Matplotlib Module

Introduction

Matplotlib is a library package for data visualization, especially scientific information.
It can be used to create various graphs/plots, e.g. line graph, bar graph, histogram, etc.

Simple Usage Guide: Before creating any plot, the library must be included.

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

Figures: Figure is an area on a window that the Matplotlib can plot the data inside the area.

Figure can contain multiple subplots.

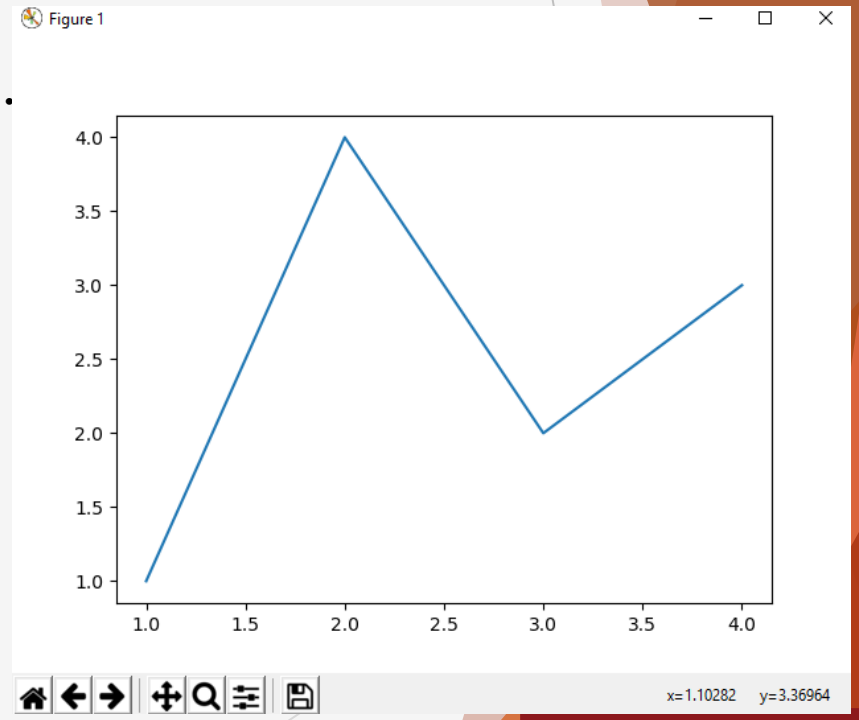
Axes: Axes is actual area where data can be mapped into coordinate: for example the x-y coordinates.

Figures and Axes

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()  # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])  # Plot some data on the axes.
plt.show()
```

This example, plt.subplots() is used to create a figure and the axes.

Then ax.plot([1,2,3,4], [1,4,2,3]) is used to put x and y coordinate pair into the axes.
plt.show() displays result on a window.



Simple/Quick Plot

Most of the time, user may want to just create a single plot. In this case, simply use **plot** method.

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [1, 4, 2, 3])  # Plot some data on the axes.
plt.show()
```

If x axis data is simply a sequence of 1,2,3,4,… , it can be omitted.

```
import matplotlib.pyplot as plt

plt.plot( [1, 4, 2, 3])  # Plot some data on the axes.
plt.show()
```

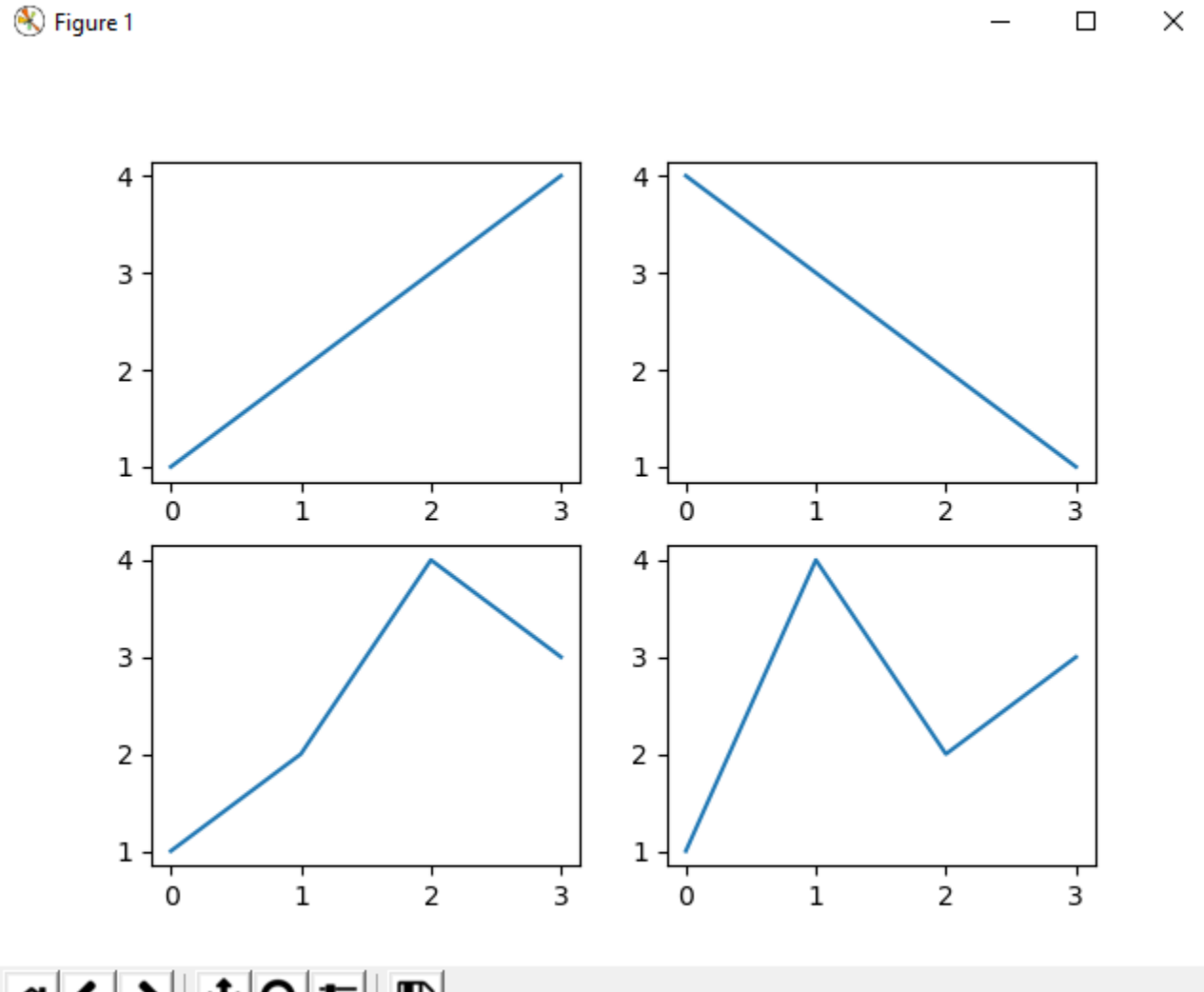
Multiple subplots

The following example create 4 subplots in a figure.

```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(2, 2)  # a figure with a 2x2 grid of Axes
```

