CLASSIFICATION EVALUATION

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
#purchase = ['No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes']
#purchase_pred = ['No', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'Yes', 'Yes', 'Yes']
purchase = [0,1,0,0,1,1,0,1,0,1]
purchase_pred=[0,0,0,1,1,0,0,1,1,1]
```

In [3]:
```
from sklearn.metrics import accuracy_score
```

In [4]:
```
accuracy_score(purchase,purchase_pred)
```

Out[4]:
```
0.6
```

In [5]:
```
from sklearn.metrics import precision_score
```

In [6]:
```
precision_score(purchase, purchase_pred,average='binary')
```

Out[6]:
```
0.6
```

In [7]:
```
from sklearn.metrics import recall_score
```

In [8]:
```
recall_score(purchase, purchase_pred,average='binary')
```

Out[8]:
```
0.6
```

In [9]:
```
from sklearn.metrics import f1_score
```

In [10]:
```
f1_score(purchase, purchase_pred,average='binary')
```

Out[10]:
```
0.6
```

In [11]:
```
from sklearn.metrics import confusion_matrix
```

In [12]:
```
confusion_matrix(purchase, purchase_pred)
```

Out[12]:
```
array([[3, 2],
       [2, 3]], dtype=int64)
```

In [13]:
```
tn, fp, fn, tp = confusion_matrix(purchase, purchase_pred).ravel()
```

In [14]:
```
tn, fp, fn, tp
```

Out[14]:
```
(3, 2, 2, 3)
```

REGRESSION EVALUATION

In [15]:
```
salary = [48000, 45000, 54000, 65000, 35000, 58000, 53000, 79000, 88000, 77000]
salary_pred = [48000, 45800, 53000, 65000, 35000, 57990, 53000, 79000, 87500, 77000]
```

In [16]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

In [17]:
```
mean_absolute_error(salary, salary_pred)
```

Out[17]:
```
231.0
```

In [18]:
```
n = 10
sum = 0
for i in range(n):
    sum+=abs(salary[i]-salary_pred[i])

MAE=sum/n
MAE
```

Out[18]:
```
231.0
```

In [19]:
```
mean_squared_error(salary, salary_pred)
```

Out[19]:
```
189010.0
```

In [20]:
```
n = 10
sum = 0
for i in range(n):
    sum=sum+pow((salary[i]-salary_pred[i]),2)

MSE=sum/n
MSE
```

Out[20]:
```
189010.0
```

In [21]:
```
r2_score(salary, salary_pred)
```

Out[21]:
```
0.9992563345923827
```

In [22]:
```
salary_mean=0
for i in range(n):
    salary_mean=salary_mean+salary[i]
salary_mean
```

Out[22]:
```
602000
```

In [23]:
```
sum1=0
sum2=0
for i in range(n):
    sum1=sum1+pow((salary[i]-salary_pred[i]),2)
    sum2=sum2+pow((salary_mean-salary[i]),2)

