Regression Model

In [4]:
```
import pandas as pd
import numpy as np
from math import sqrt
```

In [5]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_error,root_mean_squ
y_true = [10,20,30,40,50,60,70,80,90,100]
y_pred = [12,24,31,43,52,62,73,82,93,101]
mean_absolute_error(y_true, y_pred)
```

Out[5]:
```
2.3
```

In [6]:
```
mean_squared_error(y_true, y_pred)
```

Out[6]:
```
6.1
```

In [7]:
```
root_mean_squared_error(y_true, y_pred)
```

Out[7]:
```
2.4698178070456938
```

In [8]:
```
r2=r2_score(y_true, y_pred)
r2
```

Out[8]:
```
0.9926060606060606
```

In [9]:
```
adj=1-(1-r2)*(10-1)/(10-1-1)
adj
```

Out[9]:
```
0.9916818181818182
```

Classification Model

In [10]:
```
import numpy as np
from sklearn.metrics import confusion_matrix,accuracy_score,f1_score,recall_scor
y_true = [0,0,1,1,0,0,1,0,1,0]
y_pred = [0,1,1,0,0,0,1,0,0,0]
confusion_matrix(y_true, y_pred)
```

Out[10]:
```
array([[5, 1],
       [2, 2]], dtype=int64)
```

In [11]:
```
print('Accuracy: %.3f' % accuracy_score(y_true, y_pred))
```

```
Accuracy: 0.700
```

In [12]:
```
accuracy_score(y_true, y_pred)
```

Out[12]:
```
0.7
```

In [13]:
```
print('Precision: %.3f' % precision_score(y_true, y_pred))
```

```
Precision: 0.667
```

```
In [14]: recall_score(y_true, y_pred)
```

```
Out[14]: 0.5
```

```
In [15]: f1_score(y_true, y_pred)
```

```
Out[15]: 0.5714285714285714
```

Classification Model For Multivalue

```
In [16]: import numpy as np
         from sklearn.metrics import confusion_matrix,accuracy_score,f1_score,recall_scor
         y_true = [0,1,2,0,0,1,2,2,1,1]
         y_pred = [1,1,2,2,0,0,1,0,2,1]
         confusion_matrix(y_true, y_pred)
```

```
Out[16]: array([[1, 1, 1],
                [1, 2, 1],
                [1, 1, 1]], dtype=int64)
```

```
In [17]: print('Accuracy: %.3f' % accuracy_score(y_true, y_pred))
```

```
Accuracy: 0.400
```

```
In [18]: print('Precision: %.3f' % precision_score(y_true, y_pred,average='macro'))
```

```
Precision: 0.389
```

```
In [19]: print('Precision: %.3f' % precision_score(y_true, y_pred,average='micro'))
```

```
Precision: 0.400
```

```
In [20]: recall_score(y_true, y_pred, average='macro')
```

```
Out[20]: 0.38888888888888884
```

```
In [21]: recall_score(y_true, y_pred, average='micro')
```

```
Out[21]: 0.4
```

```
In [22]: f1_score(y_true, y_pred,average='micro')
```

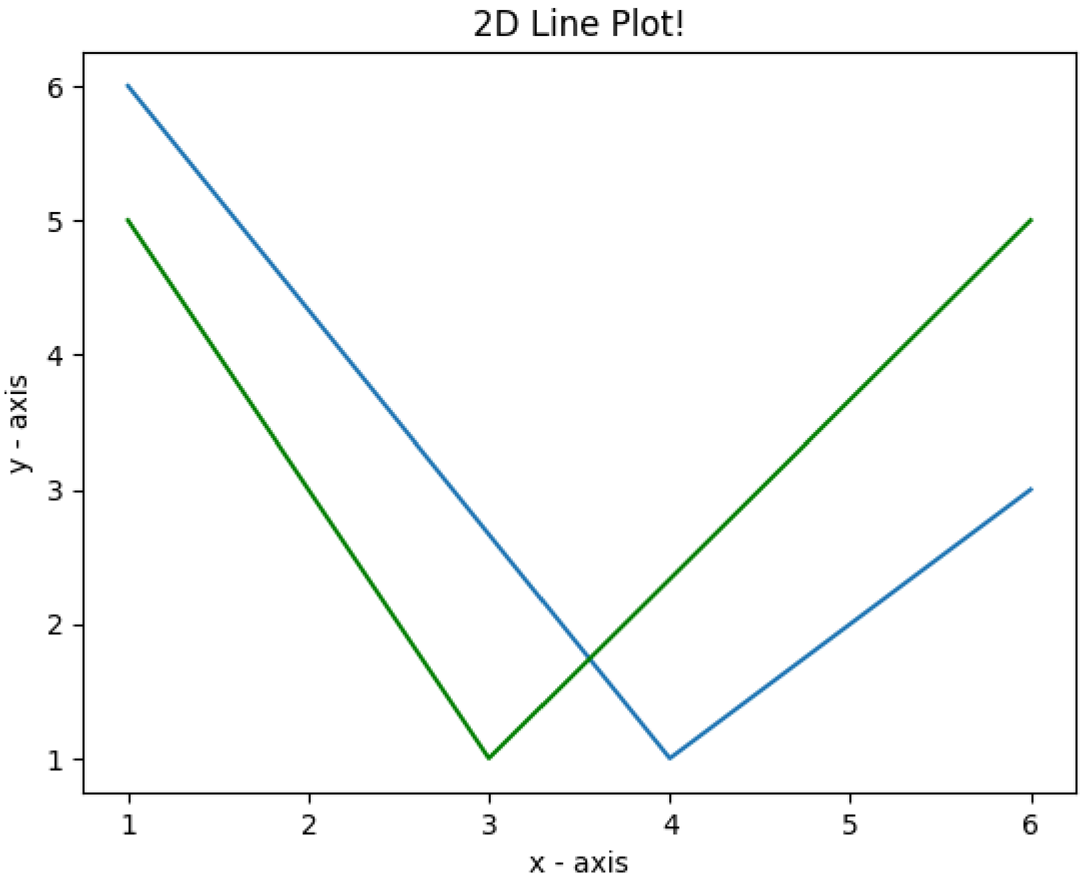
```
Out[22]: 0.4
```

Assigment

Line 2D Plot

