As technology emerges day by day, digital images can be an indispensable source of receiving data. We encounter a lot of digital images in daily life. However, these pictures can be related to anything.

In the programming world, we can process digital images using various libraries or tools. In this article, we will learn about one of the popular tools of Python is **Pillow**. However, Python provides many other useful libraries such as OpenCV, Python Image Library (PIL), and **Scikit-image**. This article completely focuses on the **Python Pillow module (PIL)**.

What is Image Processing?

Image processing is a method of analyzing and manipulating digital images. The main goal is to extracting some information that can be used somewhere else. Let's make it simple to understand with a real-life example.

For example - Nowadays, CCTV cameras are instated in the traffic light that capture digital images, identify the vehicle's number plate, and check whether the vehicle violated the traffic rules or not. All this process is related to image processing.

Introduction

Python Pillow module is built on top of PIL (Python Image Library). It is the essential modules for image processing in Python. But it is not supported by Python 3. But, we can use this module with the Python 3.x versions as PIL. It supports the variability of images such as jpeg, png, bmp, gif, ppm, and tiff.

We can do anything on the digital images using the pillow module. In the upcoming section, we will learn various operations on the images such as filtering images, Creating thumbnail, merging images, cropping images, blur an image, resizing an image, creating a water mark and many other operations.

Installation of Pillow

Before start working with Python, we need to install the pillow library to our local machine. We can do it by typing the following command in the terminal.

```
pip install pillow
or
pip install PIL
```

If pip is already installed in the system, it will simply show the **"requirement already satisfied"**.

Using the Image Module

Python pillow library is used to image class within it to show the image. The image modules that belong to the pillow package have a few inbuilt functions such as load images or create new images, etc.

Opening, Rotating and Displaying an Image

Opening an image is a basic operation of the image processing. We import the image module from the PIL library to load the image. It provides the **Image.open()** method, which takes an image filename as an argument. This method returns the image object.

We can make any modification to image object which can be saved to an image with the **save()** method. Once the image is open, we can perform operations as resize, crop, draw or many others.

Example -

```
from PIL import Image
#Open image using Image module
im = Image.open(r"C:\Users\DEVANSH SHARMA\Downloads\car.jpg")
#Show actual Image
im.show()
#Show rotated Image
imim = im.rotate(45)
im.show()
```

Output:

Image after Rotation -

Explanation

The above code displayed the original and rotated image using the standard jpeg display utility. If we run the above code in the **Jupyter notebook** then the output will be shown in the local machine image gallery.

Attributes of Image Module

The Image class contains some attributes to manipulate the image object. Let's understand these attributes using one by one.

Image.filemname

The Image.filename function returns the name of the image.

```
image = Image.open('beach1.jpg')
image.filename
```

Output:

```
'C:\\Users\\DEVANSH SHARMA\\Downloads\\car.jpg'
```

Image.format

This function is used to get file format of the image file such as 'JPEG', 'BMP', 'PNG', etc.

```
image = Image.open('beach1.jpg')
image.format
```

Output:

```
'JPEG'
```

Image.mode

The above method returns the pixel format used by the image. Typing variables are '1', 'L', 'RGB', or 'CMYK'.

```
im = Image.open(r"C:\Users\DEVANSH SHARMA\Downloads\car.jpg")
im.mode
```

Output:

```
'RGB'
```

Image.size

The Image.size function returns the tuple that contains height & weight of the image.

```
im = Image.open(r"C:\Users\DEVANSH SHARMA\Downloads\car.jpg")
im.size
```

Output:

```
(480, 320)
```

Image.width

This method returns the width of the image.

```
im = Image.open(r"C:\Users\DEVANSH SHARMA\Downloads\car.jpg")
im.width
```

Output:

```
480
```

Image.height

This method returns the width of the image.

```
im = Image.open(r"C:\Users\DEVANSH SHARMA\Downloads\car.jpg")
im.width
```

Output:

Image.info

This method returns a dictionary holding data associated with the image.

```
im = Image.open(r"C:\Users\DEVANSH SHARMA\Downloads\car.jpg")
im.info
```

Output:

```
{'jfif': 257,
 'jfif_version': (1, 1),
 'jfif_unit': 0,
 'jfif_density': (1, 1),
 'progressive': 1,
 'progression': 1}
```

Working with Images

Python pillow module makes very simple to work with the images. We can use **PIL.Image modle** function. Let's see the syntax.

