In [1]:
```
# Plotting using matplotlib library
```

In [2]:
```
import pandas as pd
tab=pd.read_excel(r'C:\Users\lenovo\Downloads\IBM-313 Marks.xlsx')
tab
```

Out[2]:

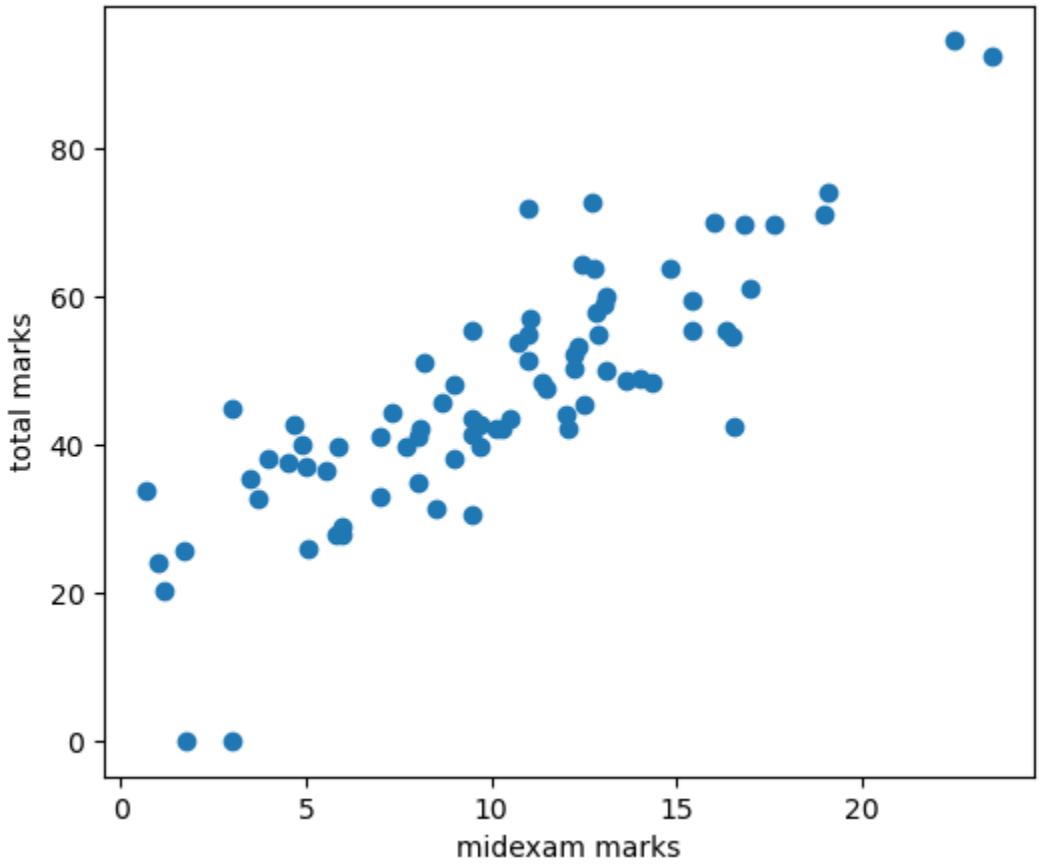
	S.No.	midexam	miniproject	total_internal	endexam	total
0	1	5.00	20	25.00	12.0	37.00
1	2	11.05	20	31.05	26.0	57.05
2	3	8.10	20	28.10	14.0	42.10
3	4	6.00	10	16.00	13.0	29.00
4	5	11.35	20	31.35	17.0	48.35
...	...	...	...	...	...	...
74	75	12.05	10	22.05	20.0	42.05
75	76	12.25	10	22.25	28.0	50.25
76	77	1.75	10	11.75	28.0	0.00
77	78	3.00	10	13.00	11.0	0.00
78	79	5.80	10	15.80	12.0	27.80

79 rows × 6 columns

Scatter plot

A scatter plot is a mathematical diagram using Cartesian coordinates to display values for two variables for a set of data. The data are displayed as a collection of points, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis. The points that are far from the population can be termed as an outlier.

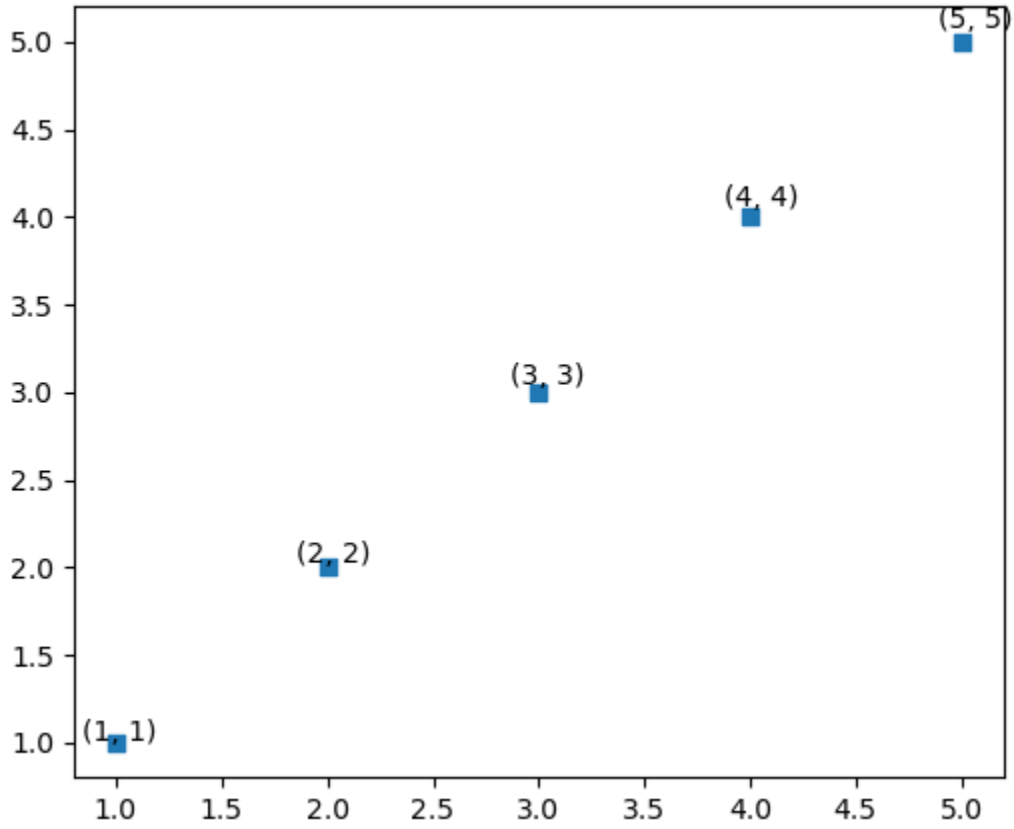
In [3]:
```
#scatter plot
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(6,5))
plt.scatter(tab['midexam'] , tab['total'])
plt.xlabel('midexam marks')
plt.ylabel('total marks')
plt.show()
```



In [4]:

```
import matplotlib.pyplot as plt

x_coords = [1,2,3,4,5]
y_coords = [1,2,3,4,5]

fig = plt.figure(figsize=(6,5))
plt.scatter(x_coords, y_coords, marker='s')


for x, y in zip(x_coords, y_coords):
    plt.annotate(
        '(%s, %s)' %(x, y),
        xy=(x, y),
        textcoords='offset points',
        ha='center')

plt.show()
```

```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_17320\495081922.py:11: UserWarning: You hav
e used the `textcoords` kwarg, but not the `xytext` kwarg.  This can lead to surprising
results.
  plt.annotate(
```

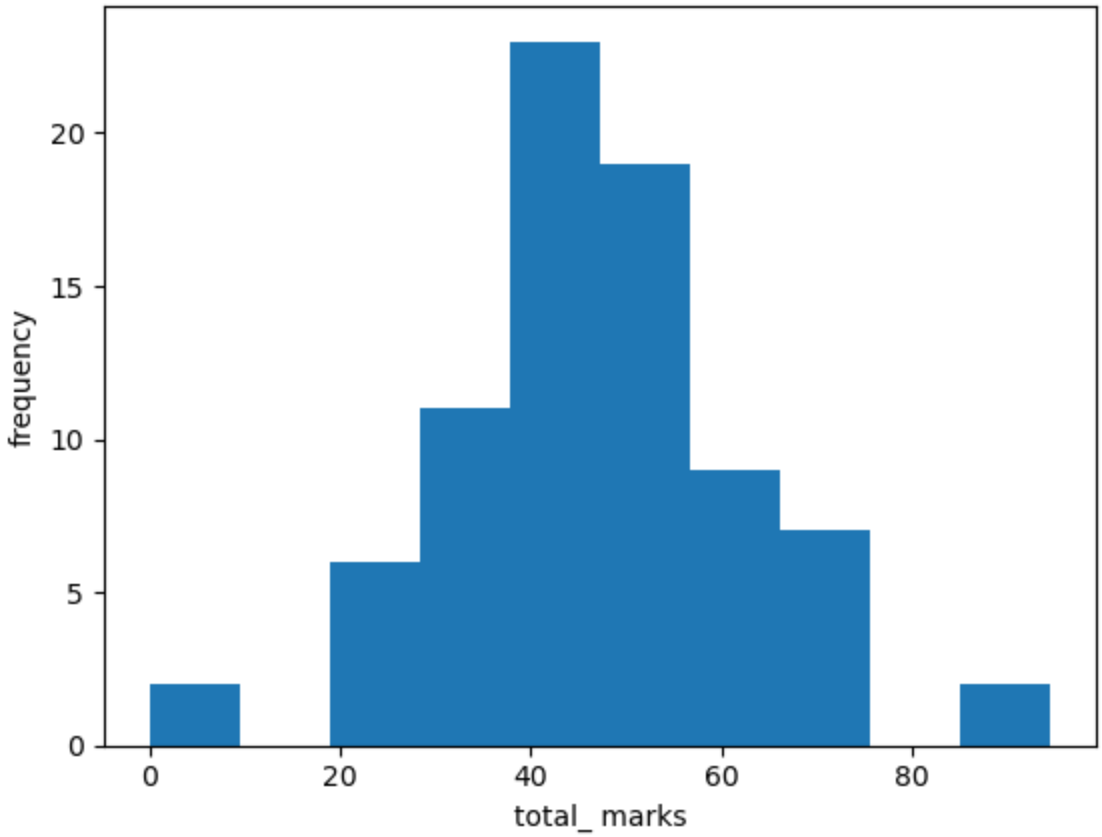


Histogram

A histogram is a graph that shows the frequency of numerical data using rectangles. The height of a rectangle (the vertical axis) represents the distribution frequency of a variable (the amount, or how often that variable appears).

In [5]:
```
import matplotlib.pyplot as plt

plt.hist(tab['total'])
plt.xlabel('total_ marks')
plt.ylabel('frequency')
```

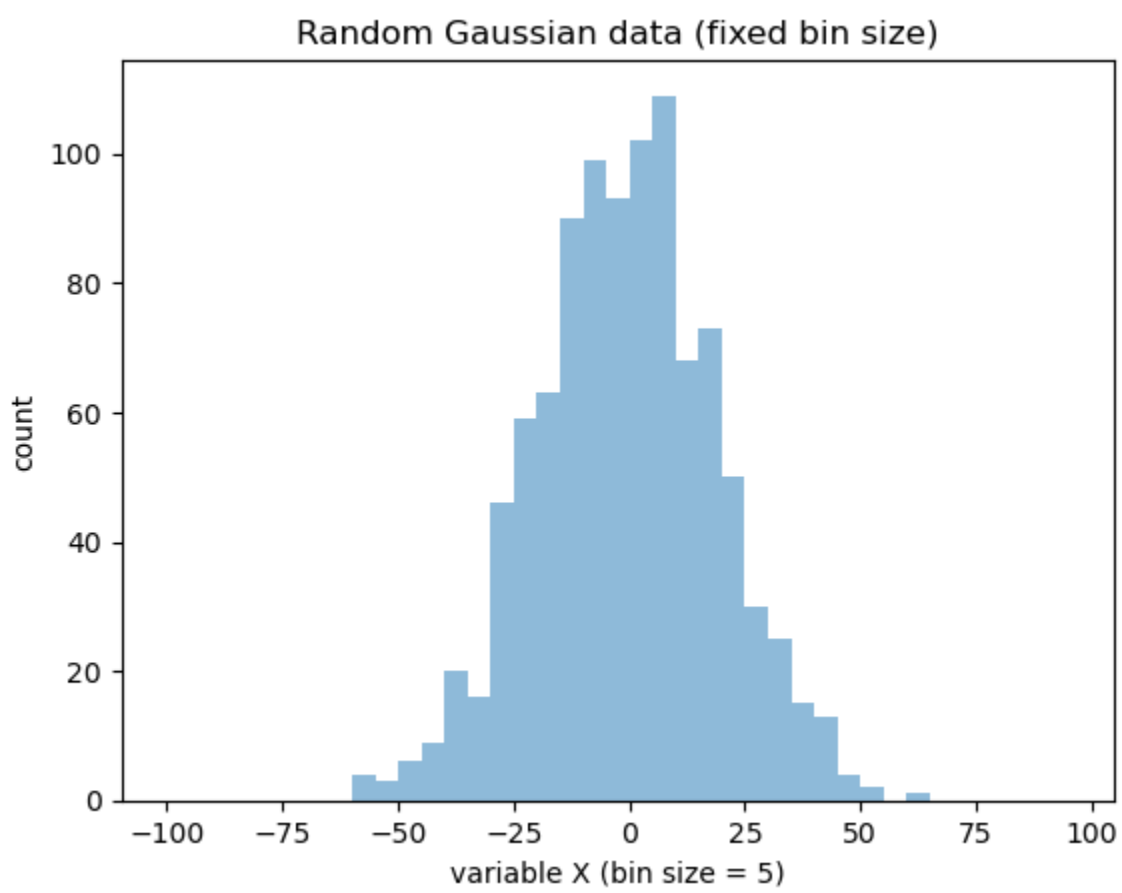
Out[5]:
```
Text(0, 0.5, 'frequency')
```



Histogram by defining Bins

In [6]:

```
import numpy as np
import random
from matplotlib import pyplot as plt

data = np.random.normal(0, 20, 1000)

# fixed bin size
bins = np.arange(-100, 100, 5) # fixed bin size

plt.hist(data, bins=bins, alpha=0.5)
plt.title('Random Gaussian data (fixed bin size)')
plt.xlabel('variable X (bin size = 5)')
plt.ylabel('count')

plt.show()
```

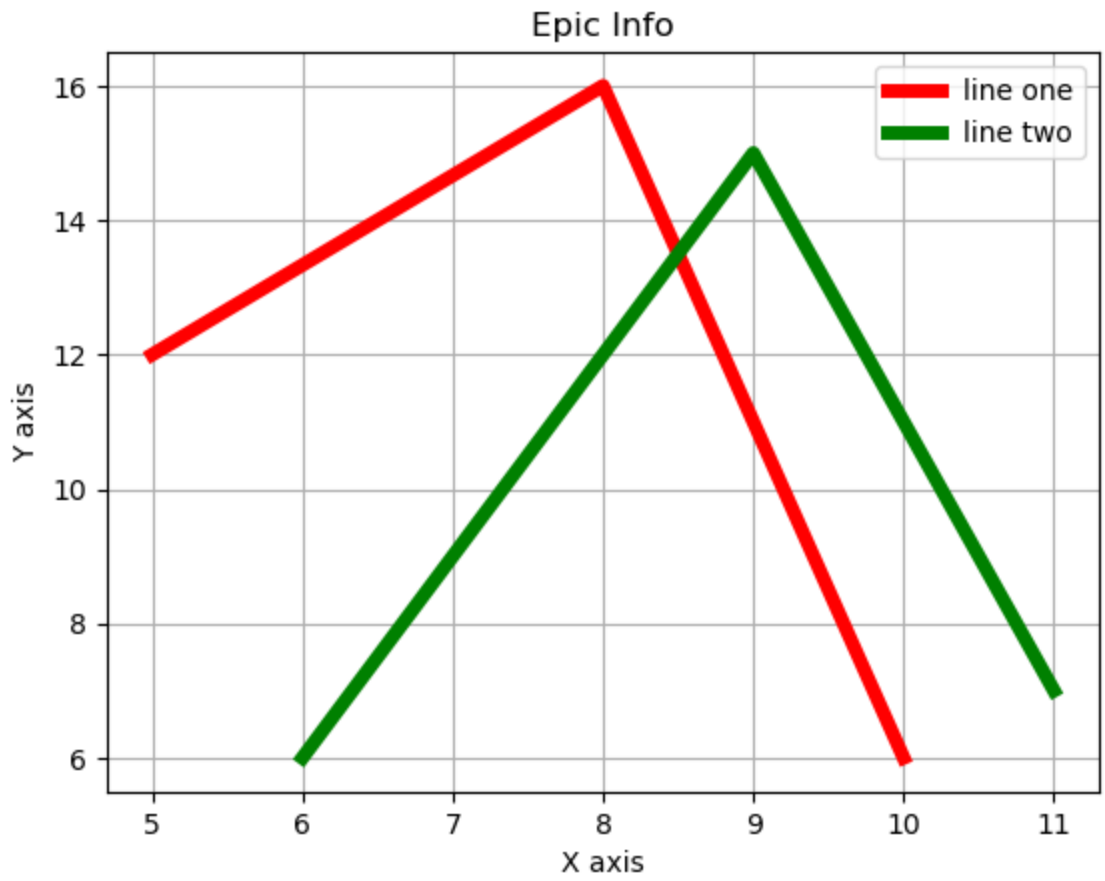


In the context of a histogram, "bins" refer to the intervals into which the entire range of data is divided. Each bin represents a range of values, and the height (or length) of each bar in the histogram indicates how many data points fall within that bin's range.

Line graph

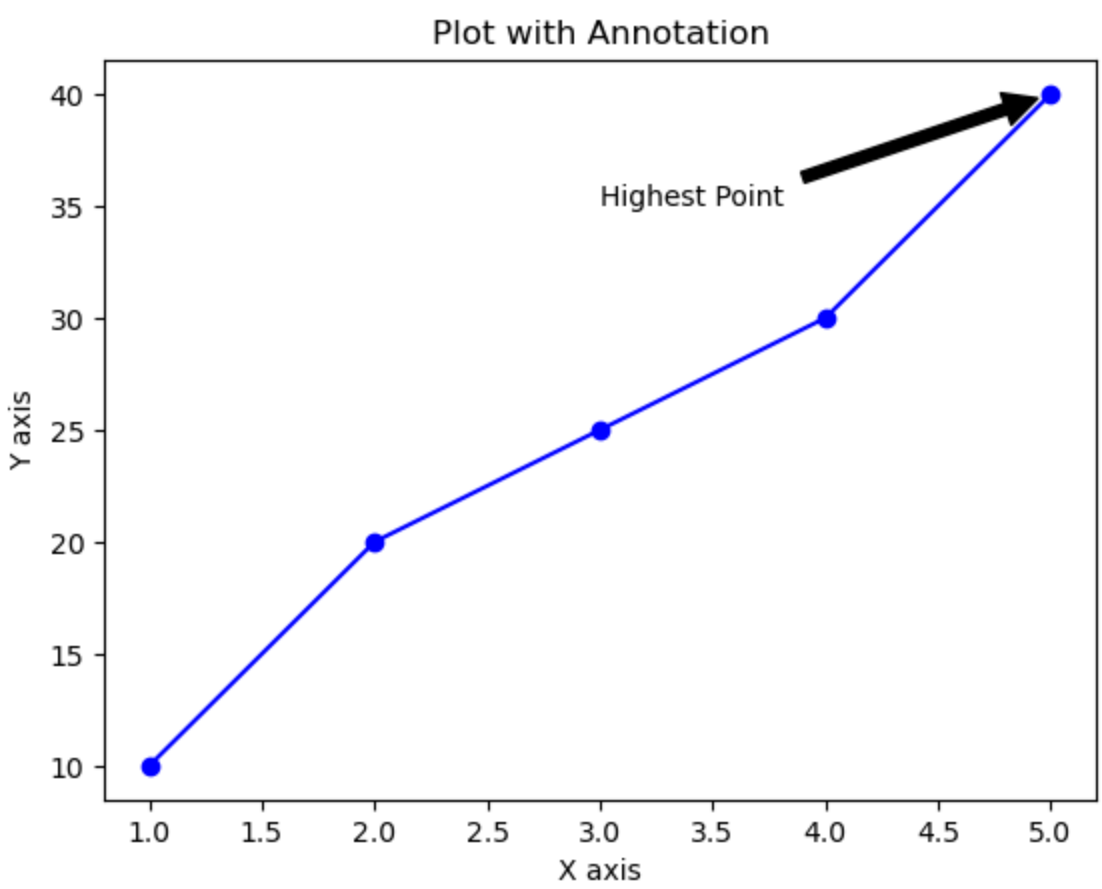
Line charts are used to represent the relation between two data X and Y on a different axis

In [7]:
```
# line graph

from matplotlib import pyplot as plt

x = [5,8,10]
y = [12,16,6]
x2 = [6,9,11]
y2 = [6,15,7]
plt.plot(x,y,label='line one', linewidth=5, color='r')
plt.plot(x2,y2,label='line two',linewidth=5, color='g')
plt.title('Epic Info')
plt.ylabel('Y axis')
plt.xlabel('X axis')
plt.legend()
plt.grid(True)
plt.show()
```

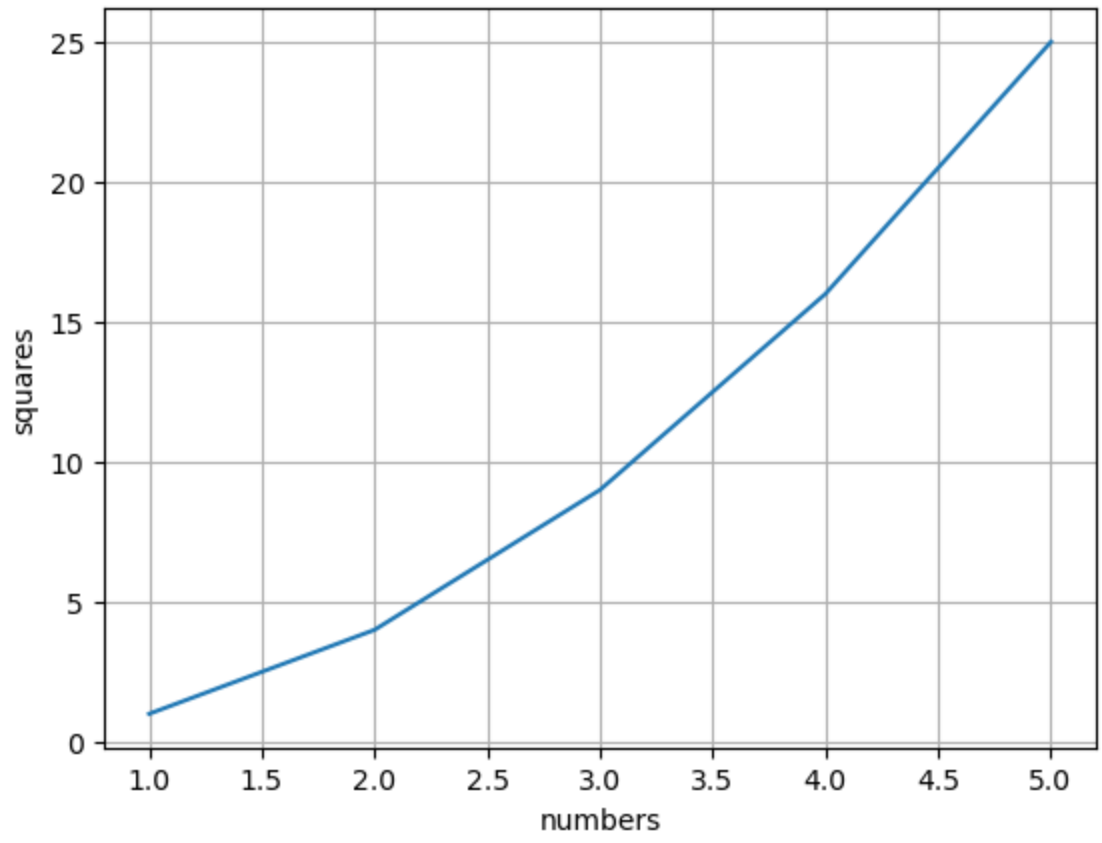


In [8]:

```python
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 40]

# Create a plot
plt.plot(x, y, 'bo-')

# Annotate a specific point
#arrowprops: (Optional) A dictionary of properties for the arrow that connects the text
plt.annotate('Highest Point', xy=(5, 40), xytext=(3, 35),
             arrowprops=dict(facecolor='black', shrink=0.05))

# Add labels and title
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.title('Plot with Annotation')

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

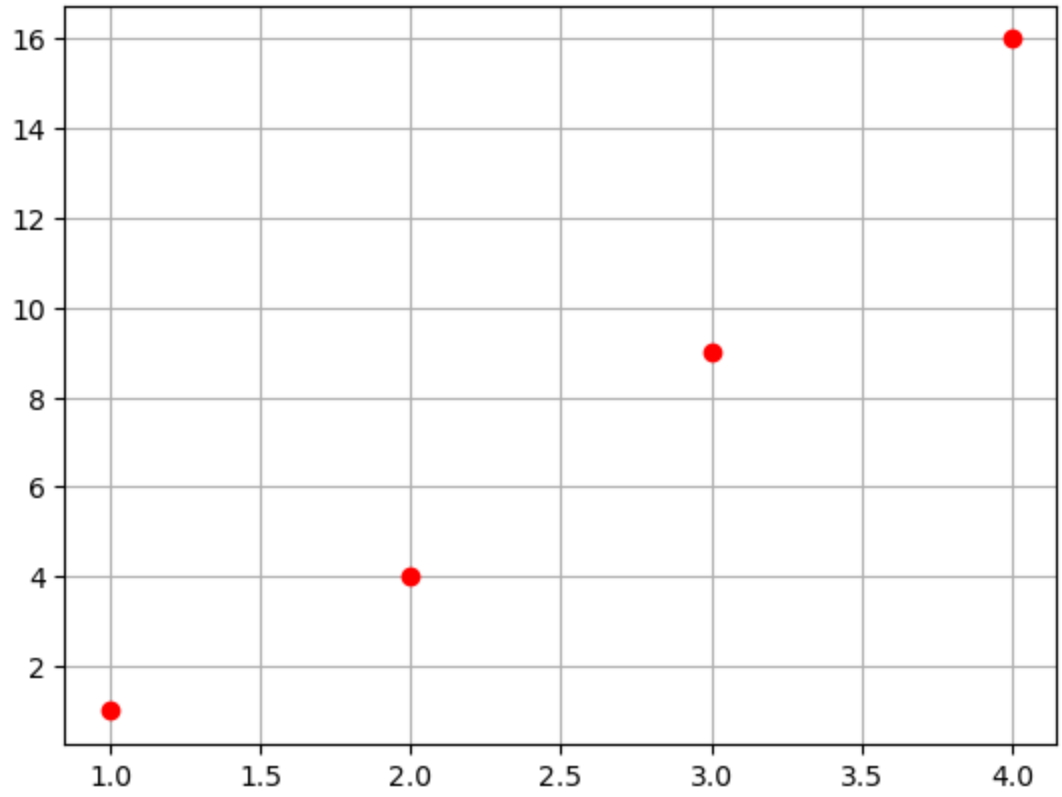


In [9]:
```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4,5], [1, 4, 9, 16,25])
plt.ylabel('squares')
plt.xlabel('numbers')
plt.grid() # grid on
plt.show()
```

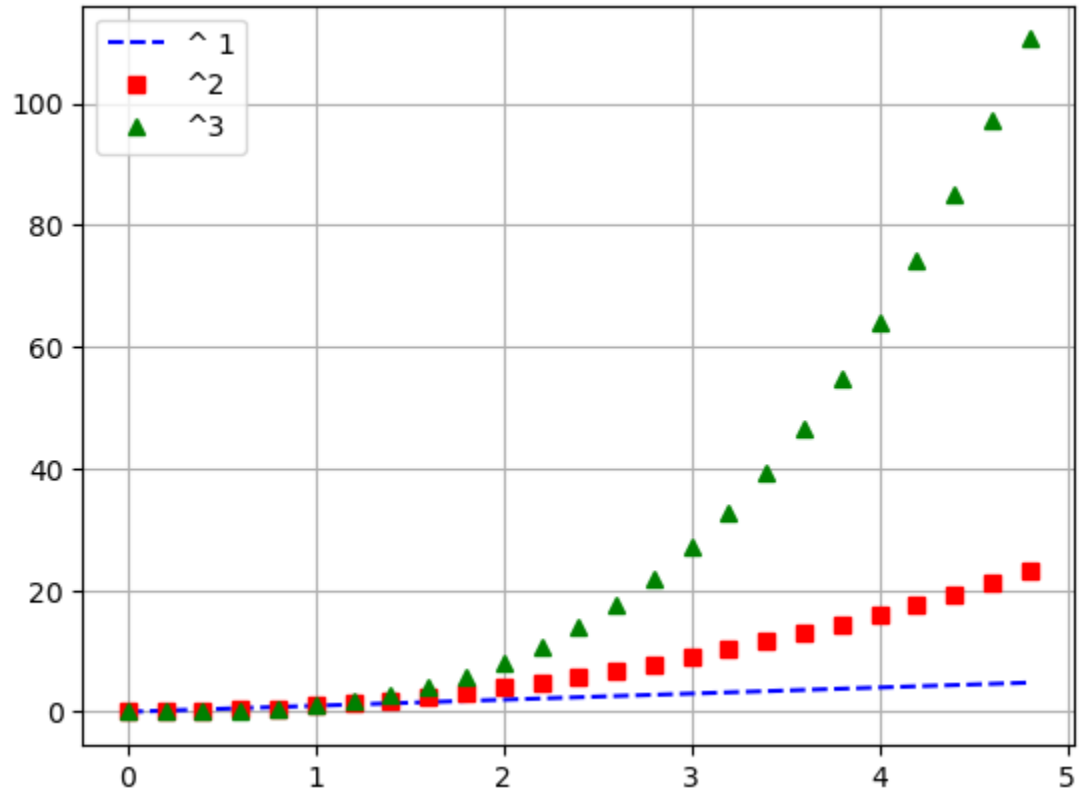


In [10]:
```
# save a plot
```

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
plt.grid()
# to save the graph
plt.savefig(r'C:\Users\lenovo\Downloads\myplot.png')
plt.show()
```



In [11]:
```
import numpy as np
t = np.arange(0., 5., 0.2)
#blue dashes, red squares and green triangles
plt.plot(t, t**1, 'b--', label='^ 1')#   'rs',   'g^')
plt.plot(t,t**2, 'rs', label='^2')
plt.plot(t, t**3, 'g^', label='^3')
plt.grid()
plt.legend() # add legend based on line labels
plt.show()
```



arange in numpy, you specify the start, stop, and step values, such as array = np. arange(start=0, stop=10, step=2) . This function generates a sequence of numbers within the specified range, which can be incredibly useful in data analysis and scientific computing

In [12]:
```
t = np.arange(0., 5., 0.2)
print(t)
```

```
[0.  0.2 0.4 0.6 0.8 1.  1.2 1.4 1.6 1.8 2.  2.2 2.4 2.6 2.8 3.  3.2 3.4
 3.6 3.8 4.  4.2 4.4 4.6 4.8]
```

In [13]:
```
t**1
```

Out[13]:
```
array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. , 2.2, 2.4,
       2.6, 2.8, 3. , 3.2, 3.4, 3.6, 3.8, 4. , 4.2, 4.4, 4.6, 4.8])
```

In [14]:
```
t**2
```

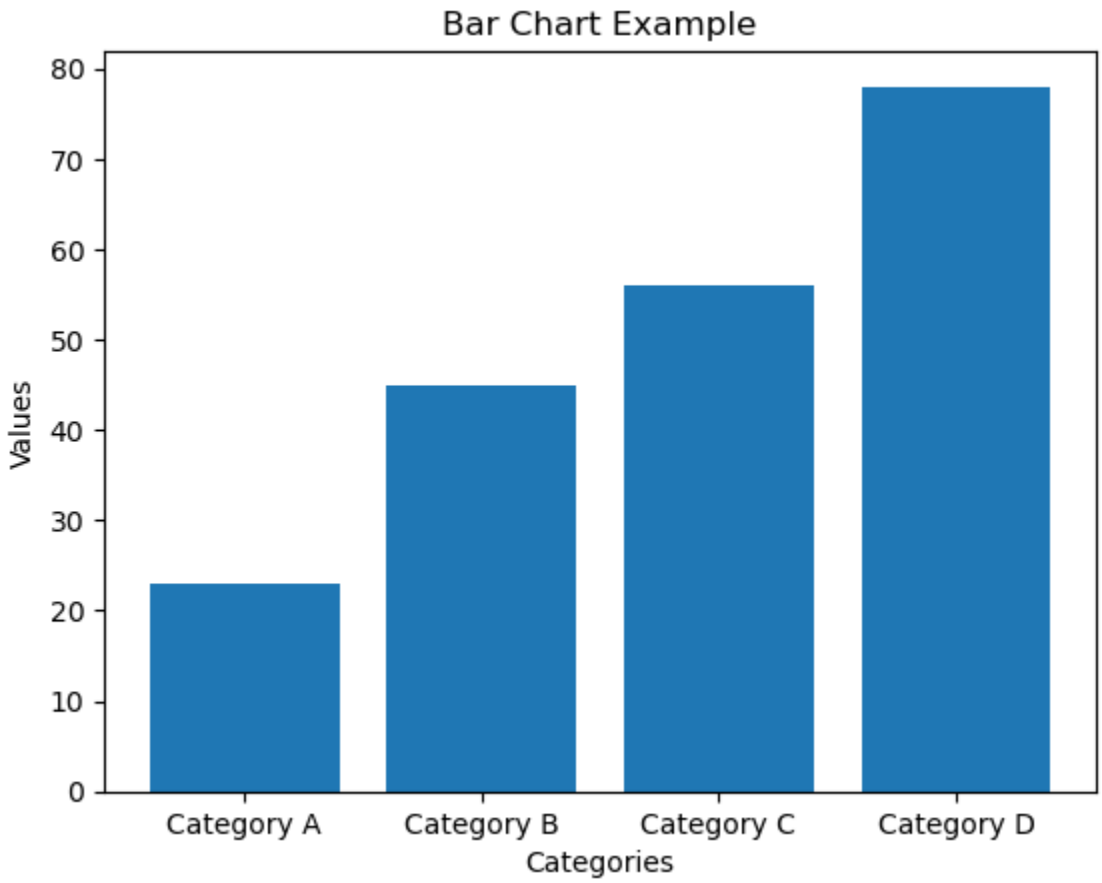
Out[14]:
```
array([ 0.  ,  0.04,  0.16,  0.36,  0.64,  1.  ,  1.44,  1.96,  2.56,
        3.24,  4.  ,  4.84,  5.76,  6.76,  7.84,  9.  , 10.24, 11.56,
       12.96, 14.44, 16.  , 17.64, 19.36, 21.16, 23.04])
```

In [15]:
```
results=t**3
```

Bar Graph

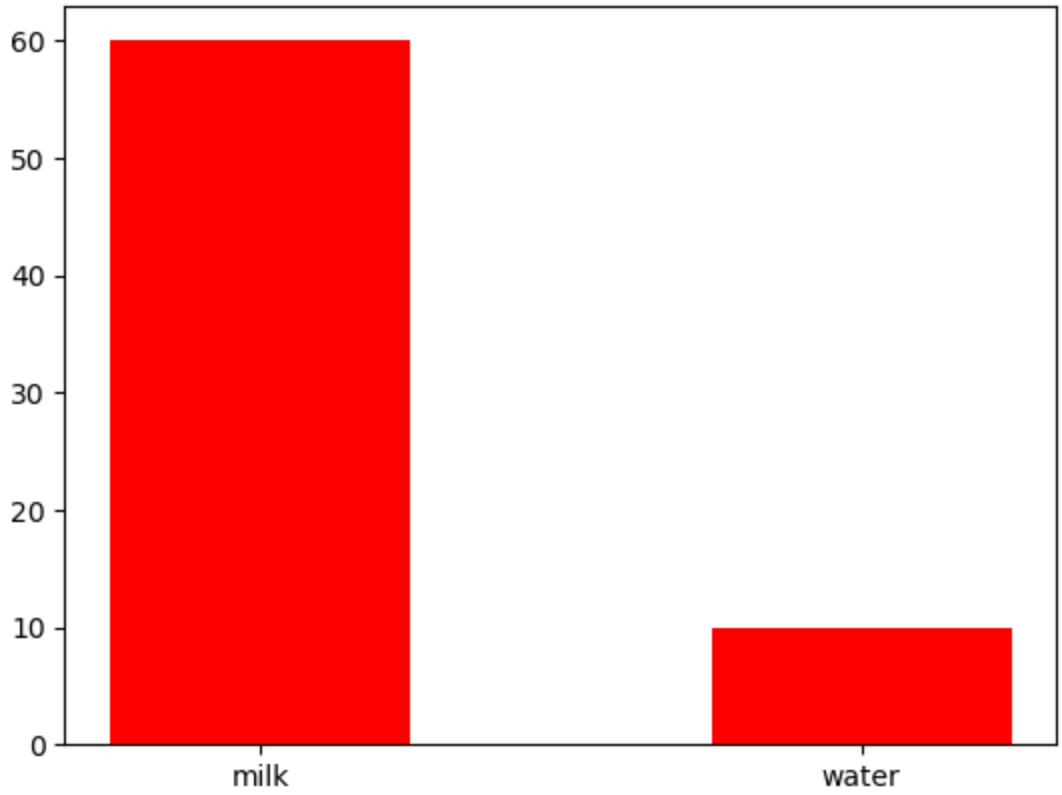
In [16]:
```
# bar plot

import matplotlib.pyplot as plt

# Sample data
categories = ['Category A', 'Category B', 'Category C', 'Category D']
values = [23, 45, 56, 78]

# Create a bar chart
plt.bar(categories, values)
```

```
# Add title and labels
plt.title('Bar Chart Example')
plt.xlabel('Categories')
plt.ylabel('Values')

# Display the chart
plt.show()
```



In [17]:
```
#Plot a bar using matplotlib using a dictionary
data = {'milk': 60, 'water': 10}
names = list(data.keys())
values = list(data.values())
plt.bar(names, values, color = 'r', width=0.5)
```

Out[17]:
```
<BarContainer object of 2 artists>
```



arange in numpy, you specify the start, stop, and step values, such as array = np. arange(start=0, stop=10, step=2) . This function generates a sequence of numbers within the specified range, which can be incredibly useful in data analysis and scientific computing

Importing a new dataset wine-quality-white-and-red.csv

In [18]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [19]:
```
# read dataset

import matplotlib.pyplot as plt

df = pd.read_csv(r'C:\Users\lenovo\Downloads\wine-quality-white-and-red.csv')
```

In [20]:
```
df
```

Out[20]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	qu
0	white	7.0	0.270	0.36	20.7	0.045	45.0	170.0	1.00100	3.00	0.45	8.8	
1	white	6.3	0.300	0.34	1.6	0.049	14.0	132.0	0.99400	3.30	0.49	9.5	
2	white	8.1	0.280	0.40	6.9	0.050	30.0	97.0	0.99510	3.26	0.44	10.1	
3	white	7.2	0.230	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	
4	white	7.2	0.230	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	
...	...	...	...	...	...	...	...	...	...	...	...	...	...

6492	red	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
6493	red	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
6494	red	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
6495	red	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
6496	red	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

6497 rows × 13 columns

Create a histogram

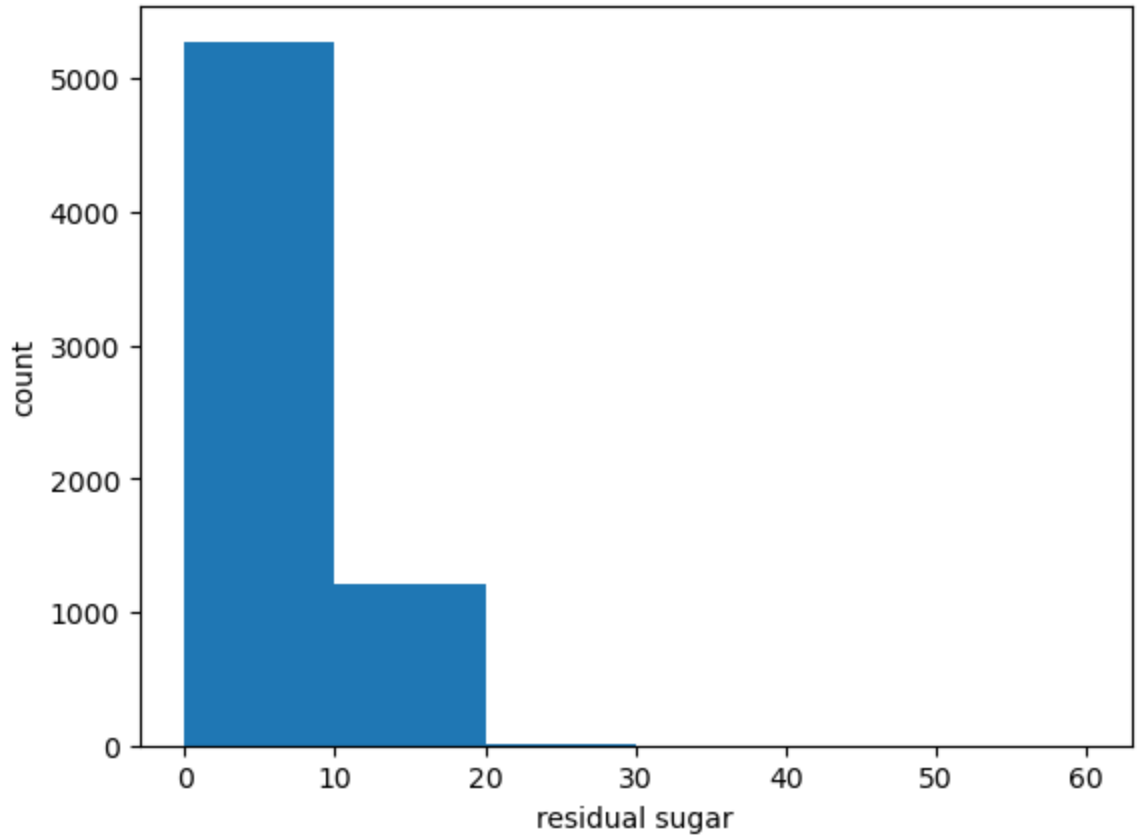
In [21]:
```
# create histogram
bin_edges = np.arange(0, df['residual sugar'].max() + 1, 10)
df['residual sugar'].max()+1
df['residual sugar']
```

Out[21]:
```
0       20.7
1        1.6
2        6.9
3        8.5
4        8.5
        ...
6492     2.0
6493     2.2
6494     2.3
6495     2.0
6496     3.6
Name: residual sugar, Length: 6497, dtype: float64
```

In [22]:
```
bin_edges
```

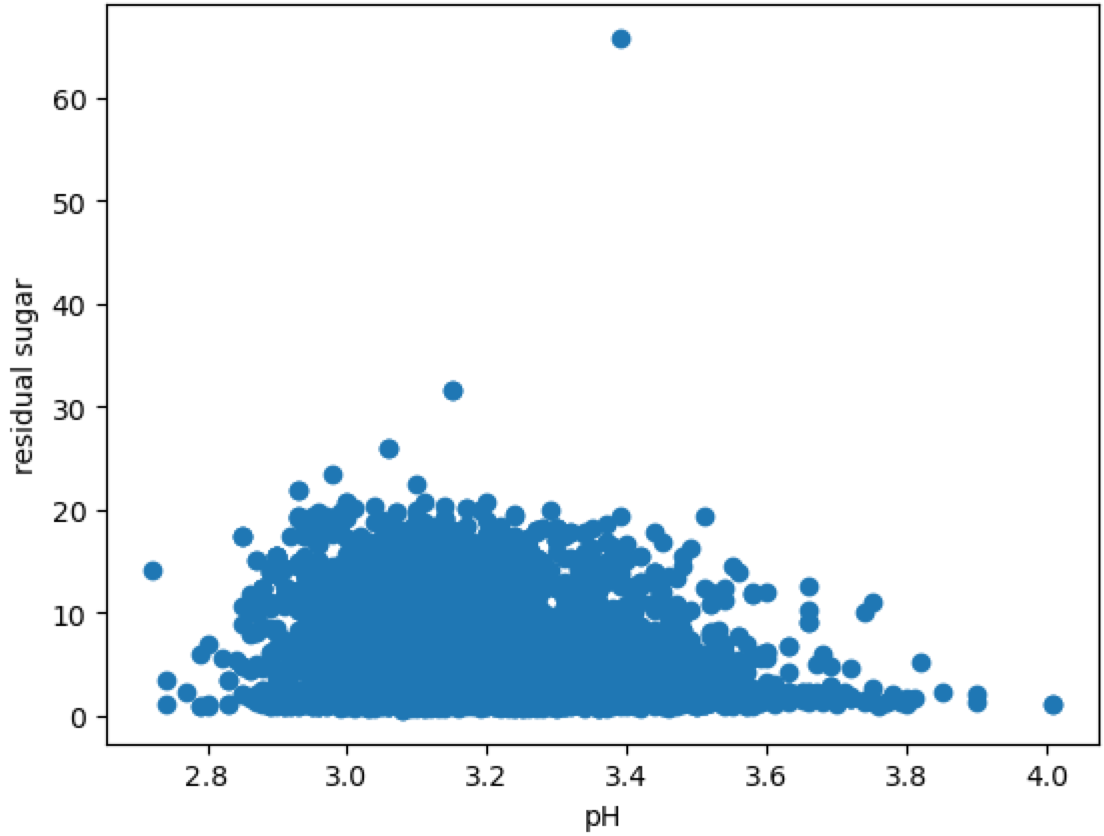
Out[22]:
```
array([ 0., 10., 20., 30., 40., 50., 60.])
```

In [23]:
```
fig = plt.hist(df['residual sugar'], bins=bin_edges)
# add plot labels
plt.xlabel('residual sugar')
plt.ylabel('count')
plt.show()
```



Create scatterplot

In [24]:
```
# create scatterplot
fig = plt.scatter(df['pH'], df['residual sugar'])

# add plot labels
plt.xlabel('pH')
plt.ylabel('residual sugar')
plt.show()
```



In [25]:
```
df.columns
```

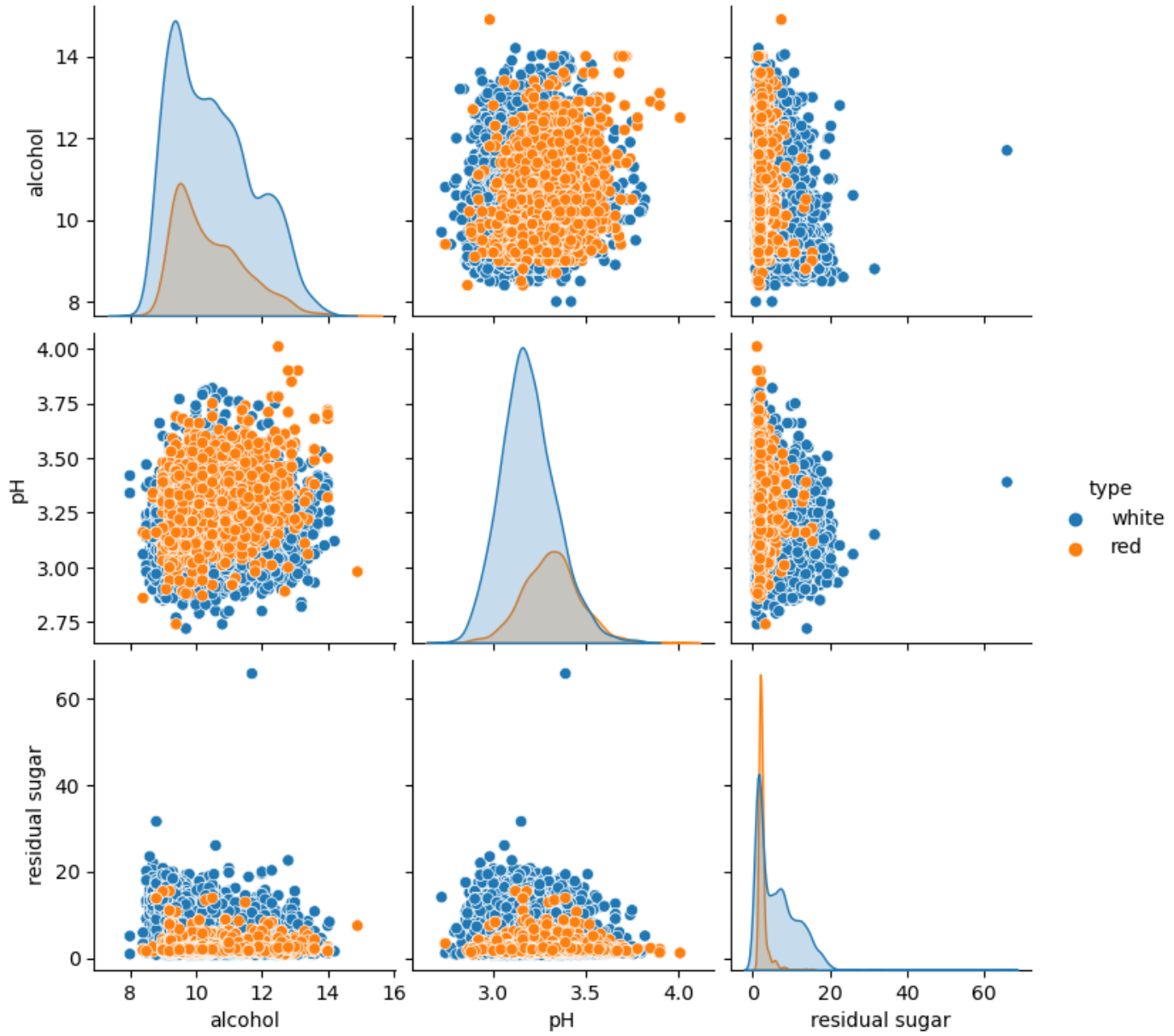
Out[25]:
```
Index(['type', 'fixed acidity', 'volatile acidity', 'citric acid',
       'residual sugar', 'chlorides', 'free sulfur dioxide',
       'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol',
       'quality'],
      dtype='object')
```

Pair plot

A scatterplot matrix is a grid of scatterplots that allows us to see how different pairs of variables are related to each other. You can use the hue parameter when creating pairplots in seaborn to color plot aspects based on the values of a specific variable

In [26]:
```
# create scatterplot matrix
import seaborn as sns
fig = sns.pairplot(data=df[['alcohol', 'pH', 'residual sugar', 'type']],
                   hue='type')
plt.show()
```

```
C:\Users\lenovo\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use
_inf_as_na option is deprecated and will be removed in a future version. Convert inf val
ues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\lenovo\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use
_inf_as_na option is deprecated and will be removed in a future version. Convert inf val
ues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\lenovo\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use
_inf_as_na option is deprecated and will be removed in a future version. Convert inf val
ues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Bee Swarm Plot

A beeswarm plot or swarmplot is a type of data visualization that displays individual data points in a way that they don't overlap, resulting in a "swarming" effect that resembles a swarm of bees.

This chart type helps in revealing the distribution of the data along a numeric variable, highlighting the density and variation of the data more effectively than traditional scatter plots or box plots.

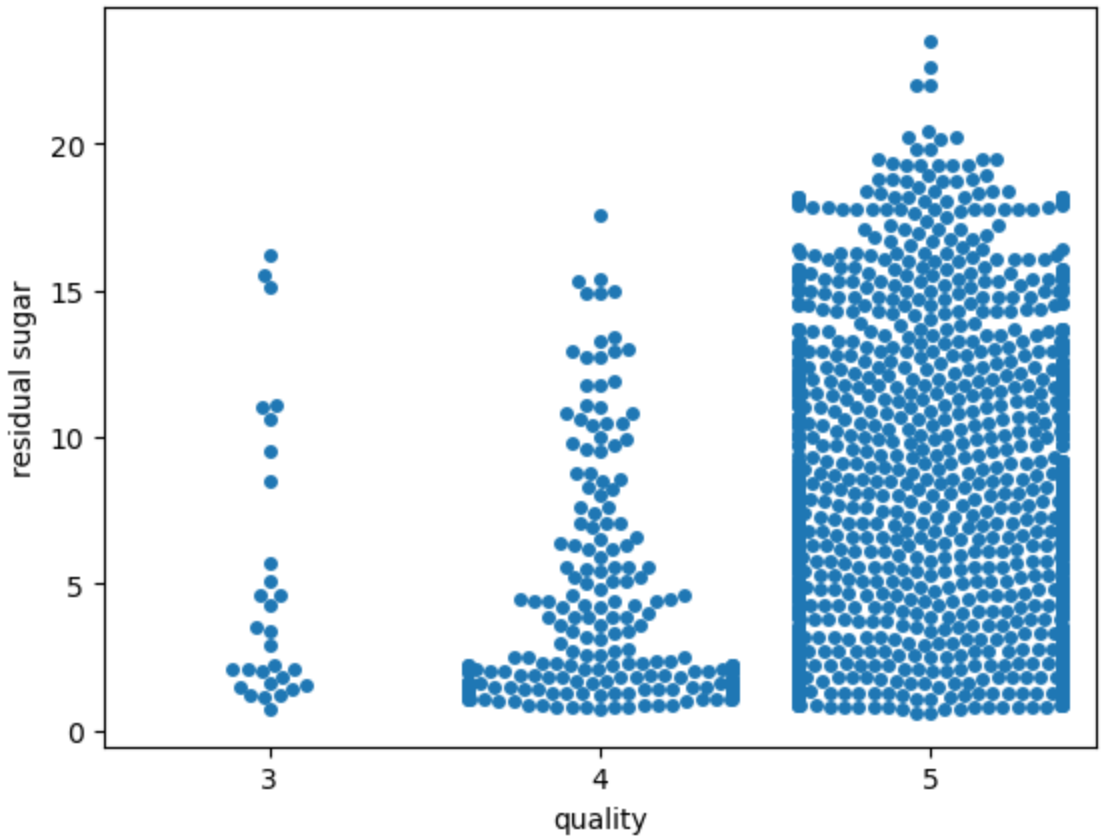
In [27]:
```
df['quality'].unique()
```

Out[27]:
```
array([6, 5, 7, 8, 4, 3, 9], dtype=int64)
```

In [28]:
```
# Bee Swarm Plot
#useful for small datasets but can be slow on large datasets
# create bee swarm plot
# create bee swarm plot
sns.swarmplot(x='quality', y='residual sugar',
              data=df[df['quality'] < 6])
plt.show()
```

```
C:\Users\lenovo\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use
```

```
_inf_as_na option is deprecated and will be removed in a future version. Convert inf val
ues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\lenovo\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use
_inf_as_na option is deprecated and will be removed in a future version. Convert inf val
ues to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\lenovo\anaconda3\Lib\site-packages\seaborn\categorical.py:3544: UserWarning: 1
5.3% of the points cannot be placed; you may want to decrease the size of the markers or
use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\lenovo\anaconda3\Lib\site-packages\seaborn\categorical.py:3544: UserWarning: 6
0.7% of the points cannot be placed; you may want to decrease the size of the markers or
use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\lenovo\anaconda3\Lib\site-packages\seaborn\categorical.py:3544: UserWarning: 2
4.5% of the points cannot be placed; you may want to decrease the size of the markers or
use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\lenovo\anaconda3\Lib\site-packages\seaborn\categorical.py:3544: UserWarning: 6
9.4% of the points cannot be placed; you may want to decrease the size of the markers or
use stripplot.
  warnings.warn(msg, UserWarning)
```

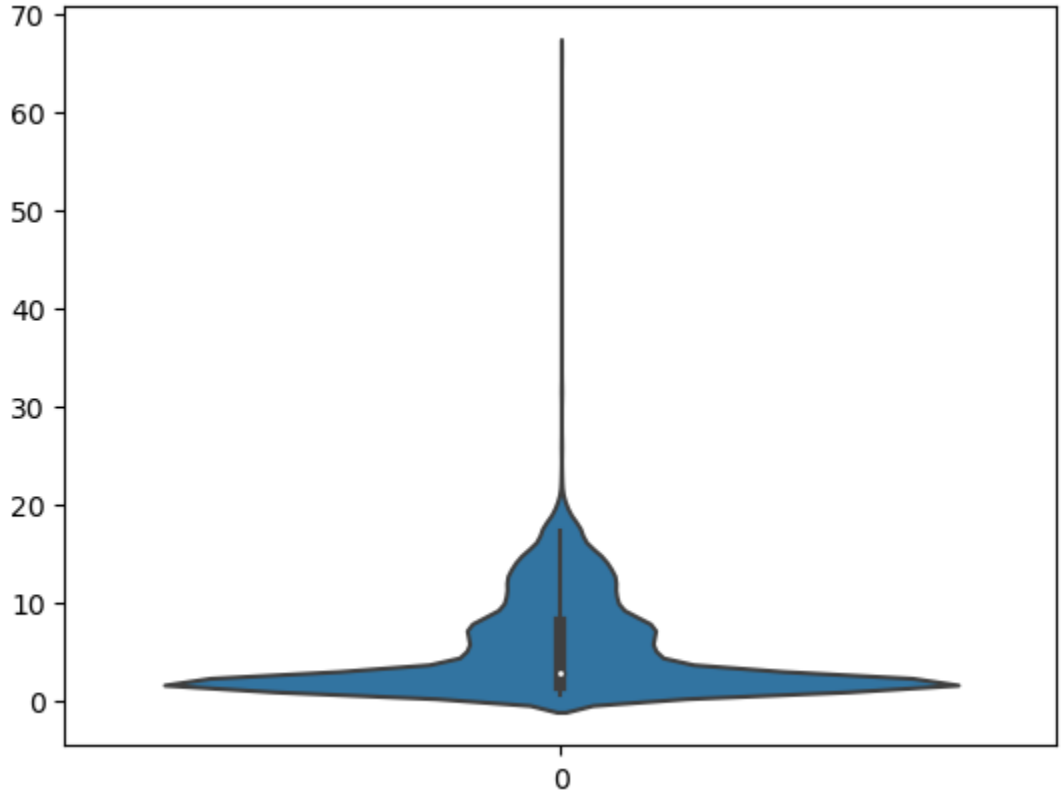


Violin Plots

A violin plot is a hybrid of a box plot and a kernel density plot, which shows peaks in the data. It is used to visualize the distribution of numerical data. Unlike a box plot that can only show summary statistics, violin plots depict summary statistics and the density of each variable.

In [29]:
```
sns.violinplot(df['residual sugar'])
```

Out[29]:
```
<Axes: >
```



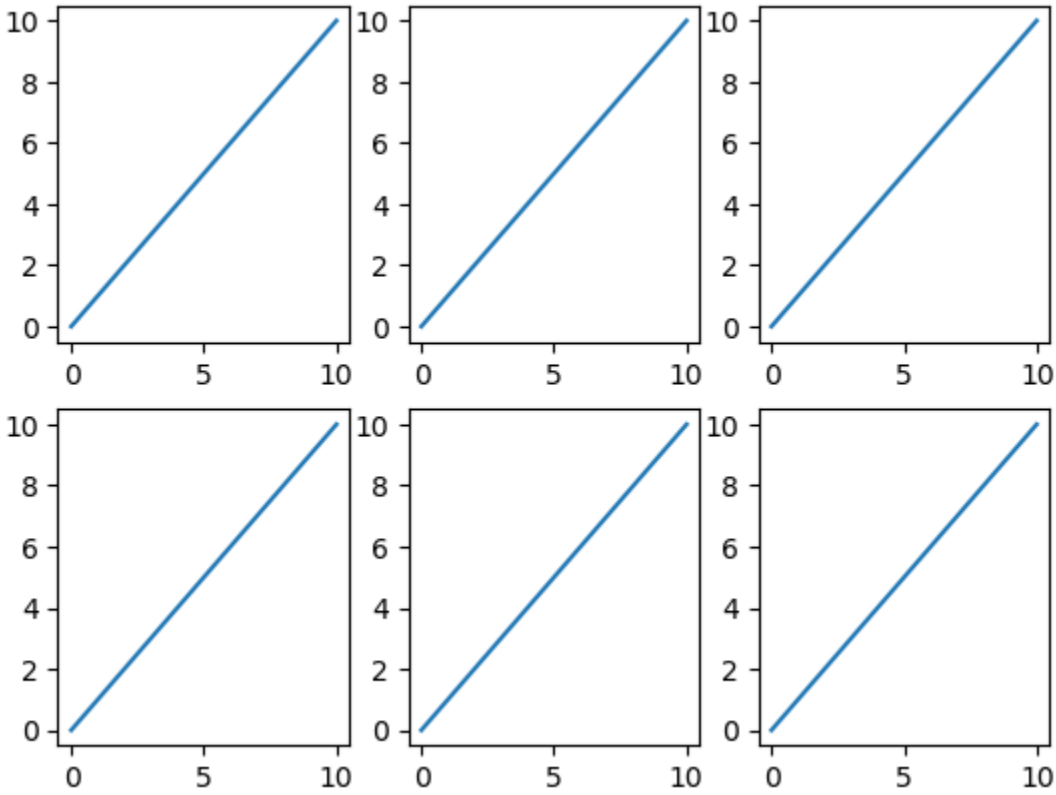
Subplots

Subplots: With the subplot() function you can draw multiple plots in one figure

The subplot() function takes three arguments that describes the layout of the figure.

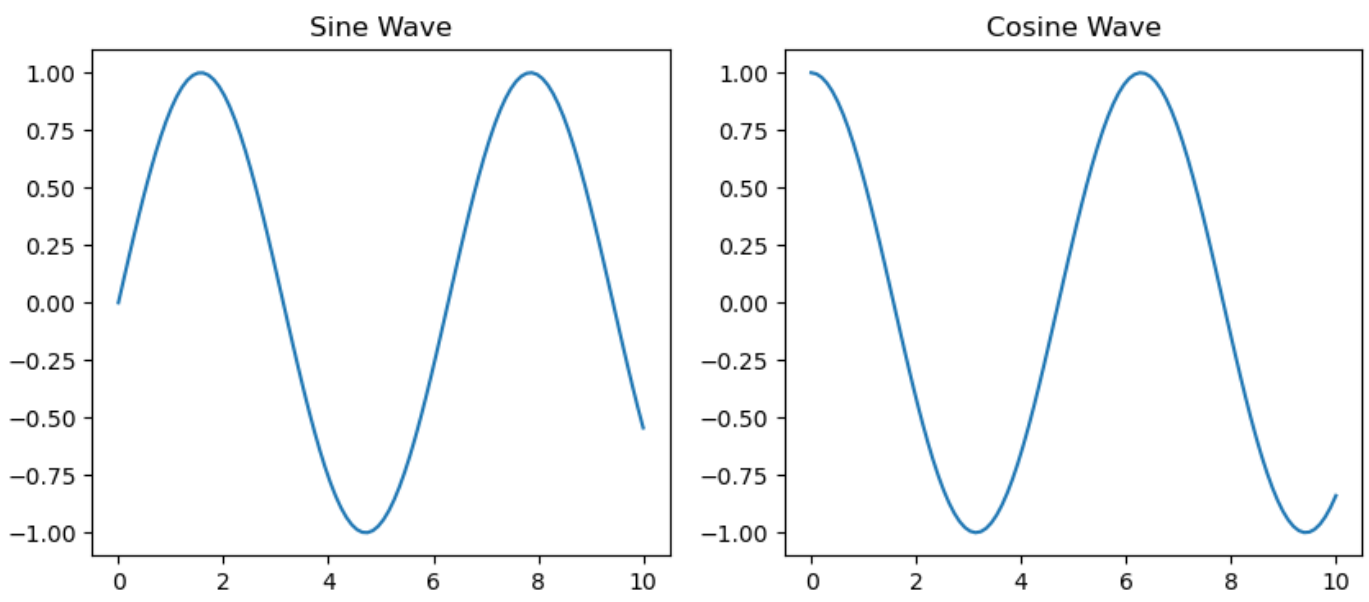
The layout is organized in rows and columns, which are represented by the first and second argument. The third argument represents the index of the current plot.

In [30]:
```
x = range(11)
y = range(11)

fig, ax = plt.subplots(nrows=2, ncols=3)

for row in ax:
    for col in row:
        col.plot(x, y)

plt.show()
```



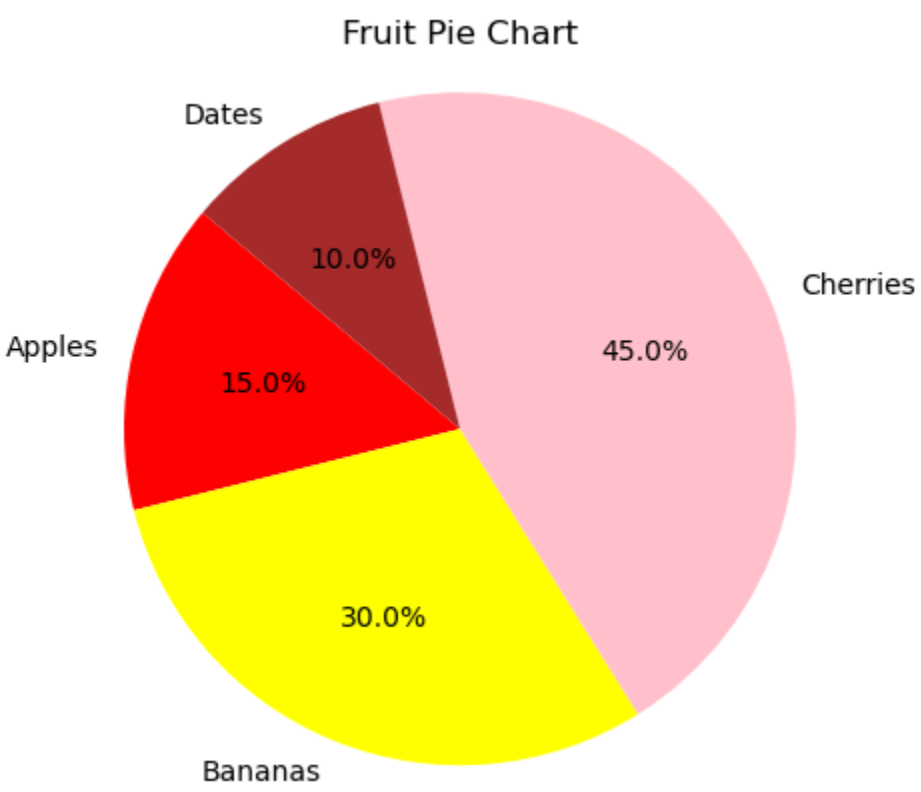
In [31]:

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

# Data for plotting
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

# Create a 1x2 grid of subplots
fig, axs = plt.subplots(1, 2, figsize=(10, 4))

# Plot on the first subplot
axs[0].plot(x, y1)
axs[0].set_title('Sine Wave')

# Plot on the second subplot
axs[1].plot(x, y2)
axs[1].set_title('Cosine Wave')

# Display the plots
plt.show()
```



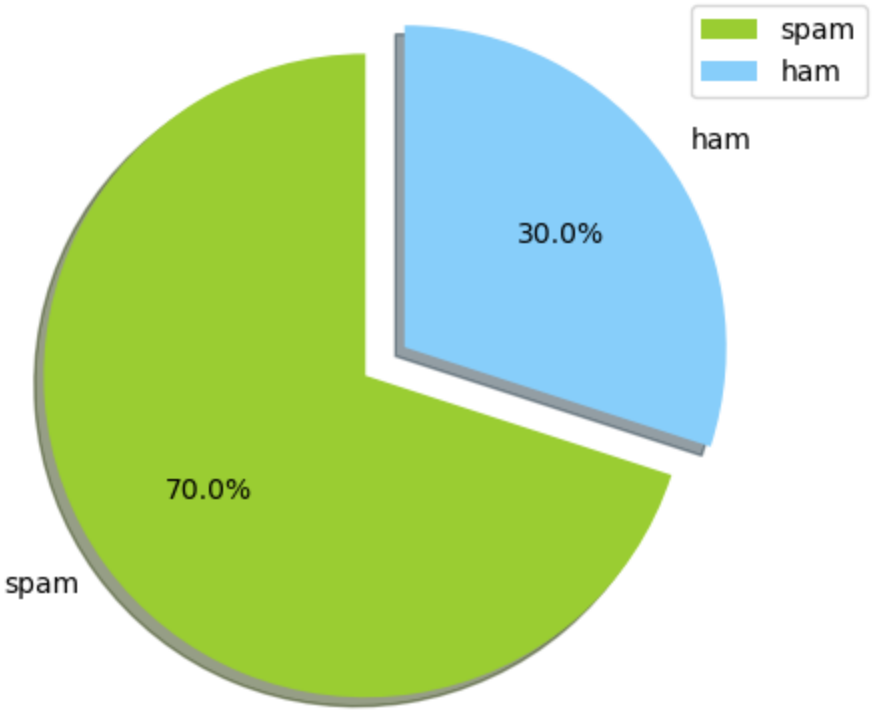
Pie Chart

In [32]:
```python
import matplotlib.pyplot as plt

# Data to plot
labels = ['Apples', 'Bananas', 'Cherries', 'Dates']
sizes = [15, 30, 45, 10]  # Sizes represent the proportions
colors = ['red', 'yellow', 'pink', 'brown']  # Colors for each slice

# Create the pie chart
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)

# Equal aspect ratio ensures that pie is drawn as a circle.
plt.axis('equal')

# Add a title
plt.title('Fruit Pie Chart')

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



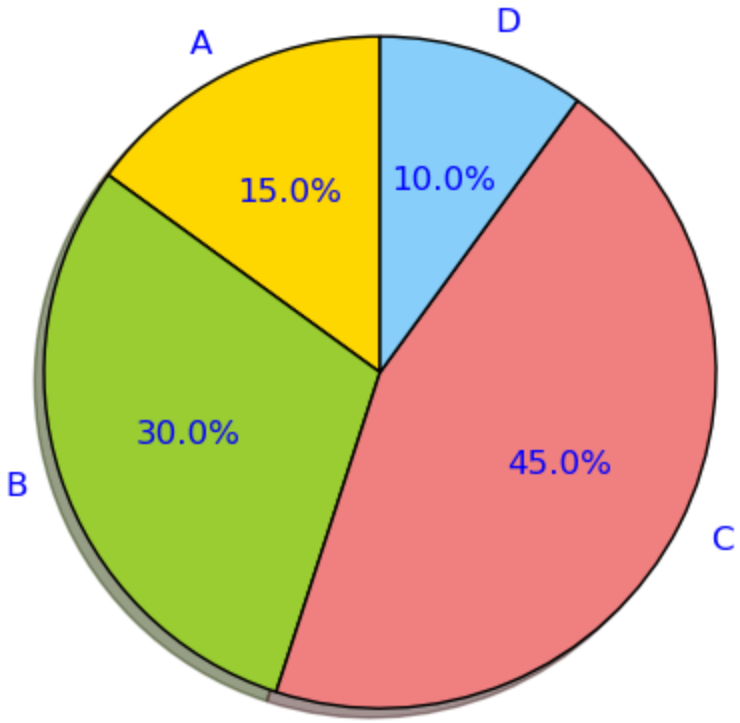
In [33]:

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

plt.pie(
    (7,3),
    labels=('spam','ham'),
    shadow=True,
    colors=('yellowgreen', 'lightskyblue'),
    explode=(0,0.15), # space between slices
    startangle=90,    # rotate conter-clockwise by 90 degrees
    autopct='%1.1f%%',# display fraction as percentage
    )
plt.legend()
plt.axis('equal')     # plot pyplot as circle

plt.show()
```

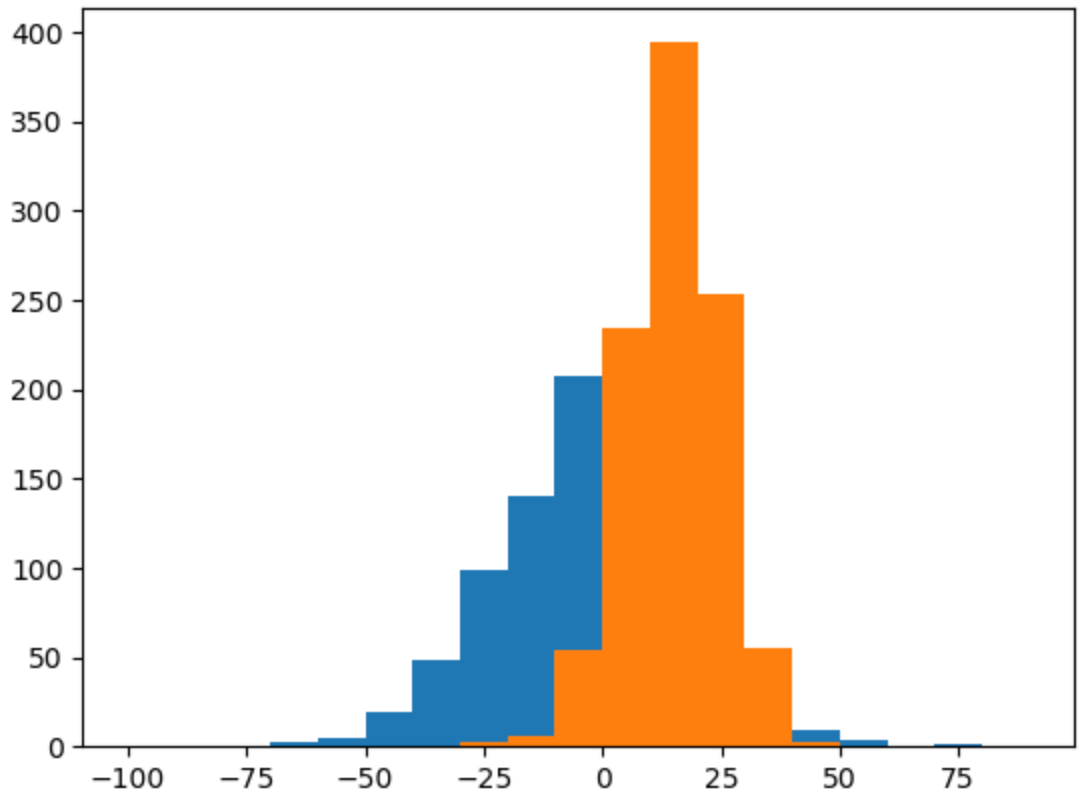


In [34]:

```
import matplotlib.pyplot as plt

# Data to plot
labels = ['A', 'B', 'C', 'D']
sizes = [15, 30, 45, 10]
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']
wedgeprops = {'edgecolor': 'black', 'linewidth': 1}
textprops = {'fontsize': 12, 'color': 'blue'}

# Create a pie chart with custom properties
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=

# Equal aspect ratio ensures that the pie is drawn as a circle
plt.axis('equal')

# Display the chart
plt.show()
```



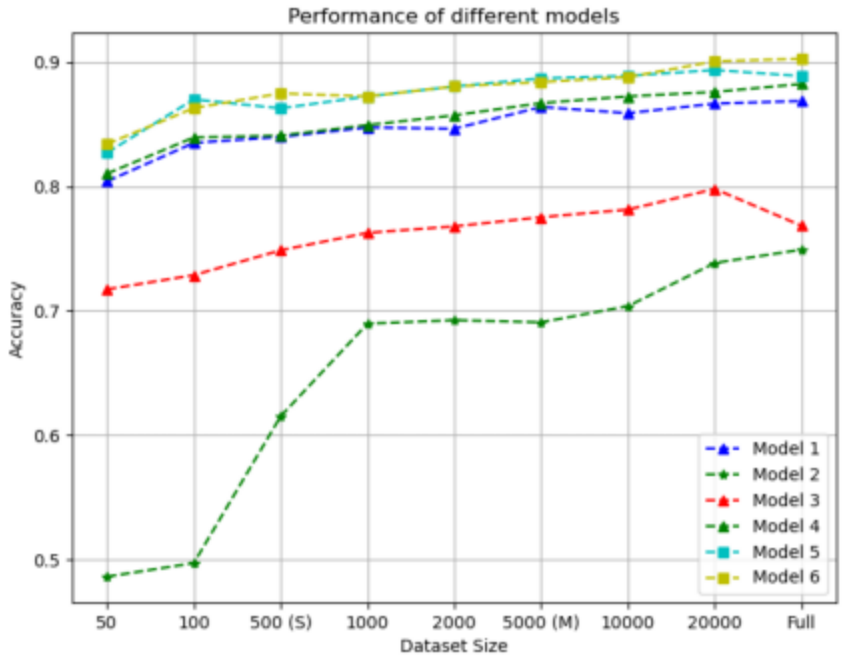
In [35]:
```
rng = np.random.RandomState(123)
x1 = rng.normal(0, 20, 1000)
x2 = rng.normal(15, 10, 1000)

# fixed bin size
bins = np.arange(-100, 100, 10) # fixed bin size

plt.hist(x1, bins=bins)
plt.hist(x2, bins=bins)
plt.show()
```



Assignment

Make the below graph, using the attached csv file in your mail (model_data.csv). No alteration to the .csv should be doen.

In [36]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread('C:/Users/lenovo/Downloads/mymodel.png')
plt.imshow(img)
plt.axis('off')  # Hide axes if not needed
plt.show()
```



In [ ]: