Import Modules

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [3]:
```
#loading dATA SET
train = pd.read_csv("TITANIC.csv")
test = pd.read_csv("TITANIC.csv")
train.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [4]:
```
## statistical info
train.describe()
```

Out[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	0.363636	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.481622	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	0.000000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	0.000000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	1.000000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000	512.329200

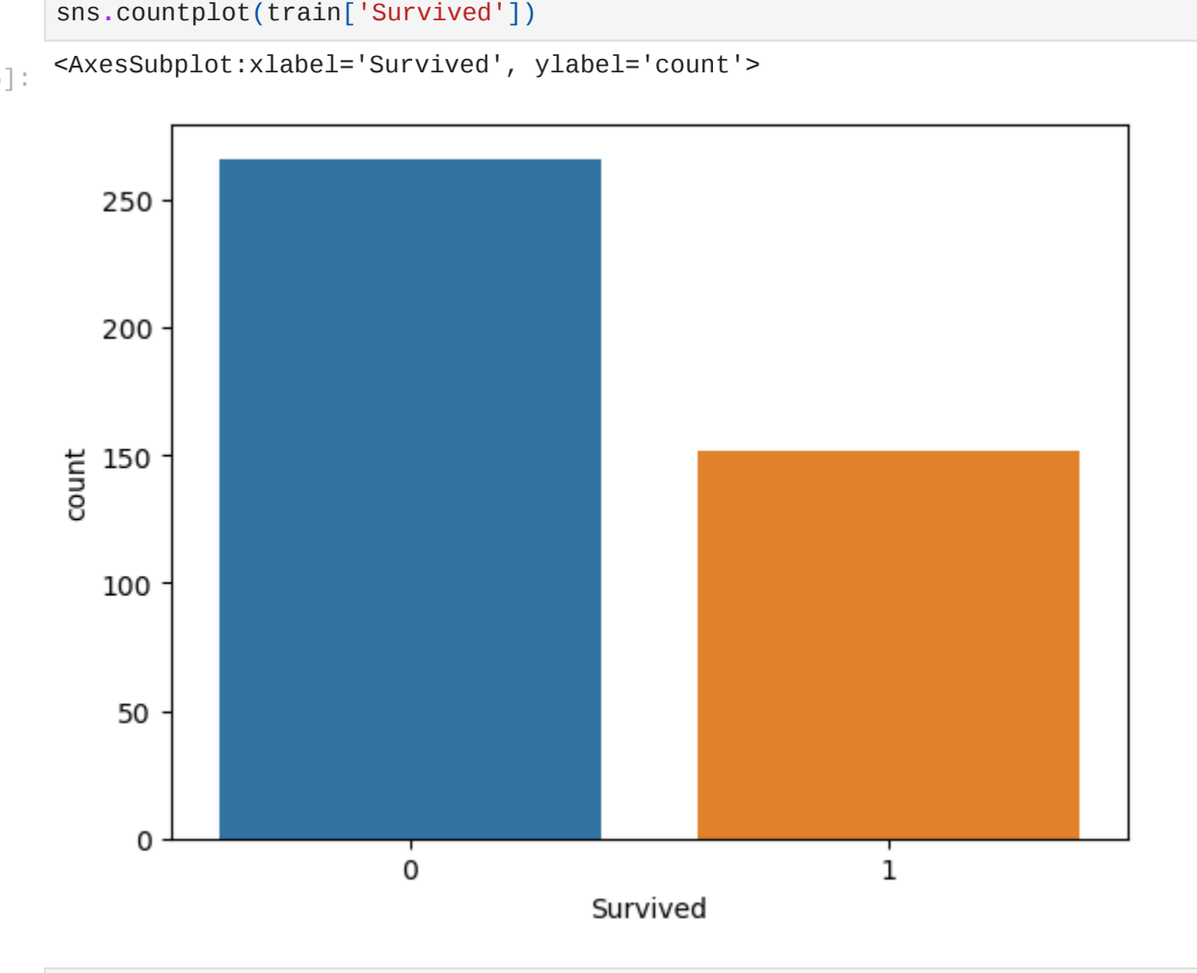
In [5]:
```
## datatype info
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Survived     418 non-null    int64
 2   Pclass       418 non-null    int64
 3   Name         418 non-null    object
 4   Sex          418 non-null    object
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64
 7   Parch        418 non-null    int64
 8   Ticket       418 non-null    object
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object
 11  Embarked     418 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

In [ ]:

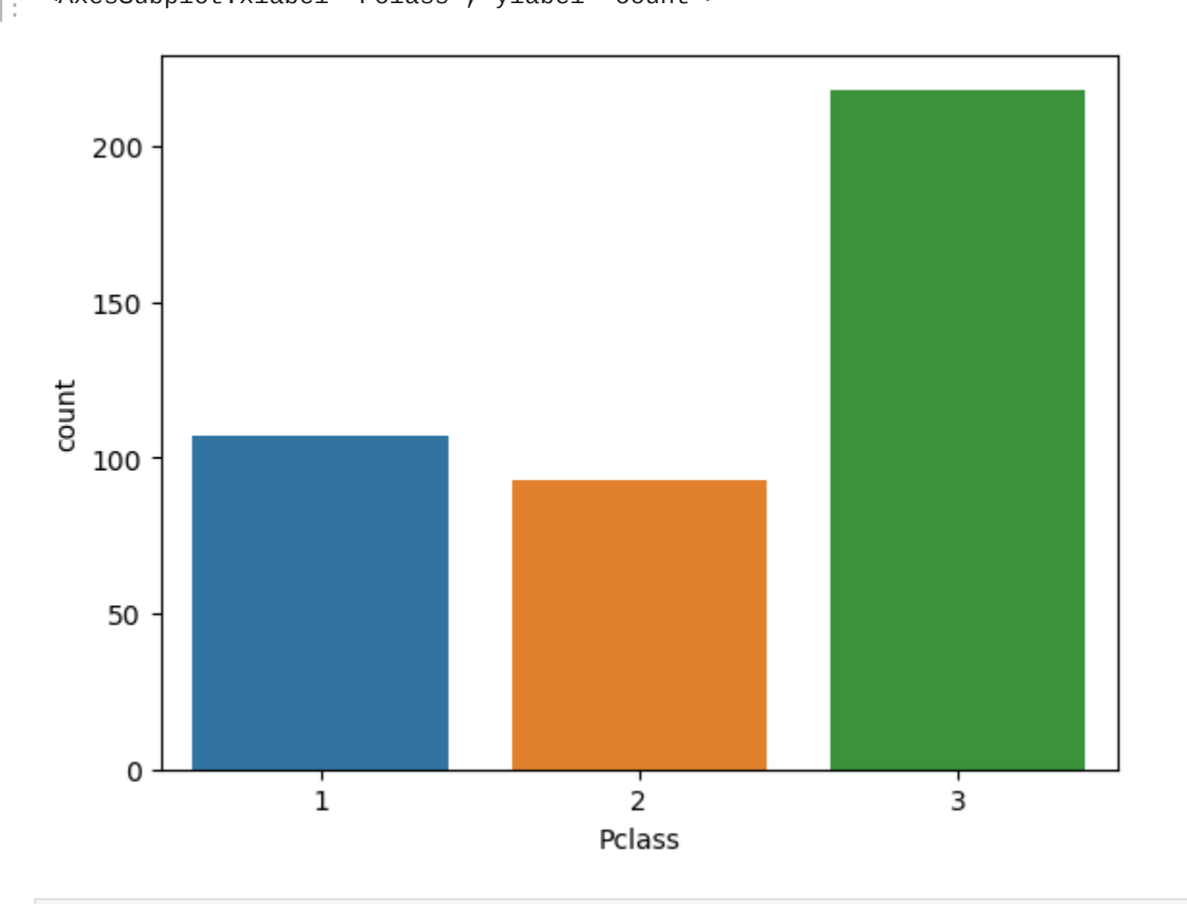
In [6]:
```
#EXPLORATORY DATA ANALYSIS
## categorical attributes
sns.countplot(train['Survived'])
```