```
In [61]: import matplotlib.pyplot as plt

         # x axis values
         x = [1,4,6]
         x1 = [1,3,6]
```

```
# corresponding y axis values
y = [6,1,3]
y1 = [5,1,5]


# plotting the points
plt.plot(x, y)
plt.plot(x1, y1, color='green')

# 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()
```

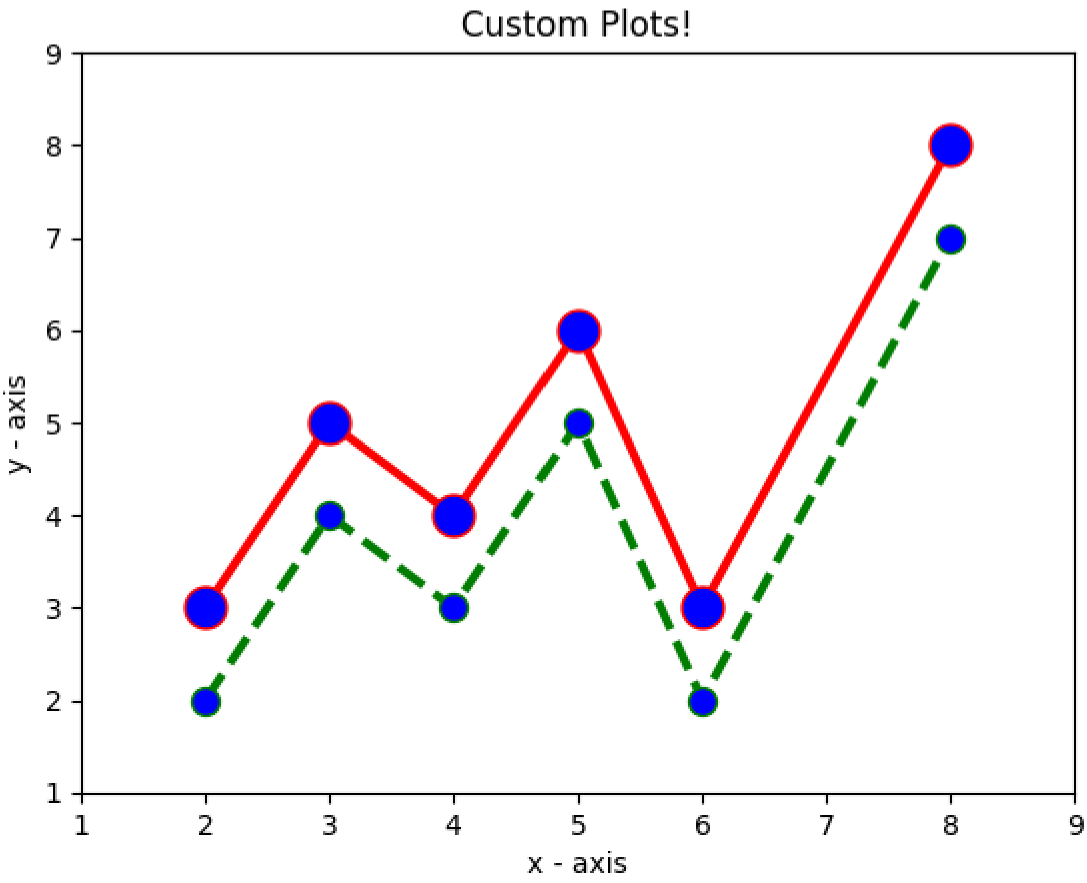


Custom 2D Plot

In [161…

```
import matplotlib.pyplot as plt

# x axis values
x = [2,3,4,5,6,8]
x1 = [2,3,4,5,6,8]
# corresponding y axis values
y = [2,4,3,5,2,7]
```

```
y1 = [3,5,4,6,3,8]

# plotting the points
plt.plot(x, y, color='green', linestyle='dashed', linewidth = 3,
        marker='o', markerfacecolor='blue', markersize=10)
plt.plot(x1, y1, color='red', linestyle='solid', linewidth = 3,
        marker='o' , markerfacecolor='blue', markersize=15)

# setting x and y axis range
plt.ylim(1,9)
plt.xlim(1,9)

# 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()
```

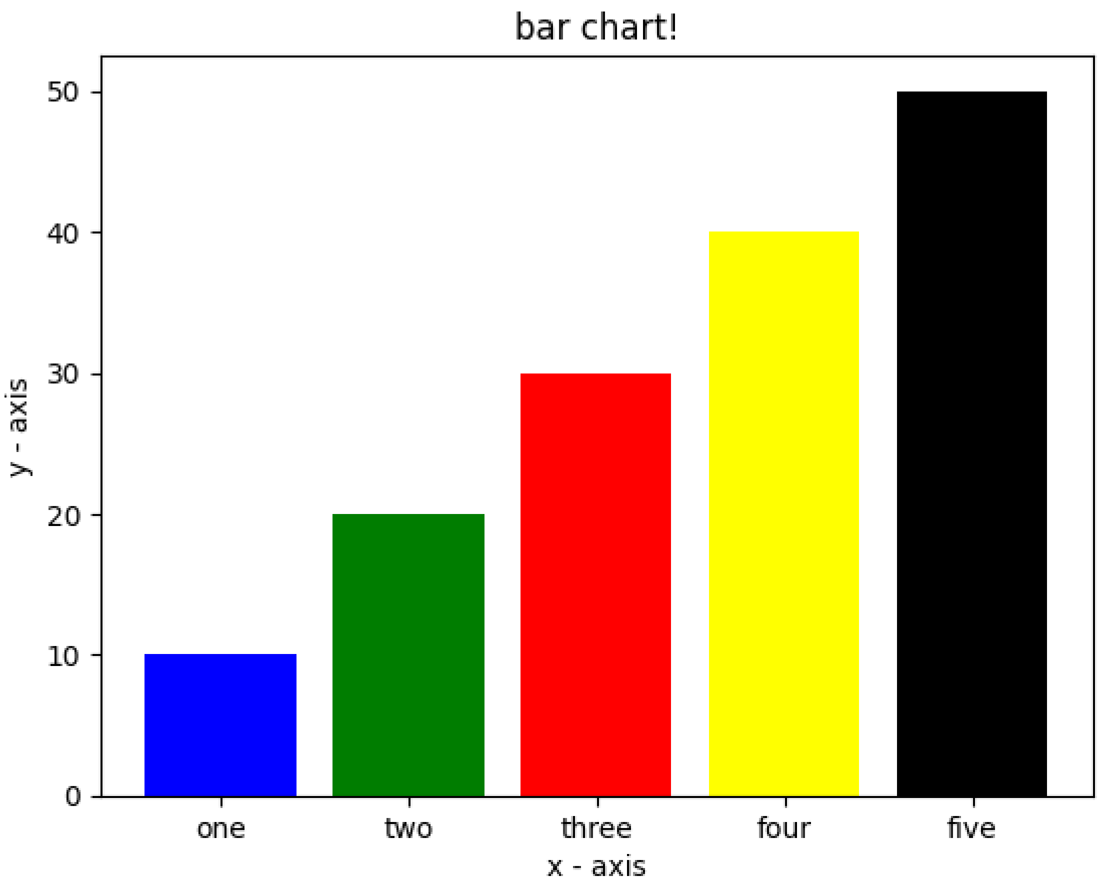


Bar 2D Chart

In [63]:

```
import matplotlib.pyplot as plt

# x-coordinates of left sides of bars
left = [1, 2, 3, 4, 5]
```

```
# heights of bars
height = [10, 20, 30, 40, 50]

# labels for bars
tick_label = ['one', 'two', 'three', 'four', 'five']

# plotting a bar chart
plt.bar(left, height, tick_label = tick_label,
        width = 0.8, color = ['blue', 'green', 'red','yellow','black'])

# naming the x-axis
plt.xlabel('x - axis')
# naming the y-axis
plt.ylabel('y - axis')
# plot title
plt.title('bar chart!')

# function to show the plot
plt.show() Grouped Bar Chart
```

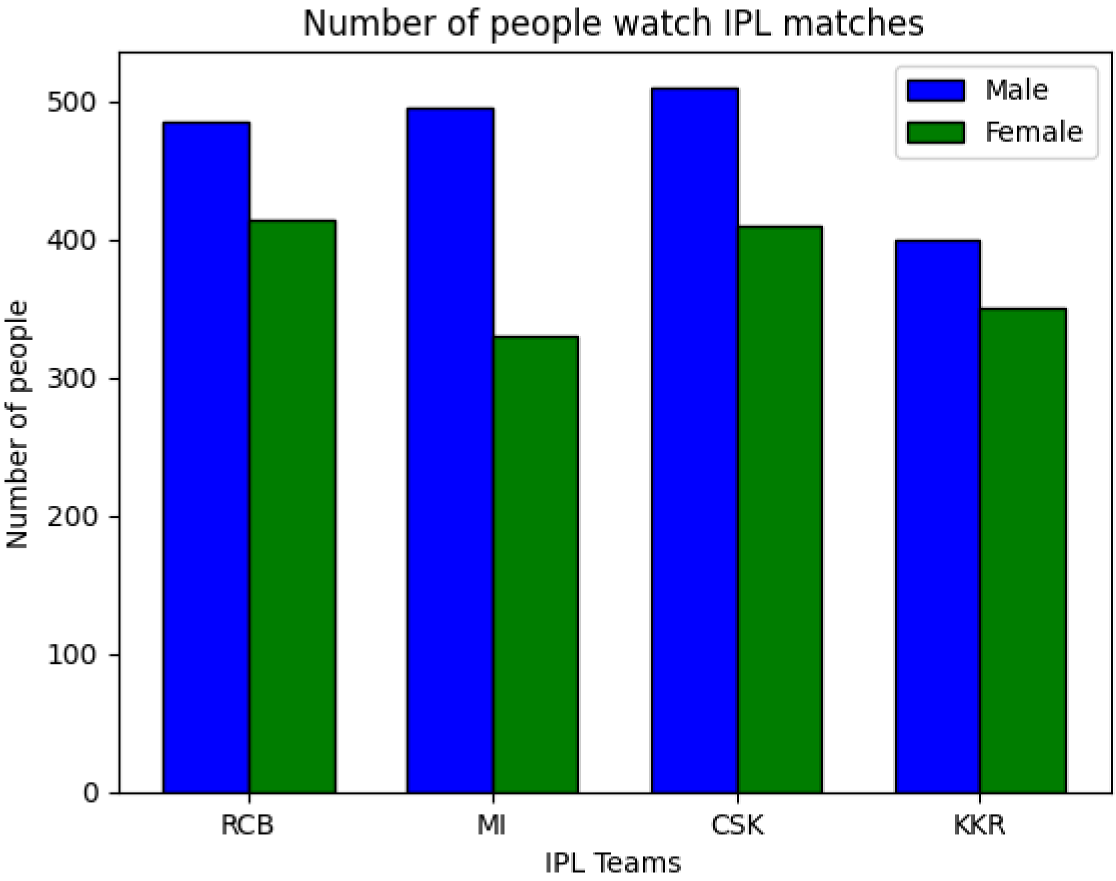


Grouped Bar Chart

In [77]:

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

# IPL Team data
teams = ['RCB', 'MI', 'CSK', 'KKR']
women_votes = [485, 495, 510, 400]
men_votes = [414, 330, 410, 350]
```

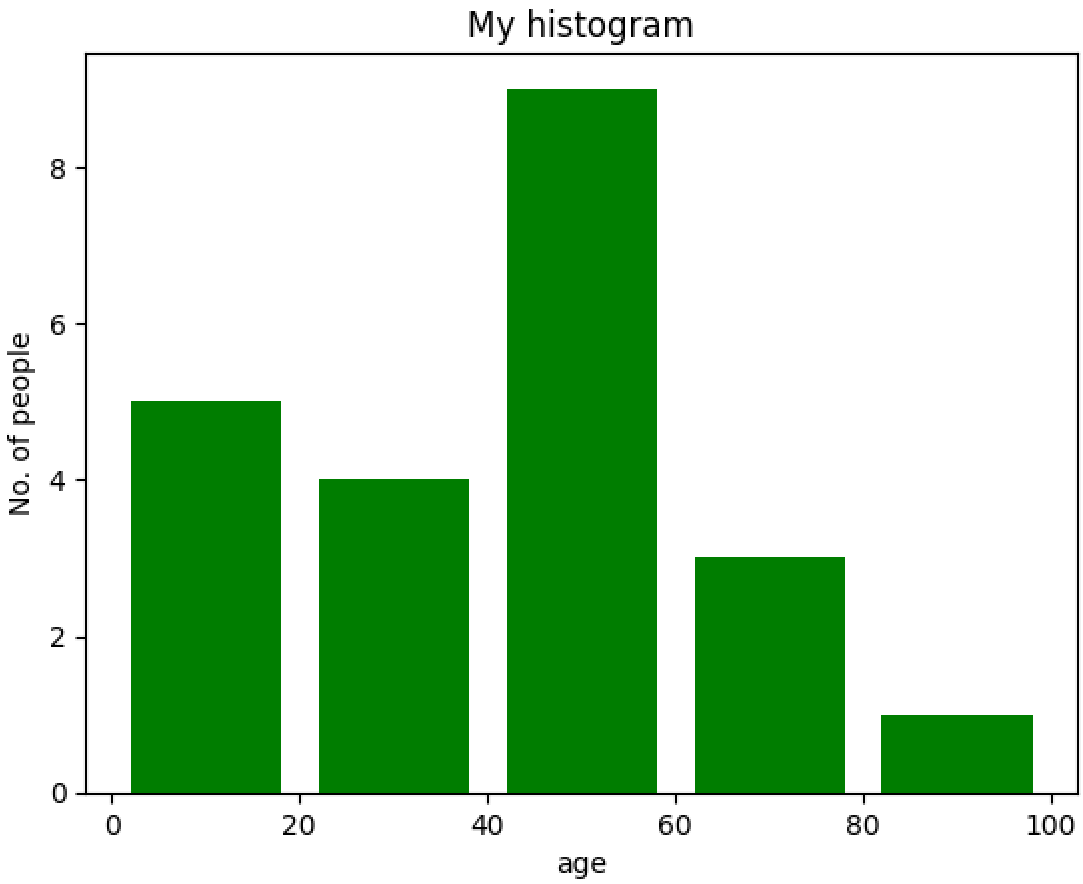
```
n = len(teams)
r = np.arange(n)
width = 0.35

plt.bar(r, women_votes, color='b', width=width, edgecolor='black', label='Male')
plt.bar(r + width, men_votes, color='g', width=width, edgecolor='black', label='

plt.xlabel("IPL Teams")
plt.ylabel("Number of people")
plt.title("Number of people watch IPL matches")

plt.xticks(r + width / 2, teams)
plt.legend()

plt.show()
```



