Q1

In the RGBA model:

Red (R): Represents the intensity of the red color component.

Green (G): Represents the intensity of the green color component.

Blue (B): Represents the intensity of the blue color component.

Alpha (A): Represents the transparency or opacity of the color. A higher alpha value indicates greater opacity, while a lower value indicates greater transparency.

Q2

from PIL import Image

# Open an image using PIL

image = Image.open("your\_image\_path.png") # Replace with your image path

# Define the coordinates of the pixel you want to get the RGBA value from

x = 100

y = 50

# Get the RGBA value of the pixel at the specified coordinates

rgba\_value = image.getpixel((x, y))

print("RGBA value:", rgba\_value)

from PIL import Image

# Open an image using PIL

image = Image.open("your\_image\_path.png") # Replace with your image path

# Define the coordinates of the pixel

x = 100

y = 50

rgba\_value = image.getpixel((x, y))

print("RGBA value:", rgba\_value)

Q3

A box tuple, often referred to as a bounding box tuple, is a data structure used to define a rectangular region within an image or a two-dimensional space. It's represented as a tuple containing four values that define the boundaries of the rectangle

(left, upper, right, lower)

left: The x-coordinate of the left edge of the rectangle.

upper: The y-coordinate of the upper edge of the rectangle.

right: The x-coordinate of the right edge of the rectangle.

lower: The y-coordinate of the lower edge of the rectangle.

Q4

!pip install pillow

from PIL import Image

image\_path = 'your\_image\_path.jpg'

image = Image.open(image\_path)

width, height = image.size

print(f"Image width: {width}, Image height: {height}")

Q5

from PIL import Image

# Open the original image (replace 'your\_image\_path.jpg' with the actual image path)

image\_path = 'your\_image\_path.jpg'

original\_image = Image.open(image\_path)

# Define the coordinates for the cropping box (left, upper, right, lower)

crop\_box = (0, 0, original\_image.width // 2, original\_image.height // 2)

# Crop the image using the defined cropping box

cropped\_image = original\_image.crop(crop\_box)

# Display the cropped image

cropped\_image.show()

Q6

from PIL import Image

# Open the original image (replace 'your\_image\_path.jpg' with the actual image path)

image\_path = 'your\_image\_path.jpg'

original\_image = Image.open(image\_path)

# Perform image modifications (e.g., cropping, resizing, filtering, etc.)

# Save the modified image to a file

modified\_image\_path = 'modified\_image.jpg'

original\_image.save(modified\_image\_path)

print("Modified image saved successfully.")

Q7

from PIL import Image, ImageDraw

from PIL import Image, ImageDraw

# Create a new blank image

width, height = 400, 400

image = Image.new("RGB", (width, height), "white")

draw = ImageDraw.Draw(image)

# Draw a red rectangle

rectangle\_coords = [(100, 100), (300, 300)]

draw.rectangle(rectangle\_coords, outline="red")

# Draw a blue ellipse

ellipse\_coords = [(150, 150), (250, 250)]

draw.ellipse(ellipse\_coords, outline="blue")

# Save or display the modified image

image.show()

Q8

from PIL import Image, ImageDraw

# Open an image

image\_path = 'your\_image\_path.jpg'

image = Image.open(image\_path)

# Create an ImageDraw.Draw object associated with the image

draw = ImageDraw.Draw(image)

# Now you can use drawing methods on the draw object

draw.line([(10, 10), (100, 100)], fill="red")

draw.rectangle([(150, 50), (250, 150)], outline="blue")

draw.text((50, 200), "Hello, Pillow!", fill="green")

# Save or display the modified image

image.show()