axs contains 2 by 2 subplots and are accessible as in the following example

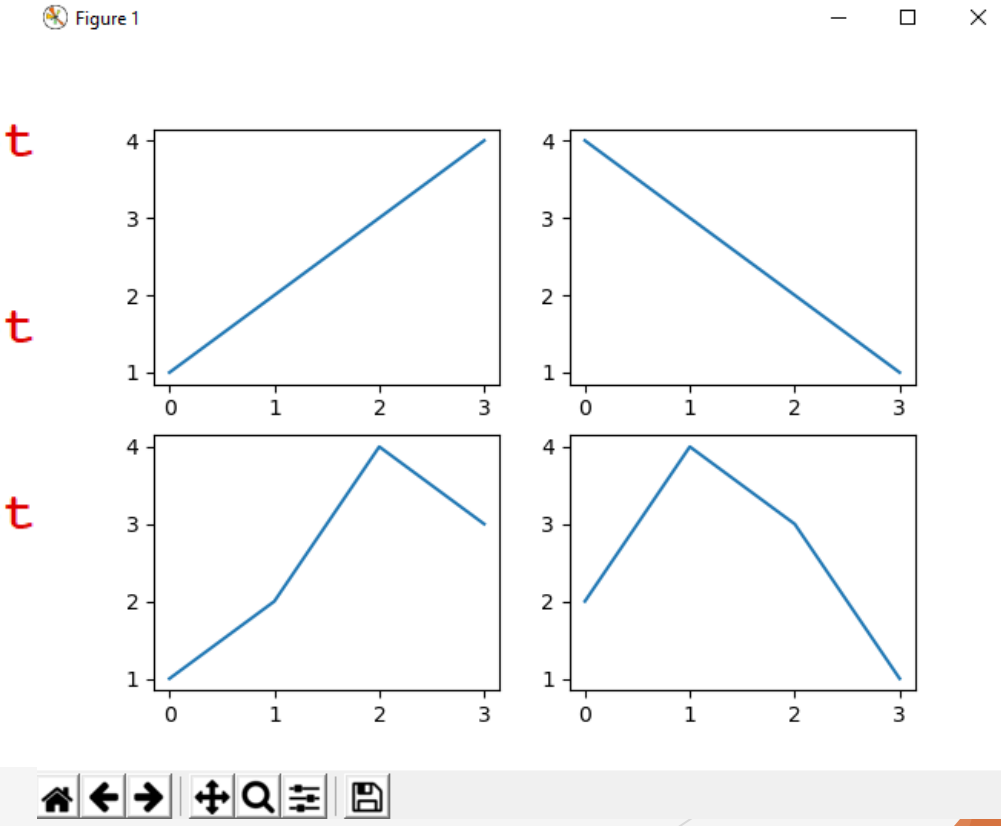
```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(2, 2)  #
axs[0][0].plot([1,2,3,4])
axs[0][1].plot([4,3,2,1])
axs[1][0].plot([1,2,4,3])
axs[1][1].plot([1,4,2,3])
plt.show()
```



Multiple subplots (another approach)

The following example create 4 subplots in a figure.

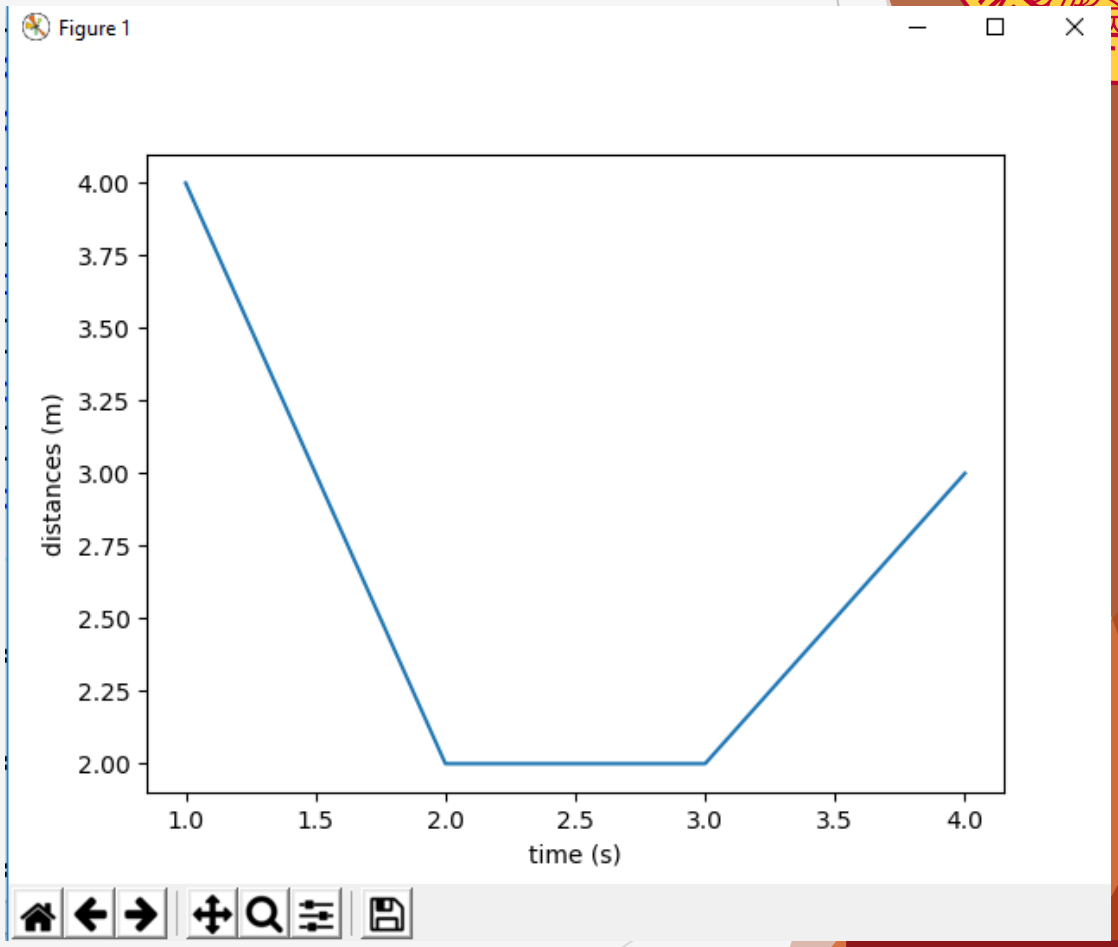
```
import matplotlib.pyplot as plt

plt.figure()  # create a new figure
plt.subplot(221) #221 means 2 rows 2 columns and 1st subplot
plt.plot([1,2,3,4])

plt.subplot(222) # 2nd subplot
plt.plot([4,3,2,1])

plt.subplot(223) # 3rd subplot
plt.plot([1,2,4,3])

plt.subplot(224) # 4th subplot
plt.plot([2,4,3,1])

plt.show()
```



Adding labels

Use **xlabel()** and **ylabel()** to add y-axis's label to the plot.

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4],[4, 2, 2, 3])
plt.ylabel('distances (m)')
plt.xlabel('time (s)')
plt.show()
```



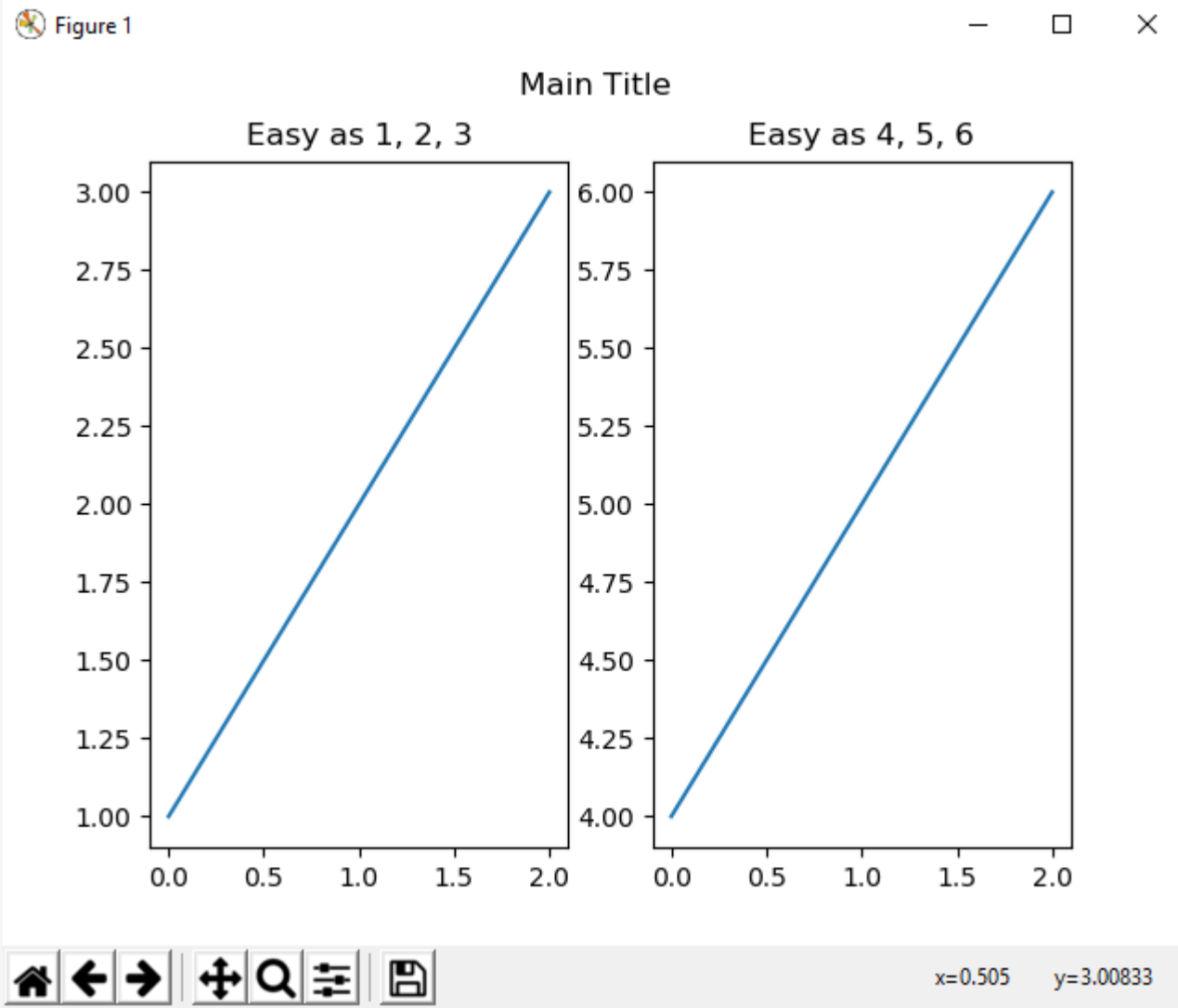
Title

Use **title()** to add title text to the current subplot.
Use **suptitle()** to create a main title for the figure.

