Welcome to the tutorial on Matplotlib ¶

In this document, we'll talk about the following:

- Matplotlib

Matplotlib

Matplotlib (https://matplotlib.org/) is a comprehensive library primarily used for creating static, animated, and interactive visualizations.
We generally import Matplotlib with the following line:

In [1]:
```
from matplotlib import pyplot as plt
```

We also include OpenCV and NumPy packages here:

In [2]:
```
import numpy as np
import cv2
```

Some frequently used functions are:

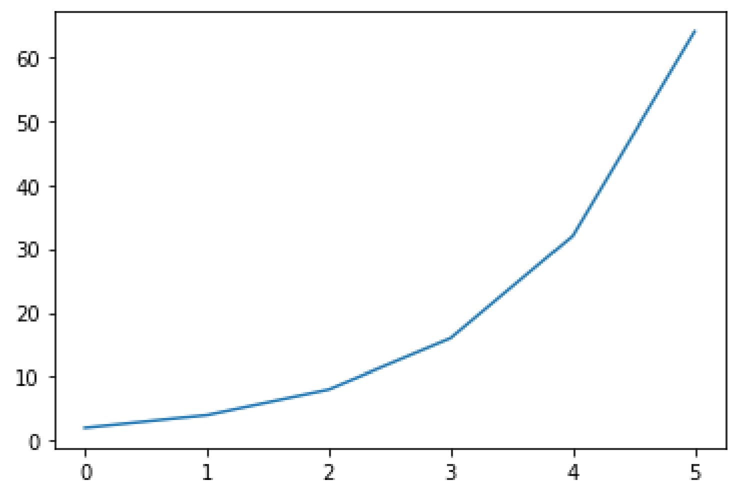
`plt.plot([x], y [,fmt])`

Parameters

x(Optional): X coordinates of the plot points
y: Y coordinates of the plot points
fmt(Optional): Format strings to format the look of the graph

In [3]:
```
plt.plot([2,4,8,16,32,64])
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x7f9c89e8d2b0>]
```



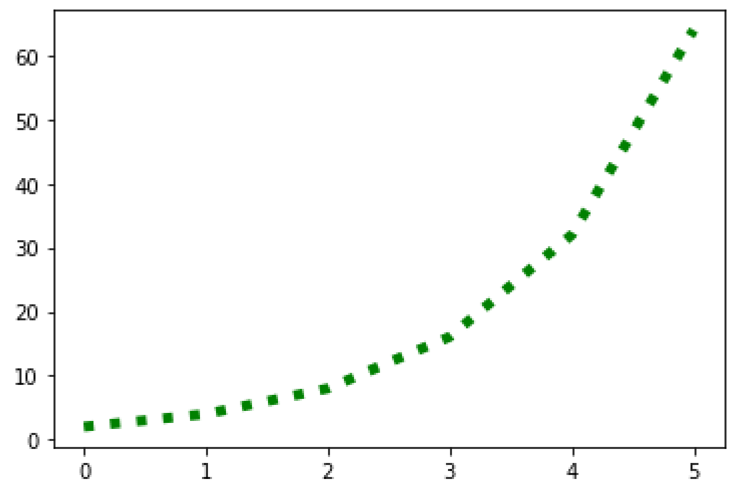
We can use Line2D properties for more control on the appearance. Some useful Line2D properties are:

Property	Description or Values
animated	`bool`
color	Color of the graph line
dash_capstyle	*'butt', 'round', 'projecting'*
dash_joinstyle	*'miter', 'round', 'bevel'*
data	`(2, N) array or two 1D arrays`
linestyle or ls	'-', '--', '-.', ':', ''
linewidth	`float`

You can see a list of all the properties by following this link (https://matplotlib.org/api/_as_gen/matplotlib.lines.Line2D.html#matplotlib.lines.Line2D).

```
In [4]: plt.plot([0,1,2,3,4,5], [2,4,8,16,32,64], color="green", linestyle=':', linewidth=5.0)
```

```
Out[4]: [<matplotlib.lines.Line2D at 0x7f9c81d64c70>]
```



`plt.xlabel(xlabel)`

Parameters

xlabel: Text for the X axis Label

`plt.ylabel(ylabel)`

Parameters

ylabel: Text for the Y axis Label

`plt.show()`

Displays all open figures.

`plt.imshow(src [,aspect,  cmap] )`

We will be often using NumPy arrays throughout this course. So it is helpful to visualize images using NumPy array. These functions will help us in this regard.

Parameters

src: source image in numpy array format
aspect (optional): Set the aspect ratio of the displayed image. Can be either of the following:

- equal - Ensures an aspect ratio of 1
- auto - The axes is kept fixed and the aspect is adjusted so that the data fit in the axes

cmap (optional): Set the Colormap instance. The some commonly used cmap value are:

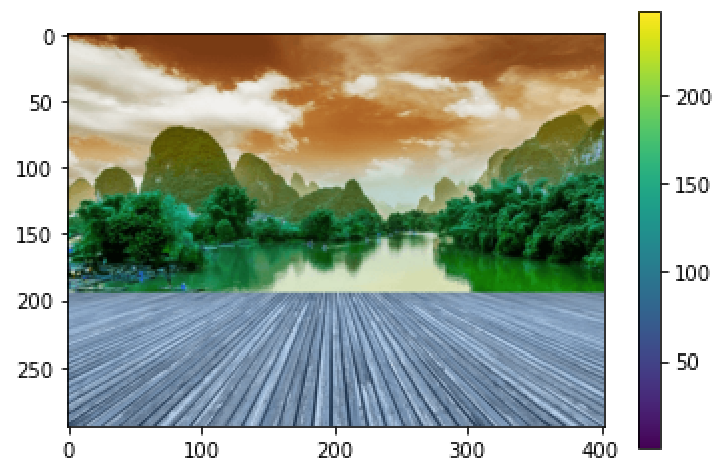
- "hot"
- "grey"
- "blues"
- "cool"

Full list of possible cmap values can be found by following [this link (https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html)](https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html).

Let's try loading an image through cv2 and displaying it.
Note: If you don't understand the `cv2` functions, worry not! They are explained in the respective Notebooks.

In [5]:
```
img = cv2.imread('./Assets/Scenery.png')
plt.imshow(img)
plt.colorbar()
```

Out[5]:
```
<matplotlib.colorbar.Colorbar at 0x7f9c81d01700>
```



Note: `plt.colorbar()` adds a color bar next to the plot. It's helpful to have an idea of what value a color represents.

`plt.savefig( fname [, dpi=None, orientation='portrait', quality,  format=None, transparent=False,]   )`

This function is to be used when an image is read using `cv2.imread()` and the color space is changed using `cv2.cvtColor()` . For an elaborate explanation please check the OpenCV notebook section `cv2.imwrite()`

Parameters

fname: File name of the saved image
dpi (optional): The resolution in dots per inch.
orientation (optional): *'landscape', 'portrait'*
quality (optional): The image quality, on a scale from 1 (worst) to 95 (best). Lower quality helps in reducing file size
format (optional): The file format, e.g. 'png', 'pdf', 'svg', etc, When this is not set, it is assumed from fname

Figures and Subplots

So far we've worked on a single plot at a time. But there are instances where we need to plot various plots next to each other to compare them or see the various outputs. Well, Matplotlib can do that too!! Let's check it out.

Some things we'll be using to plot multiple plots:

- Figure: You can think of figure as a Canvas. You can plot various things on a Figure. We'll be using it to add several subplots
- Subplot: In Real Life, we need to buy plots before we build a house. Similarly, before plot an image, we need to make subplots within the given figure.

`plt.figure( [num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None] )`

Parameters

num(Optional): `int` or `string` . A unique identifier for the figure.
figsize(Optional): `( float, float)` Width, height in inches.
dpi(Optional): `float` The resolution of the figure in dots-per-inch.
facecolor(Optional): The background color of the figure
edgecolor(Optional): The border color of the figure

figure **`.add_subplot( position )`**

Parameters

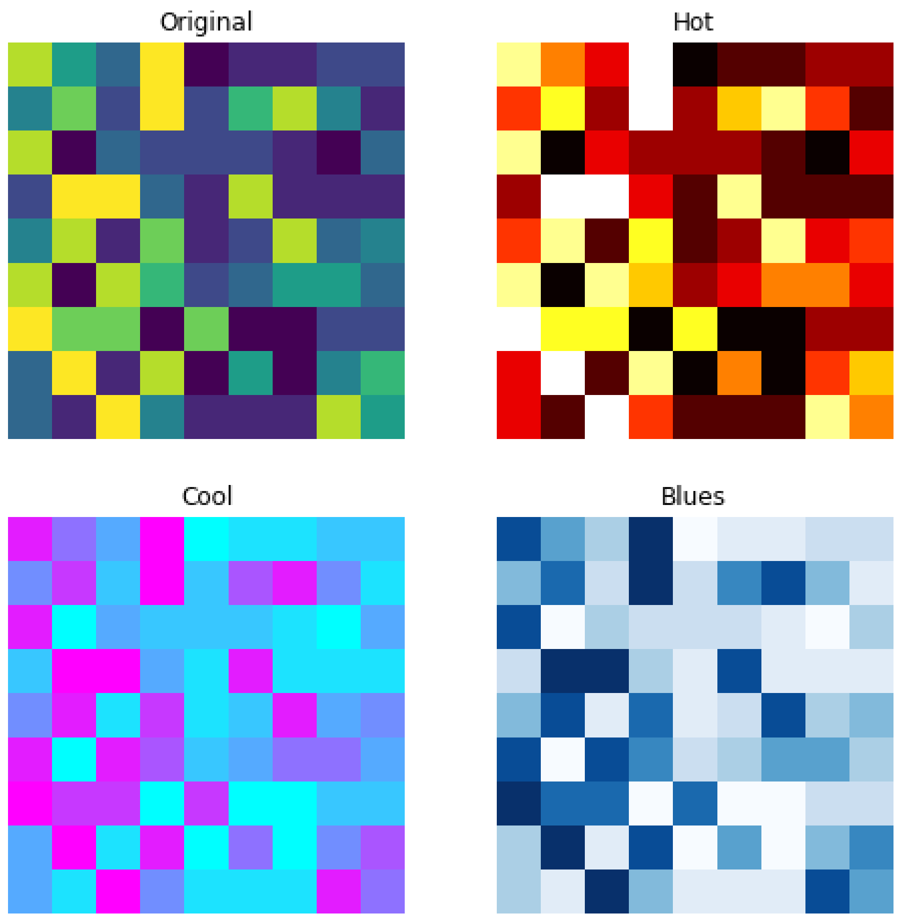
The position of the subplot is commonly passed in the following ways:

1. Three integers (nrows, ncols, index) where
   - nrows is number of rows and ncols is number of columns in the grid
   - index is position on the grid that the subplot will be placed in

2. A 3-digit integer. The digits are interpreted as if given separately as three single-digit integers. For example, if the integer given is `235` it is interpreted as `(2, 3, 5)` and the above rules apply.

Below some commands are written in a single line seperated by a semicolon `;`. This prevents the output of the function being displayed. The effect will be the same if written out in seperate lines.

In [6]:
```
fig=plt.figure(figsize=(8,8))
rows = 2
columns = 2


img = np.random.randint(10, size=(9, 9))

fig.add_subplot(rows, columns, 1)
plt.imshow(img); plt.axis('off'); plt.title("Original")

fig.add_subplot(rows, columns, 2)
plt.imshow(img, cmap="hot"); plt.axis('off'); plt.title("Hot")

fig.add_subplot(rows, columns, 3)
plt.imshow(img, cmap="cool"); plt.axis('off'); plt.title("Cool")

fig.add_subplot(rows, columns, 4)
plt.imshow(img, cmap="Blues"); plt.axis('off'); plt.title("Blues")

plt.show()
```


Original
Hot
Cool
Blues