Matplotlib

Data Visualization is the process of presenting data in the form of graphs or charts. Data visualization can be done with various tools like Tableau, Power BI, Python. In this article, we will discuss how to visualize data with the help of the **Matplotlib** library of Python.

In [1]:

```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\pc
\appdata\local\programs\python\python37\lib\site-package
s (3.5.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users
\pc\appdata\local\programs\python\python37\lib\site-pack
ages (from matplotlib) (9.4.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\u
sers\pc\appdata\local\programs\python\python37\lib\site-
packages (from matplotlib) (1.4.4)
Requirement already satisfied: cycler>=0.10 in c:\users
\pc\appdata\local\programs\python\python37\lib\site-pack
ages (from matplotlib) (0.11.0)
Requirement already satisfied: numpy>=1.17 in c:\users\p
c\appdata\local\programs\python\python37\lib\site-packag
es (from matplotlib) (1.21.6)
Requirement already satisfied: fonttools>=4.22.0 in c:\u
sers\pc\appdata\local\programs\python\python37\lib\site-
packages (from matplotlib) (4.38.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\us
ers\pc\appdata\local\programs\python\python37\lib\site-p
ackages (from matplotlib) (3.0.9)
Requirement already satisfied: packaging>=20.0 in c:\use
rs\pc\appdata\local\programs\python\python37\lib\site-pa
ckages (from matplotlib) (22.0)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\pc\appdata\local\programs\python\python37\lib\s
ite-packages (from matplotlib) (2.8.2)
Requirement already satisfied: typing-extensions in c:\u
sers\pc\appdata\local\programs\python\python37\lib\site-
packages (from kiwisolver>=1.0.1->matplotlib) (4.4.0)
Requirement already satisfied: six>=1.5 in c:\users\pc\a
ppdata\local\programs\python\python37\lib\site-packages
(from python-dateutil>=2.7->matplotlib) (1.12.0)
Note: you may need to restart the kernel to use updated
packages.


[notice] A new release of pip is available: 23.0 -> 23.
0.1
[notice] To update, run: python.exe -m pip install --upg
rade pip
```

In [2]:

```python
# import nacessary datas
import matplotlib.pyplot as plt
from matplotlib.figure import Figure
import numpy as np
```

1. Type of plots and his uses

Line Plot: Line plots are used to represent the relationship between two variables on a continuous axis. It is used to visualize trends over time or across different categories.

Scatter Plot: Scatter plots are used to visualize the relationship between two continuous variables. They are useful for identifying patterns and trends in data.

Bar Plot: Bar plots are used to compare different categories of data. They are useful for displaying data that is not continuous, such as nominal or ordinal data.

Histogram: Histograms are used to visualize the distribution of a continuous variable. They display the frequency of values falling into different intervals or bins.

Box Plot: Box plots are used to visualize the distribution of a continuous variable. They display the quartiles of the data as well as any outliers.

Heatmap: Heatmaps are used to visualize the relationship between two variables on a 2D grid. They are useful for identifying patterns and trends in large datasets.

Pie Chart: Pie charts are used to represent the proportions of different categories in a dataset. They are useful for displaying data that is nominal or ordinal.

2. Matplotlib usefull functions list

plot(x, y): This function is used to create line plots.

scatter(x, y): This function is used to create scatter plots.

bar(x, height): This function is used to create bar plots.

barh(y, width): This function is used to create horizontal bar plots.

hist(x, bins): This function is used to create histograms.

boxplot(x): This function is used to create box plots.

pie(x): This function is used to create pie charts.

imshow(image): This function is used to display images.

subplot(nrows, ncols, index): This function is used to create subplots within a figure.

figure(): This function is used to create a new figure.

xlabel(label), ylabel(label): These functions are used to add labels to the x-axis and y-axis of a plot.

title(label): This function is used to add a title to a plot.

legend(): This function is used to add a legend to a plot.

xlim(left, right) and **ylim(bottom, top)**: These functions are used to set the limits of the x-axis and y-axis.

xticks(ticks, labels) and yticks(ticks, labels): These functions are used to set the tick marks on the x-axis and y-axis.

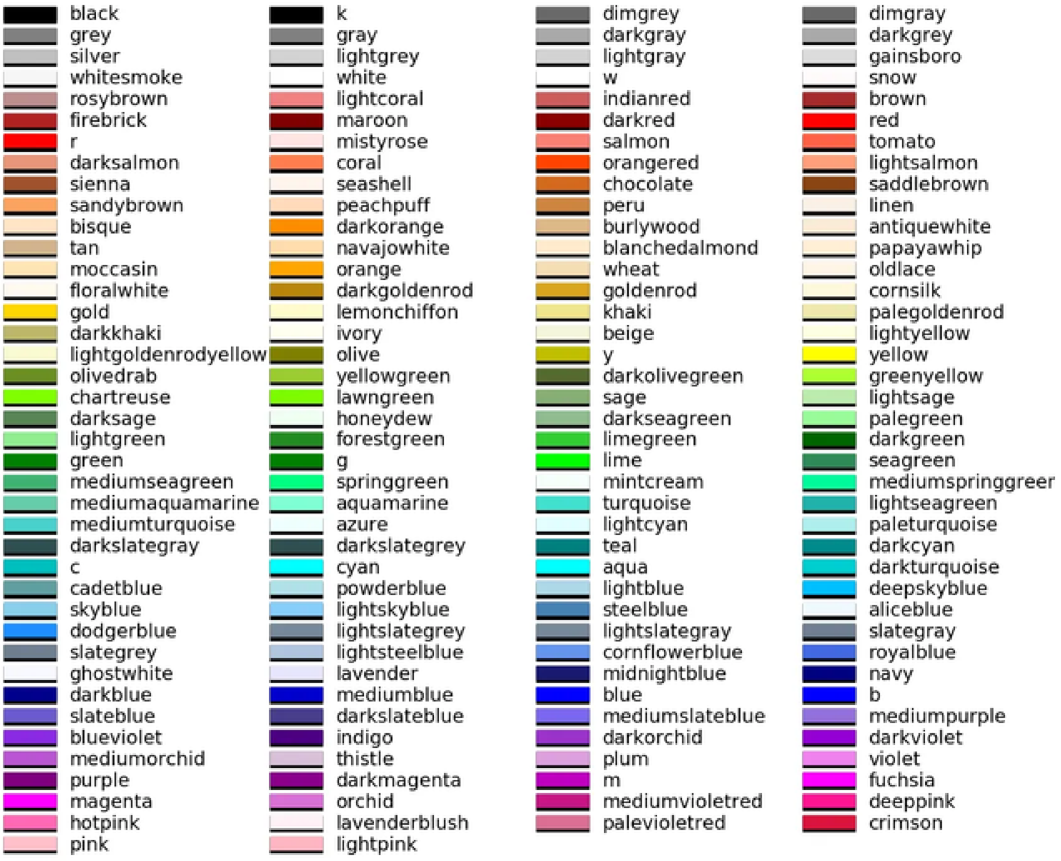
savefig(filename): This function is used to save the current figure to a file.

clf(): This function is used to clear the current figure.

grid(): This function is used to add grid lines to a plot.

annotate(text, xy, xytext): This function is used to add annotations to a plot.

3. Matplotlib all colors



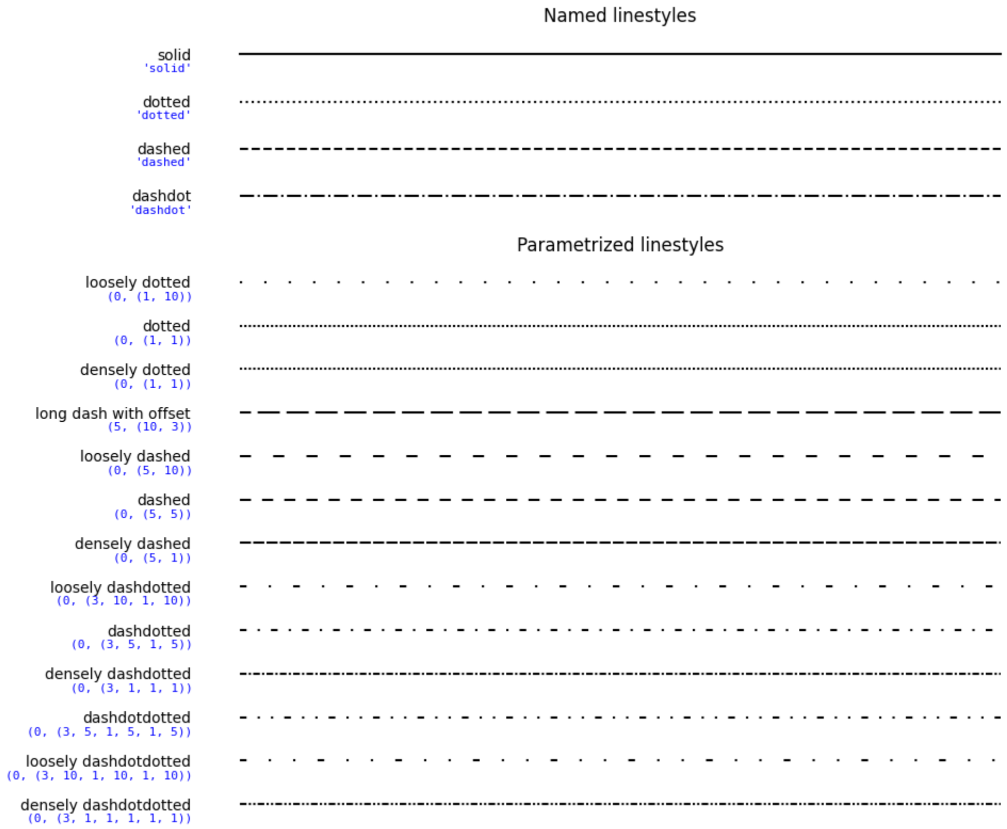
black
grey
silver
whitesmoke
rosybrown
firebrick
r
darksalmon
sienna
sandybrown
bisque
tan
moccasin
floralwhite
gold
darkkhaki
lightgoldenrodyellow
olivedrab
chartreuse
darksage
lightgreen
green
mediumseagreen
mediumaquamarine
mediumturquoise
darkslategray
c
cadetblue
skyblue
dodgerblue
slategrey
ghostwhite
darkblue
slateblue
blueviolet
mediumorchid
purple
magenta
hotpink
pink

k
gray
lightgrey
white
lightcoral
maroon
mistyrose
coral
seashell
peachpuff
darkorange
navajowhite
orange
darkgoldenrod
lemonchiffon
ivory
olive
yellowgreen
lawngreen
honeydew
forestgreen
g
springgreen
aquamarine
azure
darkslategrey
cyan
powderblue
lightskyblue
lightslategrey
lightsteelblue
lavender
mediumblue
darkslateblue
indigo
thistle
darkmagenta
orchid
lavenderblush
lightpink

dimgrey
darkgray
lightgray
w
indianred
darkred
salmon
orangered
chocolate
peru
burlywood
blanchedalmond
wheat
goldenrod
khaki
beige
y
darkolivegreen
sage
darkseagreen
limegreen
lime
mintcream
turquoise
lightcyan
teal
aqua
lightblue
steelblue
lightslategray
cornflowerblue
midnightblue
blue
mediumslateblue
darkorchid
plum
m
mediumvioletred
palevioletred

dimgray
darkgrey
gainsboro
snow
brown
red
tomato
lightsalmon
saddlebrown
linen
antiquewhite
papayawhip
oldlace
cornsilk
palegoldenrod
lightyellow
yellow
greenyellow
lightsage
palegreen
darkgreen
seagreen
mediumspringgreer
lightseagreen
paleturquoise
darkcyan
darkturquoise
deepskyblue
aliceblue
slategray
royalblue
navy
b
mediumpurple
darkviolet
violet
fuchsia
deeppink
crimson

4. Different Linestyle available



In [ ]:

```
# dir(plt)
```

In [ ]:

```
# x = np.random.rand(1,50)   #rand() parameter as a shape
# y = np.random.rand(1,50)
```

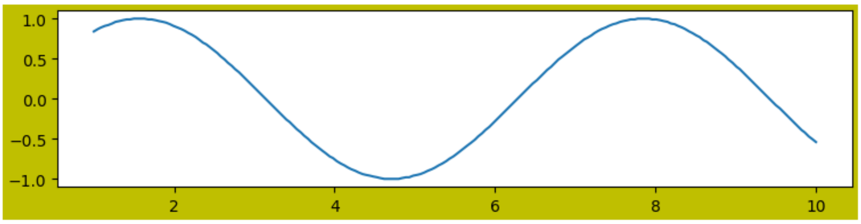
In [3]:

```
x = np.linspace(1,10 , 200).round(2)
y = np.sin(x).round(2)
```

5. Graph figure control

In [4]:

```
fig = plt.figure(figsize =(9, 2), facecolor='y', edgecolor='w', li
plt.plot(x,y)
plt.show()
```

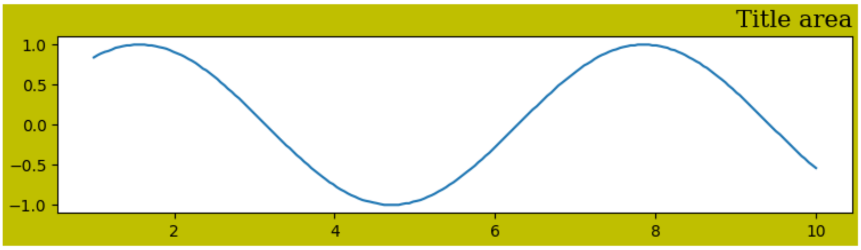


6. Graph title

fontdict : font Dictionary
font = {'family': 'serif','color': 'darkred','weight': 'bold','size': 16,}

In [5]:

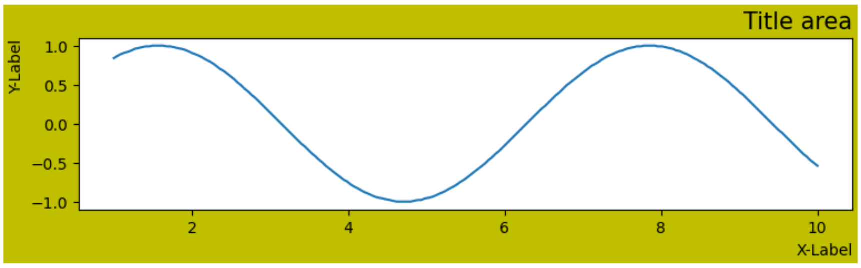
```
fig = plt.figure(figsize =(9, 2), facecolor='y', edgecolor='w', li
plt.title("Title area", fontsize=15, color="black", loc = "right",
plt.plot(x,y)
plt.show()
```



7. Graph label x and y axis

In [6]:

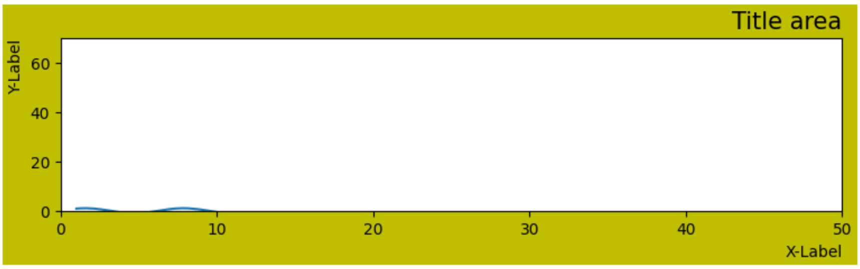
```
fig = plt.figure(figsize =(9, 2), facecolor='y', edgecolor='w', li
plt.title("Title area", fontsize=15, color="black", loc = "right")
#------------X and Y label
plt.xlabel("X-Label", fontdict=None, labelpad=None, loc='right',)
plt.ylabel("Y-Label", fontdict=None, labelpad=None, loc='top',)
plt.plot(x,y)
plt.show()
```



8. Setting Limits and Tick labels

In [13]:

```
fig = plt.figure(figsize =(9, 2), facecolor='y', edgecolor='w', li
plt.xlabel("X-Label", fontdict=None, labelpad=None, loc='right',)
plt.ylabel("Y-Label", fontdict=None, labelpad=None, loc='top',)
plt.title("Title area", fontsize=15, color="black", loc = "right")
#------------x and y limits-----------------
plt.xlim(0,50)
plt.ylim(0,70)
#------------x and y tick label-----------------
# plt.xticks(x, labels=["one", "two", "three", "four"],rotation=45
# plt.yticks(x, labels=["one", "two", "three", "four"])
plt.plot(x,y)
plt.show()
```



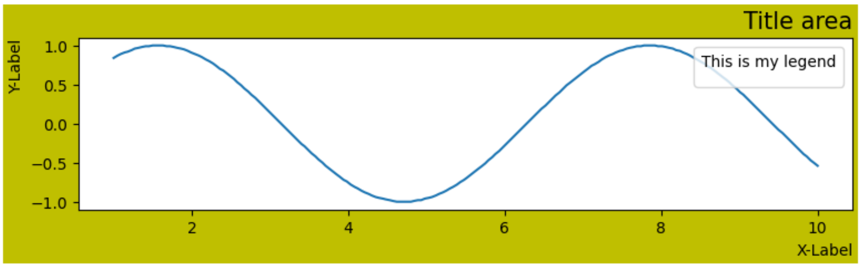
9. Adding Legends

frameon=True / False : It is used to show or hide border from legend_title

In [14]:

```
fig = plt.figure(figsize =(9, 2), facecolor='y', edgecolor='w', li
plt.xlabel("X-Label", fontdict=None, labelpad=None, loc='right',)
plt.ylabel("Y-Label", fontdict=None, labelpad=None, loc='top',)
plt.title("Title area", fontsize=15, color="black", loc = "right")
#-------------adding legends-------------------
plt.legend(title = "This is my legend",fontsize=12, frameon=True,l
plt.plot(x,y)
plt.show()
```

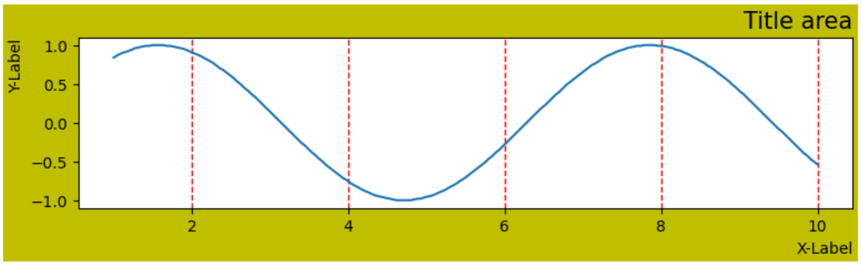
```
No artists with labels found to put in legend.  Note tha
t artists whose label start with an underscore are ignor
ed when legend() is called with no argument.
```



10. Show grid in graph

In [15]:

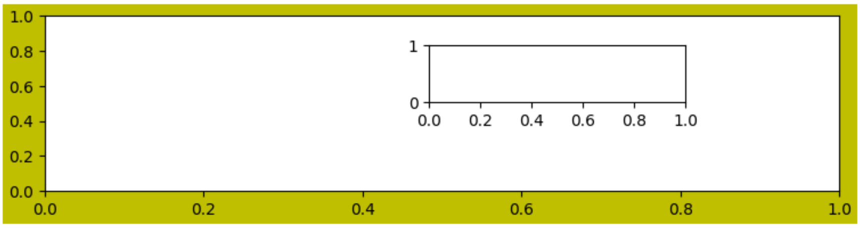
```
fig = plt.figure(figsize =(9, 2), facecolor='y', edgecolor='w', li
plt.xlabel("X-Label", fontdict=None, labelpad=None, loc='right',)
plt.ylabel("Y-Label", fontdict=None, labelpad=None, loc='top',)
plt.title("Title area", fontsize=15, color="black", loc = "right")
#----------grid and his styling
plt.grid(axis='x', color = "red", linewidth = 1, linestyle='--')
plt.plot(x,y)
plt.show()
```



11. Create a plot inside another plot using axes()

In [16]:

```
fig = plt.figure(figsize =(9, 2), facecolor='y', edgecolor='w', li
ax1 = plt.axes() # standard axes
ax2 = plt.axes([0.5, 0.5, 0.25, 0.25])
```

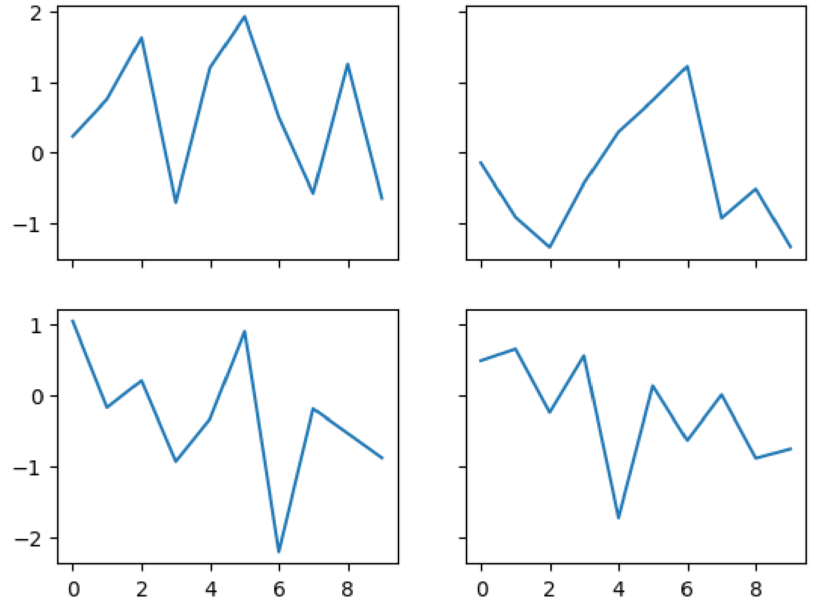


12. Use plt.subplots to create figure and multiple axes (most useful)

In [17]:

```
# plt.subplots example
x  = np.arange(0,10,1)
y1 = np.random.randn(10)
y2 = np.random.randn(10)
y3 = np.random.randn(10)
y4 = np.random.randn(10)

# Create subplots
fig, ax = plt.subplots(2, 2, sharex='col', sharey='row')
ax[0][0].plot(x,y1)
ax[0][1].plot(x,y2)
ax[1][0].plot(x,y3)
ax[1][1].plot(x,y4)
```

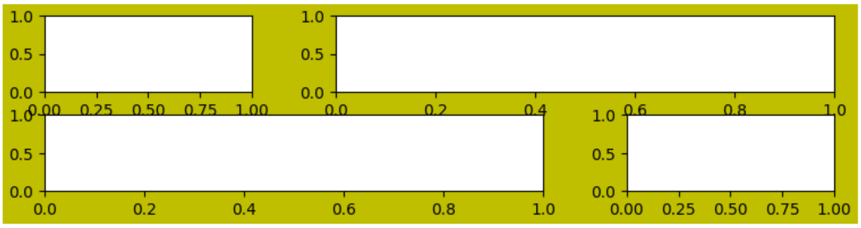
```
[<matplotlib.lines.Line2D at 0x1f99a2502788>]
```



13. Using GridSpec() function to create customized axes

In [18]:

```
fig = plt.figure(figsize =(9, 2), facecolor='y', edgecolor='w', li
grid = plt.GridSpec(2, 3, wspace=0.4, hspace=0.3)
plt.subplot(grid[0, 0])
plt.subplot(grid[0, 1:])
plt.subplot(grid[1, :2])
plt.subplot(grid[1, 2]);
```

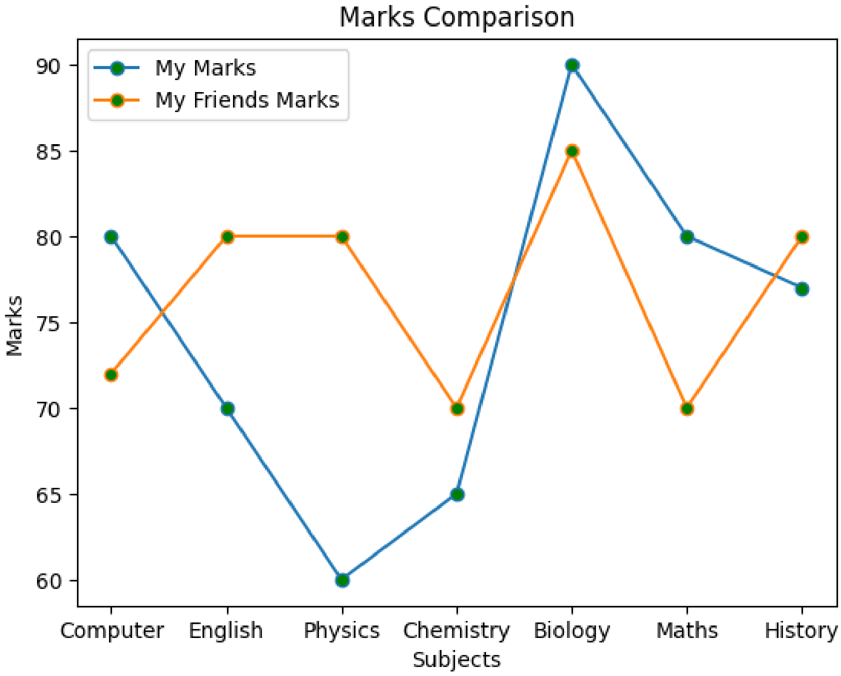


More multiple plots :
[https://matplotlib.org/3.5.0/api/_as_gen/matplotlib.pyplot.subplot.html (https://matplotlib.org/3.5.0/api/_as_gen/matplotlib.pyplot.subplot.html)](https://matplotlib.org/3.5.0/api/_as_gen/matplotlib.pyplot.subplot.html)

14. Multiline graph

In [19]:

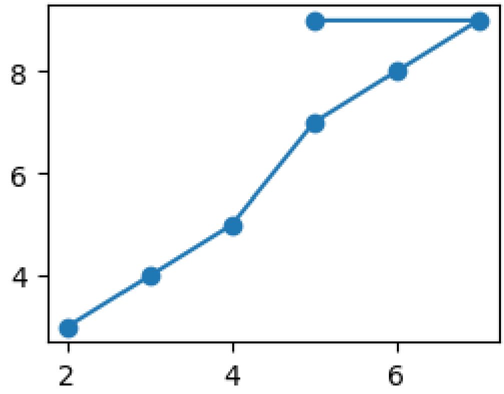
```
import matplotlib.pyplot as plt
subjects = ['Computer', 'English', 'Physics', 'Chemistry', 'Biolog
my_marks = [80, 70, 60, 65,90,80,77]
my_friends_marks = [72,80,80, 70, 85, 70, 80]
plt.plot(subjects, my_marks, label='My Marks', marker='o', markerf
plt.plot(subjects, my_friends_marks, label='My Friends Marks', mar
plt.xlabel('Subjects')
plt.ylabel('Marks')
plt.title('Marks Comparison')
plt.legend()
plt.show()
```



15. Multiple Plots

In [20]:

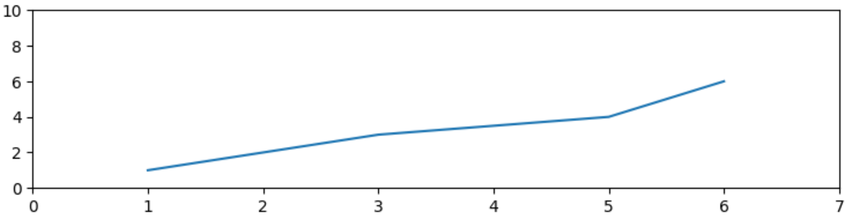
```
ax = plt.subplot(2,2,1)
x = np.array([2,3,4,5,6,7,5])
y = np.array([3,4,5,7,8,9,9])
ax.plot(x,y)
x = np.array([2,3,4,5,6,7,5])
y = np.array([3,4,5,7,8,9,9])
ax.scatter(x,y, linewidths=1)
plt.show()
```



16. Line Plot

In [21]:

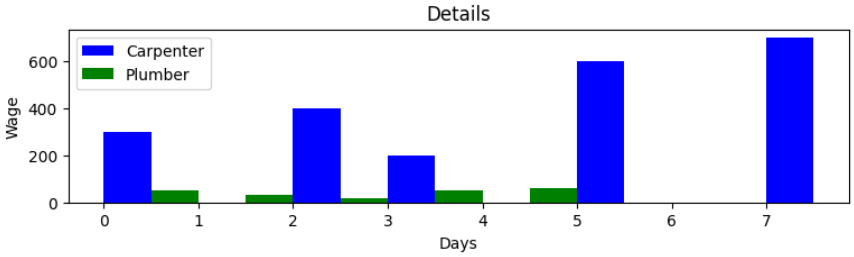
```
fig = plt.figure(figsize =(9, 2), edgecolor='w', linewidth=7)
plt.plot([1,2,3,5,6], [1, 2, 3, 4, 6])
plt.axis([0, 7, 0, 10])
plt.show()
```



17. Bar Plot

In [22]:

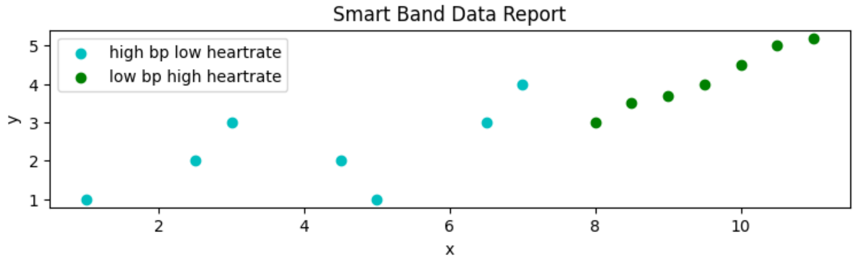
```
fig = plt.figure(figsize =(9, 2), edgecolor='w', linewidth=7)
plt.bar([0.25,2.25,3.25,5.25,7.25],[300,400,200,600,700],
label="Carpenter",color='b',width=0.5)
plt.bar([0.75,1.75,2.75,3.75,4.75],[50,30,20,50,60],
label="Plumber", color='g',width=.5)
plt.legend()
plt.xlabel('Days')
plt.ylabel('Wage')
plt.title('Details')
plt.show()
```



18. Scatter Plot

In [23]:

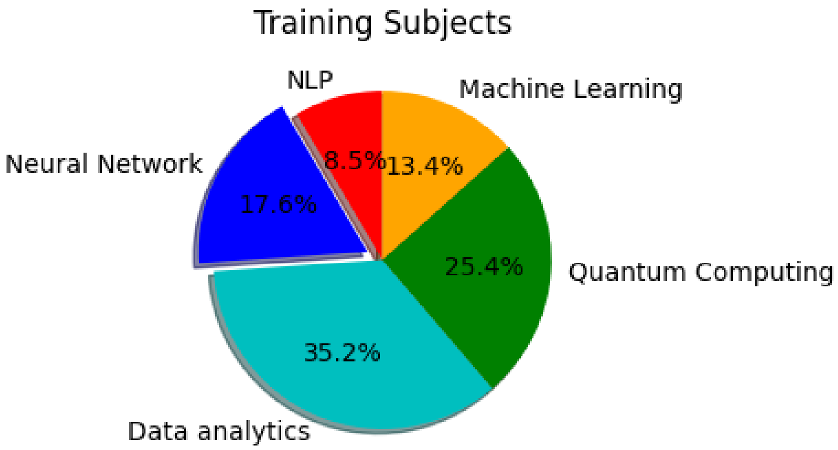
```
fig = plt.figure(figsize =(9, 2), edgecolor='w', linewidth=7)
x1 = [1, 2.5,3,4.5,5,6.5,7]
y1 = [1,2, 3, 2, 1, 3, 4]
x2=[8, 8.5, 9, 9.5, 10, 10.5, 11]
y2=[3,3.5, 3.7, 4,4.5, 5, 5.2]
plt.scatter(x1, y1, label = 'high bp low heartrate', color='c')
plt.scatter(x2,y2,label='low bp high heartrate',color='g')
plt.title('Smart Band Data Report')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```



19. Pie Plot

In [24]:

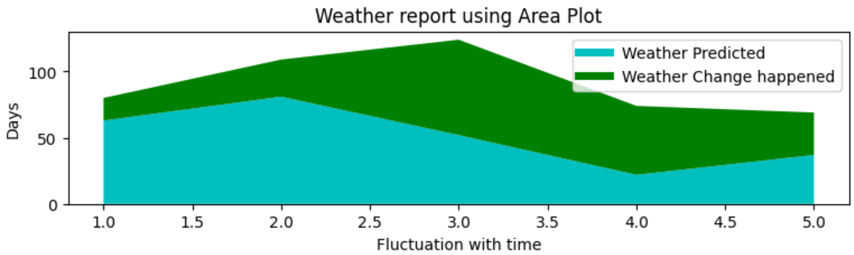
```
fig = plt.figure(figsize =(11, 3), edgecolor='w', linewidth=7)
slice = [12, 25, 50, 36, 19]
activities = ['NLP','Neural Network', 'Data analytics', 'Quantum C
cols = ['r','b','c','g', 'orange']
plt.pie(slice,
labels =activities,
colors = cols,
startangle = 90,
shadow = True,
explode =(0,0.1,0,0,0),
autopct ='%1.1f%%')
plt.title('Training Subjects')
plt.show()
```



20. Area Plot

In [25]:

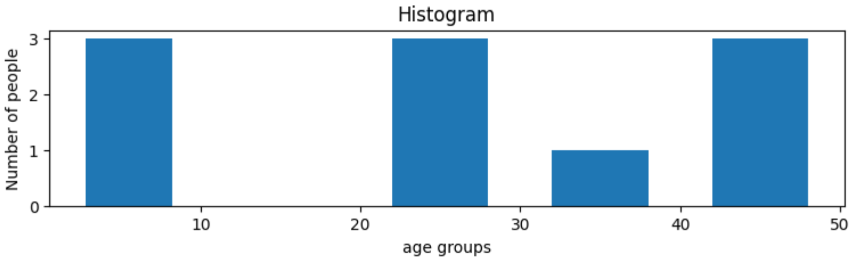
```
fig = plt.figure(figsize =(9, 2), edgecolor='w', linewidth=7)
days = [1,2,3,4,5]
age =[63, 81, 52, 22, 37]
weight =[17, 28, 72, 52, 32]
plt.plot([],[], color='c', label = 'Weather Predicted', linewidth=
plt.plot([],[],color = 'g', label='Weather Change happened', linew
plt.stackplot(days, age, weight, colors = ['c', 'g'])
plt.xlabel('Fluctuation with time')
plt.ylabel('Days')
plt.title('Weather report using Area Plot')
plt.legend()
plt.show()
```



21. Histogram Plot

In [26]:

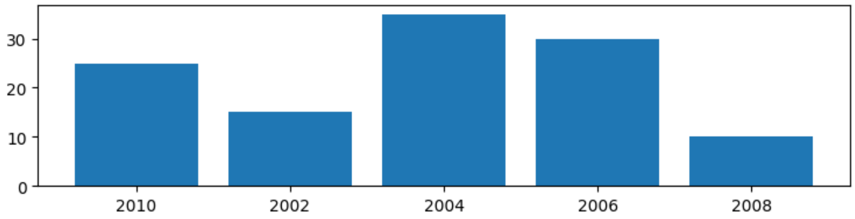
```
fig = plt.figure(figsize =(9, 2), edgecolor='w', linewidth=7)
pop = [22,55,62,45,21,22,34,42,42,4,2,8]
bins = [1,10,20,30,40,50]
plt.hist(pop, bins, rwidth=0.6)
plt.xlabel('age groups')
plt.ylabel('Number of people')
plt.title('Histogram')
plt.show()
```



If you are interested to know more about plot visit now : [https://matplotlib.org/stable/plot_types/index.html (https://matplotlib.org/stable/plot_types/index.html)](https://matplotlib.org/stable/plot_types/index.html)

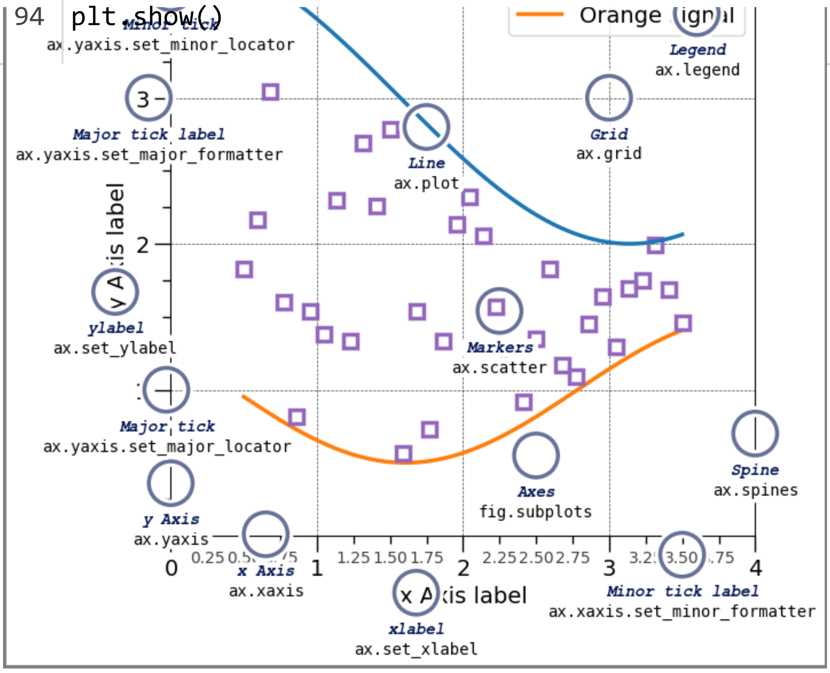
In [27]:

```
fig = plt.figure(figsize =(9, 2), edgecolor='w', linewidth=7)
# Creating data
year = ['2010', '2002', '2004', '2006', '2008']
production = [25, 15, 35, 30, 10]

# Plotting barchart
plt.bar(year, production)

# Saving the figure.
plt.savefig("output.jpg")

# Saving figure by changing parameter values
plt.savefig("output1", facecolor='y', bbox_inches="tight",
            pad_inches=0.3, transparent=True)
```



In [28]:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
from matplotlib.patheffects import withStroke
from matplotlib.ticker import AutoMinorLocator, MultipleLocator

royal_blue = [0, 20/256, 82/256]


# make the figure

np.random.seed(19680801)

X = np.linspace(0.5, 3.5, 100)
Y1 = 3+np.cos(X)
Y2 = 1+np.cos(1+X/0.75)/2
Y3 = np.random.uniform(Y1, Y2, len(X))

fig = plt.figure(figsize=(7.5, 7.5))
ax = fig.add_axes([0.2, 0.17, 0.68, 0.7], aspect=1)

ax.xaxis.set_major_locator(MultipleLocator(1.000))
ax.xaxis.set_minor_locator(AutoMinorLocator(4))
ax.yaxis.set_major_locator(MultipleLocator(1.000))
ax.yaxis.set_minor_locator(AutoMinorLocator(4))
ax.xaxis.set_minor_formatter("{x:.2f}")

ax.set_xlim(0, 4)
ax.set_ylim(0, 4)

ax.tick_params(which='major', width=1.0, length=10, labelsize=14)
ax.tick_params(which='minor', width=1.0, length=5, labelsize=10,
               labelcolor='0.25')

ax.grid(linestyle="--", linewidth=0.5, color='.25', zorder=-10)

ax.plot(X, Y1, c='C0', lw=2.5, label="Blue signal", zorder=10)
ax.plot(X, Y2, c='C1', lw=2.5, label="Orange signal")
ax.plot(X[::3], Y3[::3], linewidth=0, markersize=9,
        marker='s', markerfacecolor='none', markeredgecolor='C4',
        markeredgewidth=2.5)

ax.set_title("Anatomy of a figure", fontsize=20, verticalalignment
ax.set_xlabel("x Axis label", fontsize=14)
ax.set_ylabel("y Axis label", fontsize=14)
ax.legend(loc="upper right", fontsize=14)


# Annotate the figure

def annotate(x, y, text, code):
    # Circle marker
    c = Circle((x, y), radius=0.15, clip_on=False, zorder=10, line
               edgecolor=royal_blue + [0.6], facecolor='none',
               path_effects=[withStroke(linewidth=7, foreground='w
    ax.add_artist(c)

    # use path_effects as a background for the texts
    # draw the path_effects and the colored text separately so tha
    # path_effects cannot clip other texts
    for path_effects in [[withStroke(linewidth=7, foreground='whit
        color = 'white' if path_effects else royal_blue
        ax.text(x, y-0.2, text, zorder=100,
                ha='center', va='top', weight='bold', color=color,
                style='italic', fontfamily='Courier New',
                path_effects=path_effects)

        color = 'white' if path_effects else 'black'
        ax.text(x, y-0.33, code, zorder=100,
                ha='center', va='top', weight='normal', color=colc
                fontfamily='monospace', fontsize='medium',
                path_effects=path_effects)


annotate(3.5, -0.13, "Minor tick label", "ax.xaxis.set_minor_forma
annotate(-0.03, 1.0, "Major tick", "ax.yaxis.set_major_locator")
annotate(0.00, 3.75, "Minor tick", "ax.yaxis.set_minor_locator")
annotate(-0.15, 3.00, "Major tick label", "ax.yaxis.set_major_form
annotate(1.68, -0.39, "xlabel", "ax.set_xlabel")
annotate(-0.38, 1.67, "ylabel", "ax.set_ylabel")
annotate(1.52, 4.15, "Title", "ax.set_title")
annotate(1.75, 2.80, "Line", "ax.plot")
annotate(2.25, 1.54, "Markers", "ax.scatter")
annotate(3.00, 3.00, "Grid", "ax.grid")
annotate(3.60, 3.58, "Legend", "ax.legend")
annotate(2.5, 0.55, "Axes", "fig.subplots")
annotate(4, 4.5, "Figure", "plt.figure")
annotate(0.65, 0.01, "x Axis", "ax.xaxis")
annotate(0, 0.36, "y Axis", "ax.yaxis")
annotate(4.0, 0.7, "Spine", "ax.spines")

# frame around figure
fig.patch.set(linewidth=4, edgecolor='0.5')
plt.show()
```



In [ ]:

```

```

In [ ]:

```

```

In [29]:

```
from IPython.display import display, HTML
display(HTML("<style>.container { width:80% !important; }</style>"
```