Out[6]: <AxesSubplot:xlabel='Survived', ylabel='count'>



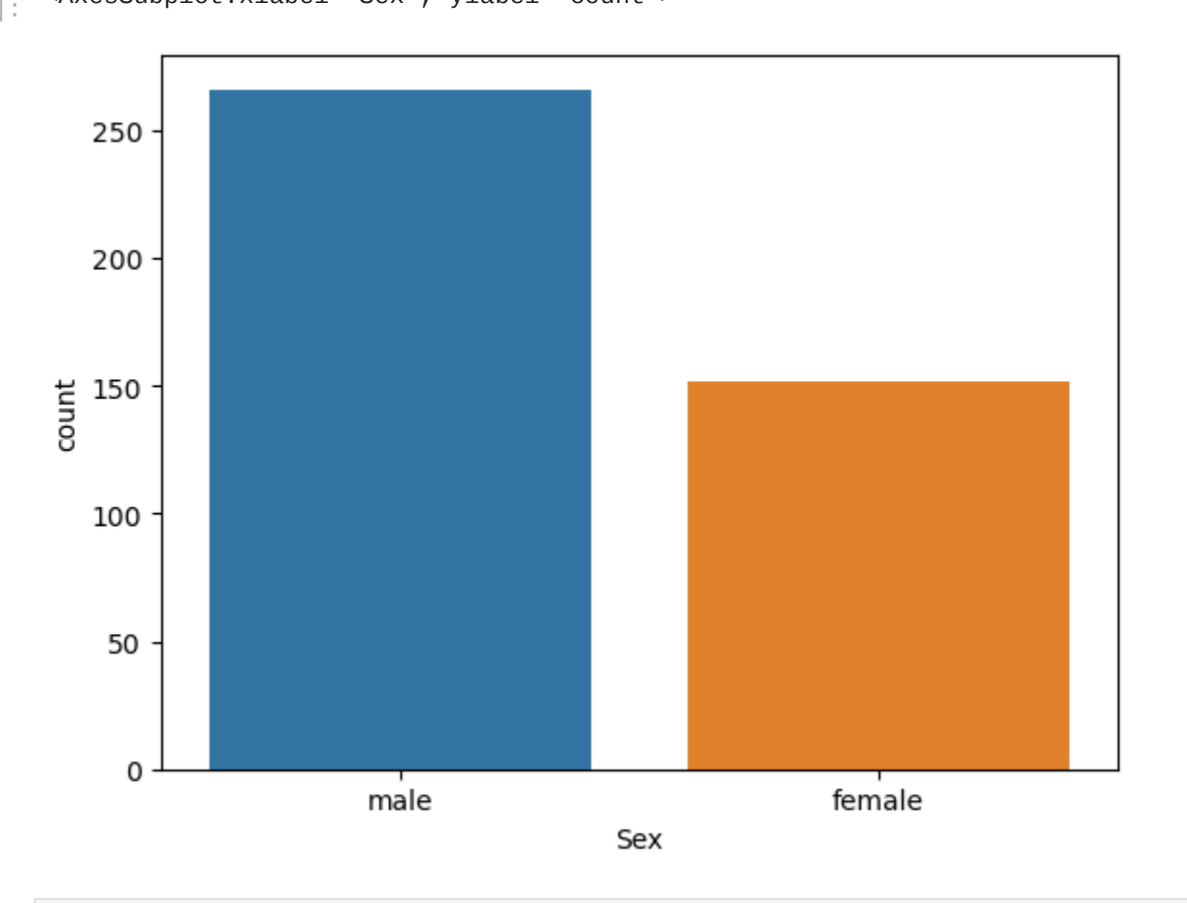
In [7]:
```
sns.countplot(train['Pclass'])
```

