bit.ly/39pAu0V')

     1  io.imshow(img[:,:,0])
```

```
<matplotlib.image.AxesImage at 0x7f9aff4d1cf8>
```

# 色空間の変換

skimage.color を使ってみよう

# scikit-imageの色変換機能をインポート

- skimage モジュールの color をインポート

```
from skimage import color
```

- 様々な色空間をサポート
  - RGB
  - GRAY
  - LAB
  - HSV
  - 他

# RGBからグレースケールへの変換(1)

- color.rgb2gray 関数に画像を渡す

```
img = color.rgb2gray(入力画像)
```

# RGBからグレースケールへの変換(2)

```
[1]  1  from skimage import io
     2  img = io.imread('https://bit.ly/39pAu0V')
```

```
[3]  1  from skimage import color
     2  img = color.rgb2gray(img)
```

```
[4]  1  io.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f19ce13d860>
```

# 画像への図形の描画

skimage.draw を使ってみよう

# scikit-imageの矩形描画関数をインポート

- skimage モジュールの draw から
  `rectangle_perimeter` 関数をインポート

```
from skimage.draw import rectangle_perimeter as rect
```

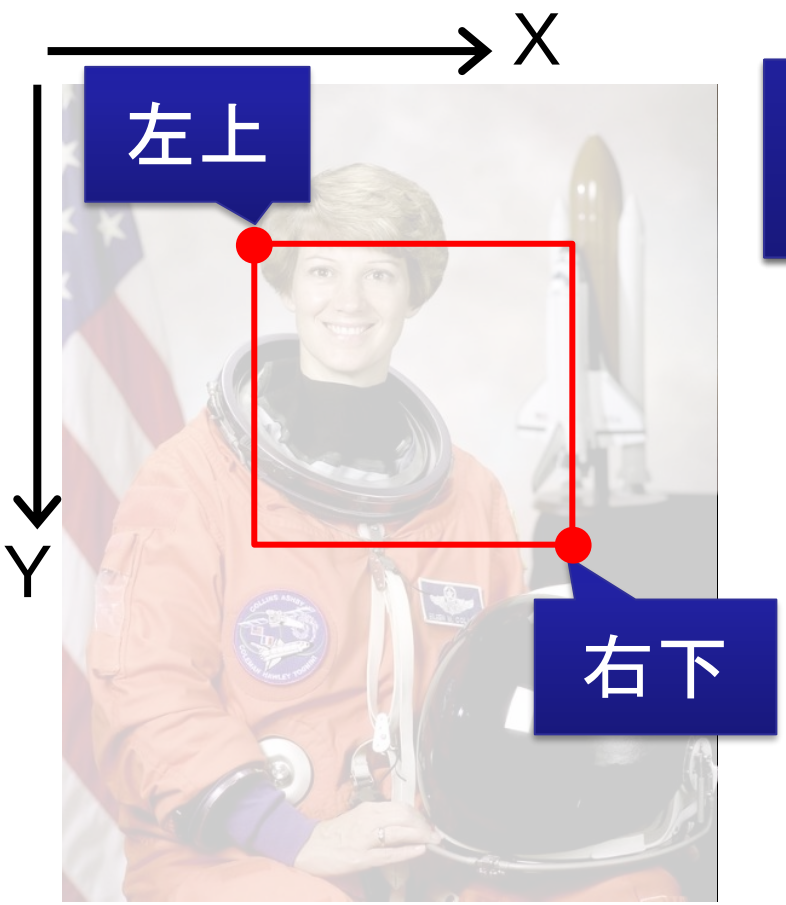
- 画像への図形の描画は2ステップ
  ① 図形を表す座標の計算
  ② 画像上への描画

図形を座標の集合として表現

# 矩形を描画(座標の計算)

- 左上と右下の座標から矩形を描画する座標を計算

```
rr,cc = rect(左上, 右下, shape=画像サイズ)
```




(y,x)のタプル形式で指定

画像のshapeを渡す

# 【補足】numpy 配列の要素アクセス

- 配列で指定したインデックスの要素だけ取得できる

```
配列の変数[[r1,r2,…], [c1,c2,…],…]
```

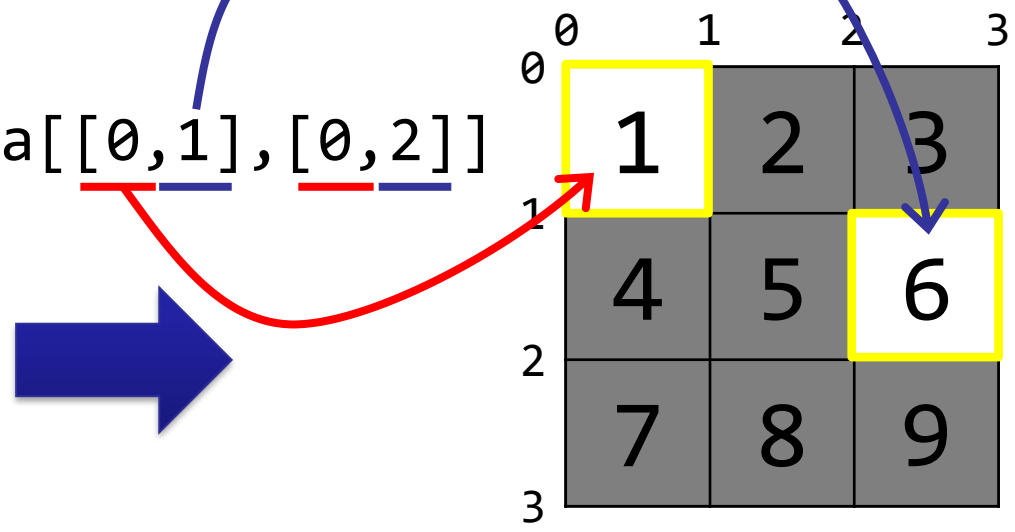
1次元目で取り出すインデックス

2次元目で取り出すインデックス

a =

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