Syntax -

1. Image.open(fp, mode = 'r')

In the above syntax -

- **fp** - It is a file object or represents the file name, **Path** object. This file object can implement the **read(), seek(),** and **tell()** methods.
- **mode** - It represents a file mode. It is an optional argument if given, by default is 'r'.

Consider the following example.

Example -

In the following example, we will open an image in .jpg format. First, we will display the image and then save it with another file format.

```
from PIL import Image
#We are Opening image using Image module
img = Image.open("images/cuba.jpg")
#It will display the actual image
```

```
imimg = img.rotate(45)
img.show()
```

The above code will display the actual and rotate image at 45 degree.

Explanation -

In the above code, we have imported the Image module from **PIL library**, called **Image.open()** function to read an image. The **Image.open()** function returns an image object. This method takes a **filename(string)**, a path object or an **image(file)** object.

Saving an Image

The Image module provides the **save()** method to save the image to file. It takes a filename, a file object that has been opened to write. Let's understand the following syntax.

Syntax &ndash

```
Image.save(fp, format = None, **params)
```

In the above syntax -

- **fp -** It represents a filename (string), **Path** object or file object.
- **format -** It is an optional format override. It is used when a file object was used in its place of a filename.
- **options -** It is an additional parameter to the image writer.
- **Return value -** The return value is
- **KeyError -** It is used when the output format could not be determined from the file name.
- **IOError -** It is used when the file may have been formed and may have fractional data.

If we sum up the above syntax, the file saves based on the given filename. If we don't specify the format, it is based on the current filename extension.

```
image.save('dev.png')
```

In the above example, it keeps the file based on the file extension the image type. **For example -** The code will save as a **png** file in our current working directory.

We can also obviously identify the file type as a second parameter. Let's see the following line.

```
image.save('dev.gif', 'GIF')
```

Creating Thumbnails

Thumbnail is images of equal height and width. Sometimes, we need to create a thumbnail to specify the actual image. The pillow library provides **thumbnail()** function. This method transforms the actual image to contain a thumbnail version of it. The size of the thumbnail will not be same as the given size.

The **thumbnail()** estimates a suitable thumbnail size to preserve the aspect of the image. We need to call another method **draft()** method, configuring the file reader and, finally, resizing the image. Let's see the following syntax.

Syntax -

```
Image.thumbnail(size, resample=3)
```

The above syntax -

- **Size -** It is the defined size of a thumbnail.
- **Resample -** It is an optional resampling filter. This filter can be one of these **Image.NEAREST**, BILNEAR, PIL.Image.BICUBIC, or PIL.Image.LANCZOS. By default, it is PIL.Image.BICUBIC.
- **Return -** It returns **None**.

Let's understand the following example.

Example -

```
from PIL import Image
def thumbs():
  try:
    img = Image.open(r"C:\Users\DEVANSH SHARMA\Downloads\car.jpg")
     img.thumbnail((80,80))
    img.save('images/thumbnail.jpg')
     image1 = Image.open('images/thumbnail.jpg')
    image1.show()
  except IOError:
    pass
thumb()
```

Output:

Explanation -

In the above code, we have defined the **thumbs()** function that contains the try-expect to handle any occurrence. In the try block, we have opened an image which we want to modify as thumbnail.

Then, we have called the **thumbnail()** function to the image object and we have passed the thumbnail size. Somehow, if the try block would through an error, it will jump to **except** block where we defined the **IOError** exception.

Merging Images Using Pillow

We can paste an image onto another one. Pillow library provides **merge()** function which takes a mode and tuple of the images parameters. Then, it associates them with a single image. Let's see the following syntax.

Syntax -

```
Image.merge(mode, bands)
```

In the above syntax -

- **mode -** It is used for the output image.
- **bands -** It is an order that contains one single-band image for each band in the output.
- **Return value -** It returns an image object.

