Opening-Image-Files-in-a-Notebook

August 1, 2021

Table of Contents

1 Opening Image Files in a Notebook

[1]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
img = cv2.imread('../Figures/1. Onmyoji illustration - 1.png')
```

[3]:
```
img
```

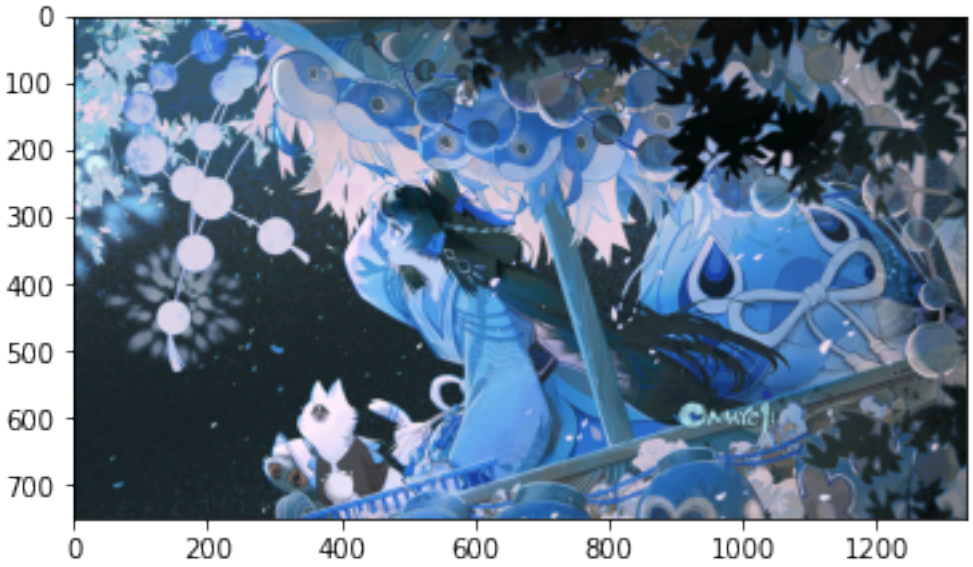
[3]:
```
array([[[110, 157, 172],
        [ 66, 122, 153],
        [ 63, 131, 169],
        …,
        [  8,  12,  12],
        [  2,   9,   9],
        [  6,  11,  11]],

       [[127, 169, 201],
        [ 72, 123, 147],
        [ 55, 117, 156],
        …,
```

```
        [ 12,  14,  14],
        [  3,   9,   9],
        [  0,   8,   8]],

       [[126, 179, 213],
        [ 80, 135, 171],
        [ 49, 117, 151],
        …,
        [  1,   8,   9],
        [  2,   9,   9],
        [  1,   8,   8]],

       …,

       [[ 24,  43,  43],
        [ 31,  34,  37],
        [ 30,  34,  49],
        …,
        [ 81, 102, 111],
        [ 81, 102, 111],
        [ 81, 102, 111]],

       [[ 31,  37,  48],
        [ 32,  34,  43],
        [ 24,  36,  40],
        …,
        [ 81, 102, 112],
        [ 81, 102, 111],
        [ 81, 102, 111]],

       [[ 32,  40,  50],
        [ 26,  34,  47],
        [ 28,  36,  43],
        …,
        [ 81, 102, 112],
        [ 81, 102, 111],
        [ 85, 102, 118]]], dtype=uint8)
```

---

[4]:
```
img_bgr = cv2.imread('../Figures/1. Onmyoji illustration - 1.png')
plt.imshow(img_bgr)
```

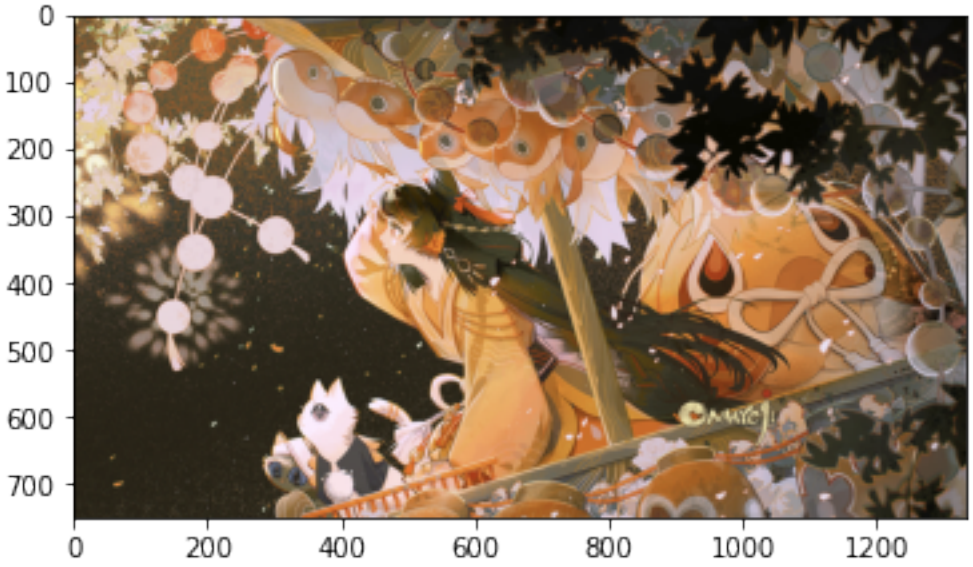
[4]:
```
<matplotlib.image.AxesImage at 0x7fa448f05150>
```



The image has been correctly loaded by openCV as a numpy array, but the color of each pixel has been sorted as BGR. Matplotlib's plot expects an RGB image so, for a correct display of the image, it is necessary to swap those channels. This operation can be done either by using openCV conversion functions cv2.cvtColor() or by working directly with the numpy array.

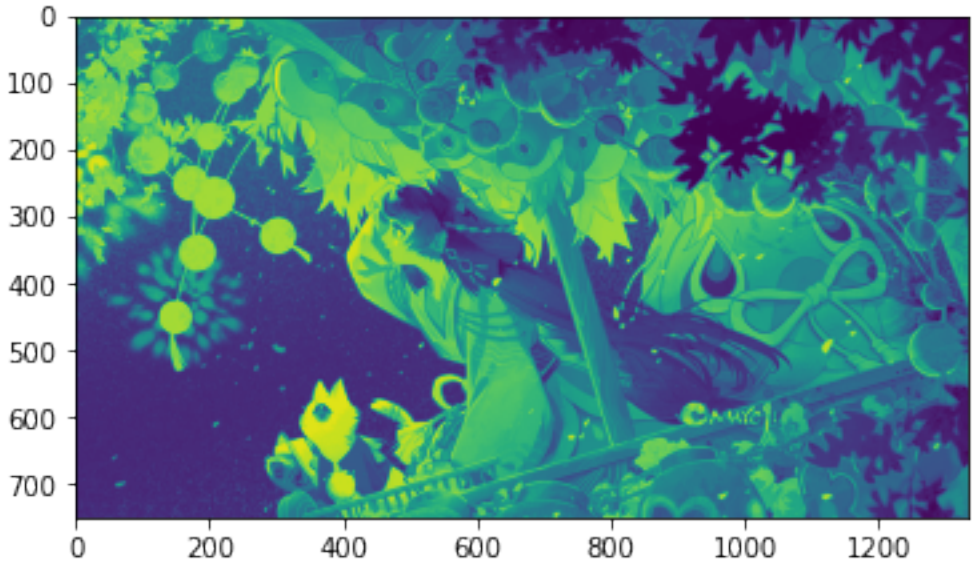
[5]:
```
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img_rgb)
```

[5]: <matplotlib.image.AxesImage at 0x7fa449b4d550>



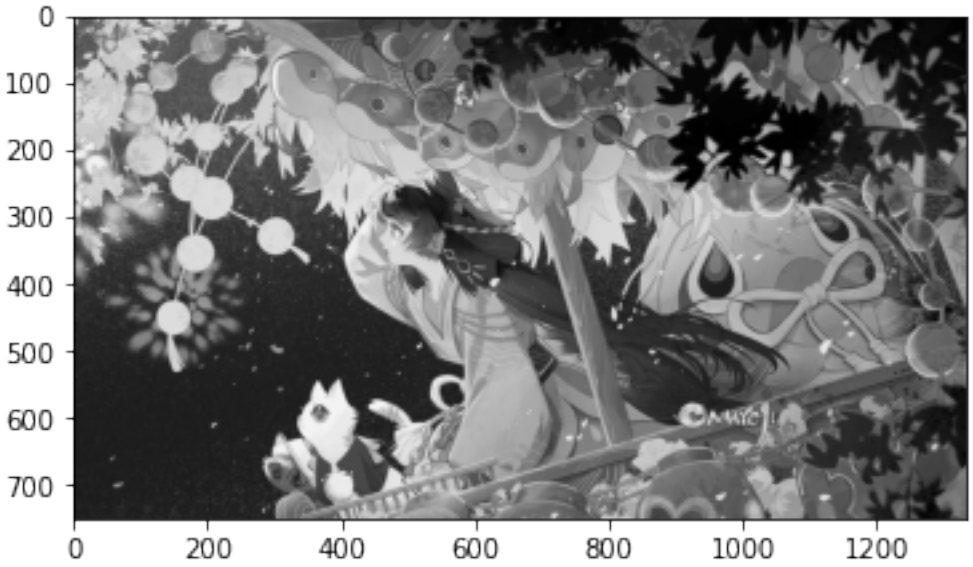
[6]:
```
img_gray = cv2.imread('../Figures/1. Onmyoji illustration - 1.png',cv2.
 ↪IMREAD_GRAYSCALE)
plt.imshow(img_gray)
```

[6]: <matplotlib.image.AxesImage at 0x7fa44a3ffdd0>



[7]:
```
img_gray = cv2.imread('../Figures/1. Onmyoji illustration - 1.png',cv2.
 ↪IMREAD_GRAYSCALE)
plt.imshow(img_gray,cmap='gray')
```

[7]: <matplotlib.image.AxesImage at 0x7fa448fc8910>



1.1 Resize Images

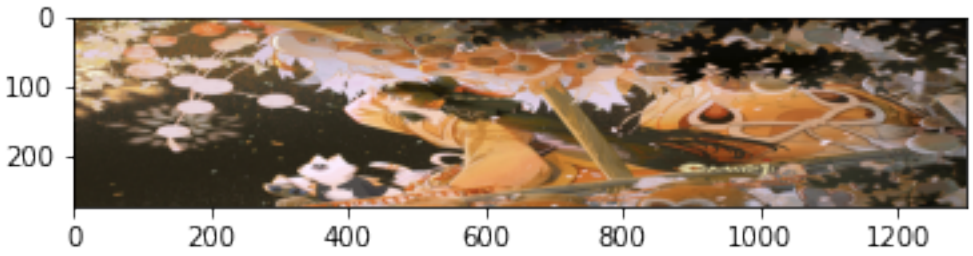
[8]:
```
img_rgb.shape
# width, height, color channels
```

[8]:
```
(750, 1334, 3)
```

[9]:
```
img =cv2.resize(img_rgb,(1300,275))
```

[10]:
```
plt.imshow(img)
```

[10]:
```
<matplotlib.image.AxesImage at 0x7fa4491f03d0>
```



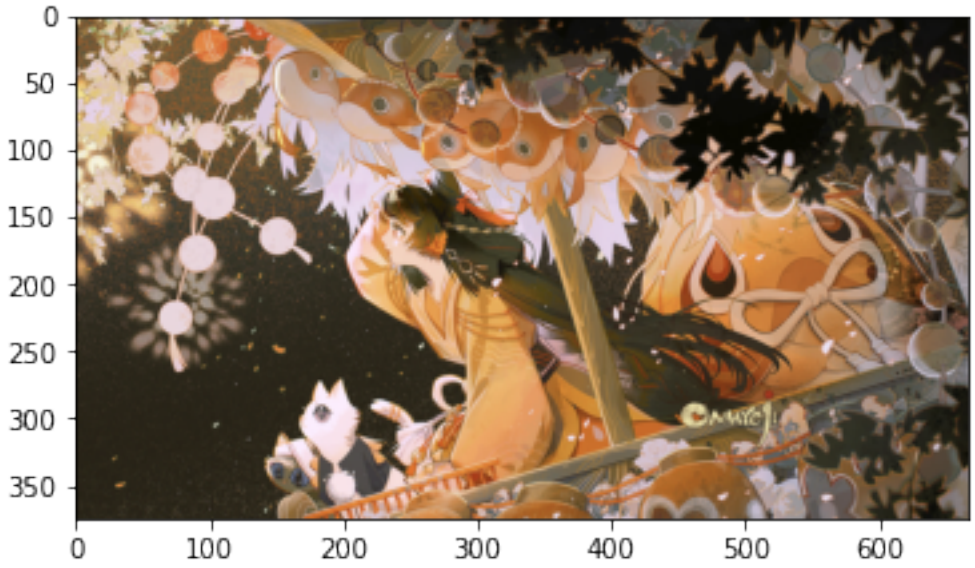
1.1.1 By ratio

[11]:
```
w_ratio = 0.5
h_ratio = 0.5
```

[12]:
```
new_img =cv2.resize(img_rgb,(0,0),img,w_ratio,h_ratio)
```

[13]:
```
plt.imshow(new_img)
```

[13]: <matplotlib.image.AxesImage at 0x7fa44a0e8b50>



[14]:
```
%matplotlib qt
plt.imshow(new_img)
```

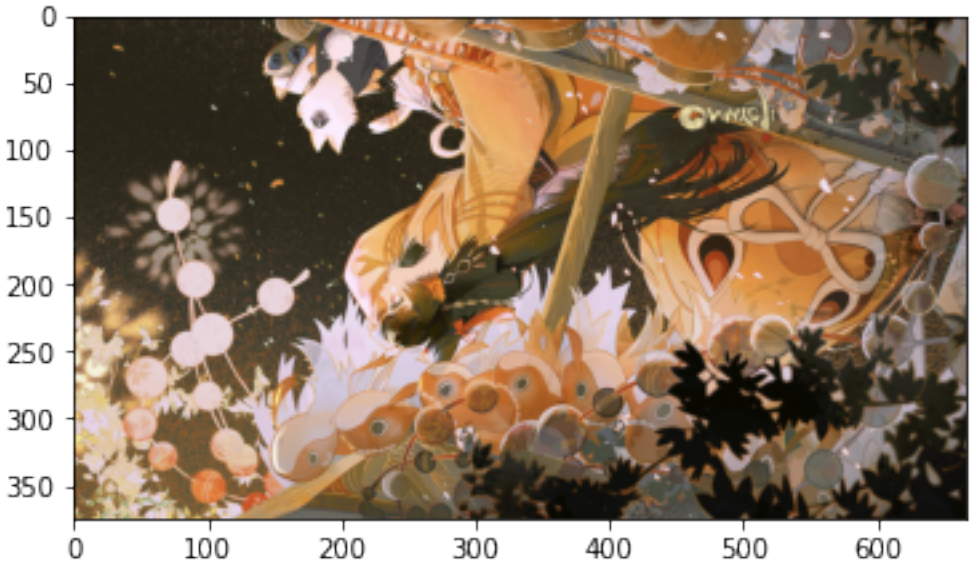
[14]: <matplotlib.image.AxesImage at 0x7fa44aa60f10>

1.1.2 Flipping Images

[15]:
```
%matplotlib inline
```

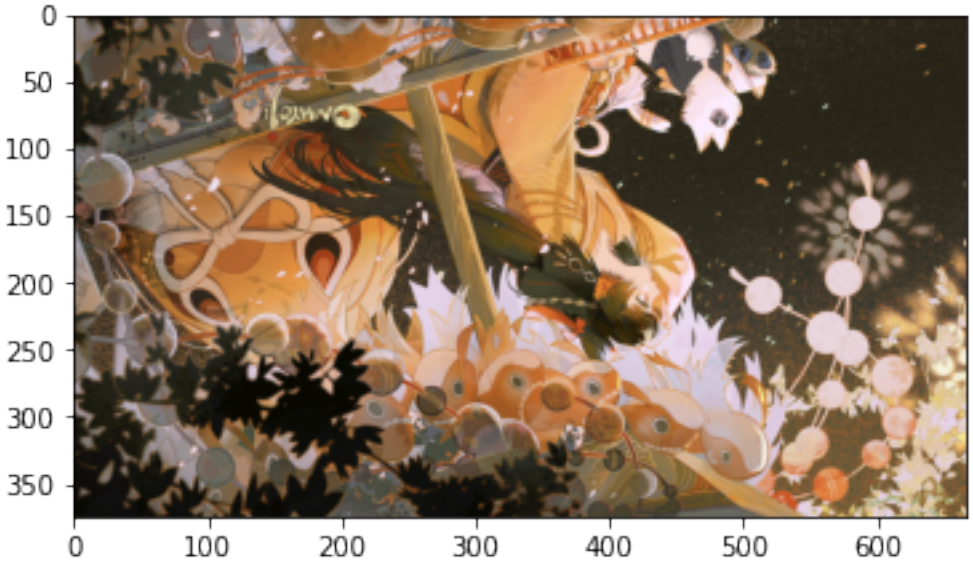
[16]:
```
# Along central x axis
new_img = cv2.flip(new_img,0)
plt.imshow(new_img)
```

[16]: <matplotlib.image.AxesImage at 0x7fa44c796c50>



[17]:
```
# Along central y axis
new_img = cv2.flip(new_img,1)
plt.imshow(new_img)
```

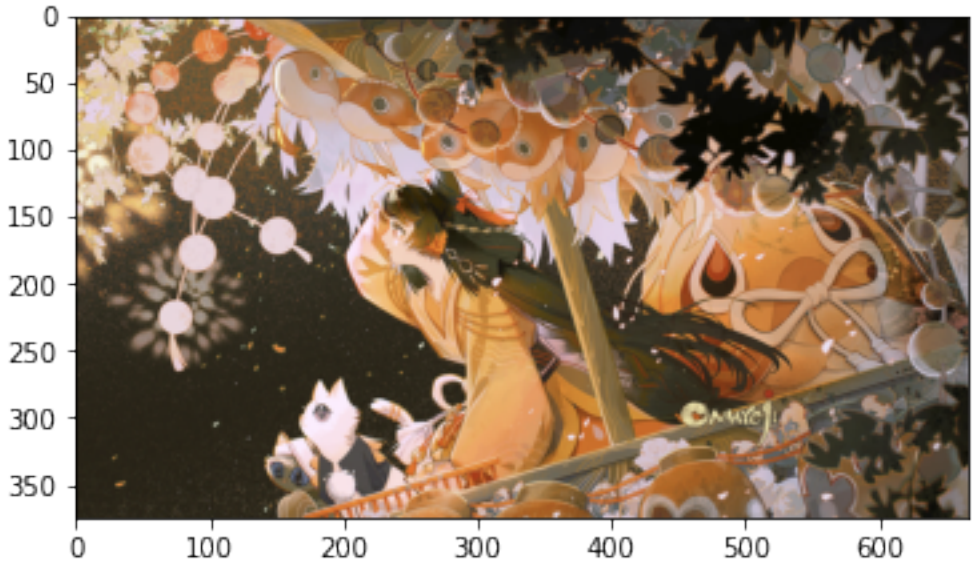
[17]: <matplotlib.image.AxesImage at 0x7fa44c8c3510>



[18]:
```
# Along both axis
new_img = cv2.flip(new_img,-1)
```

```
plt.imshow(new_img)
```

[18]: <matplotlib.image.AxesImage at 0x7fa44c336b10>



---

2 Saving Image Files

[19]:
```
type(new_img)
```

[19]: numpy.ndarray

[20]:
```
cv2.imwrite('../Outputs/my_new_picture.jpg',new_img)
```

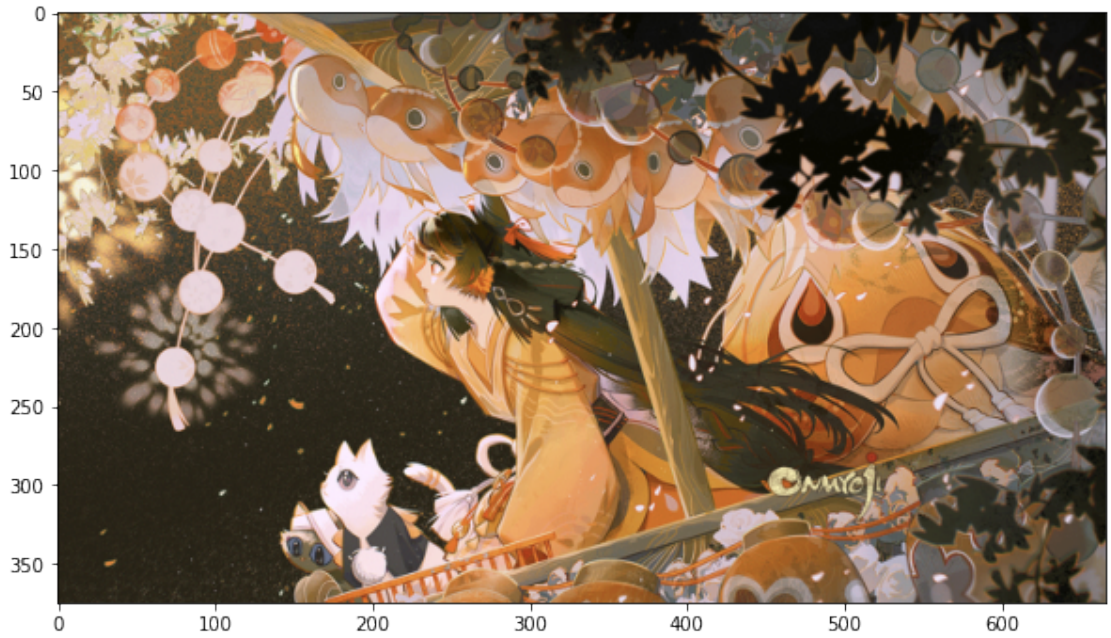
[20]: True

Keep in mind, the above stored the BGR version of the image.

2.1 Larger Displays in the Notebook

[21]:
```
fig = plt.figure(figsize=(10,8))
ax = fig.add_subplot(111)
ax.imshow(new_img)
```

[21]: <matplotlib.image.AxesImage at 0x7fa44bf6f7d0>



3 Drawing on Images

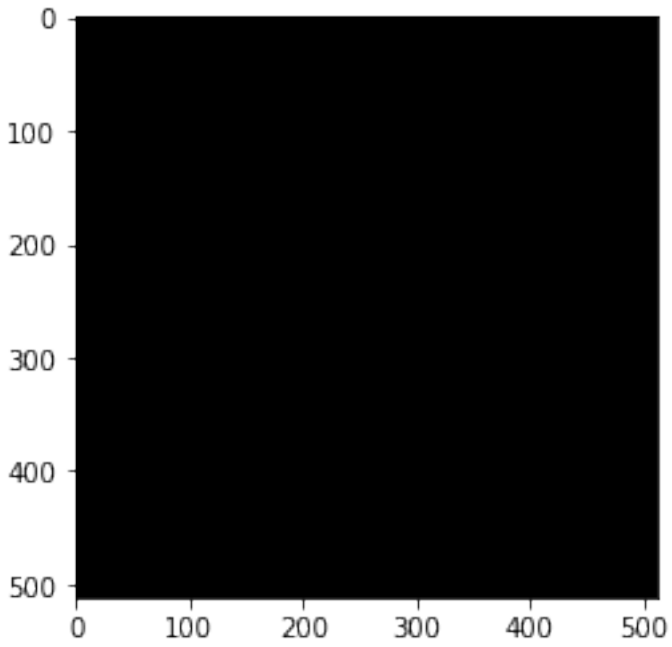
[22]:
```
blank_img = np.zeros(shape=(512,512,3),dtype=np.int16)
```

[23]:
```
blank_img.shape
```

[23]:
```
(512, 512, 3)
```

[24]:
```
plt.imshow(blank_img)
```

[24]:
```
<matplotlib.image.AxesImage at 0x7fa44caa9d10>
```



3.1 Shapes

3.1.1 Rectangles

- img Image.
- pt1 Vertex of the rectangle.
- pt2 Vertex of the rectangle opposite to pt1 .
- color Rectangle color or brightness (grayscale image).
- thickness Thickness of lines that make up the rectangle. Negative values, like #FILLED,mean that the function has to draw a filled rectangle.
- lineType Type of the line. See #LineTypes
- shift Number of fractional bits in the point coordinates.

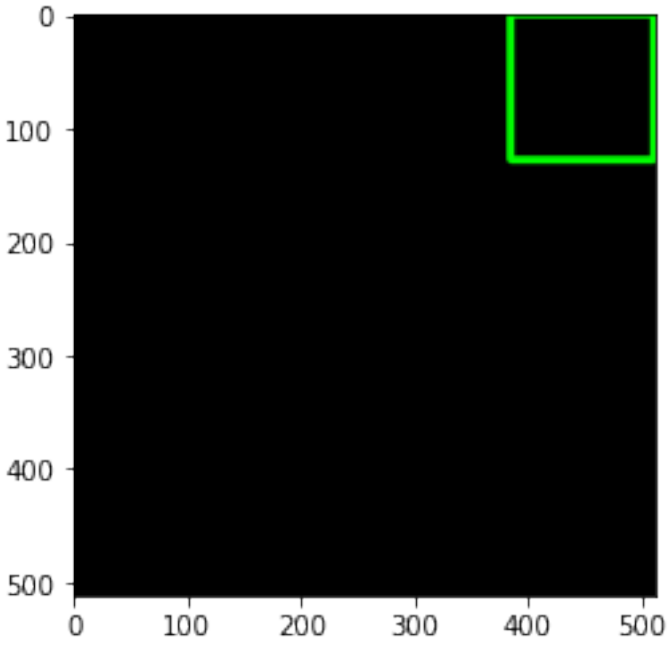
[25]:
```
# pt1 = top left
# pt2 = bottom right
cv2.rectangle(blank_img,pt1=(384,0),pt2=(510,128),color=(0,255,0),thickness=5)
```

[25]:
```
array([[[  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0],
        …,
        [  0, 255,   0],
        [  0, 255,   0],
        [  0, 255,   0]],

       [[  0,   0,   0],
```

```
        [  0,   0,   0],
        [  0,   0,   0],
        …,
        [  0, 255,   0],
        [  0, 255,   0],
        [  0, 255,   0]],

       [[  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0],
        …,
        [  0, 255,   0],
        [  0, 255,   0],
        [  0, 255,   0]],

       …,

       [[  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0],
        …,
        [  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0]],

       [[  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0],
        …,
        [  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0]],

       [[  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0],
        …,
        [  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0]]], dtype=int16)
```

[26]:
```
# cv2.rectangle(blank_img,pt1=(384,0),pt2=(510,128),color=(0,255,0))
plt.imshow(blank_img)
```

[26]:
```
<matplotlib.image.AxesImage at 0x7fa44cb79c50>
```


0
100
200
300
400
500
0
100
200
300
400
500

[ ]: