ch_14_assignment

May 1, 2023

Ch_14_assignment

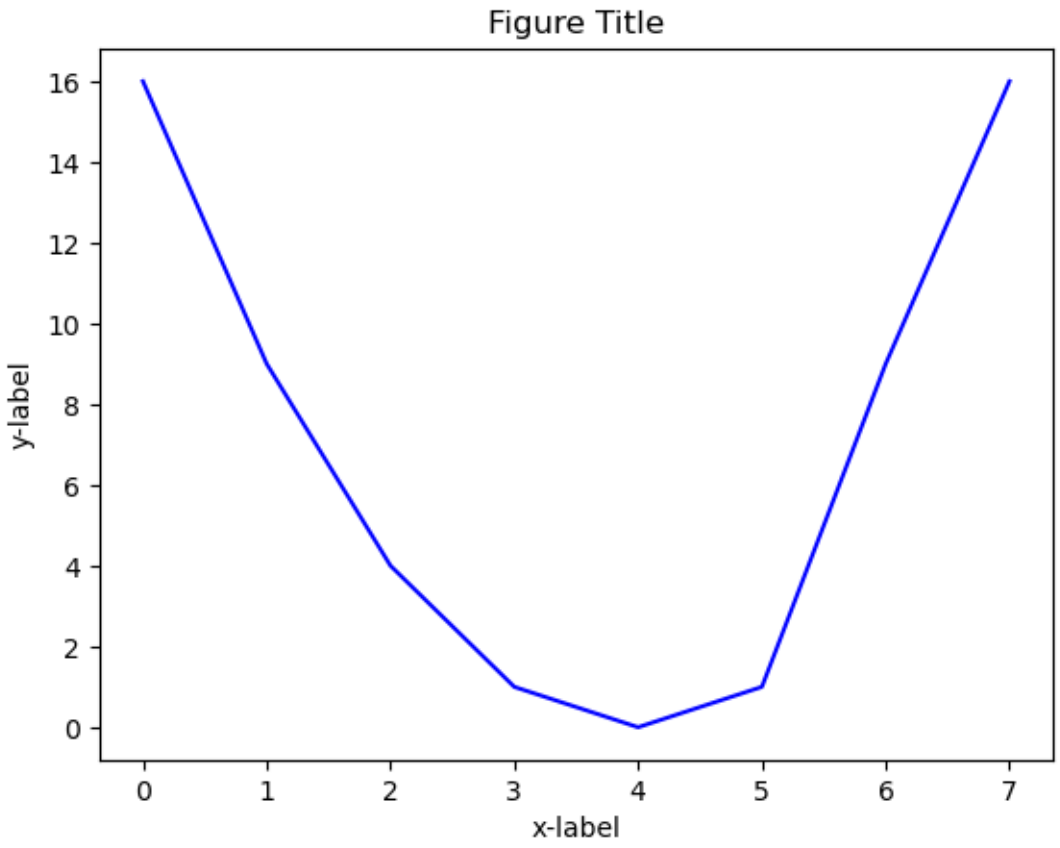
[ ]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

1 Matplotlib

1.1 Drawing a simple curve line(1)

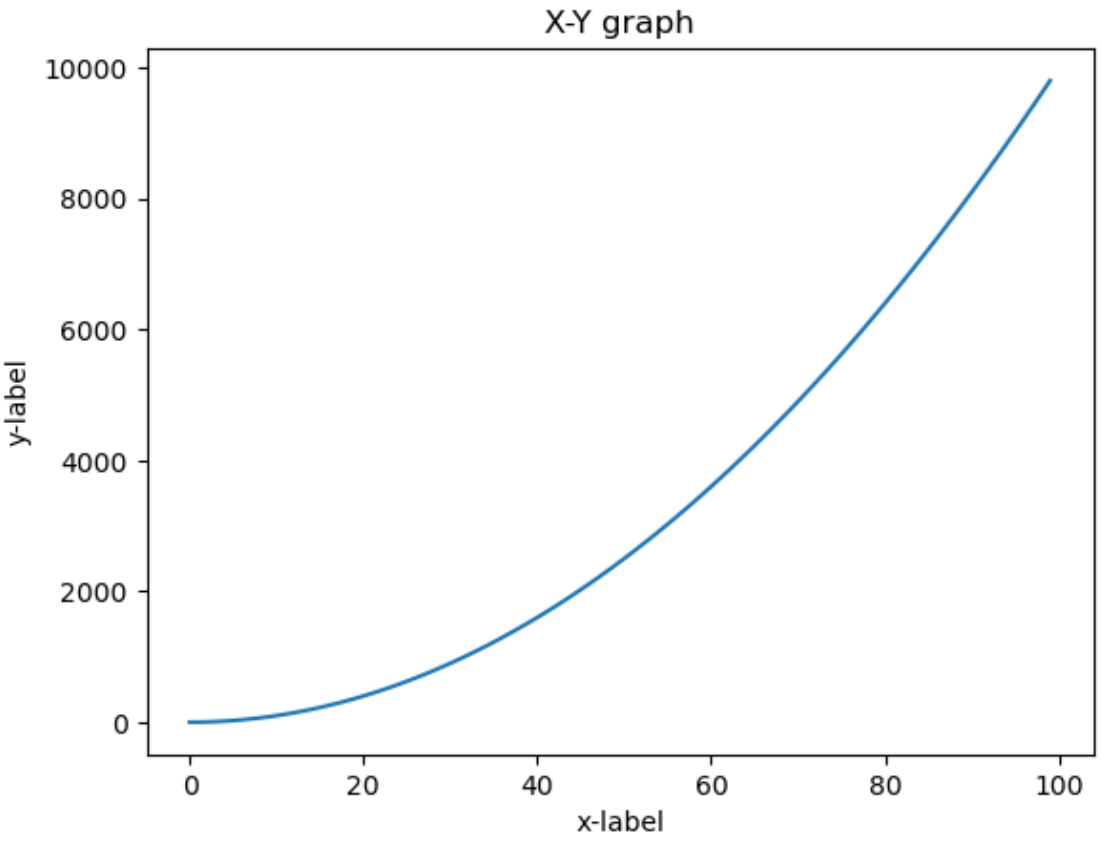
[ ]:
```
import matplotlib.pyplot as plt
```

[ ]:
```
y = (16,9,4,1,0,1,9,16)

plt.plot(y, 'b')
plt.title('Figure Title')
plt.ylabel('y-label')
plt.xlabel('x-label')
plt.show()
```



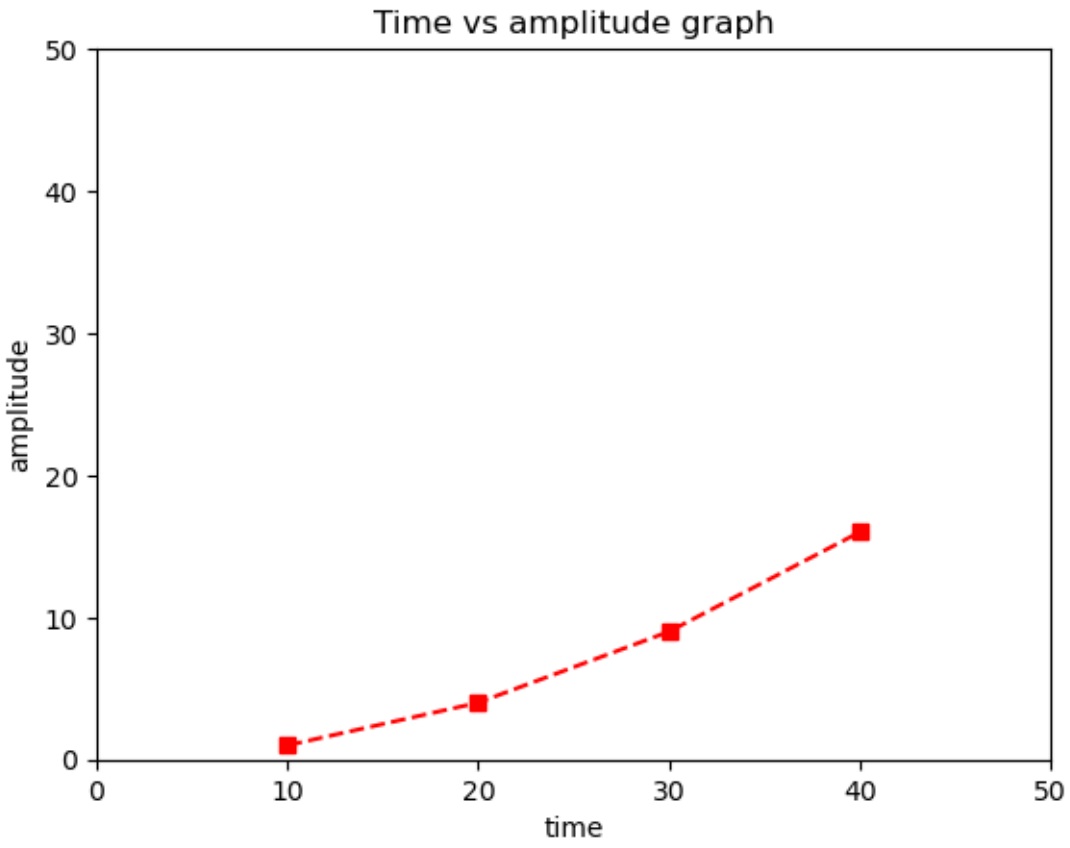
1.2 Drawing a simple curve line(2)

[ ]:
```
X = range(100)
Y = [x**2 for x in X]

plt.plot(X, Y)
plt.title('X-Y graph')
plt.ylabel('y-label')
plt.xlabel('x-label')
plt.show()
```



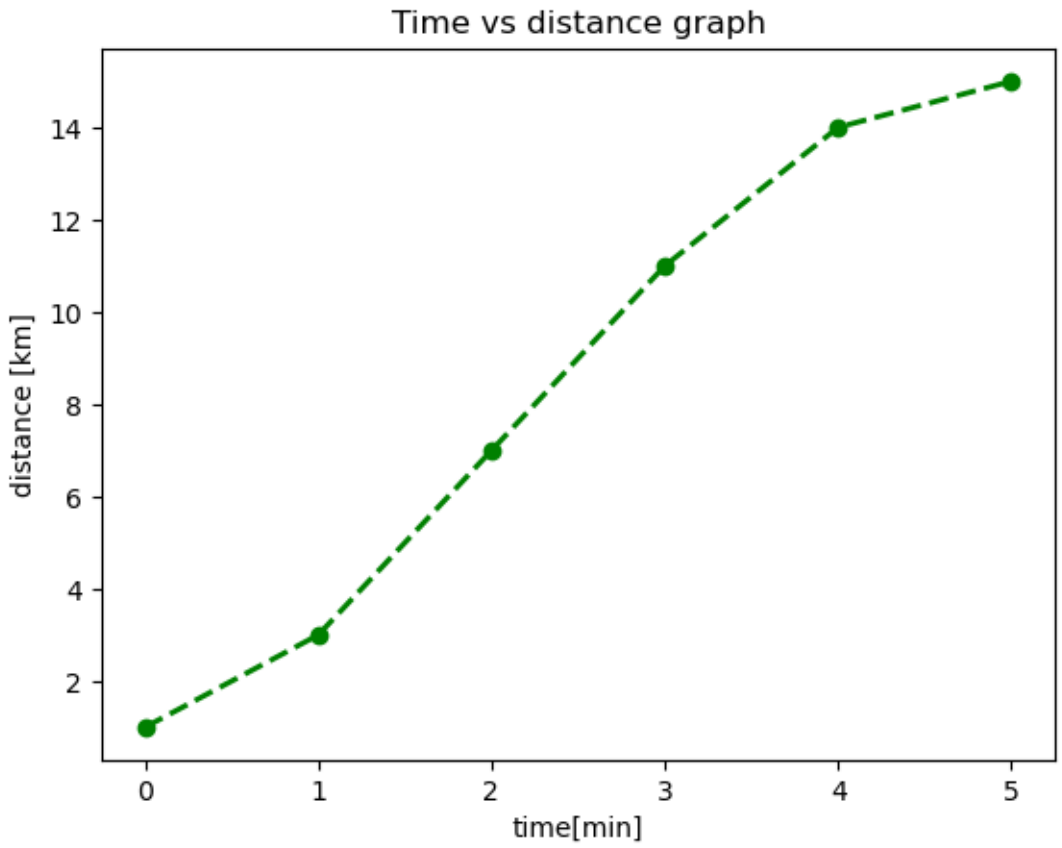
1.3 Drawing a simple curve line(3)

[ ]:
```
plt.plot([10, 20, 30, 40], [1, 4, 9, 16],'rs--')
plt.title('Time vs amplitude graph')
plt.xlabel('time')
plt.ylabel('amplitude')
plt.xlim(0, 50)
plt.ylim(0, 50)
plt.show()
```



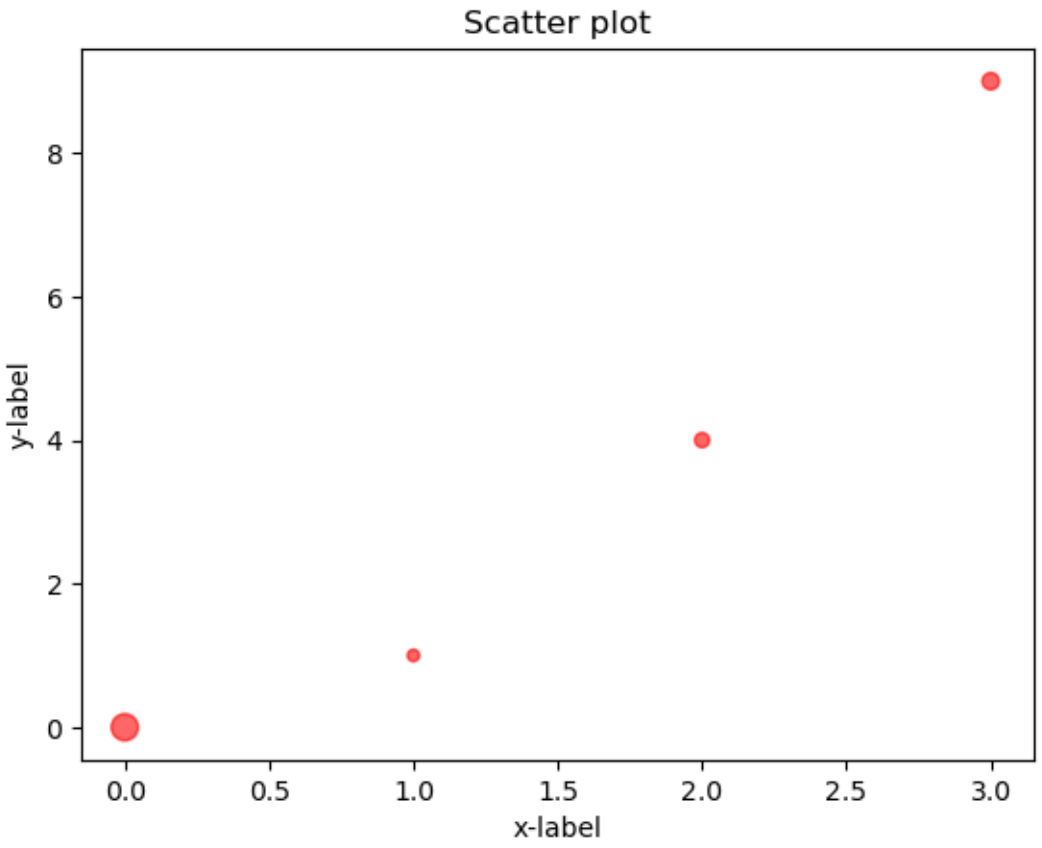
1.4 Drawing a simple curve line(4)

[ ]:
```
data = [1, 3, 7, 11, 14, 15]
plt.plot(data, color='green', marker='.', linestyle='--', linewidth=2,␣
  ↪markersize=12)
plt.title('Time vs distance graph')
plt.xlabel('time[min]')
plt.ylabel('distance [km]')
plt.show()
```



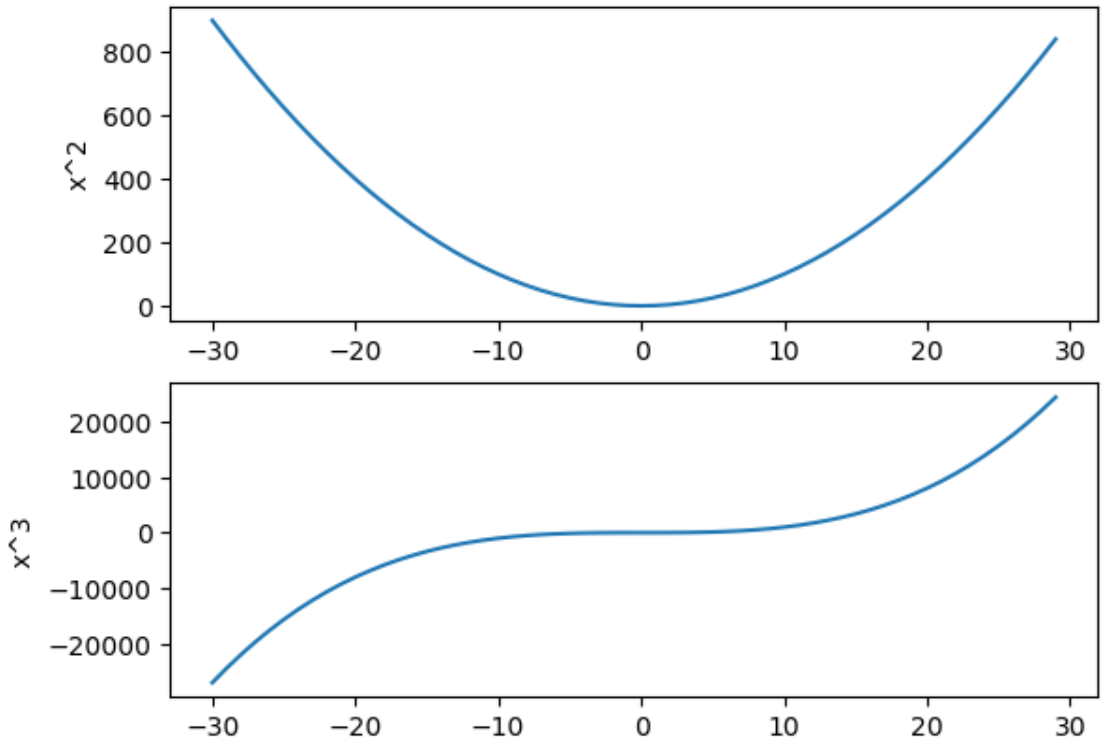
1.5 Drawing a simple graph(1)

[ ]:
```
X = range(0, 4)
Y = [v**2 for v in X]
size = [100, 20, 30, 40]

plt.scatter(x = X, y = Y, s = size, c = 'red', alpha = 0.6)
plt.title('Scatter plot')
plt.xlabel('x-label')
plt.ylabel('y-label')
plt.show()
```



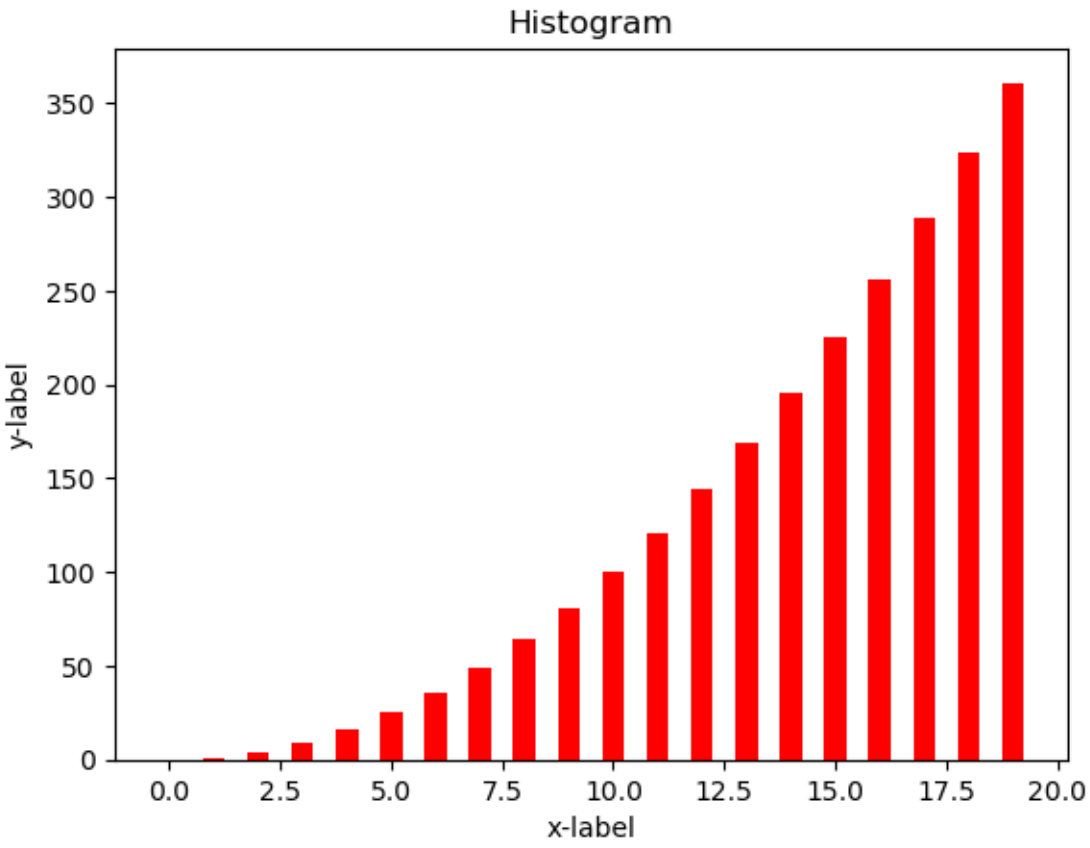
1.6 Drawing a simple graph(2)

[ ]:
```
x = range(-30, 30)
y1 = [v**2 for v in x]
y2 = [v**3 for v in x]

plt.subplot(2,1,1)
plt.ylabel('x^2')
plt.plot(x, y1)

plt.subplot(2,1,2)
plt.ylabel('x^3')
plt.plot(x,y2)
plt.show()
```



1.7 Drawing a simple graph(3)

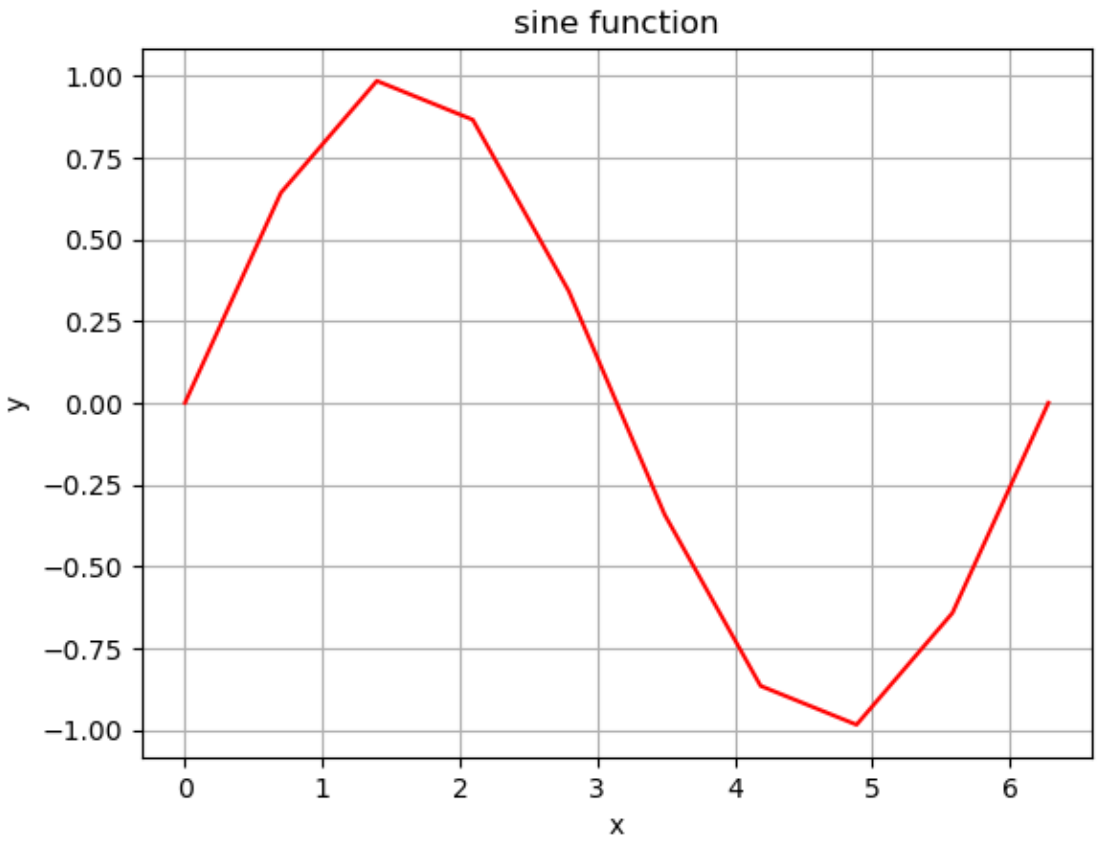
[ ]:
```
x = range(0, 20)
y = [v**2 for v in x]

plt.bar(x, y, width=0.5, color='red')
plt.title('Histogram')
plt.xlabel('x-label')
plt.ylabel('y-label')
plt.show()
```



1.8 Drawing a trigonometric function graph(1)

[ ]:
```
import numpy as np
```

[ ]:
```
x = np.linspace(0, 2*np.pi, 10)
y = np.sin(x)

plt.plot(x, y, 'r-')
plt.title('sine function')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()
```

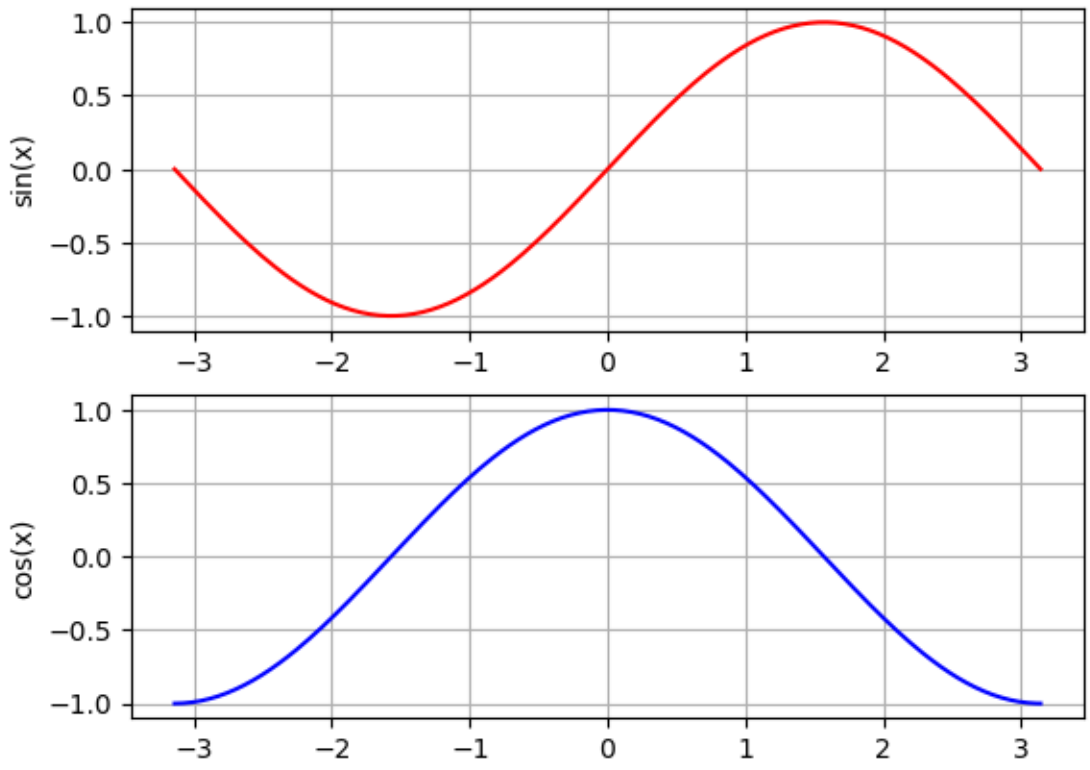


1.9 Drawing a trigonometric function graph(2)

[ ]:
```
x = np.linspace(-np.pi, np.pi, 200)
y1 = np.sin(x)
y2 = np.cos(x)

plt.title('trigonometric functions')
plt.subplot(2, 1, 1)
plt.ylabel('sin(x)')
plt.plot(x, y1, 'r-')
plt.grid()
plt.subplot(2, 1, 2)
plt.ylabel('cos(x)')
plt.plot(x, y2, 'b-')
plt.grid()
plt.show()
```

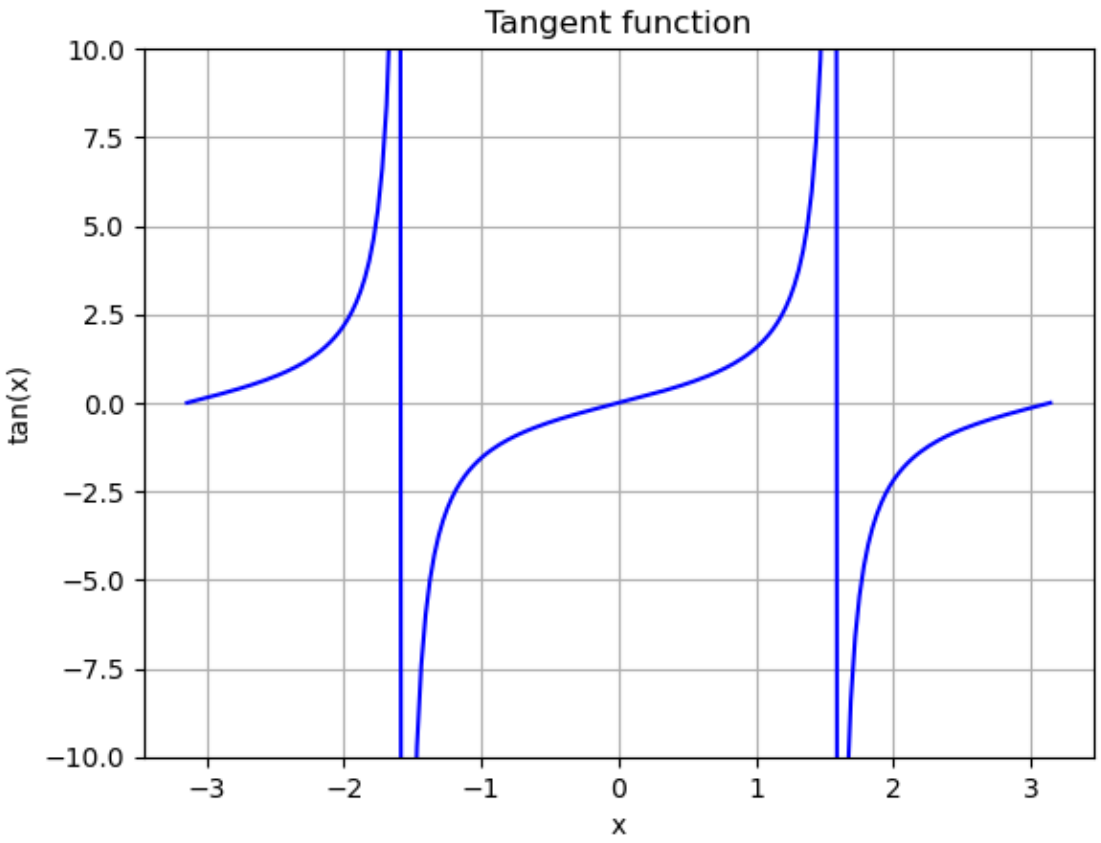
```
/var/folders/r1/8vnnkyjn3h3b_tnp2010w6nm0000gn/T/ipykernel_4281/458831005.py:6:
MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
```

```
ax.remove() as needed.
  plt.subplot(2, 1, 1)
```



1.10 Drawing a trigonometric function graph(3)

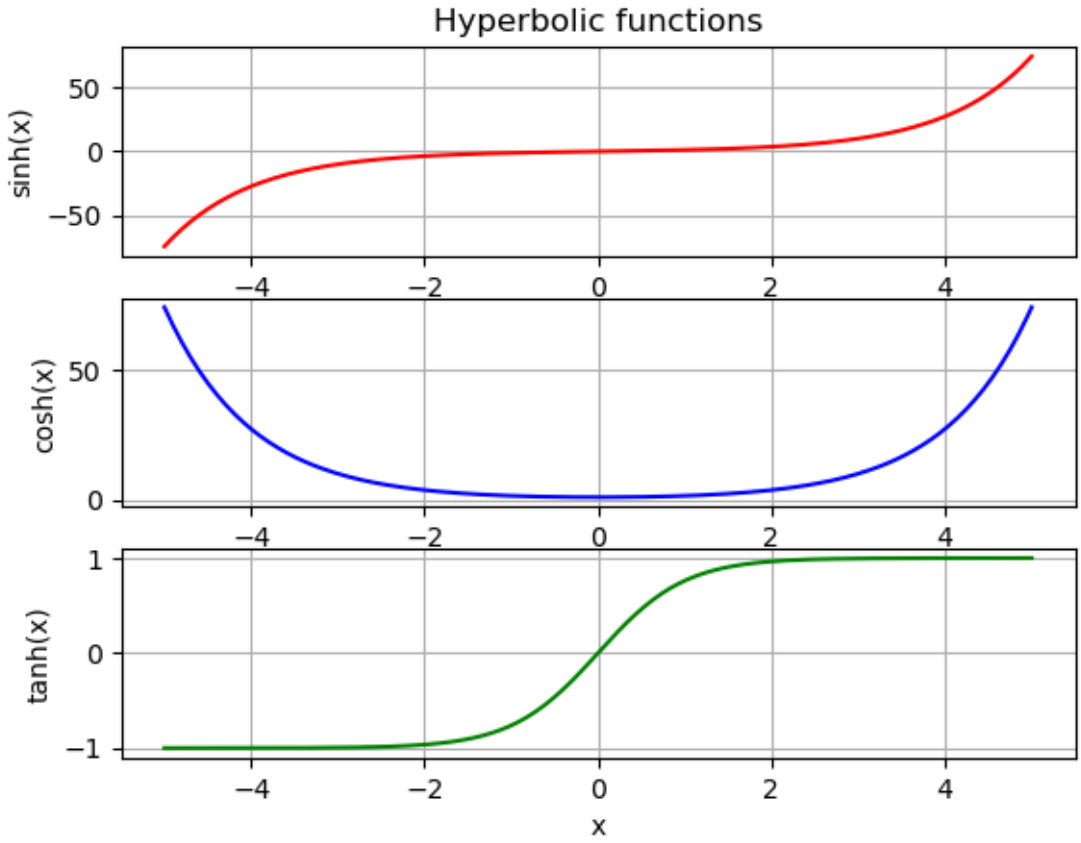
[ ]:
```
x = np.linspace(-np.pi, np.pi, 200)
y = np.tan(x)

plt.plot(x, y, 'b-')
plt.title('Tangent function')
plt.ylabel('tan(x)')
plt.xlabel('x')
plt.ylim(-10, 10)
plt.grid()
plt.show()
```



1.11 Drawing a simple trigonometric function graph(4)

[ ]:
```
x = np.linspace(-5, 5, 200)
y1 = np.sinh(x)
y2 = np.cosh(x)
y3 = np.tanh(x)

plt.subplot(3,1,1)
plt.title('Hyperbolic functions')
plt.ylabel('sinh(x)')
plt.plot(x, y1, 'r-')
plt.grid()

plt.subplot(3,1,2)
plt.ylabel('cosh(x)')
plt.plot(x, y2, 'b-')
plt.grid()
plt.subplot(3,1,3)

plt.xlabel('x')
```

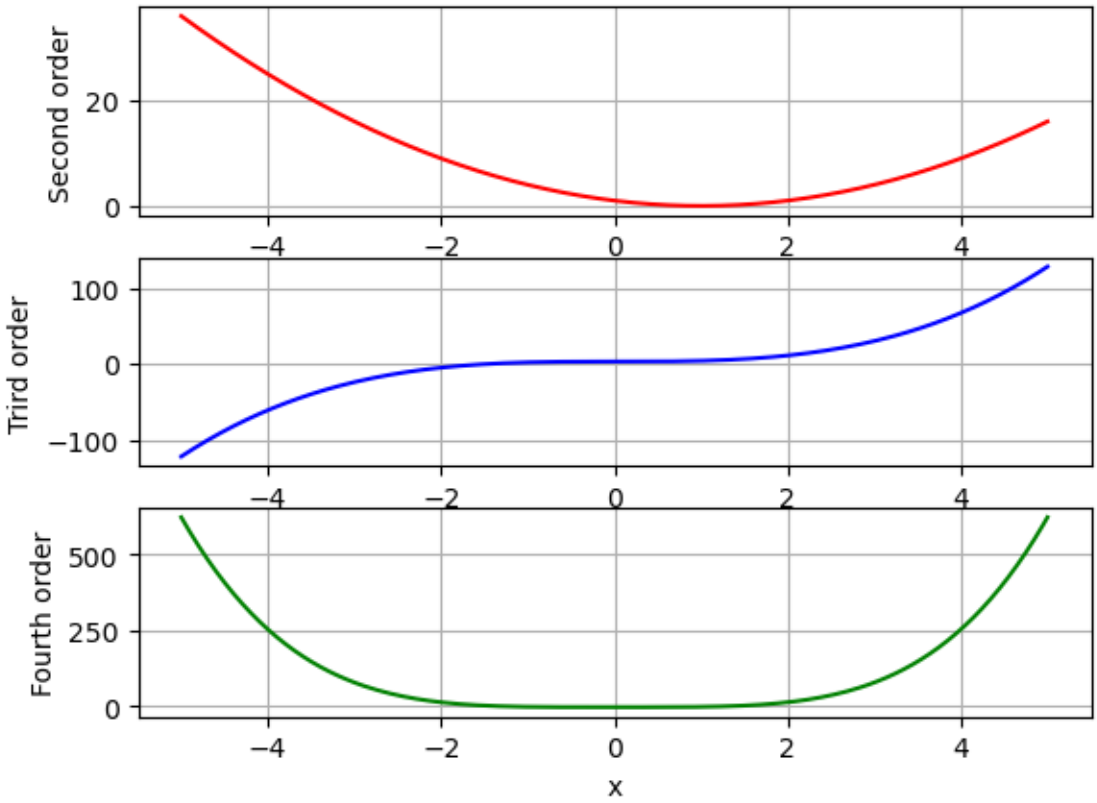
```
plt.ylabel('tanh(x)')
plt.plot(x, y3, 'g-')
plt.grid()

plt.show()
```



1.12 Drawing a polynomial graph(1)

[ ]:
```
x = np.linspace(-5, 5, 200)
y1 = x**2 - 2*x + 1
y2 = x**3 + 3
y3 = x**4 - 2

plt.subplot(3,1,1)
plt.ylabel('Second order')
plt.plot(x, y1, 'r-')
plt.grid()

plt.subplot(3,1,2)
```

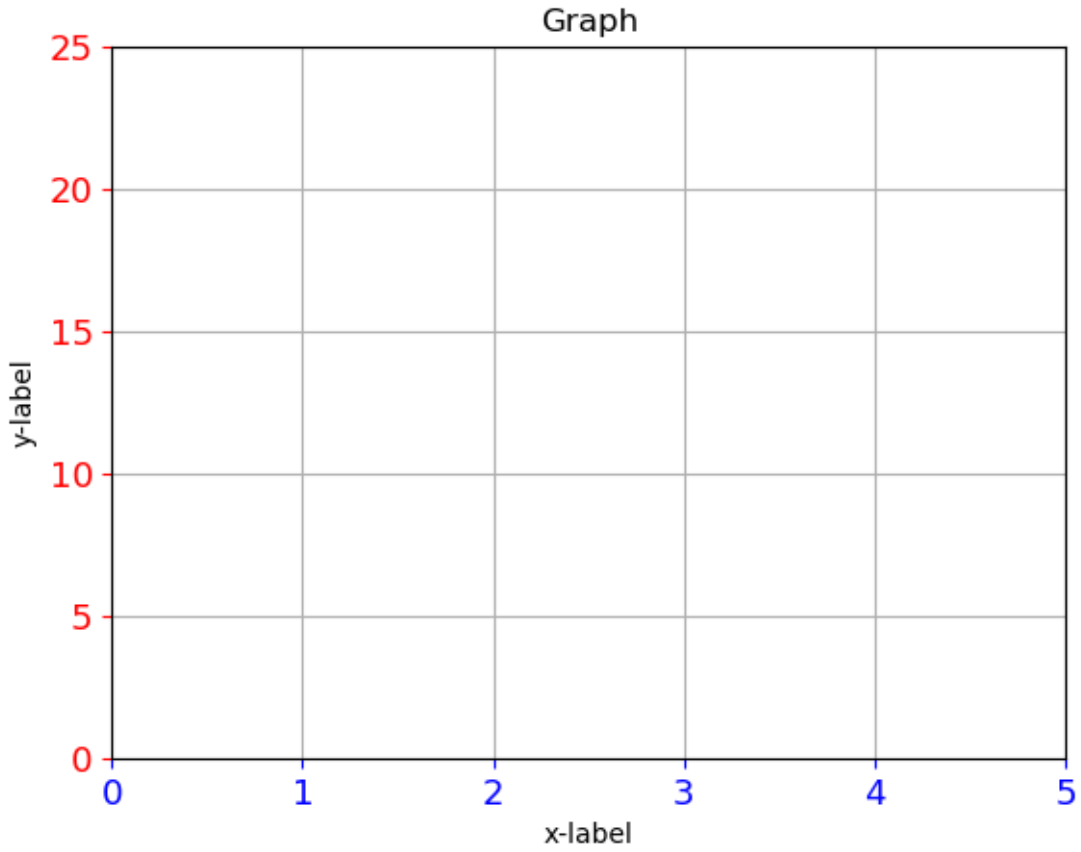
```
plt.ylabel('Trird order')
plt.plot(x, y2, 'b-')
plt.grid()

plt.subplot(3,1,3)
plt.xlabel('x')
plt.ylabel('Fourth order')
plt.plot(x, y3, 'g-')
plt.grid()

plt.show()
```



1.13 Editing tick of graph

[ ]:
```
plt.title('Graph')
plt.xlabel('x-label')
plt.ylabel('y-label')

plt.xticks(np.arange(6), ('0','1','2','3','4','5'))
plt.yticks([0,5,10,15,20,25], ('0','5','10','15','20','25'))
```

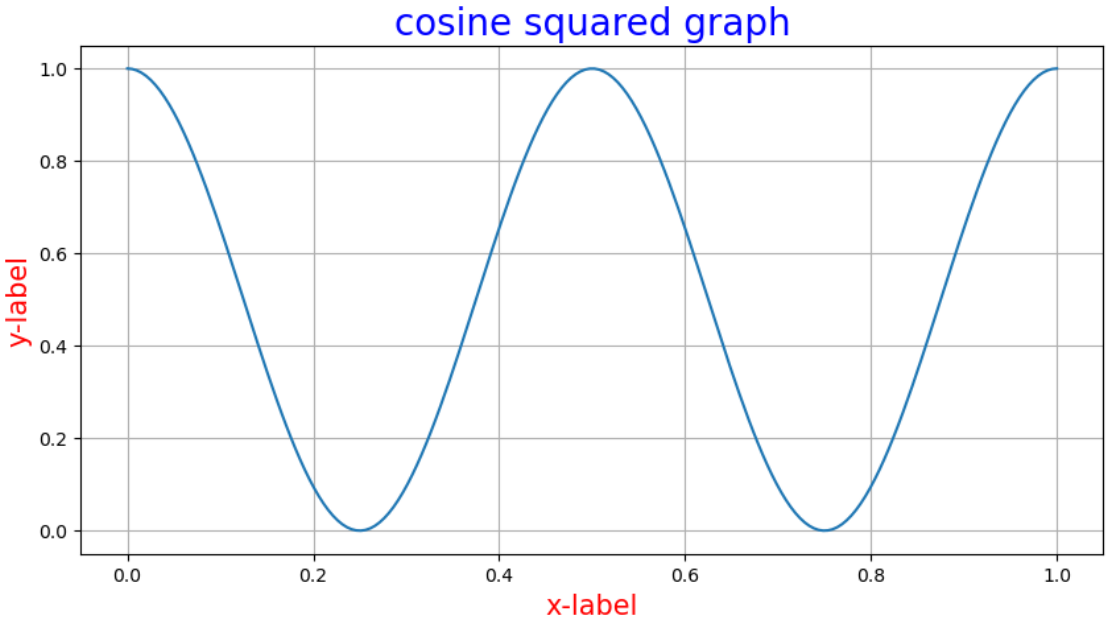
```
plt.tick_params(axis='x', labelsize=13, colors= 'b')
plt.tick_params(axis='y', labelsize=13, colors= 'r')

plt.grid()
plt.show()
```



1.14 Changing size of graph

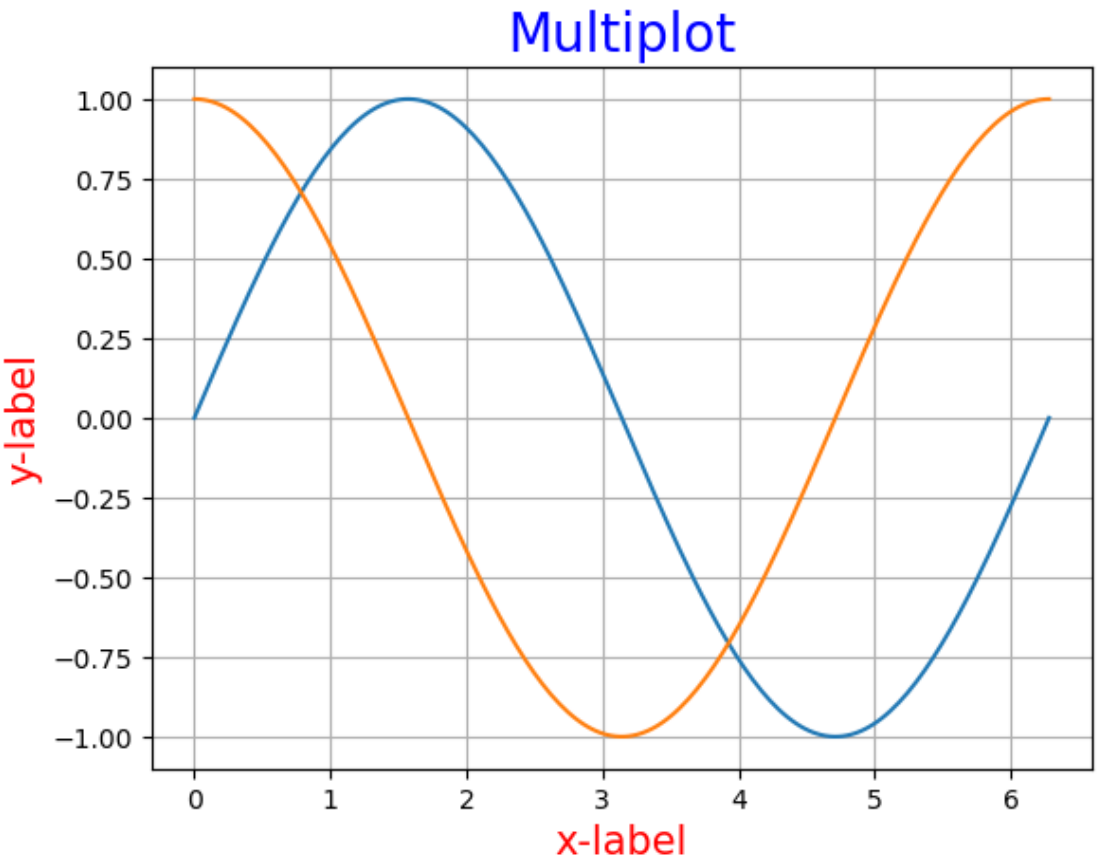
[ ]:
```
plt.figure(figsize = (10, 5))
plt.title('cosine squared graph', size=20, color='b')
plt.xlabel('x-label', size=15, color='r')
plt.ylabel('y-label', size=15, color='r')

x = np.linspace(0, 1, 200)
y = np.cos(np.pi*2*x)**2
plt.tick_params(axis='x', labelsize=10, colors='black')
plt.tick_params(axis='y', labelsize=10, colors='black')

plt.plot(x,y)
```

```
plt.grid()
plt.show()
```



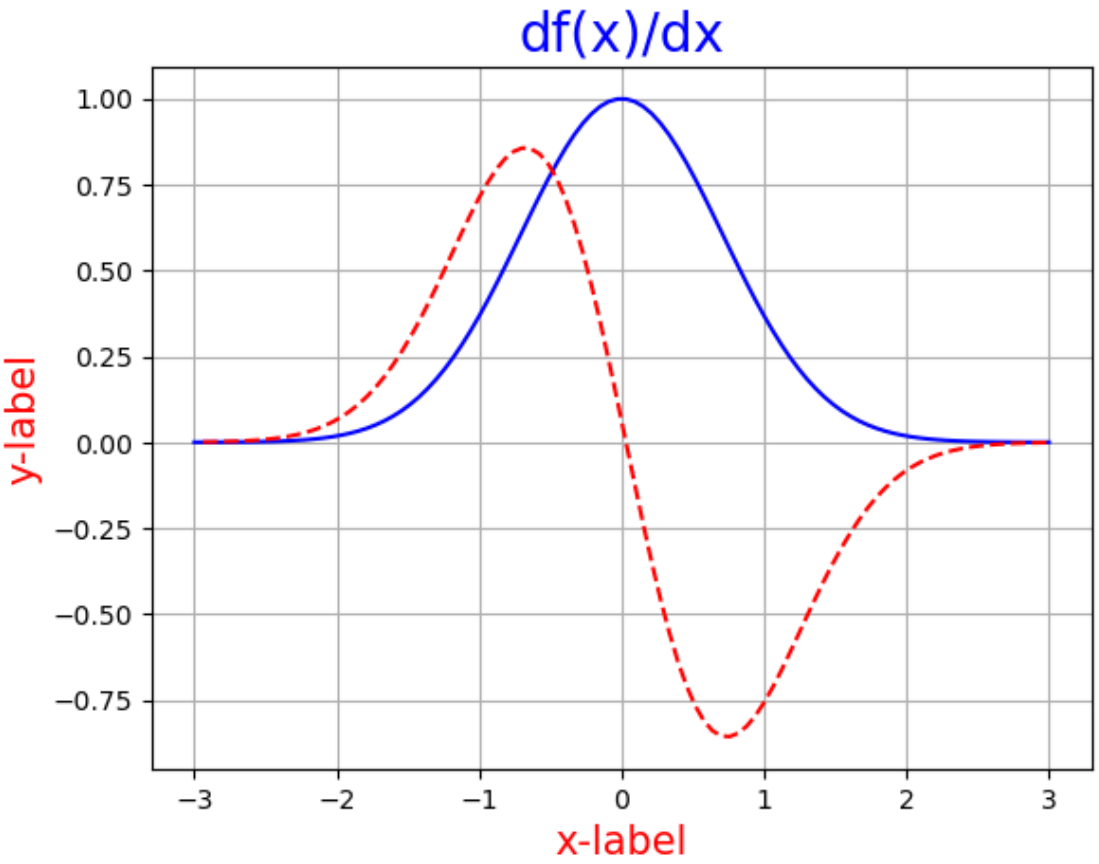
1.15 Drawing a multi curve(1)

[ ]:
```
plt.title('Multiplot', size=20, color='b')
plt.xlabel('x-label', size=15, color='r')
plt.ylabel('y-label', size=15, color='r')

x = np.linspace(0, 2*np.pi, 200)
y1 = np.sin(x)
y2 = np.cos(x)
plt.plot(x, y1)
plt.plot(x, y2)
plt.grid()
plt.show()
```



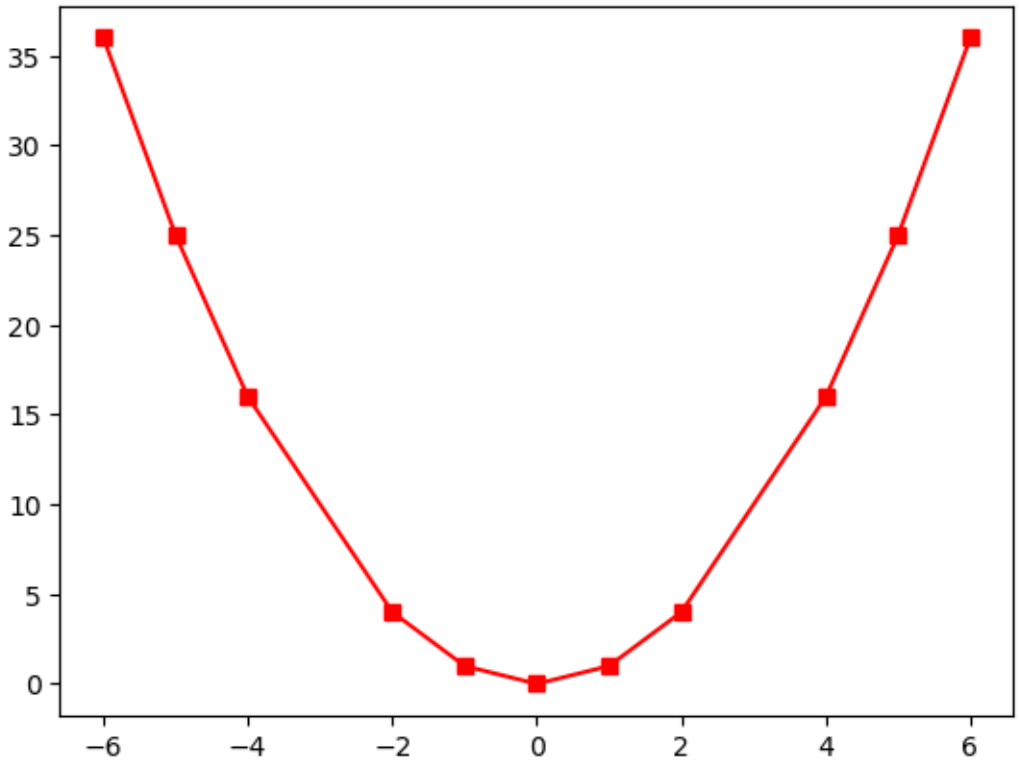
1.16 Drawing a multi curve(2)

[ ]:
```
def plot_slope(X, Y):
    Xs = X[1 :] - X[: -1]
    Ys = Y[1 :] - Y[: -1]
    plt.plot(X[1:], Ys/Xs, 'r--' )

plt.title('df(x)/dx', size=20, color='b')
plt.xlabel('x-label', size=15, color='r')
plt.ylabel('y-label', size=15, color='r')

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

plt.plot(x, y, 'b-')
plot_slope(x, y)
plt.grid()
plt.show()
```



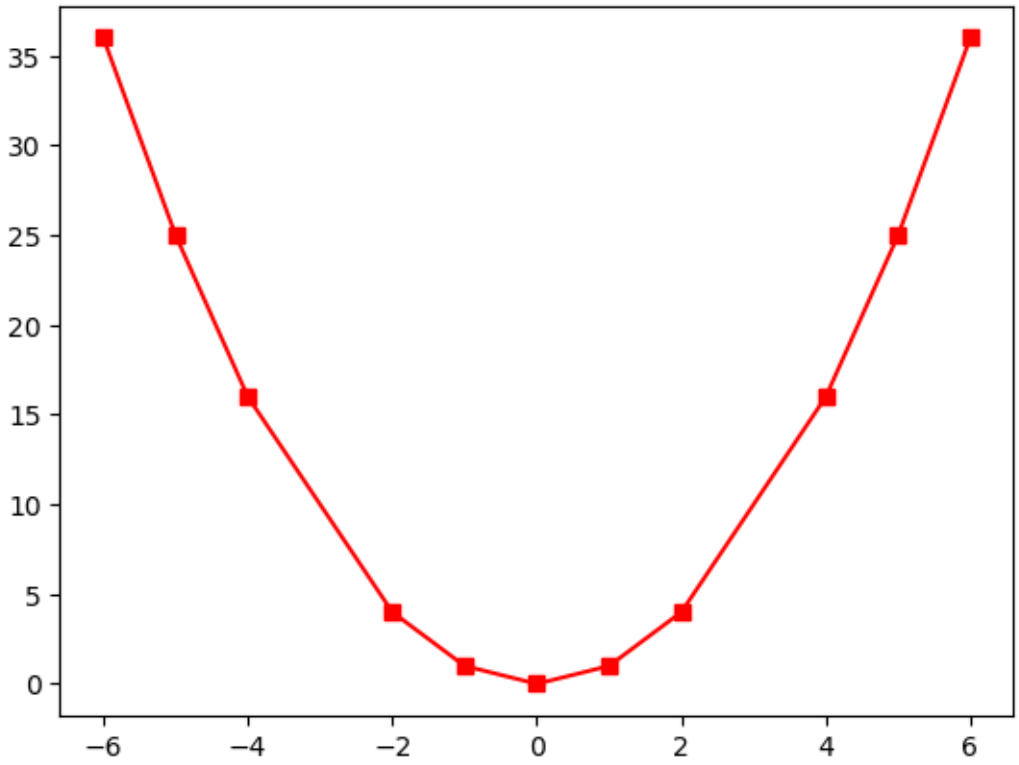
1.17 Drawing a graph by the file (1)

[ ]:
```
X, Y = [], []

for line in open('simple.txt', 'r'):
    values = [float(s) for s in line.split()]
    X.append(values[0])
    Y.append(values[1])

plt.plot(X,Y, 'rs-')
plt.show()
```



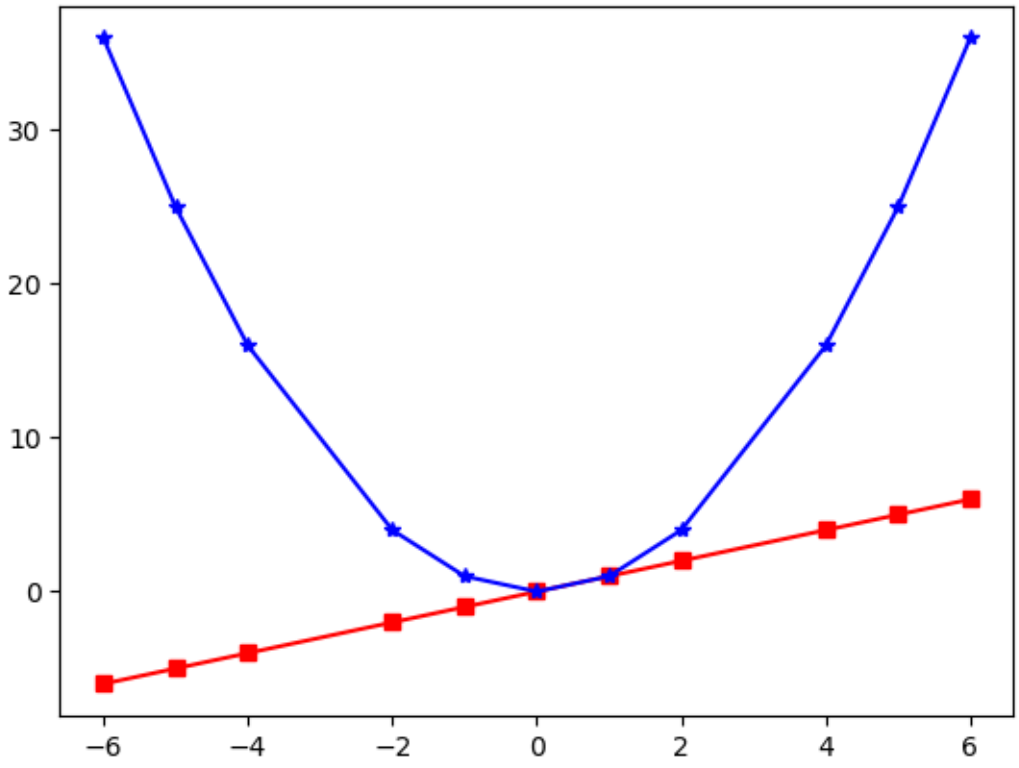
1.18 Drawing a graph by the file (2)

[ ]:
```
data = np.loadtxt('simple.txt')
X = data[:, 0]
Y = data[:, 1]

plt.plot(X, Y, 'rs-')
plt.show()
```



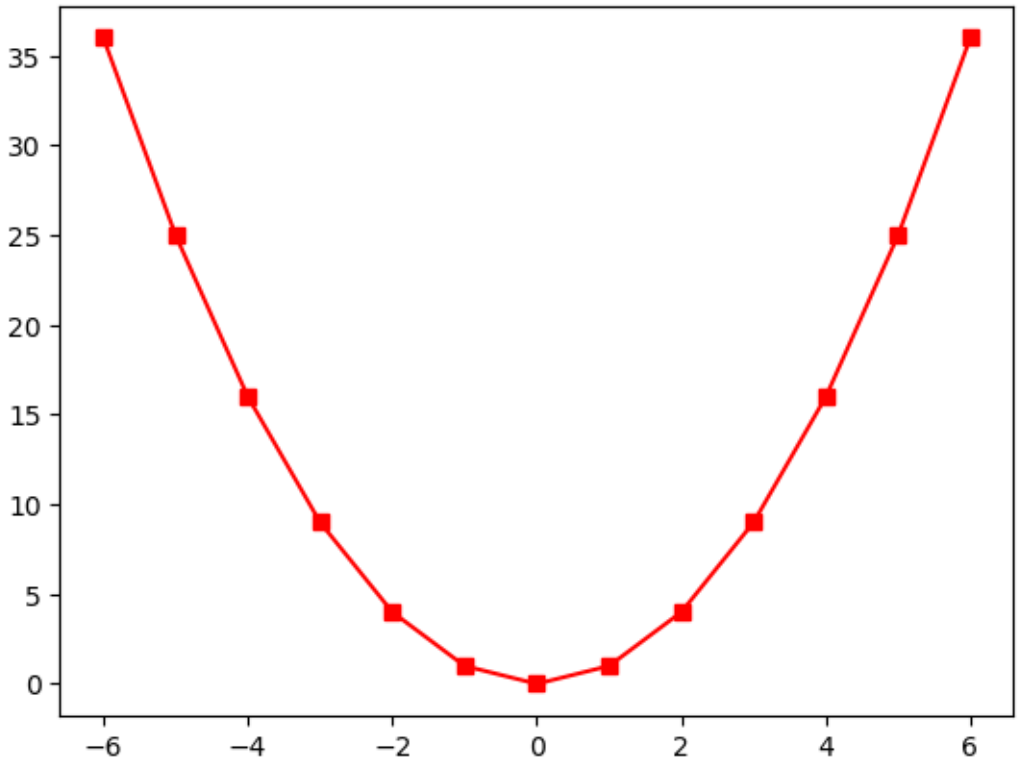
1.19 Drawing a graph by the file (3)

[ ]:
```
data = np.loadtxt('simple.txt')
col = ['red', 'blue']
mar = ['s', '*']

i = 0
for column in data.T:
    plt.plot(data[:,0], column, color = col[i], marker = mar[i])
    i+=1

plt.show()
```



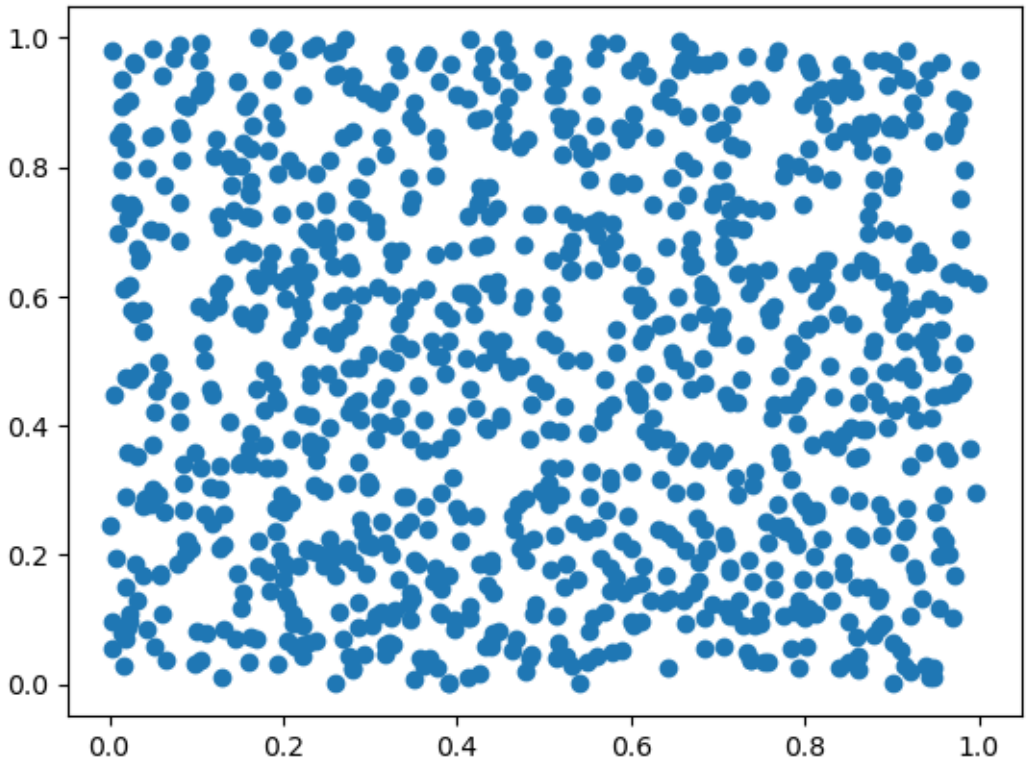
1.20 Drawing a graph from excel file

[ ]:
```
import csv

X, Y = [], []

f = open('sim.csv', 'r', encoding='utf-8-sig')
rdr = csv.reader(f)
for line in rdr:
    X.append(float(line[0]))
    Y.append(float(line[1]))
f.close()

plt.plot(X, Y, 'rs-')
plt.show()
```



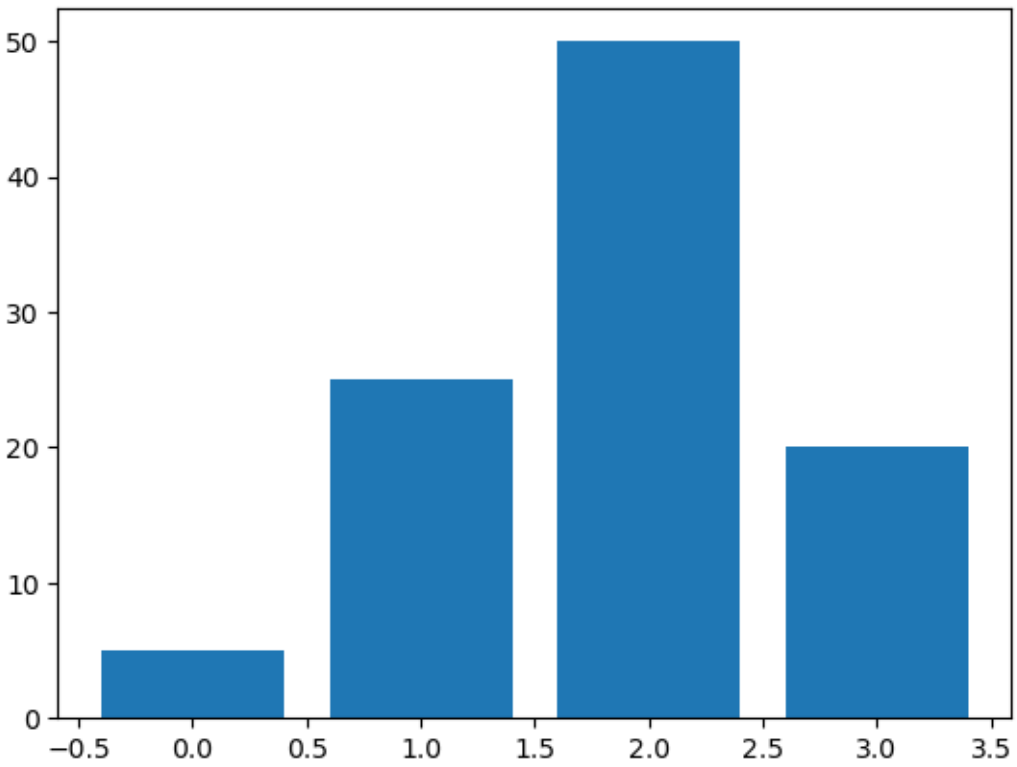
1.21 Scattering

[ ]:
```
import random

count = 1024
X = [random.random() for i in range(count)]
Y = [random.random() for i in range(count)]

plt.scatter(X,Y)
plt.show()
```

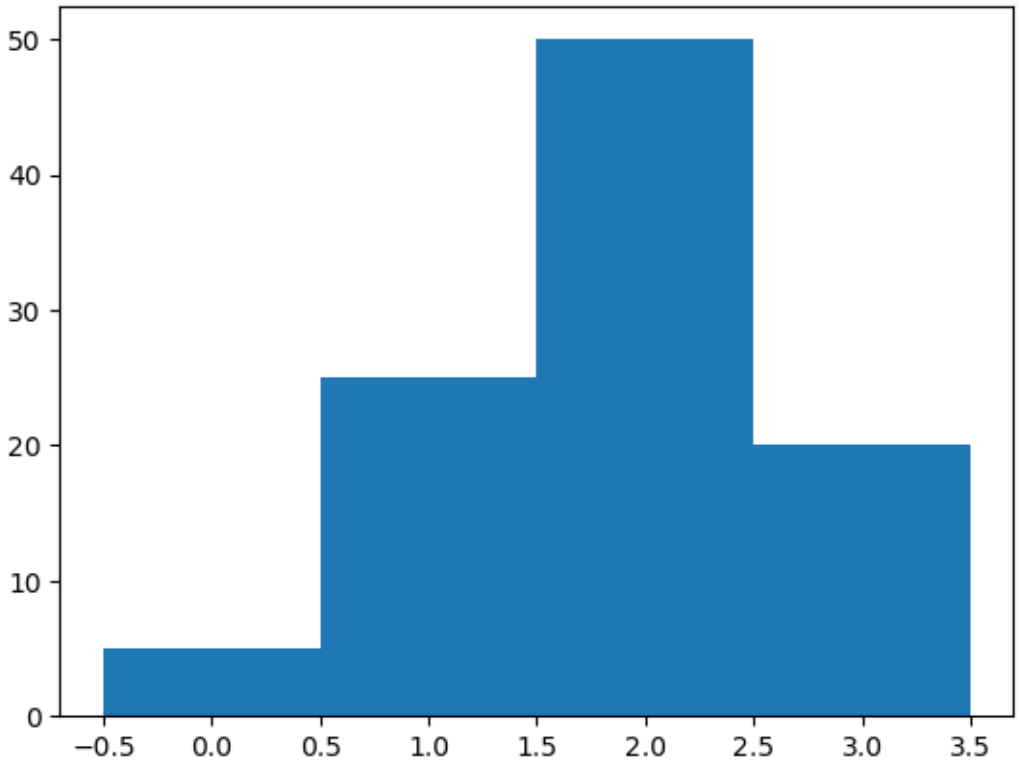


1.22 Bar chart

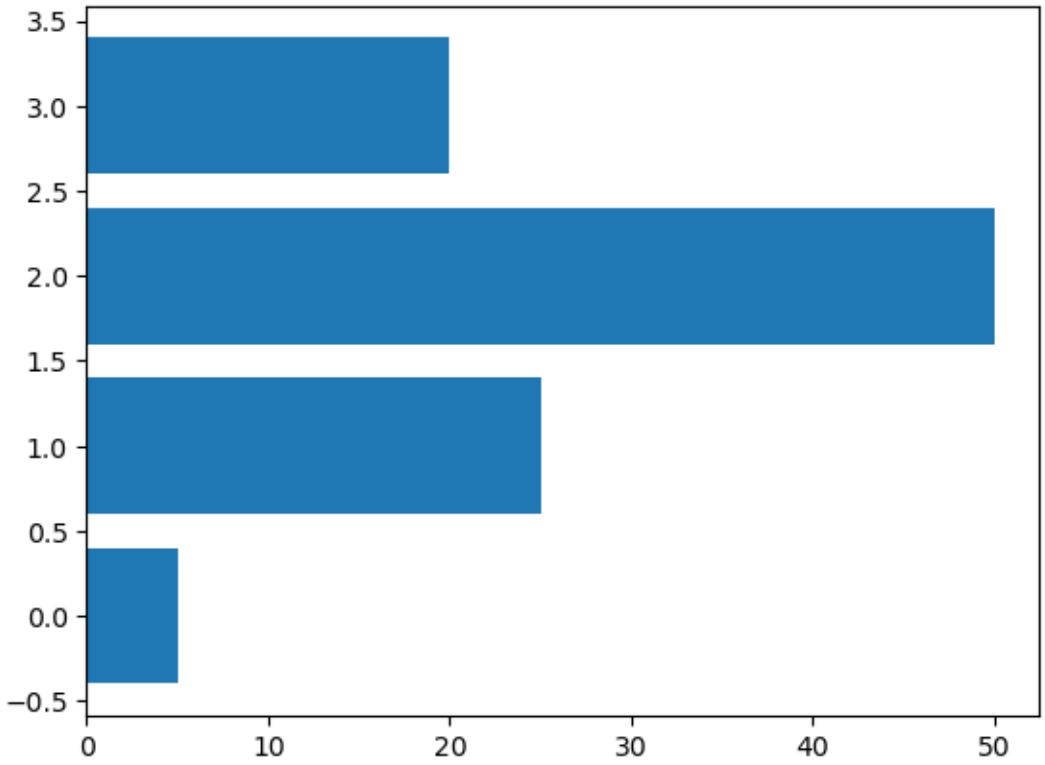
[ ]:
```
data = [5., 25., 50., 20.]

plt.bar(range(len(data)), data)
plt.show()
```



[ ]:
```
data = [5., 25., 50., 20.]

plt.bar(range(len(data)), data, width = 1.)
plt.show()
```

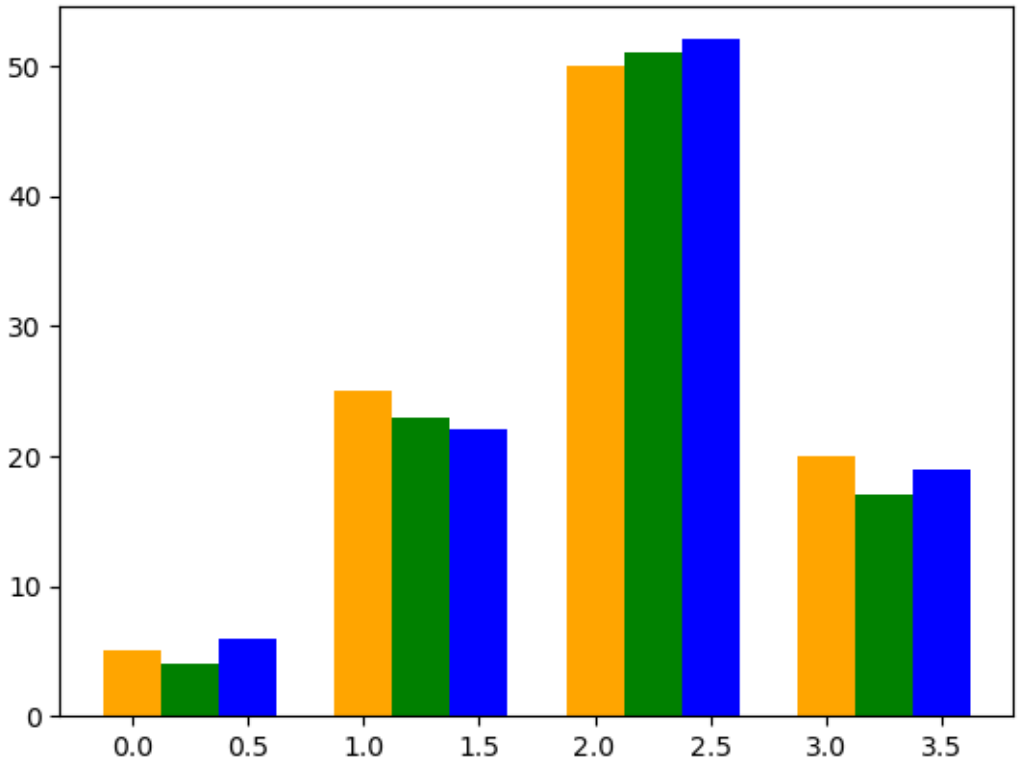


[ ]:
```
data = [5., 25., 50., 20.]

plt.barh(range(len(data)), data)
plt.show()
```



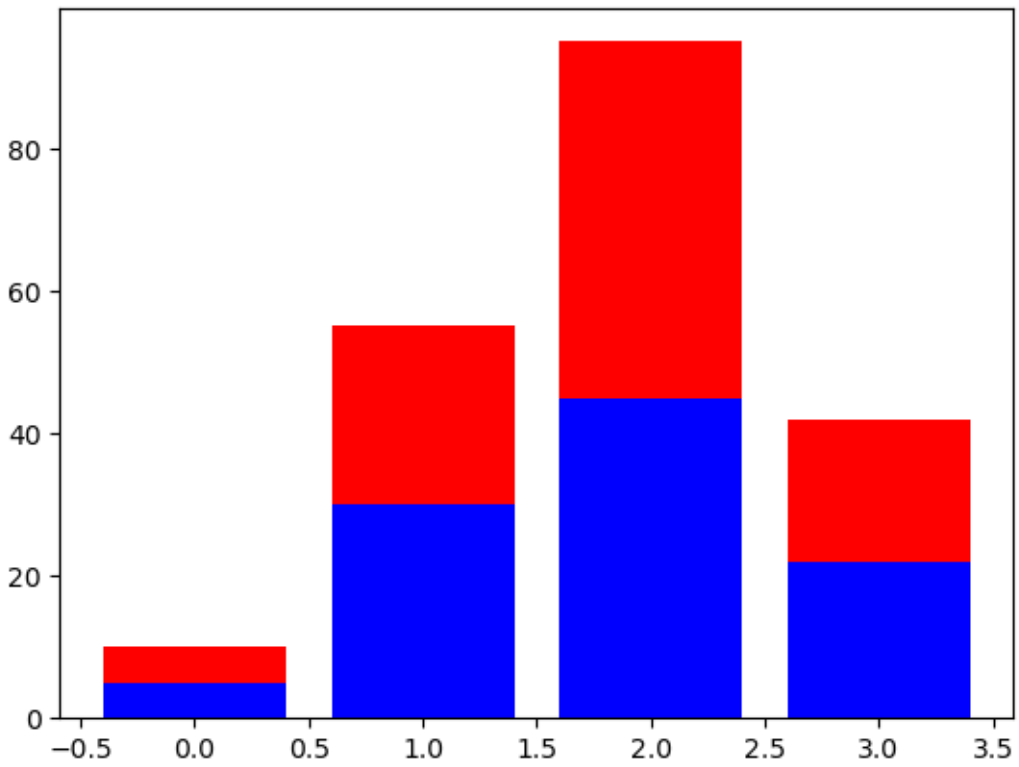
1.23 Multi bar chart

[ ]:
```
data = [[5., 25., 50., 20.,],
        [4., 23., 51., 17. ],
        [6., 22., 52., 19. ]]

plt.bar(range(4), data[0], width=0.25, color='orange')
plt.bar([x+0.25 for x in range(4)], data[1], width=0.25, color='green')
plt.bar([x+0.50 for x in range(4)], data[2], width=0.25, color='blue')
plt.show()
```

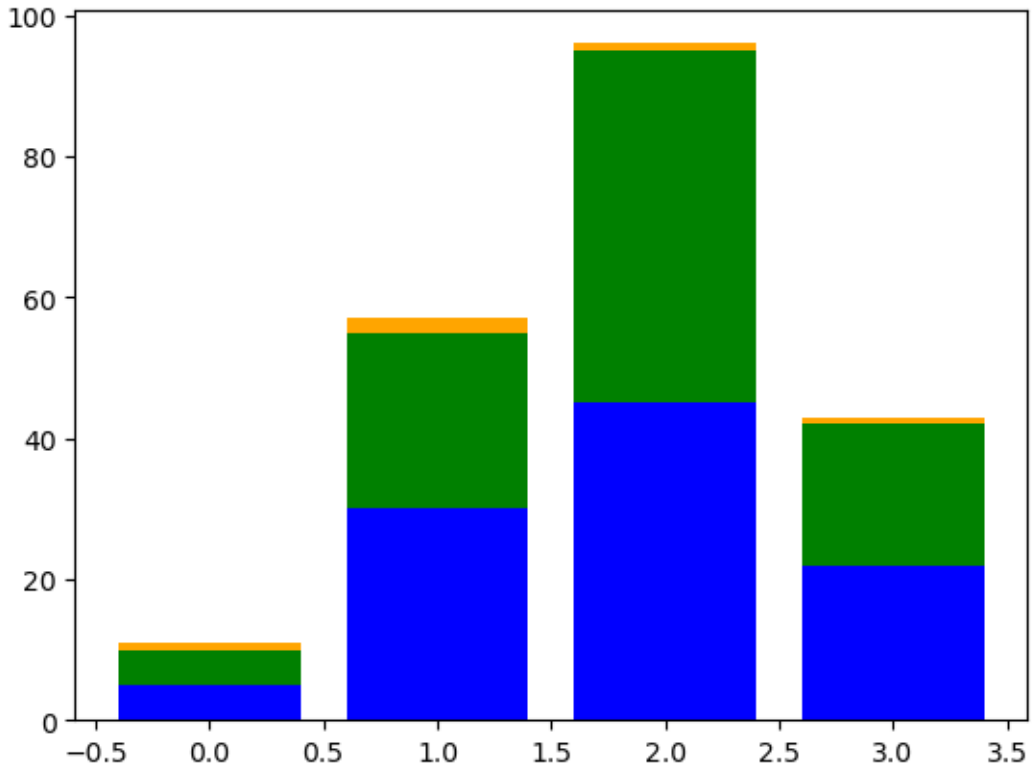


1.24 Split bar chart

[ ]:
```
A = [5., 30., 45., 22. ]
B = [5., 25., 50., 20. ]

X = range(4)

plt.bar(X, A, color='blue')
plt.bar(X, B, color='red', bottom = A)
plt.show()
```

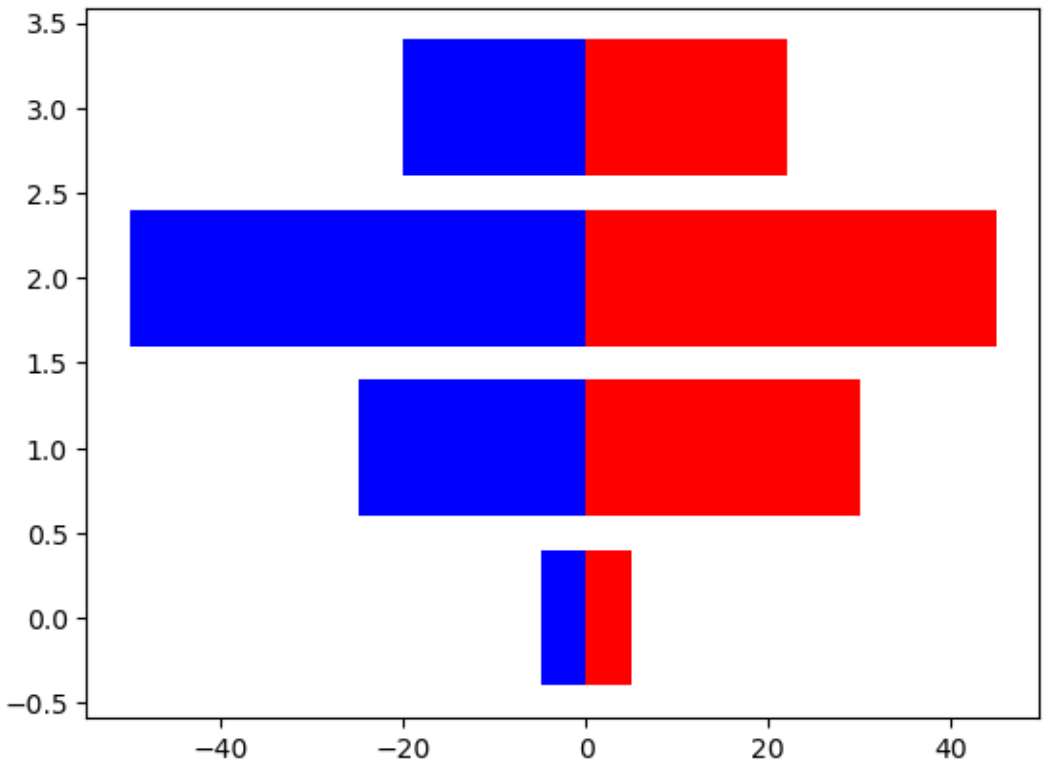


[ ]:
```
data = np.array([[5., 30., 45., 22. ],
                 [5., 25., 50., 20. ],
                 [1., 2., 1., 1. ]])

color_list = ['blue', 'green', 'orange']

X = np.arange(data.shape[1])
for i in range(data.shape[0]):
    S = np.sum(data[:i], axis=0)
    plt.bar(X, data[i], bottom=S, color=color_list[i%len(color_list)])

plt.show()
```



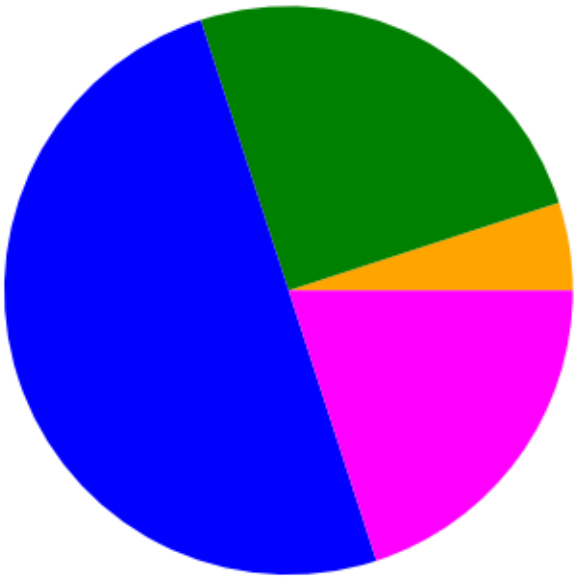
1.25 Two-way bar chart

[ ]:
```
A_pop = [5., 30., 45., 22.]
B_pop = [5., 25., 50., 20.]

X = range(4)
plt.barh(X, A_pop, color='r')
plt.barh(X, [-value for value in B_pop], color='b')
plt.show()
```



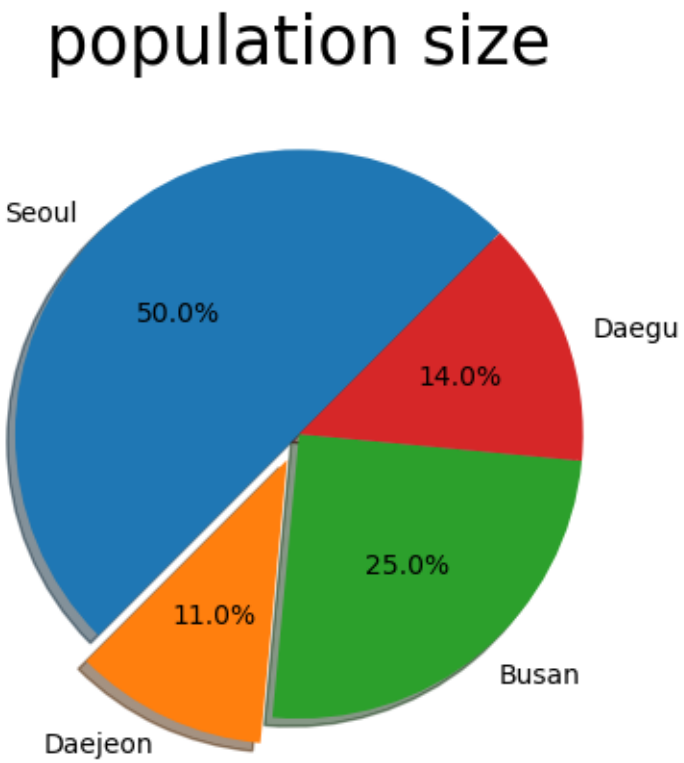
1.26 Pie chart(1)

```
[ ]: data = [5, 25, 50, 20]
color_list = ['orange', 'green', 'blue', 'magenta']

plt.pie(data, colors=color_list)
plt.show()
```



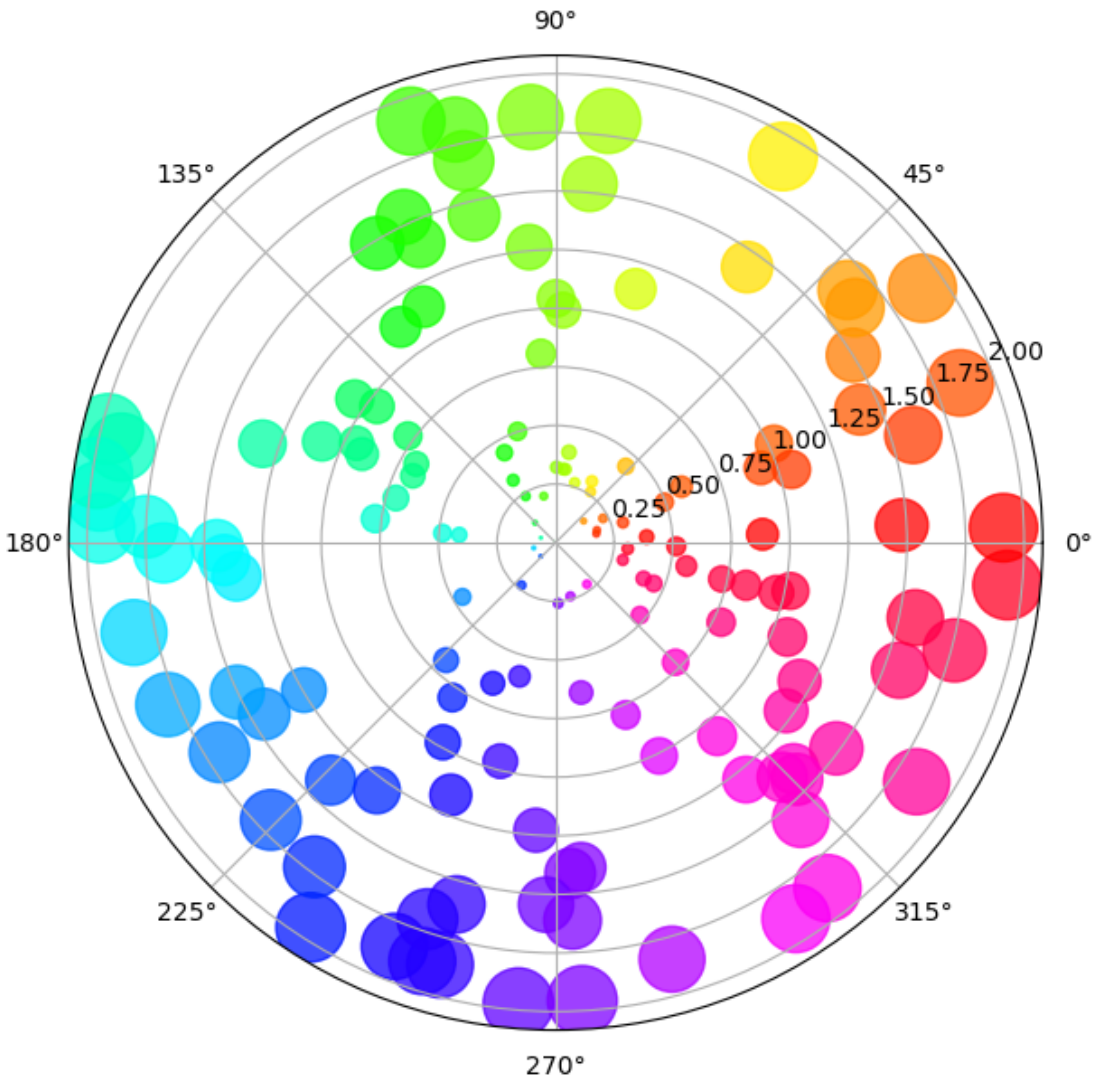
1.27 Pie chart(2)

[ ]:
```
labels = 'Seoul', 'Daejeon', 'Busan', 'Daegu'
pop_size = [50, 11, 25, 14]
explode = (0, 0.1, 0, 0)

plt.title("population size", size=25, color='black')
plt.pie(pop_size, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=45)

plt.show()
```

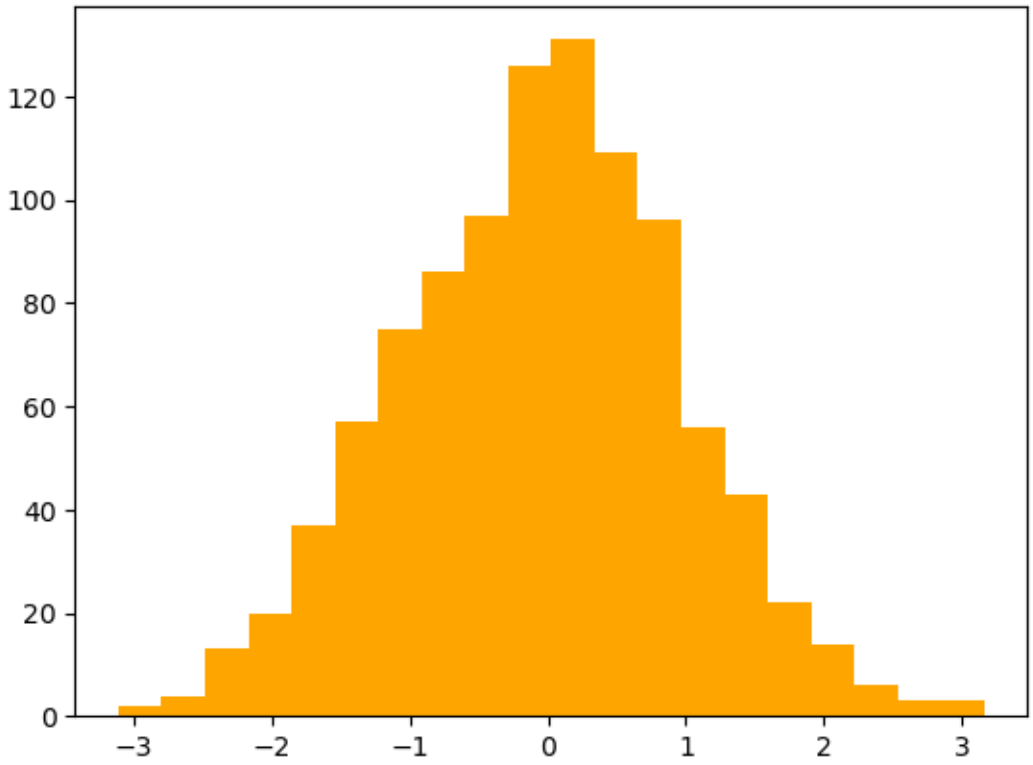


1.28 Pie chart(3)

[ ]:
```
np.random.seed(19680801)

N = 150
r = 2*np.random.rand(N)
theta = 2*np.pi*np.random.rand(N)
area = 200*r**2
colors = theta

fig = plt.figure(figsize=(7,7))
fig.add_subplot(projection='polar')
plt.scatter(theta, r, c=colors, s=area, cmap='hsv', alpha=0.75)
plt.show()
```

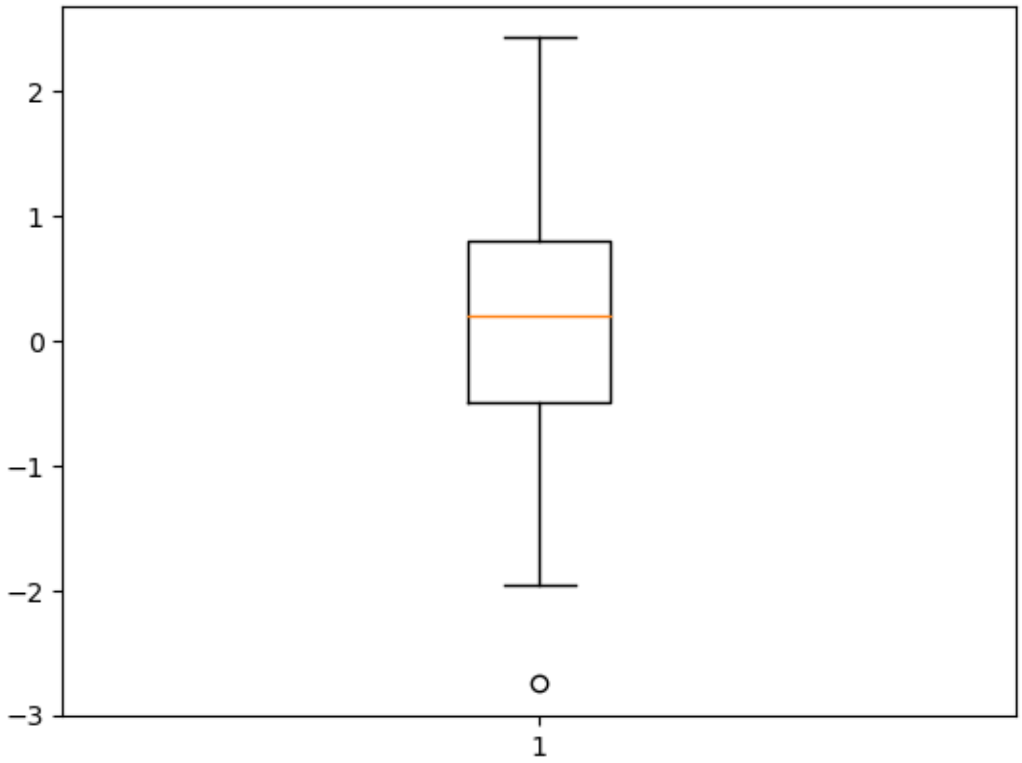


[ ]:
```
count = 1000
X = [random.gauss(0,1.) for i in range(count)]

plt.hist(X, bins=20, color='orange')
plt.show()
```



1.29 Box Plot(1)

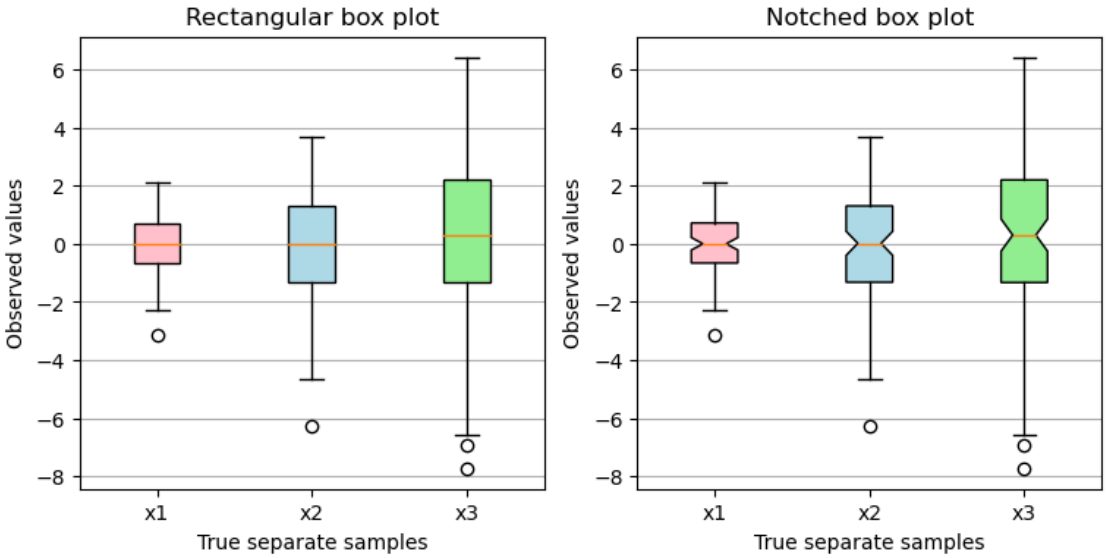
[ ]:
```
count = 100
data = [random.gauss(0., 1.) for i in range(count)]

plt.boxplot(data)
plt.show()
```



1.30 Box Plot(2)

[ ]:
```
# Random test data
np.random.seed(19680801)
data = [np.random.normal(0, std, size=100) for std in range(1,4)]
labels = ['x1', 'x2', 'x3']

fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(9,4))

# rectangular box plot
bplot1 = ax1.boxplot(data,
                     vert = True,
                     patch_artist=True,
                     labels=labels)
ax1.set_title('Rectangular box plot')

# notch shape box plot
bplot2 = ax2.boxplot(data,
                     notch=True,
                     vert=True,
                     patch_artist=True,
                     labels=labels)
```

```
ax2.set_title('Notched box plot')

# fill with colors
colors = ['pink', 'lightblue', 'lightgreen']
for bplot in (bplot1, bplot2):
    for patch, color in zip(bplot['boxes'], colors):
        patch.set_facecolor(color)

# adding horizontal grid lines
for ax in [ax1, ax2]:
    ax.yaxis.grid(True)
    ax.set_xlabel('True separate samples')
    ax.set_ylabel('Observed values')

plt.show()
```



1.31 Zip function

[ ]:
```
numbers = [1, 2, 3]
letters = ["A", "B", "C"]
for pair in zip(numbers, letters):
    print(pair)
```

```
(1, 'A')
(2, 'B')
(3, 'C')
```

[ ]:
```
numbers = [1, 2, 3]
letters = ["A","B","C"]
for i in range(3):
    pair = (numbers[i], letters[i])
    print(pair)
```

```
(1, 'A')
(2, 'B')
(3, 'C')
```

[ ]:
```
for number, upper, lower in zip("12345", "ABCDE", "abcde"):
    print(number, upper, lower)
```

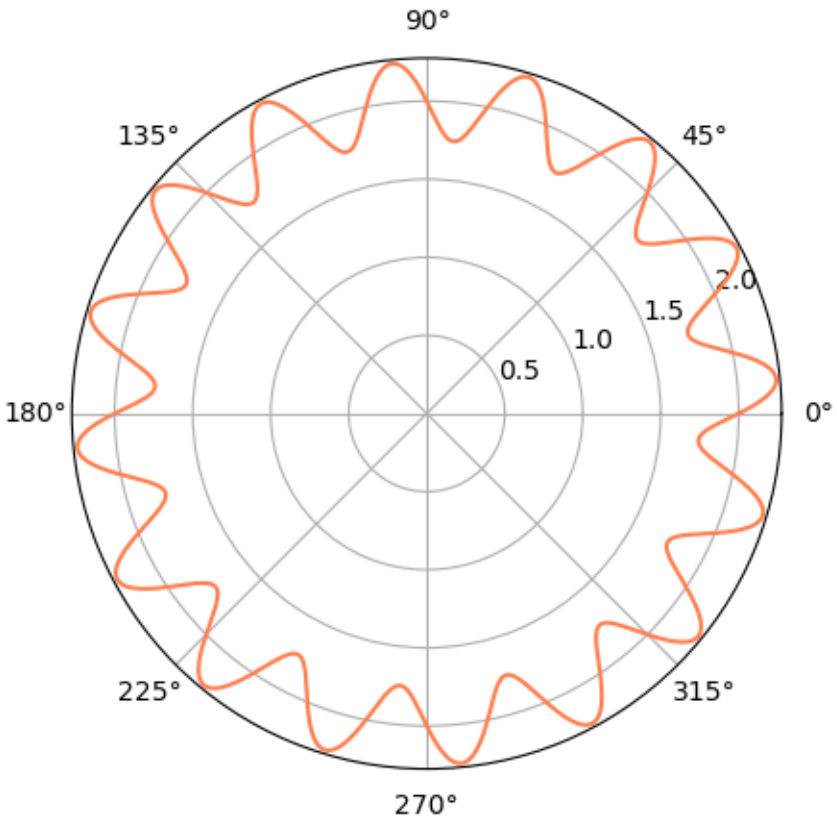
```
1 A a
2 B b
3 C c
4 D d
5 E e
```

[ ]:
```
for number, upper, lower in zip("12345", "ABC", "abcde"):
    print(number, upper, lower)
```

```
1 A a
2 B b
3 C c
```

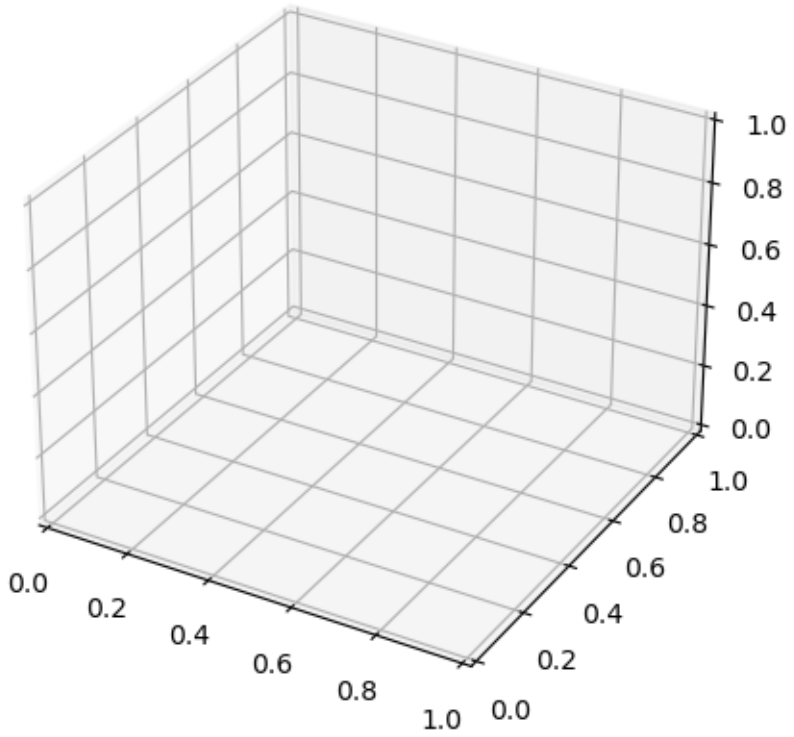
1.32 Polar plot

[ ]:
```
X = np.linspace(0, 2*np.pi, 1000)
Y = 2. + 0.25*np.sin(16*X)

plt.axes(polar = True)
plt.plot(X, Y, color = 'coral')
plt.show()
```



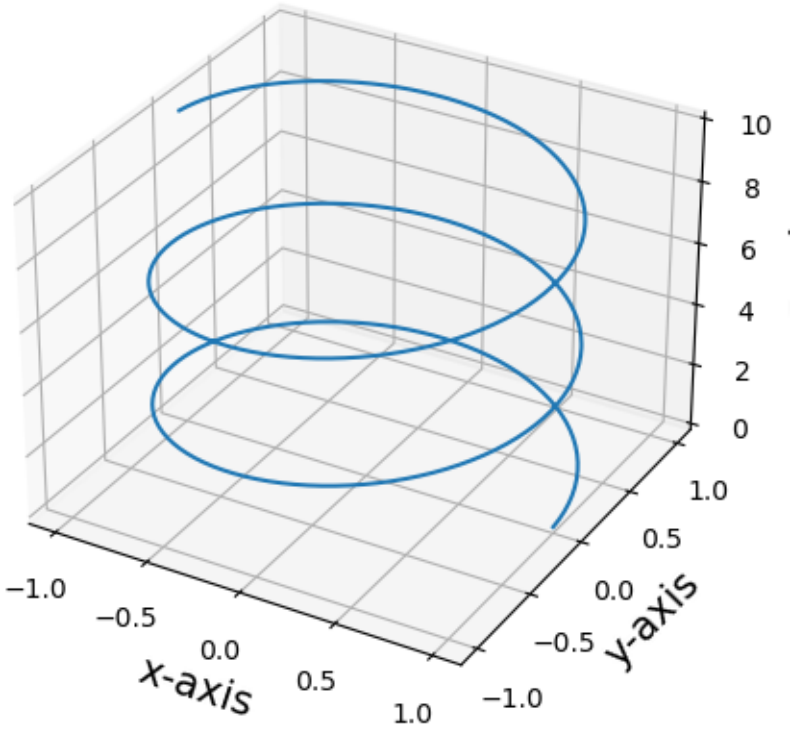
1.33 3D Graph(1)

[ ]:
```
fig = plt.figure(figsize=(10,5))
axis = fig.add_axes([0.1, 0.1, 0.8, 0.8], projection='3d')
plt.show()
```



1.34 3D Graph(2)

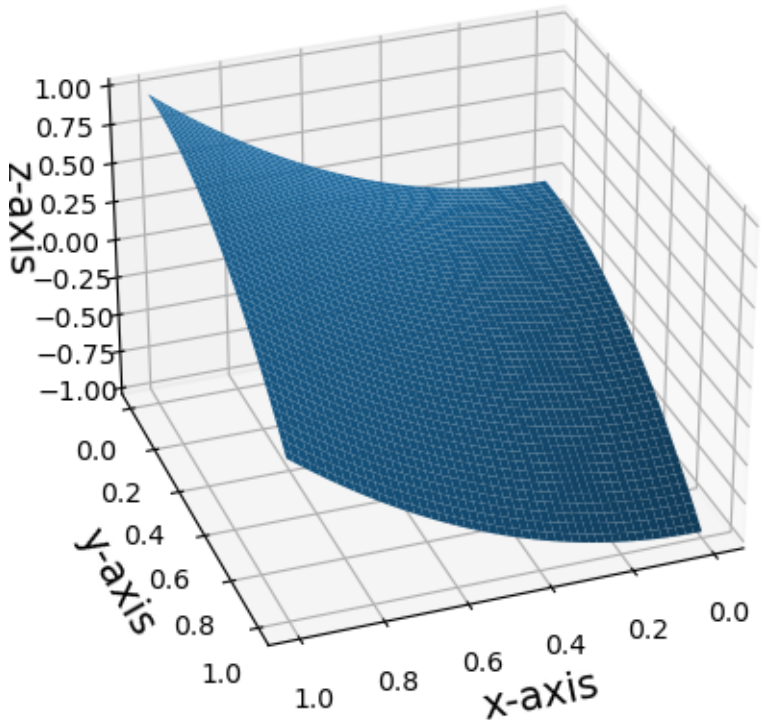
[ ]:
```
fig = plt.figure(figsize=(10,5))
axis = fig.add_axes([0.1, 0.1, 0.8, 0.8], projection='3d')

t = np.linspace(0., 5., 500)
x = np.cos(np.pi*t)
y = np.sin(np.pi*t)
z = 2*t

axis.plot(x, y, z)
axis.set_xlabel('x-axis', size=15)
axis.set_ylabel('y-axis', size=15)
axis.set_zlabel('z-axis', size=15)
axis.set_xticks([-1.0, -0.5, 0, 0.5, 1.0])
axis.set_yticks([-1.0, -0.5, 0, 0.5, 1.0])

plt.show()
```



1.35 3D Graph(3)

[ ]:
```
fig = plt.figure(figsize=(10,5))
axis = fig.add_axes([0.1, 0.1, 0.8, 0.8], projection='3d')

x = np.linspace(0.0, 1.0, 300)
y = np.linspace(0.0, 1.0, 300)

X, Y = np.meshgrid(x,y)
Z = X**2 - Y**2

axis.plot_surface(X, Y, Z)
axis.set_xlabel('x-axis', size=15)
axis.set_ylabel('y-axis', size=15)
axis.set_zlabel('z-axis', size=15)

axis.view_init(elev=30, azim=70)
axis.dist = 10

plt.show()
```

```
/var/folders/r1/8vnnkyjn3h3b_tnp2010w6nm0000gn/T/ipykernel_4281/3426452401.py:16
: MatplotlibDeprecationWarning: The dist attribute was deprecated in Matplotlib
3.6 and will be removed two minor releases later.
  axis.dist = 10
```

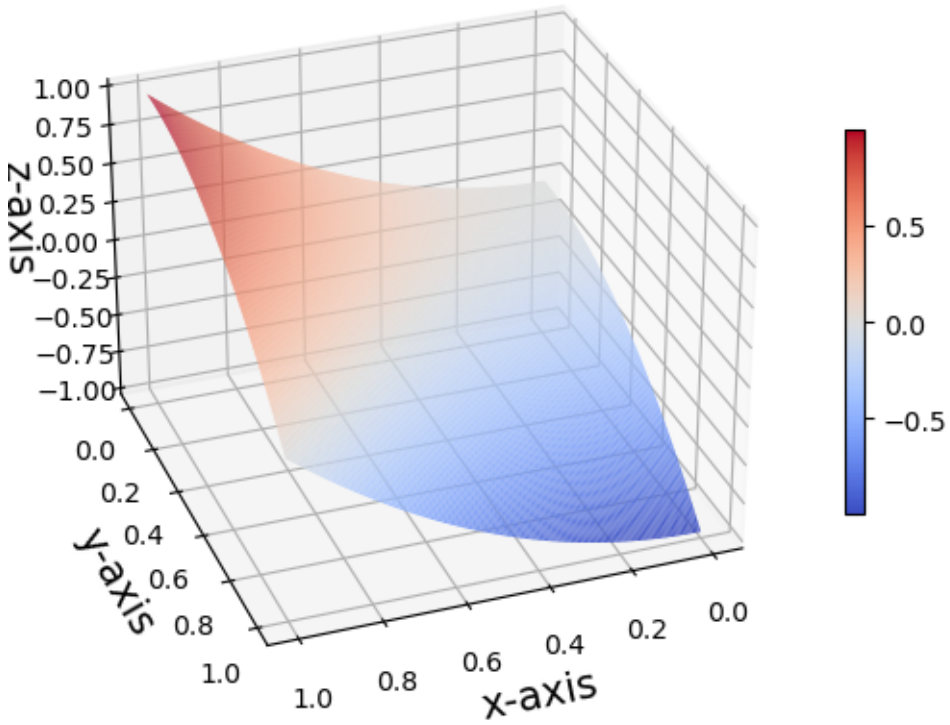


1.36 3D-Graph(3)

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

fig = plt.figure(figsize = (10, 5))
axis = fig.add_axes([0.1, 0.1, 0.5, 0.8], projection='3d')

x = np.linspace(0.0, 1.0, 300)
y = np.linspace(0.0, 1.0, 300)

X, Y = np.meshgrid(x, y)
Z = X**2 - Y**2

p = axis.plot_surface(X, Y, Z,
```

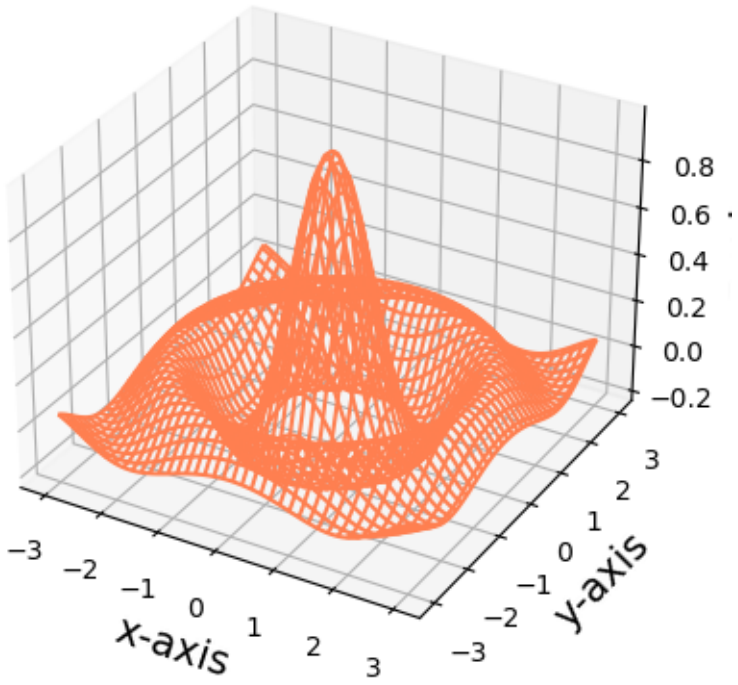
```
                      rstride=1, cstride=2, cmap=cm.coolwarm)
axis.set_xlabel('x-axis', size=15)
axis.set_ylabel('y-axis', size=15)
axis.set_zlabel('z-axis', size=15)

axis.view_init(elev=30, azim=70)
axis.dist = 10
fig.colorbar(p, shrink=0.5)
fig.show()
```

```
/var/folders/r1/8vnnkyjn3h3b_tnp2010w6nm0000gn/T/ipykernel_4281/2734696395.py:21
: MatplotlibDeprecationWarning: The dist attribute was deprecated in Matplotlib
3.6 and will be removed two minor releases later.
  axis.dist = 10
/var/folders/r1/8vnnkyjn3h3b_tnp2010w6nm0000gn/T/ipykernel_4281/2734696395.py:23
: UserWarning: Matplotlib is currently using
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```



1.37 3D-Graph(4)

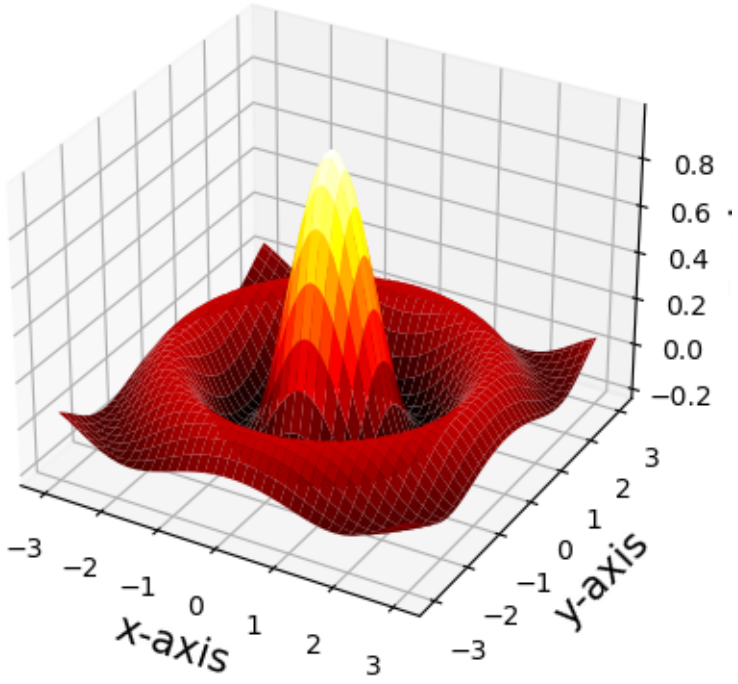
```
[ ]: x = np.linspace(-3, 3, 300)
     y = np.linspace(-3, 3, 300)
     X, Y = np.meshgrid(x,y)
     Z = np.sinc(np.sqrt(X**2 + Y**2))

     fig = plt.figure()
     ax = fig.add_subplot(projection = '3d') # gca -> add_subplot
     ax.plot_wireframe(X, Y, Z, cstride=8, rstride=8, color='coral')
     ax.set_xlabel('x-axis', size=15)
     ax.set_ylabel('y-axis', size=15)
     ax.set_zlabel('z-axis', size=15)

     plt.show()
```



1.38 3D-Graph

```
[ ]: x = np.linspace(-3, 3, 300)
     y = np.linspace(-3, 3, 300)
     X, Y = np.meshgrid(x,y)
     Z = np.sinc(np.sqrt(X**2 + Y**2))
```

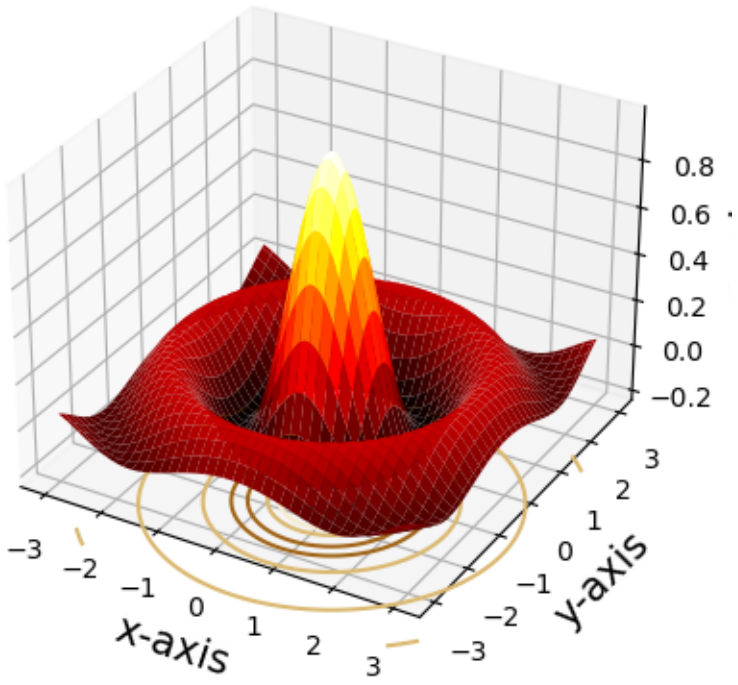
```
fig = plt.figure()
ax = fig.add_subplot(projection = '3d') # gca -> add_subplot
ax.plot_surface(X, Y, Z, cstride=8, rstride=8, cmap=cm.hot)
ax.set_xlabel('x-axis', size=15)
ax.set_ylabel('y-axis', size=15)
ax.set_zlabel('z-axis', size=15)

plt.show()
```



1.39 3D-Graph

[ ]:
```
x = np.linspace(-3, 3, 300)
y = np.linspace(-3, 3, 300)
X, Y = np.meshgrid(x,y)
Z = np.sinc(np.sqrt(X**2 + Y**2))

fig = plt.figure()
ax = fig.add_subplot(projection = '3d') # gca -> add_subplot
ax.plot_surface(X, Y, Z, cstride=8, rstride=8, cmap=cm.hot)
ax.contour(X, Y, Z, zdir='z', offset=-0.5, cmap = cm.BrBG)
ax.set_xlabel('x-axis', size=15)
ax.set_ylabel('y-axis', size=15)
```

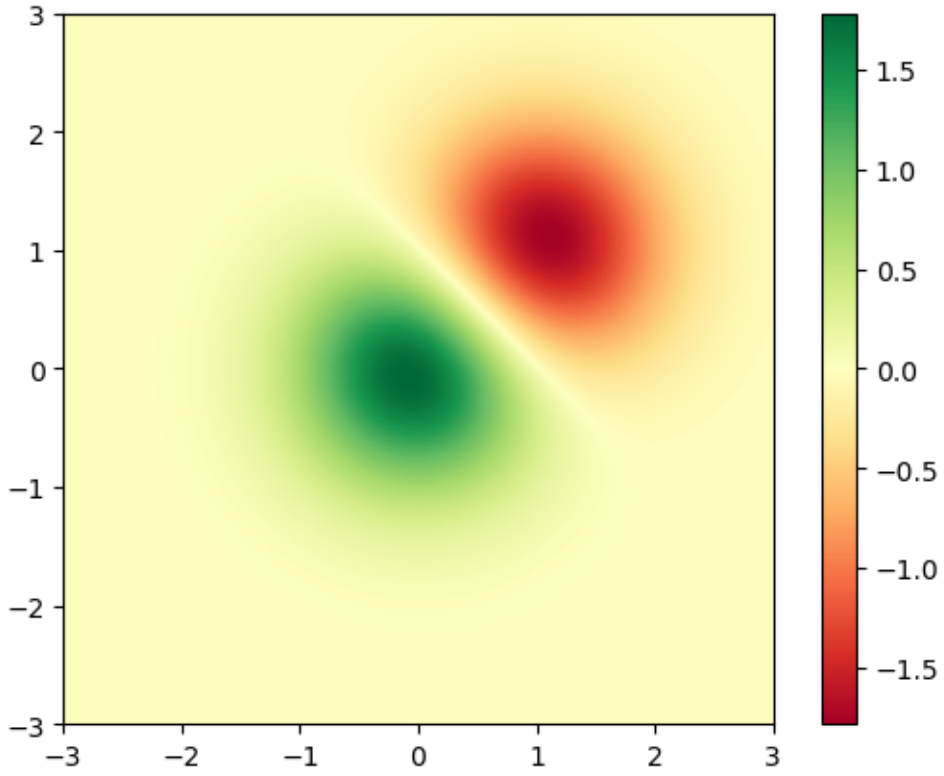
```
ax.set_zlabel('z-axis', size=15)

plt.show()
```



1.40 Density function graph

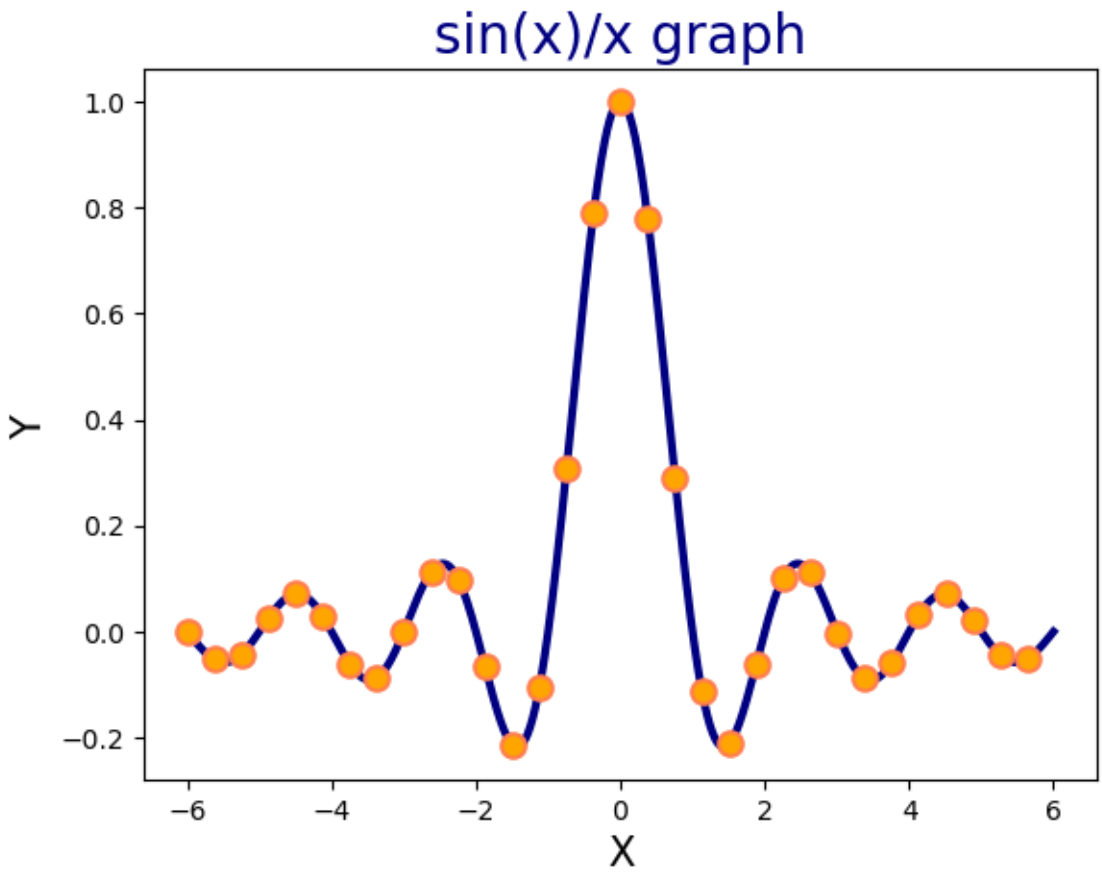
[ ]:
```
from matplotlib import colorbar
import numpy as np
import matplotlib.cm as cm
import matplotlib.pyplot as plt

# Fixing random state for reproducibility
np.random.seed(19680801)

delta = 0.025
x = y = np.arange(-3.0, 3.0, delta)
X, Y = np.meshgrid(x, y)
Z1 = np.exp(-X**2 - Y**2)
Z2 = np.exp(-(X-1)**2 - (Y-1)**2)
Z = (Z1 - Z2) * 2

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

```
im = ax.imshow(Z, interpolation = 'bilinear', cmap = cm.RdYlGn,
               origin='lower', extent=[-3, 3, -3, 3],
               vmax=abs(Z).max(), vmin=-abs(Z).max())

fig.colorbar(im)
plt.show()
```



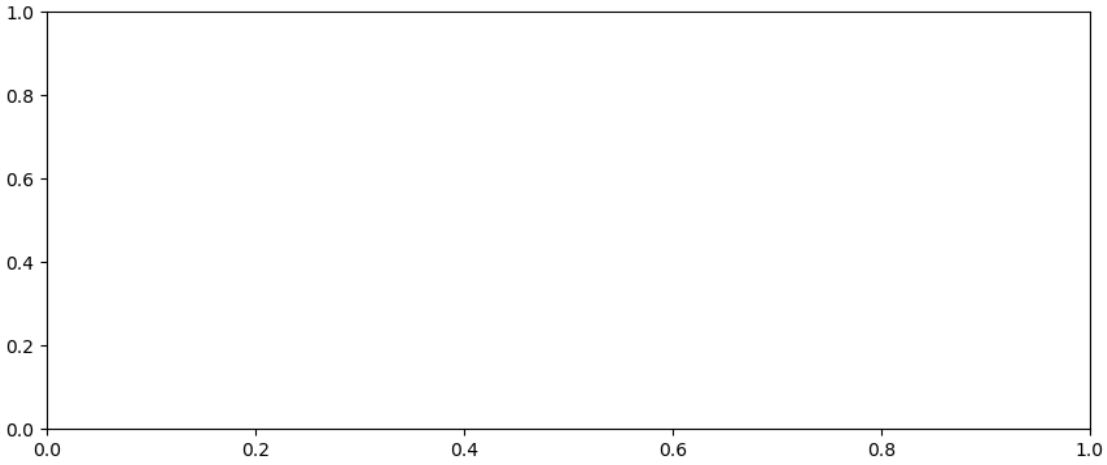
1.41 Plotting data value

[ ]:
```
X = np.linspace(-6, 6, 1024)
Y = np.sinc(X)

plt.plot(X, Y, linewidth = 3., color = 'navy',
         markersize = 9, markeredgewidth = 1.5,
         markerfacecolor = 'orange',
         markeredgecolor = 'coral',
         marker = 'o', markevery = 32)
plt.title('sin(x)/x graph', size=20, color='navy')
plt.xlabel('X', size = 15)
plt.ylabel('Y', size = 15)
plt.show()
```

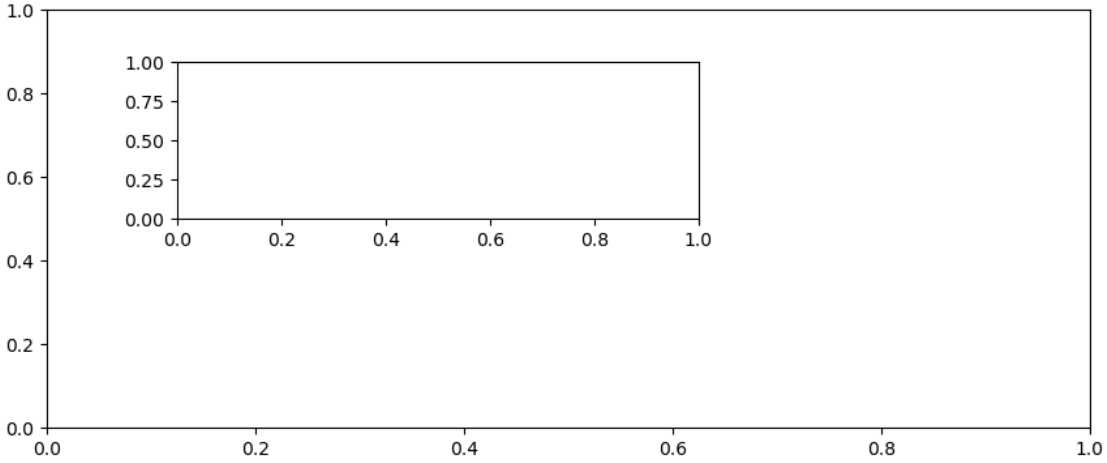


1.42 Adjust graph position and size(1)

[ ]:
```
fig = plt.figure(figsize=(10, 4))
axis = fig.add_axes([0.1, 0.1, 0.8, 0.8])

plt.show()
```

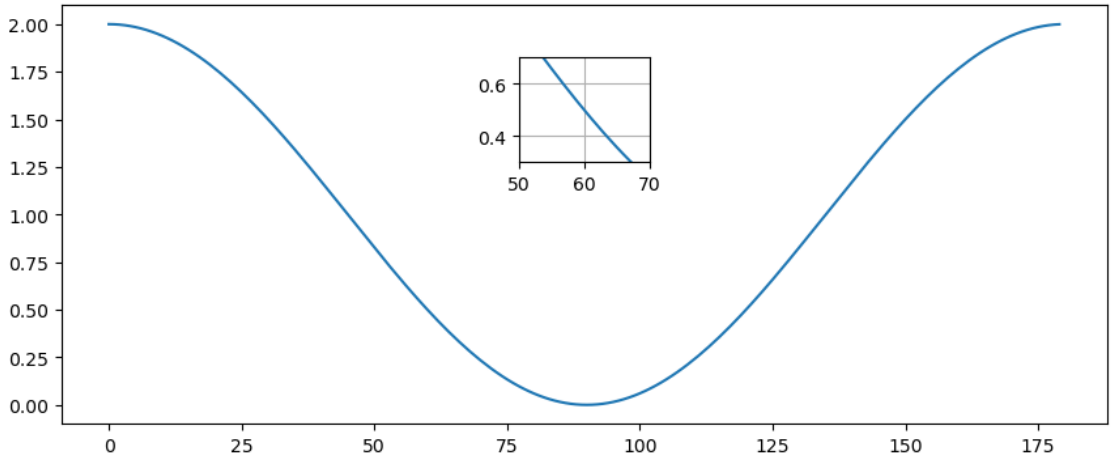


[ ]:
```
fig = plt.figure(figsize=(10, 4))
axis1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axis2 = fig.add_axes([0.2, 0.5, 0.4, 0.3])

plt.show()
```

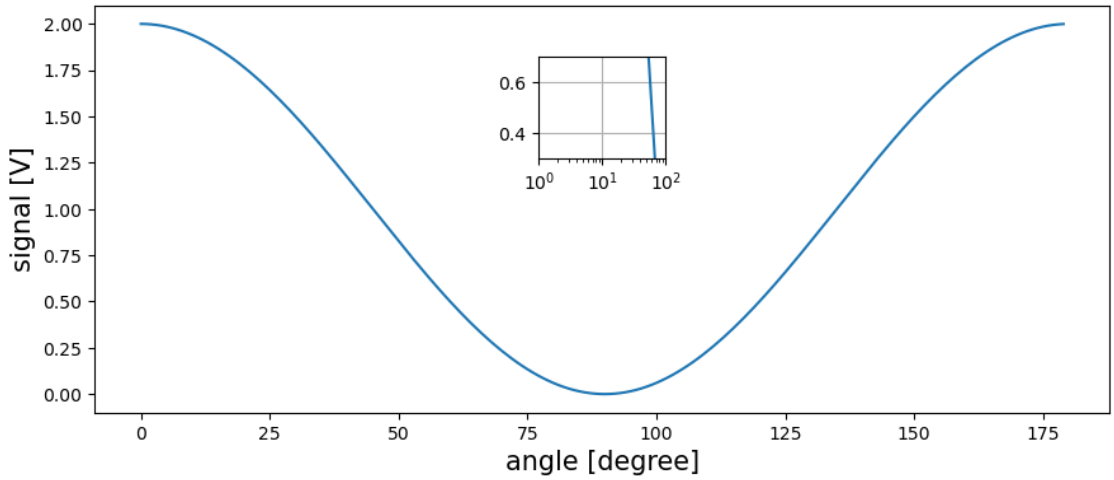


[ ]:
```
fig = plt.figure(figsize=(10, 4))
axis1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axis2 = fig.add_axes([0.45, 0.6, 0.1, 0.2])

x = np.arange(0, 180, 1)
y = np.cos(2*x*np.pi/180) + 1
axis1.plot(x, y)
axis2.plot(x, y)
```

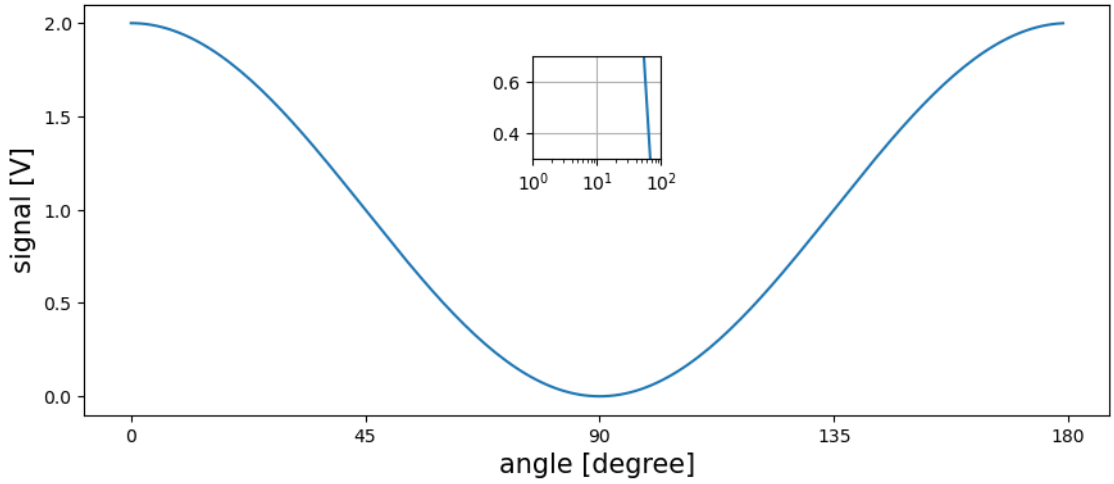
```
axis2.set_xbound(50, 70)
axis2.set_ybound(0.3, 0.7)
axis2.grid()

plt.show()
```



[ ]:
```
fig = plt.figure(figsize=(10, 4))
axis1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axis2 = fig.add_axes([0.45, 0.6, 0.1, 0.2])

x = np.arange(0, 180, 1)
y = np.cos(2*x*np.pi/180) + 1
axis1.plot(x, y)
axis2.plot(x, y)
axis1.set_xlabel('angle [degree]', fontsize=15)
axis1.set_ylabel('signal [V]', fontsize=15)
axis2.set_xscale('log')
axis2.set_xbound(1, 100)
axis2.set_ybound(0.3, 0.7)
axis2.grid()

plt.show()
```

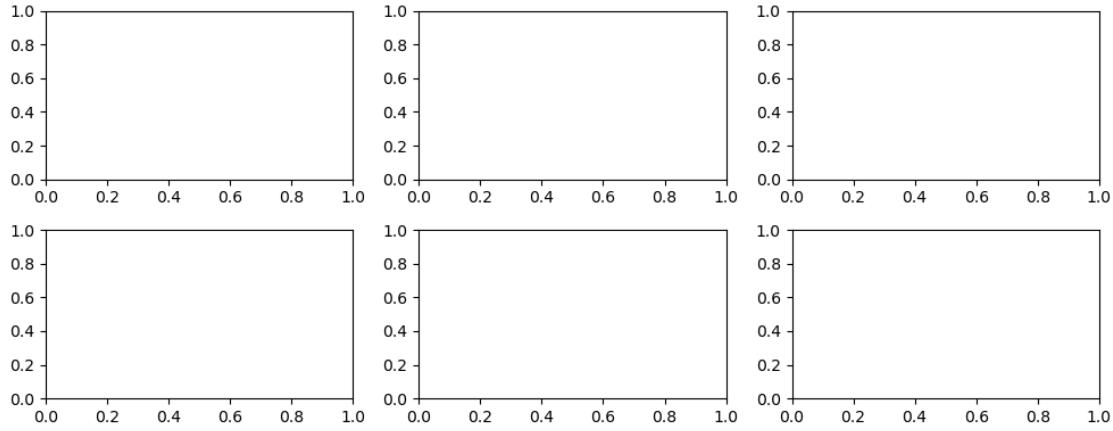


[ ]:
```
fig = plt.figure(figsize=(10, 4))
axis1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])
axis2 = fig.add_axes([0.45, 0.6, 0.1, 0.2])

x = np.arange(0, 180, 1)
y = np.cos(2*x*np.pi/180) + 1
axis1.plot(x, y)
axis2.plot(x, y)
axis1.set_xlabel('angle [degree]', fontsize=15)
axis1.set_ylabel('signal [V]', fontsize=15)
axis1.set_xticks([0, 45, 90, 135, 180])
axis1.set_xticklabels(['0','45','90','135','180'])
axis1.set_yticks([0.0, 0.5, 1.0, 1.5, 2])
axis1.set_yticklabels(['0.0', '0.5', '1.0', '1.5', '2.0'])
axis2.set_xscale('log')
axis2.set_xbound(1, 100)
axis2.set_ybound(0.3, 0.7)
axis2.grid()

plt.show()
```



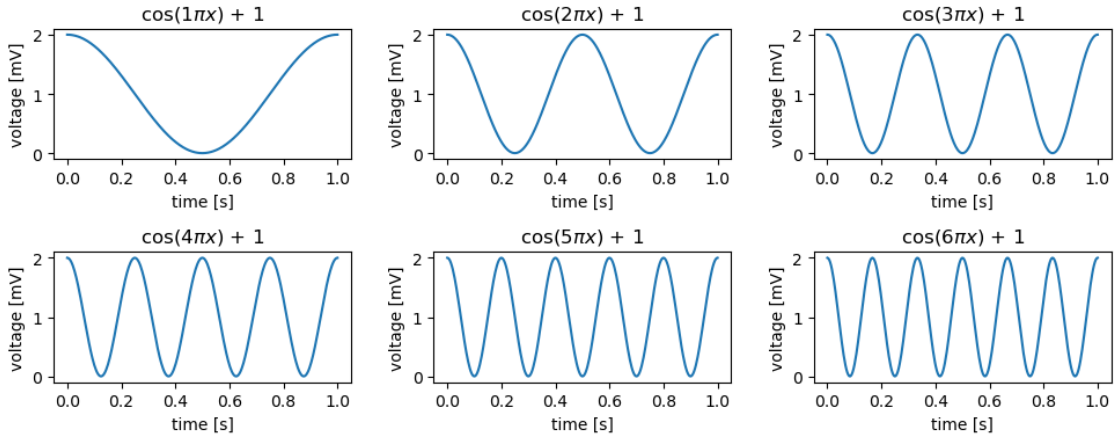
1.43 Adjust graph position and size(2)

[ ]:
```
fig = plt.figure(figsize=(10, 4))
for subplot in range(1,7):
    axis = fig.add_subplot(2, 3, subplot)
fig.tight_layout(pad=1.08)
plt.show()
```



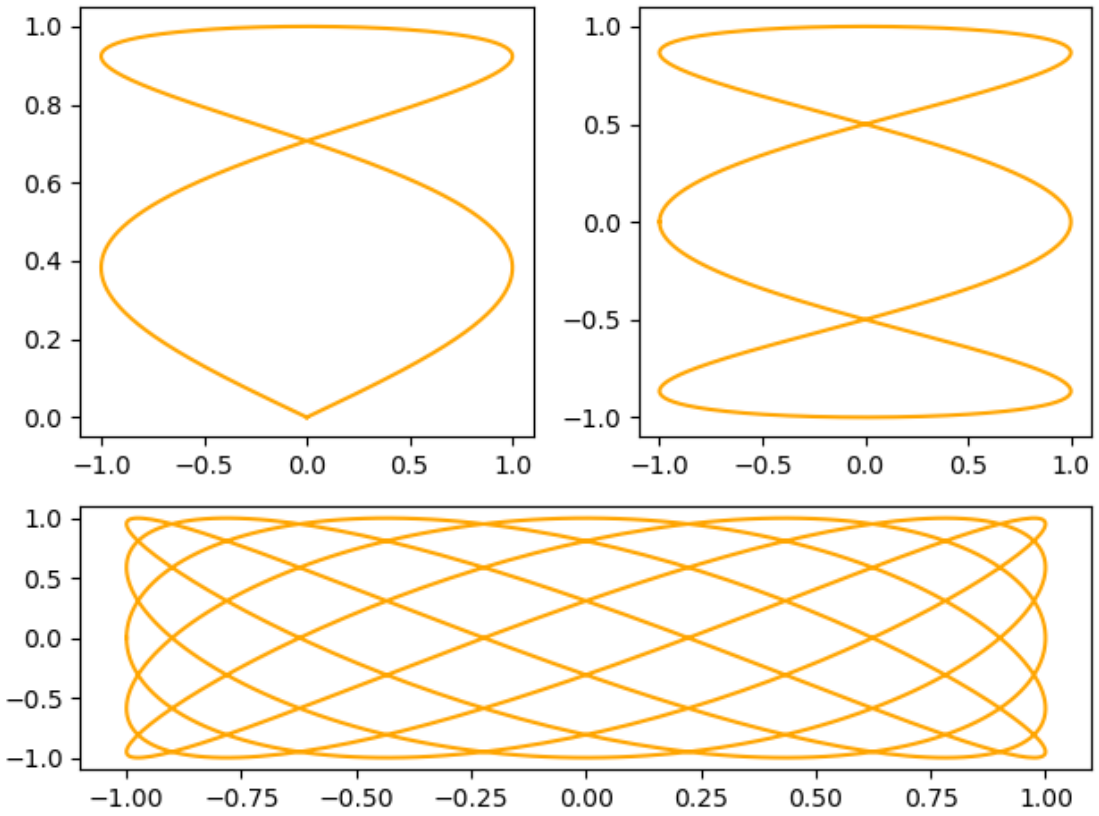
[ ]:
```
fig = plt.figure(figsize=(10,4))
x = np.linspace(0.0, 1.0, 300)
for subplot in range(1, 7):
    axis = fig.add_subplot(2, 3, subplot)
    y = np.cos(2*np.pi*x*subplot)  + 1
    axis.plot(x, y)
```

```
    axis.set_xlabel('time [s]')
    axis.set_ylabel('voltage [mV]')
    axis.set_title('$\cos({0} \pi x)$ + 1'.format(subplot))
fig.tight_layout(pad=1.08)
plt.show()
```



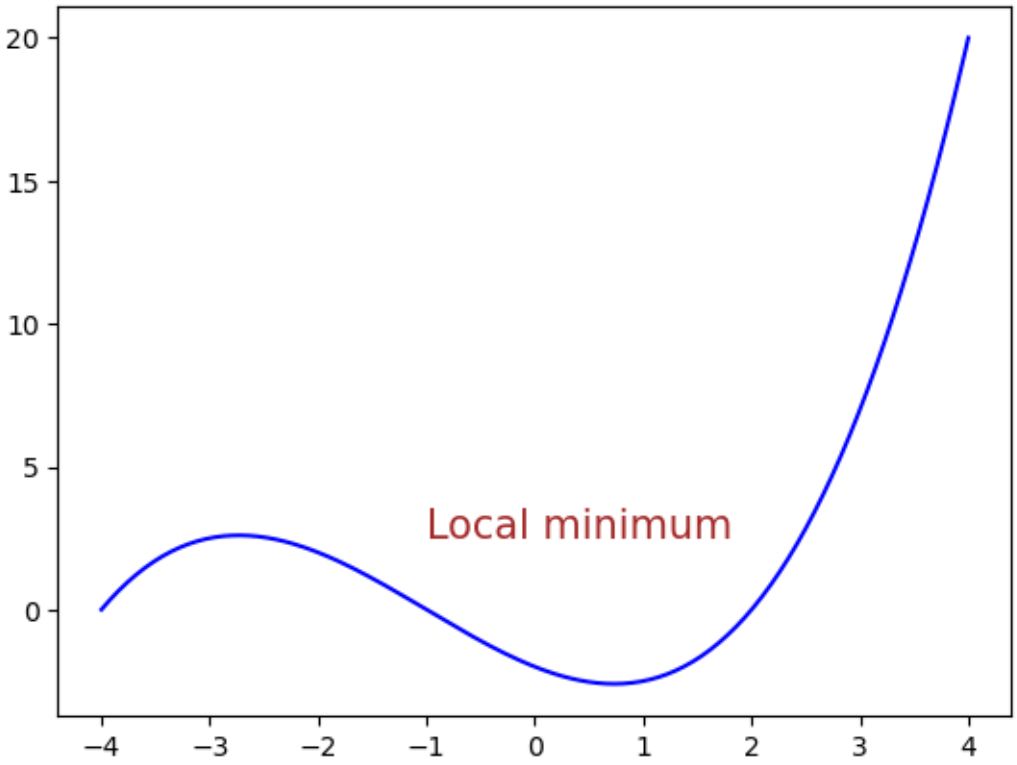
1.44 Combination of mutilayers(1)

[ ]:
```
X = np.linspace(-np.pi, np.pi, 1000)

# Shape of grid in which to place axis.
# First entry is number of rows, second entry is number of columns.
grid_size = (5, 2)

plt.subplot2grid(grid_size, (0, 0), rowspan=3, colspan=1)
plt.plot(np.sin(2*X), np.cos(0.5*X), color='orange')

plt.subplot2grid(grid_size, (0, 1), rowspan=3, colspan=1)
plt.plot(np.cos(3*X), np.sin(X), color='orange')

plt.subplot2grid(grid_size, (3, 0), rowspan=2, colspan=2)
plt.plot(np.cos(5*X), np.sin(7*X), color='orange')

plt.tight_layout(pad=1.08)
plt.show()
```



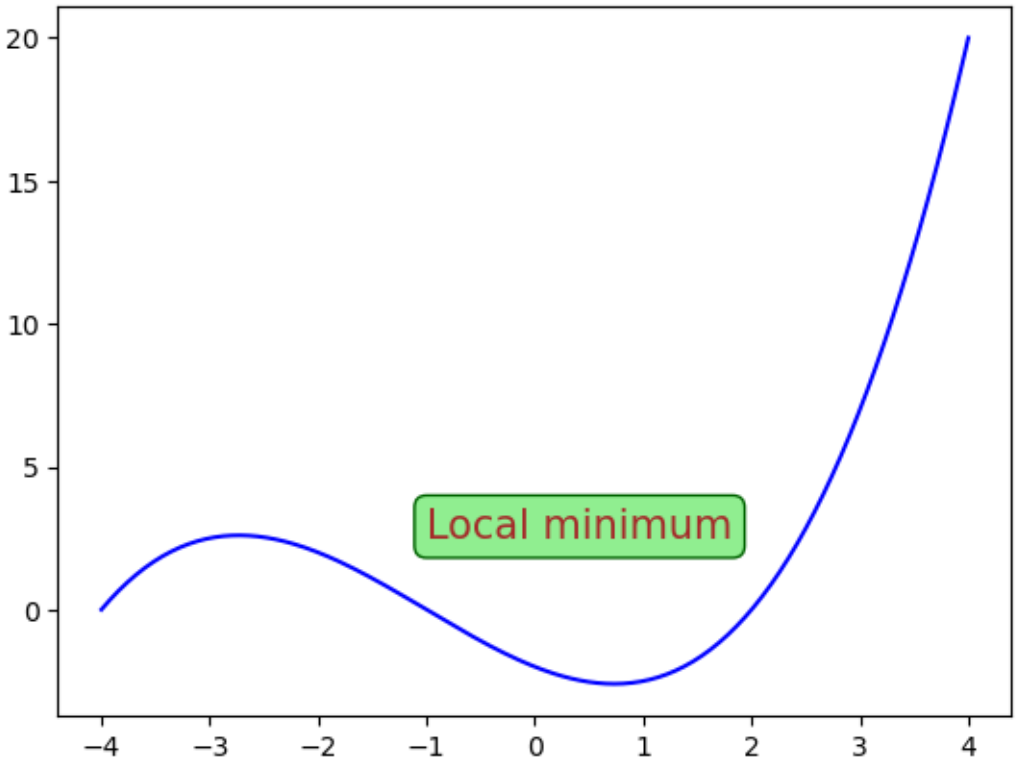
1.45 Adding text(1)

[ ]:
```
X = np.linspace(-4, 4, 300)
Y = 0.25*(X+4.)*(X+1.)*(X-2.)

plt.text(-1.0, 2.5, 'Local minimum', color='brown', size=15)
plt.plot(X, Y, color='blue')
plt.show()
```



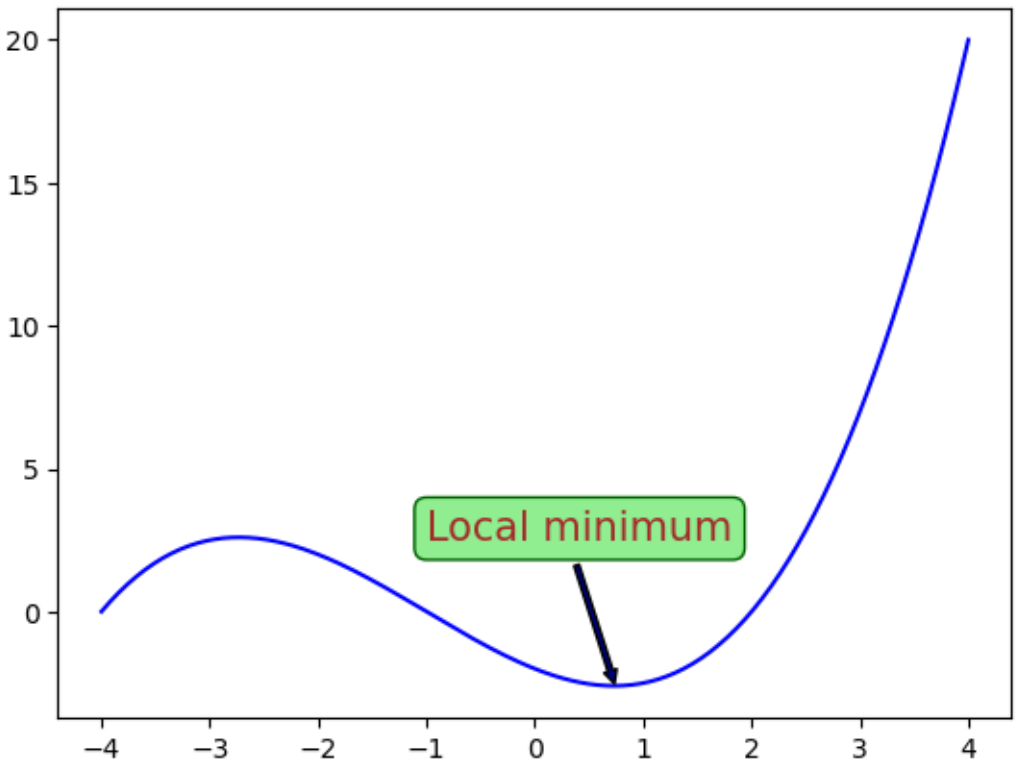
1.46 Adding text(2)

[ ]:
```
X = np.linspace(-4, 4, 300)
Y = 0.25*(X+4.)*(X+1.)*(X-2.)

box = {'facecolor':'lightgreen',
       'edgecolor':'darkgreen',
       'boxstyle':'round'}

plt.text(-1.0, 2.5, 'Local minimum', color='brown', size=15, bbox=box)
plt.plot(X, Y, color='blue')
plt.show()
```



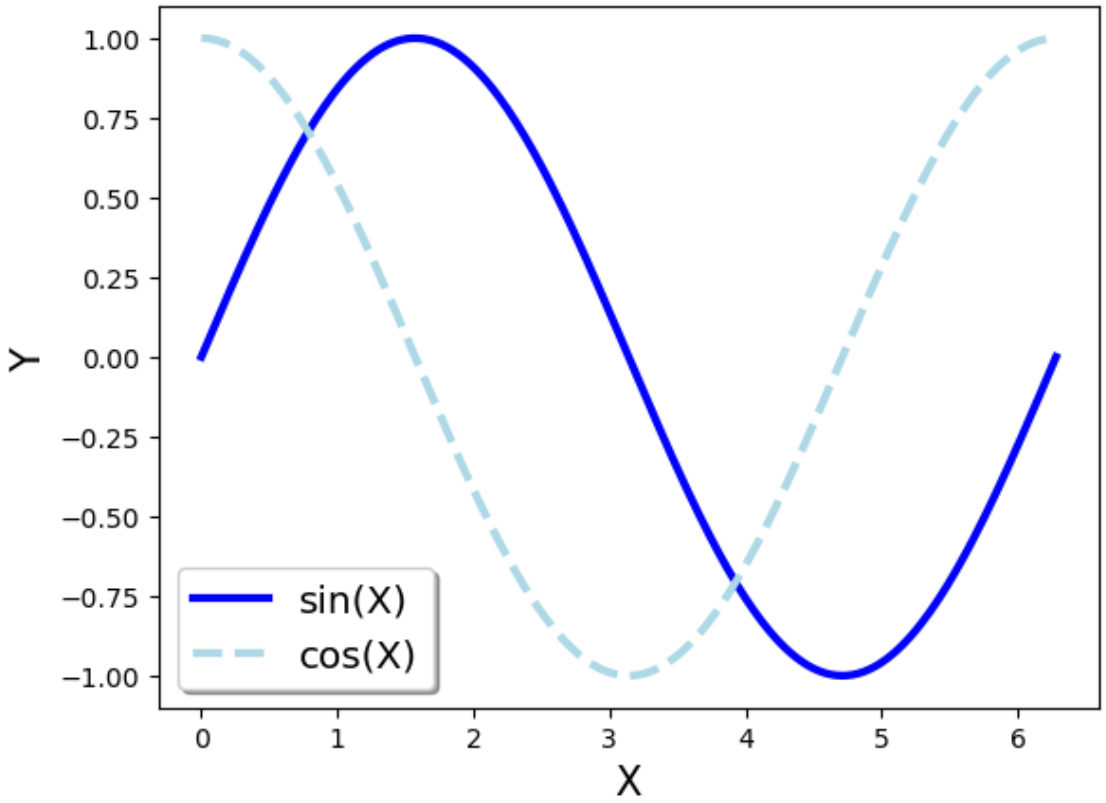
1.47 Adding text(3)

[ ]:
```
X = np.linspace(-4, 4, 300)
Y = 0.25*(X+4.)*(X+1.)*(X-2.)

box = {'facecolor':'lightgreen',
       'edgecolor':'darkgreen',
       'boxstyle':'round'}
arrow = {'facecolor':'navy',
         'arrowstyle':'simple'}
plt.text(-1.0, 2.5, 'Local minimum', color='brown', size=15, bbox=box)
plt.annotate("", xytext=(0.37, 1.8),
             xy = (0.75, -2.7),
             arrowprops=arrow)
plt.plot(X, Y, color='blue')
plt.show()
```



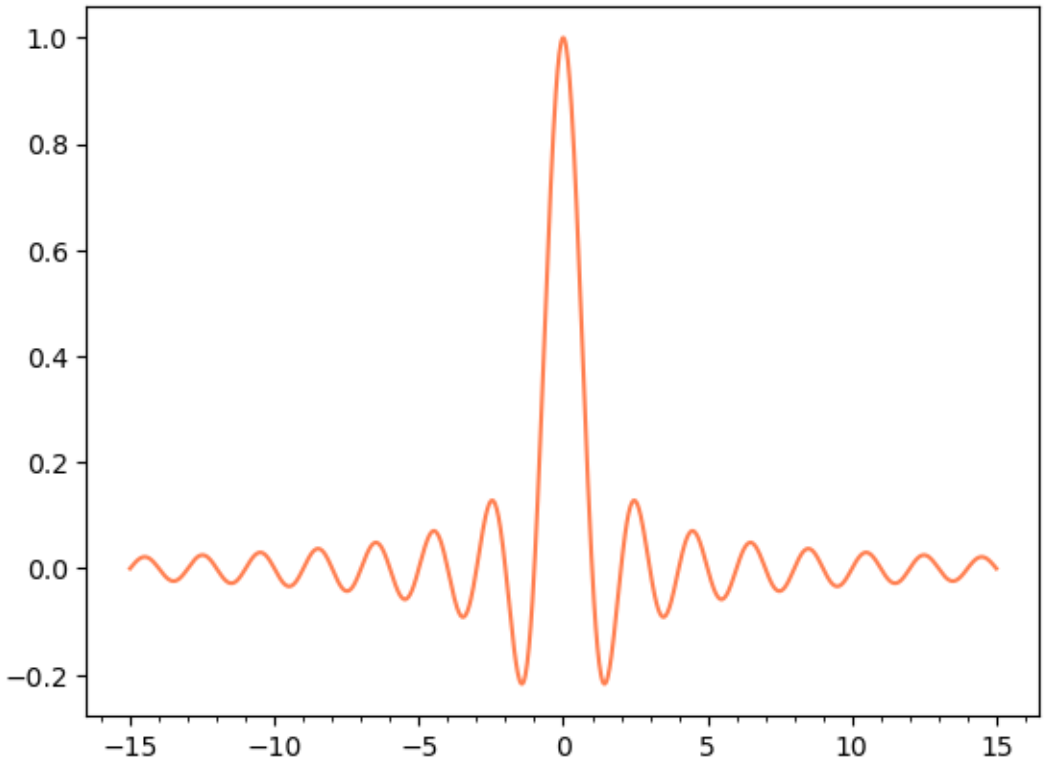
1.48 Adding legend

[ ]:
```
X = np.linspace(0, 2*np.pi, 300)
Y1 = np.sin(X)
Y2 = np.cos(X)

plt.xlabel('X', size=15)
plt.ylabel('Y', size=15)
plt.plot(X, Y1, color='blue',lw=3.,
         label='sin(X)')
plt.plot(X, Y2, color='lightblue', ls='--',
         lw = 3, label = 'cos(X)')
plt.legend(loc='lower left', shadow = True, fontsize='x-large')
plt.show()
```



1.49 Scale interval adjustment

[ ]:
```
import matplotlib.ticker as ticker

X = np.linspace(-15, 15, 1000)
Y = np.sinc(X)

ax = plt.axes()
ax.xaxis.set_major_locator(ticker.MultipleLocator(5))
ax.xaxis.set_minor_locator(ticker.MultipleLocator(1))
ax.tick_params(axis='both', direction='out')

plt.plot(X, Y, color='coral')
plt.show()
```



Reference * Title: Physics Programming Lecture Note (INU) * Author: Jeongwoo Kim, Ph.D. * Availability: https://sites.google.com/view/jeongwookim

---