Getting started with Images

The image methods in cv2 starts with `im` most of the times
we use the `imread` method to read images. this takes in 2 parameters the first the file name the second parameter is the channles to read from the image

In [13]:

```
from cv2 import cv2
import numpy

image = cv2.imread("images\\lena.jpg",1)
```

Second argument is a flag which specifies the way image should be read.

- cv2.IMREAD_COLOR : Loads a color image. Any transparency of image will be neglected. It is the default flag.
- cv2.IMREAD_GRAYSCALE : Loads image in grayscale mode
- cv2.IMREAD_UNCHANGED : Loads image as such including alpha channel

The cv2 imread wont complain if the image file is not rpesent instead the image object which is infact a numpy array return NONE. else it would return the actual numpy array.

In [19]:

```
from cv2 import cv2
import numpy

image = cv2.imread("images\\lenasa.jpg",1)
print(image)
```

```
None
```

We use the `imshow` method to show images

In [1]:

```
from cv2 import cv2
import numpy as np

image = cv2.imread("images\\lena.jpg",1)
cv2.imshow('image',image)
```

The below code will only show the image for a split second. The reason being the program ends after showing the image. TO show the image longer we need to make the program execution to pasue. We can do this by adding a keyboard interrupt.

We add the `waitKey()` function to wait for a keyboard interrupt or a timeout event. The timeout has to be provided in milliseconds. If the timeout is `0` then the function waits forever unless a keyboard stroke is recorded

In [ ]:

```
from cv2 import cv2
import numpy as np

image = cv2.imread("images\\lena.jpg",1)
cv2.imshow('image',image)
cv2.waitKey(5000)
```

The reason why `waitkey()` works is because of the fact that it is a blocking function which means it stops the execution of the program until an event occurs.

It is always recomended to destroy a window object explicitly once its use is over.

In [ ]:

```
from cv2 import cv2
import numpy as np

image = cv2.imread("images\\lena.jpg",1)
cv2.imshow('image',image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Writing Images

we can write/create images using the `imwrite()` method

In [3]:

```
import numpy as np
import cv2

image = cv2.imread('images\\lena.jpg',0)
cv2.imshow('image',image)
k = cv2.waitKey(0)
if k == 27:          # wait for ESC key to exit
    cv2.destroyAllWindows()
elif k == ord('s'): # wait for 's' key to save and exit
    cv2.imwrite('images\\lena_copy.png',image)
    cv2.destroyAllWindows()
```

The problem with the above code is that even if we press any other key (apart from escape) the program exits which is not desirable. so we can solve the above problem by using a while loop.

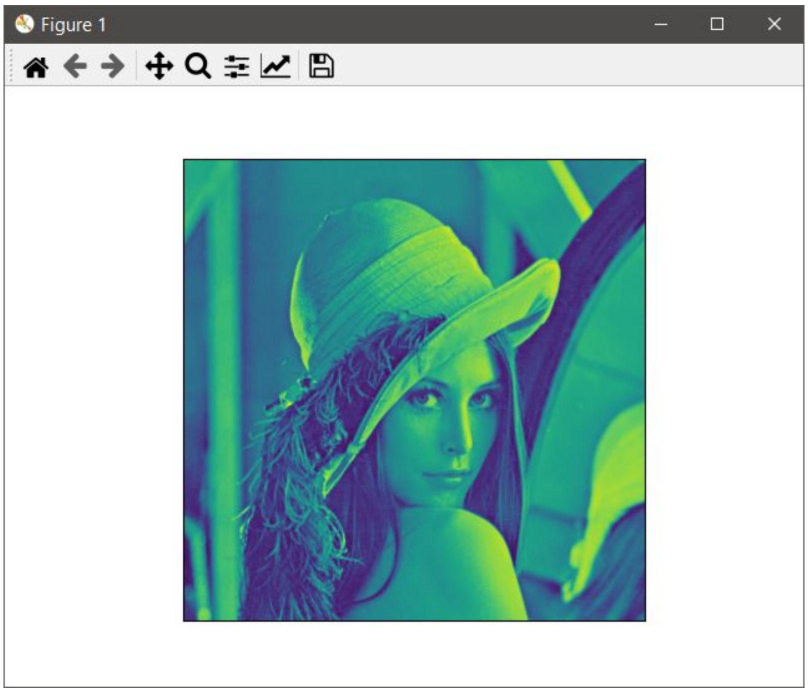
In [ ]:

```
import numpy as np
import cv2

image = cv2.imread('images\\lena.jpg',0)
cv2.imshow('image',image)

while True:
    k = cv2.waitKey(0)
    if k == 27:         # wait for ESC key to exit
        break
    elif k == ord('s'): # wait for 's' key to save and exit
        cv2.imwrite('images\\lena_copy.png',image)
        break
    else:
        pass
cv2.destroyAllWindows()
```

Getting the sequence correct

In [1]:

```
%matplotlib qt
#just to make matplotlib to make a separate window for showing image
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('images\\lena.jpg',0)
plt.imshow(img)
plt.xticks([]), plt.yticks([])  # to hide tick values on X and Y axis
plt.show()
```



The problem is that the matplotlib uses RGB format while opencv uses GRB format to solve the above problem

Since `img` object created using cv2 is a numpy array we can solve this problem with numpy indexing and slicing.

In [1]:

```
#just to make matplotlib to make a separate window for showing image
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('images\\lena.jpg',1)
# cv2.imshow('image',img)

img = img[:,:,::-1]

plt.imshow(img)
# plt.xticks([]), plt.yticks([])  # to hide tick values on X and Y axis
plt.show()
```