R2S=1-(sum1/sum2)
R2S
```

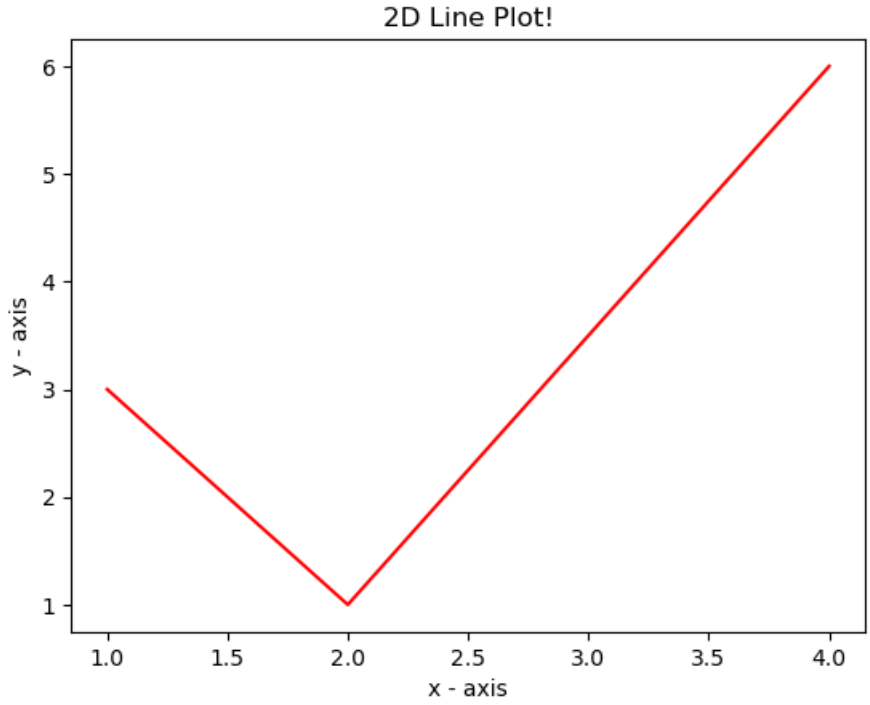
Out[23]:
```
0.9999993566742704
```

In [24]:
```
#Root Mean Square Error
RMSE=np.sqrt(MSE)
RMSE
```

Out[24]:
```
434.752803326212
```

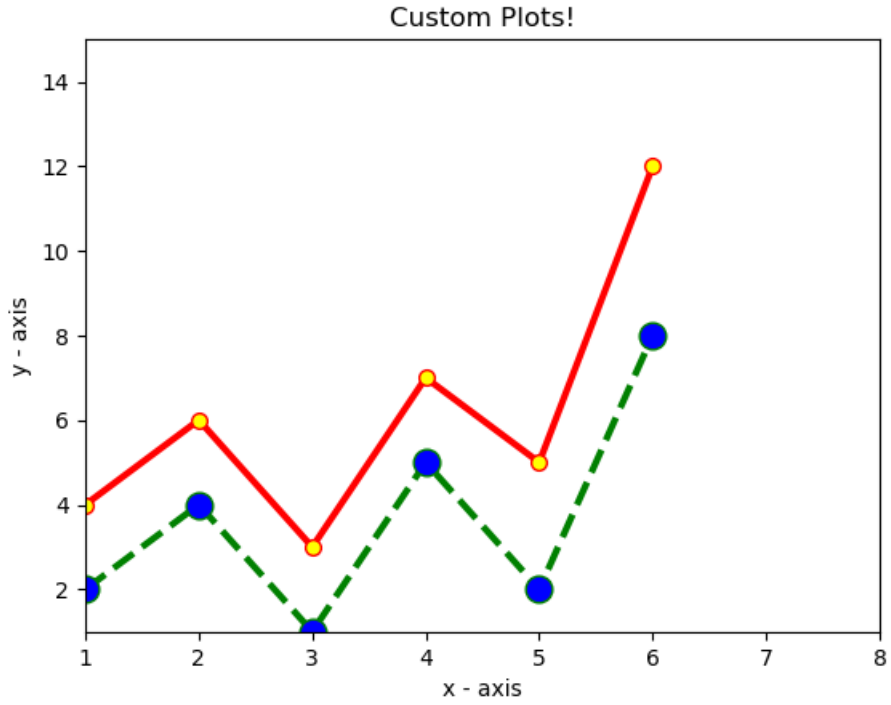
Plotting 2D Graphs

In [26]:
```
#Line 2D Plot
import matplotlib.pyplot as plt

# x axis values
x = [1,2,4]
# corresponding y axis values
y = [3,1,6]

# plotting the points
plt.plot(x, y,color='red')

# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')

# giving a title to my graph
plt.title('2D Line Plot!')

# function to show the plot
plt.show()
```

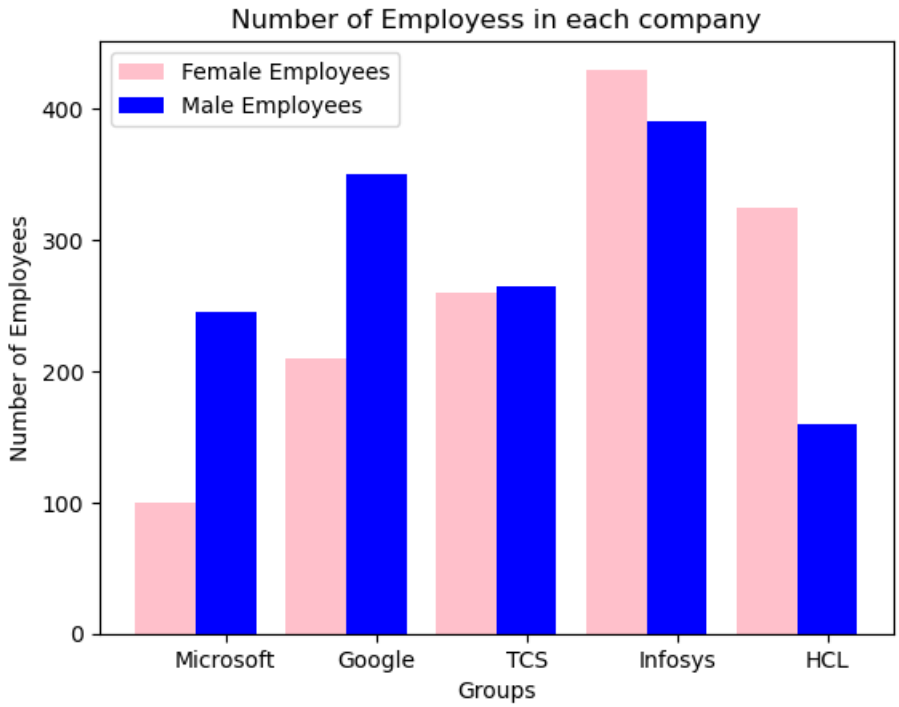


In [27]:

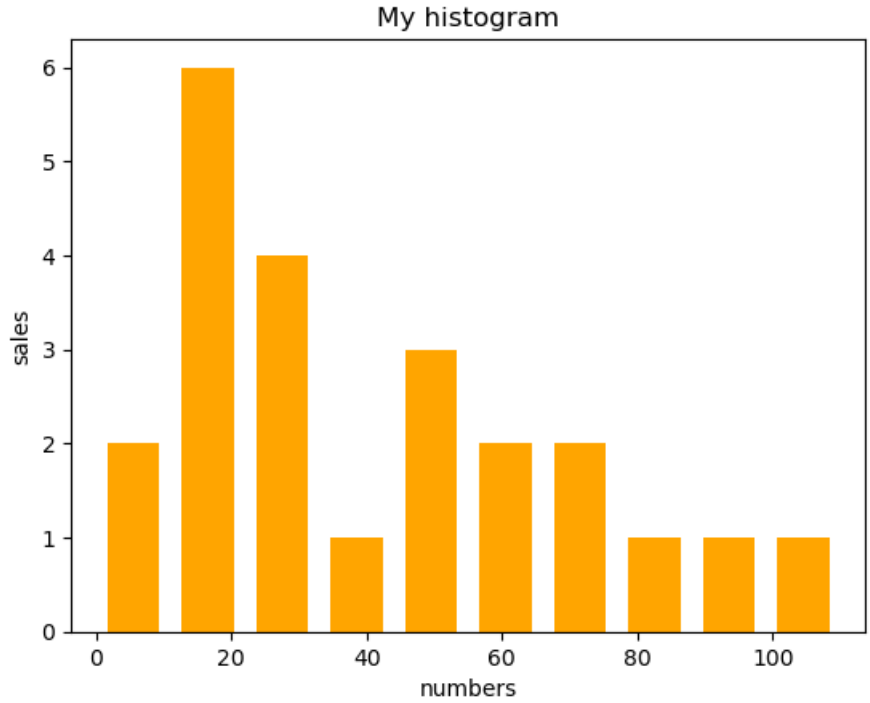
```
#Custom 2D Plot
import matplotlib.pyplot as plt

# x axis values
x = [1,2,3,4,5,6]
# corresponding y axis values
y = [2,4,1,5,2,8]

# plotting the points
plt.plot(x, y, color='green', linestyle='dashed', linewidth = 3,
        marker='o', markerfacecolor='blue', markersize=12)

# setting x and y axis range
plt.ylim(1,8)
plt.xlim(1,8)

# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')

# giving a title to my graph
plt.title('Custom Plots!')

# function to show the plot
#plt.show()

########################################
x = [1,2,3,4,5,6]
# corresponding y axis values
y = [4,6,3,7,5,12]

# plotting the points
plt.plot(x, y, color='red', linestyle='solid', linewidth = 3,
        marker='o', markerfacecolor='yellow', markersize=7)

# setting x and y axis range
plt.ylim(1,15)
plt.xlim(1,8)

# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')

# giving a title to my graph
plt.title('Custom Plots!')

# function to show the plot
plt.show()
```



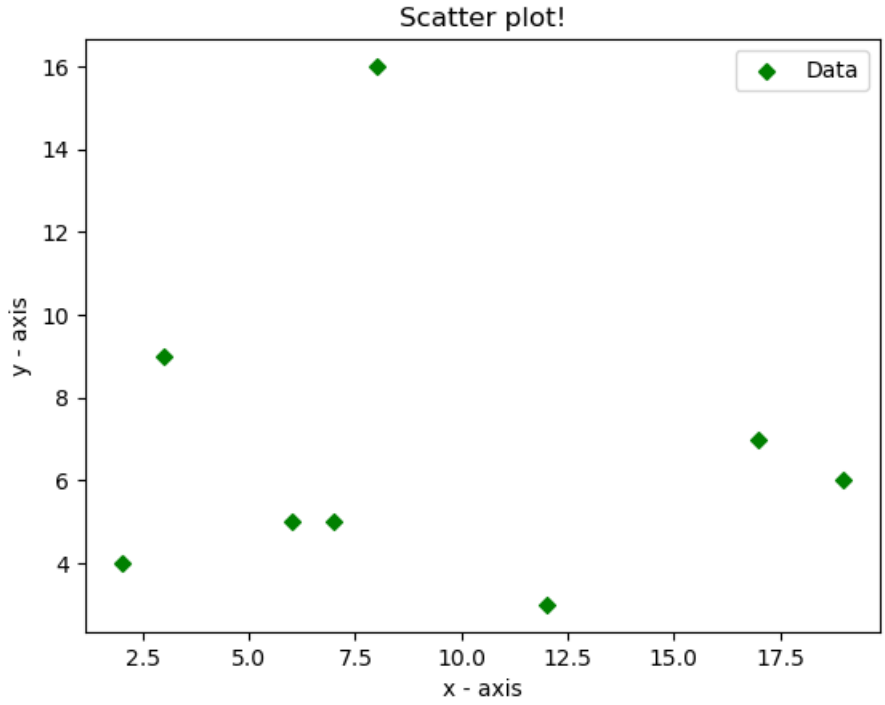
In [15]:
```
# Bar 2D Chart
import numpy as np
import matplotlib.pyplot as plt

companies = ['Microsoft','Google','TCS','Infosys','HCL']
Females= [100,210,260,430,325]
Males = [245,350,265,390,160]

X_axis = np.arange(len(companies))
'''generates an array of indices corresponding to the number of
elements in list companies, which will be used to position the bars on the x-axis.
 '''
plt.bar(X_axis-0.4, Females, 0.4, label = 'Female Employees', color='pink')
plt.bar(X_axis, Males, 0.4, label = 'Male Employees',color='blue')

plt.xticks(X_axis, companies)
plt.xlabel("Groups")
plt.ylabel("Number of Employees")
plt.title("Number of Employees in each company")
plt.legend()
plt.show()
```



In [16]:
```
#Histogram 2D Plot
import matplotlib.pyplot as plt

# frequencies
numbers = [22,13,60,45,20,8,30,75,43,110,24,
        60,7,18,47,18,90,77,32,21,20,67,50]

# setting the ranges and no. of intervals
range = (0, 110)
bins = 10

# plotting a histogram
plt.hist(numbers, bins, range, color ='orange',
        histtype = 'barstacked', rwidth = 0.7)

# x-axis label
plt.xlabel('numbers')
# frequency label
plt.ylabel('sales')
# plot title
plt.title('My histogram')

# function to show the plot
plt.show()
```

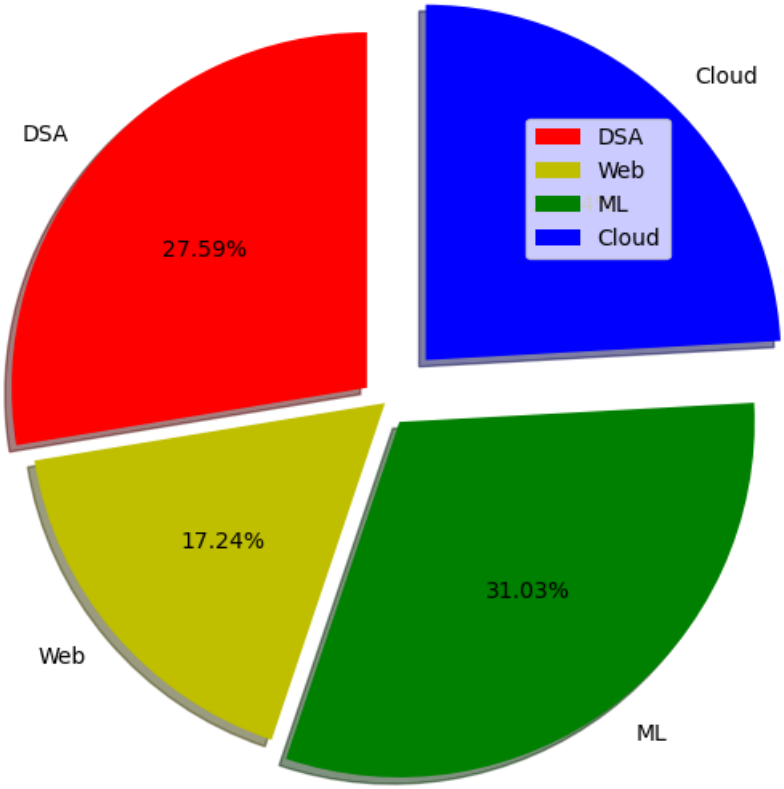


In [31]:
```
#Scatter 2D Plot
import matplotlib.pyplot as plt

# x-axis values
x = [2,7,12,3,6,19,17,8]
# y-axis values
y = [4,5,3,9,5,6,7,16]

# plotting points as a scatter plot
plt.scatter(x, y, label= "Data", color= "green",
            marker= "D", s=25)

# x-axis label
plt.xlabel('x - axis')
# frequency label
plt.ylabel('y - axis')
# plot title
plt.title('Scatter plot!')
# showing legend
plt.legend()

# function to show the plot
plt.show()
```



In [43]:

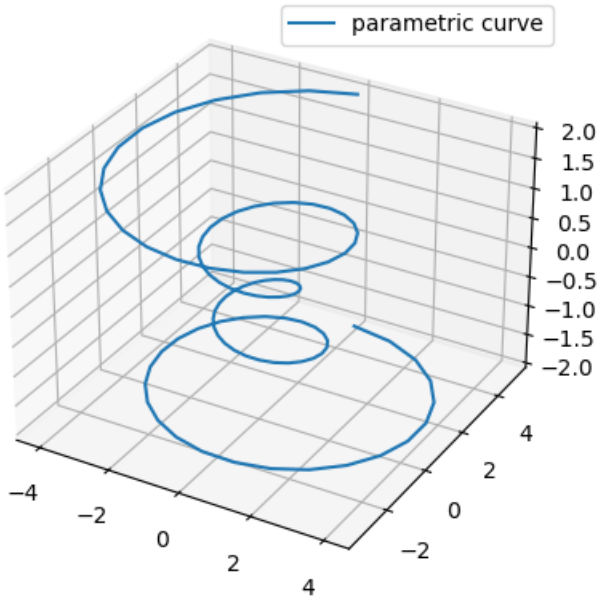
```
#Pie-Chart 2D Plot
import matplotlib.pyplot as plt

# defining labels
activities = ['DSA', 'Web', 'ML', 'Cloud']

# portion covered by each label
slices = [8, 5, 9, 7]

# color for each label
colors = ['r', 'y', 'g', 'b']

# plotting the pie chart
plt.pie(slices, labels = activities, colors=colors,
        startangle=90, shadow = True, explode = (0.1, 0, 0.1, 0.25),
        radius = 1.5, autopct = '%1.2f%%')

# plotting legend
plt.legend()

# showing the plot
plt.show()
```



Plotting 3D Graphs

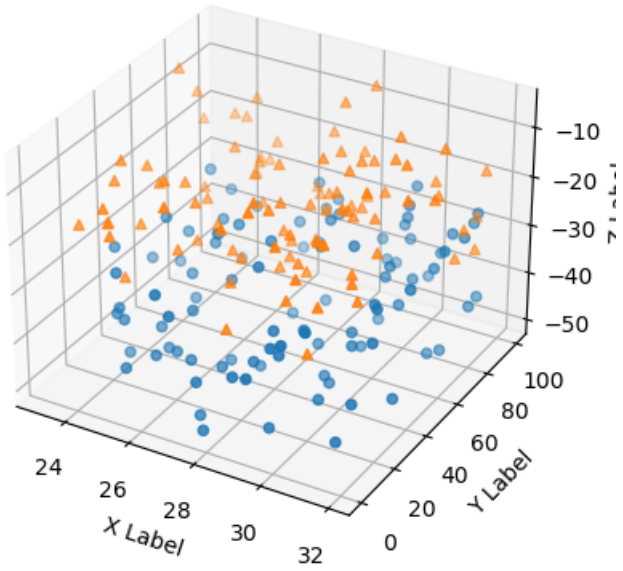
In [33]:

```
#Line 3D Plot
# This import registers the 3D projection but is otherwise unused.
from mpl_toolkits.mplot3d import Axes3D  #F401 unused import

import numpy as np
import matplotlib.pyplot as plt


plt.rcParams['legend.fontsize'] = 10

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

# Prepare arrays x, y, z
theta = np.linspace(-4 * np.pi, 4 * np.pi, 100)
z = np.linspace(-2, 2, 100)
r = z**2 + 1
x = r * np.sin(theta)
y = r * np.cos(theta)

ax.plot(x, y, z, label='parametric curve')
ax.legend()

plt.show()
```

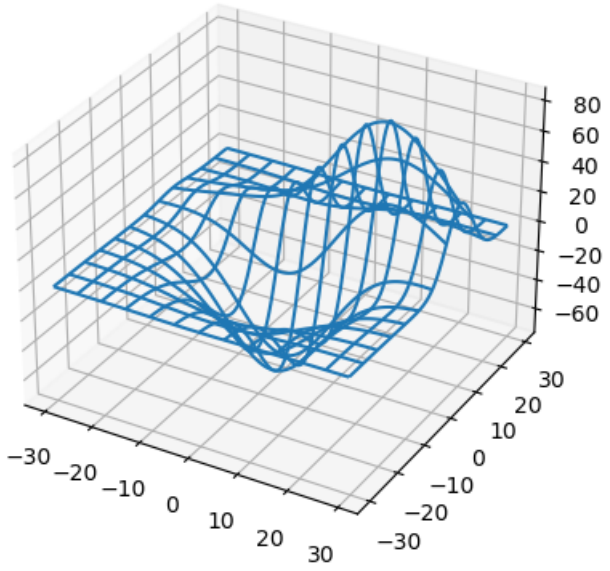


In [34]:

```python
#Scatter 3D Plot
# This import registers the 3D projection, but is otherwise unused.
from mpl_toolkits.mplot3d import Axes3D  # noqa: F401 unused import

import matplotlib.pyplot as plt
import numpy as np

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


def randrange(n, vmin, vmax):
    '''
    Helper function to make an array of random numbers having shape (n, )
    with each number distributed Uniform(vmin, vmax).
    '''
    return (vmax - vmin)*np.random.rand(n) + vmin

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

n = 100

# For each set of style and range settings, plot n random points in the box
# defined by x in [23, 32], y in [0, 100], z in [zlow, zhigh].
for m, zlow, zhigh in [('o', -50, -25), ('^', -30, -5)]:
    xs = randrange(n, 23, 32)
    ys = randrange(n, 0, 100)
    zs = randrange(n, zlow, zhigh)
    ax.scatter(xs, ys, zs, marker=m)

ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')

plt.show()
```

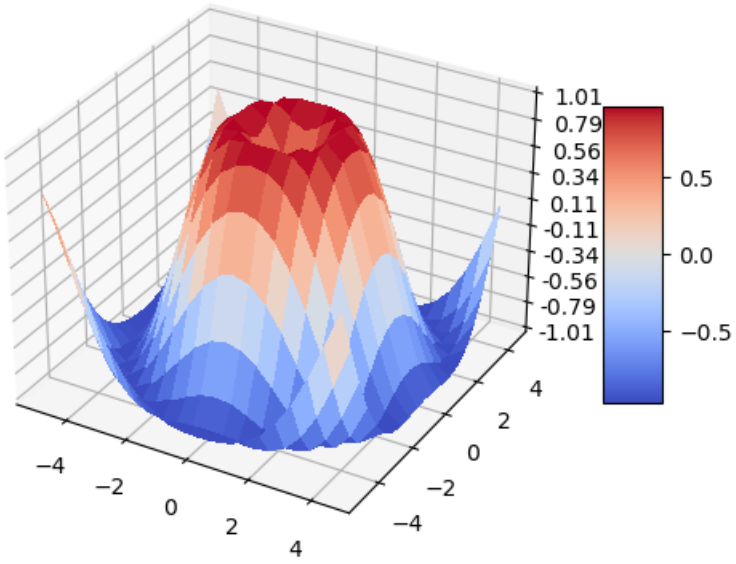


In [35]:
```
#WireFrame 3D Plot
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt


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

# Grab some test data.
X, Y, Z = axes3d.get_test_data(0.05)

# Plot a basic wireframe.
ax.plot_wireframe(X, Y, Z, rstride=10, cstride=10)

plt.show()
```



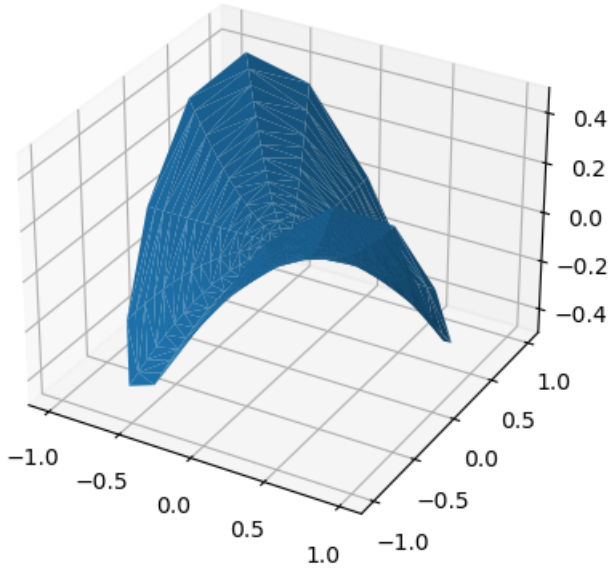
In [36]:
```
#Surface 3D Plot
# This import registers the 3D projection but is otherwise unused.
from mpl_toolkits.mplot3d import Axes3D  # F401 unused import

import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np


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

```
# Make data.
X = np.arange(-5, 5, 0.5)
Y = np.arange(-5, 5, 0.5)
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.coolwarm,
                       linewidth=0, antialiased=False)

# Customize the z axis.
ax.set_zlim(-1.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```

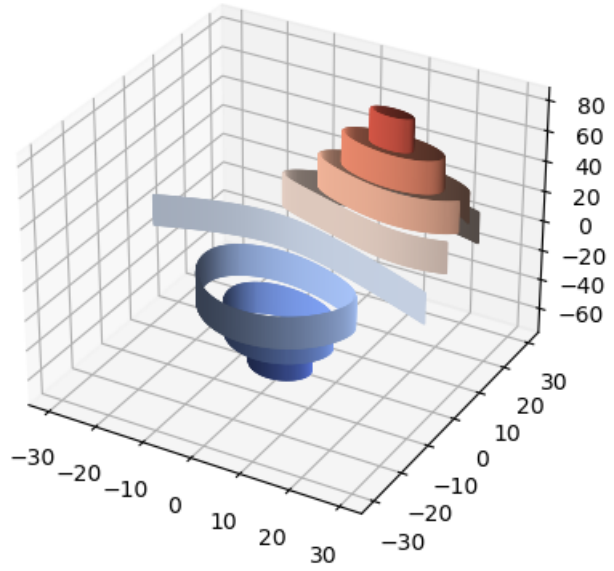


In [37]:

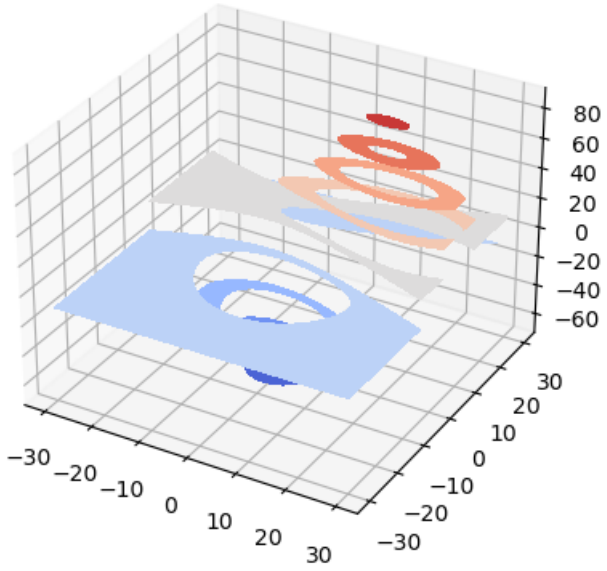
```
#Tri-Surface 3D Plot
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np


n_radii = 16
n_angles = 16

# Make radii and angles spaces (radius r=0 omitted to eliminate duplication).
radii = np.linspace(0.125, 1.0, n_radii)
angles = np.linspace(0, 2*np.pi, n_angles, endpoint=False)

# Repeat all angles for each radius.
angles = np.repeat(angles[..., np.newaxis], n_radii, axis=1)

# Convert polar (radii, angles) coords to cartesian (x, y) coords.
# (0, 0) is manually added at this stage,  so there will be no duplicate
# points in the (x, y) plane.
x = np.append(0, (radii*np.cos(angles)).flatten())
y = np.append(0, (radii*np.sin(angles)).flatten())

# Compute z to make the pringle surface.
z = np.sin(-x*y)

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

ax.plot_trisurf(x, y, z, linewidth=0.2, antialiased=True)

plt.show()
```



In [42]:
```
# Contour 3D Plot
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from matplotlib import cm

fig = plt.figure()
ax = fig.add_subplot(projection='3d')
X, Y, Z = axes3d.get_test_data(0.005)
cset = ax.contour(X, Y, Z, extend3d=True, cmap=cm.coolwarm)
ax.clabel(cset, fontsize=9, inline=1)

plt.show()
```

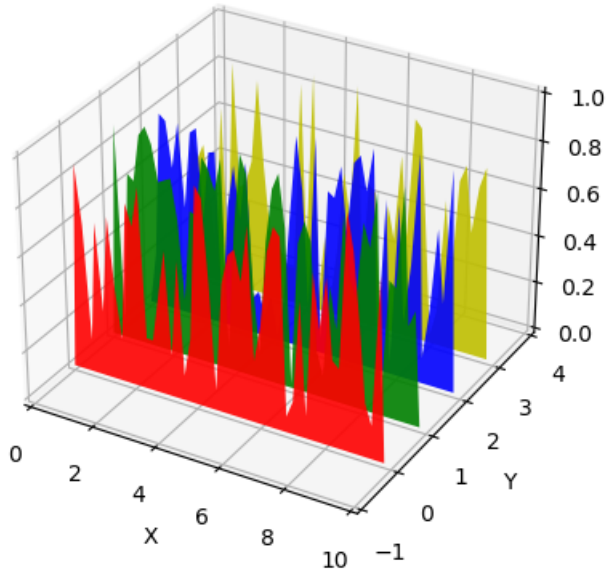


In [38]:
```
#Filled Contour 3D Plot
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from matplotlib import cm

fig = plt.figure()
ax = fig.add_subplot(projection='3d')
X, Y, Z = axes3d.get_test_data(0.005)
cset = ax.contourf(X, Y, Z, cmap=cm.coolwarm)
ax.clabel(cset, fontsize=9, inline=1)

plt.show()
```

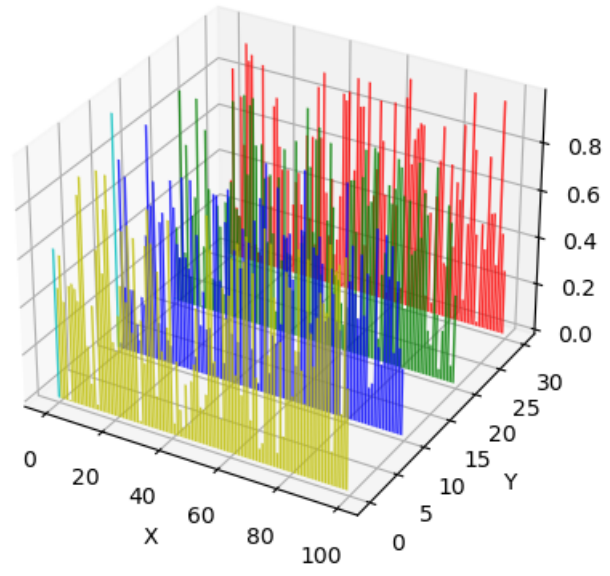


In [39]:

```
#Polygon 3D Plot
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.collections import PolyCollection
import matplotlib.pyplot as plt
from matplotlib import colors as mcolors
import numpy as np


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


def cc(arg):
    return mcolors.to_rgba(arg, alpha=0.9)

xs = np.arange(0, 10, 0.2)
verts = []
zs = [0.0, 1.0, 2.0, 3.0]
for z in zs:
    ys = np.random.rand(len(xs))
    ys[0], ys[-1] = 0, 0
    verts.append(list(zip(xs, ys)))

poly = PolyCollection(verts, facecolors=[cc('r'), cc('g'), cc('b'),
                                         cc('y')])
poly.set_alpha(0.9)
ax.add_collection3d(poly, zs=zs, zdir='y')

ax.set_xlabel('X')
ax.set_xlim3d(0, 10)
ax.set_ylabel('Y')
ax.set_ylim3d(-1, 4)
ax.set_zlabel('Z')
ax.set_zlim3d(0, 1)

plt.show()
```



In [40]:
```python
#Bar 3D Plot
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()
ax = fig.add_subplot(projection='3d')
for c, z in zip(['r', 'g', 'b', 'y'], [30, 20, 10, 0]):
    xs = np.arange(100)
    ys = np.random.rand(100)

    # You can provide either a single color or an array. To demonstrate this,
    # the first bar of each set will be colored cyan.
    cs = [c] * len(xs)
    cs[0] = 'c'
    ax.bar(xs, ys, zs=z, zdir='y', color=cs, alpha=0.8)

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

plt.show()
```



In [41]:
```python
#Quiver 3D
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()
```

```
ax = fig.add_subplot(projection='3d')

# Make the grid
x, y, z = np.meshgrid(np.arange(-0.8, 1, 0.4),
                      np.arange(-0.8, 1, 0.3),
                      np.arange(-0.8, 1, 0.3))

# Make the direction data for the arrows
u = np.sin(np.pi * x) * np.cos(np.pi * y) * np.cos(np.pi * z)
v = -np.cos(np.pi * x) * np.sin(np.pi * y) * np.cos(np.pi * z)
w = (np.sqrt(2.0 / 3.0) * np.cos(np.pi * x) * np.cos(np.pi * y) *
     np.sin(np.pi * z))

ax.quiver(x, y, z, u, v, w, length=0.2, normalize=True)

plt.show()
```