Python Pillow - Working with Images

This chapter elaborates on topics including how to read and save an image in Pillow.

Reading an Image

Reading and writing images using pillow library is very simple, with the help of **PIL.Image module function**.

Syntax

Image.open(fp, mode=’r’)

Where

* **fp** − A filename (string), pathlib.Path object or a file object. The file object must implement read(), seek() and tell() methods and be opened in binary mode.
* **mode** − It’s an optional argument, if given, must be ‘r’.
* **Return value** − An Image object.
* **Error** − If the file cannot be found, or the image cannot be opened and identified.

Example

Following is a very simple example, where we are going to open an image of any format (We are using .**jpg**), display it in a window and then save it (default location) with another file format (.**png**).

from PIL import Image

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

image.show()

image.save('beach1.bmp')

image1 = Image.open('beach1.bmp')

image1.show()

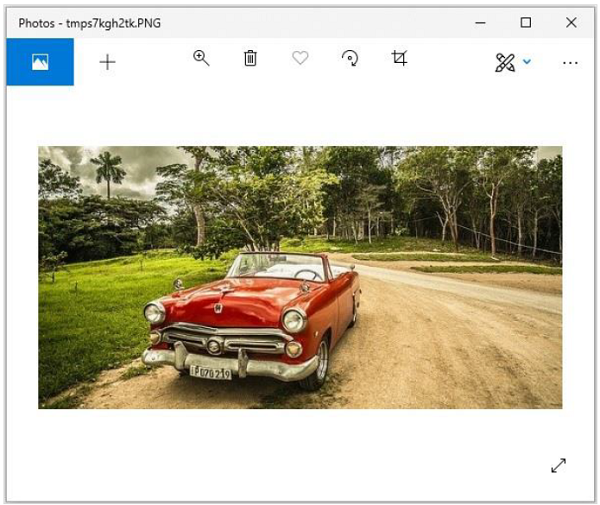
In the above example, we import the Image module from PIL library and then, call the **Image.open()** function to read an image from disk, which returns an image object data type. It will automatically determine the type of file by looking at the file content. For reading, the **open()** function accepts a **filename(string)**, a **path object** or an **image(file)** object.

So, by using the open() function, we are actually reading the image. Image.open() will read the image and get all the relevant information from the image.

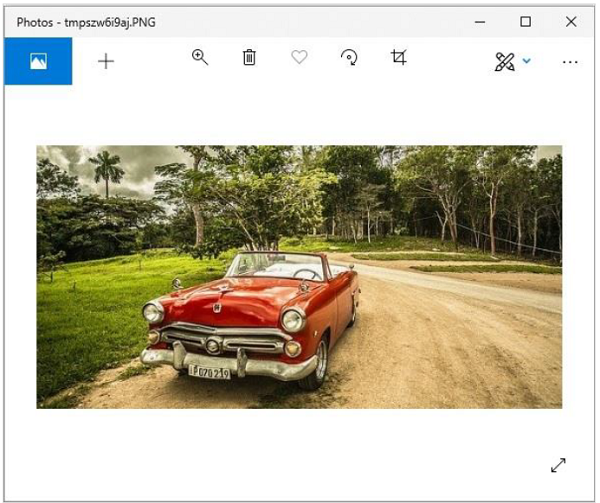
Output

If you save the above program as Example.py and execute, it displays the original (.jpg) and resaved (.bmp) images using standard PNG display utility, as follows −

**Actual image**



**Resaved image (.bmp)**



Saving an Image

The save() function writes an image to file. Like for reading (open() function), the save() function accepts a filename, a path object or a file object that has been opened to write.

Syntax

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

Where,

* **fp** − A filename (string), pathlib.Path object or file object.
* **format** − Optional format override. If omitted, the format to use is determined from the filename extension. If a file object was used instead of a filename, this parameter should always to used.
* **options** − Extra parameters to the image writer.
* **Return value** − None
* **KeyError** − If the output format could not be determined from the file name, use the format option to solve this.
* **IOError** − If the file could not be written, the file may have been created, and may contain partial data.

In short, the above syntax will save the image under the given filename. If no format is specified, then it is based on current filename extension. To provide the additional instructions to the writer, we use keyword options.

image.save('beach1.bmp')

In the above example, it saves the file based on the file extension to determine the type of image, for example – the above will create a bmp file in our current working directory.

You can also explicitly specify the file type as a second parameter −

image.save('beach1.gif', 'GIF')