Histogram 2D Plot

In [80]:
```
import matplotlib.pyplot as plt
# frequencies
ages = [2,5,70,40,30,45,50,45,43,40,44,
        60,7,13,57,18,90,77,32,21,20,40]

# setting the ranges and no. of intervals
range = (0, 100)
bins = 5

# plotting a histogram
plt.hist(ages, bins, range, color ='green',
        histtype = 'bar', rwidth = 0.8)
```

```
# x-axis label
plt.xlabel('age')
# frequency label
plt.ylabel('No. of people')
# plot title
plt.title('My histogram')

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

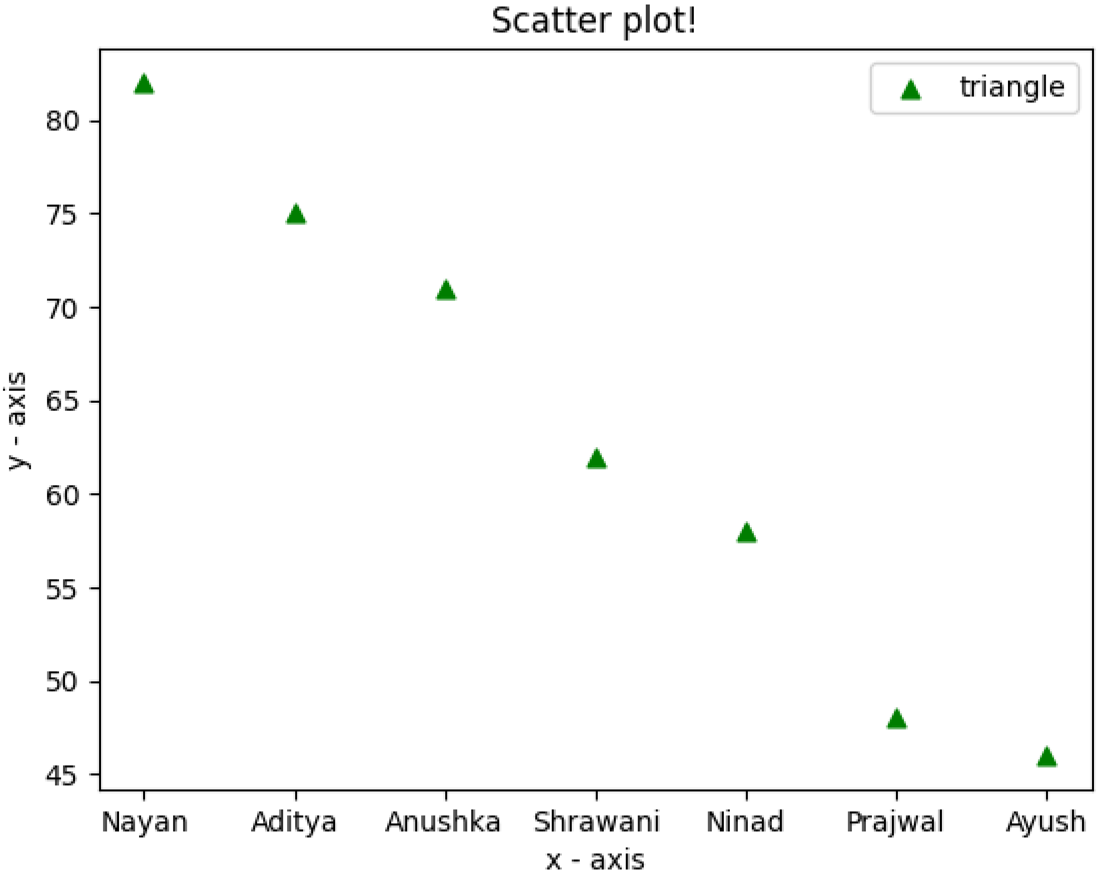


Scatter 2D Plot

In [114…

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

# Read the CSV file
df = pd.read_csv("name_and_marks.csv")

# Extracting x and y values from the DataFrame
x = df['Names']
y = df['Marks']

# Plotting points as a scatter plot
plt.scatter(x, y, label="triangle", color="green", marker="^", s=40)

# X-axis label
plt.xlabel('x - axis')
# Y-axis label
plt.ylabel('y - axis')
```

```
# Plot title
plt.title('Scatter plot!')
# Showing legend
plt.legend()

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

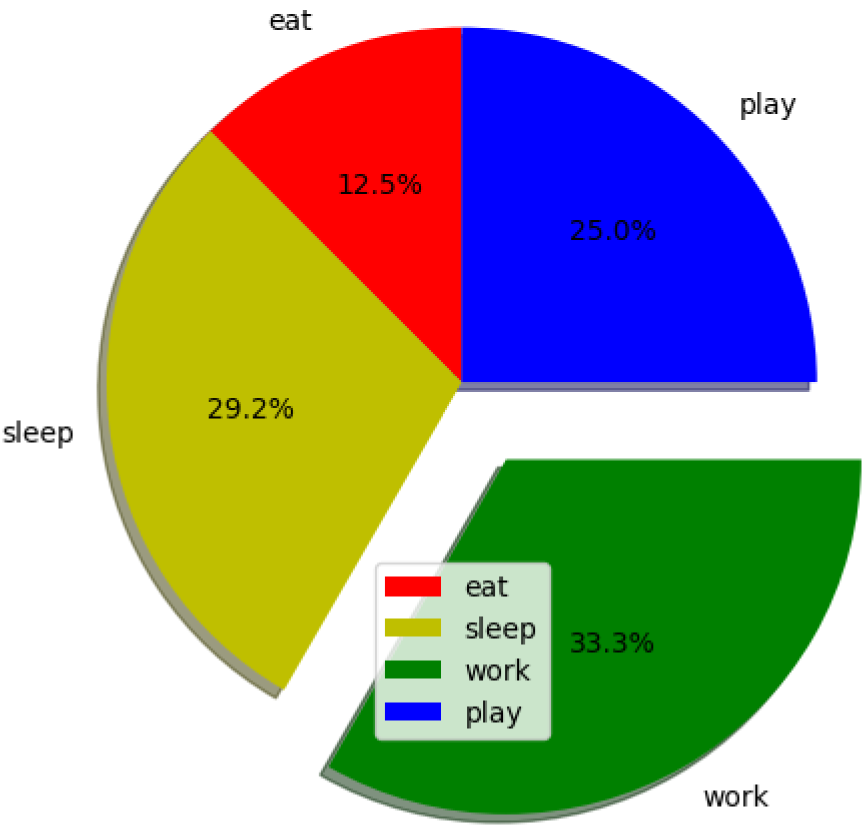


Pie-Chart 2D Plot

In [132…

```
import matplotlib.pyplot as plt

# Defining labels
activities = ['eat', 'sleep', 'work', 'play']

# Portion covered by each label
slices = [3, 7, 8, 6]

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

# Explode parameter (0.3 indicates the explode distance for 'play' slice)
explode = (0, 0, 0.3, 0)

# Plotting the pie chart
plt.pie(slices, labels=activities, colors=colors,
        startangle=90, shadow=True, explode=explode,
        radius=1.2, autopct='%1.1f%%')

# Plotting legend
```

```
plt.legend()

