Lab 2 : Image Processing in Python with Pillow

A crucial class in the Python Imaging Library is the Image class

```
from PIL import Image

image = Image.open('demo_image.jpg')
image.show()

# The file format of the source file.
print(image.format) # Output: JPEG

# The pixel format used by the image. Typical values are "1", "L", "RGB", or "CMYK."
print(image.mode) # Output: RGB

# Image size, in pixels. The size is given as a 2-tuple (width, height).
print(image.size) # Output: (1920, 1280)

# Colour palette table, if any.
print(image.palette) # Output: None
```

Resizing Images To resize an image, you call the resize() method on it, passing in a two-integer tuple argument representing the width and height of the resized image. The function doesn't modify the used image; it instead returns another Image with the new dimensions.

```
# Changing the image type

image = Image.open('demo_image.jpg')
image.save('new_image.png')
```

```
[3] image = Image.open('demo_image.jpg')
    new_image = image.resize((400, 400))
    new_image.save('image_400.jpg')

    print(image.size) # Output: (1920, 1280)
    print(new_image.size) # Output: (400, 400)
```

The resize() method has two drawbacks, however:

Oftentimes, resizing to an exact width and height changes the image's aspect ratio, leading to distortions. If you set the size of the new instance to be larger than that of the original, resize() "blows up" the instance, reducing its quality.

You can see this in the newly created image from the above code: image_400.jpg. It looks a bit squished horizontally. As a solution, resize the image with the more advanced Pillow method, thumbnail(): Perform steps 1 and 2 of the above procedure. Call the thumbnail() method on the Image instance, passing a tuple argument with two integers to specify the width and height you desire:

```
image = Image.open('demo_image.jpg')
image.thumbnail((400, 400))
image.save('image_thumbnail.jpg')

print(image.size) # Output: (400, 267)
```

The above will result in an image sized 400x267, having kept the aspect ratio of the original image. As you can see, this results in a better-looking image.

IMAGE CROPPING : When an image is cropped, a rectangular region inside the image is selected and retained while everything else outside the region is removed. With the Pillow library, you can crop an image with the crop() method of the Image class. The method takes a box tuple that defines the position and size of the cropped region and returns an Image object representing the cropped image. The coordinates for the box are (left, upper, right, lower). The cropped section includes the left column and the upper row of pixels and goes up to (but doesn't include) the right column and bottom row of pixels. This is better explained with an example.

Pasting an Image onto Another Image

```
image = Image.open('demo_image.jpg')
box = (200, 300, 700, 600)
cropped_image = image.crop(box)
cropped_image.save('cropped_image.jpg')

# Print size of cropped image
print(cropped_image.size) # Output: (500, 300)
```

```
(500, 300)
```

```
[7] #Pasting an Image onto Another Image
image = Image.open('demo_image.jpg')
logo = Image.open('logo.png')
image_copy = image.copy()
position = ((image_copy.width - logo.width), (image_copy.height - logo.height))
image_copy.paste(logo, position)
image_copy.save('pasted_image.jpg')
```

```
[9] #Rotation
image = Image.open('demo_image.jpg')
```

By default, the rotated image keeps the dimensions of the original image. This means that for angles other than multiples of 180, the image will be cut and/or padded to fit the original dimensions. If you look closely at the first image above, you'll notice that some of it has been cut to fit the original height, and its sides have been padded with black background (transparent pixels on some OSs) to fit the original width. The example below shows this more clearly.

```
image.rotate(18).save('image_rot_18.jpg')
```

```
[17] #Now the contents of the image will be fully visible,
     #and the dimensions of the image will have increased to account for this.
     image.rotate(18, expand=True).save('image_rot_18.jpg')
```

```
[19] #Flipping Images
     image = Image.open('demo_image.jpg')

     image_flip = image.transpose(Image.FLIP_LEFT_RIGHT)
     image_flip.save('image_flip.jpg')
```