Quick Links for Python
Recent Articles
MCQ / Quizzes
Practice Problems

Basics
Introduction
New Generation Language
Keywords, Set 1 Set 2
Explore More...
Variables
Variables,Expressions & Functions
Global and Local Variables
Type Conversion
Explore More...
Operators
Increment and Decrement Operator
Teranry Operator & Divison Operator
Logical and Bitwise Not Operators on Boolean
Any & ALL

Operator Functions Set 1 & Set 2
Data Types
Introduction
Arrays Set 1, Set 2
String Methods Set 1, Set 2, Set 3
String Template Class & String Formatting using %
List Methods Set 1, Set 2, Set 3
Tuples & Sets
Dictionary Methods Set 1, Set 2
ChainMap
Explore More...
Control Flow
Loops and Control Statements
Counters & Accessing Counters
Iterators & Iterator Functions Set 1, Set 2
Generators
Explore More...
Functions
Function Decorators
Returning Multiple Values
Yield instead of Return
Python Closures & Coroutine
Explore More...
Modules
Introduction

Numeric Functions & Logarithmic and Power functions
Calender Functions Set 1, Set 2
Complex Numbers Introduction & Important functions
Explore More...
Object Oriented Concepts
Class, Object and Members
Data Hiding and Object Printing
Inheritance, Subclass and super
Class method vs static method & Class or Static Variables
Explore More...
Exception Handling
Exception Handling
User-Defined Exceptions
Built-in Exceptions
Libraries and Functions
Timeit
Numpy Set 1, Set 2
Get and Post
import module & reload module
Collection Modules Deque, Namedtuple & Heap
Explore More...
Machine Learning with Python
Classifying data using Support Vector Machines(SVMs) in Python

K means Clustering
How to get synonyms/antonyms from NLTK WordNet in Python?
Explore More...
Misc
Sql using Python & MongoDB and Python
Json formatting & Python Virtual environment
Metaprogramming with Metaclasses in Python
Python Input Methods for Competitive Programming
Explore More...
Applications and Projects
Creating a proxy webserver Set 1, Set 2
Send Messsage to FB friend
Twitter Sentiment Analysis & Whatsapp using Python
Desktop Notifier & Junk File Organizer
Explore More...

Working with Images in Python

PIL is the Python Imaging Library which provides the python interpreter with image editing capabilities. It was developed by Fredrik Lundh and several other contributors. Pillow is the friendly PIL fork and an easy to use library developed by Alex Clark and other contributors. We'll be working with Pillow.

Installation:

- **Linux:** On linux terminal type the following:

```
pip install Pillow
```

  Installing pip via terminal:

```
sudo apt-get update
sudo apt-get install python-pip
```

- **Windows:** Download the appropriate Pillow package according to your python version. Make sure to download according to the python version you have.

We'll be working with the Image Module here which provides a class of the same name and provides a lot of functions to work on our images.To import the Image module, our code should begin with the following line:

```
from PIL import Image
```

Operations with Images:

- **Open a particular image from a path:**

```
#img  = Image.open(path)
# On successful execution of this statement,
# an object of Image type is returned and stored in img variable)

try:
    img  = Image.open(path)
except IOError:
    pass
# Use the above statement within try block, as it can
# raise an IOError if file cannot be found,
# or image cannot be opened.
```

Run on IDE

- **Retrieve size of image**: The instances of Image class that are created have many attributes, one of its useful attribute is size.

```
from PIL import Image

filename = "image.png"
with Image.open(filename) as image:
    width, height = image.size
#Image.size gives a 2-tuple and the width, height can be obtained
```

Run on IDE

Some other attributes are: Image.width, Image.height, Image.format, Image.info etc.

- **Save changes in image:** To save any changes that you have made to the image file, we need to give path as well as image format.

