Matplotlib Tutorial

I. Pendahuluan ¶

In [ ]:

```
%matplotlib inline
#%matplotlib notebook
#%matplotlib qt
```

https://matplotlib.org/gallery/style_sheets/style_sheets_reference.html (https://matplotlib.org/gallery/style_sheets/style_sheets_reference.html)

In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
style_list = ['default', 'classic'] + sorted(
        style for style in plt.style.available if style != 'classic')
style_list
```

In [ ]:

```
plt.style.use('default')
```

In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
```

1. Figure

In [ ]:

```
%matplotlib qt
#fig = plt.figure()  # figure kosong tanpa sumbu
#plt.show()
fig, ax_lst = plt.subplots(2, 3)  # figure dengan 4 sumbu
plt.show()
```

In [ ]:

```
import matplotlib
help(matplotlib.axes)
```

2. Plot Garis

In [ ]:

```
markers = {'.': 'point', ',': 'pixel', 'o': 'circle', 'v': 'triangle_down', '^': 'trian
gle_up', '<': 'triangle_left', '>': 'triangle_right', '1': 'tri_down', '2': 'tri_up',
'3': 'tri_left', '4': 'tri_right', '8': 'octagon', 's': 'square', 'p': 'pentagon', '*':
'star', 'h': 'hexagon1', 'H': 'hexagon2', '+': 'plus', 'x': 'x', 'D': 'diamond', 'd':
'thin_diamond', '|': 'vline', '_': 'hline', 'P': 'plus_filled', 'X': 'x_filled', 0: 'ti
ckleft', 1: 'tickright', 2: 'tickup', 3: 'tickdown', 4: 'caretleft', 5: 'caretright', 6
: 'caretup', 7: 'caretdown', 8: 'caretleftbase', 9: 'caretrightbase', 10: 'caretupbase'
, 11: 'caretdownbase', 'None': 'nothing', None: 'nothing', ' ': 'nothing', '': 'nothin
g'}
markers
```

In [ ]:

```
lineStyles = {'': '_draw_nothing', ' ': '_draw_nothing', '-': '_draw_solid', '--': '_dr
aw_dashed', '-.': '_draw_dash_dot', ':': '_draw_dotted', 'None': '_draw_nothing'}
lineStyles
```

In [ ]:

```
color = {'b' : 'blue', 'g' : 'green', 'r' : 'red', 'c' : 'cyan', 'm' : 'magenta', 'y':
'yellow', 'k':'black', 'w':'white'}
color
```

In [ ]:

```
%matplotlib qt
#contoh 1
fig = plt.figure()

x = np.linspace(0, 2, 100)
y = x*3

#plt.plot(4*x, label='linier_1') #plot dari data sumbu y
plt.plot(x, y, label='linear_2') #plot dari data sumbu y dan x

#plot dengan bentuk garis yang berbeda
plt.plot(x, x**2,'bo-',label='quadratic')
plt.plot(x, x**3,'r+',label='cubic', markersize=2)
plt.plot(x, x**4,'r--',label='cubic',linewidth=5,markersize=2)


plt.xlabel('x label')
plt.ylabel('y label')

plt.title("Simple Plot")

plt.legend() #menampilkan label dari setiap data
plt.grid(b=True, which='major', color='m', linestyle='--')

plt.show()
```

In [ ]:

```
#contoh dua dengan menggunakan subplot
x = np.arange(0, 10, 0.2)
y = np.sin(x)

fig, ax = plt.subplots(2,2)
#ax = [[b1k1,b1k2],[b2k1,b2k2]]
print(ax)
ax[0][0].plot(x, y,'bo-',label='quadratic')
ax[1][0].plot(y, x)
plt.show()
```

In [ ]:

```
#contoh tiga dengan membuat multiple plot dalam 1 figure
x = np.arange(0, 10, 0.2)
y = np.sin(x)


fig, ((ax1, ax2),(ax3, ax4)) = plt.subplots(2, 2)
ax1.plot(x, y)
ax2.plot(x, x**2)
ax2.grid(True)
ax3.plot(x, x**2, 'b+')
ax4.plot(x, x**4, linewidth=5)

plt.show()
```

In [ ]:

```
#contoh empat, mesaukkan nilai x & y melalui objek data
fig = plt.figure()
data = {'data_h': np.arange(0, 10, 0.2)
        }
data['data_v'] = np.cos(data['data_h'])

plt.plot('data_h', 'data_v','r>', data=data)
plt.xlabel('entry a')
plt.ylabel('entry b')
plt.show()
```

3. Plot Scatter

In [ ]:

```
 %matplotlib qt
import numpy as np
import matplotlib.pyplot as plt

fig,(ax) = plt.subplots(1,1)

# Randam state 0-(2^32-1) menginisiai generator angka pseudorandom
np.random.seed(1000)

N = 2
x = [50,70]
#print(x)
y = [100,200]
colors = np.random.rand(N)
print(colors)
#print(colors)
area = (30 * np.random.rand(N))**2
print(area)
#s = ukuran marker, c= warna, alpha= transparansi


#b = ax.scatter(x, y, s='200', c=''r'', alpha=0.6, marker ='o', cmap = 'Blues')
ax.scatter(x, y, s=area, c=colors, alpha=1, marker ='<')
#fig.colorbar(b)
plt.grid(True)
plt.show()
```

In [ ]:

```
help(plt.imshow)
```

In [ ]:

```
import matplotlib
help(plt.scatter)
```

In [ ]:

```
# membuat scatter dari object data
fig = plt.figure()
np.random.seed(10000)
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

plt.scatter('a', 'b', c='c', s='d', data=data)
plt.xlabel('entry a')
plt.ylabel('entry b')
plt.show()
```

4. Bar

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure()
N = 20
ind = np.arange(N)
y = ind*5
width = 0.35

# ind= sumbu x, y, lebar bar
plt.bar(ind, y, width)

# sumbux x
plt.xticks(ind)
plt.show()
```

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure()
N = 5
menMeans = (20, 35, 30, 35, 27)
womenMeans = (25, 32, 34, 20, 25)
menStd = (2, 3, 4, 1, 2)
womenStd = (3, 5, 2, 3, 3)
ind = np.arange(N)    # sumbu x
width = 0.2

# indeks, nilai, yerr= error sumbu y
p1 = plt.bar(ind, menMeans, width, yerr=menStd)
p2 = plt.bar(ind+0.2, womenMeans, width, yerr=womenStd)

plt.ylabel('Scores')
plt.title('Scores by group and gender')
plt.xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))
plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('Men', 'Women'))

plt.show()
```

5. Histogram

In [ ]:

```
mu, sigma = 100, 15
x = mu + sigma *np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=1, facecolor='g', alpha=0.6)



plt.plot(bins[1:],n)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(50, .025, '$\mu=100, \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

6. Multi plot

In [ ]:

```
names = ['group_a', 'group_b', 'group_c']
values = [1, 10, 100]

plt.figure(figsize=(10, 3))

plt.subplot(221)
plt.bar(names, values)
plt.subplot(222)
plt.scatter(names, values)
plt.subplot(223)
plt.plot(names, values)
plt.suptitle('Categorical Plotting')
plt.show()
```

6. Error bar

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

# contoh data
x = np.arange(0.1, 4, 0.5)
y = np.exp(-x)

plt.figure()
plt.errorbar(x, y,fmt='>--',color='b',ecolor='r', xerr=0.2, yerr=0.4)
plt.title("Simplest errorbars, 0.2 in x, 0.4 in y")
```

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

# contoh data
x = np.arange(0.1, 4, 0.5)
y = np.exp(-x)

# error data
yerr = 0.1 + 0.2*np.sqrt(x)
xerr = 0.1 + yerr

fig, axs = plt.subplots(nrows=2, ncols=2, sharex=True)
ax = axs[0,0]
ax.errorbar(x, y, yerr=yerr, fmt='o')
ax.set_title('Vert. symmetric')

# mengurangi jumlah angka pada sumbu
ax.locator_params(nbins=4)

ax = axs[0,1]
ax.errorbar(x, y, xerr=xerr, fmt='o')
ax.set_title('Hor. symmetric')

ax = axs[1,0]
ax.errorbar(x, y, yerr=[yerr, 2*yerr], xerr=[xerr, 2*xerr], fmt='--o')
ax.set_title('H, V asymmetric')

ax = axs[1,1]
ax.set_yscale('log')

# seluruh nilai harus positif
ylower = np.maximum(1e-2, y - yerr)
yerr_lower = y - ylower

ax.errorbar(x, y, yerr=[yerr_lower, 2*yerr], xerr=xerr,
            fmt='o', ecolor='g',color = 'r', capthick=2)
ax.set_title('Mixed sym., log y')

fig.suptitle('Variable errorbars')

plt.show()
```

In [ ]:

```

```

2D plot

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib qt

cmaps = [('Perceptually Uniform Sequential', [
            'viridis', 'plasma', 'inferno', 'magma']),
         ('Sequential', [
            'Greys', 'Purples', 'Blues', 'Greens', 'Oranges', 'Reds',
            'YlOrBr', 'YlOrRd', 'OrRd', 'PuRd', 'RdPu', 'BuPu',
            'GnBu', 'PuBu', 'YlGnBu', 'PuBuGn', 'BuGn', 'YlGn']),
         ('Sequential (2)', [
            'binary', 'gist_yarg', 'gist_gray', 'gray', 'bone', 'pink',
            'spring', 'summer', 'autumn', 'winter', 'cool', 'Wistia',
            'hot', 'afmhot', 'gist_heat', 'copper']),
         ('Diverging', [
            'PiYG', 'PRGn', 'BrBG', 'PuOr', 'RdGy', 'RdBu',
            'RdYlBu', 'RdYlGn', 'Spectral', 'coolwarm', 'bwr', 'seismic']),
         ('Qualitative', [
            'Pastel1', 'Pastel2', 'Paired', 'Accent',
            'Dark2', 'Set1', 'Set2', 'Set3',
            'tab10', 'tab20', 'tab20b', 'tab20c']),
         ('Miscellaneous', [
            'flag', 'prism', 'ocean', 'gist_earth', 'terrain', 'gist_stern',
            'gnuplot', 'gnuplot2', 'CMRmap', 'cubehelix', 'brg', 'hsv',
            'gist_rainbow', 'rainbow', 'jet', 'nipy_spectral', 'gist_ncar'])]


nrows = max(len(cmap_list) for cmap_category, cmap_list in cmaps)
gradient = np.linspace(0, 1, 256)
gradient = np.vstack((gradient, gradient))


def plot_color_gradients(cmap_category, cmap_list, nrows):
    fig, axes = plt.subplots(nrows=nrows)
    fig.subplots_adjust(top=0.95, bottom=0.01, left=0.2, right=0.99)
    axes[0].set_title(cmap_category + ' colormaps', fontsize=14)

    for ax, name in zip(axes, cmap_list):
        ax.imshow(gradient, aspect='auto', cmap=plt.get_cmap(name))
        pos = list(ax.get_position().bounds)
        x_text = pos[0] - 0.01
        y_text = pos[1] + pos[3]/2.
        fig.text(x_text, y_text, name, va='center', ha='right', fontsize=10)

    # Turn off *all* ticks & spines, not just the ones with colormaps.
    for ax in axes:
        ax.set_axis_off()


for cmap_category, cmap_list in cmaps:
    plot_color_gradients(cmap_category, cmap_list, nrows)

plt.show()
```

In [ ]:

```
import matplotlib
import matplotlib.pyplot as plt
from matplotlib.colors import BoundaryNorm
from matplotlib.ticker import MaxNLocator
import numpy as np


# resolusi sumbu x dan y
dx, dy = 0.05, 0.05

# membuat grid 2d
y, x = np.mgrid[slice(1, 5 + dy, dy),
                slice(1, 5 + dx, dx)]

z = np.sin(x)**10 + np.cos(10 + y*x) * np.cos(x)


# menghapus nilai baris terakhir dan kolom terakhir
z = z[:-1, :-1]
levels = MaxNLocator(nbins=15).tick_values(z.min(), z.max())
print(levels)
# memilih color map
# menormalisasi berdasarkan level
cmap = plt.get_cmap('Spectral')
norm = BoundaryNorm(levels, ncolors=cmap.N, clip=True)

fig, (ax0, ax1) = plt.subplots(nrows=2)

im = ax0.pcolormesh(x, y, z, cmap=cmap, norm=norm)
print(x.shape,y.shape,z.shape)

fig.colorbar(im, ax=ax0)
ax0.set_title('pcolormesh with levels')


# contour berbasis poin center dari sumbu x dan y

cf = ax1.contourf(x[:-1, :-1] + dx/2.,
                  y[:-1, :-1] + dy/2., z, levels=levels,
                  cmap=cmap)

fig.colorbar(cf, ax=ax1)
ax1.set_title('contourf with levels')

# mengatur layout agar tidak berimpit

fig.tight_layout()

plt.show()
```

In [ ]:

```
def fungsi(x,y=5,z=7):
    return x+y+z

print(fungsi(1))
```

3D Plot

In [ ]:

```
%matplotlib qt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np

fig = plt.figure()
ax = fig.gca(projection='3d')

# membuat data
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)


# Plot the surface.
surf = ax.plot_surface(X, Y, Z, cmap=cm.Purples)

# Customize the z axis.
ax.set_zlim(-1.01, 1.01)

# jumlah z yang ditampilkan
ax.zaxis.set_major_locator(LinearLocator(5))
# menampilan bilangan desimal
ax.zaxis.set_major_formatter(FormatStrFormatter('%.01f'))

# menampilkan color bar
fig.colorbar(surf, shrink=0.4, aspect=5)

plt.show()
```

Animasi

Untuk dapat menyimpan animasi video sebelumnya disarankan menginstall FFmpeg. Ikuti petunjuk di bawah ini: https://www.wikihow.com/Install-FFmpeg-on-Windows (https://www.wikihow.com/Install-FFmpeg-on-Windows)

In [ ]:

```
%matplotlib qt
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

fig, ax = plt.subplots()

x = np.arange(0, 2*np.pi, 0.01)
line, = ax.plot(x, np.sin(x))


def init(): #fungsi awal
    line.set_ydata([np.nan] * len(x))
    return line,


def animate(i):
    line.set_ydata(np.sin(x + i / 100))  # update the data.
    return line,

ani = animation.FuncAnimation(
    fig, animate, init_func=init, interval=2,
    blit=True, save_count=50)

# menyimpan video.

# ani.save("movie.mp4")


plt.show()
```

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d.axes3d as p3
import matplotlib.animation as animation


np.random.seed(19680801)


def Gen_RandLine(length, dims=2):
    """
    Create a line using a random walk algorithm

    length is the number of points for the line.
    dims is the number of dimensions the line has.
    """
    lineData = np.empty((dims, length))
    lineData[:, 0] = np.random.rand(dims)
    for index in range(1, length):
        # scaling the random numbers by 0.1 so
        # movement is small compared to position.
        # subtraction by 0.5 is to change the range to [-0.5, 0.5]
        # to allow a line to move backwards.
        step = ((np.random.rand(dims) - 0.5) * 0.1)
        lineData[:, index] = lineData[:, index - 1] + step

    return lineData


def update_lines(num, dataLines, lines):
    for line, data in zip(lines, dataLines):
        # NOTE: there is no .set_data() for 3 dim data...
        line.set_data(data[0:2, :num])
        line.set_3d_properties(data[2, :num])
    return lines

# Attaching 3D axis to the figure
fig = plt.figure()
ax = p3.Axes3D(fig)

# Fifty lines of random 3-D lines
data = [Gen_RandLine(25, 3) for index in range(50)]

# Creating fifty line objects.
# NOTE: Can't pass empty arrays into 3d version of plot()
lines = [ax.plot(dat[0, 0:1], dat[1, 0:1], dat[2, 0:1])[0] for dat in data]

# Setting the axes properties
ax.set_xlim3d([0.0, 1.0])
ax.set_xlabel('X')

ax.set_ylim3d([0.0, 1.0])
ax.set_ylabel('Y')

ax.set_zlim3d([0.0, 1.0])
ax.set_zlabel('Z')

ax.set_title('3D Test')
```

```
# Creating the Animation object
line_ani = animation.FuncAnimation(fig, update_lines, 25, fargs=(data, lines),
                                   interval=50, blit=False)

plt.show()
```

In [ ]:

```
%matplotlib qt
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib import style

fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)

def animate(i):
    graph_data = open('data.txt','r').read()
    lines = graph_data.split('\n')
    xs = []
    ys = []
    for line in lines:
        if len(line) > 1:
            x, y = line.split(',')
            xs.append(float(x))
            ys.append(float(y))
    ax1.clear()
    ax1.plot(xs, ys)

ani = animation.FuncAnimation(fig, animate, interval=1000)
plt.show()
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
import numpy as np
print('numpy: '+np.version.full_version)
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation
import matplotlib
print('matplotlib: '+matplotlib.__version__)
```

In [ ]:

```
N = 150 # Meshsize
fps = 10 # frame per sec
frn = 50 # frame number of the animation

x = np.linspace(-4,4,N+1)
x, y = np.meshgrid(x, x)
zarray = np.zeros((N+1, N+1, frn))

f = lambda x,y,sig : 1/np.sqrt(sig)*np.exp(-(x**2+y**2)/sig**2)

for i in range(frn):
    zarray[:,:,i] = f(x,y,1.5+np.sin(i*2*np.pi/frn))
```

In [ ]:

```
def update_plot(frame_number, zarray, plot):
    plot[0].remove()
    plot[0] = ax.plot_surface(x, y, zarray[:,:,frame_number], cmap="magma")

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

plot = [ax.plot_surface(x, y, zarray[:,:,0], color='0.75', rstride=1, cstride=1)]
ax.set_zlim(0,1.1)
ani = animation.FuncAnimation(fig, update_plot, frn, fargs=(zarray, plot), interval=100
0/fps)
plt.show()
```

In [ ]:

```

```