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

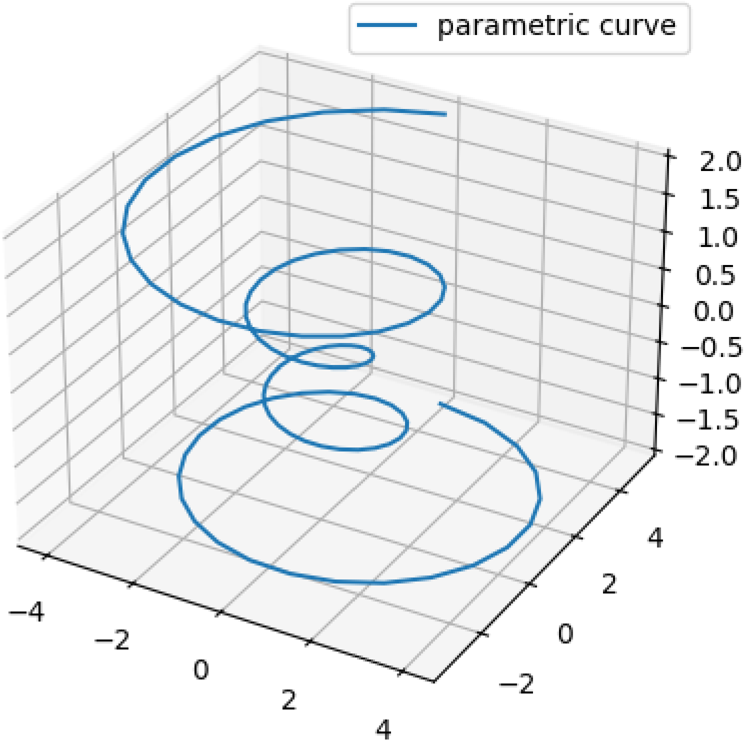


Plotting 3D Graphs

line 3D Plot

In [134…

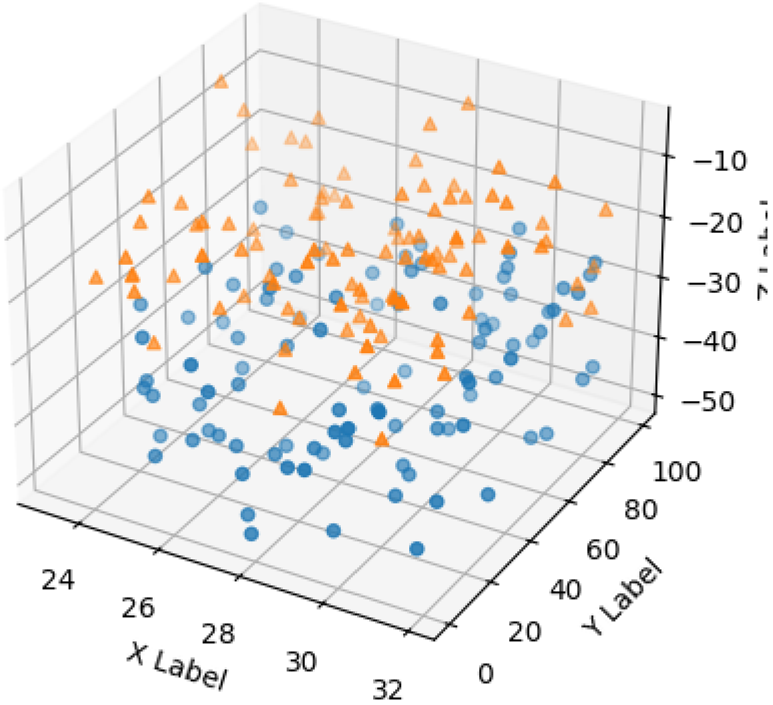
```
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt

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

fig = plt.figure()
ax = fig.add_subplot(111, 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()
```



Scatter 3D Plot

In [135…

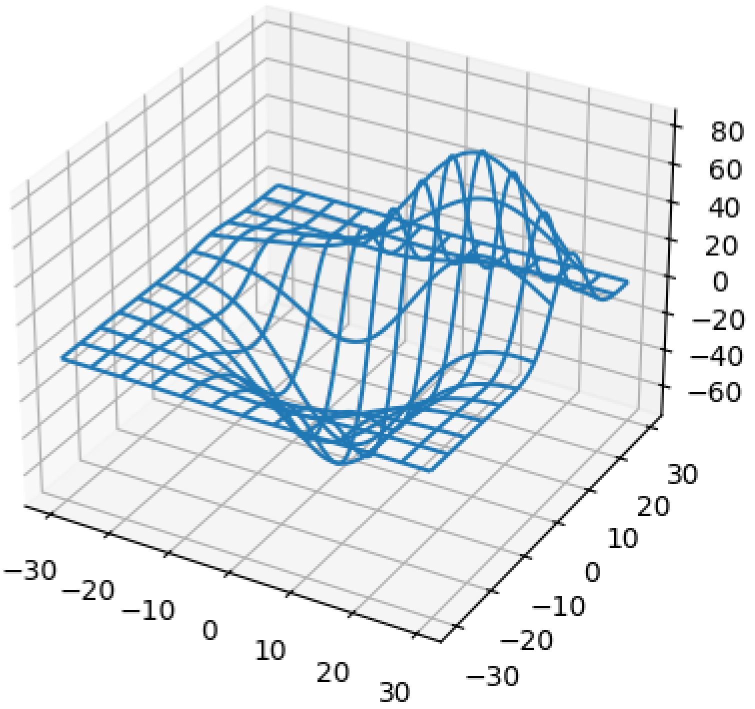
```
# This import registers the 3D projection, but is otherwise unused.
from mpl_toolkits.mplot3d import Axes3D  # noqa: F401 unused import

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(111, 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()
```



WireFrame 3D Plot

In [136…

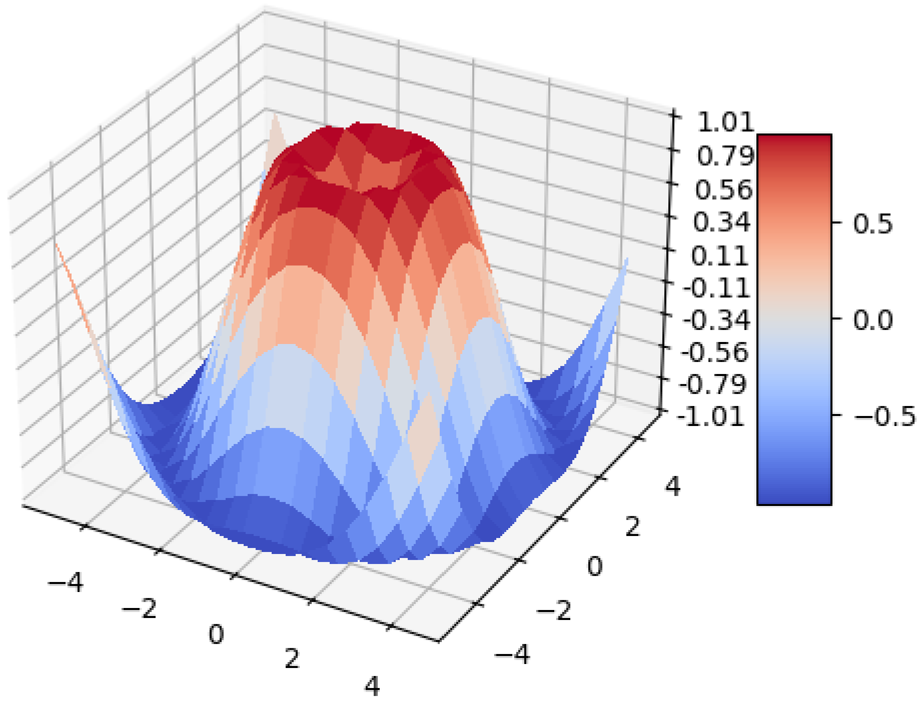
```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt


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()
```