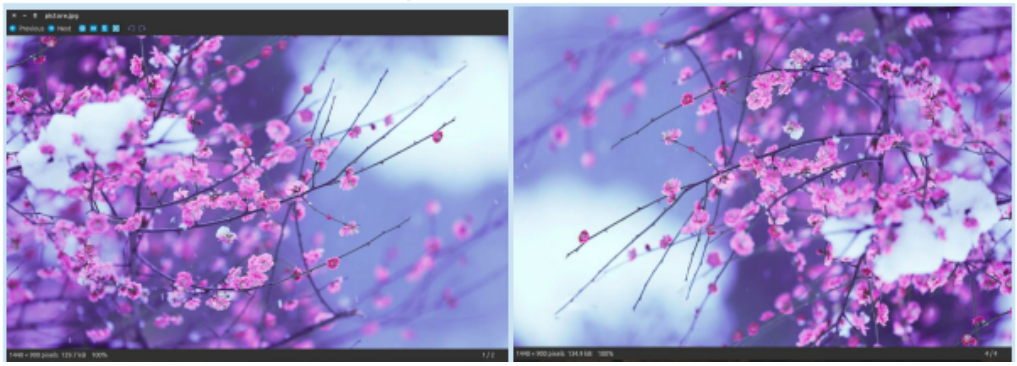
```
img.save(path, format)
# format is optional, if no format is specified,
#it is determined from the filename extension
```

Run on IDE

- **Rotating an Image:** The image rotation needs angle as parameter to get the image rotated.

```
from PIL import Image

def main():
    try:
        #Relative Path
        img = Image.open("picture.jpg")

        #Angle given
        img = img.rotate(180)

         #Saved in the same relative location
        img.save("rotated_picture.jpg")
    except IOError:
        pass

if __name__ == "__main__":
    main()
```

Run on IDE



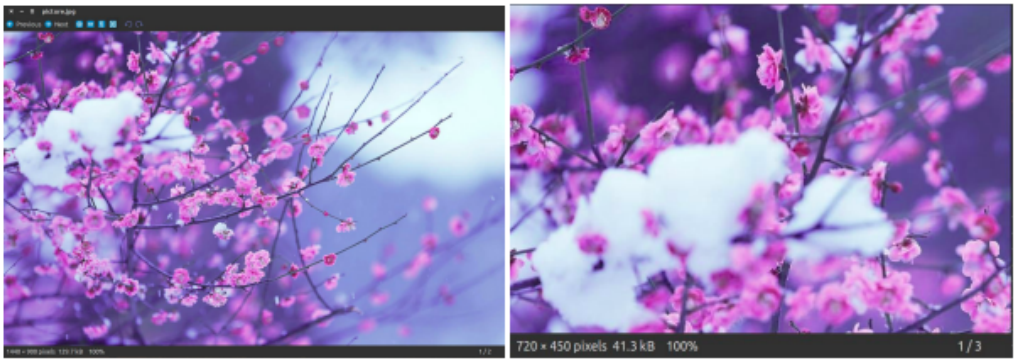
Note: There is an optional expand flag available as one of the argument of the rotate method, which if set true, expands the output image to make it large enough to hold the full rotated image.
As seen in the above code snippet, I have used a relative path where my image is located in the same directory as my python code file, an absolute path can be used as well.

- **Cropping an Image:** Image.crop(box) takes a 4-tuple (left, upper, right, lower) pixel coordinate, and returns a rectangular region from the used image.

```
from PIL import Image

def main():
    try:
        #Relative Path
        img = Image.open("picture.jpg")
        width, height = img.size

        area = (0, 0, width/2, height/2)
        img = img.crop(area)

        #Saved in the same relative location
        img.save("cropped_picture.jpg")
```

```
    except IOError:
        pass

if __name__ == "__main__":
    main()
```

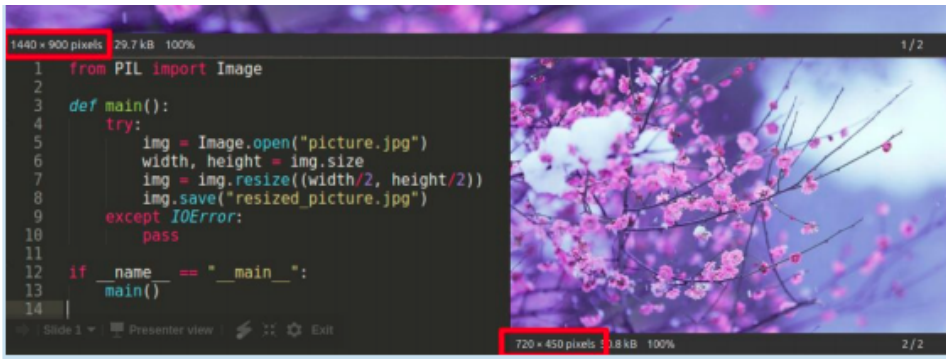
Run on IDE



- **Resizing an Image:** Image.resize(size)- Here size is provided as a 2-tuple width and height.

```
from PIL import Image

def main():
    try:
         #Relative Path
        img = Image.open("picture.jpg")
        width, height = img.size

        img = img.resize((width/2, height/2))

        #Saved in the same relative location
        img.save("resized_picture.jpg")
    except IOError:
        pass

if __name__ == "__main__":
    main()
```

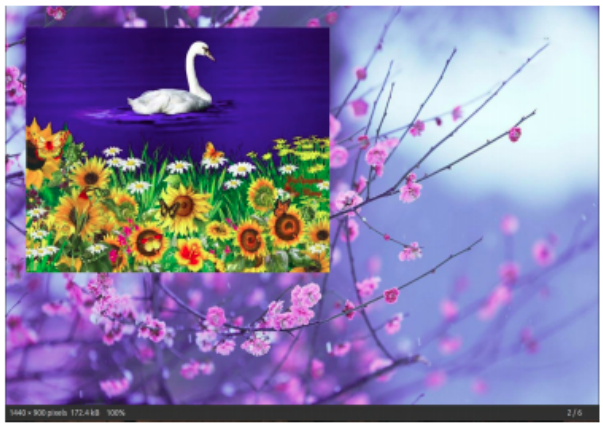
Run on IDE



- **Pasting an image on another image:** The second argument can be a 2-tuple (specifying the top left corner), or a 4-tuple (left, upper, right, lower) – in this case the size of pasted image must match the size of this box region, or None which is equivalent to (0, 0).

```
from PIL import Image

def main():
    try:
        #Relative Path
        #Image on which we want to paste
        img = Image.open("picture.jpg")

        #Relative Path
        #Image which we want to paste
        img2 = Image.open("picture2.jpg")
        img.paste(img2, (50, 50))

        #Saved in the same relative location
        img.save("pasted_picture.jpg")

    except IOError:
        pass

if __name__ == "__main__":
    main()

##An additional argument for an optional image mask image is also available.
```

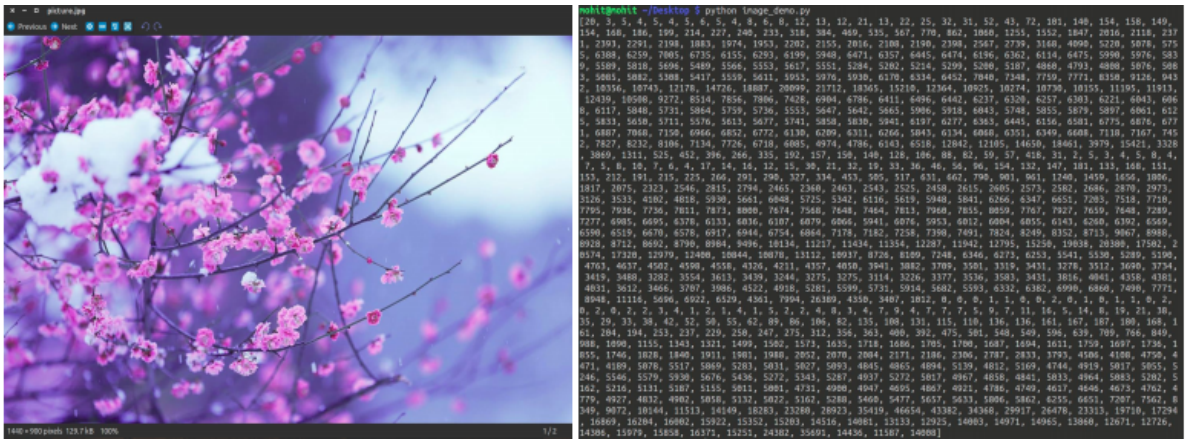
Run on IDE



- **Getting a Histogram of an Image:** This will return a histogram of the image as a list of pixel counts, one for each pixel in the image. (A histogram of an image is a graphical representation of the tonal distribution in a digital image. It contains what all the brightness values contained in an image are. It plots the number of pixels for each brightness value. It helps in doing the exposure settings.)
  from PIL import Image

```
def main():
    try:
        #Relative Path
        img = Image.open("picture.jpg")

        #Getting histogram of image
        print img.histogram()

    except IOError:
        pass

if __name__ == "__main__":
    main()
```

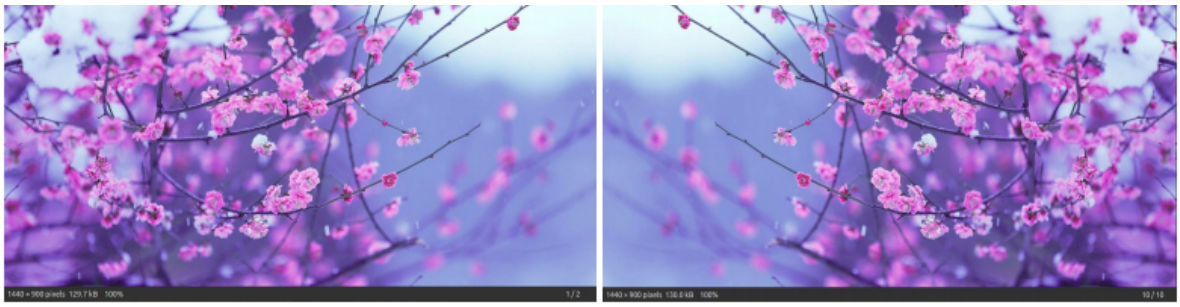
Run on IDE



- **Transposing an Image:** This feature gives us the mirror image of an image

```
    from PIL import Image

def main():
    try:
        #Relative Path
        img = Image.open("picture.jpg")

        #transposing image
        transposed_img = img.transpose(Image.FLIP_LEFT_RIGHT)

        #Save transposed image
        transposed_img.save("transposed.jpg")
    except IOError:
        pass

if __name__ == "__main__":
    main()
```

Run on IDE



- **Split an image into individual bands:** Splitting an image in RGB mode, creates three new images each containing a copy of the original individual bands.

```
from PIL import Image

def main():
    try:
        #Relative Path
        img = Image.open("picture.jpg")

        #splitting the image
        print img.split()
    except IOError:
        pass

if __name__ == "__main__":
    main()
```

Run on IDE

```
from PIL import Image

def main():
    try:
        img = Image.open("picture.jpg")
        print img.mode
        print
        print img.split()
    except IOError:
        pass

if __name__ == "__main__":
    main()
```

```
mohit@mohit ~/Desktop $ python image_demo.py
RGB

(<PIL.Image.Image image mode=L size=1440x900 at 0x7F34B1958D40>,
 <PIL.Image.Image image mode=L size=1440x900 at 0x7F34B1958AB8>,
 <PIL.Image.Image image mode=L size=1440x900 at 0x7F34B1958E60>)
```

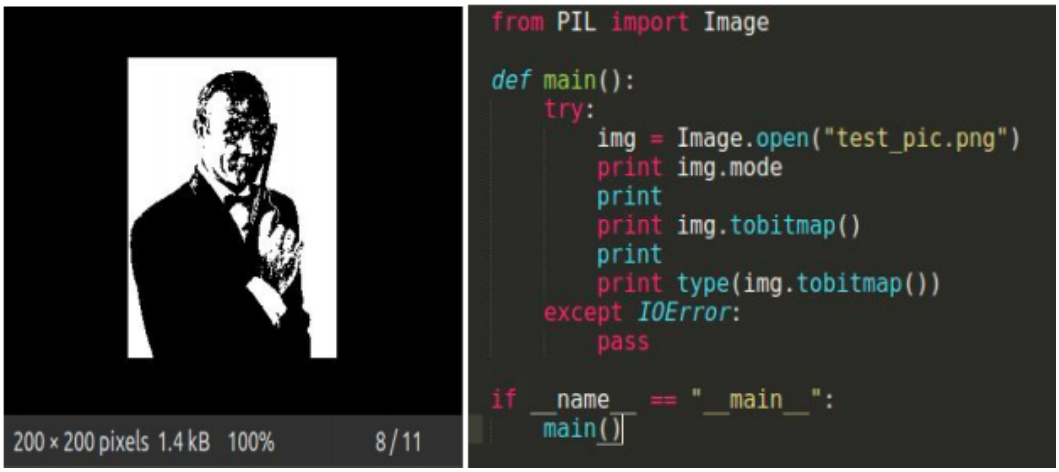
- **tobitmap:** Converting an image to an X11 bitmap (A plain text binary image format). It returns a string

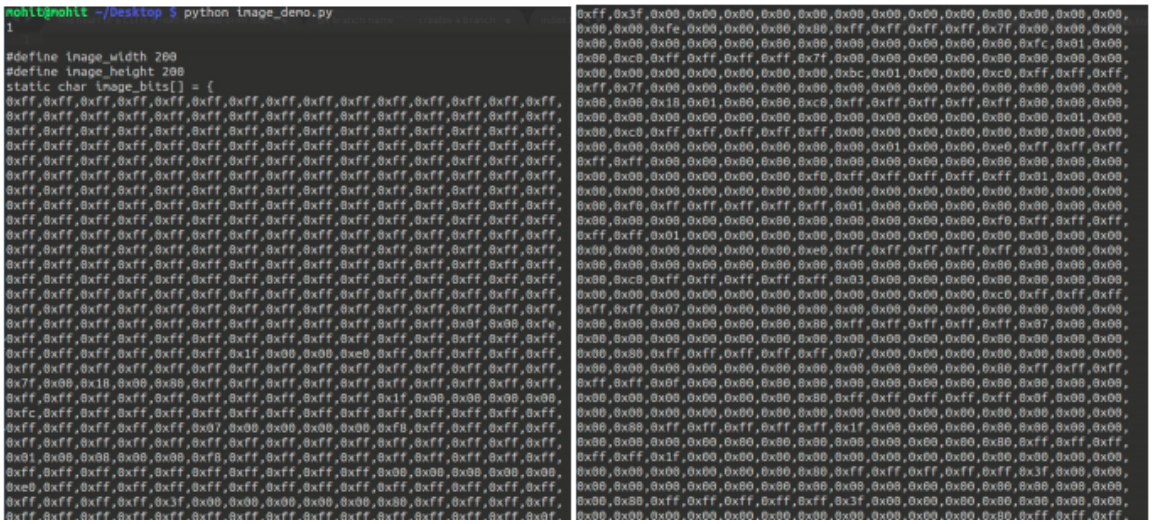
containing an X11 bitmap, it can only be used for mode “1” images, i.e. 1 bit pixel black and white images.

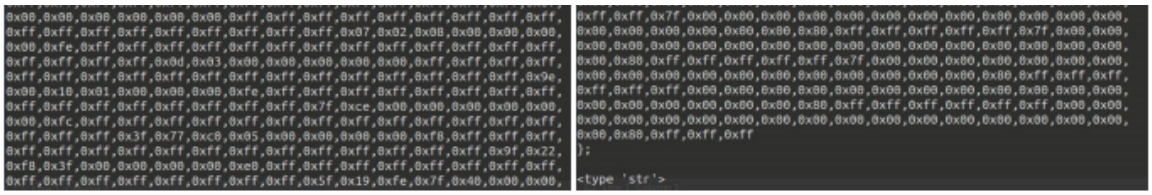
from PIL import Image

```
def main():
    try:
        #Relative Path
        img = Image.open("picture.jpg")
        print img.mode

        #converting image to bitmap
        print img.tobitmap()

        print type(img.tobitmap())
    except IOError:
        pass

if __name__ == "__main__":
    main()
```

Run on IDE





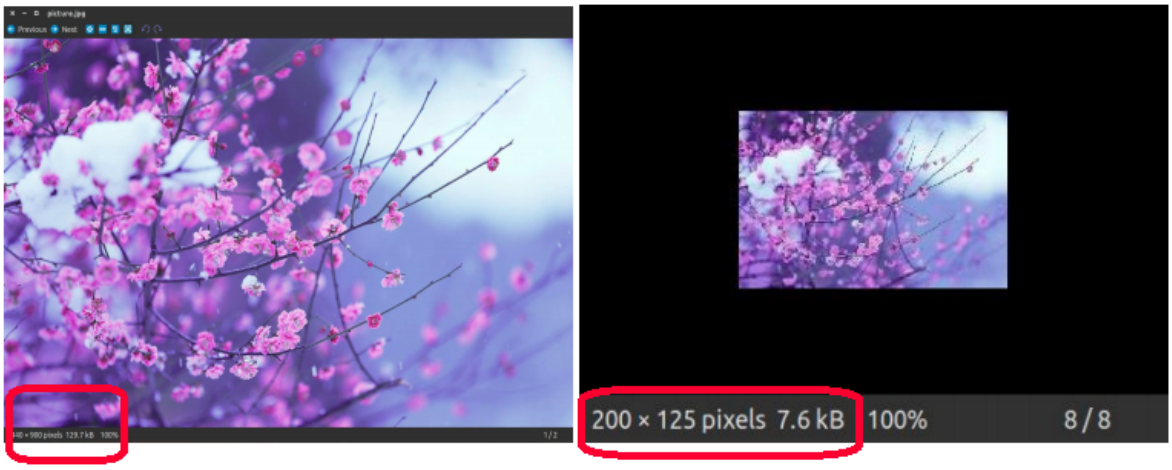


- **Creating a thumbnail:** This method creates a thumbnail of the image that is opened. It does not return a new image object, it makes in-place modification to the currently opened image object itself. If you do not want to change the original image object, create a copy and then apply this method. This method also evaluates the appropriate to maintain the aspect ratio of the image according to the size passed.

from PIL import Image

```
def main():
    try:
        #Relative Path
        img = Image.open("picture.jpg")

        #In-place modification
        img.thumbnail((200, 200))

        img.save("thumb.jpg")
    except IOError:
        pass

if __name__ == "__main__":
    main()
```

Run on IDE



This article is contributed by **Mohit Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner **Company Wise Coding Practice**

Project Python Image-Processing

Recommended Posts:

Graph Plotting in Python | Set 1
OpenCV Python program for Vehicle detection in a Video frame
Whatsapp using Python!
Opencv Python program for Face Detection
Graph Plotting in Python | Set 3

(Login to Rate and Mark)

3.5 Average Difficulty : **3.5/5.0**
Based on **2** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!