Let's understand the following example.

Example -

```
from PIL import Image
image = Image.open(r"C:\Users\DEVANSH SHARMA\Downloads\car.jpg")
r, g, b = image.split()
image.show()
image = Image.merge("RGB", (b, g, r))
image.show()
```

Output:

When we execute the above code, we can observe the difference between original image and the image with merge the RGB bands.

Merging Two Images

Merging two images is the same as above. First, it requires to create an image object for the essential image using the **open()** function. One point should be remembered by developer; both images must be the same size before merging them. Then we create empty images using the **Image.new()** function, we paste the images using the **paste()** function.

After that, we save and display the resultant image using the **save()** and **show()** functions. Let's understand the following example.

Example -

```
from PIL import Image
#Read the two images
image1 = Image.open(r"C:\Users\DEVANSH SHARMA\Downloads\car.jpg")

# # image1.show()
image2 = Image.open(r"C:\Users\DEVANSH SHARMA\Downloads\staus.jpg")

image2.show()
#we must resize, first image before merging them
image1image1 = image1.resize((266, 190))
image1image1_size = image1.size
image2image2_size = image2.size
new_image = Image.new('RGB',(2*image1_size[0], image1_size[1]), (250,250,250))

new_image.paste(image1,(0,0))
new_image.paste(image2,(image1_size[0],0))

new_image.show()
```

Output:

Explanation -

In the above code, we have imported the Image module from the PIL library. We opened two images to merge them with each other. Before perform the merge operation, we need

to make sure both images should be the same size. We made the size of the first image as the second image. Then, we used the **paste()** method to merge them.

Blur an Image Using Pillow

Blurring an image is a method of decreasing the level of noise in the image by applying a filter to an image. It is the most important aspect of image processing. The pillow library offers the **ImageFilter** class that contains several standard image filters.

In this section, we will discuss various techniques of blurring images. These techniques are mentioned below.

- Simple Blur
- Box Blur
- Gaussian Blur

The **Image.filter()** method will use the techniques mentioned above to apply the filters' images.

Simple Blur

In the simple blur, the blurring effect is applied to the image as specified through a specific kernel. Let's see the following syntax.

Syntax -

```
filter(ImageFilter.BLUR)
```

Let's understand the following example.

Example -

```
from PIL import Image, ImageFilter
#Open image
Original_Image = Image.open(r"C:\Users\DEVANSH SHARMA\Downloads\car.jpg")
Original_Image.show()
# Applying simple blur function
blur_Image = OriImage.filter(ImageFilter.BLUR)
blur_Image.show()
```

Output:

Original Image -

Blurred Image -

We have opened two standard images to perform blur operation. We can see the difference between the original image and the blurred image.

Box Blur

The Box Blur filter uses the **'radius'** as a parameter. It is directly proportional to the blur value.

Syntax -

```
ImageFilter.BoxBlur(radius)
```

In the above syntax -

- **Radius -** It represents the box in one direction.
- **Radius 0 -** It means no blur operation perform and returns the same image.
- **RRadius 1 & minnus** - It takes total of 9 pixels, one pixels in each direction.

Let's understand the following example.

Example -

Original Image -

Box Blurred Image -

Gaussian Blur

This filter works the same as box blur but slight changes in its algorithm. It also takes the radius parameter. We can generate a different intensity of **Gaussianblur** image by changing the radius value.

Syntax -

```
ImageFilter.GaussianBlur(radius=2)
```

Let's understand the following example.

Example -

```
#Import Image and ImageFiler module
from PIL import Image, ImageFilter

#Open image
Original_Image = Image.open(r"C:\Users\DEVANSH SHARMA\Downloads\car.jpg")
Original_Image.show()

# Applying gaussina blur function
blur_Image = Original_Image.filter(ImageFilter.GaussianBlur(6))
blur_Image.show()
```

Output:

Original Image -

Gaussian Blurred Image -

Creating a Watermark using Pillow

We all may see a tiny text on the bottom of the online photos which is called watermark. It is an excellent way to protect images and prevent their misuse. Watermark does not only protect from misuse but also provide the copyright to our creative photos. So, it is preferred to add a watermark to the image, before sharing to the social media.

