**Pillow**

*Pillow* is a Python Imaging Library (PIL), which adds support for opening, manipulating, and saving images. The current version identifies and reads a large number of formats. Write support is intentionally restricted to the most commonly used interchange and presentation formats.

**Showing image**

In the first example we read an image file and show it in an external program.

show\_image.py

#!/usr/bin/python3

from PIL import Image

import sys

try:

tatras = Image.open("tatras.jpg")

except IOError:

print("Unable to load image")

sys.exit(1)

tatras.show()

The program reads a JPG image and displays it in an external application.

from PIL import Image

From the PIL module, we include the Image class.

tatras = Image.open("tatras.jpg")

The Image.open() method reads the image file. Pillow can read over 30 different file formats.

tatras.show()

The show() method is mainly intended for debugging purposes. It saves the image into a temporary file and displays it in external program. This could be ImageMagic on Linux or Paint on Windows.

**Basic image information with Pillow**

Pillow allows us to get some basic information about the image.

basic\_info.py

#!/usr/bin/python3

from PIL import Image

import sys

try:

tatras = Image.open("tatras.jpg")

except IOError:

print("Unable to load image")

sys.exit(1)

print("Format: {0}\nSize: {1}\nMode: {2}".format(tatras.format,

tatras.size, tatras.mode))

The example prints basic information about the image with Pillow.

print("Format: {0}\nSize: {1}\nMode: {2}".format(tatras.format,

tatras.size, tatras.mode))

We print the image format, its size and mode.

$ ./basic\_info.py

Format: JPEG

Size: (350, 232)

Mode: RGB

This is the output of the program.

**Blurring image**

The ImageFilter module contains definitions for a pre-defined set of filters, which can be used with the filter() method.

blur\_image.py

#!/usr/bin/python3

from PIL import Image, ImageFilter

import sys

try:

img = Image.open("tatras.jpg")

except IOError:

print("Unable to load image")

sys.exit(1)

blurred = img.filter(ImageFilter.BLUR)

blurred.save("blurred.png")

The program loads an image, creates a blurred image from the original image, and saves the new image on disk.

from PIL import Image, ImageFilter

We import the Image and ImageFilter modules.

blurred = img.filter(ImageFilter.BLUR)

We apply the ImageFilter.BLUR on the original image; the operation returns a new modified image.

blurred.save("blurred.png")

With the save() method, we save the blurred image on disk.

**Converting image with Pillow**

With the save() method, we can convert an image to a different format.

convert2png.py

#!/usr/bin/python3

from PIL import Image

import sys

try:

tatras = Image.open("tatras.jpg")

except IOError:

print("Unable to load image")

sys.exit(1)

tatras.save('tatras.png', 'png')

The program reads a JPG image and converts it into PNG format.

tatras.save('tatras.png', 'png')

The second parameter of the save() method specifies the image format.

**GrayScale image**

With the Image.convert() method, we can produce a black and white image.

grayscale.py

#!/usr/bin/python3

from PIL import Image

import sys

try:

tatras = Image.open("tatras.jpg")

except IOError:

print("Unable to load image")

sys.exit(1)

grayscale = tatras.convert('L')

grayscale.show()

The program reads an image and transforms it into a grayscale image.

grayscale = tatras.convert('L')

The first parameter of the convert() method is the mode; the 'L' mode is black and white.

**Cropping image with Pillow**

The Image.crop() crops the image.

crop\_image.py

#!/usr/bin/python3

from PIL import Image

import sys

try:

tatras = Image.open("tatras.jpg")

except IOError:

print("Unable to load image")

sys.exit(1)

cropped = tatras.crop((100, 100, 350, 350))

cropped.save('tatras\_cropped.jpg')

The program crops an image. The cropped image is saved on disk.

cropped = tatras.crop((100, 100, 350, 350))

The crop() method takes a 4-tuple defining the left, upper, right, and lower pixel coordinates.

**Rotating image with Pillow**

The Image.rotate() returns a rotated copy of the image.

rotate\_image.py

#!/usr/bin/python3

from PIL import Image

import sys

try:

tatras = Image.open("tatras.jpg")

except IOError:

print("Unable to load image")

sys.exit(1)

rotated = tatras.rotate(180)

rotated.save('tatras\_rotated.jpg')

The program rotates an image by 180 degrees and saves the newly created image on disk.

**Displaying image in Tkinter**

The following program displays an image in a Tkinter program.

show\_tkinter.py

#!/usr/bin/python3

# -\*- coding: utf-8 -\*-

from PIL import Image, ImageTk

from tkinter import Tk

from tkinter.ttk import Frame, Label

import sys

class Example(Frame):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.loadImage()

self.initUI()

def loadImage(self):

try:

self.img = Image.open("tatrs.jpg")

except IOError:

print("Unable to load image")

sys.exit(1)

def initUI(self):

self.master.title("Label")

tatras = ImageTk.PhotoImage(self.img)

label = Label(self, image=tatras)

# reference must be stored

label.image = tatras

label.pack()

self.pack()

def setGeometry(self):

w, h = self.img.size

self.master.geometry(("%dx%d+300+300") % (w, h))

def main():

root = Tk()

ex = Example()

ex.setGeometry()

root.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

main()

The program displays an image in the Label widget of the Tkinter toolkit.

from PIL import Image, ImageTk

The ImageTk is a Tkinter-compatible photo image. It can be used everywhere Tkinter expects an image object.

tatras = ImageTk.PhotoImage(self.img)

We create a photo image.

label = Label(self, image=tatras)

The photoimage is given to the image parameter of the label widget.

label.image = tatras

In order not to be garbage collected, the image reference must be stored.

w, h = self.img.size

self.master.geometry(("%dx%d+300+300") % (w, h))

The size of the window fits the image size.

**Reading image from URL**

The next example reads an image from URL.

read\_from\_url.py

#!/usr/bin/python3

from PIL import Image

import requests

import sys

url = 'https://i.ytimg.com/vi/vEYsdh6uiS4/maxresdefault.jpg'

try:

resp = requests.get(url, stream=True).raw

except requests.exceptions.RequestException as e:

sys.exit(1)

try:

img = Image.open(resp)

except IOError:

print("Unable to open image")

sys.exit(1)

img.save('sid.jpg', 'jpeg')

The example reads an image from the URL and saves it on disk.

import requests

We use the requests library to download the image.

resp = requests.get(url, stream=True).raw

We read the image as raw data.

img = Image.open(resp)

Image is created from the response object.

img.save('sid.jpg', 'jpeg')

The image is saved.

**Drawing to Pillow image**

Pillow has some basic 2D graphics capabilities. ImageDraw module provides simple 2D graphics for Image objects. We can create new images, annotate or retouch existing images, and generate graphics on the fly for web use.

draw2image.py

#!/usr/bin/python3

from PIL import Image, ImageDraw

img = Image.new('RGBA', (200, 200), 'white')

idraw = ImageDraw.Draw(img)

idraw.rectangle((10, 10, 100, 100), fill='blue')

img.save('rectangle.png')

The example creates a new image and draws a blue rectangle on the image.

img = Image.new('RGBA', (200, 200), 'white')

A new Image is created. The image mode is 'RGBA'. Its size is 200x200 and the background is white.

idraw = ImageDraw.Draw(img)

From the image we create the ImageDraw object. Now we can perform some drawing operations on the image.

idraw.rectangle((10, 10, 100, 100), fill='blue')

With the rectangle() method, we draw a blue rectangle on the image.