😊 Working with Images with Python

1. In this lab, we are going to work with images using our Jupyter Notebook. So, by leveraging the power of some common libraries that you can install, such as **PILLOW**, Python gains the ability to work with and manipulate images for simple tasks. You can install Pillow by running:

   pip install pillow

2. For this particular lab, you will find some images in the Repository. When you clone or download this lab, you will also get the images used in this lab.

3. We will start by opening images. So, first we need to import image from PIL.
4. The code uses the **PIL (Pillow) library** to open an image file named **"example.jpg"** and stores it in the variable mac. The Image.open() function loads the image, allowing further processing like resizing, cropping, or filtering. The type(mac) command returns the type of the object, which confirms that it is an instance of a specialized **PIL Image** class.

```
[2]: from PIL import Image

[4]: mac = Image.open('example.jpg')

[6]: # Note this is a specialized file type from PIL (pillow)
     type(mac)

[6]: PIL.JpegImagePlugin.JpegImageFile
```

5. Now, if we only type mac then we can see the image in Jupyter Notebook.

```
[8]: # Only for jupyter notebook , use mac.show() for other IDEs
     mac
```

[8]:

6. We can also see the image information by typing below commands.

```
[10]: # (width, height)
      mac.size

[10]: (1993, 1257)

[12]: mac.filename

[12]: 'C:\\Users\\PULKIT\\ansel\\Complete-Python-3-Bootcamp-master\\14-Working-with-Images\\example.jpg'

[14]: mac.format_description

[14]: 'JPEG (ISO 10918)'
```

7. So, we have learned how to see images. Now we will learn to crop the images.
8. To crop images (that is, grab a subsection), you can use the crop() method on the image object. The crop() method returns a rectangular region from this image. The box is a 4-tuple defining the left, upper, right, and lower pixel coordinates.
9. Here, you can see we just cropped our previous image.

```
[16]: mac.crop((0,0,100,100))
```

[16]:

10. Now we have a pencil image with us, and we just opened it.

```
[18]: pencils = Image.open("pencils.jpg")
```

```
[20]: pencils
```

[20]:

11. Then look for the size of the image and we cropped it as well.

[22]:
```
pencils.size
```

[22]:
```
(1950, 1300)
```

[24]:
```
# Start at top corner (0,0)
x = 0
y = 0

# Grab about 10% in y direction , and about 30% in x direction
w = 1950/3
h = 1300/10

pencils.crop((x,y,w,h))
```

[24]:


12. Now here what we did is we looked at the cropped pencil image from the bottom.

[28]:
```
pencils.size
```

[28]:
```
(1950, 1300)
```

[30]:
```
x = 0
y = 1100
w = 1950/3
h = 1300
```

[32]:
```
pencils.crop((x,y,w,h))
```

[32]:


13. Below, you can see that the code first retrieves the **size** of the image stored in mac. It then calculates a midpoint value (halfway = 1993/2) and defines cropping boundaries.
14. The mac.crop((x, y, w, h)) function extracts a rectangular portion of the image based on these coordinates, effectively cropping a specific section.

```
[36]: mac.size
```

```
[36]: (1993, 1257)
```

```
[38]: halfway = 1993/2
```

```
[40]: x = halfway - 200
      w = halfway + 200
```

```
[42]: y = 800
      h = 1257
```

```
[44]: mac.crop((x,y,w,h))
```

[44]:

15. Here we have copied the cropped mac image and pasted it inside the real size mac image.

```
[46]: computer = mac.crop((x,y,w,h))
```

```
[48]: mac.paste(im=computer,box=(0,0))
```

```
[50]: mac
```

50]:

16. Below, we are again pasting the cropped mac image but at a different position.

```
[52]: mac.paste(im=computer,box=(796,0))
```

```
[54]: mac
```

[54]:

17. We can use resize() method to resize an image.

```
[56]: mac.size
```

```
[56]: (1993, 1257)
```

```
[58]: h,w = mac.size
```

```
[60]: new_h = int(h/3)
      new_w = int(w/3)
```

```
[62]: # Note this is not permanent change
      # for permanent change, do a reassignment
      # e.g. mac = mac.resize((100,100))
      mac.resize((new_h,new_w))
```

[62]:

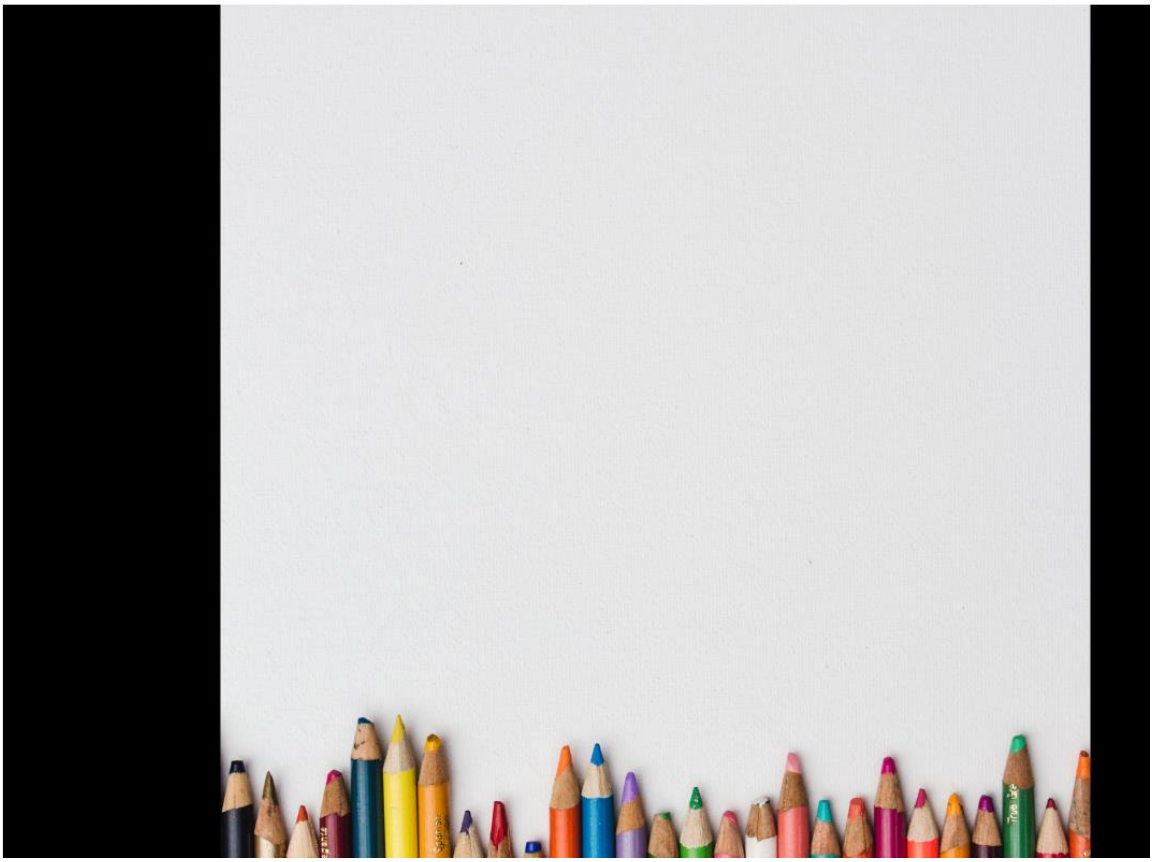
```
[64]: mac.resize((3000,500))
```

[64]:

18. We can rotate images by specifying the amount of degrees to rotate on the rotate() method. The original dimensions will be kept and "filled" in with black. You can optionally pass in the expand parameter to fill the new rotated image to the old dimensions.
19. Below, you can see that we have rotated the pencils images to 90 degrees.

```
[66]: pencils.rotate(90)
```

[66]:



20. Using the below command we have rotated the image to 90 degrees and we have also filled the black space that we were getting in the previous image.

```
[34]: pencils.rotate(90,expand=True)
```

[34]:

21. This is how the image would look when it is rotated to 120 degrees.

```
[68]: # The image is cut off
      pencils.rotate(120)
```

[68]:

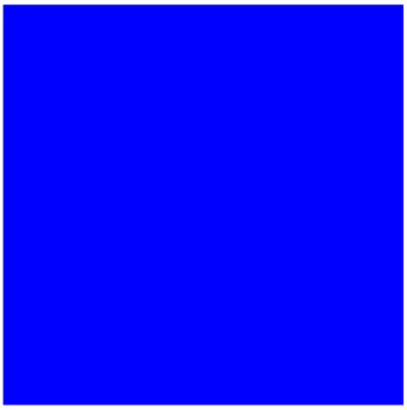
22. We can add an alpha value (RGBA stands for RED, Green, Blue, Alpha) where values can go from 0 to 255. If Alpha is 0, the image is completely transparent; if it is 255, then it's completely opaque.
23. Now we also have some coloured images as you can see below in the snapshots, we have a red and blue image.

```
[72]: red = Image.open('red_color.jpg')

[74]: red
```

[74]:

```
[76]: blue = Image.open('blue_color.png')

[78]: blue
```

[78]:



24. Using the below commands, we have changed our red image into a different color image.
25. You can play with the commands and changes the colors of the image resize them or crop them.

[80]:
```
red.putalpha(128)
```

[82]:
```
red
```

[82]: