Matplotlib library

The foundational Python data visualization library.

Library website: https://matplotlib.org/

Documentation: https://matplotlib.org/stable/users/index.html

Installation: `pip install matplotlib`

Importing matplotlib library

In [1]:
```
import matplotlib
matplotlib.__version__
```

Out[1]:
```
'3.8.0'
```

In [3]:
```
import matplotlib.pyplot as plt
import numpy as np
```

Creating a drawing object

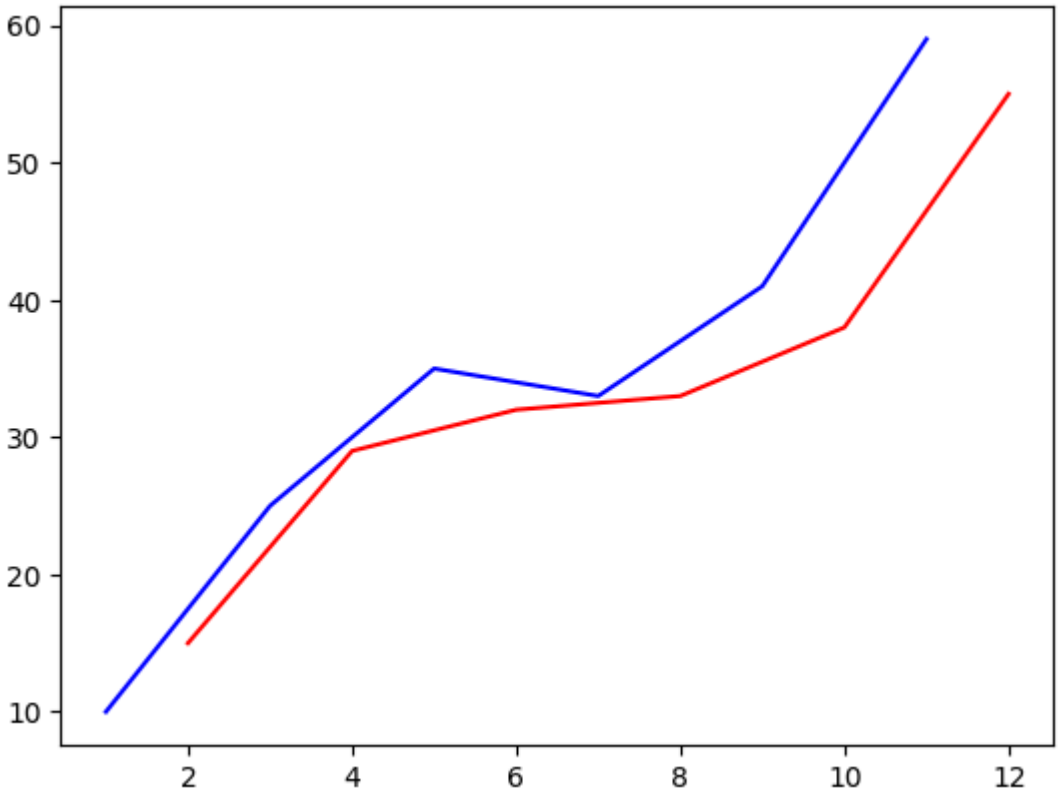
In [4]:
```
fig = plt.figure()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

Common elements for charts

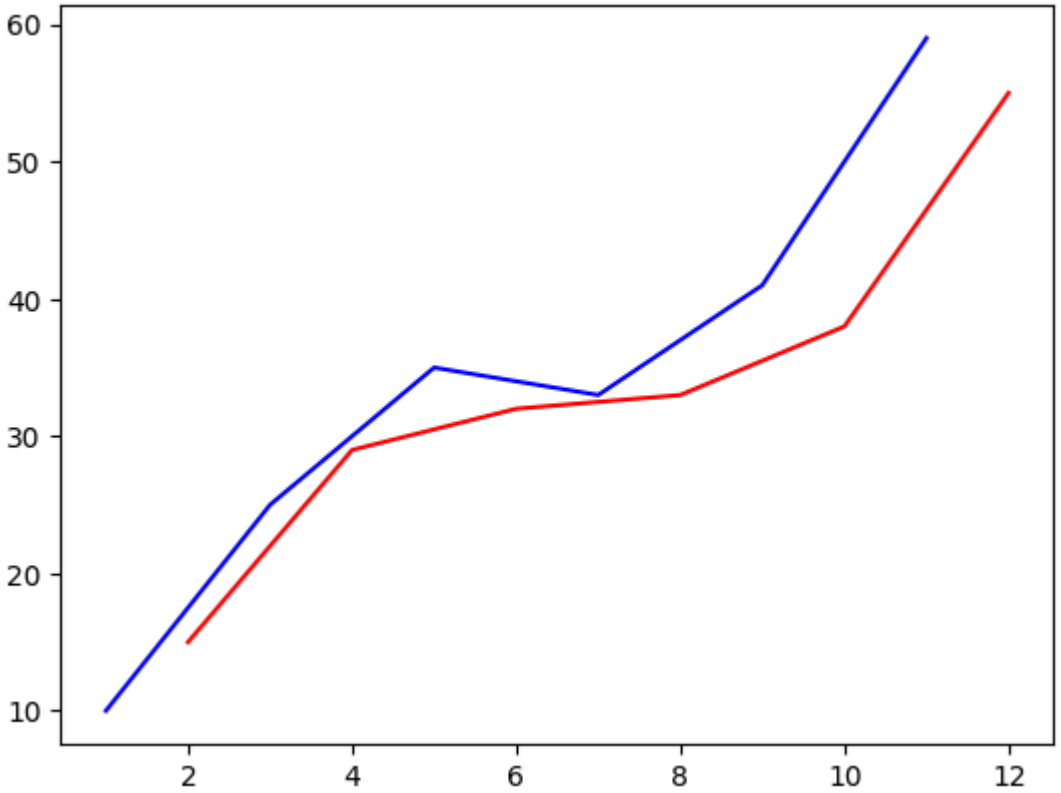
In [6]:
```
x = np.arange(start=1, stop=13, step=2)
y = [10,25,35,33,41,59]

_ = plt.plot(x, y, color='blue')

x = np.arange(start=2, stop=14, step=2)
y = [15,29,32,33,38,55]
```

```
_ = plt.plot(x, y, color='red')
```



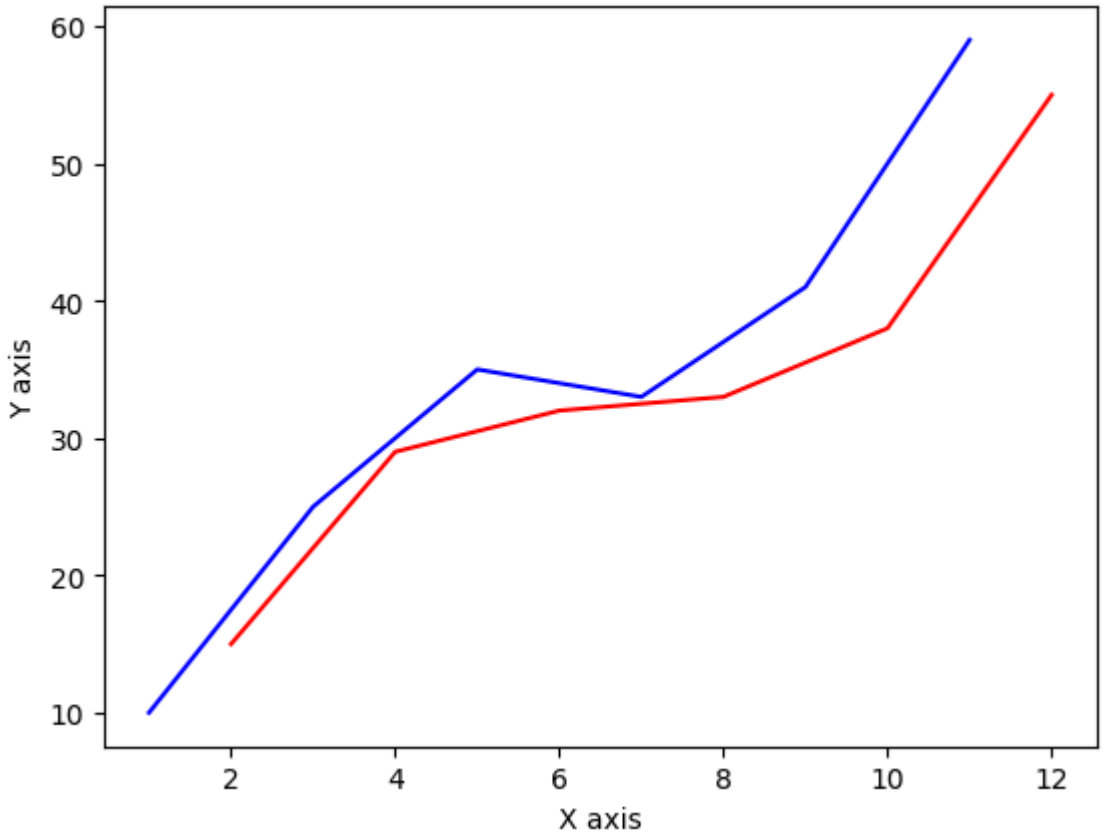
Data labels

In [8]:
```
x = np.arange(start=1, stop=13, step=2)
y = [10,25,35,33,41,59]

plt.plot(x, y, label='Series1', color='blue')

x = np.arange(start=2, stop=14, step=2)
y = [15,29,32,33,38,55]

_ = plt.plot(x, y, label='Series2', color='red')
```



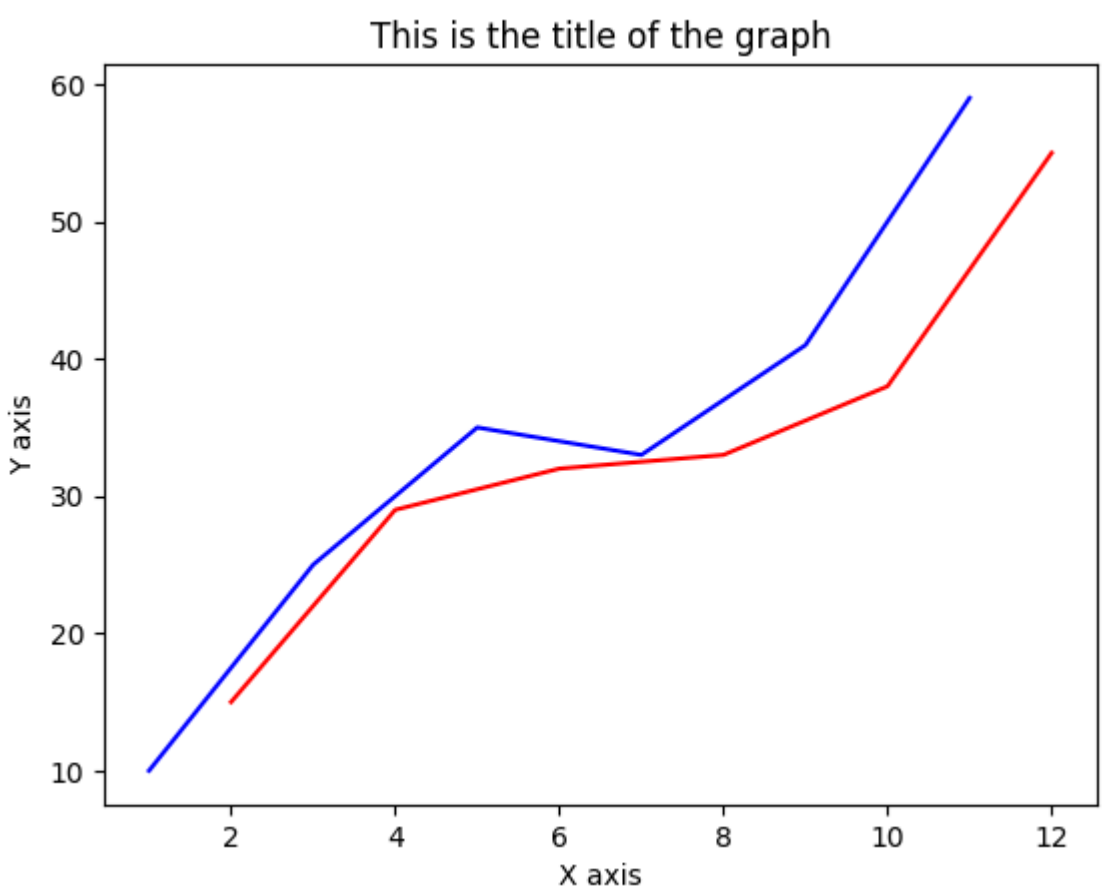
Axle labels

In [9]:
```
x = np.arange(start=1, stop=13, step=2)
y = [10,25,35,33,41,59]

plt.plot(x, y,label='Series1', color='blue')

x = np.arange(start=2, stop=14, step=2)
y = [15,29,32,33,38,55]

plt.plot(x, y, label='Series2', color='red')

plt.xlabel("X axis")
_ = plt.ylabel("Y axis")
```



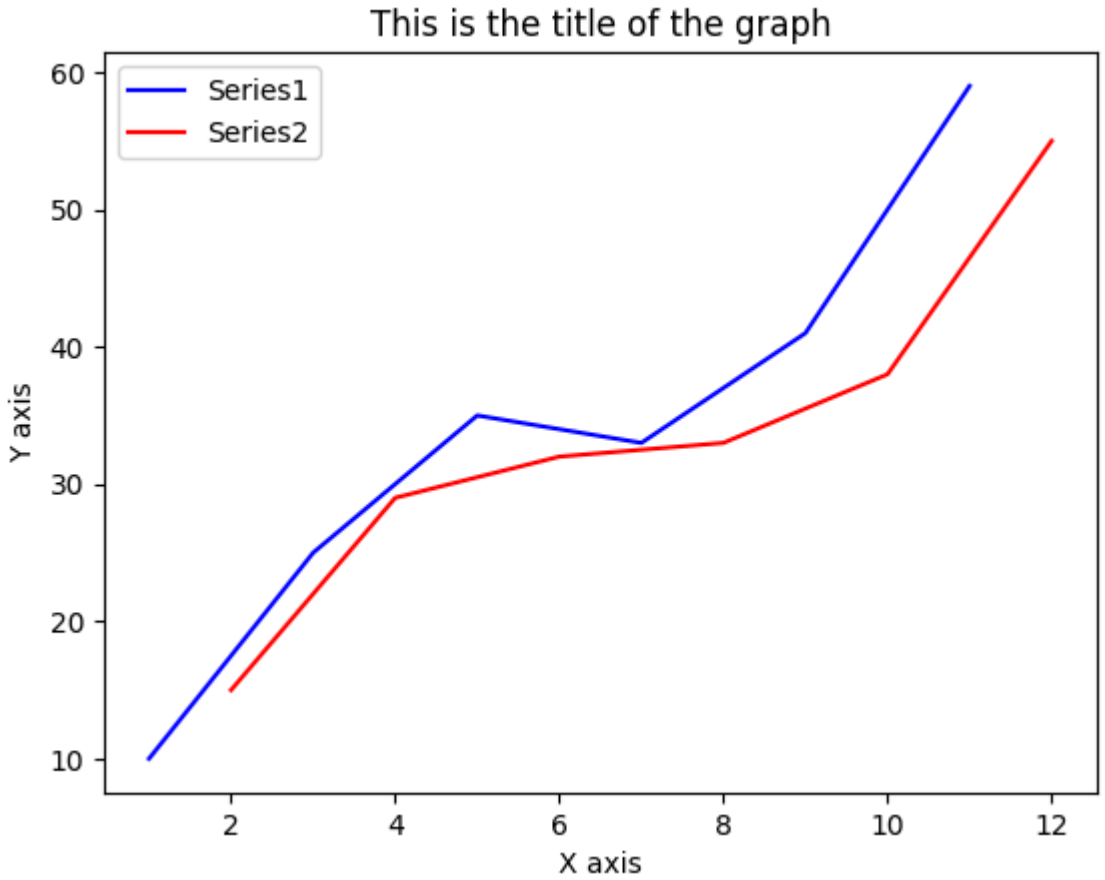
Title of graph

In [10]:
```
x = np.arange(start=1, stop=13, step=2)
y = [10,25,35,33,41,59]

plt.plot(x, y,label='Series1', color='blue')

x = np.arange(start=2, stop=14, step=2)
y = [15,29,32,33,38,55]

plt.plot(x, y, label='Series2', color='red')

plt.xlabel("X axis")
plt.ylabel("Y axis")

_ = plt.title("This is the title of the graph")
```



Legend

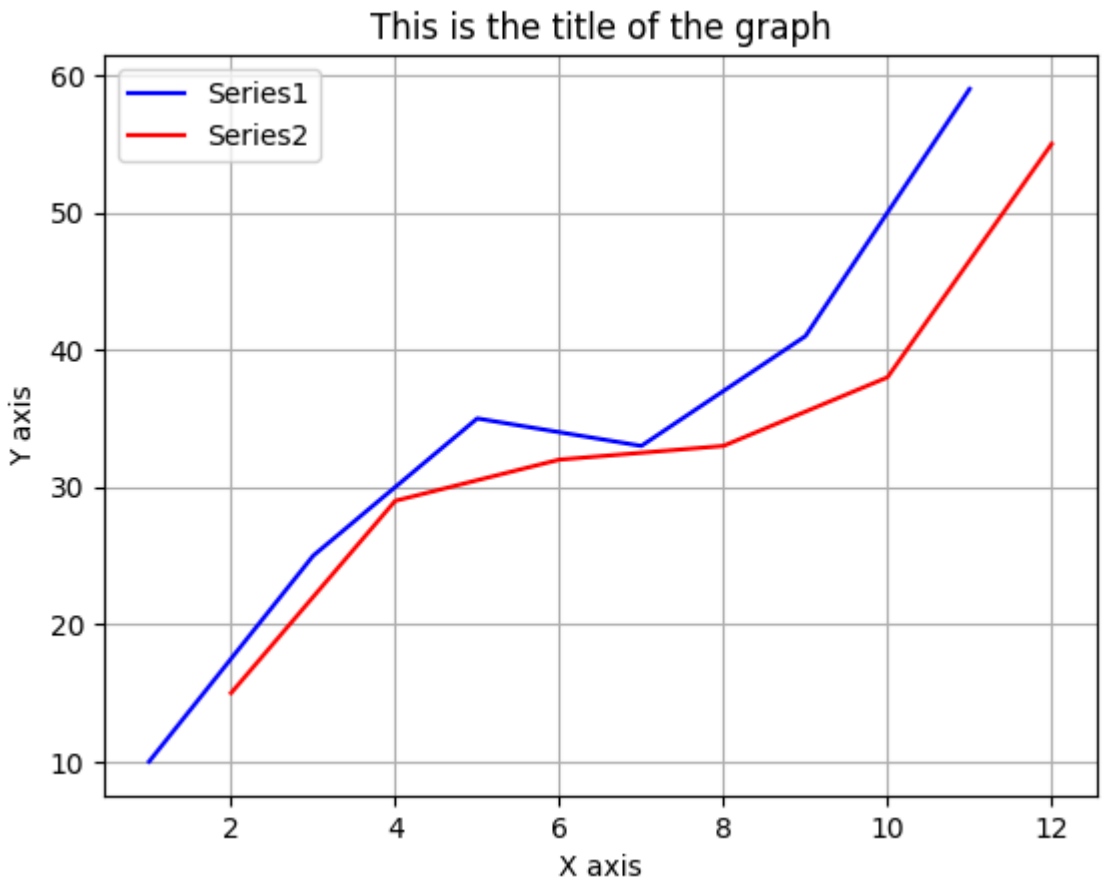
In [11]:
```
x = np.arange(start=1, stop=13, step=2)
y = [10,25,35,33,41,59]

plt.plot(x, y,label='Series1', color='blue')

x = np.arange(start=2, stop=14, step=2)
y = [15,29,32,33,38,55]

plt.plot(x, y, label='Series2', color='red')

plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("This is the title of the graph")

_ = plt.legend()
```



Grid

In [13]:
```
x = np.arange(start=1, stop=13, step=2)
y = [10,25,35,33,41,59]

plt.plot(x, y,label='Series1', color='blue')

x = np.arange(start=2, stop=14, step=2)
y = [15,29,32,33,38,55]

plt.plot(x, y, label='Series2', color='red')

plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("This is the title of the graph")
plt.legend()
```

```
_ = plt.grid()
```



Basic data graphs

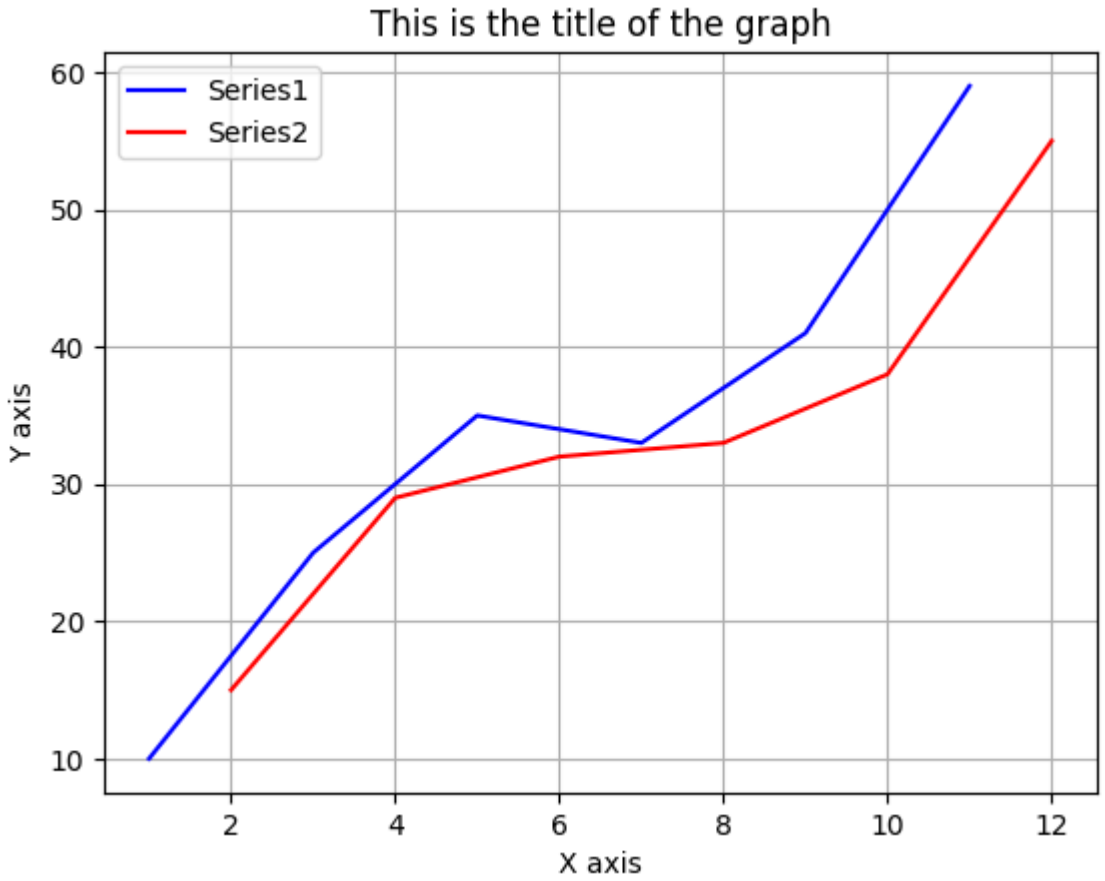
Line graph

A line graph is a graph that has a series of data points connected by a line.

The line graph is the default graph in the `matplotlib` library and is created using the `plot()` function.

In [14]:
```
x = np.arange(start=1, stop=13, step=2)
y = [10,25,35,33,41,59]

plt.plot(x, y,label='Series1', color='blue')

x = np.arange(start=2, stop=14, step=2)
```

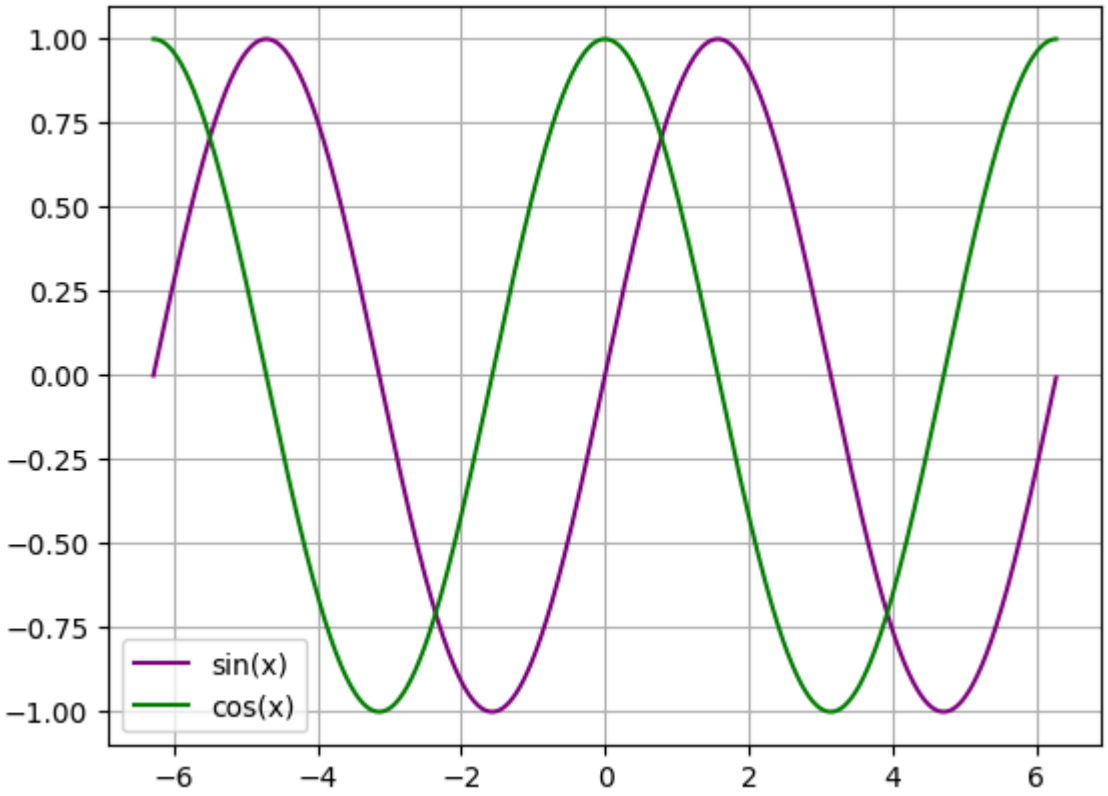
```
y = [15,29,32,33,38,55]

plt.plot(x, y, label='Series2', color='red')

plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("This is the title of the graph")
plt.legend()

_ = plt.grid()
```



In [15]:

```
x = np.arange(-2*np.pi, 2*np.pi, 0.01)
y = np.sin(x)

plt.plot(x, y,label='sin(x)', color='purple')

y= np.cos(x)
plt.plot(x, y,label='cos(x)', color='green')
```

```
plt.legend()
_ = plt.grid()
```



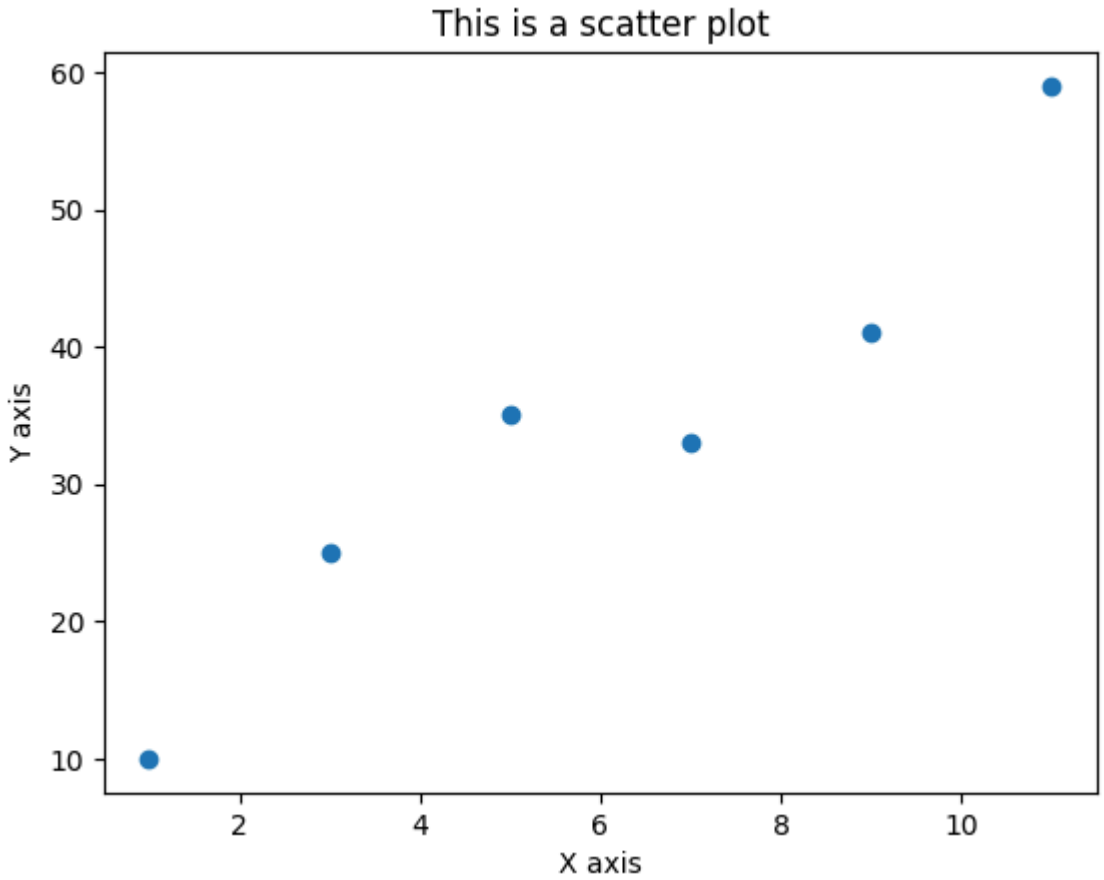
Scatter plot

Scatter plots plot data points using Cartesian coordinates to show numeric data values.

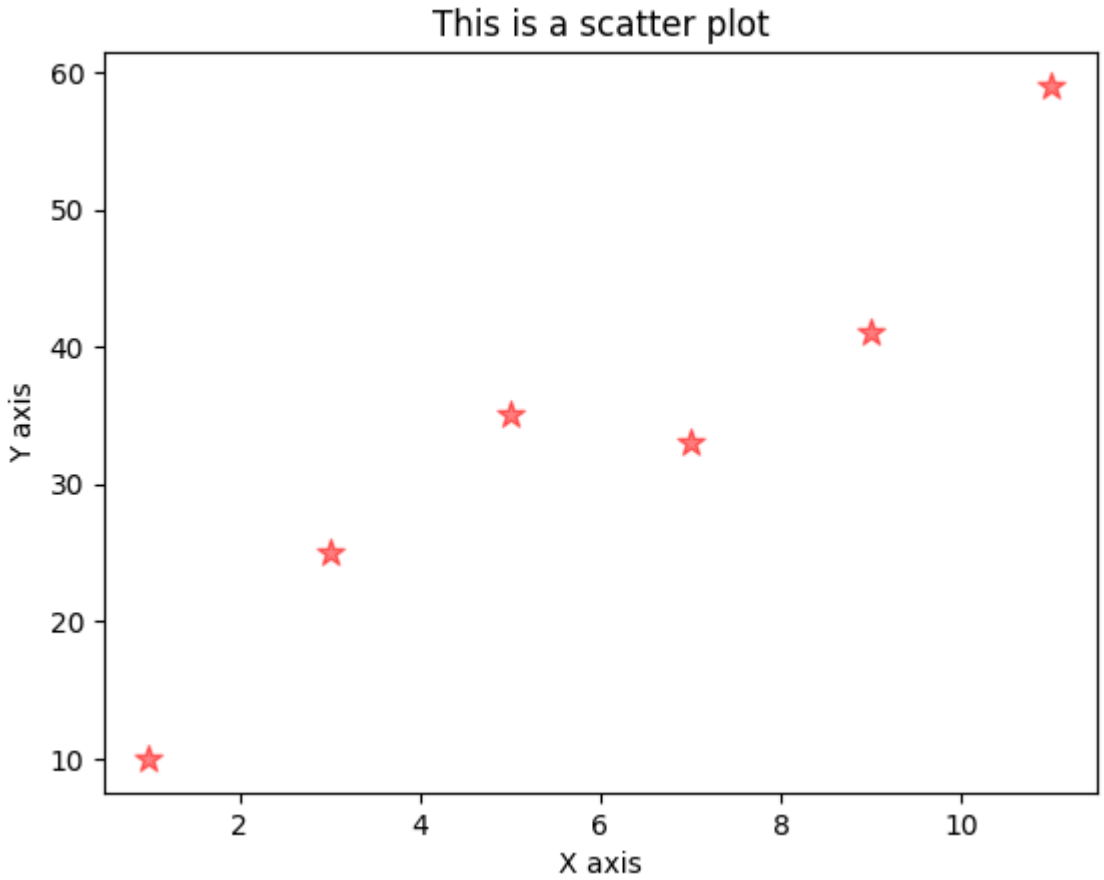
They can also represent the relationship between two numerical values.

To create a scatter plot in the Matplotlib library, use the `scatter()` function.

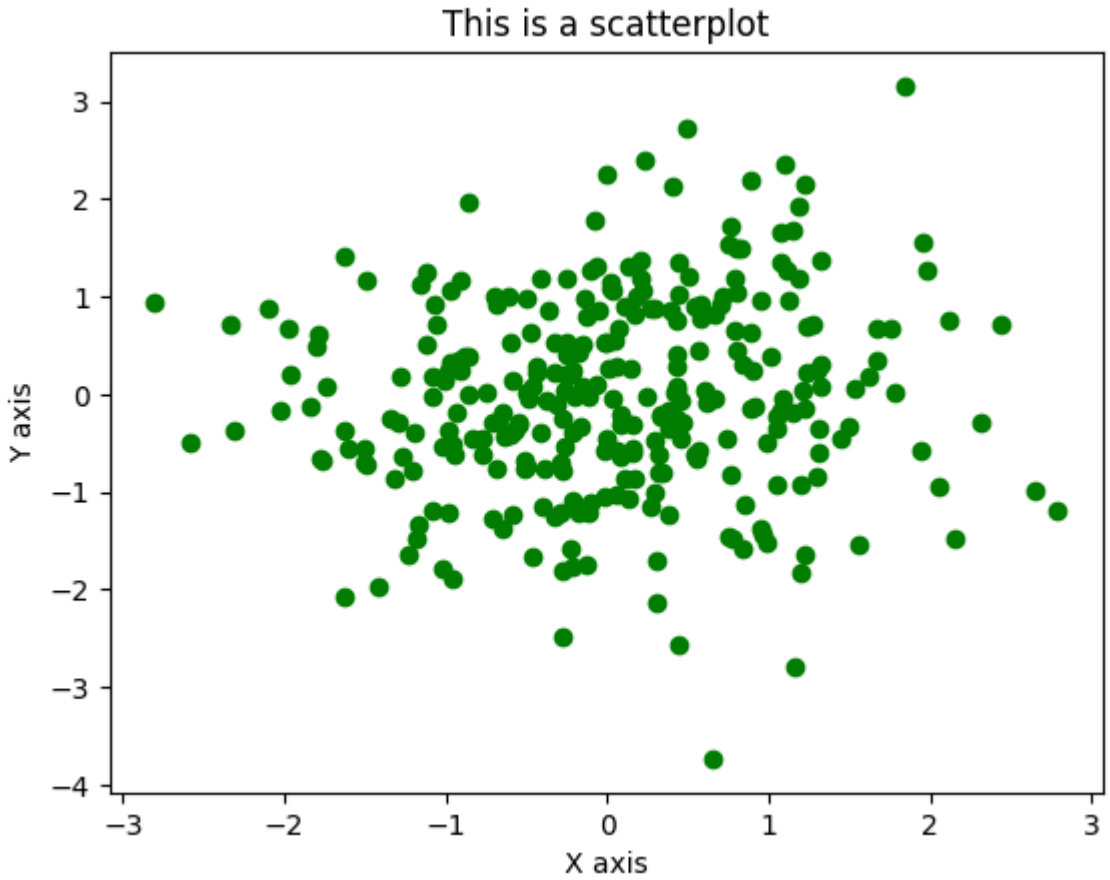
In [16]:
```
x = np.arange(start=1, stop=13, step=2)
y = [10,25,35,33,41,59]

plt.scatter(x, y)
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("This is a scatter plot")
_ = plt.show()
```



In [17]:
```
plt.scatter(x, y, c='red', marker='*', alpha=0.5, s=100)
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("This is a scatter plot")
_ = plt.show()
```

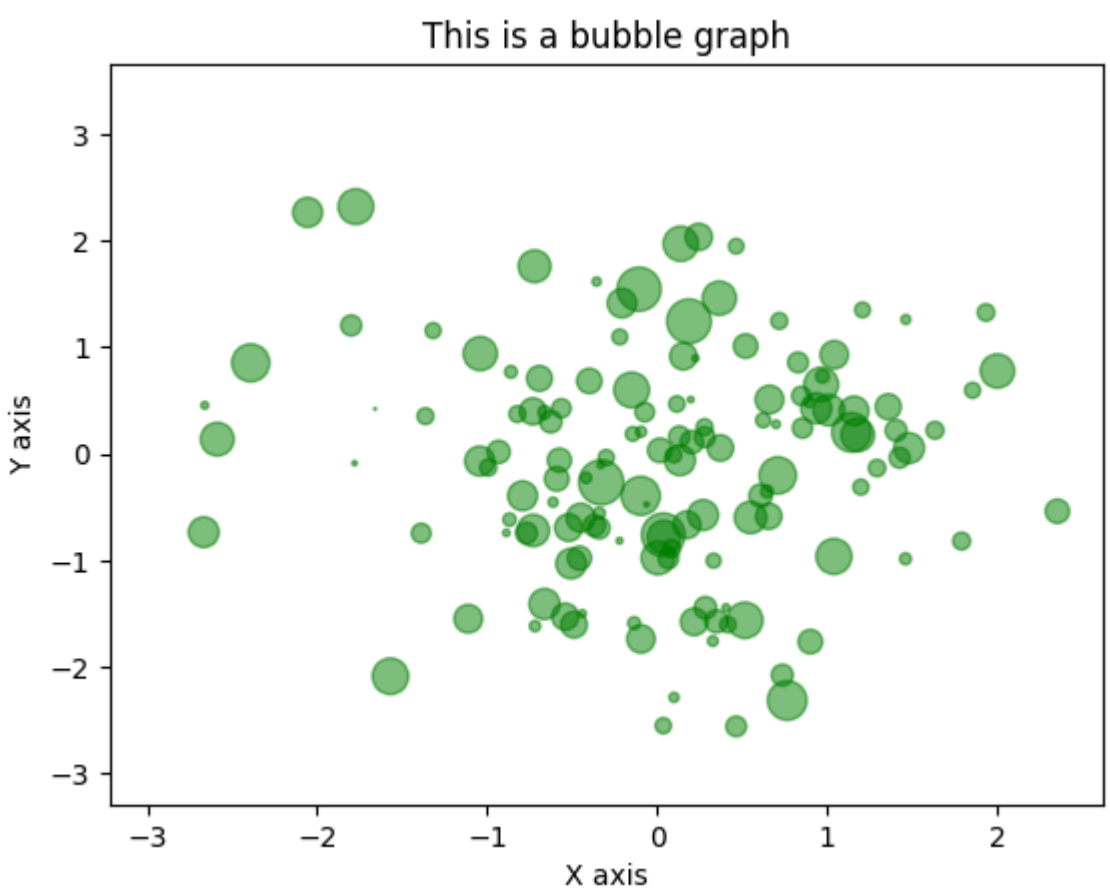


In [20]:
```
x1 = np.random.randn(300)
x2 = np.random.randn(300)
plt.scatter(x1, x2, c='green')
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("This is a scatterplot")
_ = plt.show()
```



In [22]:

```
x1 = np.random.randn(300)
x2 = np.random.randn(300)
x3 = np.random.randn(300) * 100

plt.scatter(x1, x2, c='green', s=x3, alpha=.5)
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("This is a bubble graph")
_ = plt.show()
```

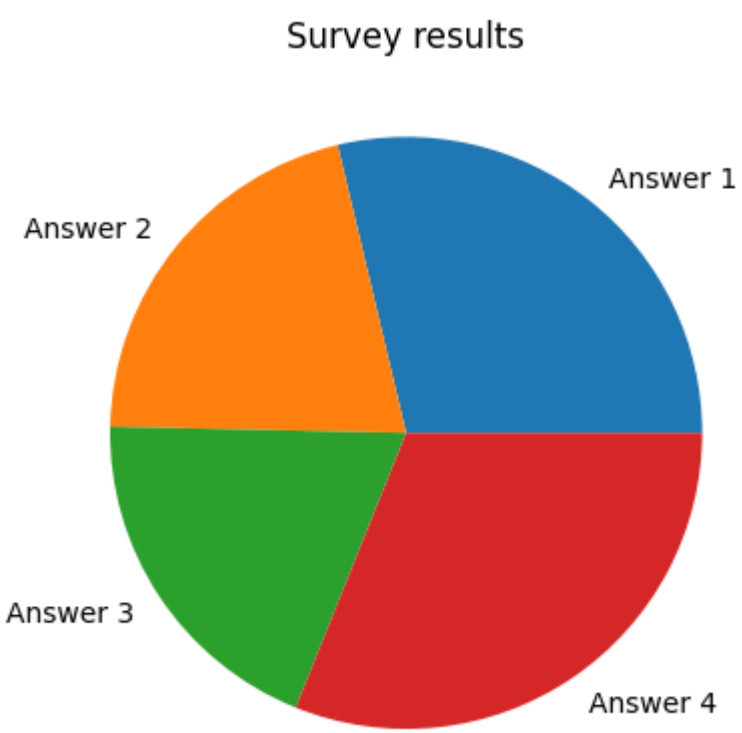


Pie chart

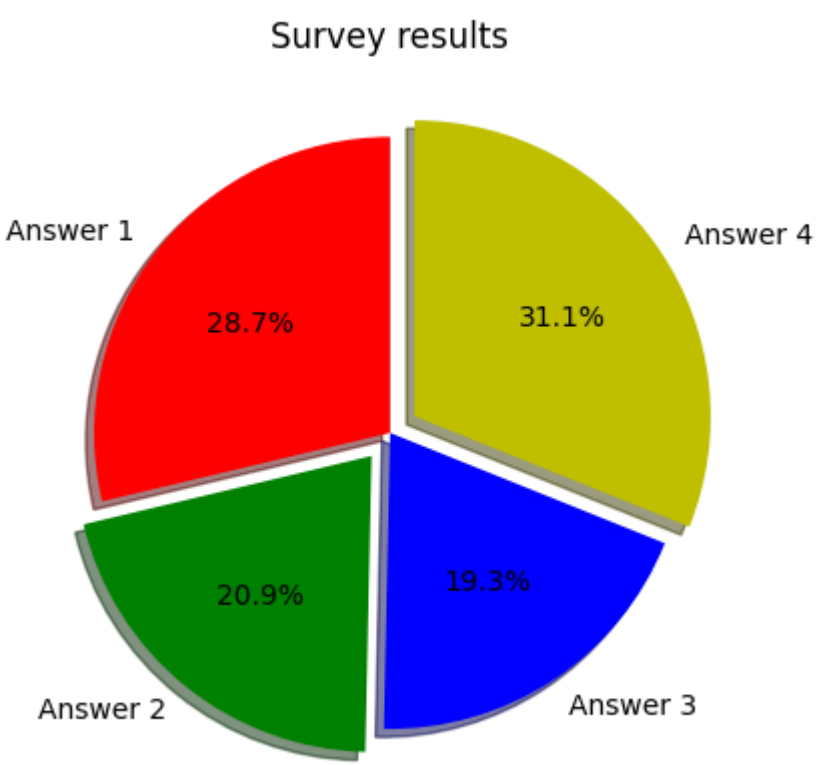
A pie chart is a chart in the form of a circle that is divided into segments in the shape of pie slices. Each slice is proportional to the value it represents. The total value of the pie is 100 percent.

A pie chart can be obtained using the `pie()` command.

In [23]:
```
answer_number = ["Answer 1","Answer 2","Answer 3","Answer 4"]
number_of_people = [85,62,57,92]

plt.pie(number_of_people, labels=answer_number)
plt.title("Survey results")

_ = plt.show()
```



In [24]:
```
plt.pie(number_of_people, labels=answer_number, colors=['r','g','b','y'], startangle=90, shadow= True, explode=(0,0.1,0,0.1), autopct='%1.1
plt.title("Survey results")

_ = plt.show()
```

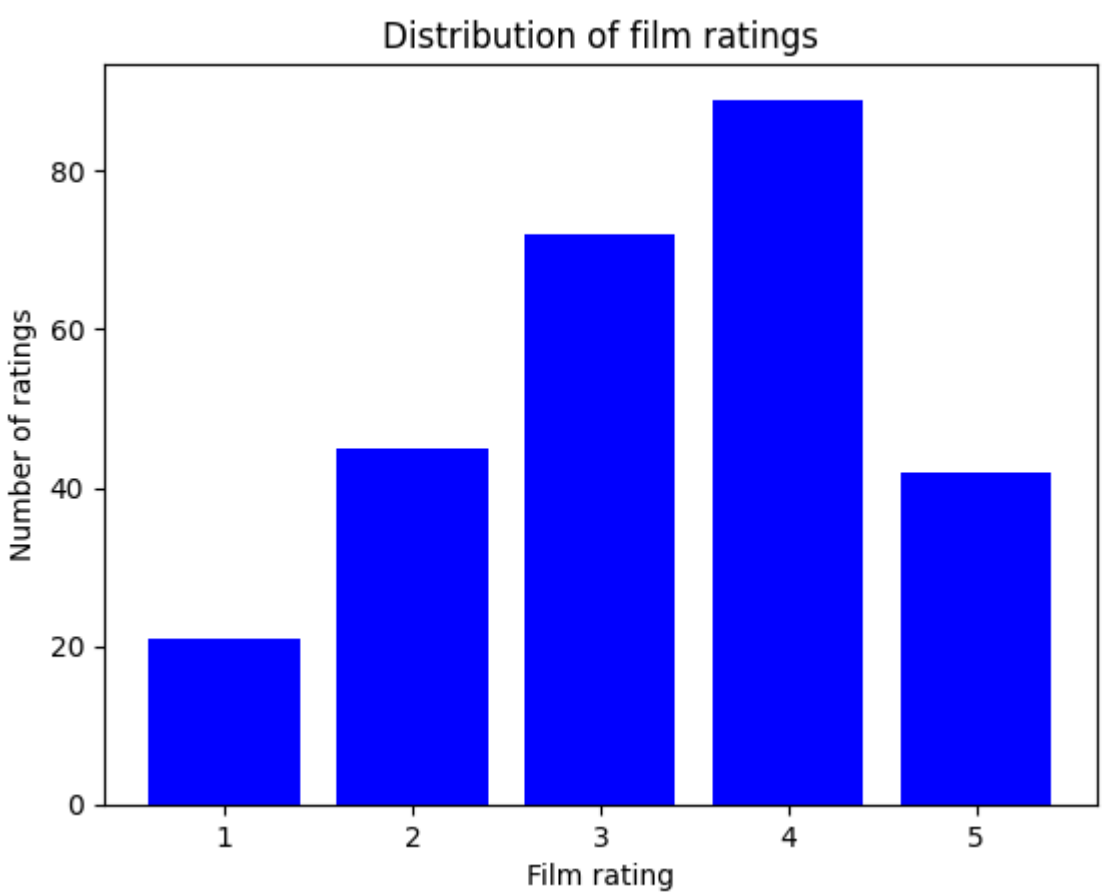


Bar chart

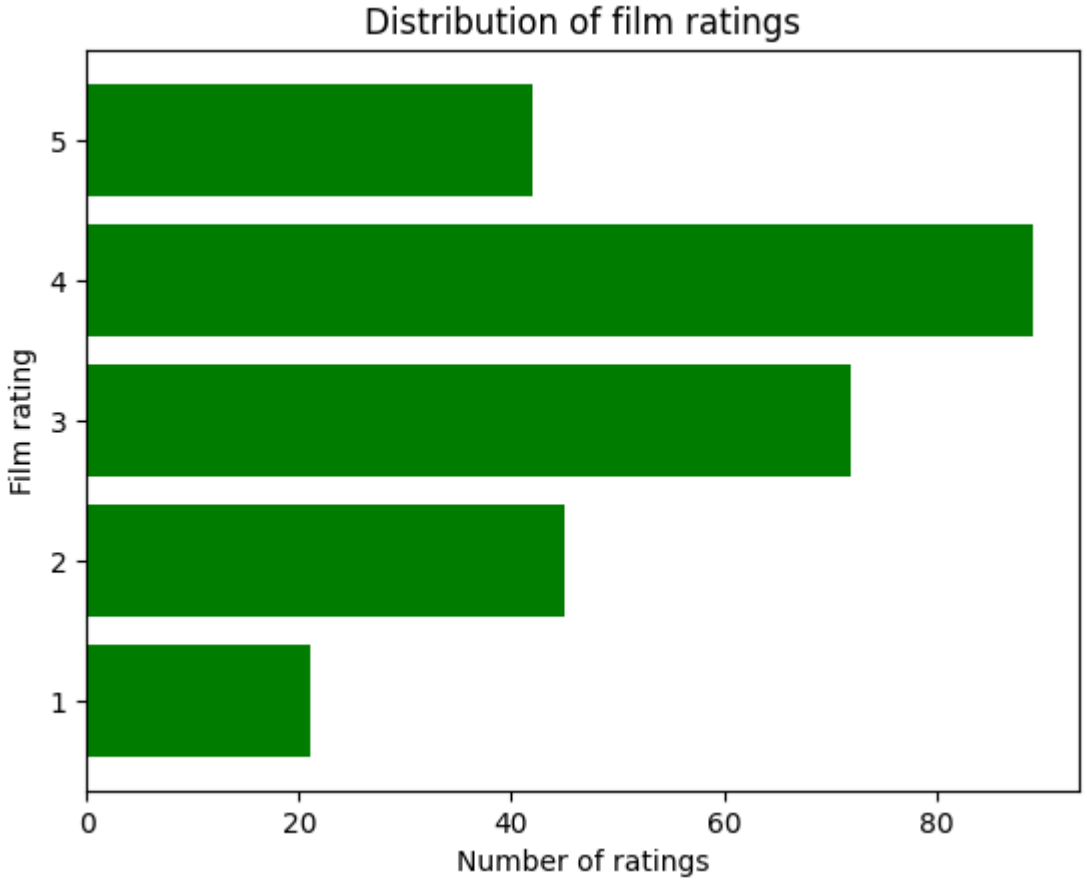
A bar chart is a visual tool for comparing the values of different groups. It can be drawn horizontally or vertically.

A bar chart can be obtained with the `bar()` command, while a horizontal version can be obtained with `barh()`.

In [26]:
```
film_rating = [1,2,3,4,5]
num_ratings = [21,45,72,89,42]
plt.bar(film_rating, num_ratings, color='blue')
plt.xlabel("Film rating")
plt.ylabel("Number of ratings")
plt.title("Distribution of film ratings")
_ = plt.show()
```



In [27]:
```
plt.barh(film_rating, num_ratings, color='green')
plt.ylabel("Film rating")
plt.xlabel("Number of ratings")
plt.title("Distribution of film ratings")
_ = plt.show()
```

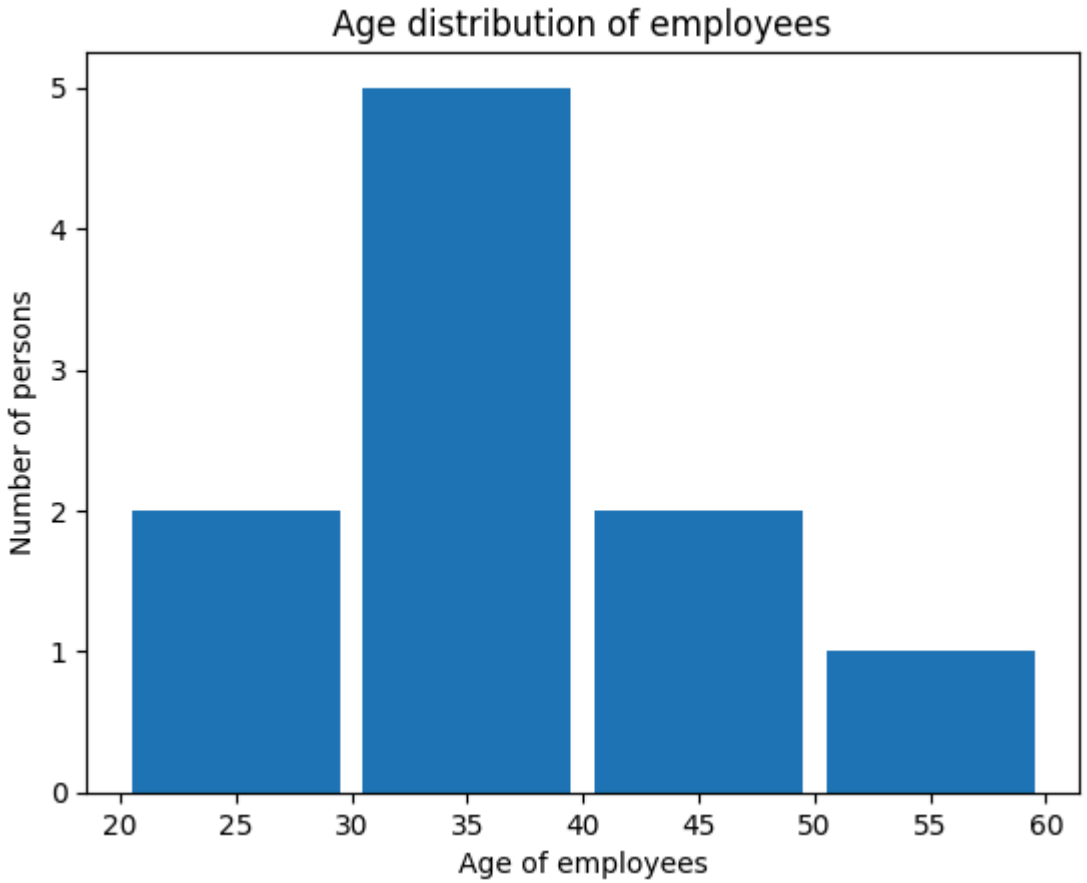


Histogram

The histogram is one of the most popular statistical graphs. It is used to present the distribution of a given trait within given ranges of values.

The histogram can be obtained using the `hist()` command.

In [31]:
```
employees_age = [21,28,32,34,35,35,37,42,47,55]
intervals = [20,30,40,50,60]
plt.hist(employees_age, intervals, rwidth=0.9)
plt.xlabel("Age of employees")
plt.ylabel("Number of persons")
plt.title("Age distribution of employees")
plt.show()
```



Creating multiple plots in a single diagram

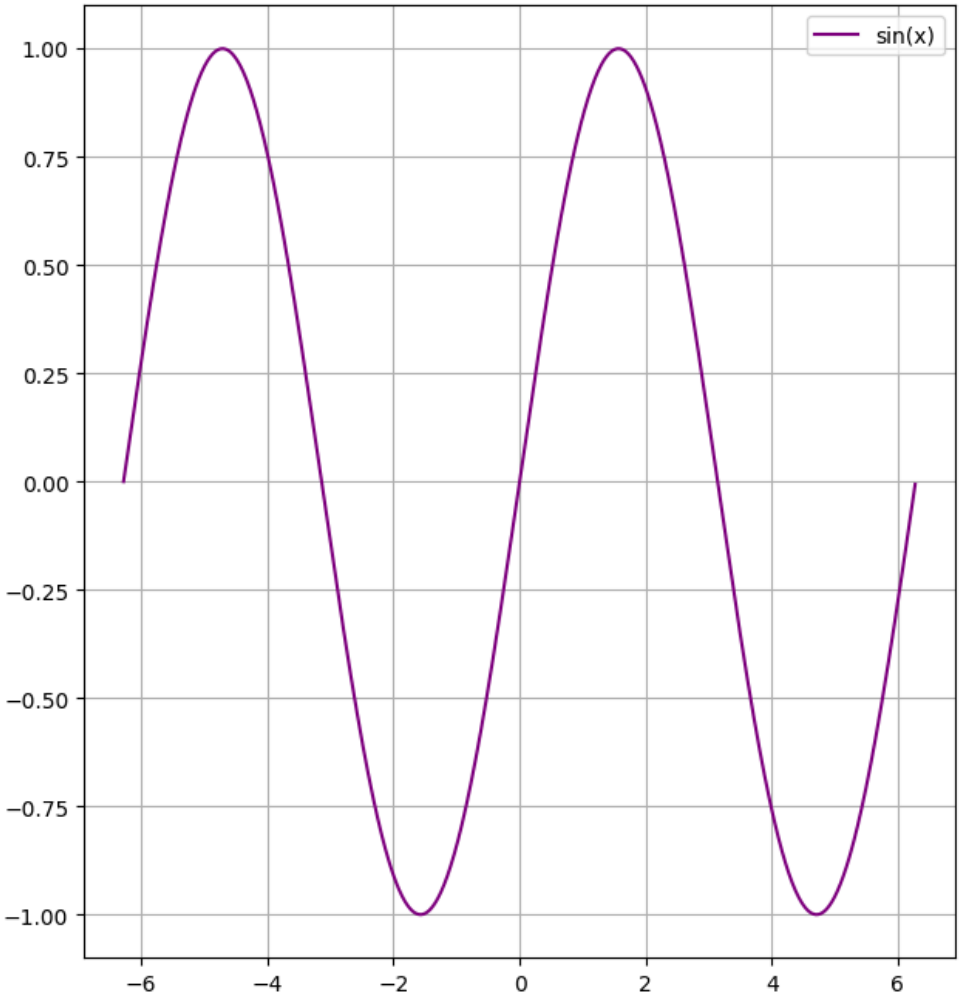
The Matplotlib library provides the ability to create multiple plots in a single diagram.

The `subplot()` function allows you to indicate the subplot on which the drawing will be performed:

`subplot(121)` - draw on a grid with one row and two columns, in graph one

In [32]:
```
x = np.arange(-2*np.pi, 2*np.pi, 0.01)
y = np.sin(x)

fig = plt.figure(figsize=(16,8)) # height and width in inches

plt.subplot(121)
plt.plot(x, y,label='sin(x)', color='purple')
plt.legend()
plt.grid()
```

```
plt.subplot(122)
y= np.cos(x)
plt.plot(x, y,label='cos(x)', color='green')
plt.legend()
plt.grid()

fig.show() # only in other tools such as PyCharm
```



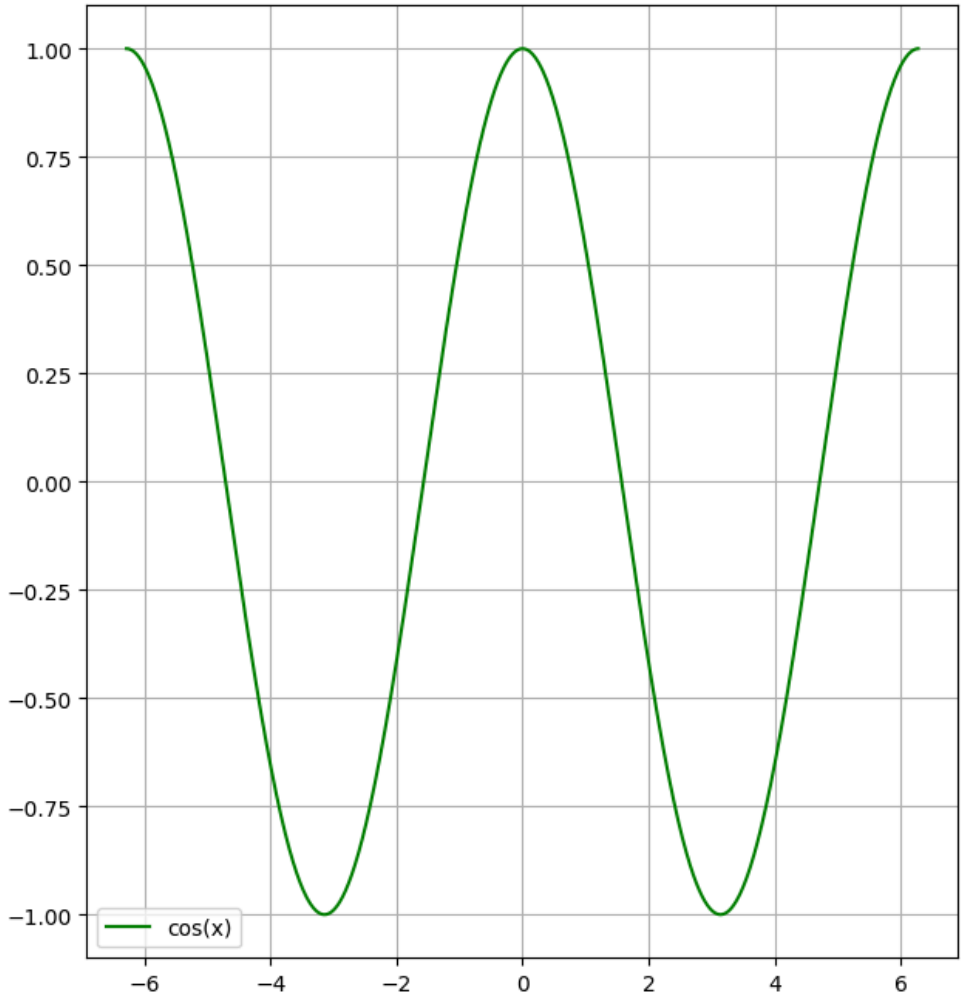
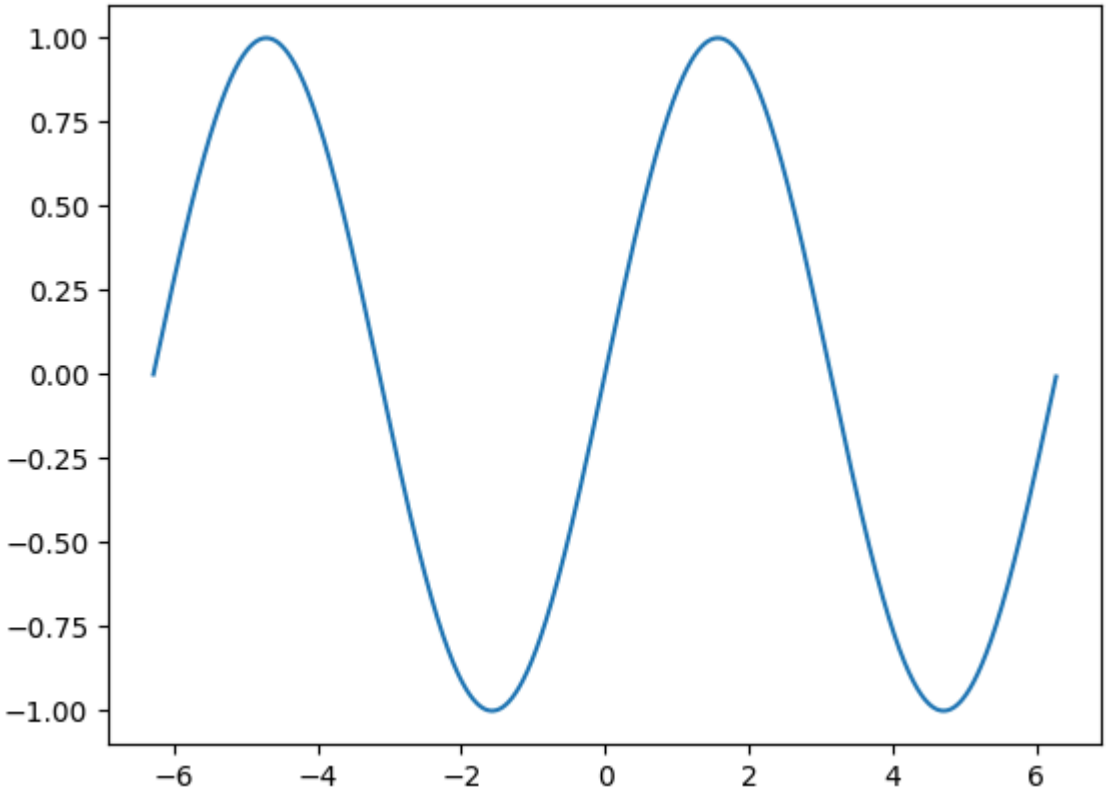


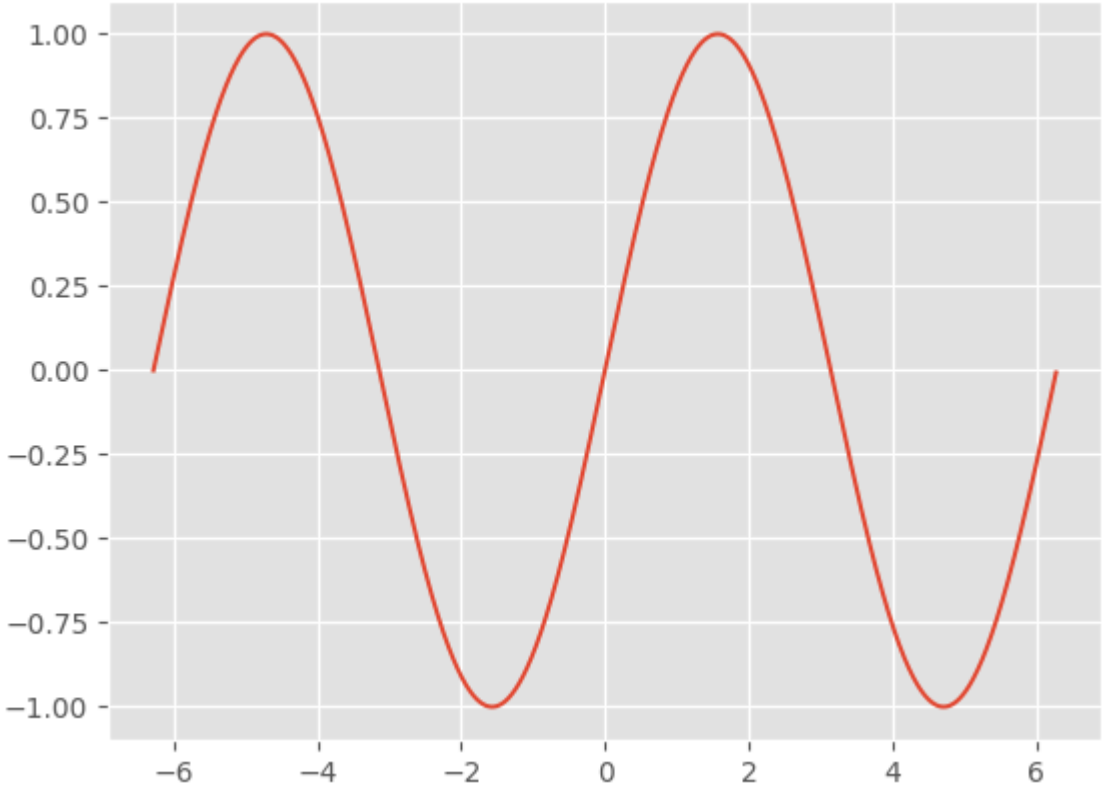
Chart styles

The Matplotlib library provides a number of styles that allow you to customize the way your charts are displayed.

In [33]:
```
x = np.arange(-2*np.pi, 2*np.pi, 0.01)
y = np.sin(x)

_ = plt.plot(x, y)
```



In [34]:
```
plt.style.use('ggplot')
_ = plt.plot(x, y)
```

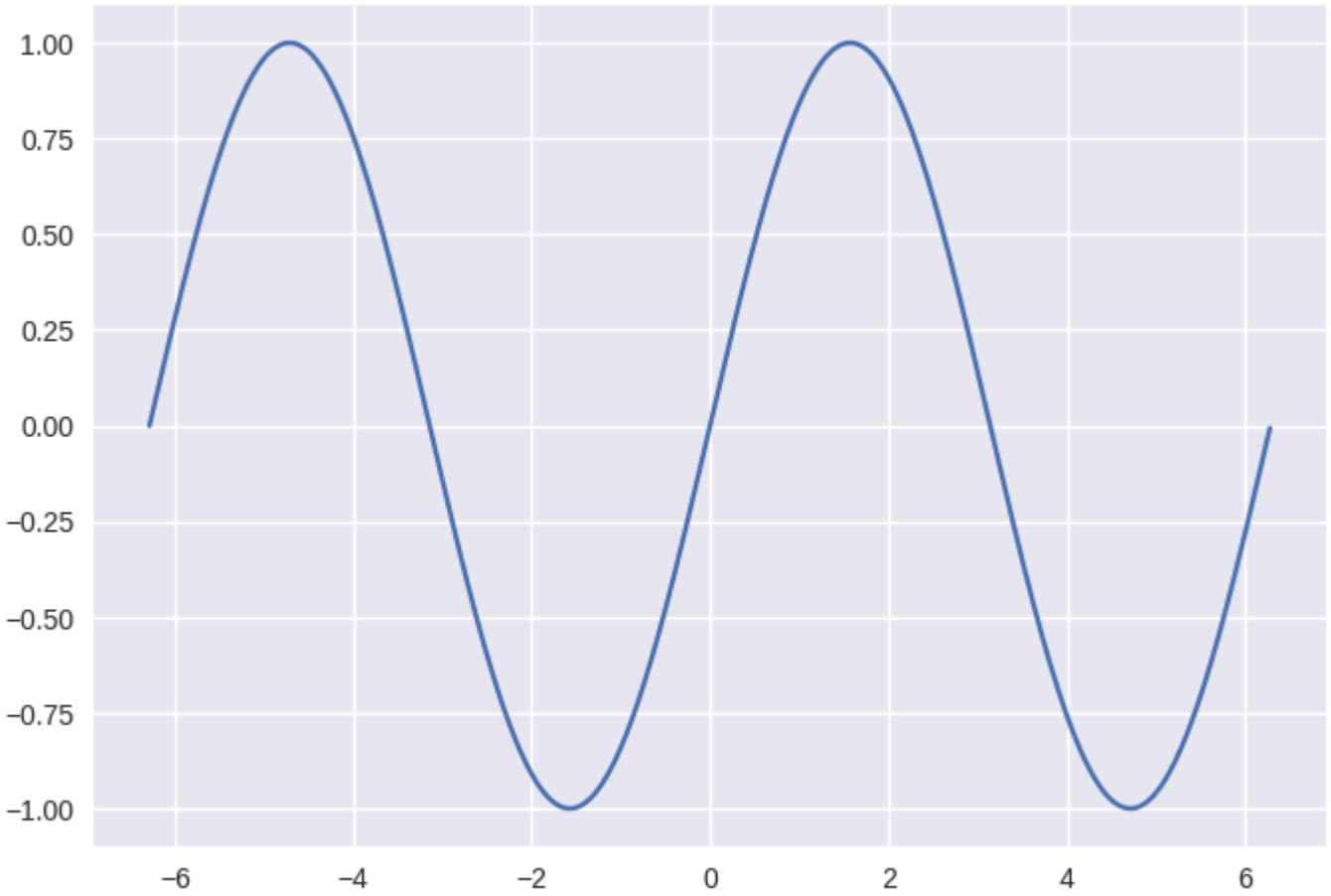


List of available styles

In [ ]:
```
plt.style.available
```

Out[ ]:
```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn-v0_8',
 'seaborn-v0_8-bright',
 'seaborn-v0_8-colorblind',
 'seaborn-v0_8-dark',
 'seaborn-v0_8-dark-palette',
 'seaborn-v0_8-darkgrid',
 'seaborn-v0_8-deep',
 'seaborn-v0_8-muted',
 'seaborn-v0_8-notebook',
 'seaborn-v0_8-paper',
 'seaborn-v0_8-pastel',
 'seaborn-v0_8-poster',
 'seaborn-v0_8-talk',
 'seaborn-v0_8-ticks',
 'seaborn-v0_8-white',
 'seaborn-v0_8-whitegrid',
 'tableau-colorblind10']
```

In [35]:
```
plt.style.use('seaborn-v0_8')
_ = plt.plot(x, y)
```



Seaborn Library

The Seaborn library provides an API based on Matplotlib that defines advanced functions for typical types of statistical graphs and integrates with data frames from the Pandas library.

Library website: https://seaborn.pydata.org/

Documentation: https://seaborn.pydata.org/api.html

Installation: `pip install seaborn`

Importing Seaborn library

In [36]:
```
import seaborn as sns
sns.__version__
```

Out[36]:
```
'0.13.2'
```

In [37]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

Sample data

In [38]:
```
data = sns.load_dataset('taxis')
data.head(10)
```

Out[38]:

	pickup	dropoff	passengers	distance	fare	tip	tolls	total	color	payment	pickup_zone	dropoff_zone	pickup_borough	dropoff_borough
0	2019-03-23 20:21:09	2019-03-23 20:27:24	1	1.60	7.0	2.15	0.0	12.95	yellow	credit card	Lenox Hill West	UN/Turtle Bay South	Manhattan	Manhattan
1	2019-03-04 16:11:55	2019-03-04 16:19:00	1	0.79	5.0	0.00	0.0	9.30	yellow	cash	Upper West Side South	Upper West Side South	Manhattan	Manhattan
2	2019-03-27 17:53:01	2019-03-27 18:00:25	1	1.37	7.5	2.36	0.0	14.16	yellow	credit card	Alphabet City	West Village	Manhattan	Manhattan
3	2019-03-10 01:23:59	2019-03-10 01:49:51	1	7.70	27.0	6.15	0.0	36.95	yellow	credit card	Hudson Sq	Yorkville West	Manhattan	Manhattan
4	2019-03-30 13:27:42	2019-03-30 13:37:14	3	2.16	9.0	1.10	0.0	13.40	yellow	credit card	Midtown East	Yorkville West	Manhattan	Manhattan
5	2019-03-11 10:37:23	2019-03-11 10:47:31	1	0.49	7.5	2.16	0.0	12.96	yellow	credit card	Times Sq/Theatre District	Midtown East	Manhattan	Manhattan
6	2019-03-26 21:07:31	2019-03-26 21:17:29	1	3.65	13.0	2.00	0.0	18.80	yellow	credit card	Battery Park City	Two Bridges/Seward Park	Manhattan	Manhattan
7	2019-03-22 12:47:13	2019-03-22 12:58:17	0	1.40	8.5	0.00	0.0	11.80	yellow	NaN	Murray Hill	Flatiron	Manhattan	Manhattan
8	2019-03-23 11:48:50	2019-03-23 12:06:14	1	3.63	15.0	1.00	0.0	19.30	yellow	credit card	East Harlem South	Midtown Center	Manhattan	Manhattan
9	2019-03-08 16:18:37	2019-03-08 16:26:57	1	1.52	8.0	1.00	0.0	13.30	yellow	credit card	Lincoln Square East	Central Park	Manhattan	Manhattan

In [39]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6433 entries, 0 to 6432
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   pickup           6433 non-null   datetime64[ns]
 1   dropoff          6433 non-null   datetime64[ns]
 2   passengers       6433 non-null   int64
 3   distance         6433 non-null   float64
 4   fare             6433 non-null   float64
 5   tip              6433 non-null   float64
 6   tolls            6433 non-null   float64
 7   total            6433 non-null   float64
 8   color            6433 non-null   object
 9   payment          6389 non-null   object
 10  pickup_zone      6407 non-null   object
 11  dropoff_zone     6388 non-null   object
 12  pickup_borough   6407 non-null   object
 13  dropoff_borough  6388 non-null   object
dtypes: datetime64[ns](2), float64(5), int64(1), object(6)
memory usage: 703.7+ KB
```

In [40]:
```
data.describe().T
```

Out[40]:

	count	mean	min	25%	50%	75%	max	std
pickup	6433	2019-03-16 08:31:28.514223616	2019-02-28 23:29:03	2019-03-08 15:50:34	2019-03-15 21:46:58	2019-03-23 17:41:38	2019-03-31 23:43:45	NaN
dropoff	6433	2019-03-16 08:45:49.491217408	2019-02-28 23:32:35	2019-03-08 16:12:51	2019-03-15 22:06:44	2019-03-23 17:51:56	2019-04-01 00:13:58	NaN
passengers	6433.0	1.539251	0.0	1.0	1.0	2.0	6.0	1.203768
distance	6433.0	3.024617	0.0	0.98	1.64	3.21	36.7	3.827867
fare	6433.0	13.091073	1.0	6.5	9.5	15.0	150.0	11.551804
tip	6433.0	1.97922	0.0	0.0	1.7	2.8	33.2	2.44856
tolls	6433.0	0.325273	0.0	0.0	0.0	0.0	24.02	1.415267
total	6433.0	18.517794	1.3	10.8	14.16	20.3	174.82	13.81557

In [41]:
```
data.describe(include=['object']).T
```

Out[41]:

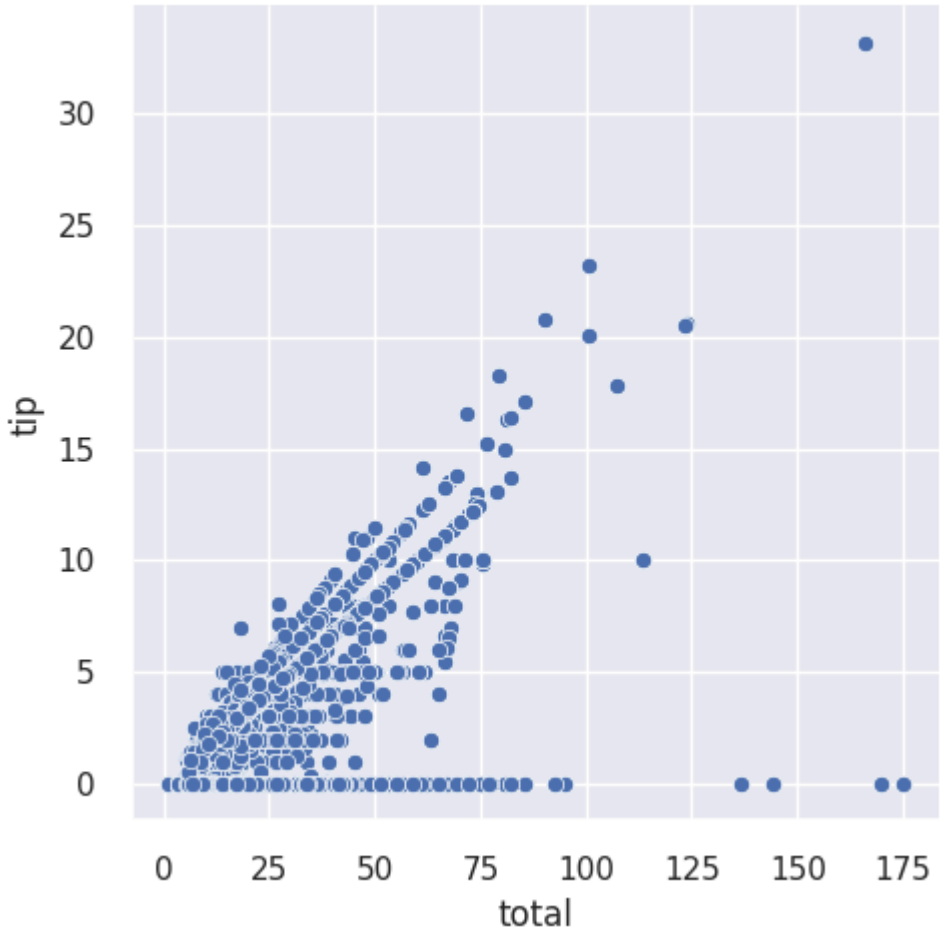
	count	unique	top	freq
color	6433	2	yellow	5451
payment	6389	2	credit card	4577
pickup_zone	6407	194	Midtown Center	230
dropoff_zone	6388	203	Upper East Side North	245
pickup_borough	6407	4	Manhattan	5268
dropoff_borough	6388	5	Manhattan	5206

Selected Seaborn Library Data Graphs

Scatter diagram

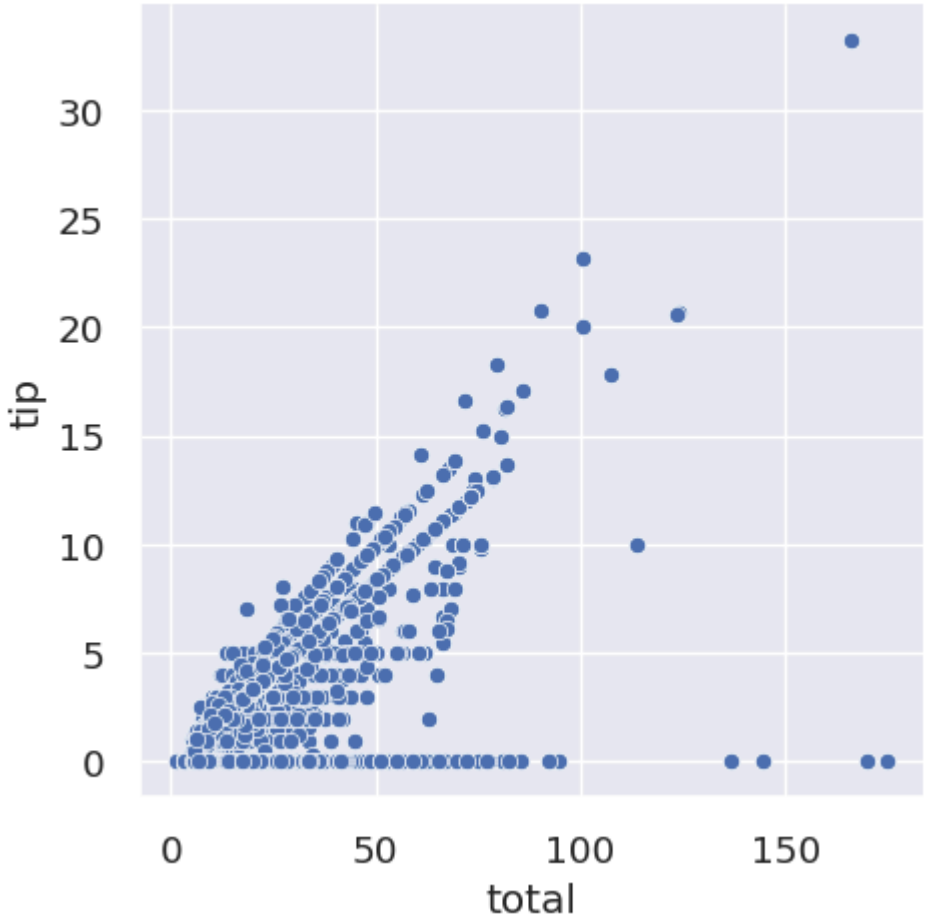
The `relplot` function can be used to draw a scatter plot.