It is kind of text or logo overlaid on the photo that have a small identity of the owner or who owns the rights to the photo.

With the Python Pillow library, we can add the watermark to the image very easily. This library provides the Image, ImageDraw and ImageFont module.

Let's understand the following example.

Example -

```
#Import all three modules from Image library
from PIL import Image, ImageDraw, ImageFont

#Creating an Image Object
img = Image.open(r'C:\Users\DEVANSH SHARMA\Downloads\car.jpg')
width, height = img.size

draw_1 = ImageDraw.Draw(img)
# passing sample watermark text
text_1 = "JavaTpoint.com"

# Specifyig the font familiy and font size
font = ImageFont.truetype('arial.ttf', 36)
textwidth, textheight = draw.textsize(text_1, font)

# calculate the x,y coordinates of the text
margin = 10
x = width - textwidth - margin
y = height - textheight - margin

# draw watermark in the bottom right corner
draw_1.text((x, y), text, fontfont=font)
im.show()
```

Explanation -

In the above code, we have imported the Image, **ImageDraw** and **ImageFont** modules from the pillow library where the **ImageDraw** module enhances the functionality to draw 2D graphics to the image.

The ImageFont module is responsible for adding font functionality while writing watermark. We created an image object where we will apply the watermark. The image

object passed to the **Draw()** method; we wrote the watermark and saved it to the text variable.

We can also set the font-family and font-size as we have set 'Times New Roman' and font-size 25. Then, we calculated the x, y coordinate to specify the text position in the image. Once we calculated the coordinate, we drew the watermark in the bottom right corner.

Apply Color to the Image

Pillow library provides the **ImageColor** module that contains colors in the several formats. It also has the converter from CSS3-style color specifiers to RGB tuples. Let's have a look at the color names which are supported by the ImageColor module.

The first format is Hexadecimal color specifiers, specified as **#rgb** or **#rrggbb**. For example, #00ff00 denotes pure green.

The second format is #00ff00 hex color, red value is 0 (0% red), green value is 255(100% green) and the blue value of its RGB is 0 (0% blue).

There are 140 standard color names in the **ImageColor** module. Most of the colors are supported by the Window system and most web browsers.

ImageColor.getrgb() Method

This method is used to convert a color string to an RGB tuple and it raises a **ValueError** exception in case of string cannot be parsed. Let's see the following syntax.

Syntax -

```
PIL.ImageColor.getrgb(color)
```

In the above syntax -

- color - It is a color string.
- Return Value - It returns RGB (red, green, blue [,alpha]).

Let's understand the following example.

Example -

```
# Importig ImageColor module
```

```
from PIL import ImageColor

# using getrgb method
img = ImageColor.getrgb("blue")
print(img)

image_1 = ImageColor.getrgb("red")
print(image_1)
```

Output:

```
(0, 0, 255)
(255, 0, 0)
```

As we can see in the above code, the getrgb() method taken the color name and returned the RGB values.

Example - 2

```
#Import necessary image modules
from PIL import Image,ImageColor

# Create new image & get color RGB tuple.
img = Image.new("RGB", (256, 256), ImageColor.getrgb("#add8e6"))

#Show image
img.show()
```

ImageDraw Module

The ImageDraw module is a very important component of the Pillow library. We can provide some simple 2D graphics support for image object.

This module helps create new images, provides some additional touches to the image, and creates some graphics. It also supports the annotation of text and drawing of shapes.

ImageDraw module uses the origin of the two-dimension co-ordinate which is in the upper left corner of the image.

It supports the various fonts for the text annotations such as bitmap, OpenType, or TrueType.

Let's understand the following example.