```
a[[0,1],[0,2]]
```



# 矩形を描画（画像上への描画）

- 求めた座標で画像（Numpy配列）に色を設定

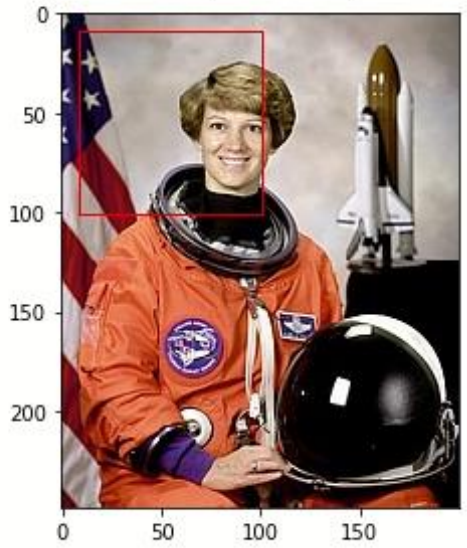
```
img[rr,cc] = (R,G,B)
```

RGBの順に色を指定
（RGBの値は0～255）

# 矩形を描画

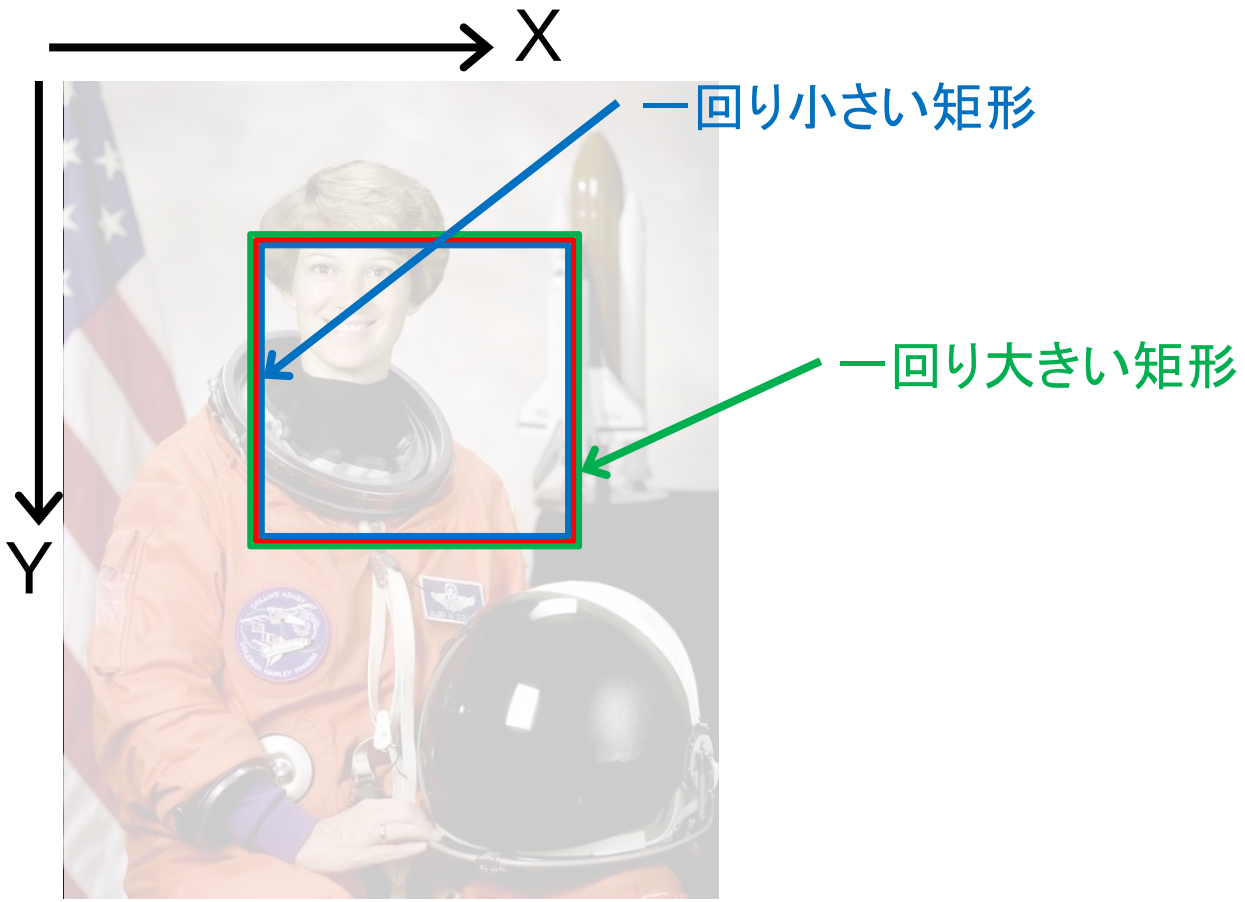
```
[1]  1  from skimage import io
     2  from skimage.draw import rectangle_perimeter as rect

[2]  1  img = io.imread('https://bit.ly/39pAu0V')

[4]  1  rr,cc = rect((10,10), (100,100), shape=img.shape)

[5]  1  img[rr,cc] = (255,0,0)

[6]  1  io.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f82ea2f2a90>
```



# 太さ3の矩形を描画

- 一回り大きい矩形と小さい矩形の座標を計算
- 3種類の大きさの矩形を同じ色で描画



# 太さ3の矩形を描画

```
[1]  1  from skimage import io
     2  from skimage.draw import rectangle_perimeter as rect

[2]  1  img = io.imread('https://bit.ly/39pAu0V')

[6]  1  rr,cc = rect((10,10),(100,100),shape=img.shape)
     2  rr1,cc1 = rect((10-1,10-1),(100+1,100+1),shape=img.shape)
     3  rr2,cc2 = rect((10+1,10+1),(100-1,100-1),shape=img.shape)

[7]  1  img[rr,cc] = (255,0,0)
     2  img[rr1,cc1] = (255,0,0)
     3  img[rr2,cc2] = (255,0,0)

[8]  1  io.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f0865f280d0>
```

# その他の図形の描画

- skimageのdrawを参照(http://bit.ly/38GcgjM)
  - line　　# 直線
  - circle　　# 塗りつぶした円
  - circle_perimeter　　# 円
  - polygon　　# 塗りつぶした多角形
  - polygon_perimeter　　# 多角形

# 画像のリサイズ

skimage.transform を使ってみよう

# scikit-imageのリサイズ機能をインポート

- skimage モジュールの transform から
  `resize` 関数をインポート

```
from skimage.transform import resize
```

- 任意のサイズに補間して画像をリサイズ可能

# 画像のリサイズ(1)

- 入力画像とリサイズ後の画像サイズを指定

```
img = resize(入力画像,出力サイズ)
```

リサイズ後の画像サイズを
(高さ,幅)のタプルで指定

入力画像


リサイズ後

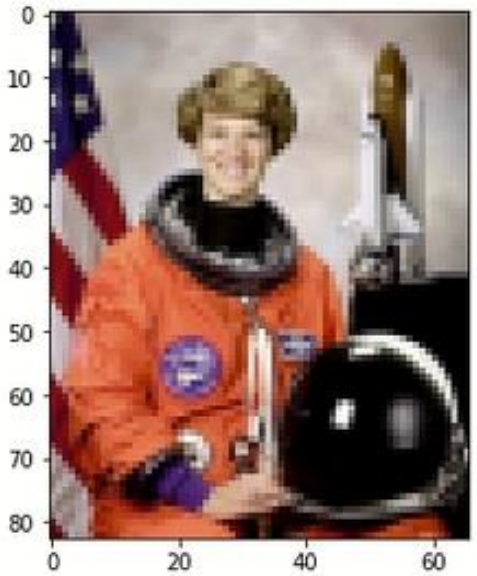
# 画像のリサイズ（2）

```
[1]  1  from skimage import io
     2  img = io.imread('https://bit.ly/39pAu0V')
```

```
[3]  1  from skimage.transform import resize
     2  h = img.shape[0]
     3  w = img.shape[1]
     4  img = resize(img, (h//3,w//3))
```

```
[4]  1  io.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f4130369dd8>
```

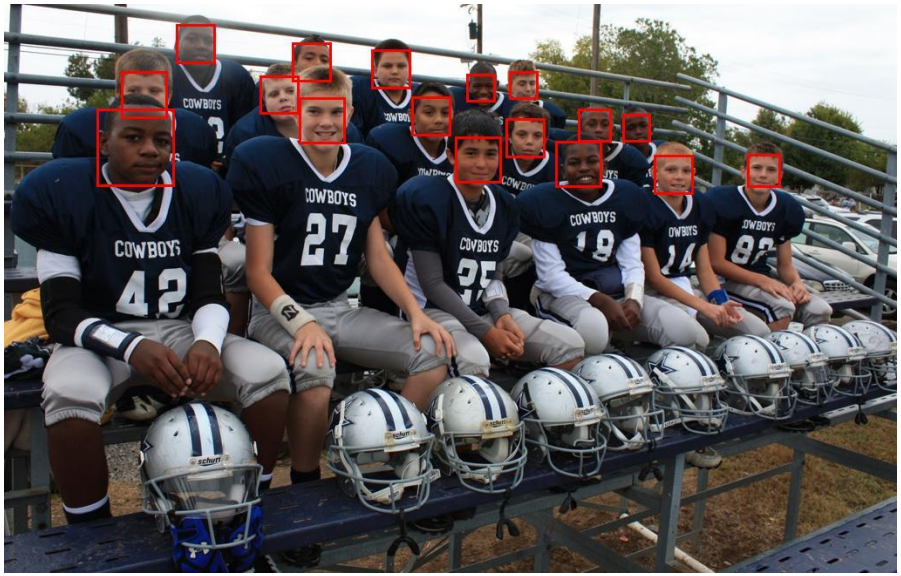


# 顔検出

dlib に触れてみよう

# Dlib（http://dlib.net/）とは？

- 基本的な画像処理から高度な物体検出手法，機械学習，など様々な技術を網羅したライブラリ
  - 顔検出，セグメンテーション，etc.
- 2002年に開発開始（C++で実装）
  - Pythonでも一部機能を利用することが可能




顔検出の例（原画像はWIDER Datasetより）

# Dlibのモジュールをインポート

- Dlibのモジュール名は dlib

- Dlibをインポート

```
import dlib
```

```
[1]  1  import dlib

[2]  1  dlib.__version__

     '19.18.0'
```

# 顔検出器を初期化

- get_frontal_face_detector 関数を呼び出して顔検出器を取得

```
detector = dlib.get_frontal_face_detector()
```

顔検出にはこれを使う

# 画像から顔を検出

- 取得した detector に入力画像と画像の拡大回数を指定する

```
dets = detector(入力画像,画像拡大回数)
```

検出された顔の矩形の配列

1以上の値を指定
（大きい値を指定すると小さい顔も検出できる）

# 顔検出器の初期化～顔検出まで

```
[1]  1  from skimage import io
     2  img = io.imread('https://bit.ly/39pAu0V')
```

```
[2]  1  io.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f6d6cb5c390>
```


```
[3]  1  import dlib
```

```
[4]  1  detector = dlib.get_frontal_face_detector()
```

```
[5]  1  dets = detector(img, 1)
```

```
[6]  1  dets
```

```
rectangles[[(69, 50) (105, 86)]]
```

# 検出された顔の矩形の詳細

dets = detector(入力画像,画像拡大回数)

- 検出結果 dets は検出された顔の数だけ矩形情報(dlib.rectangle)が格納された配列

```
for d in dets:
  l = d.left()    # 左上のX座標
  r = d.right()   # 右下のX座標
  t = d.top()     # 左上のY座標
  b = d.bottom()  # 右下のY座標
  print('({0},{1})-({2},{3})'.format(t,l,b,r))
```

# 顔検出の結果を描画

[1]
```
from skimage import io
from skimage.draw import rectangle_perimeter as rect
import dlib
```

[2]
```
img = io.imread('https://bit.ly/39pAu0V')
```

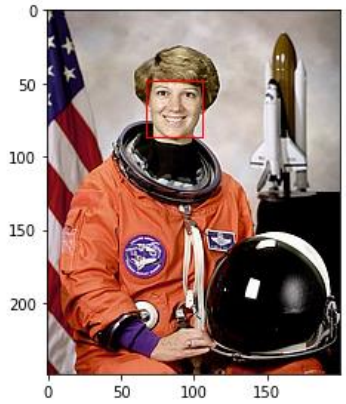
[3]
```
detector = dlib.get_frontal_face_detector()
```

[4]
```
dets = detector(img, 1)
```

[5]
```
for d in dets:
    l = d.left()
    r = d.right()
    t = d.top()
    b = d.bottom()
    rr,cc = rect((t,l), (b,r), shape=img.shape)
    img[rr,cc] = (255,0,0)
```

[6]
```
io.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f001c3256d8>
```



# 課題

- 画像からの顔検出にチャレンジしてみよう

- 下記URLから演習課題のノートブックを自身のGoogle Driveにコピーし，Google Collaboratoryを起動して各設問に回答すること
  – https://bit.ly/3yaWfgc

- 演習課題を提出する際は，ノートブックのURLで共有し，そのURLを提出すること

# draw_mask 関数の使い方

`img = draw_mask(入力画像,マスク画像,貼り付け位置)`

```
[1]  1  from skimage import io
     2  from skimage.transform import resize
     3  import dlib

     1  mask_img=io.imread('https://bit.ly/39rDGcm')
     2  io.imshow(mask_img)

[4]  1  img = io.imread('https://bit.ly/39pAuOV')
     2  mask = resize(mask_img, (36,36))
     3  eimg = draw_mask(img, mask, (75,87))
     4  io.imshow(eimg)

<matplotlib.image.AxesImage at 0x7f29375cf1d0>
```


マスクの貼り付け位置の中心座標を(y,x)のタプルで指定

# 課題提出時の注意点

- 課題提出の際には提出前に必ず以下2点を確認すること
  - Google Collaboratoryの共有設定の際には『リンクを知っている全員』にチェックを入れる
  - 課題提出時に貼り付けたURLの末尾が「?usp=sharing」となっている（共有設定を開き、「リンクのコピー」ボタンを押して共有用のURLをコピー＆ペーストする）