Numpy

```
import numpy as np
```

```
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

```
[1 2 3 4 5]
```

```
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2)
```

```
[[1 2 3]
 [4 5 6]]
```

```
arr3 = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(arr3)
```

```
[[[1 2 3]
  [4 5 6]]

 [[1 2 3]
  [4 5 6]]]
```

```
arr2.dtype
```

```
dtype('int64')
```

```
arr2.ndim
```

```
2
```

```
print(arr[1])
```

```
2
```

```
print(arr2[1, 0])
```

```
4
```

```
print(arr[2] + arr[3])
```

```
7
```

```
print(arr3[1, 0, 2])
```

```
3
```

```
arrSlice = np.array([1, 2, 3, 4, 5, 6, 7])
print([arrSlice[1:5]])
```

```
[array([2, 3, 4, 5])]
```

```
print([arrSlice[2:]])
```

```
[array([3, 4, 5, 6, 7])]
```

```
arrFloat = np.array([1.1, 2, 3, 4, 5.6, 6, 7])
arrFloat.dtype
```

```
dtype('float64')
```

```
arrStr = np.array(['afefe', 'b', 'c'])
arrStr.dtype
```

```
dtype('<U5')
```

```
arrCopy = arr.copy()
print(arrCopy)
```

```
[1 2 3 4 5]
```

```
arr[0] = 69

print(arr)

    [69  2  3  4  5]
```

```
print(arrCopy)

    [1 2 3 4 5]
```

```
arrView = arr.view()
print(arrView)

    [69  2  3  4  5]
```

```
arr[0] = 42
```

```
print(arr)
print(arrView)

    [42  2  3  4  5]
    [42  2  3  4  5]
```

```
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
arr2.shape

    (2, 3)
```

```
arr3.shape

    (2, 2, 3)
```

```
arrRes = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10 , 11 ,12])
```

```
arrRes.reshape(4, 3)

    array([[ 1,  2,  3],
           [ 4,  5,  6],
           [ 7,  8,  9],
           [10, 11, 12]])
```

```
arrRes.reshape(6, 2)

    array([[ 1,  2],
           [ 3,  4],
           [ 5,  6],
           [ 7,  8],
           [ 9, 10],
           [11, 12]])
```

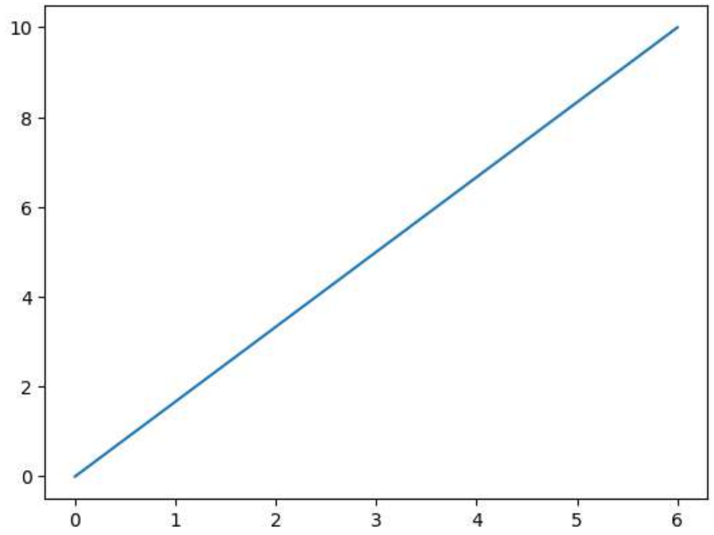
```
arrRes.reshape(12)

    array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

```
import time

x = range(10000000)
y = range(10000000, 20000000)
start_time = time.time()
c = [(x, y) for x, y in zip(x,y)]
print(time.time() - start_time)

    3.19294810295105
```

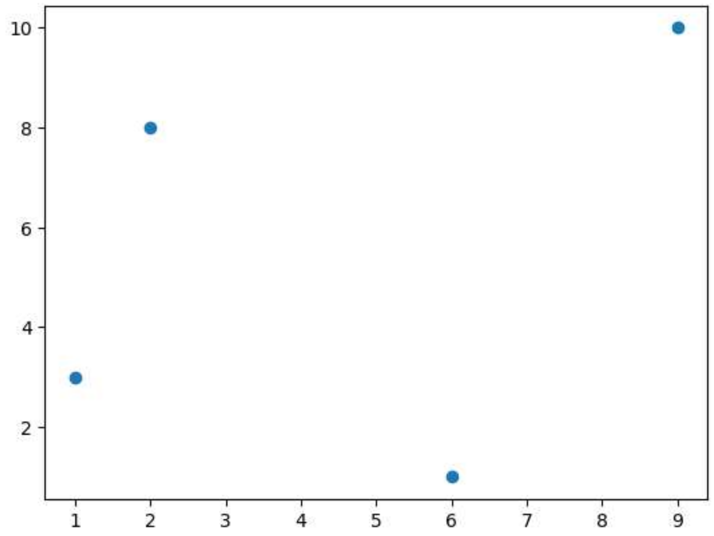
```
a = np.arange(10000000)
b = np.arange(10000000, 20000000)
start_time = time.time()
c = a + b
print(time.time() - start_time)

    0.5307042598724365
```

MatPlotLib

```
import matplotlib.pyplot as plt
xpoints = np.array([0, 6])
ypoints = np.array([0, 10])
plt.plot(xpoints, ypoints)
plt.show()
```

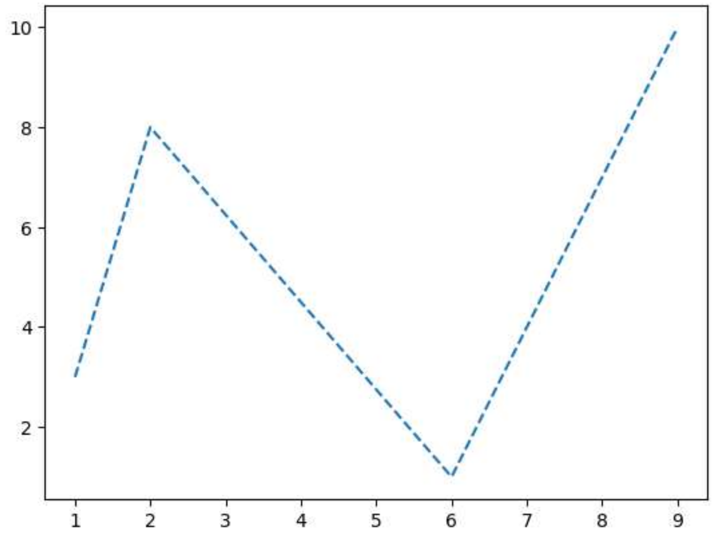


```
xpoints = np.array([1, 2, 6, 9])
ypoints = np.array([3,8, 1,  10])
plt.plot(xpoints, ypoints, 'o')
plt.show()
```

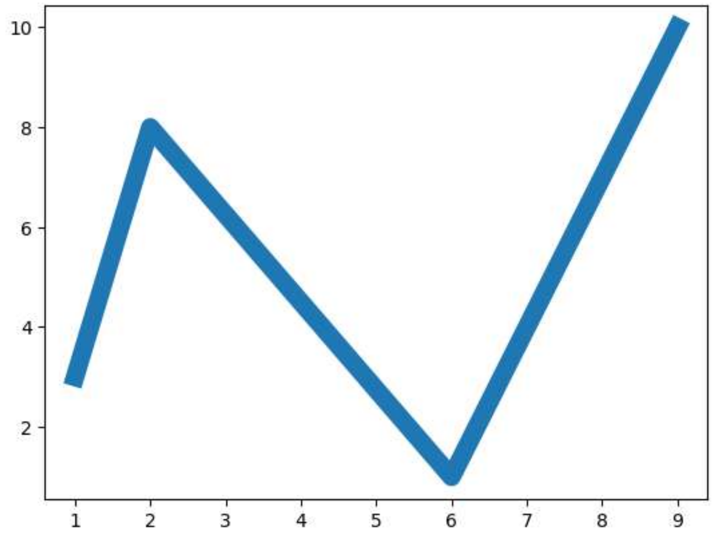


```
xpoints = np.array([1, 2, 6, 9])
ypoints = np.array([3,8, 1,  10])
plt.plot(xpoints, ypoints, 'o:r')
plt.show()
```

```
xpoints = np.array([1, 2, 6, 9])
ypoints = np.array([3,8, 1,  10])
plt.plot(xpoints, ypoints, linestyle = 'dashed')
plt.show()
```

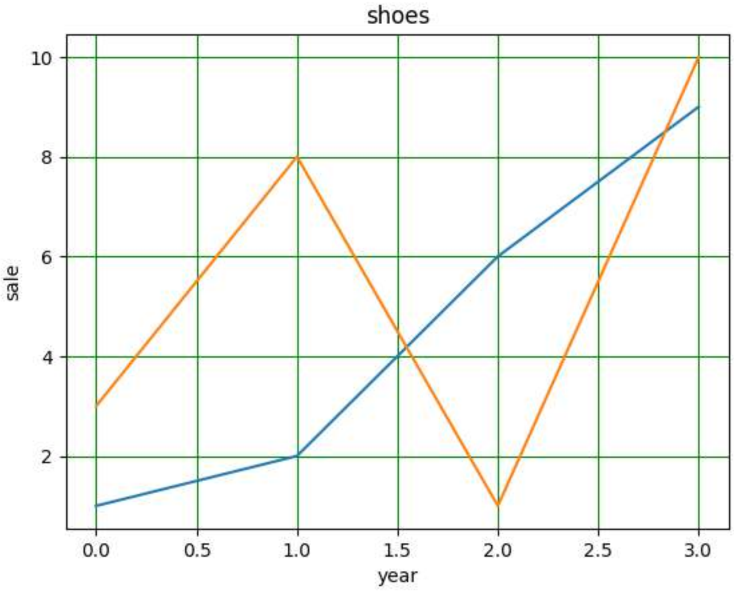


```
xpoints = np.array([1, 2, 6, 9])
ypoints = np.array([3,8, 1,  10])
plt.plot(xpoints, ypoints, linewidth = '10')
plt.show()
```



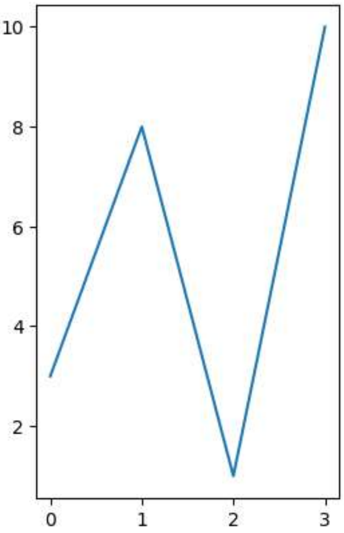
```
pts1 = np.array([1, 2, 6, 9])
pts2 = np.array([3,8, 1,  10])
plt.plot(pts1)
plt.plot(pts2)
plt.show()
```

```
pts1 = np.array([1, 2, 6, 9])
pts2 = np.array([3,8, 1,  10])
plt.plot(pts1)
plt.plot(pts2)
plt.xlabel('year')
plt.ylabel('sale')
plt.title('shoes')
plt.grid( color = 'green')
plt.show()
```



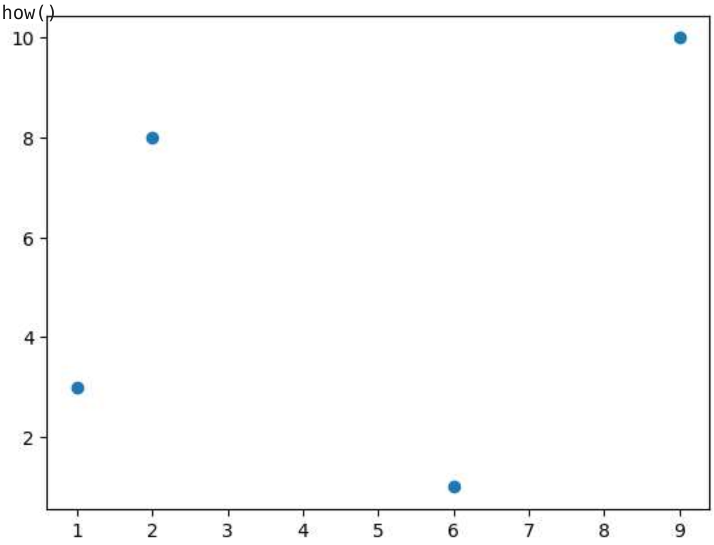
```
pts1 = np.array([1, 2, 6, 9])
pts2 = np.array([3,8, 1,  10])
plt.plot(pts1)
plt.subplot(1, 2, 1)
plt.plot(pts2)
plt.subplot(1, 2, 1)
plt.show()
```

```
<ipython-input-88-de7b5c53ff43>:4: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.
  plt.subplot(1, 2, 1)
```

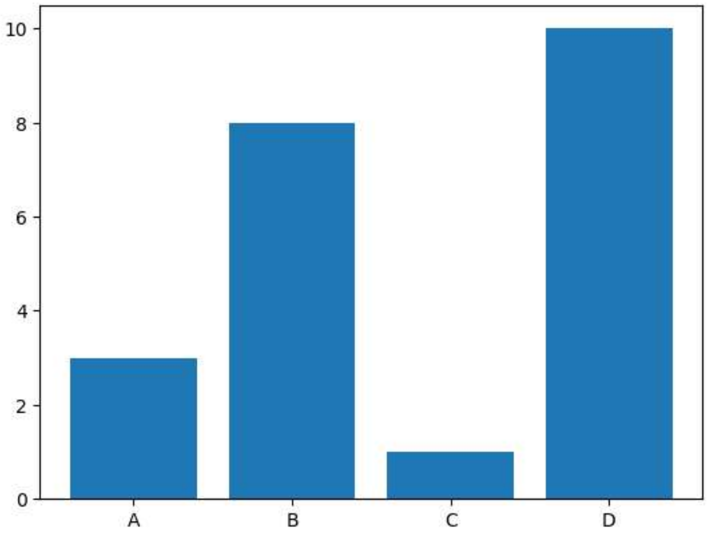


```
pts1 = np.array([1, 2, 6, 9])
pts2 = np.array([3,8, 1,  10])
```

```
plt.scatter(pts1, pts2)
plt.show()
```



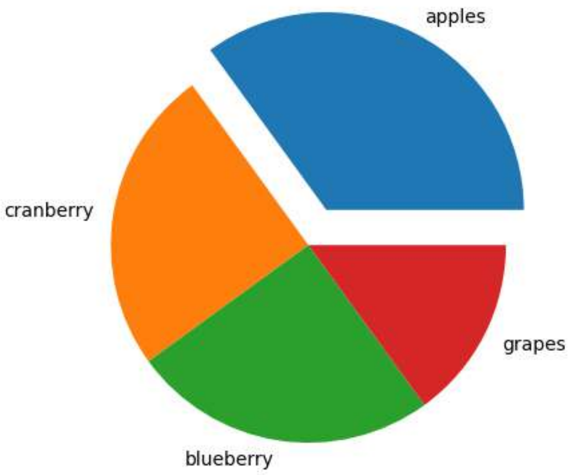
```
pts1 = np.array(['A', 'B', 'C', 'D'])
pts2 = np.array([3,8, 1,  10])
plt.bar(pts1, pts2)
plt.show()
```



```
x = np.random.normal(170, 10, 250)
print(x)
plt.hist(x)
plt.show()
```

```
[158.79310436 167.87776948 171.10746883 166.44374438 160.76750808
 177.70378844 169.28156312 164.90795985 164.60289712 184.03362931
 179.71049404 161.01111289 165.00431745 178.85908756 163.84843379
 181.26159904 170.13521436 168.74840344 183.46813674 174.30106653
 162.25258426 167.14439805 161.32618264 160.63435361 174.86073006
 185.90696408 160.19490158 156.81791609 156.5677738  176.22871062
 151.46502306 158.94202298 165.68359349 174.50108987 167.23205876
 179.46405493 156.93083223 176.98424934 174.23828737 177.92269614
 180.06158436 159.40730626 165.71469529 178.6902437  179.33849422
 174.83709449 169.67670933 183.2512808  180.46814824 163.43300639
 166.99692933 172.63303282 164.53209898 149.71419711 156.65430195
 175.30307911 181.31591763 182.5918816  177.47150947 175.37312369
 176.69113971 157.26946447 171.7661362  180.25637277 160.59594604
 183.69889227 178.27794639 174.40151836 172.40021448 175.22018679
 153.20837656 181.16343228 167.21534757 185.36866712 188.76945705
 188.32962918 168.88799322 155.81938902 152.6947564  159.84405502
 161.83261474 178.28306064 163.23470591 179.66842394 162.25744431
 181.50718538 155.98371009 170.88522797 168.59272451 181.61293344
 158.49946338 173.83967051 187.7731072  176.69021483 171.17810308
 168.38139761 172.56805737 193.26400839 165.22767461 152.95861181
 157.40770702 173.63212652 173.79309797 168.63214699 163.68707627
 178.59231715 185.73049467 171.81925576 161.50317147 171.5710294
 162.01339888 173.73073397 169.47634029 171.72446135 171.75829271
 159.16009408 166.54063909 164.13554645 175.63744921 169.35917657
 172.48214174 174.42926466 164.73168445 158.91959217 163.32661476
 164.46842869 180.11166154 166.53681658 163.64622362 177.70375395
 165.2691287  181.67026968 156.90793226 168.13634682 169.50252806
 188.30455854 151.44821809 177.55901532 189.15593837 181.72506229
 180.90829336 154.86240916 152.05948494 195.19963827 173.63045087
 182.01667787 182.0207123  159.8560867  192.33170148 155.13190738
 167.87109581 154.85377568 159.22189608 169.45799896 165.4331454
 173.18972051 166.56648009 184.25141941 173.9515716  154.94429298
 173.84148832 178.83267162 161.69431983 186.12288622 176.24333952
 169.42622033 170.46073038 171.5813003  179.3171303  172.53772401
 146.42369648 163.4975429  173.79744366 168.22005004 173.5485857
 160.76935896 167.98902456 157.71315363 159.03377145 172.50974461
 172.38664066 179.82840375 170.81246316 162.29734479 159.79909973
 159.71852151 167.39245949 177.51350717 174.91901516 155.46363045
 173.95689033 179.58730563 163.23086128 163.01317765 172.94342217
 177.80435707 175.33711314 182.83728384 166.70275105 169.79678105
 165.39900181 167.73356048 169.33879874 159.93582    146.39960163
 178.4823661  163.61824749 182.94739365 153.52583476 178.99432944
 177.84084195 196.53239627 171.7625539  176.93548313 170.50238787
 158.53825244 181.36099595 169.25763053 191.31027577 167.44321476
 174.25619074 171.29903958 180.58598443 146.82219797 165.24435306
 160.1619981  182.53911438 183.86995867 169.39951417 181.2329769
 166.04894514 165.07412383 184.8279867  167.7833026  159.57494286
 186.24033735 193.01580975 176.02444719 157.82789416 171.00280637
```

```
y = np.array([35, 25, 25, 15])
mylbl = ["apples", "cranberry", "blueberry", "grapes"]
myexplode = [0.2, 0, 0, 0]
plt.pie(y, labels = mylbl, explode = myexplode)
plt.show()
```



Pandas

```
import pandas as pd
```

```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/datasets/data.csv')
```

```
df.head()
```

	size	price
0	300	1200000
1	325	1300000
2	350	1400000
3	375	1500000
4	400	1600000

```
df
```

	size	price
0	300	1200000
1	325	1300000
2	350	1400000
3	375	1500000
4	400	1600000
...	...	...
94	2650	10600000
95	2675	10700000
96	2700	10800000
97	2725	10900000
98	2750	11000000

99 rows × 2 columns

```
df.sample(10)
```

	size	price
30	1050	4200000
3	375	1500000
13	625	2500000
69	2025	8100000
21	825	3300000
66	1950	7800000
90	2550	10200000
68	2000	8000000
22	850	3400000
18	750	3000000

```
df.price
```

```
0      1200000
1      1300000
2      1400000
3      1500000
4      1600000
         ...
94    10600000
95    10700000
96    10800000
97    10900000
98    11000000
Name: price, Length: 99, dtype: int64
```

```
df.size
```

```
198
```

```
df.rename(columns={"size": "size_house"}, inplace=True)
```

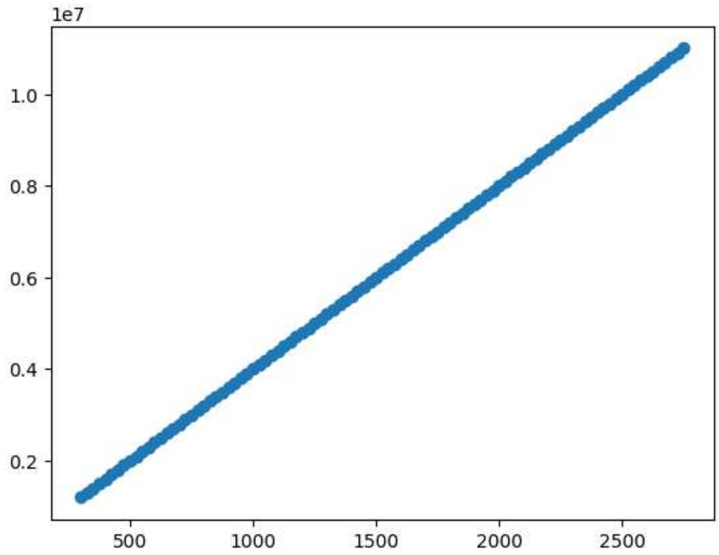
```
df
```

	size_house	price
0	300	1200000
1	325	1300000
2	350	1400000
3	375	1500000
4	400	1600000
...	...	...
94	2650	10600000
95	2675	10700000
96	2700	10800000
97	2725	10900000
98	2750	11000000

99 rows × 2 columns

```
plt.scatter(df.size_house   , df.price)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



```
import pandas as pd
```

```
from google.colab import drive
```

```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/datasets/titanic.csv')
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs	female	38.0	1	0	PC 17599

```
df.shape
```

```
(891, 12)
```

```
df.tail()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	1
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	3

```
df.sample(5)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
707	708	1	1	Calderhead, Mr. Edward Pennington	male	42.0	0	0	174
155	156	0	1	Williams, Mr. Charles Duane	male	51.0	0	1	175
				de					

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14

```
df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

```
df.corr
```

```
<bound method DataFrame.corr of      PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                   Name     Sex   Age  SibSp  \
0                               Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                                Heikkinen, Miss. Laina  female  26.0      0
3          Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                              Allen, Mr. William Henry    male  35.0      0
..                                                  ...     ...   ...    ...
886                               Montvila, Rev. Juozas    male  27.0      0
887                        Graham, Miss. Margaret Edith  female  19.0      0
888            Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                               Behr, Mr. Karl Howell    male  26.0      0
890                                 Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     ...               ...      ...   ...      ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]>
```

```
df.corr()
```

```
<ipython-input-66-2f6f6606aa2c>:1: FutureWarning: The default value of numeri
  df.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fa
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.0126
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.2573
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.5495
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.0960

```
df.corr()['Survived']
```

```
<ipython-input-67-57d70bb92b5b>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In
  df.corr()['Survived']
PassengerId   -0.005007
Survived       1.000000
Pclass        -0.338481
Age           -0.077221
SibSp         -0.035322
Parch          0.081629
Fare           0.257307
Name: Survived, dtype: float64
```

Univariate Analysis

```
import seaborn as sns
```

```
df['Survived'].value_counts()
```

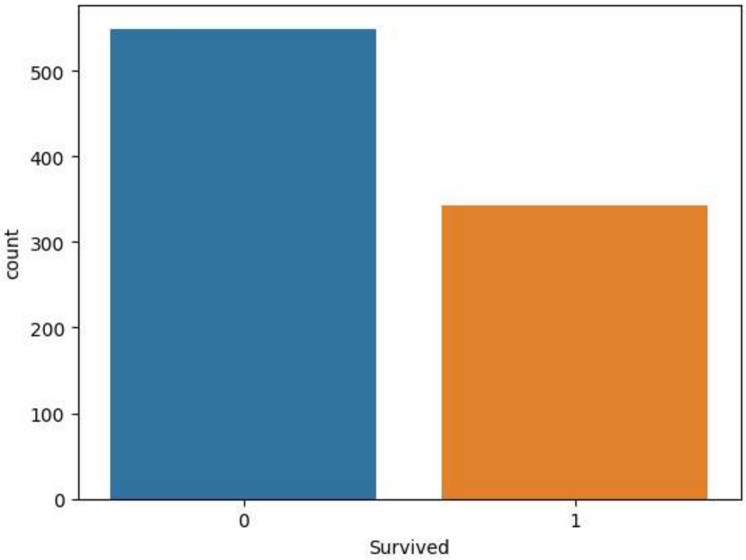
```
0    549
1    342
Name: Survived, dtype: int64
```

```
df['Sex'].value_counts()
```

```
male      577
female    314
Name: Sex, dtype: int64
```

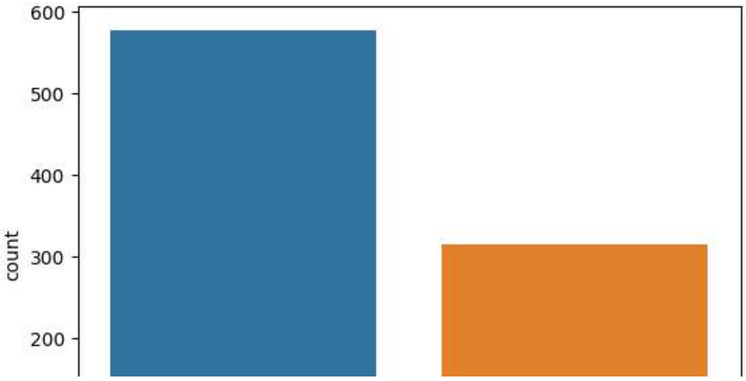
```
sns.countplot(data = df, x = 'Survived')
```

```
<Axes: xlabel='Survived', ylabel='count'>
```



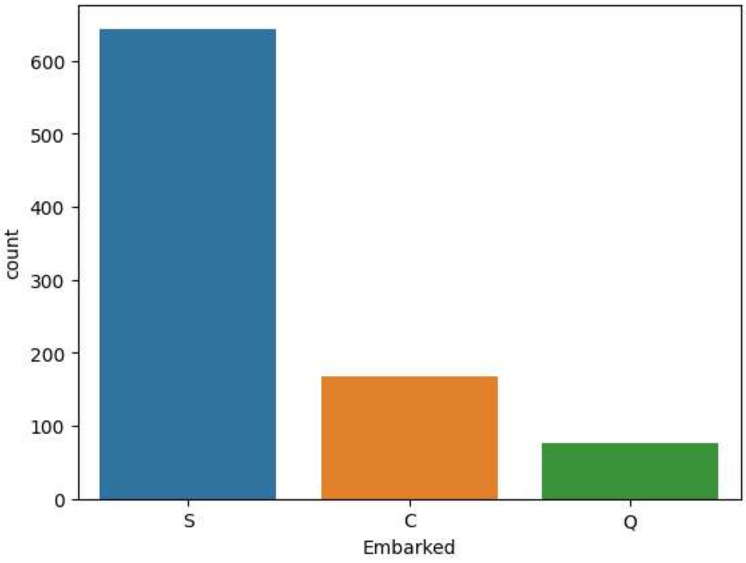
```
sns.countplot(data = df, x = 'Sex')
```

```
<Axes: xlabel='Sex', ylabel='count'>
```



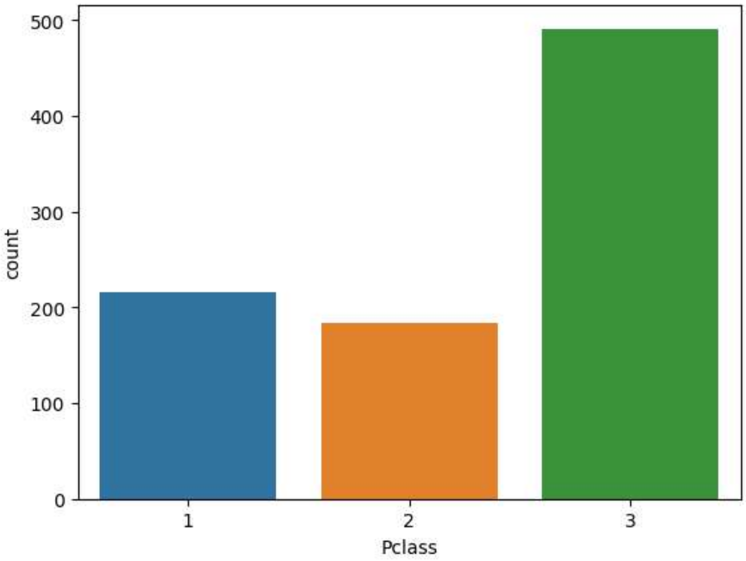
```
sns.countplot(data = df, x = 'Embarked')
```

```
<Axes: xlabel='Embarked', ylabel='count'>
```



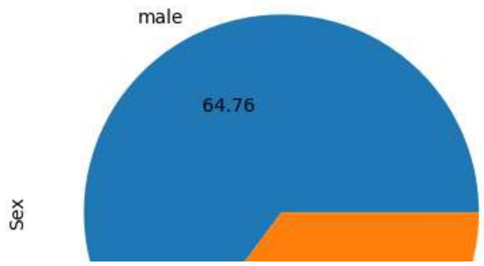
```
sns.countplot(data = df, x = 'Pclass')
```

```
<Axes: xlabel='Pclass', ylabel='count'>
```



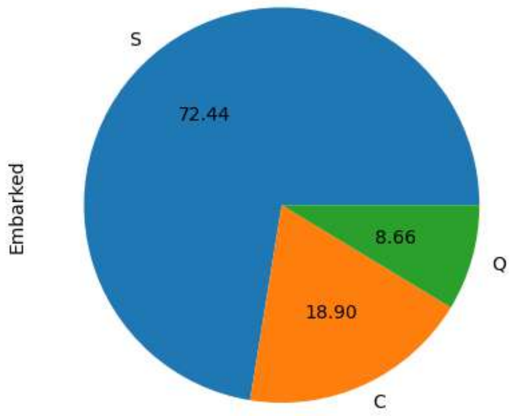
```
df['Sex'].value_counts().plot(kind='pie', autopct = '%.2f')
```

```
    <Axes: ylabel='Sex'>
```



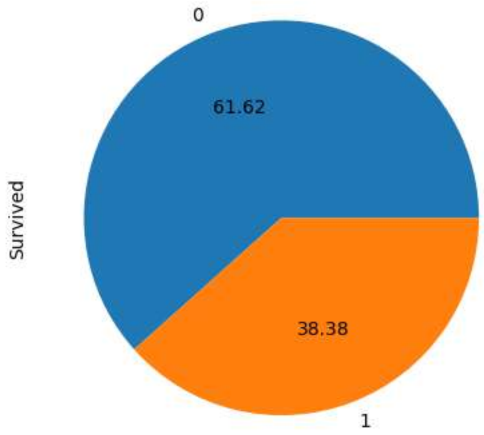
```
df['Embarked'].value_counts().plot(kind='pie', autopct = '%.2f')
```

```
    <Axes: ylabel='Embarked'>
```



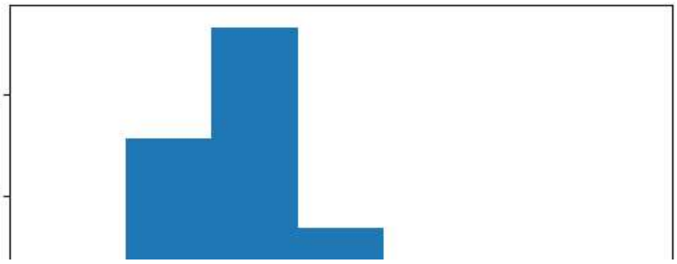
```
df['Survived'].value_counts().plot(kind='pie', autopct = '%.2f')
```

```
    <Axes: ylabel='Survived'>
```



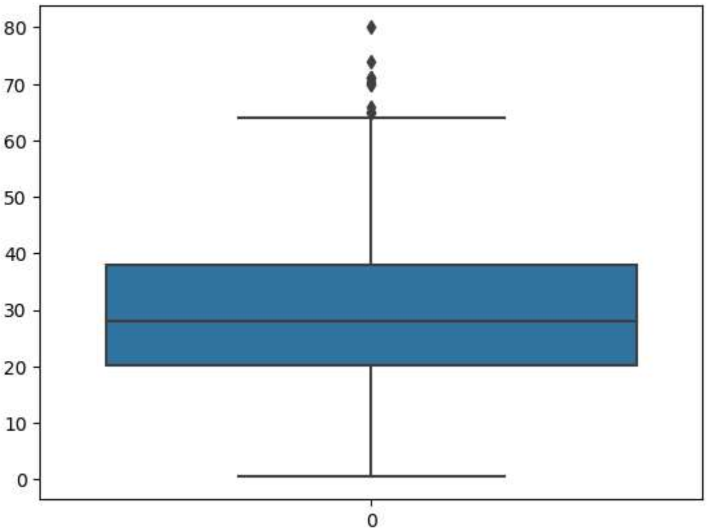
```
import matplotlib.pyplot as plt
```

```
plt.hist(df['Age'], bins=7)
plt.show()
```



```
sns.boxplot(df['Age'])
```

```
    <Axes: >
```



Bivariate analysis

```
df2 = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/datasets/tips.csv')
```

```
df2.head(

)
```

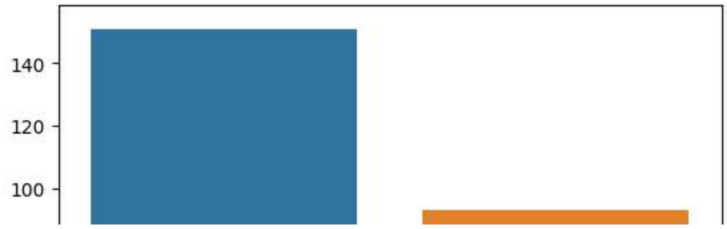
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
df2.shape
```

```
    (244, 7)
```

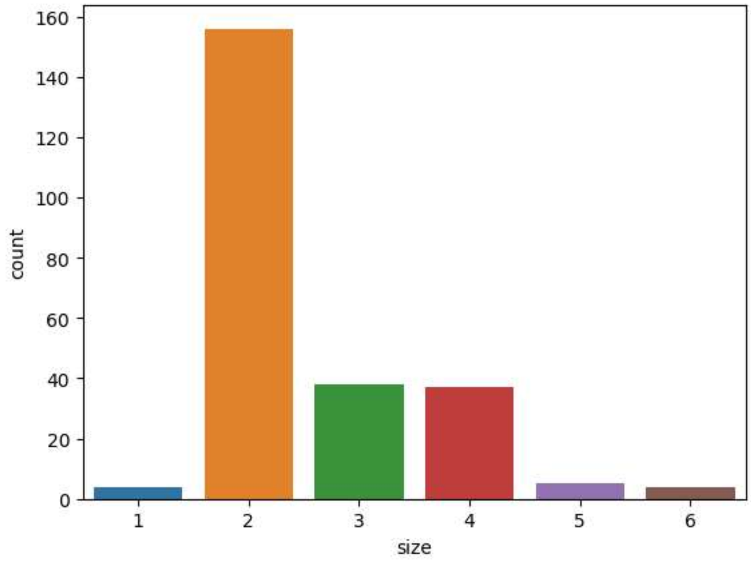
```
sns.countplot(data = df2, x = 'smoker')
```

```
<Axes: xlabel='smoker', ylabel='count'>
```



```
sns.countplot(data = df2, x = 'size')
```

```
<Axes: xlabel='size', ylabel='count'>
```



```
df2.describe()
```

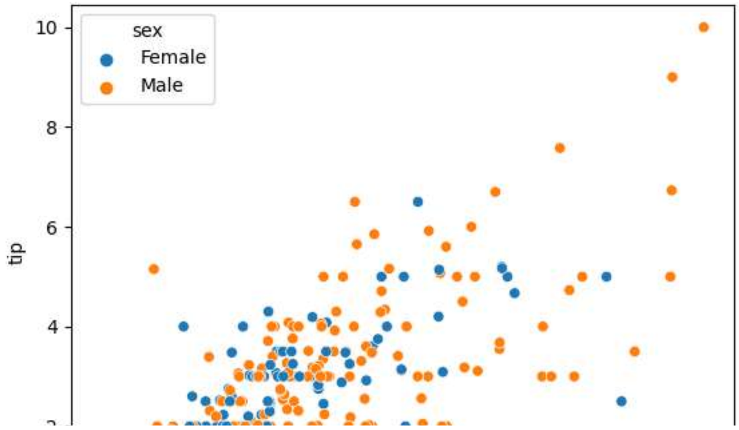
	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    object
 3   smoker      244 non-null    object
 4   day         244 non-null    object
 5   time        244 non-null    object
 6   size        244 non-null    int64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

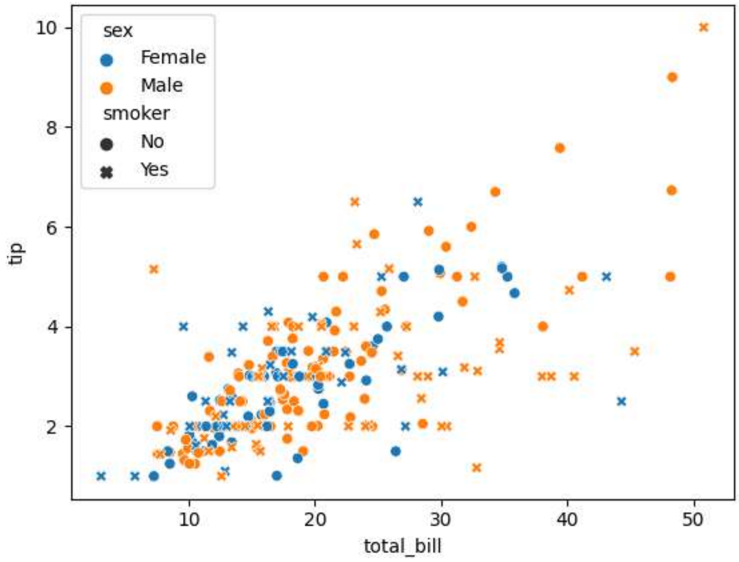
```
#bivariate
sns.scatterplot(data=df2, x='total_bill', y='tip', hue='sex')
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```



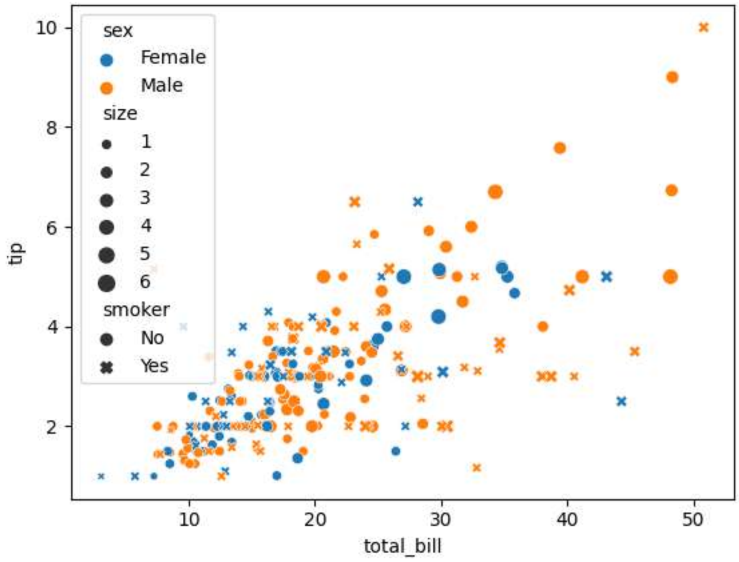
```
#bivariate
sns.scatterplot(data=df2, x='total_bill', y='tip', hue='sex', style='smoker')
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```



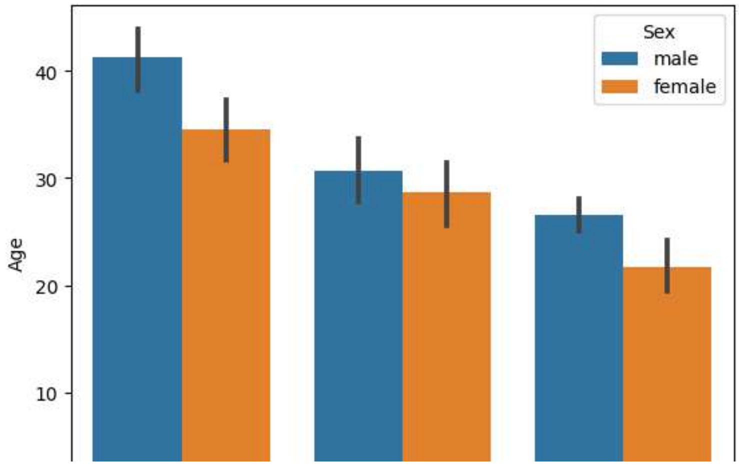
```
#bivariate
sns.scatterplot(data=df2, x='total_bill', y='tip', hue='sex', style='smoker', size='size')
```

```
<Axes: xlabel='total_bill', ylabel='tip'>
```



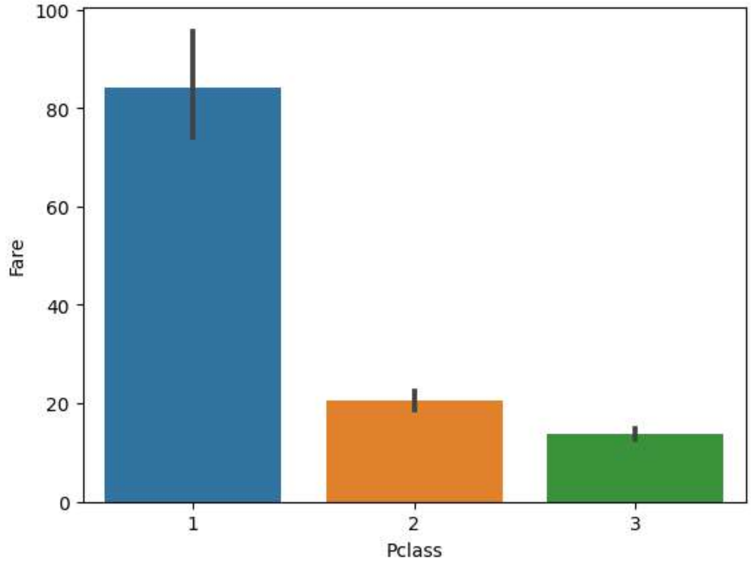
```
sns.barplot(data=df, x='Pclass', y='Age', hue='Sex')
```

```
<Axes: xlabel='Pclass', ylabel='Age'>
```



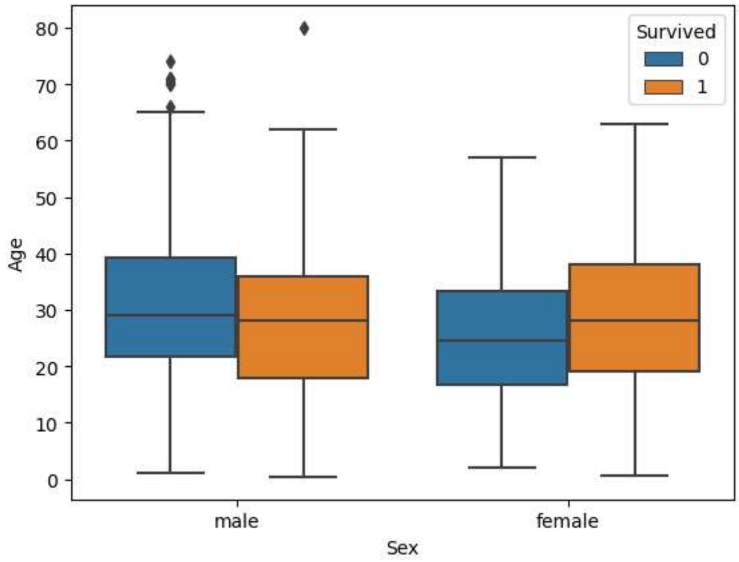
```
sns.barplot(data=df, x='Pclass', y='Fare')
```

```
<Axes: xlabel='Pclass', ylabel='Fare'>
```



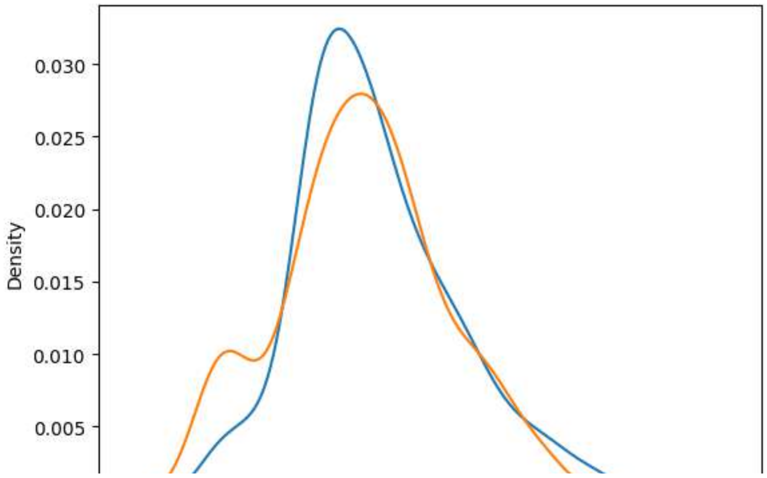
```
sns.boxplot(data=df, x='Sex', y='Age', hue='Survived')
```

```
<Axes: xlabel='Sex', ylabel='Age'>
```



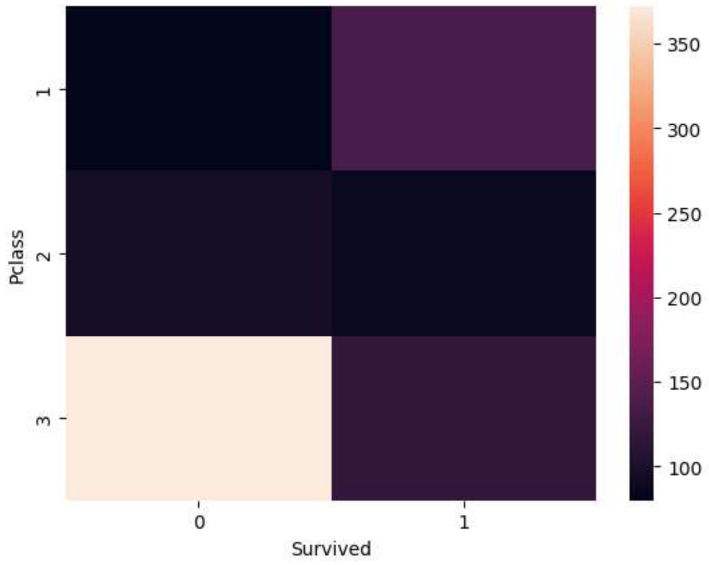
```
sns.distplot(df[df['Survived'] == 0]['Age'],hist=False)
sns.distplot(df[df['Survived'] == 1]['Age'],hist=False)
```

```
<ipython-input-94-da45b6ba878b>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function wit
similar flexibility) or `kdeplot` (an axes-level function for kernel densit

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[df['Survived'] == 0]['Age'],hist=False)
<ipython-input-94-da45b6ba878b>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function wit
similar flexibility) or `kdeplot` (an axes-level function for kernel densit

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[df['Survived'] == 1]['Age'],hist=False)
<Axes: xlabel='Age', ylabel='Density'>
```



```
sns.heatmap(pd.crosstab(df['Pclass'],df['Survived']))
```

```
<Axes: xlabel='Survived', ylabel='Pclass'>
```



```
(df.groupby('Sex').mean()['Survived']*100)
```

```
<ipython-input-96-d763aad4bce8>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecat
  (df.groupby('Sex').mean()['Survived']*100)
Sex
female    74.203822
male      18.890815
Name: Survived, dtype: float64
```

```
(df.groupby('Embarked').mean()['Survived']*100)
```

```
<ipython-input-97-77b5216c7294>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecat
  (df.groupby('Embarked').mean()['Survived']*100)
```

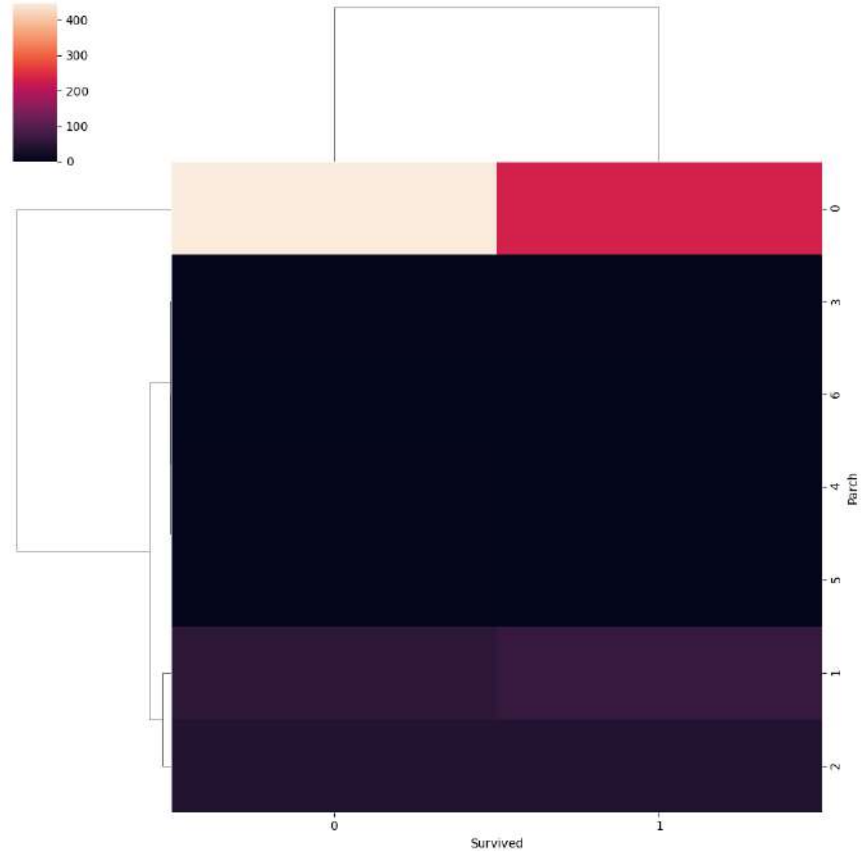
```
Embarked
C    55.357143
Q    38.961039
S    33.695652
Name: Survived, dtype: float64
```

```
pd.crosstab(df['Parch'],df['Survived'])
```

Survived	0	1
Parch		
0	445	233
1	53	65
2	40	40
3	2	3
4	4	0
5	4	1
6	1	0

```
sns.clustermap(pd.crosstab(df['Parch'], df['Survived']))
```

```
<seaborn.matrix.ClusterGrid at 0x7f811cc2f310>
```



● ×