Surface 3D Plot

In [138…

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

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

fig = plt.figure()
ax = fig.add_subplot(111, 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=plt.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()
```



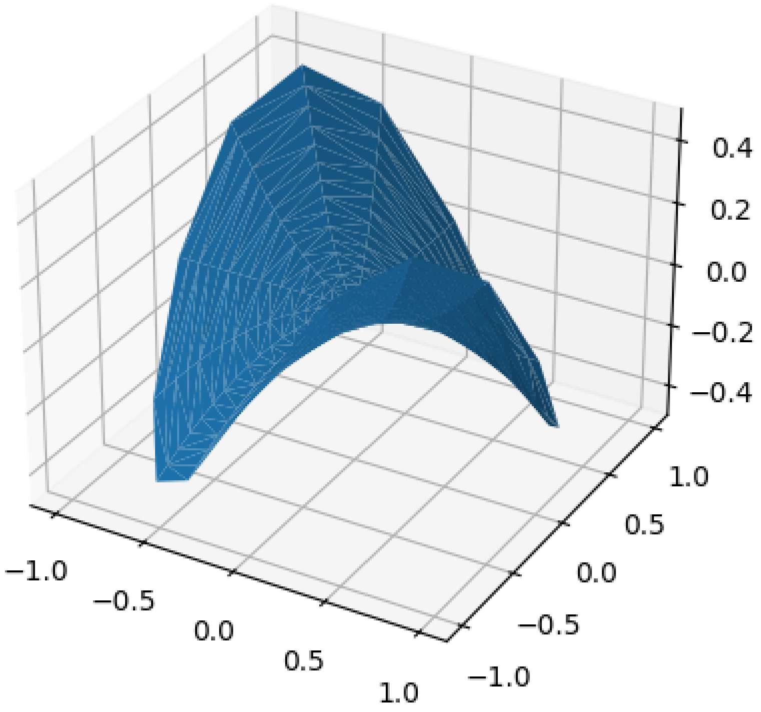
Tri-Surface 3D Plot

In [141…

```
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt

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

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

# Make radii and angles spaces (radius r=0 omitted to eliminate duplication).
n_radii = 16
n_angles = 16
radii = np.linspace(0.125, 1.0, n_radii)
angles = np.linspace(0, 2*np.pi, n_angles, endpoint=False)
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)

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

Out[141…

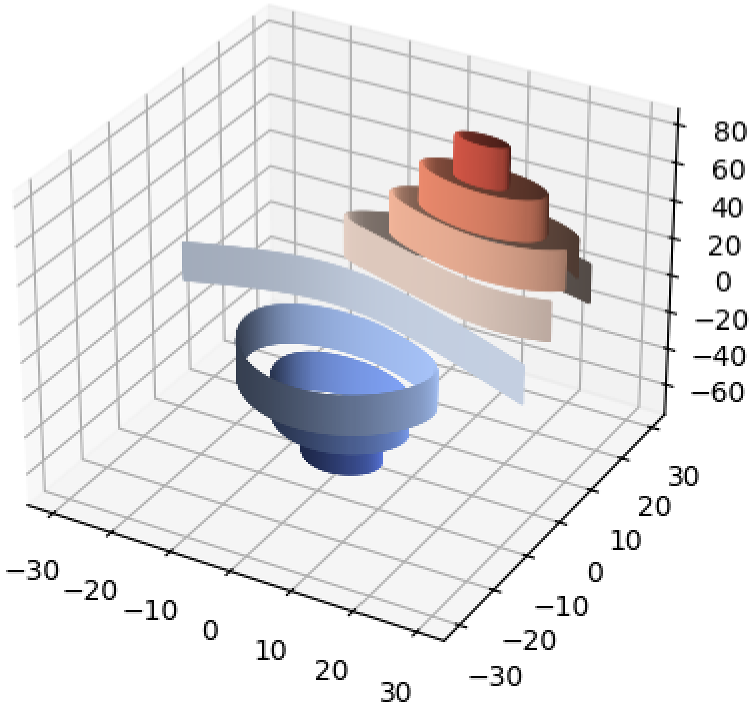
```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x2e09fb0f390>
```



Contour 3D Plot

In [153…

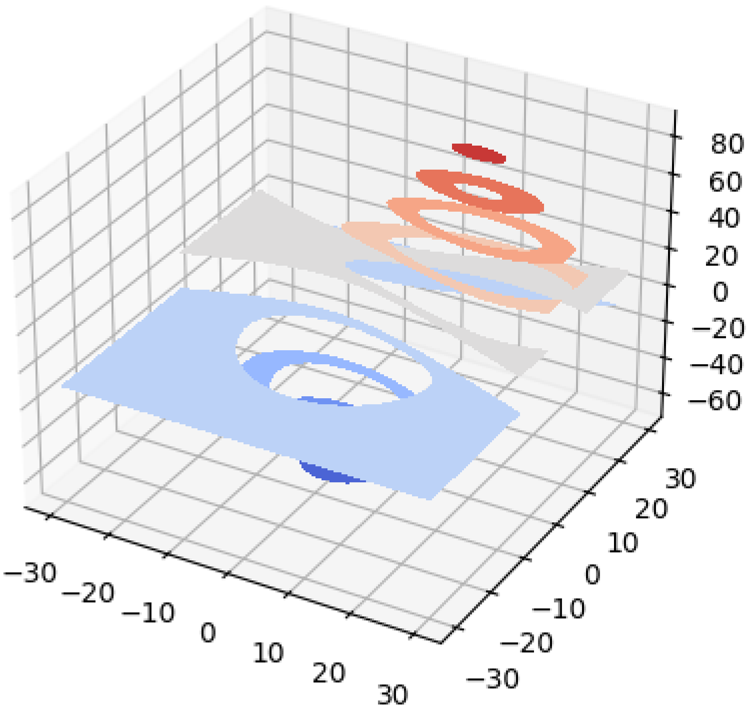
```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from matplotlib import cm

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()
```