Example -

```
from PIL import Image, ImageDraw, ImageFont

# get an image
img = Image.open('C:').convert('RGBA')

# making a blank image for the text, initialized to transparent text color
text1 = Image.new('RGBA', img.size, (255,255,255,0))

# getting a font from the saved library
fnt = ImageFont.truetype('E:/PythonPillow/Fonts/Pacifico.ttf', 40)
d = ImageDraw.Draw(text1)

# draw text with the half opacity
d.text((14,14), "Tutorials", font=fnt, fill=(255,255,255,128))

# draw text with the  full opacity
d.text((14,60), "Point", font=fnt, fill=(255,255,255,255))
out = Image.alpha_composite(img, text1)

out.show()
```

Output:

Image Sequence in Pillow

Image Sequence is equivalent to the animation formats support by the Python PIL library. FLI/FLC, GIF and some new formats are the maintained by sequence formats. TIFF files can contain more than one frame as well.

PIL uploads the image in the sequence like first frame is loads according to the image sequence. We can use seek() and tell() methods to move the different frames. Let's understand the following example.

Example -

```
from PIL import Image
img = Image.open(r'C:\Users\DEVANSH SHARMA\Downloads\car.jpg')
#Here are Skipping to the second frame
```

```
img.seek(1)
try:
  while 1:
    img.seek(img.tell() + 1)

except EOFError:
  #End of sequence
  pass
```

Output:

```
raise EOFError
EOFError
```

Explanation -

In the above code, we can see that the EOFError exception is thrown when the sequence ends.

To overcome that situation, we can create a sequence iterator class.

A sequence iterator class

```
class ImageSequence:
  def __init__(self, img1):
    self.img1 = img1
  def __getitem__(self, xi):
    try:
      if xi:
        self.img1.seek(xi)
      return self.img1
    except EOFError:
      raise IndexError # end of sequence
for frame in ImageSequence(img):
```

Writing Text on the Image Using Pillow

This is different from the creating a watermark. We can write text to the image by passing the location where we want to write text on the image. We can pass the multiple parameters to text method. Let's understand the following example.

Example -

```
from PIL import Image, ImageDraw
#Creating an Image Object
img = Image.open(r'C:\Users\DEVANSH SHARMA\Downloads\car.jpg')
d1 = ImageDraw.Draw(img)
d1.text((32, 44), "Hello, CSE 3RD SEM!", fill=(255, 255, 0))
img.show()
```

Flip and Rotate Image Using Pillow

Sometimes, we need to perform a few basic operations such as rotating and flipping the image. These operations are helpful to get the insight form the existing image, it also enhance the visibility of the image.

Pillow library allows us to flip and rotate an image very easily. We will use the transpose (method) function from the image module to perform the flip operation. Some of the most common methods are given below.

- **FLIP_LEFT_RIGHT** - It is used for flipping the image horizontally.
- **FLIP_TOP_BOTTOM** - It is used to flipping the image vertically.
- **ROTATE_90** - It is used to rotate the image by specifying degree.

Let's understand the example how to flip an image horizontally.

Example -

```
# import  image module
from PIL import Image

# Open an already existing image and create image object
imageObject = Image.open(r'C:\Users\DEVANSH SHARMA\Downloads\car.jpg')


# perform a flip of left and right
hori_flippedImage = imageObject.transpose(Image.FLIP_LEFT_RIGHT)

# display the original image
imageObject.show()

# display the horizontal flipped image
hori_flippedImage.show()
```

Output:

Original Image

Flipped Image -

Explanation -

In the above code, we have read the image and flipped it horizontally. Both images original and flipped displayed image using standard JPG display utility.

Example - 2 Vertically Flipped Image

```
# import  image module
from PIL import Image

# Open an already existing image and create image object
imageObject = Image.open(r'C:\Users\DEVANSH SHARMA\Downloads\car.jpg')
# perform a flip of left and right
hori_flippedImage = imageObject.transpose(Image.FLIP_LEFT_RIGHT)

# display the original image
imageObject.show()

# display the horizontal flipped image
hori_flippedImage.show()
```

Output:

Original Image -

Flipped Image -

Let's understand another example of rotation

Example - 3

```
# import  image module
from PIL import Image

# Open an already existing image and create image object
imageObject = Image.open(r'C:\Users\DEVANSH SHARMA\Downloads\car.jpg')

# perform a flip of left and right
horizonFlippedImage = imageObject.transpose(Image.FLIP_LEFT_RIGHT)
imageObject.show()

horizonFlippedImage.show()
```

Output:

Original Image -

Rotated Image -