In [42]:
```
_ = sns.relplot(data=data, x='total', y='tip')
```



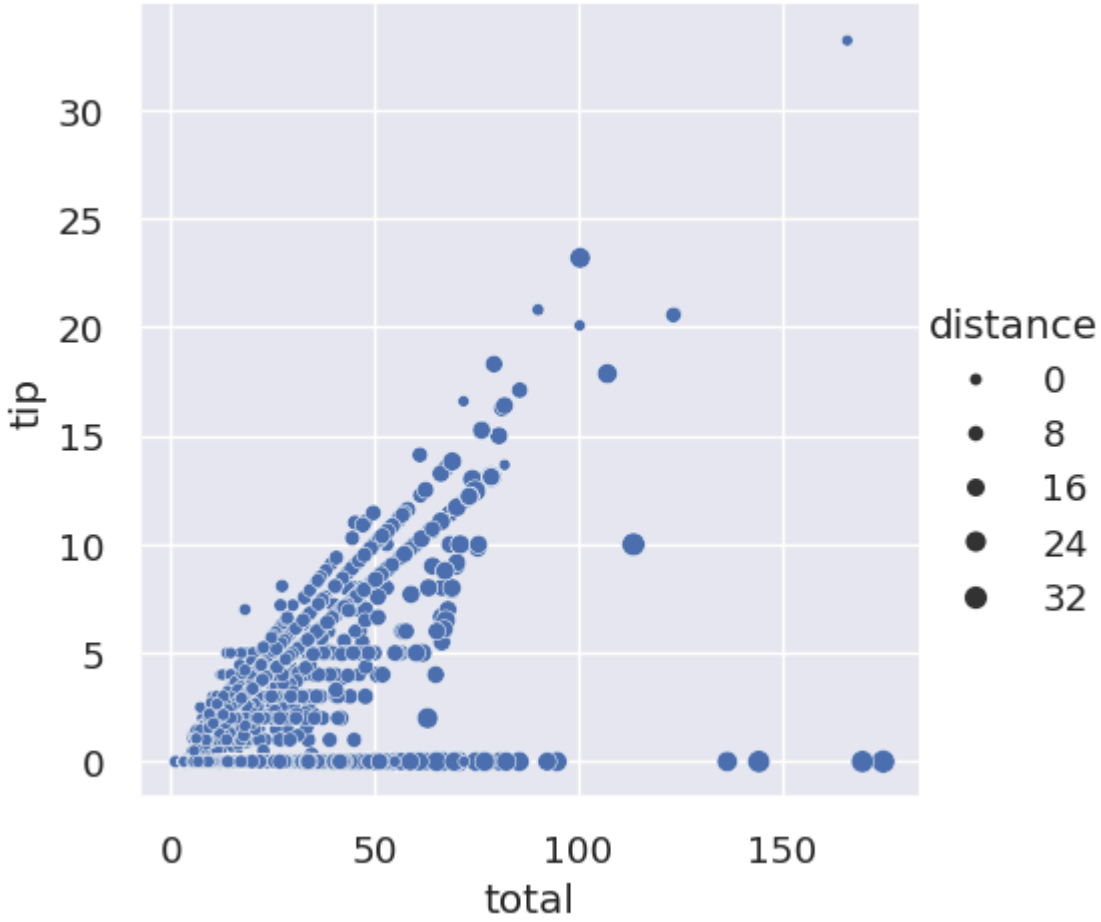
Font scaling

In [45]:
```
sns.set(font_scale=1.2)
_ = sns.relplot(data=data, x='total', y='tip')
```



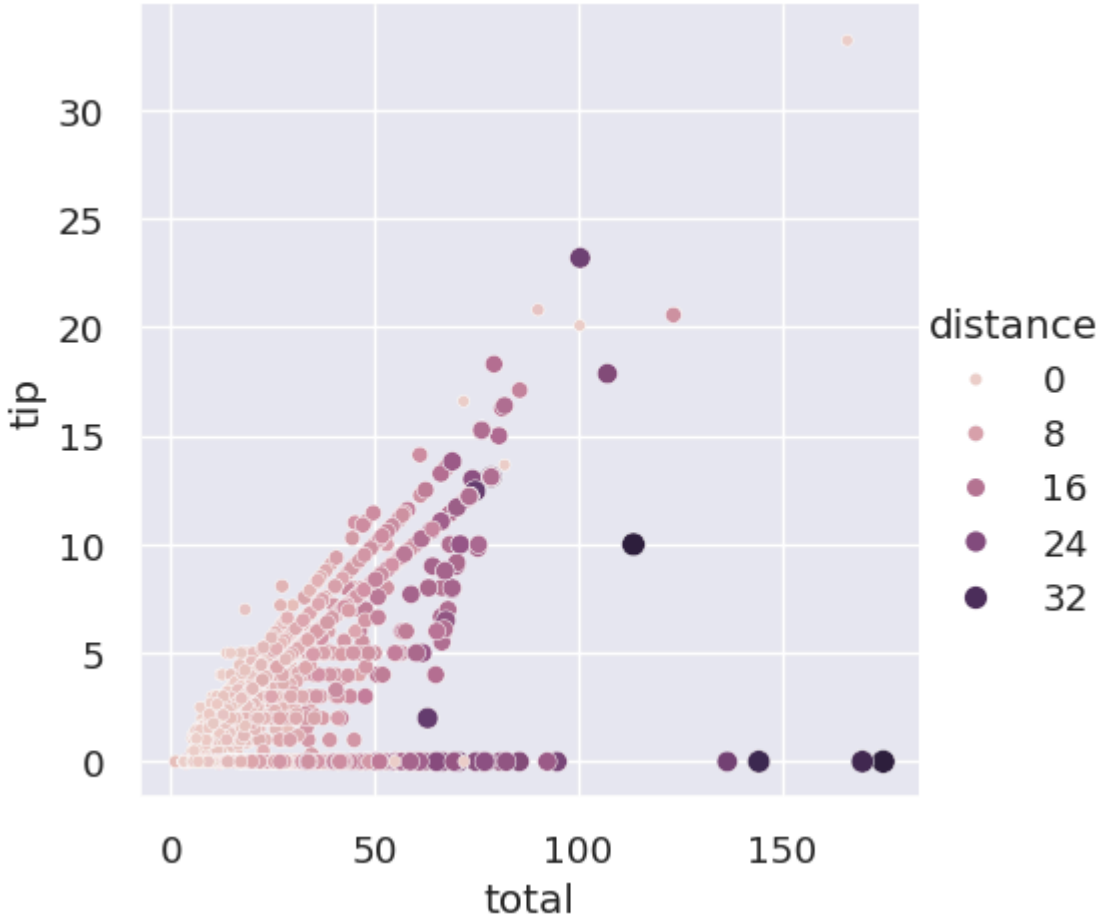
Additional data dimension.

In [46]:
```
_ = sns.relplot(data=data, x='total', y='tip', size='distance')
```



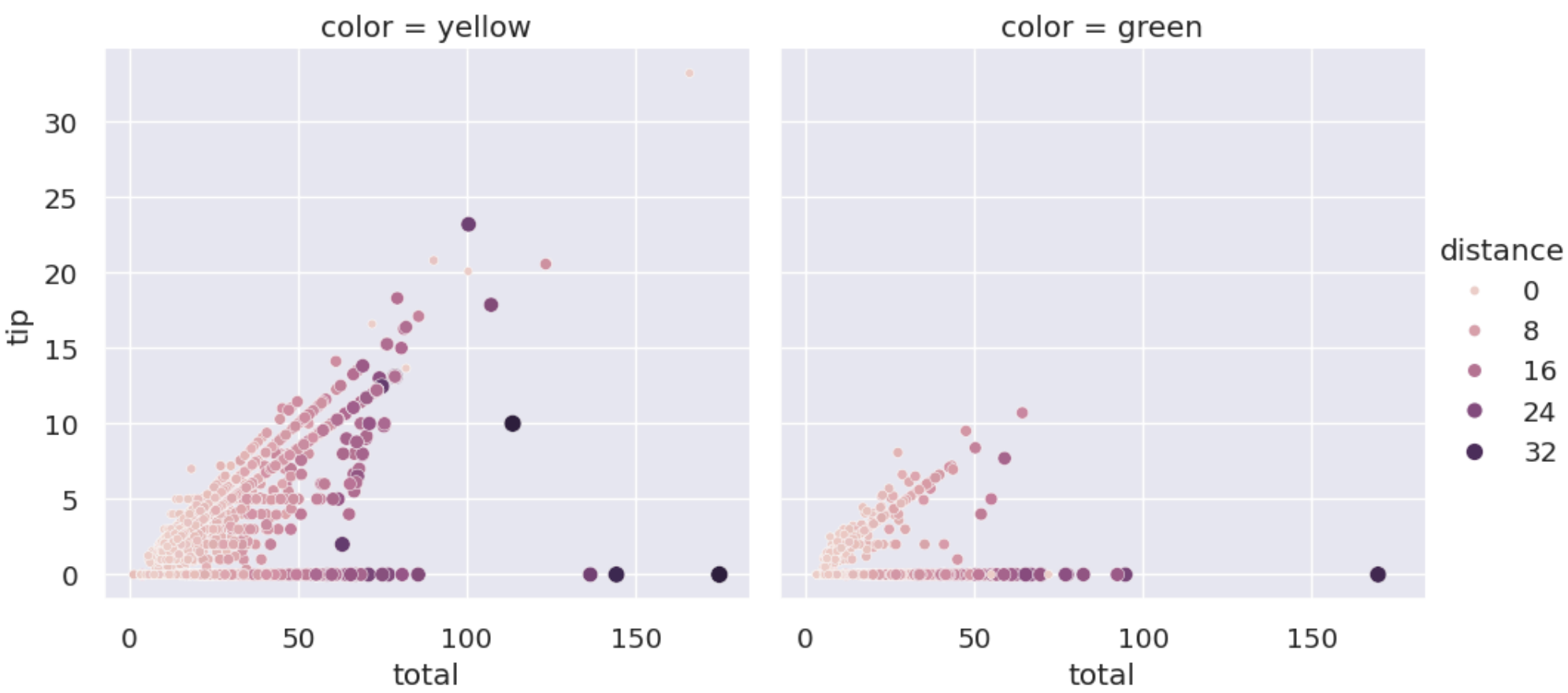
Change of colour scheme for additional data dimension

In [48]:
```
_ = sns.relplot(data=data, x='total', y='tip', size='distance', hue='distance')
```



Scatter plots by category (`col` parameter for columns)

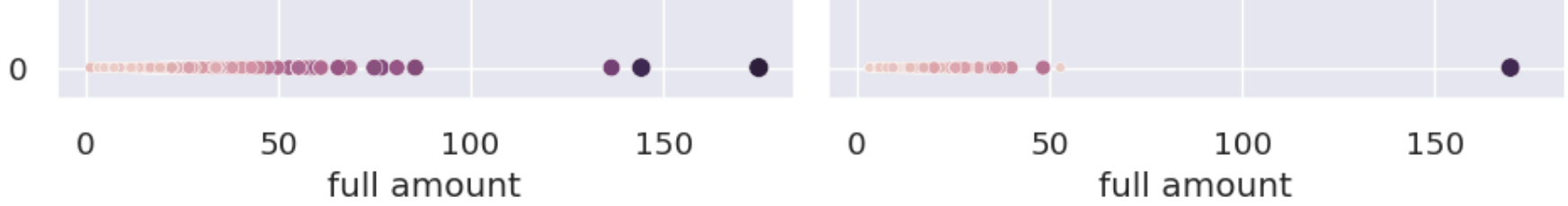
In [49]:
```
_ = sns.relplot(data=data, x='total', y='tip', size='distance', hue='distance', col='color')
```



Scatter plots by category (`row` parameter for rows)

In [50]:
```
_ = sns.relplot(data=data, x='total', y='tip', size='distance', hue='distance', col='color', row='payment').set_axis_labels('full amount',
```

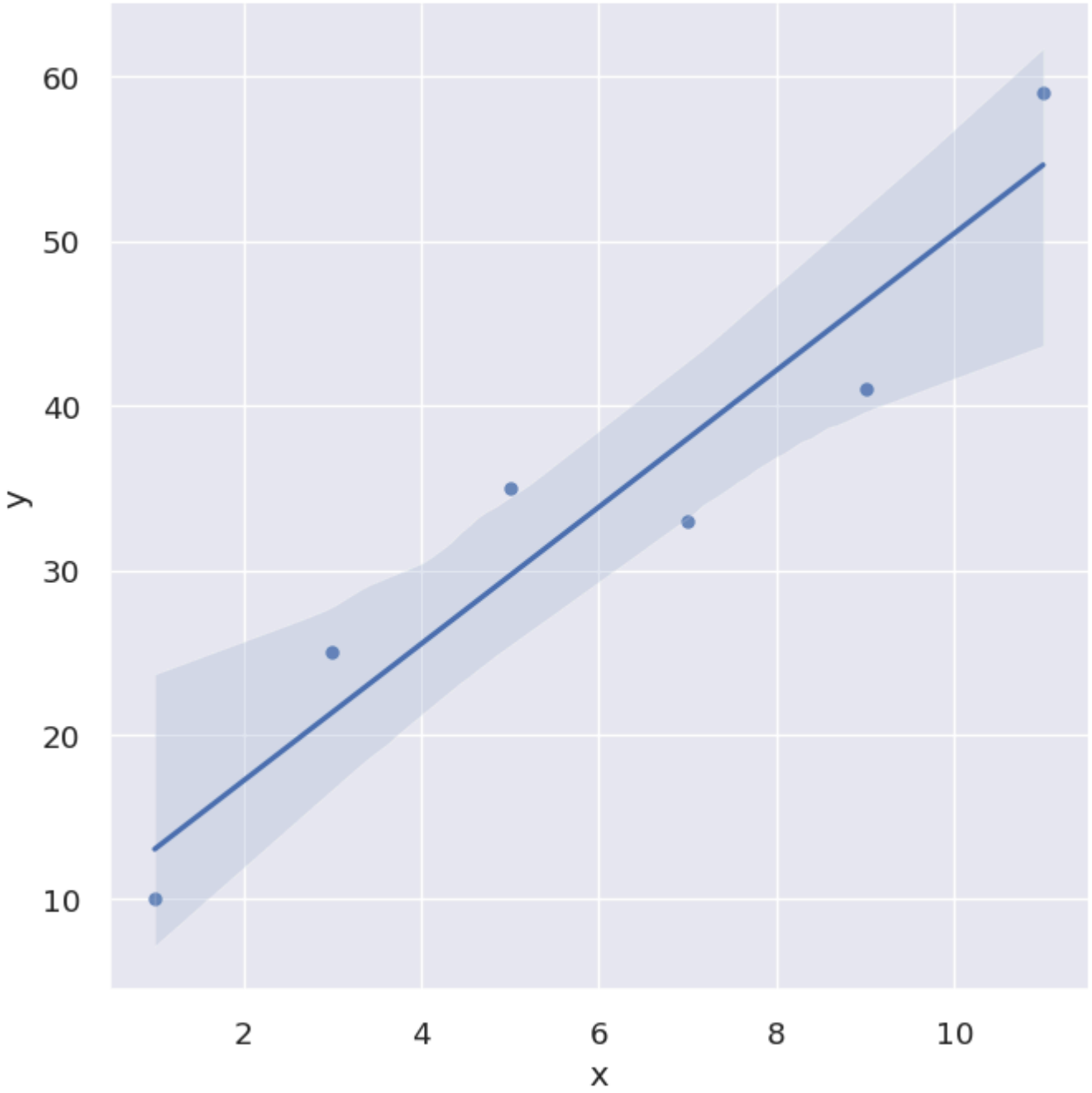

payment = credit card | color = yellow
payment = credit card | color = green
payment = cash | color = yellow
payment = cash | color = green
tip
0
5
10
15
20
25
30
distance
0
8
16
24
32



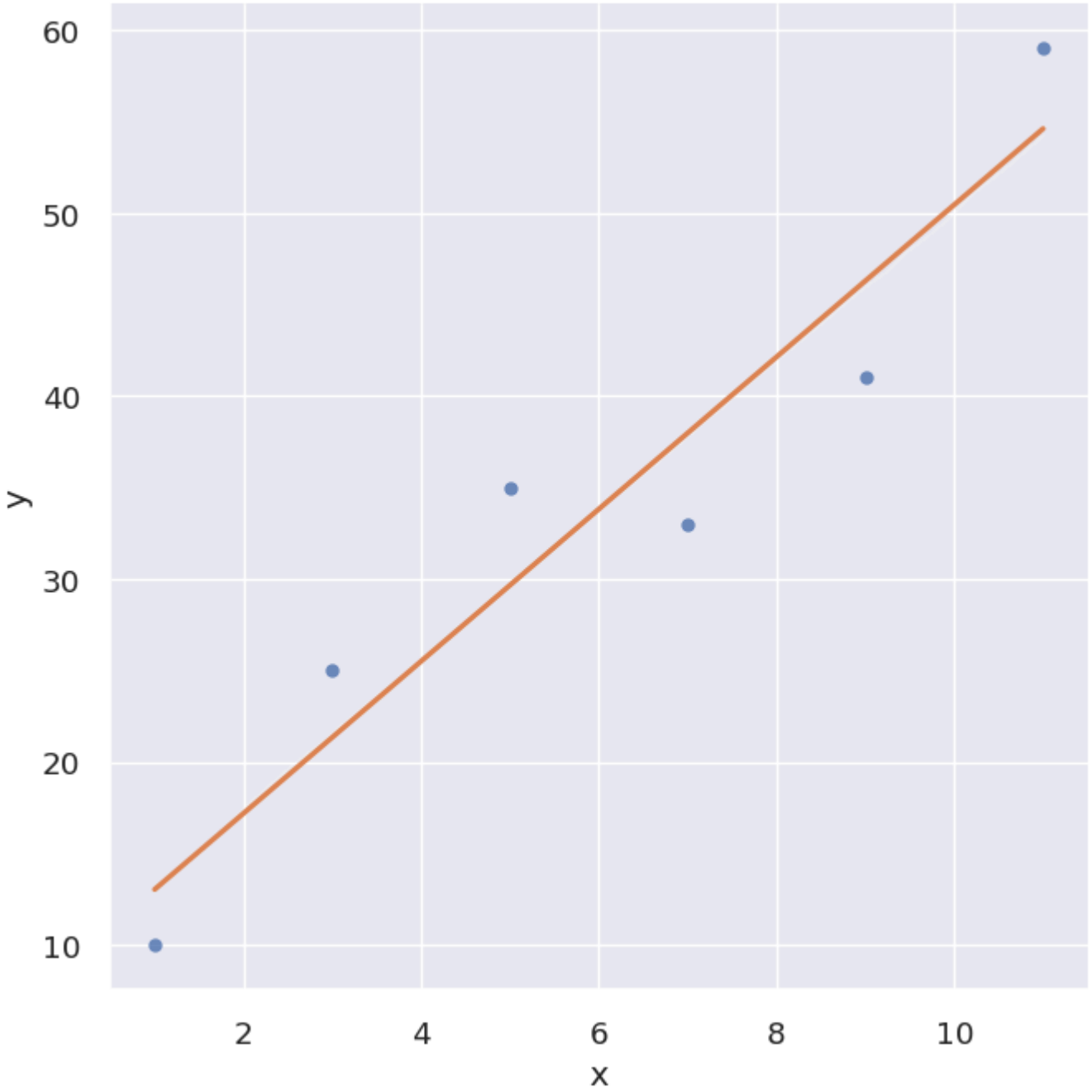
Scatter plot with regression model

A scatter plot with the regression model can be obtained using the `lmplot` command.

In [51]:
```
dane_reg = pd.DataFrame({'x':[1,3,5,7,9,11],'y':[10,25,35,33,41,59]})
_ = sns.lmplot(x='x', y='y', data=dane_reg, height=7)
```



In [52]:
```
_ = sns.lmplot(x='x', y='y', data=dane_reg, height=7, ci=False, line_kws={"color": "C1"})
```



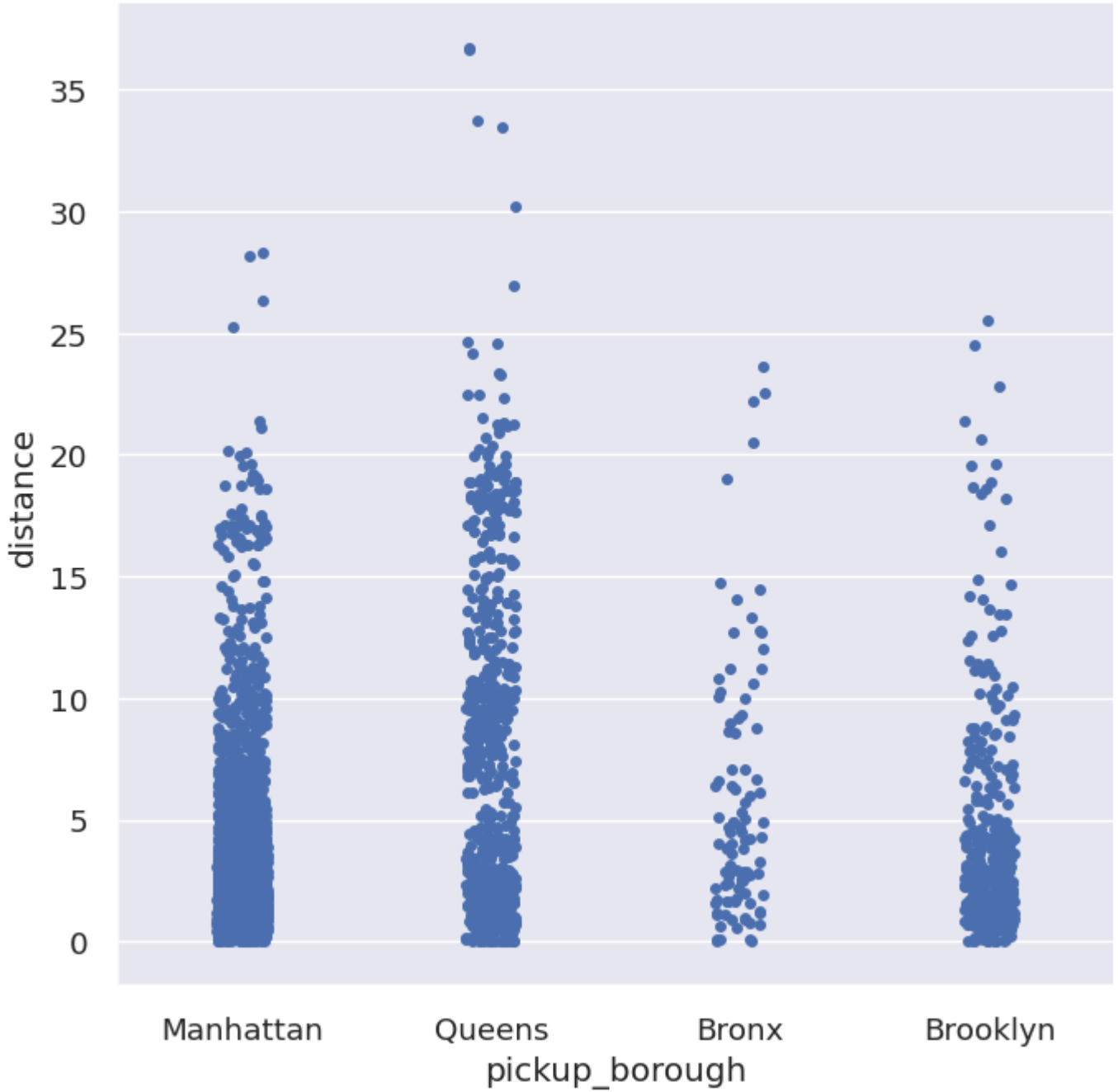
Categorical value charts

To plot a graph of categorical values you can use the `catplot` function.

Distribution diagram

Default graph for the `catplot` function.

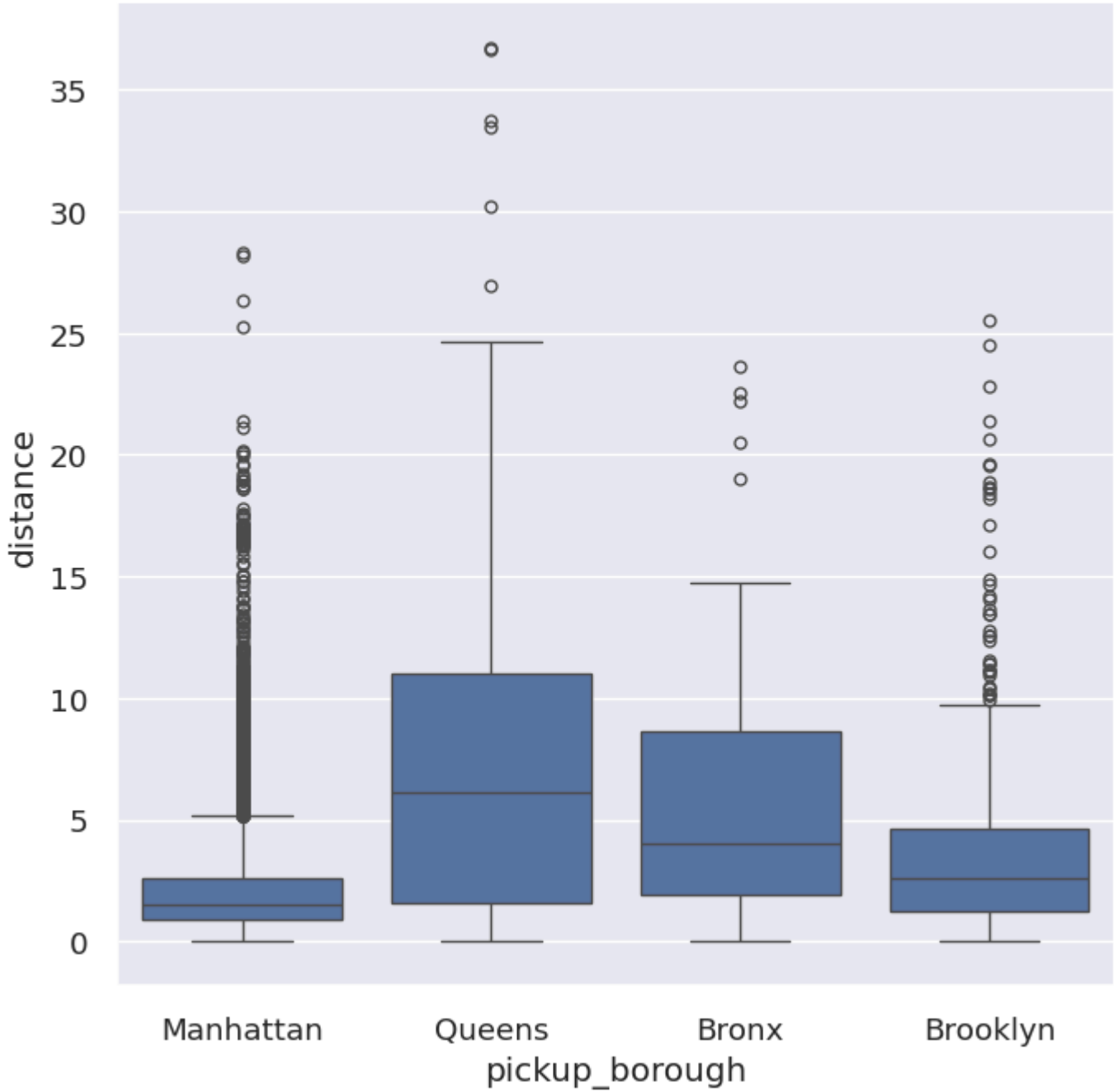
In [53]:
```
_ = sns.catplot(data=data, x='pickup_borough', y='distance', height=7)
```



Box plot chart

Parameter `kind='box'` .

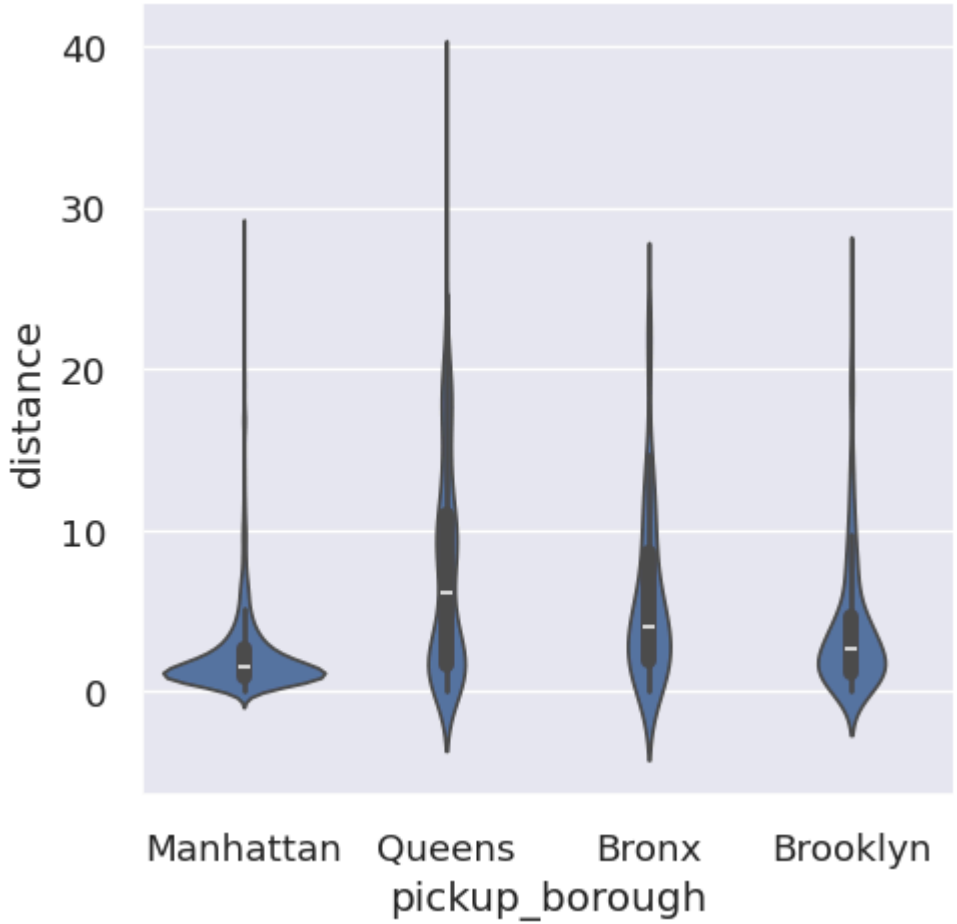
In [55]:
```
_ = sns.catplot(data=data, x='pickup_borough', y='distance', kind='box', height=7)
```



Violin plot

Parameter `kind='violin'` .

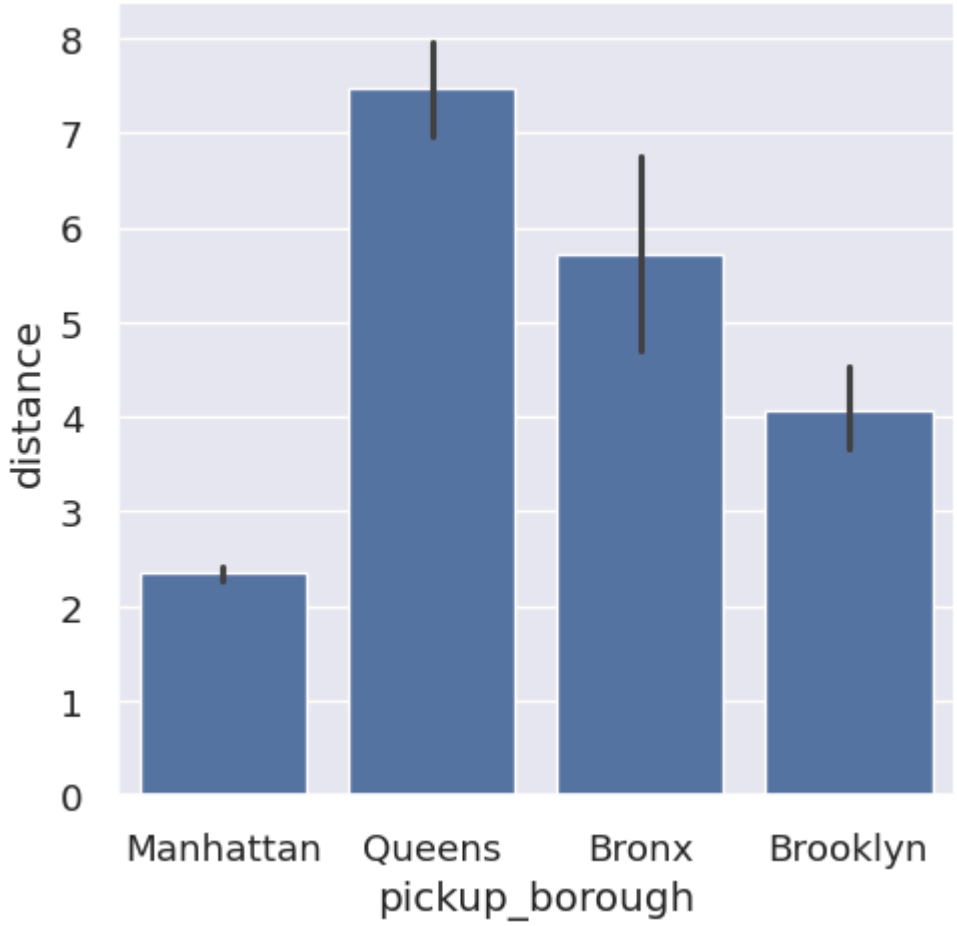
In [56]:
```
_ = sns.catplot(data=data, x='pickup_borough', y='distance', kind='violin')
```



Bar chart

Parameter `kind='bar'`.

In [ ]:
```
_ = sns.catplot(data=data, x='pickup_borough', y='distance', kind='bar')
```



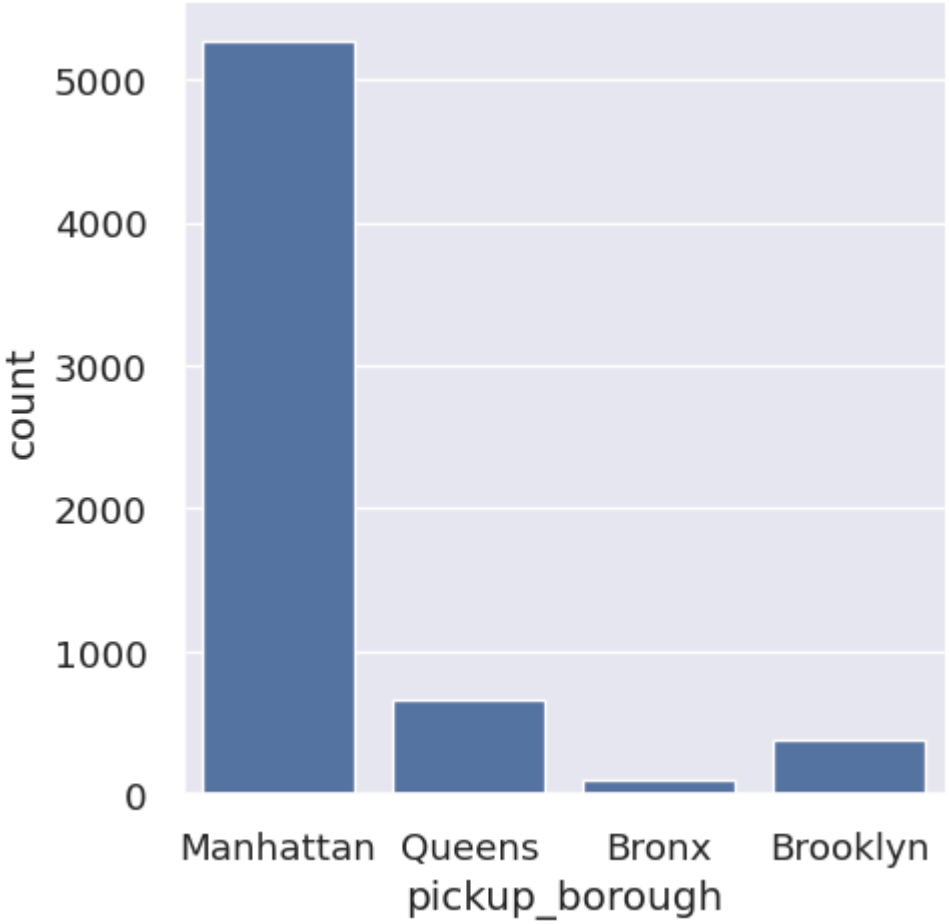
Data count - parameter `kind='count'`

In [58]:
```
grouped = data.groupby(by='pickup_borough')
grouped.distance.describe()
```

Out[58]:

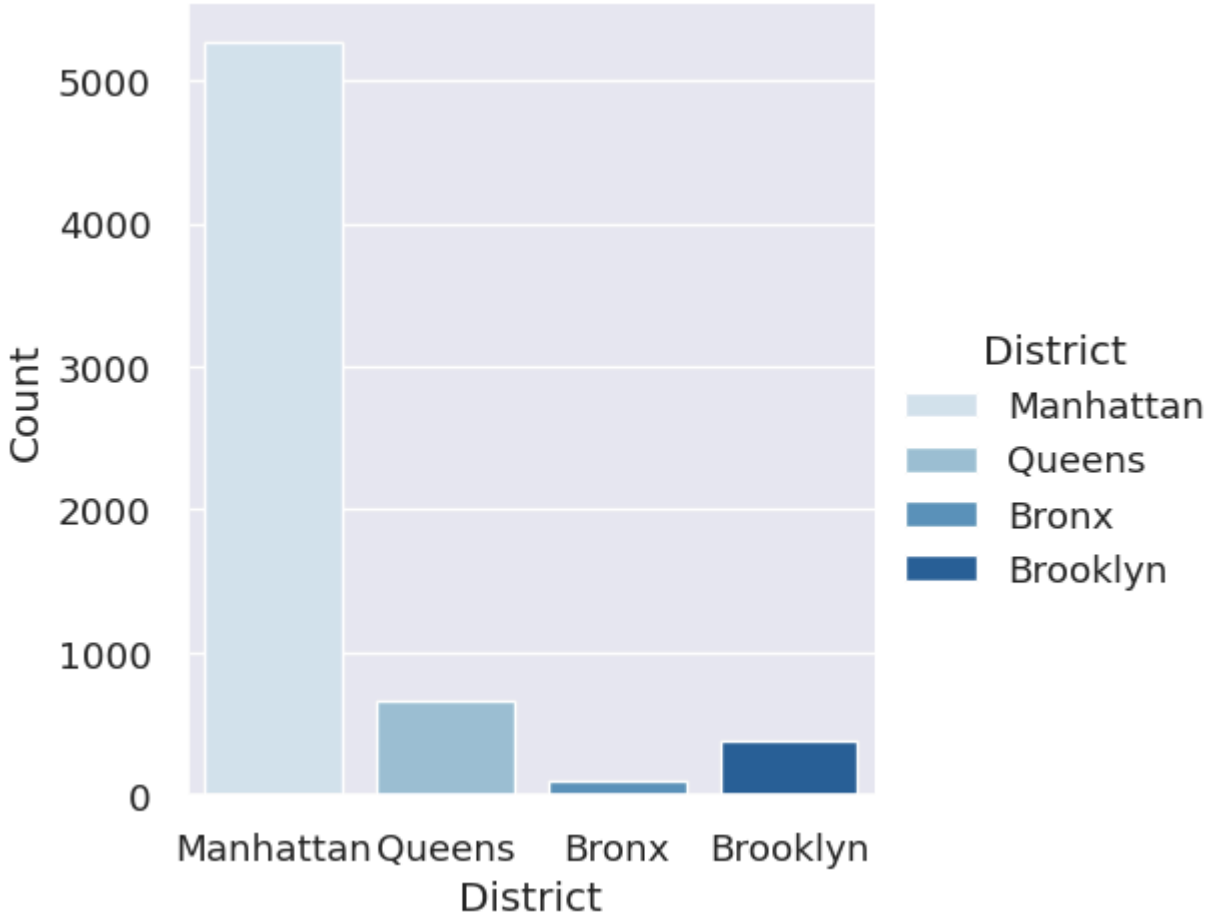
pickup_borough	count	mean	std	min	25%	50%	75%	max
Bronx	99.0	5.725859	5.301477	0.0	1.945	4.0	8.615	23.61
Brooklyn	383.0	4.058668	4.419275	0.0	1.220	2.6	4.625	25.51
Manhattan	5268.0	2.349723	2.668469	0.0	0.930	1.5	2.630	28.30
Queens	657.0	7.465830	6.712125	0.0	1.600	6.1	10.990	36.70

In [59]:
```
_ = sns.catplot(data=data, x='pickup_borough', kind='count')
```



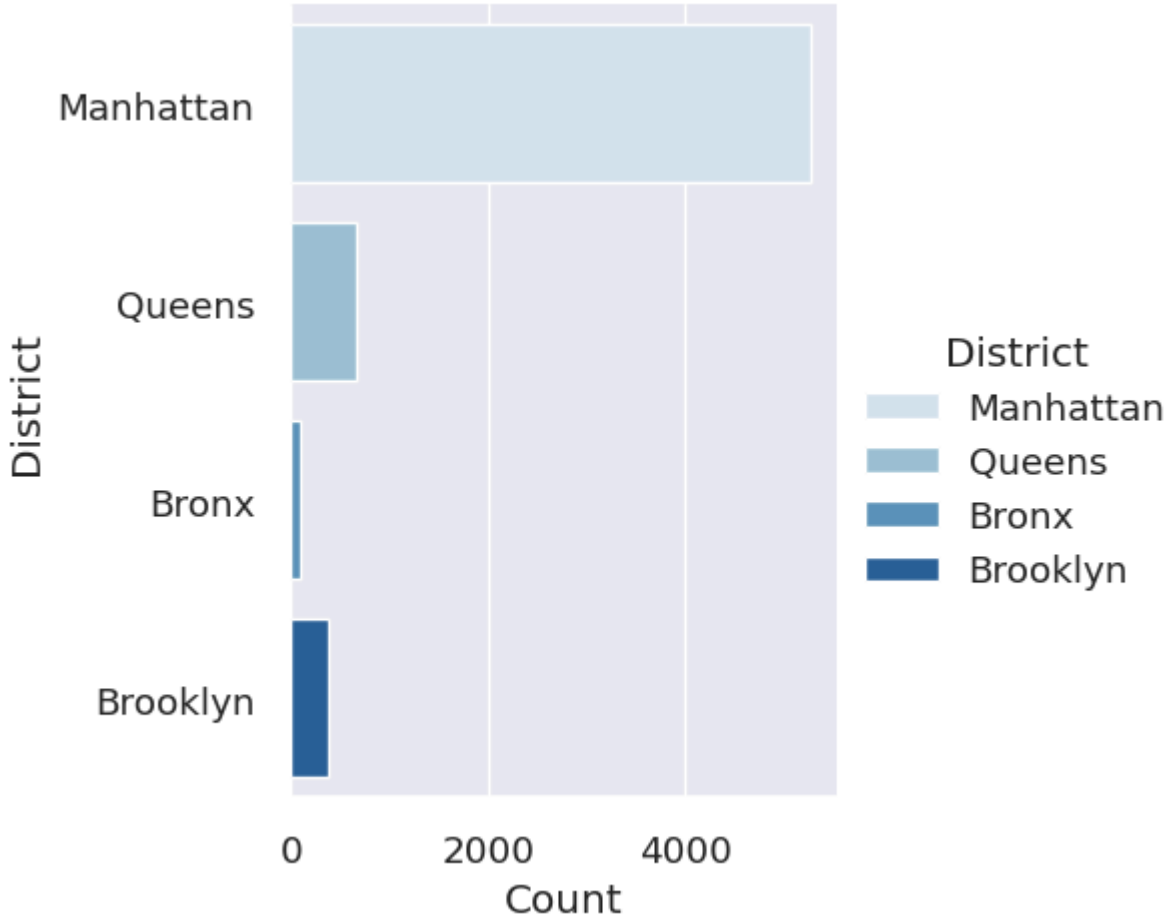
Parameter for setting the colour palette `palette`

In [ ]:
```
_ = sns.catplot(data=data, x='pickup_borough', hue='pickup_borough', kind='count', palette='Blues').set_axis_labels('District','Count').leg
```



Horizontal graph - data on the y-axis.

In [60]:
```
_ = sns.catplot(data=data, y='pickup_borough', hue='pickup_borough', kind='count', palette='Blues').set_axis_labels('Count', 'District').le
```



Pairs chart

Plotting relationships between pairs of data in a set.

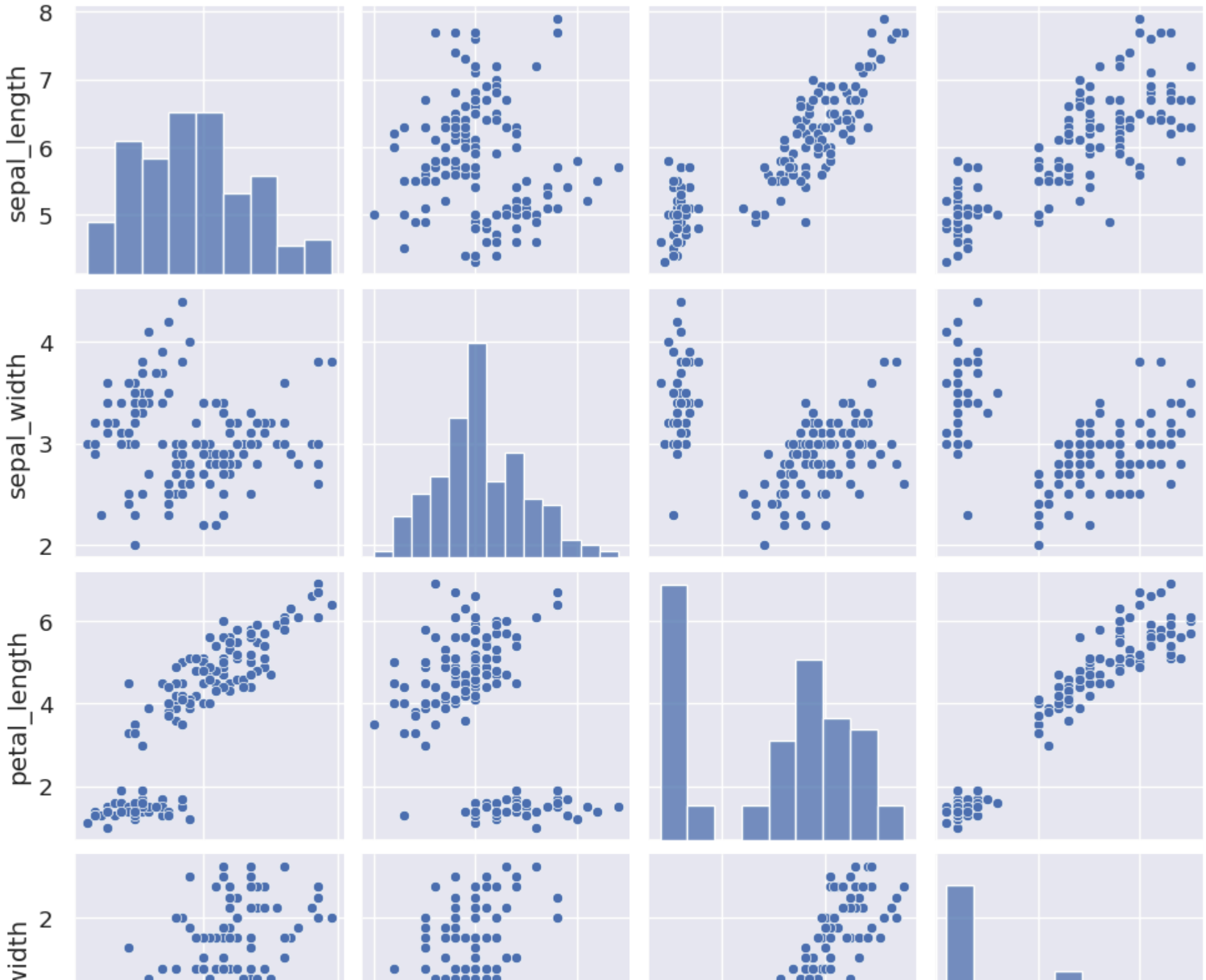
In [61]:
```
data_iris = sns.load_dataset('iris')
data_iris.head()
```

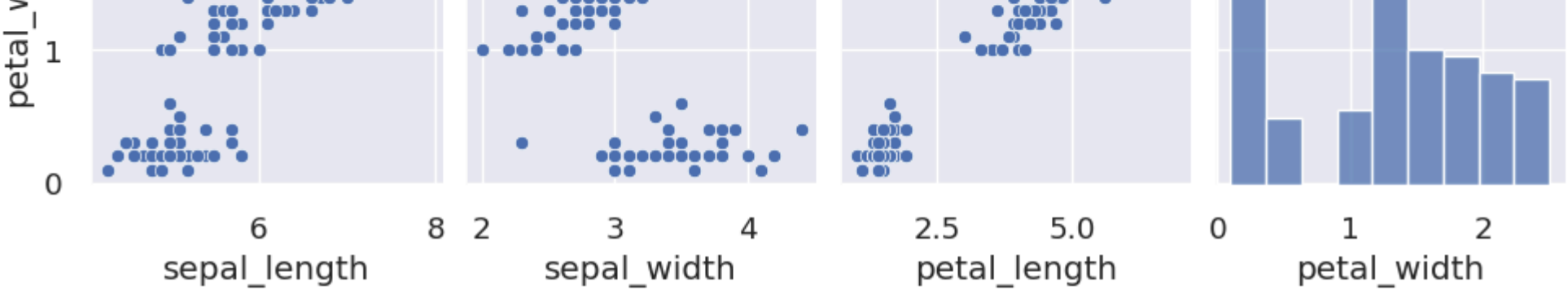
Out[61]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Drawing a graph using the `pairplot` function

In [ ]:
```
_= sns.pairplot(data_iris)
```


sepal_length
sepal_width
petal_length
width
8
7
6
5
4
3
2
6
4
2
2



Drawing a diagram of pairs by class.

In [62]:
```
abcd = sns.pairplot(data_iris, hue='species')
```


sepal_length
sepal_width
petal_length
petal_width
8
7
6
5
4
3
2
6
4
2
2
1
species
setosa
versicolor
virginica

Correlation matrix

The correlation matrix can be obtained using the `heatmap` command.

In [63]:
```
iris_correlation = data_iris.iloc[:,:4].corr()
_ = sns.heatmap(iris_correlation, annot=True, cmap='Blues').set(title='Correlation matrix')
```