1. What does RGBA stand for?

2. From the Pillow module, how do you get the RGBA value of any images?

3. What is a box tuple, and how does it work?

4. Use your image and load in notebook then, How can you find out the width and height of an

Image object?

5. What method would you call to get Image object for a 100×100 image, excluding the lower-left

quarter of it?

6. After making changes to an Image object, how could you save it as an image file?

7. What module contains Pillow’s shape-drawing code?

8. Image objects do not have drawing methods. What kind of object does? How do you get this kind

of object?

Sol1:

RGBA(Red-Green-Blue-Alpha)

The RGB color model is extended in this specification to include “alpha” to allow specification of the opacity of a color.

Sol2:

Pillow offers the ImageColor. getcolor() function so you don't have to memorize RGBA values for the colors you want to use. This function takes a color name string as its first argument, and the string 'RGBA' as its second argument, and it returns an RGBA tuple.

Sol3:

The box.tuple submodule provides read-only access for the tuple user data type. It allows, for a single [tuple](https://www.tarantool.io/en/doc/latest/concepts/data_model/value_store/#index-box-tuple): selective retrieval of the field contents, retrieval of information about size and iteration over all the fields.

Sol4:

The image size (width, height) can be obtained using the shape attribute, which returns a tuple of dimensions. Besides OpenCV, when using other libraries such as Pillow to read an image file and convert it into an ndarray , you can also obtain the image size using the shape attribute.

Sol5:

To get an Image object for a 100x100 image, excluding the lower-left quarter of it, you would need to use the HTML5 Canvas API to manipulate the image data. Here's an example of how you can achieve this:

// Create a new canvas element

const canvas = document.createElement('canvas');

canvas.width = 100;

canvas.height = 100;

// Get the canvas context and load the image

const ctx = canvas.getContext('2d');

const img = new Image();

img.src = 'path/to/image.png';

// Once the image has loaded, draw it on the canvas

img.onload = function() {

// Draw the image on the canvas

ctx.drawImage(img, 0, 0, 100, 100);

// Exclude the lower-left quarter of the image

ctx.clearRect(0, 75, 25, 25);

// Get the resulting image data

const imageData = ctx.getImageData(0, 0, 100, 100);

// Create a new Image object from the resulting image data

const newImg = new Image();

newImg.src = URL.createObjectURL(new Blob([imageData], {type: 'image/png'}));

// Append the new Image object to the document

document.body.appendChild(newImg);

};

In this example, we create a new canvas element with a width and height of 100 pixels. We then load the image and draw it on the canvas using the drawImage method. We then use the clearRect method to exclude the lower-left quarter of the image. Finally, we use the getImageData method to get the resulting image data and create a new Image object from it using the Blob constructor. The new Image object can then be appended to the document as needed.

Sol6:

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

**Parameters:**

**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 be used.  
**options** – Extra parameters to the image writer.

**Returns:** None

**Raises:**

**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.

Sol7:

The 'ImageDraw' module provides simple 2D graphics support for Image Object. Generally, we use this module to create new images, annotate or retouch existing images and to generate graphics on the fly for web use. The graphics commands support the drawing of shapes and annotation of text.

Sol8:

If you need to draw lines, rectangles, circles, or other simple shapes on an image, use Pillow’s ImageDraw module.

>>> from PIL import Image, ImageDraw

>>> im = Image.new('RGBA', (200, 200), 'white')

>>> draw = ImageDraw.Draw(im)

First, we import Image and ImageDraw. Then we create a new image, in this case, a 200×200 white image, and store the Image object in im. We pass the Image object to the ImageDraw.Draw() function to receive an ImageDraw object. This object has several methods for drawing shapes and text onto an Image object. Store the ImageDraw object in a variable like draw so you can use it easily somewhere else.