Matplotlib

Matplotlib is a commonly used libraries, that provide various tools for data visualization in Python. Matplotlib is based on NumPy.

Matplotlib works on ndarrays, but also on lists and tuples of numbers, and pandas objects.

Import Matplotlib, its submodule Pyplot

```
import matplotlib.pyplot as plt
```

There are two Matplotlib Interfaces: the Object-Oriented Interface and the Functional Interface.
In this course we will cover the Object-Oriented Interface

Object Oriented Interface

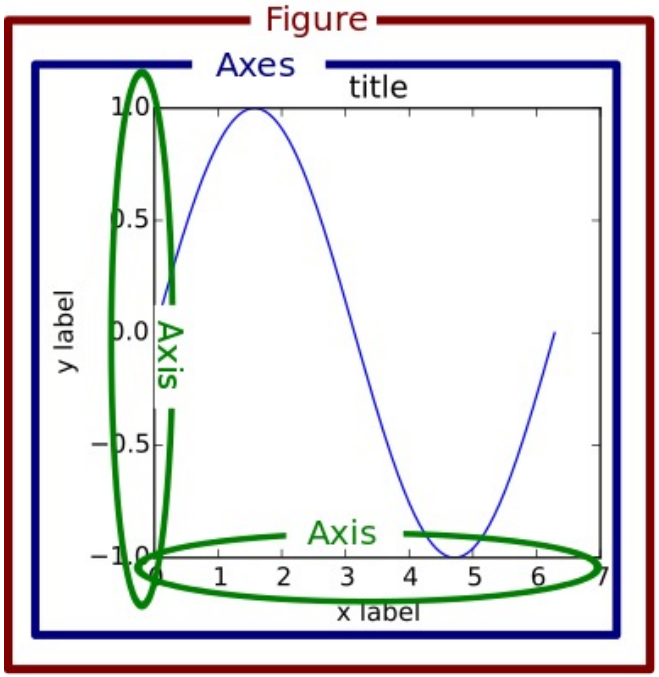
```
fig, ax = plt.subplots() #creates one Figure Object and one Axes Object

Apply methods to the Axes Object
#some plotting methods
    ax.plot()
    ax.bar()

#Label the axis:
    ax.set_xlabel()
    ax.set_ylabel()

# Show the grid:
     ax.grid()

#Setting x and y limits
    ax.set_xlim()
    ax.set_ylim()

#Setting x and y ticks
    ax.set_yticks()
    ax.set_xticks()

#Show the legend
   ax.legend()
```

Show the graph:

```
plt.show()
```

Apply methods to the fig object.

```
fig.savefig('figurename.png') #save figure in png
```



Do not confuse the x axis, and y axis with the Axes Object. The x axis and y axis are elements of the Axes Object.

Object Oriented Interface

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

x = np.arange(0, 10, 0.1)
y = np.sin(np.pi * x) + x


fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_xlabel("x")
ax.set_ylabel("y")
```

Functional Interface

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

x = np.arange(0,10,0.1)
y = np.sin(np.pi*x) + x


plt.figure()
plt.plot(x, y)
plt.xlabel("x")
plt.ylabel("y")
```

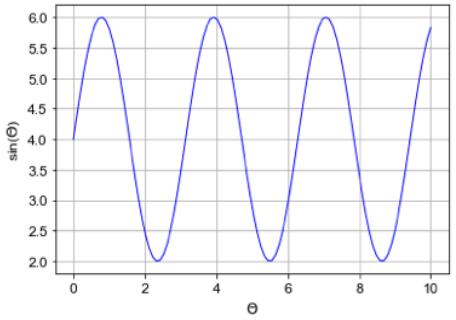
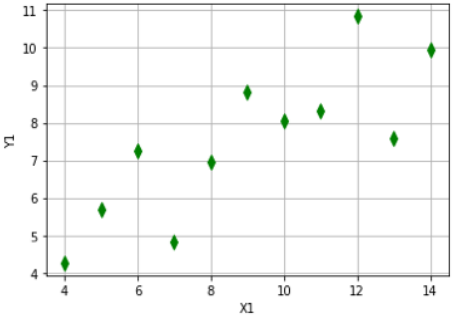
OO Interface useful links:
https://matplotlib.org/stable/gallery/showcase/anatomy.html

There are very helpful Matplotlib Cheatsheets and Handouts you can use:
https://matplotlib.org/cheatsheets/

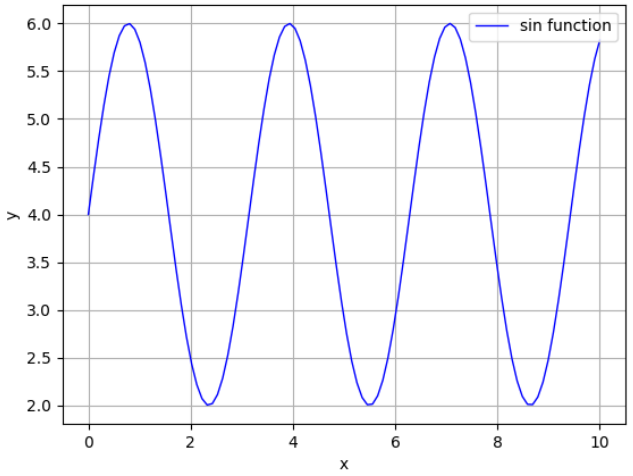
One useful handout for beginners can be found here:
https://matplotlib.org/cheatsheets/_images/handout-beginner.png

The plot() method

https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.plot.html

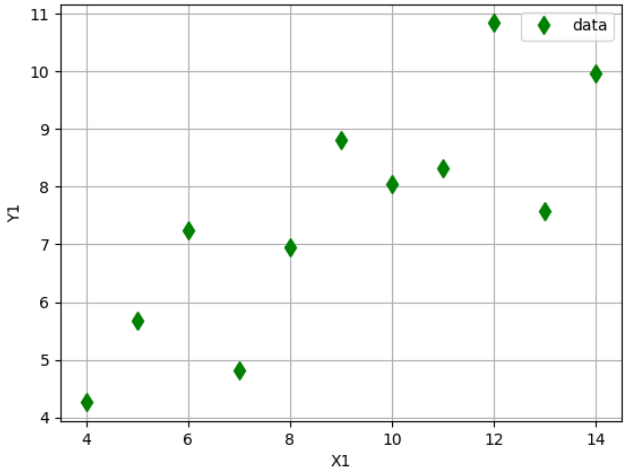
Axes method	plot type	use	figure plot
`ax.plot(x, y, '-b')`	Line plot	track changes over time, visualize mathematical functions	
`ax.plot(x, y, 'dg')`	plot with markers – scatterplot	Visualize possible relationships between two parameters, Visualize experimental data	

line plot




```
# make data
x = np.linspace(0, 10, 100)
y = 4 + 2 * np.sin(2 * x)

fig, ax = plt.subplots() #Create one Figure object fig and one Axes
object ax

#apply methods to the Axes object ax
ax.plot(x, y, '-b', linewidth=1, label='sin function') #blue solid
line is defined by the string '-b'

ax.set_xlabel('x')  #label x axis
ax.set_ylabel('y')  #label y axis

ax.grid()   #show the grid
ax.legend() #show the legend. Text is reported in the label of plot
plt.show()  #show the plot. Notice show is applied to plt.
```

scatter plot




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


x=np.array([10.0, 8.0, 13.0, 9.0, 11.0, 14.0, 6.0, 4.0, 12.0, 7.0, 5.0])
y=np.array([8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82,
5.68])


fig, ax = plt.subplots()
ax.plot(x, y, 'dg', markersize=8, label='data') #string 'dg' sets a green
diamond marker
ax.set_xlabel("X1")
ax.set_ylabel("Y1")

ax.grid()
ax.legend()
plt.show()
```

Specify line, marker, color and type in plot()

```
ax.plot(x, y, 'ob-', linewidth=1, markersize=8) #example
```

A **format string** is a string type containing characters that specify markers, lines and colors.
A **format string** is given by **'marker line color'** #each of them is optional

Example format strings:

```
'og'    # green circles
'-b'    # blue solid line
'--'    # dashed line with default color
'^k:'   # black triangle up markers connected by a dotted line
```

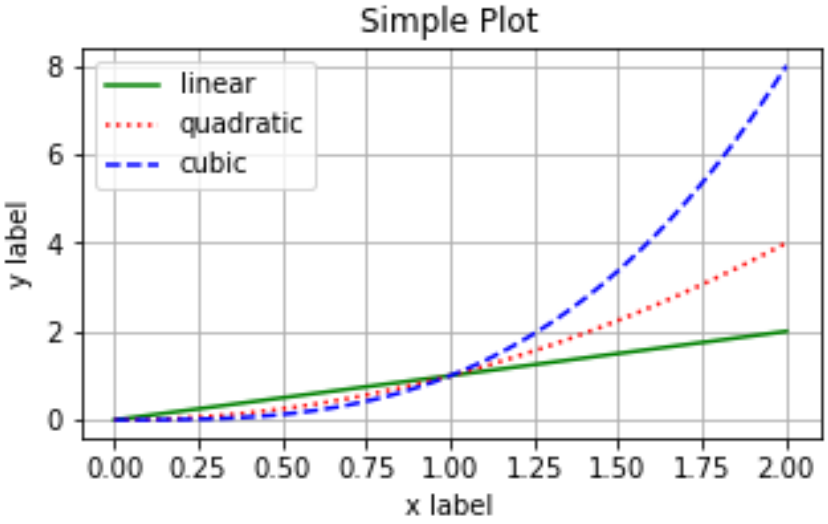
Here we report some characters for the format string

Markers		Line Styles		color	
.	point marker	-	solid line style	b	blue
o	circle marker	--	dashed line style	g	green
v	triangle down marker	-.	dash-dot line style	r	red
s	square marker	:	dotted line style	k	black
*	star marker				
D	diamond marker				

https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.plot.html

Multiple plots on the same Axes

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 2, 100)
```




```
fig, ax = plt.subplots(figsize=(5, 2.7)) # figure size in inches
ax.plot(x, x, '-g', label='linear')      #Plot some data on the Axes.
ax.plot(x, x**2, ':r', label='quadratic') #Plot more data on the same Axes
ax.plot(x, x**3, '--b', label='cubic') #Plot more data on the same Axes.

ax.set_xlabel('x label')
ax.set_ylabel('y label')


ax.set_title("Simple Plot")
ax.legend() #Add a legend
ax.grid()
plt.show()
```

set x and y limits and x and y ticks

```
ax.set_xlim(0, 12)   #sets x axis limit in the data coordinate
ax.set_ylim(0, 12)

ax.set_yticks(np.arange(0,13)) #setting x ticks, array of
tick's location

ax.set_xticks(np.arange(0,13))
```

set the figure size, and save the figure

```
fig, ax = plt.subplots(figsize=(5, 2.7)) #figure size in inches
```

```
fig.savefig('test', dpi=300) #saves figure with 300 dpi, by
default as .png
```

```
image_format = 'eps' # e.g .png, .svg, etc.
image_name = 'myimage.eps'

fig.savefig(image_name, format=image_format, dpi=1200)
```

Write Mathematical Expressions and Greek Symbols

You can use a subset of TeX markup in any Matplotlib text string by placing it inside a pair of dollar signs ($).

Any text element can use math text. You should use raw strings (precede the quotes with an 'r'), and surround the math text with dollar signs ($), as in TeX

```
ax.set_title(r'$\alpha > \beta$')
```

$\alpha > \beta$

https://matplotlib.org/stable/users/explain/text/mathtext.html

rcParams to change default settings – e.g. change font size, font name

https://matplotlib.org/stable/users/explain/customizing.html

rcParams is a like-dictionary type, storing different settings.

The see the default settings type:

```
import matplotlib.pyplot as plt
print(plt.rcParams)
```

```
#the rcParams must be placed before the plt.subplots() to work
```

To change setting, overwrite a value, like in a dictionary

```
plt.rcParams['lines.linewidth'] = 2
plt.rcParams['lines.linestyle'] = '--'
plt.rcParams['font.size']=12
plt.rcParams['font.family']='Ariel'
plt.rcParams['figure.subplot.wspace']= 0.4 #set width of the
padding between subplots

plt.rcParams['figure.subplot.hspace']= 0.3 #set the height of
the padding between subplots
```

Set the legend

There are different ways to make a legend, and here we will go over one of them, which is the automatic detection of elements to be shown in the legend taken from the label parameter of the plotting methods.

```
ax.plot(x, x, '-g', label='linear')
ax.legend()    # detect the string of the label parameter of
the plot method and construct the legend.
```

In this example, we have multiple plotting methods on the same Axes.

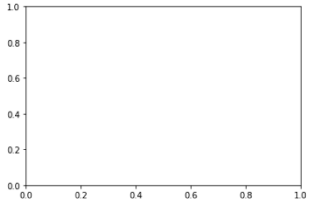
```
ax.plot(x, x, '-g', label='linear')
ax.plot(x, x**2, ':r' label='quadratic')
ax.plot(x, x**3, '--b', label='cubic')
ax.legend()    # detect each string of each label in the plot
methods and construct the legend.
```

subplots

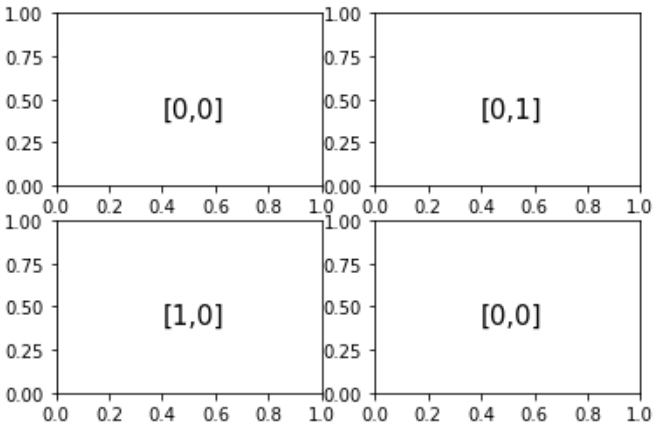
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html

```
plt.subplots(nrows, ncols) # default nrows=1 and ncols=1
```

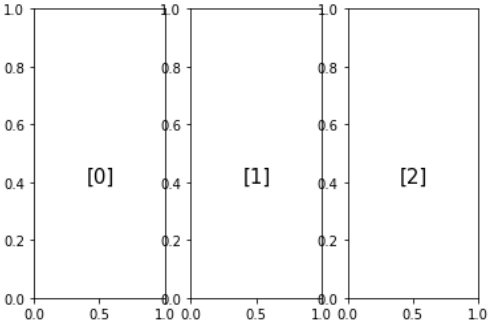
```
fig, ax = plt.subplots() # make a single Axes object
```



```
#make a 2x2 matrix of Axes Objects,
referenced by axs
fig, axs = plt.subplots(2, 2)
axs[0,0].plot()  #apply to Axes [0,0]
axs[0,0].set_xlabel()

axs[0,1].plot() #apply to Axes [0,1]
```

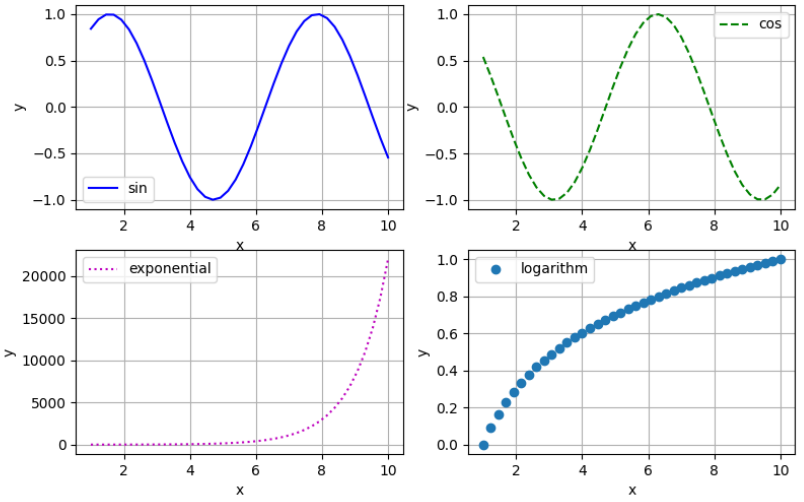



```
#make a row vector of 3 Axes Objects,
referenced by axs
fig, axs= plt.subplots(1, 3)
axs[0].bar() # apply to Axes [0]
axs[1].bar() # apply to Axes [1]
```



Subplot example

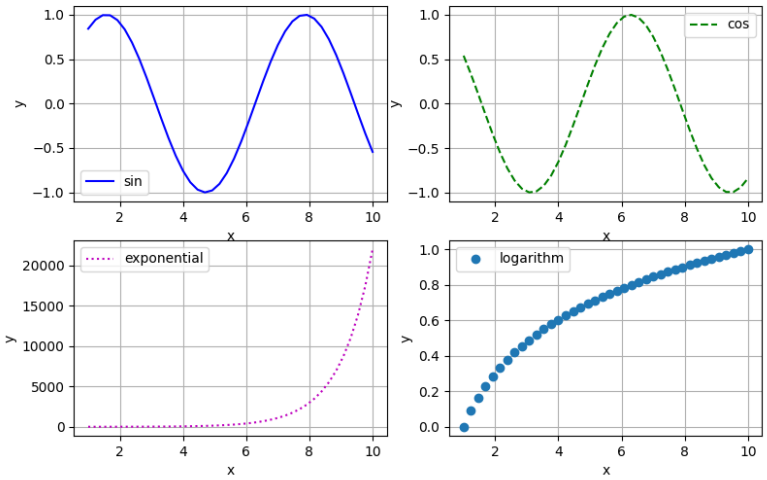

```
import matplotlib.pyplot as plt
import numpy as np
fig,axs =  plt.subplots(2,2, figsize=(12, 12))
x = np.linspace(1,10,num=40)

#Apply method to Axes element [0,0]
axs[0,0].plot(x,np.sin(x),'-b',label='sin')
axs[0,0].set_xlabel('x')
axs[0,0].set_ylabel('y')
axs[0,0].grid()
axs[0,0].legend()

#Apply method to Axes element [0.1]
axs[0,1].plot(x,np.cos(x),'--g', label='cos')
axs[0,1].set_xlabel('x')
axs[0,1].set_ylabel('y')
axs[0,1].grid()
axs[0,1].legend()

#Apply method to Axes element [1,0]
axs[1,0].plot(x,np.exp(x),':m',label='exponential')
axs[1,0].set_xlabel('x')
axs[1,0].set_ylabel('y')
axs[1,0].grid()
axs[1,0].legend()

#Apply method to Axes element [1,1]
axs[1,1].plot(x,np.log10(x),'o', label='logarithm')
axs[1,1].set_xlabel('x')
axs[1,1].set_ylabel('y')
axs[1,1].grid()
axs[1,1].legend()
plt.show()
```



Let's make the same subplots by using a for loop →

Use a loop to make subplots


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

fig,axs =  plt.subplots(2,2, figsize=(12, 12))
x = np.linspace(1,10,num=40)


#make lists that contain data we use within the loop
formats=['-b','g--', ':m','o'] # format strings for the plot
labels=['sin','cos','exponential','logarithm']
func=[np.sin,np.cos,np.exp,np.log10] # ufunc


for i,j in enumerate(axs.flatten()): # we flat the matrix of Axes, i is the
index and j is the Axes element
      j.plot(x,func[i](x),formats[i],label=labels[i])
      j.set_xlabel('x')
      j.set_ylabel('y')
      #j.set_xlim(1,10)
      #j.set_xticks(np.arange(1,10))
      j.grid()
      j.legend()

plt.show()
```



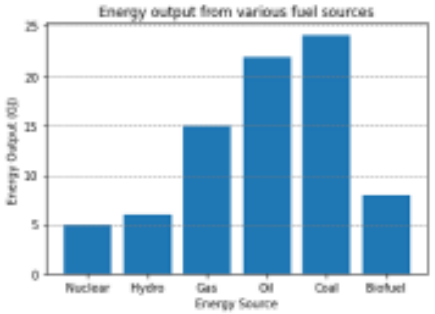
Create a custom plotting function


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

This function takes parameters: the Axes, etc, and returns the customized Axes

```
def myplot(ax0,xvalue,yvalue,formatstr, labelx, labely):
    ax0.plot(xvalue, yvalue,formatstr)
    ax0.set_xlabel(labelx)
    ax0.set_ylabel(labely)
    return ax0


x = np.linspace(1,10,num=40)
y=np.sin(x)


fig, ax = plt.subplots()
ax_out=myplot(ax,x,y,'--b', 'x','y') #call the function
plt.show()

fig, ax1 = plt.subplots()
y1=np.cos(x)
ax_out1=myplot(ax1,x,y1,'--r', 'x','y') # call the function
ax_out1.set_title("popolo") # you can apply methods to the returned
object
plt.show()
```

The bar graph

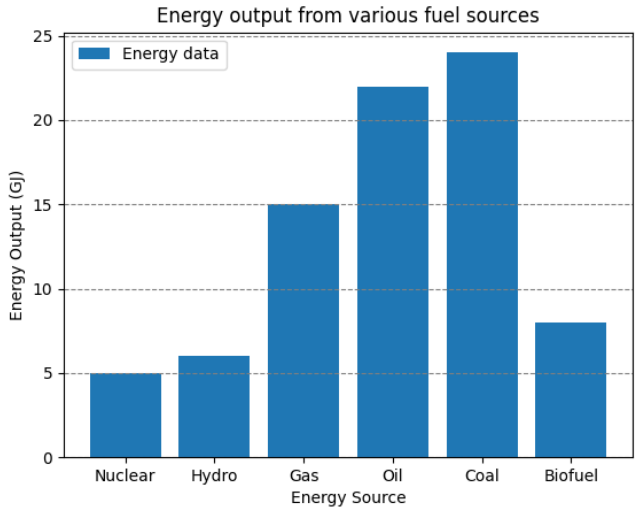
`ax.bar(x,energy)`	bar graph	compare data between different groups	

https://matplotlib.org/stable/gallery/lines_bars_and_markers/index.html

You can make a bar graph by using 1D arrays, lists or Series

Making a bar graph by using Series


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


x = ['Nuclear', 'Hydro', 'Gas', 'Oil',
     'Coal', 'Biofuel']
y = [5, 6, 15, 22, 24, 8]
```




```
#We can create a pandas DataFrame
energy_df = pd.DataFrame({'Source': x, 'Energy Production': y})
source = energy_df['Source'] # Series
energy = energy_df.loc[:,'Energy Production'] # Series

fig, ax = plt.subplots()
ax.bar(source, energy, label="Energy data")
ax.set_xlabel("Energy Source")
ax.set_ylabel("Energy Output (GJ)")

ax.set_title("Energy output from various fuel sources") #Add title

# we set the grid to be only horizontal
ax.grid(visible=True, which='major', axis='y', color='grey', linestyle='--')
ax.legend()
plt.show()
```

The horizontal bar graph

To make a horizontal bar graph use barh(y,x)
In barh(y,x) the first argument (y) is plotted on the y axis, and the second (x) on the x axis.