```
import matplotlib.pyplot as plt
plt.figure()                    # the first figure
plt.subplot(121)                 # the first subplot in the first figure
plt.plot([1, 2, 3])
plt.title('Easy as 1, 2, 3') # subplot 211 title

plt.subplot(122)                 # the second subplot in the first figure
plt.plot([4, 5, 6])
plt.title('Easy as 4, 5, 6') # subplot 212 title

plt.suptitle('Main Title') # use suptitle() for main figure's title
plt.show()
```

Titles (cont.)

The result from the previous code.

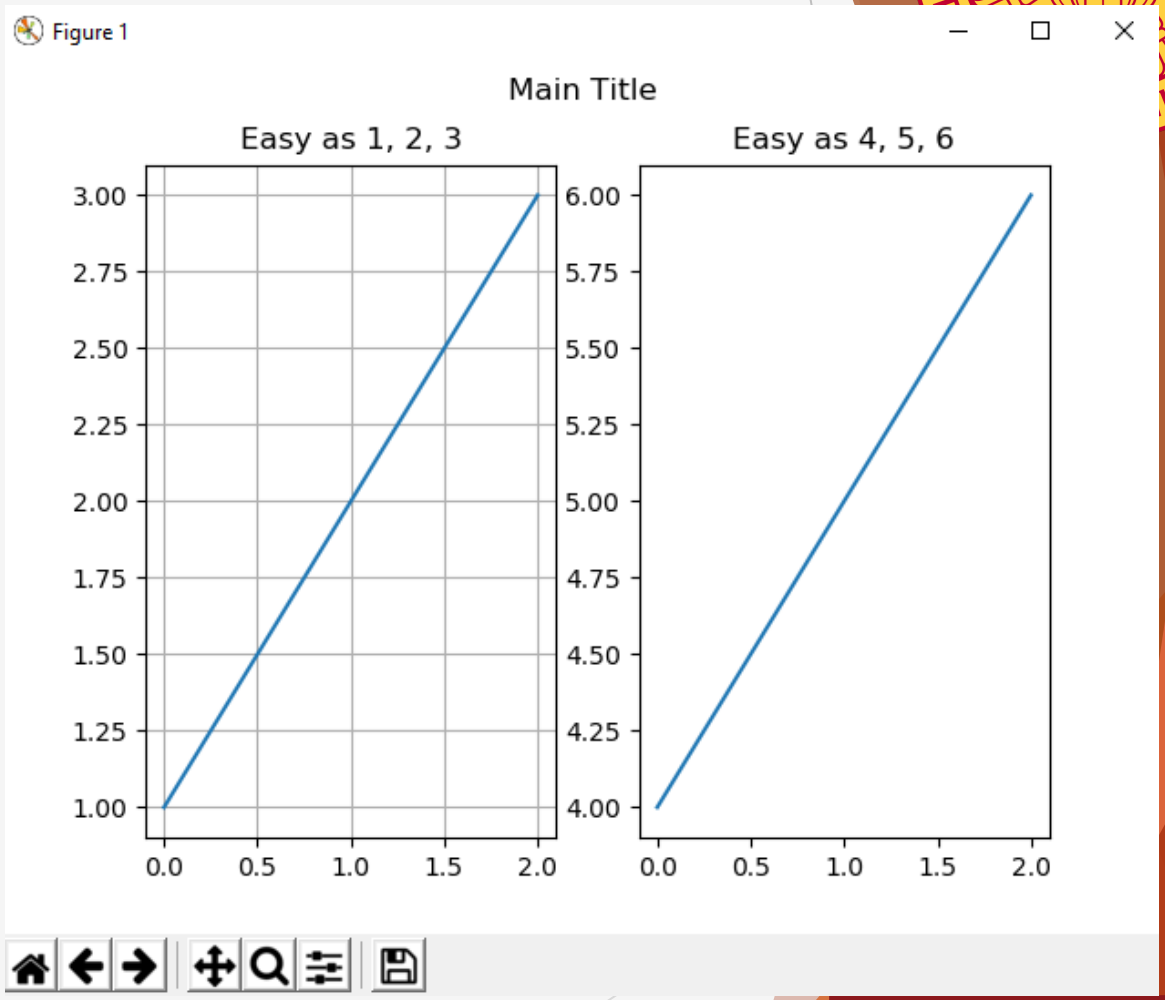


Grid

Turn grid ON or OFF

Use **grid(True)** to show grid on the plot. Use **grid(False)** to turn it off.

```
import matplotlib.pyplot as plt
plt.figure()
plt.subplot(121)
plt.plot([1, 2, 3])
plt.title('Easy as 1, 2, 3')
plt.grid(True) #grid on

plt.subplot(122)
plt.plot([4, 5, 6])
plt.title('Easy as 4, 5, 6')
plt.grid(False) #grid off
plt.suptitle('Main Title')
plt.show()
```

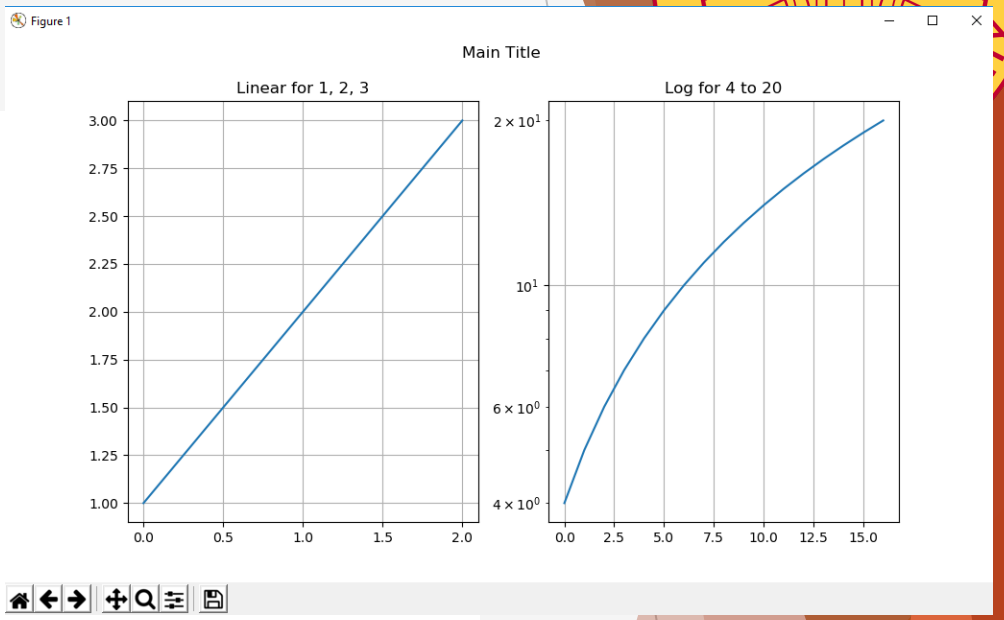


Scale

Graph can be plotted in different scales. Normally linear scale is the default option.
It is also possible to plot data in logarithmatic (log) scale both x-axis and y-axis.

xscale('linear'), xscale('log')
Yscale('linear'), yscale('log')

```
import matplotlib.pyplot as plt
plt.figure()
plt.subplot(121)
plt.plot([1, 2, 3])
plt.yscale('linear') #linear scale
plt.title('Linear for 1, 2, 3')
plt.grid(True) #grid on

plt.subplot(122)
plt.plot([4, 5, 6,7,8,9,10,11,12,13,14,15,16,17,18,19,20])
plt.yscale('log') #log scale
plt.title('Log for 4 to 20')
plt.grid(True) #grid on
plt.suptitle('Main Title')
plt.show()
```

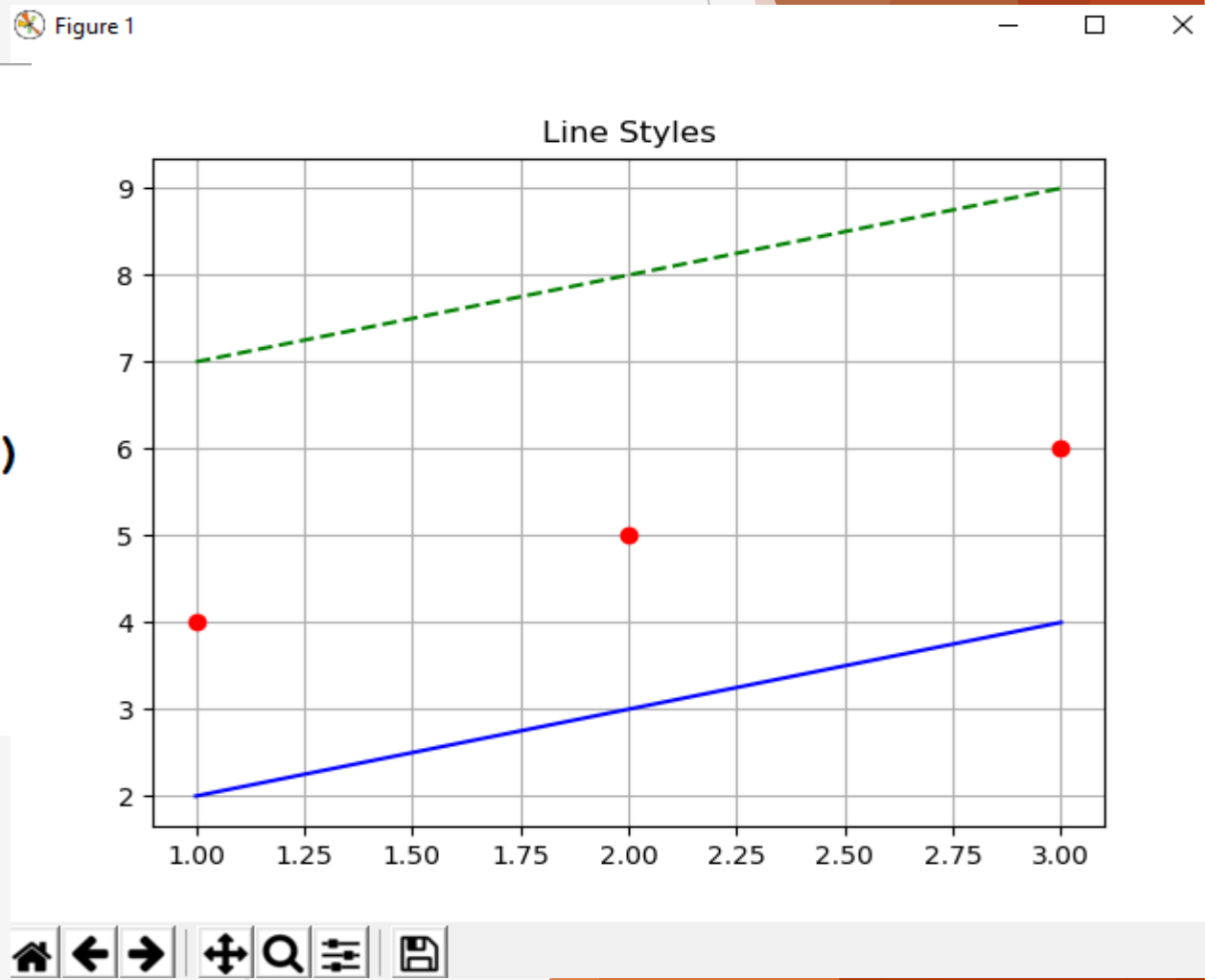


TSE THAMMASAT SCHOOL OF ENGINEERING

Formatting the Style

Plot's properties such as line style, color, marker can be specified.
Color and line style can be specified using a string
The default style is 'b-' which represents blue (b) color and solid (-) line.
Style can be used to differentiate multiple data series in the same plot.

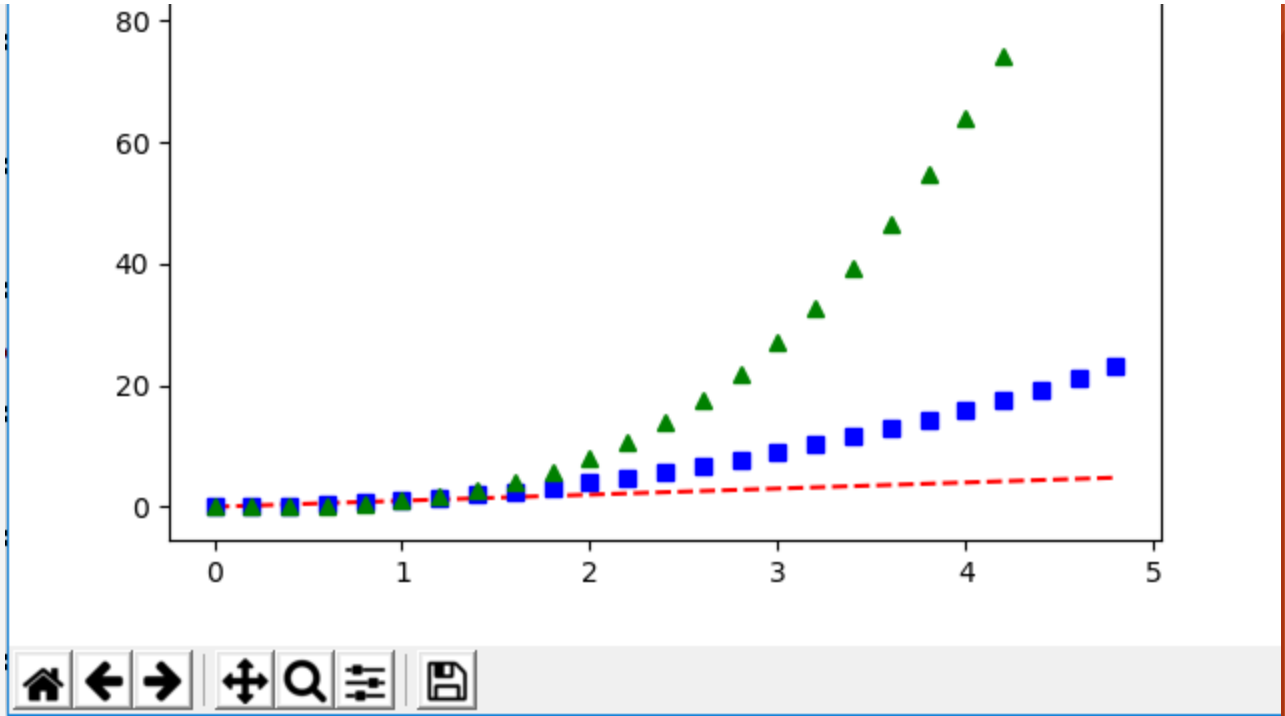
```
import matplotlib.pyplot as plt
plt.figure()
x = [1,2,3]
y1 = [2,3,4]
y2 = [4,5,6]
y3 = [7,8,9]
plt.plot(x,y1,'b-',x,y2,'ro',x,y3,'g--')
plt.yscale('linear') #linear scale
plt.title('Line Styles')
plt.grid(True) #grid on
plt.show()
```



Using numpy array

For large amount of data, numpy's array is used to store the data.

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

t = np.arange(0.0, 5.0, 0.2) #generate array from 0 to 5 with interval 0.2
#each element in the array has value 0.2 apart from each other.
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```

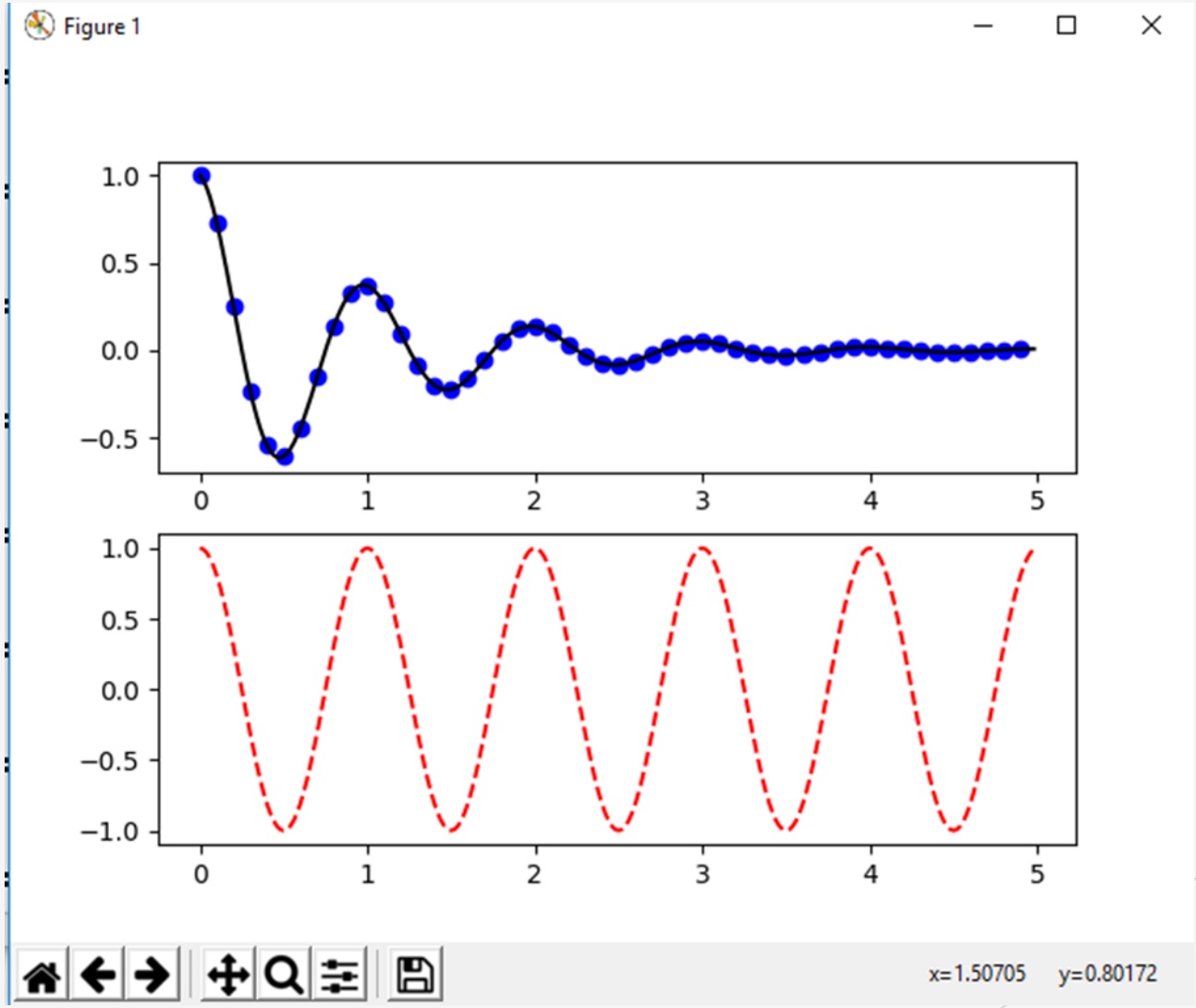


Using function to generate y values

It is also possible to use equations to generate y values through the use of function.

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

#this is the function that is used to generate y values.
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1) #x data
t2 = np.arange(0.0, 5.0, 0.02) #another x data

plt.figure()
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k') #passing f(t1) as y values

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```

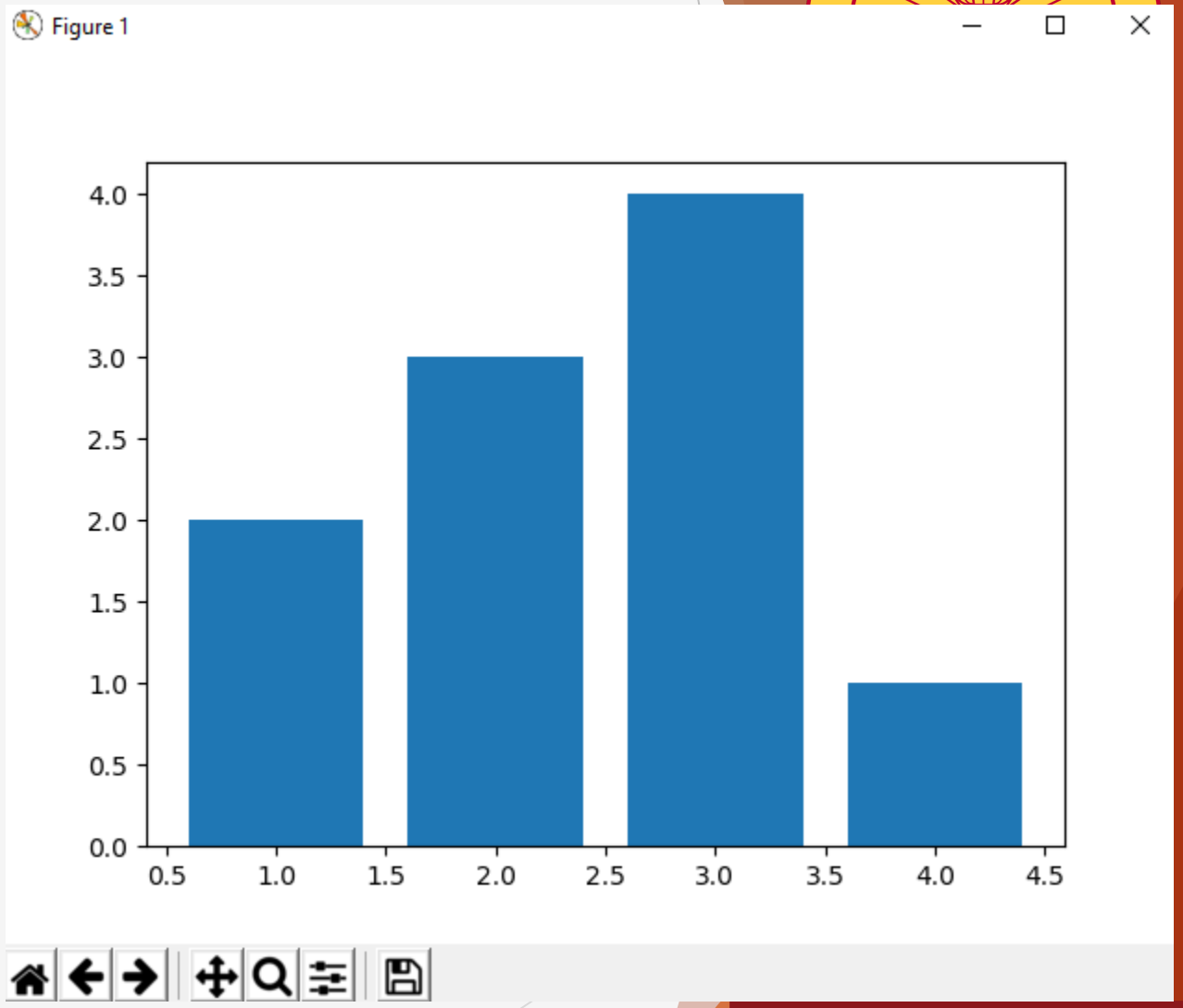
Result from the previous code.



Bar plot

By default, plot generate line plot.
Use **bar()** to plot bar graph.

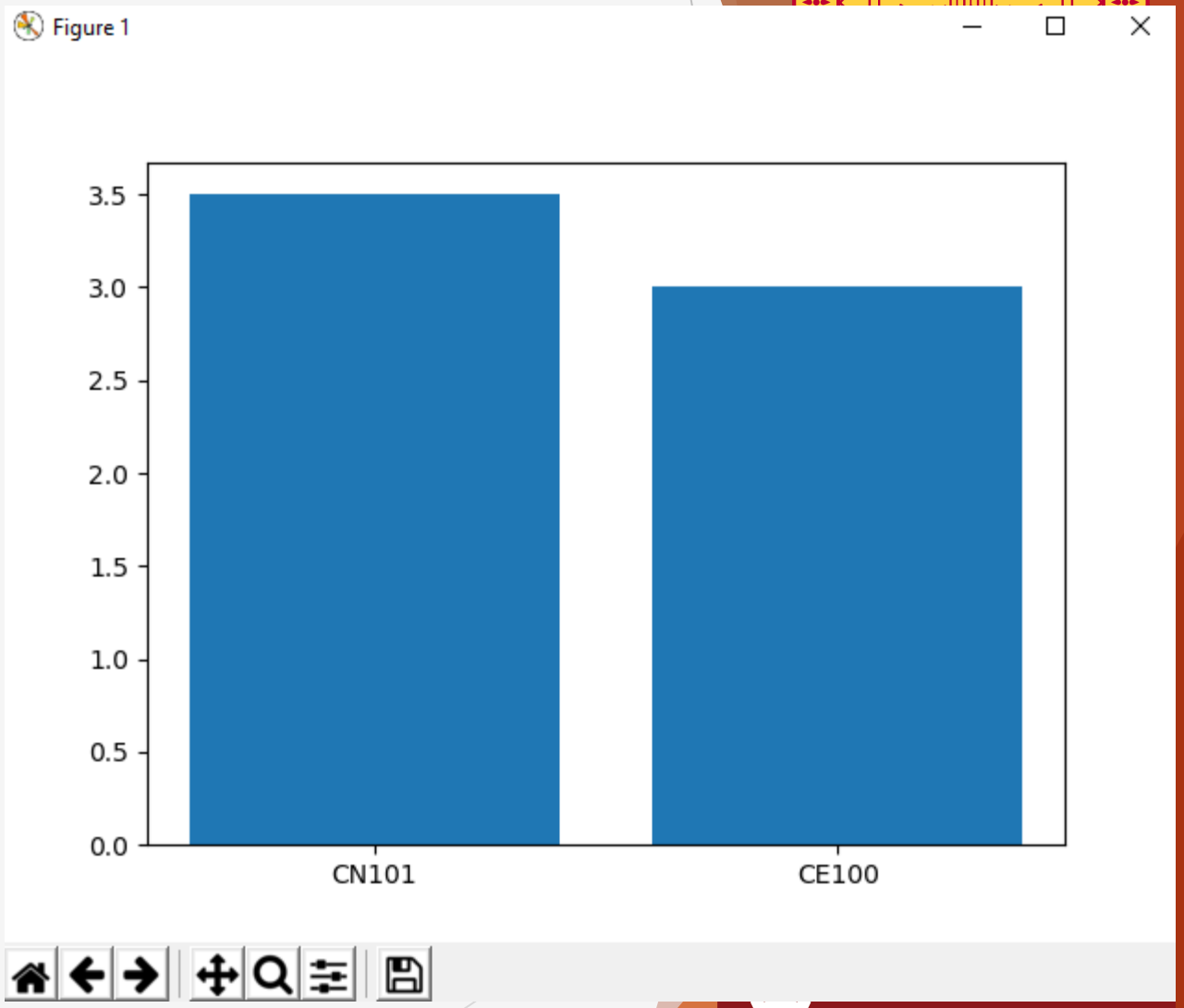
```
import matplotlib.pyplot as plt
import numpy as np

plt.bar([1,2,3,4],[2,3,4,1])
plt.show()
```



Use Strings as x-axis values

List of strings can be used as an input for x-axis values.

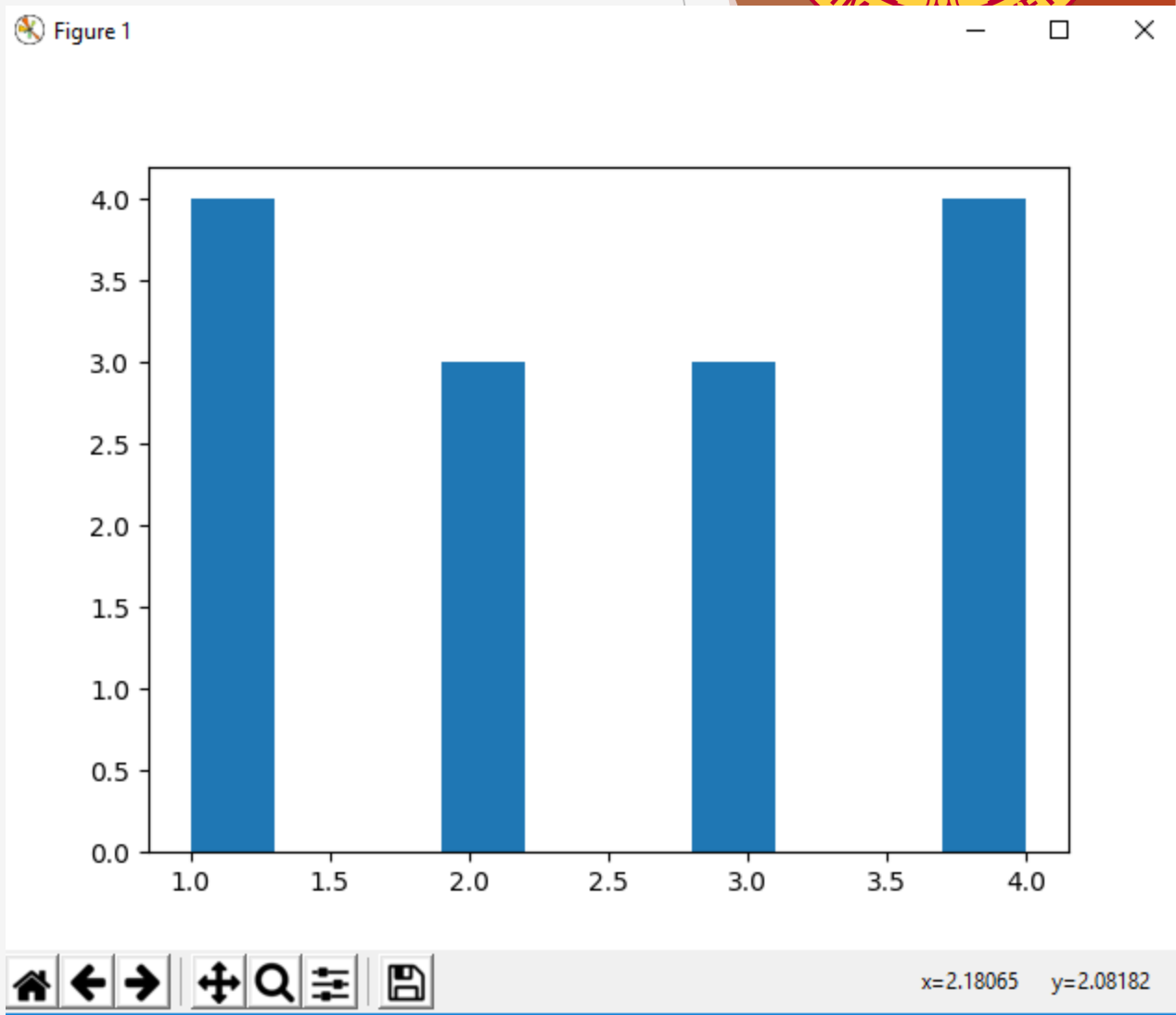
```
import matplotlib.pyplot as plt
import numpy as np

subjects = ['CN101', 'CE100']
classgpas = [3.5,3.00]
plt.bar(subjects,classgpas)
plt.show()
```



Histogram

Histogram plot displays the frequencies of the occurrence of the sample data.

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

data = [1,2,3,4,2,3,4,4,4,1,1,1,2,3]
plt.hist(data)
plt.show()
```



Exercises:

1. Write a program to compute the distance y(t) and speed v(t) of an object that accelerates at a = 0.5 m/s using the following equations.

y(t) = y0+ v0*t + 0.5a*t**
v(t) = v0*t + a*t

Where t is time in second (s).
v0 is the initial speed at t = 0.
y0 is the initial distance at t = 0.

Then plot the distance and speed separately using your preferable plotting styles.

2. Repeat the 1st exercise with multiple objects and plot the distances and speeds of those objects.
Plot their speeds into the same subplot.
Plot their distance into the same subplot.
Allow users to set different initial conditions for the objects.