**FSDS MAY BATCH 2022(Python Basics 14)**

**Submitted by: Shubham Tiwari**

Q1. What does RGBA stand for?

Ans: RGBA stands for Red, Green, Blue, and Alpha. It is a color model used to represent colors in digital images and graphics. In Python, RGBA values are commonly used to represent colors in graphics libraries such as Pygame and Matplotlib, where the alpha value represents the opacity or transparency of the color.

Q2. From the Pillow module, how do you get the RGBA value of any images?

Ans: To get the RGBA value of any image using the Pillow module in Python, we can use the **getpixel** method of the **Image** class. Here's an example code:

from PIL import Image

**# Open the image**

image = Image.open('path/to/image.png')

**# Get the RGBA value of a pixel at (x, y) coordinates**

r, g, b, a = image.getpixel((x, y))

In this example, ‘**r’**, ‘**g’**, ‘**b’**, and ‘**a’** represent the red, green, blue, and alpha values of the pixel at the specified coordinates **(x, y)**. We can replace **'path/to/image.png'** with the actual path of our image file.

Q3. What is a box tuple, and how does it work?

Ans: In the context of image processing libraries like Pillow, a box tuple is a tuple of four values **(left, upper, right, lower)** that defines a rectangular region within an image.

The **left** and **upper** values represent the x and y coordinates of the upper-left corner of the rectangle, while the **right** and **lower** values represent the x and y coordinates of the lower-right corner of the rectangle.

Box tuples are commonly used in functions and methods that operate on a specific region of an image. For example, the **crop** method of the **Image** class in Pillow takes a box tuple as an argument to specify the region of the image to be cropped. Similarly, the **paste** method can be used to paste an image into another image at a specific location specified by a box tuple.

Here's an example code snippet that demonstrates the use of a box tuple:

from PIL import Image

**# Open an image**

image = Image.open('path/to/image.jpg')

**# Define a box tuple**

box = (100, 100, 200, 200)

**# Crop the image using the box tuple**

cropped\_image = image.crop(box)

**# Display the cropped image**

cropped\_image.show()

In this example, the box tuple ‘**(100, 100, 200, 200)’** defines a rectangular region of the image that starts at the pixel **(100, 100)** and extends to the pixel ‘**(200, 200)’**. The **crop** method is then used to crop the image to this region, and the resulting cropped image is displayed using the **show** method.

Q4. Use your image and load in notebook then, How can you find out the width and height of an Image object?

Ans: To find out the width and height of an image using the Pillow library in a Jupyter Notebook, we can use the **size** attribute of the **Image** object. Here's an example code:

from PIL import Image

from IPython.display import display

**# Open an image and display it**

image = Image.open('path/to/image.jpg')

display(image)

# **Get the width and height of the image**

width, height = image.size

# **Print the width and height**

print('Width:', width)

print('Height:', height)

In this example, the **Image** object is first opened using the **open** method, and the image is then displayed using the **display** function from the **IPython.display** module. The **size** attribute of the **Image** object is then used to obtain the width and height of the image, which are stored in the **width** and **height** variables. Finally, the width and height are printed using the **print** function. Note that we should replace **'path/to/image.jpg'** with the actual path to our image file.

Q5. What method would you call to get Image object for a 100×100 image, excluding the lower-left quarter of it?

Ans: Assume that we have an Image object in Python, you could use the crop() method to exclude the lower-left quarter of a 100x100 image. Here's an example code:

from PIL import Image

**# Open the image file**

img = Image.open('image\_file.jpg')

**# Crop the image to exclude the lower-left quarter**

cropped\_img = img.crop((0, 0, 50, 50))

**# Show the cropped image**

cropped\_img.show()

In the code above, the crop() method is called with the parameters (0, 0, 50, 50), which specifies that the left and top edges of the crop should be at the image's origin (0, 0), and the right and bottom edges of the crop should be at (50, 50). This creates a new image object containing the upper-right three-quarters of the original 100x100 image.

Q6. After making changes to an Image object, how could you save it as an image file?

Ans: To save changes made to an Image object as an image file, we can use the save() method. Here's an example code:

from PIL import Image

**# Open the image file**

img = Image.open('image\_file.jpg')

**# Make changes to the image**

**………………………….**

**# Save the modified image as a new file**

img.save('modified\_image\_file.jpg')

In the code above, the save() method is called on the Image object with the filename for the new image file as the argument. The format of the new image file will be inferred from the file extension of the filename we provide. For example, if we use a filename of 'modified\_image\_file.png', the new image file will be saved in PNG format.

Q7. What module contains Pillow’s shape-drawing code?

Ans: Pillow's shape-drawing code is contained in the ImageDraw module. The ImageDraw module is part of the Pillow library, which is a fork of the Python Imaging Library (PIL) that provides additional functionality and improved support for modern Python versions.

The ImageDraw module provides methods for drawing shapes such as lines, rectangles, ellipses, and polygons on an image object. The module also provides methods for drawing text and applying image filters.

Q8. Image objects do not have drawing methods. What kind of object does? How do you get this kind of object?

Ans:In the Python Imaging Library (PIL) and its fork Pillow, the ImageDraw module provides drawing methods for image objects.

To use ImageDraw, we can create a new ImageDraw object by calling the ImageDraw.Draw() function and passing in an Image object. Here's an example code:

from PIL import Image, ImageDraw

**# Open the image file**

img = Image.open('image\_file.jpg')

**# Create a new ImageDraw object**

draw = ImageDraw.Draw(img)

**# Use the drawing methods to modify the image**

……………………………….

# Save the modified image as a new file

img.save('modified\_image\_file.jpg')

In the code above, the ImageDraw.Draw() function is called with the Image object as the argument, creating a new ImageDraw object that can be used to draw on the image. We can then use the drawing methods provided by the ImageDraw object to modify the image, such as drawing lines, rectangles, circles, and text.

Top of Form