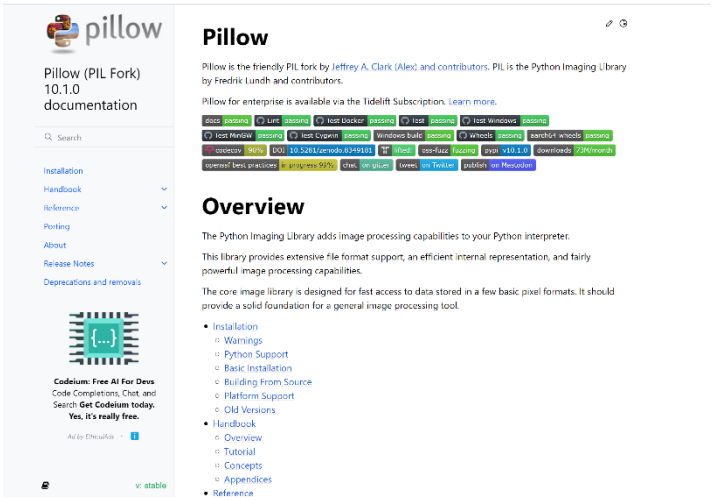
pillow によるイメージ処理

一般に、イメージ処理は「レタッチ」ソフトが使われるが、一定の処理を繰り返していくのはプログラミング言語の方が向いている。

pillow は、これまでよく使われてきた「PIL」（Python Image Library）の開発が止まったため、PIL からフォークされて開発されたルール。

現在の Anaconda には pillow はすでに組み込まれている。



■画像表示の基本

```
from PIL import Image
img = Image.open('rose.jpg')
img
```

すべては、Image クラスの派生クラスとして設計されている。

◆ファイル保存とフォーマット変換

表示した画像をそのまま保存することができる。
また、同時に、フォーマットを変換することもできる。
変換できるフォーマットは、
「JPEG」、「PNG」、「TIFF」、「GIF」

「rose.jpg」を開き、「image」と言う名前で「png」形式で保存する

```
img = Image.open('rose.jpg')
img.save('image.png')
```

・JPEG

quality：画質（圧縮率）1 から 95 の整数。デフォルトは 75
progressive：True でプログレッシブ JPEG として保存
dpi：ピクセル密度の指定。(x,y) と 2 つの値をタプルで指定

PNG：

transparency：透過色の指定
compress_level：圧縮レベル 0〜9 の整数
dpi：ピクセル密度の指定。(x,y) と 2 つの値をタプルで指定

TIFF：

save_all：True で全てのフレームを保存
compression：圧縮モードを指定「tiff_jpg」、「iff_raw_16」、「None」などなど

GIF：

save_all：True で全てのフレームを保存
loop：繰り返す回数を指定
duration：各フレームの表示間隔をミリ秒で指定

■エラー処理

ファイルが存在しないなどのエラーに対する処理（「例外処理」）をする必要がある。
エラーが発生したときには中断し、問題がなければ処理を継続する。

```
＃2

import sys
from PIL import Image

img;
try:
    img = Image.open('image.jpg')
except IOError:
    print("Can't Open File!")
    sys.exit()

img
```

◆サイズ変更

縦横の指定はピクセル単位をタプルで指定。

文法：
変数 ＝ [Image].resize（(横、縦)）

縦横に対する「比率の変更」も resize を使う

```
#3
from PIL import Image

img = Image.open('rose.jpg')
(w, h) = img.size

img2 = img.resize((w, h // 2), resample=Image.NEAREST)
img2.show()
```

拡大・縮小するときのリサンプリング方法を resample オプションで指定する。

次のような値が用意されている。resample を省略した場合は、Image.NEAREST でリサンプリングされる。

NEAREST、BOX、BILINEAR、HAMMING、BICUBIC、LANCZOS

◆リサンプル

拡大縮小に対してのリサンプリングは「resample」を使う。
つまり、サイズ変更は、元の画像を変更はしていないことになる。

縦横を 1/2 にするサンプル

```
from PIL import Image
img = Image.open('rose.jpg')
(w, h) = img.size
img2 = img.resize((w // 2, h // 2))
img2.show()
```

※プログラムを完動させるために表現方法を変えたものもある

◆イメージの回転

反時計回りに回転させる

文法：

変数 ＝ [Image].rotate（角度）

元の画像を反時計回りに 90 度傾けるサンプル

```
img = Image.open('rose.jpg')
img2 = img.rotate(90, expand=True)
img2
```

※以降、「from PIL import Image」は省略

回転では、回転後の描画で枠内に収まる。そのため、縮小処理が行われる。
この処理をせず、元の画像のまま回転処理を実行するためには、

expand = True

の指定が必要。ただし、はみ出した部分は描かれない。

◆イメージの反転

文法：
変数 ＝ [Image].transpose（値）

FLIP_LEFT_RIGHT	：左右反転
FLIP_TOP_BOTTOM	：上下反転
ROTATE_90	：90 度回転
ROTATE_180	：180 度回転
ROTATE_270	：270 度回転

```
img = Image.open('rose.jpg')
img2 = img.transpose(Image.FLIP_TOP_BOTTOM)
img2
```

・rotate と transpose との違い

rotate では、黒いエリアが表示される。はみ出した部分はカットされる。
transpose の場合には、全体を描画するため縦横比が狂う

◆RGB、グレースケースの変換

convert の「値」に、RGB ならカラー、L ならモノクロになる

文法：

```
変数 ＝ [Image].convert（値）
```

写真をグレースケールに変換

```
img = Image.open('rose.jpg')
img2 = img.convert('L')
img2
```

一度、グレースケールに変換した画像のカラー情報は失われる。したがって、その後、カラーに変換できない。

■イメージ処理

◆フィルター

pillow にはいくつものイメージ処理のための「filter」が用意されている。

文法：

```
[Image].filter（フィルター）
```

filter モジュールを利用する際には、「PIL.ImageFilter」が用意されているので、import 文を使い読み込む

文法：

```
from PIL import ImageFilter
```

◆ブラー処理

ブラーには、3 段階の強度がある。

弱：ImageFilter.SMOOTH

中：ImageFilter.SMOOTH_MORE

強：ImageFilter.BLUR

```
from PIL import Image
from PIL import ImageFilter
img = Image.open('rose.jpg')
img2= img.filter(ImageFilter.BLUR)
img2
```

強であるはずの BLUR でも、それほど強くはない

◆ブラーの強度を値で変えられる GaussianBlur

GaussianBlur（値）の値を変えれば、ボケ感が変わる。

かなり強いブラー

```
img = Image.open('rose.jpg')
img2= img.filter(ImageFilter.GaussianBlur(10.0))
img2
```

※from の 2 行は省略

他にも、「BoxBulr」などがある。

BoxBlur は、その名の示すとおり、ある一定の四角ごとにぼかす。

◆アンシャープマスク

ブラーとは逆に、輪郭を際立たせる方法

強度によって 3 段階ある

弱：ImageFilter.SHARPEN

中：ImageFilter.EDGE_ENHANCE

強：ImageFilter.EDGE_ENHANCE_MORE

```
img = Image.open('rose.jpg')
img2= img.filter(ImageFilter.EDGE_ENHANCE)
img2
```

アンシャープには、ほかにも「UnsharpMask」クラスもある。

◆輝度調整

文法：

変数＝[Image].point(リスト または 変数)

ラムダ式を使い全ての値を 2 倍にしている

```
img = Image.open('rose.jpg')
img2=img.point(lambda x: x*2)
img2
```

◆リストによる輝度変換

たとえば、256×3 個のリストを用意するなどの処理は Python は得意。

次の例では、RGB の R と B は輝度を逆転して、G は全ての輝度が最大になるように変換されている。したがって、色は RGB の組み合わせであるので、結果、輝度だけではなく色も変化している。

```
from PIL import Image
from PIL import ImageFilter

img = Image.open('rose.jpg')
(w, h) = img.size
img2 = img.point(list(range(256, 0, -1)) + list([255] * 256) + list(range(256, 0, -1)))
img2.show()
```

◆イメージを重ねる

文法：

```
[Image].paste([Image], 位置)
```

```
from PIL import Image

img = Image.open('rose.jpg')
(w, h) = img.size
img2 = img.resize((w // 3, h // 3))
img.paste(img2, (0, 0))
img.paste(img2, (w // 3, h // 3))
img.paste(img2, (w // 3 * 2, h // 3 * 2))
```