Filled Contour 3D Plot

In [152…

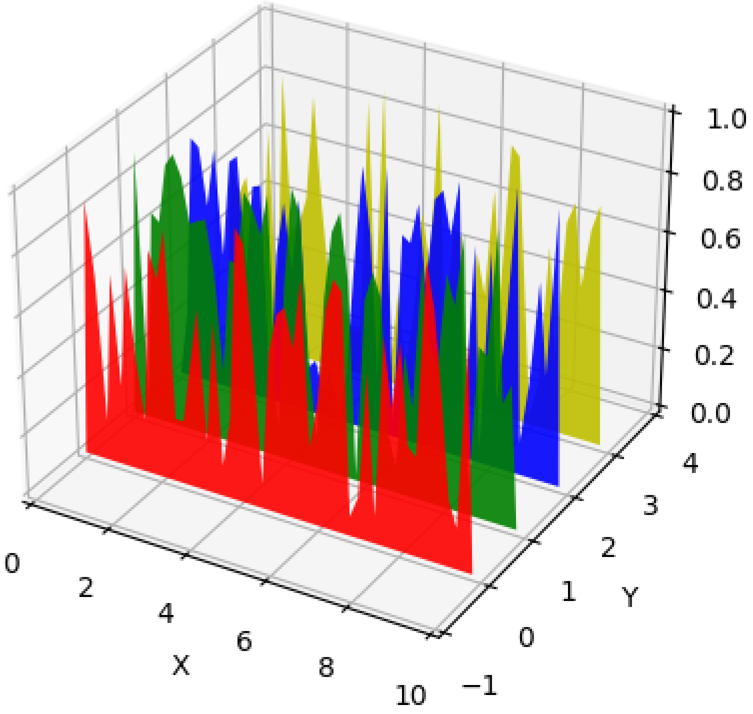
```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from matplotlib import cm

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()
```



Polygon 3D Plot

In [154…

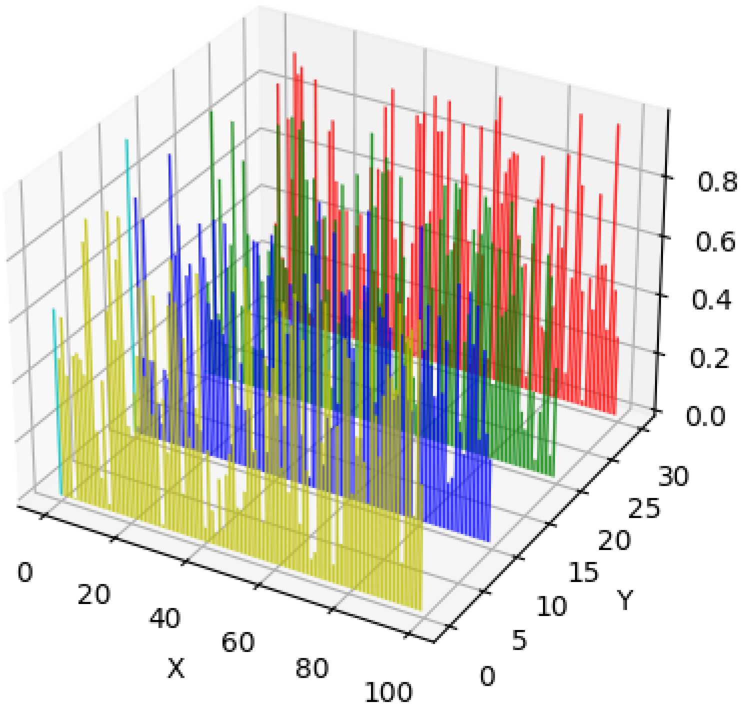
```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.collections import PolyCollection
import matplotlib.pyplot as plt
from matplotlib import colors as mcolors
import numpy as np


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()
```



Bar 3D Plot

In [155…

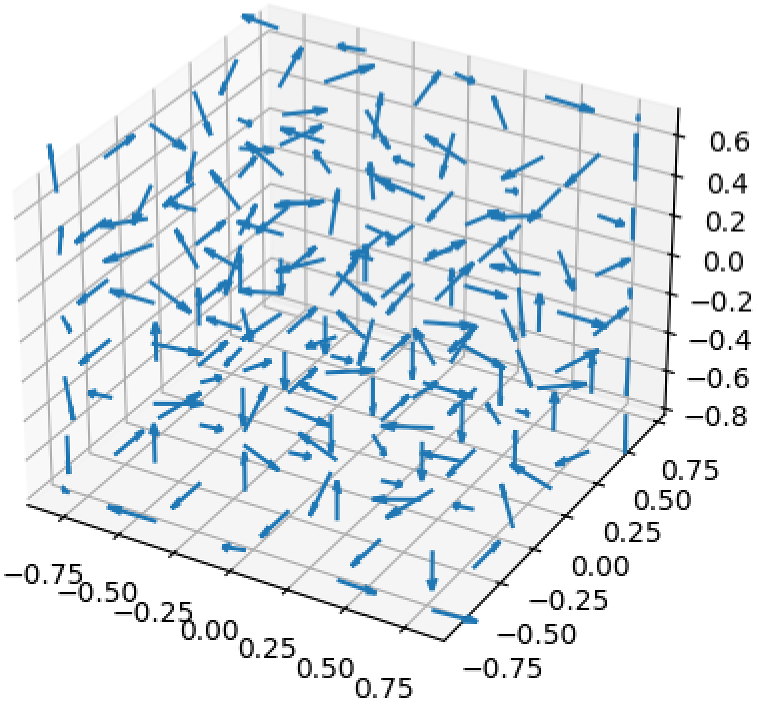
```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()
ax = fig.add_subplot(111, 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()
```



Quiver 3D

In [158…

```
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()
```


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75

```
In [ ]:
```