Out[7]: <AxesSubplot:xlabel='Pclass', ylabel='count'>



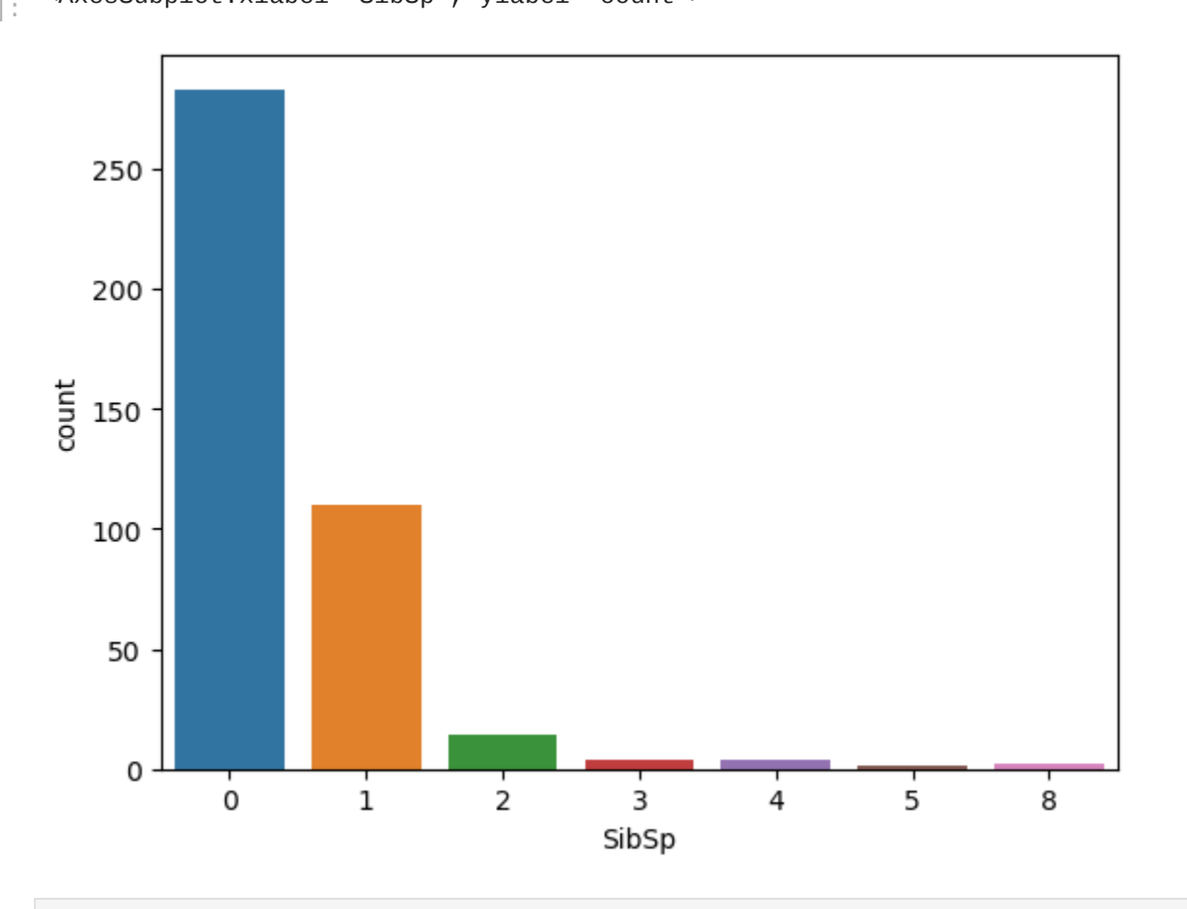
In [8]:
```
sns.countplot(train['Sex'])
```

Out[8]: <AxesSubplot:xlabel='Sex', ylabel='count'>



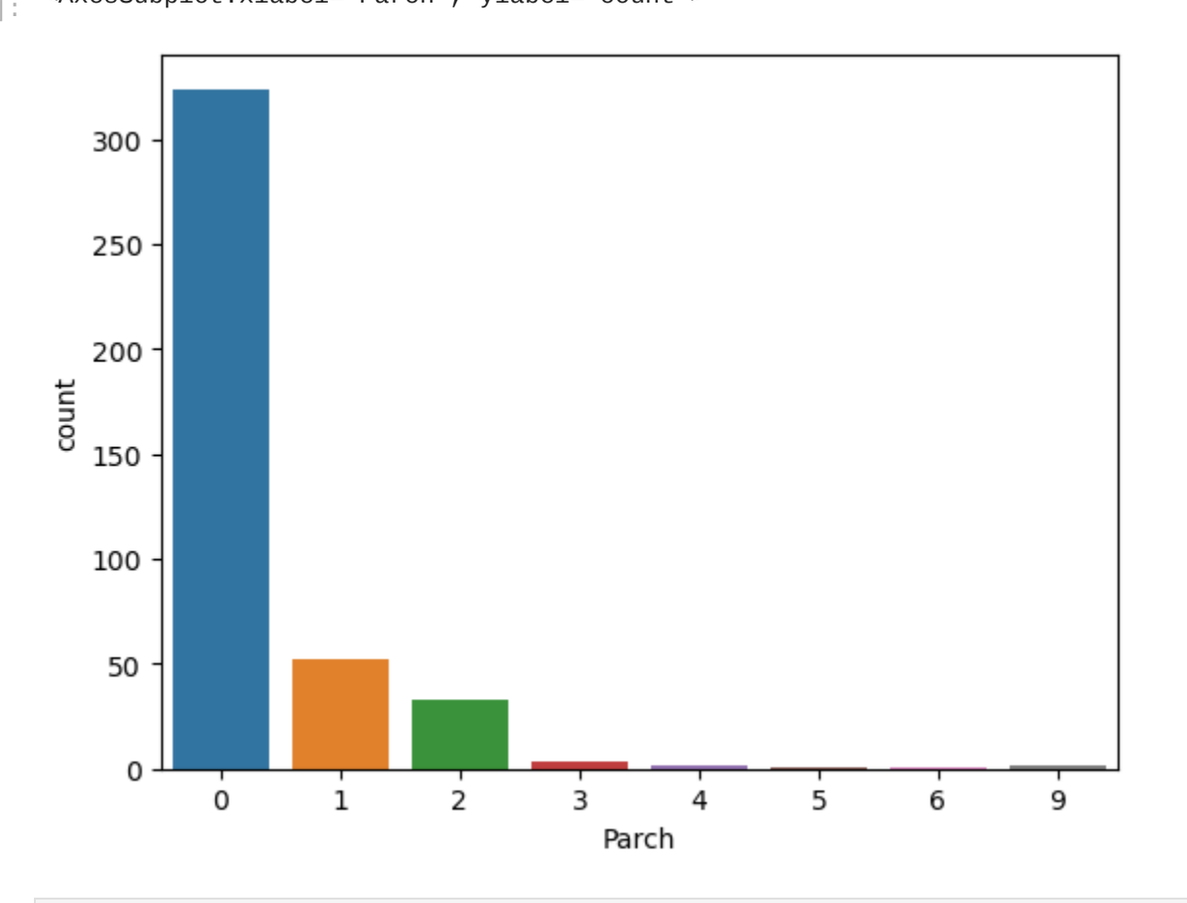
In [9]:
```
sns.countplot(train['SibSp'])
```

Out[9]: <AxesSubplot:xlabel='SibSp', ylabel='count'>



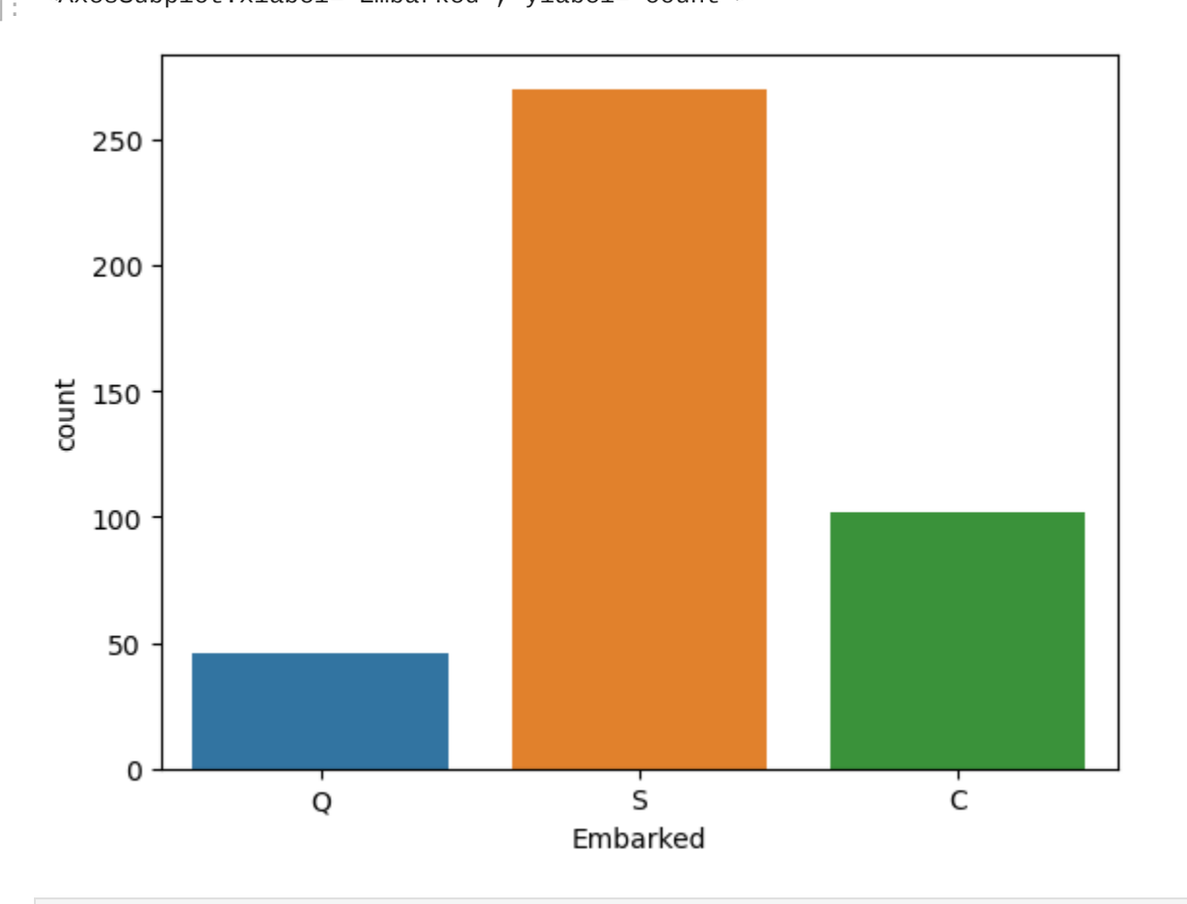
In [10]:
```
sns.countplot(train['Parch'])
```

Out[10]: <AxesSubplot:xlabel='Parch', ylabel='count'>



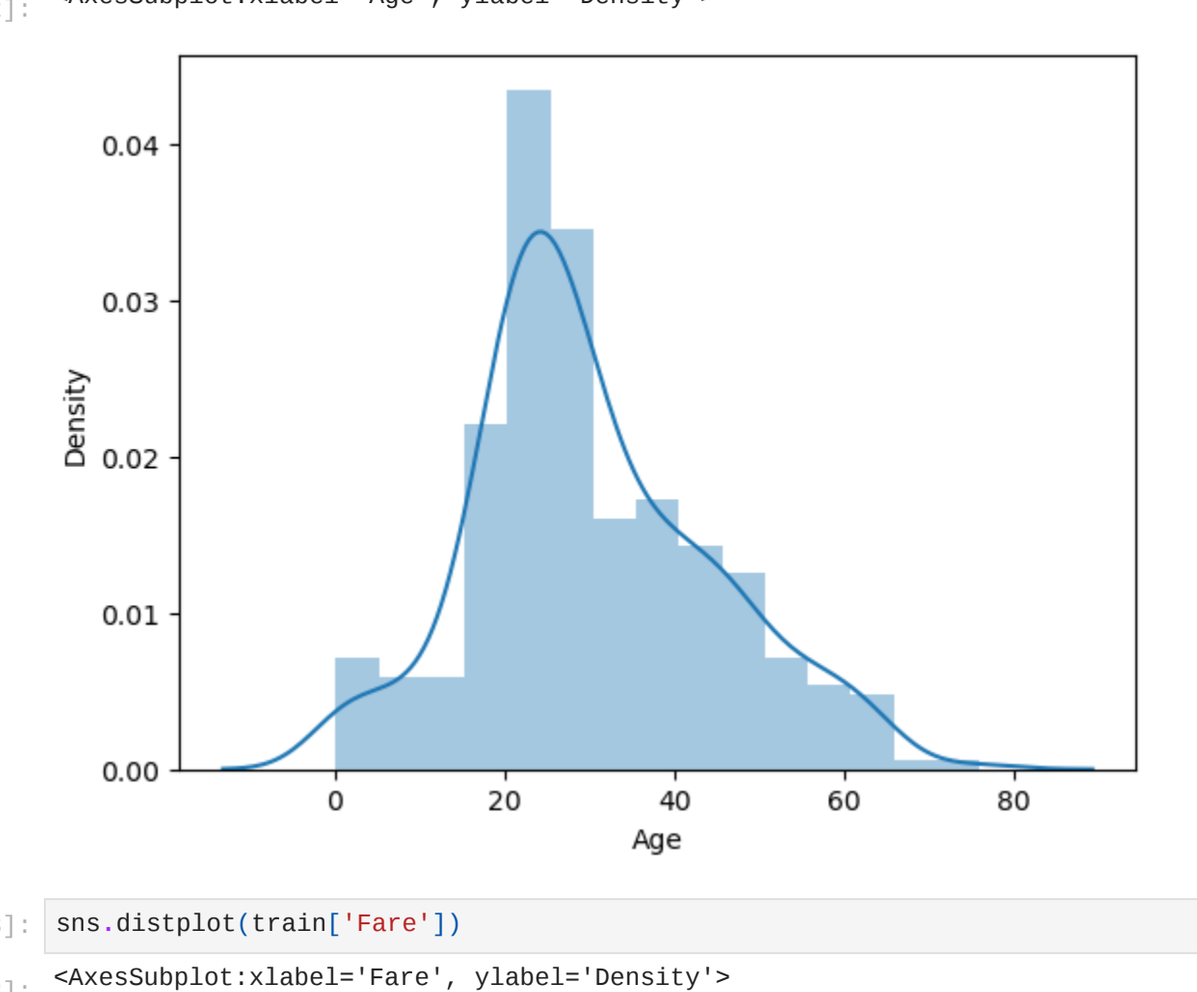
In [11]:
```
sns.countplot(train['Embarked'])
```

Out[11]: <AxesSubplot:xlabel='Embarked', ylabel='count'>



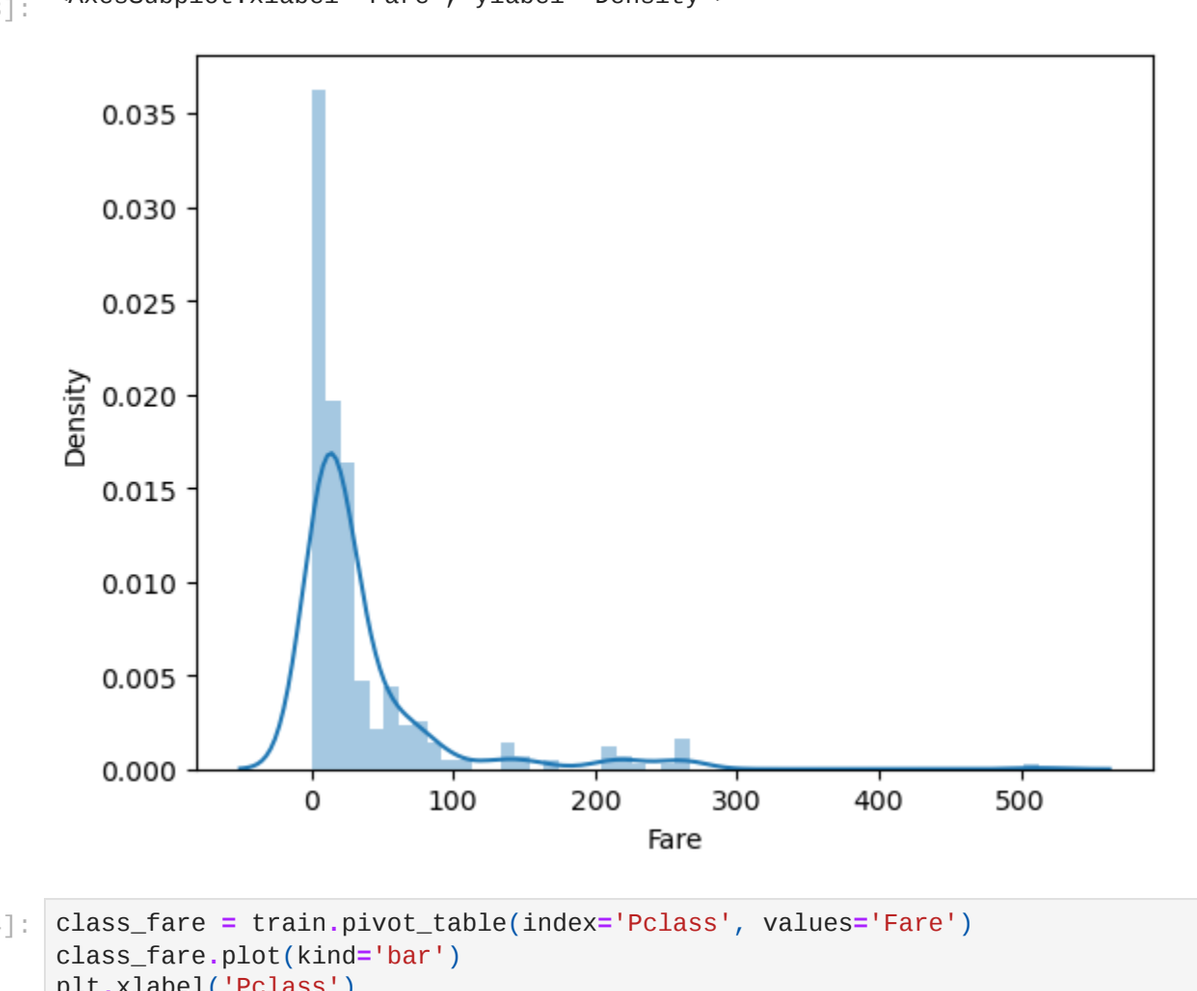
In [12]:
```
## numerical attributes
sns.distplot(train['Age'])
```

Out[12]: <AxesSubplot:xlabel='Age', ylabel='Density'>

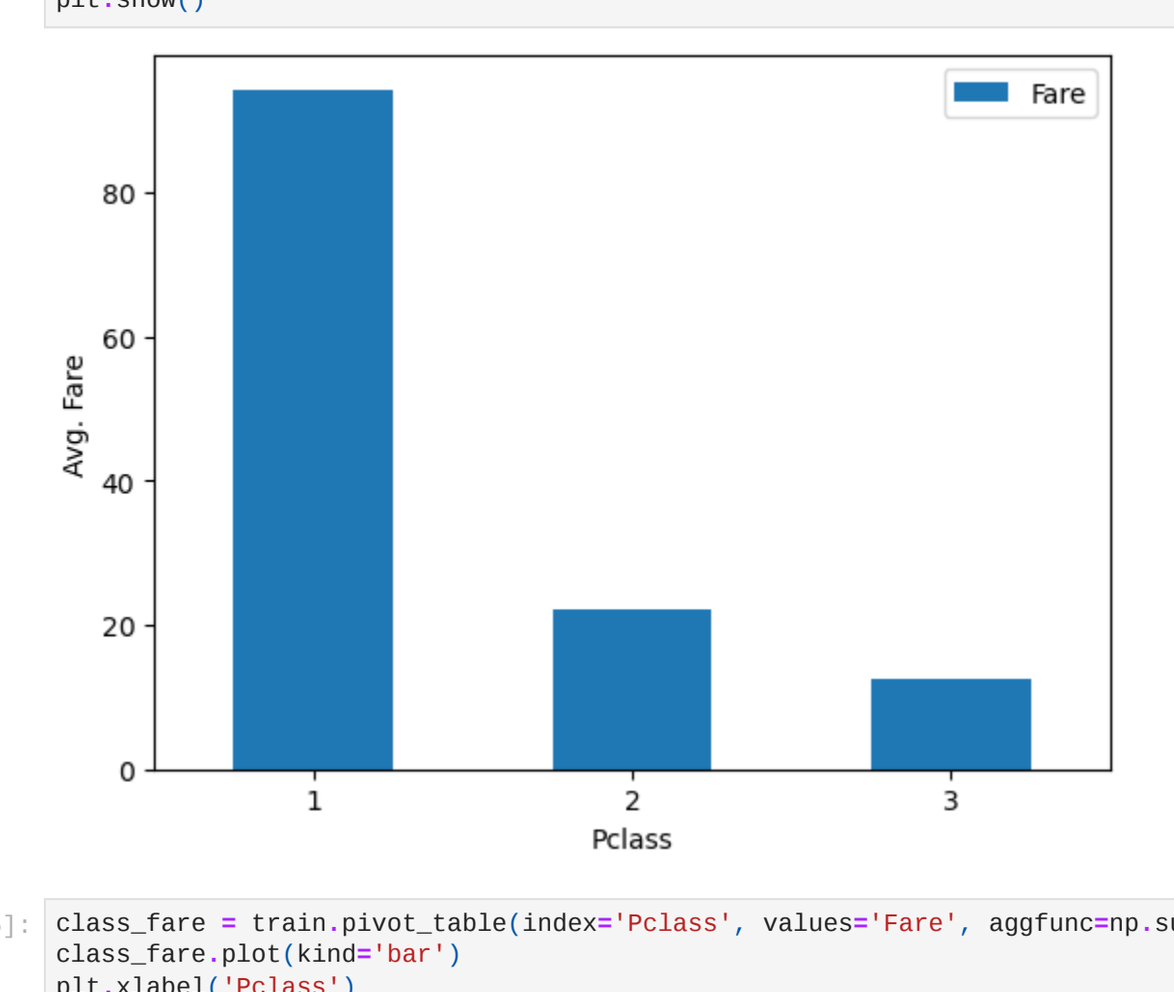


In [13]:
```
sns.distplot(train['Fare'])
```

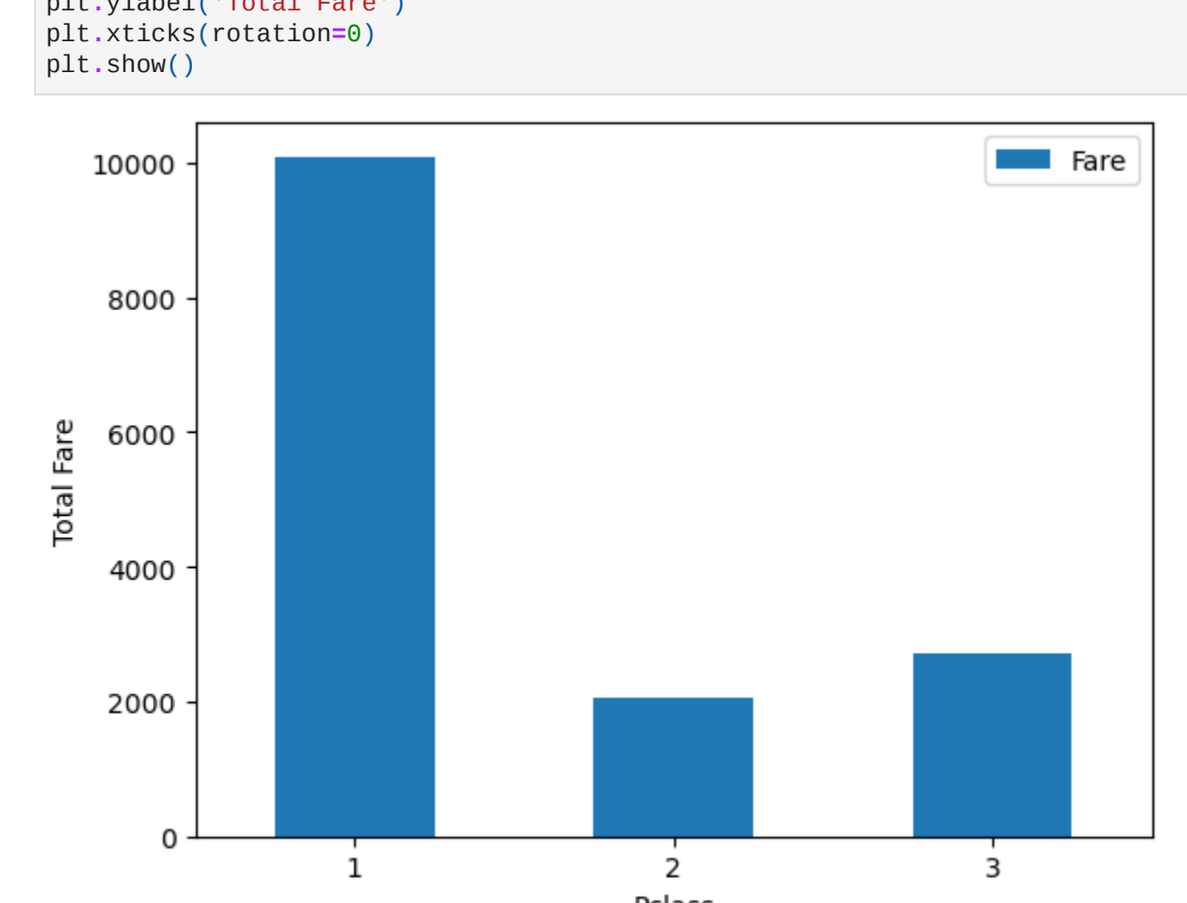
Out[13]: <AxesSubplot:xlabel='Fare', ylabel='Density'>



In [14]:
```
class_fare = train.pivot_table(index='Pclass', values='Fare')
class_fare.plot(kind='bar')
plt.xlabel('Pclass')
plt.ylabel('Avg. Fare')
plt.xticks(rotation=0)
plt.show()
```

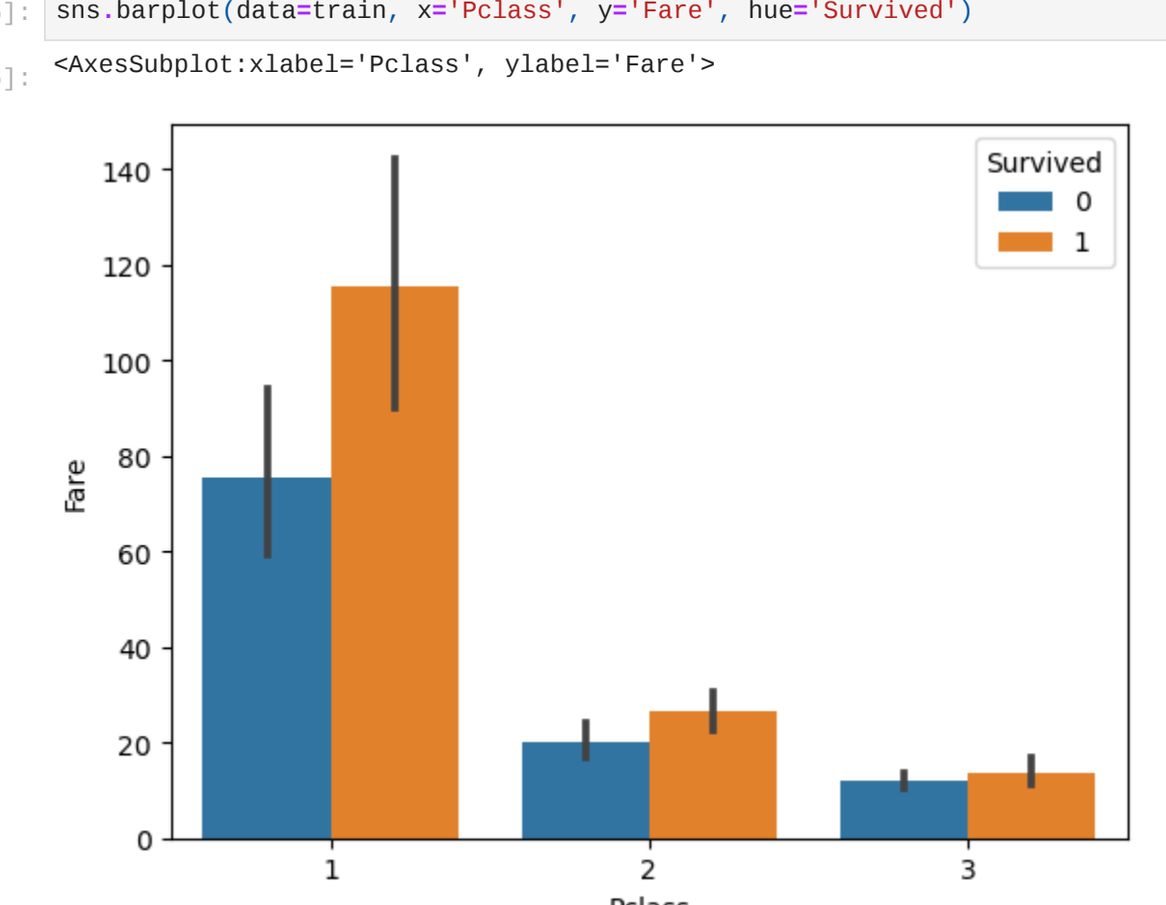


In [15]:
```
class_fare = train.pivot_table(index='Pclass', values='Fare', aggfunc=np.sum)
class_fare.plot(kind='bar')
plt.xlabel('Pclass')
plt.ylabel('Total Fare')
plt.xticks(rotation=0)
plt.show()
```



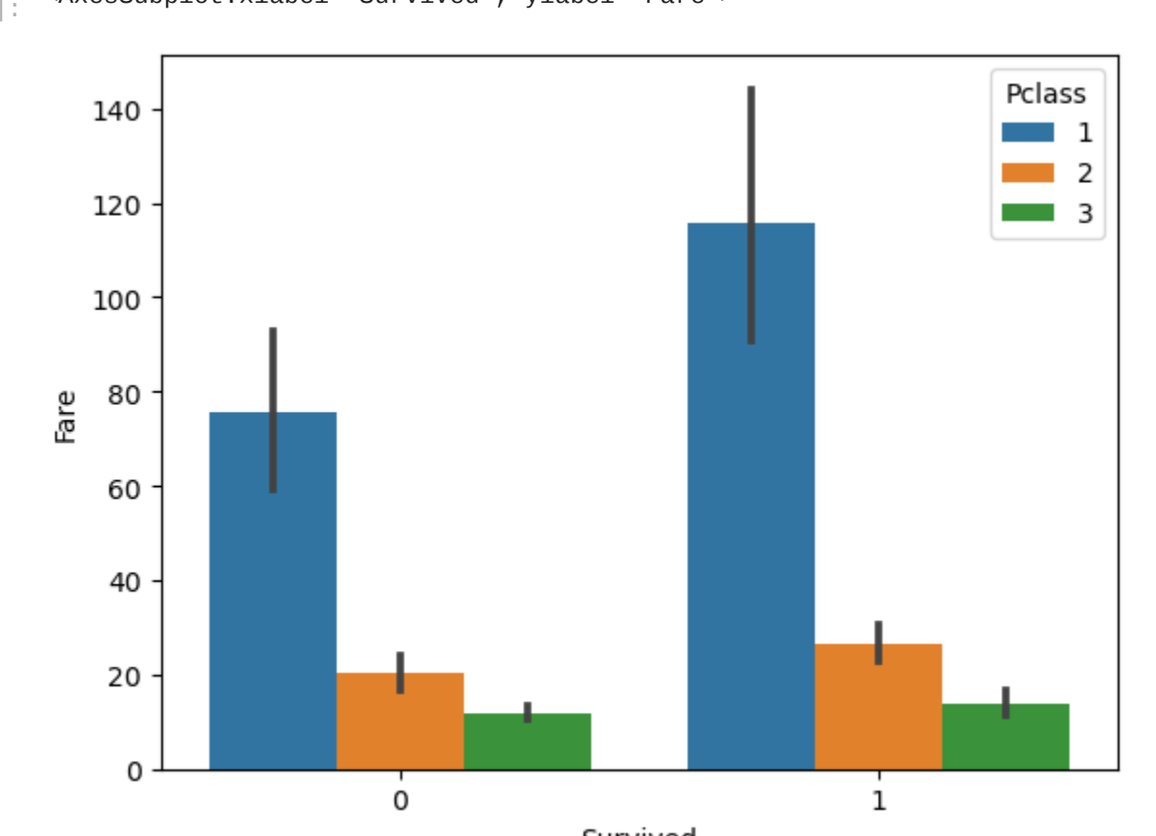
In [16]:
```
sns.barplot(data=train, x='Pclass', y='Fare', hue='Survived')
```

Out[16]: <AxesSubplot:xlabel='Pclass', ylabel='Fare'>



In [17]:
```
sns.barplot(data=train, x='Survived', y='Fare', hue='Pclass')
```

Out[17]: <AxesSubplot:xlabel='Survived', ylabel='Fare'>



Preprocessing of Data

In [18]:
```
train_len = len(train)
# combine two dataframes
df = pd.concat([train, test], axis=0)
df = df.reset_index(drop=True)
df.head()
```

Out[18]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [19]:
```
## find the null values
df.isnull().sum()
```

Out[19]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            172
SibSp            0
Parch            0
Ticket           0
Fare             2
Cabin          654
Embarked         0
dtype: int64
```

In [20]:
```
# drop or delete the column
df = df.drop(columns=['Cabin'], axis=1)
```

In [21]:
```
df['Age'].mean()
```

Out[21]: 30.272590361445783

In [22]:
```
# fill missing values using mean of the numerical column
df['Age'] = df['Age'].fillna(df['Age'].mean())
df['Fare'] = df['Fare'].fillna(df['Fare'].mean())
```

In [23]:
```
df['Embarked'].mode()[0]
```

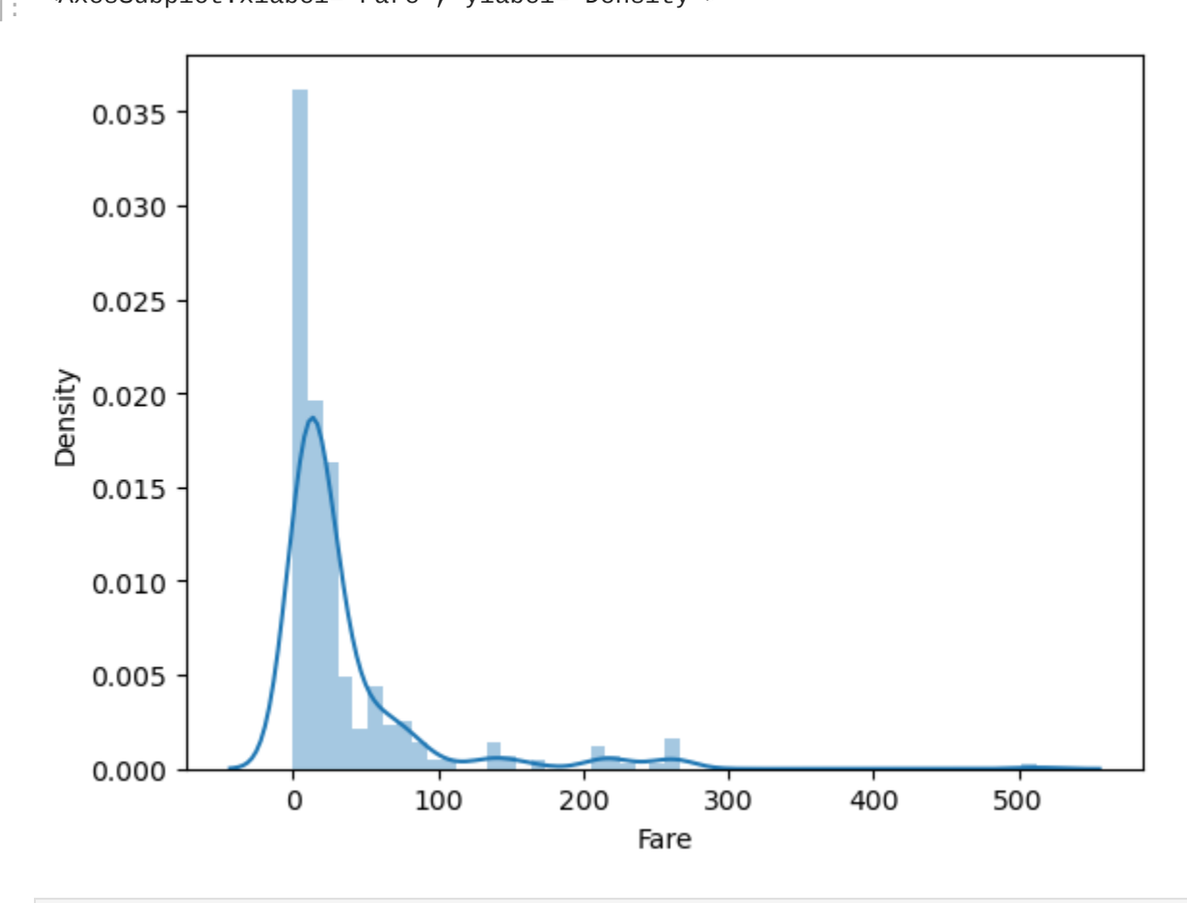
Out[23]: 'S'

In [24]:
```
# fill missing values using mode of the categorical column
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
```

Log transformation for uniform data distribution

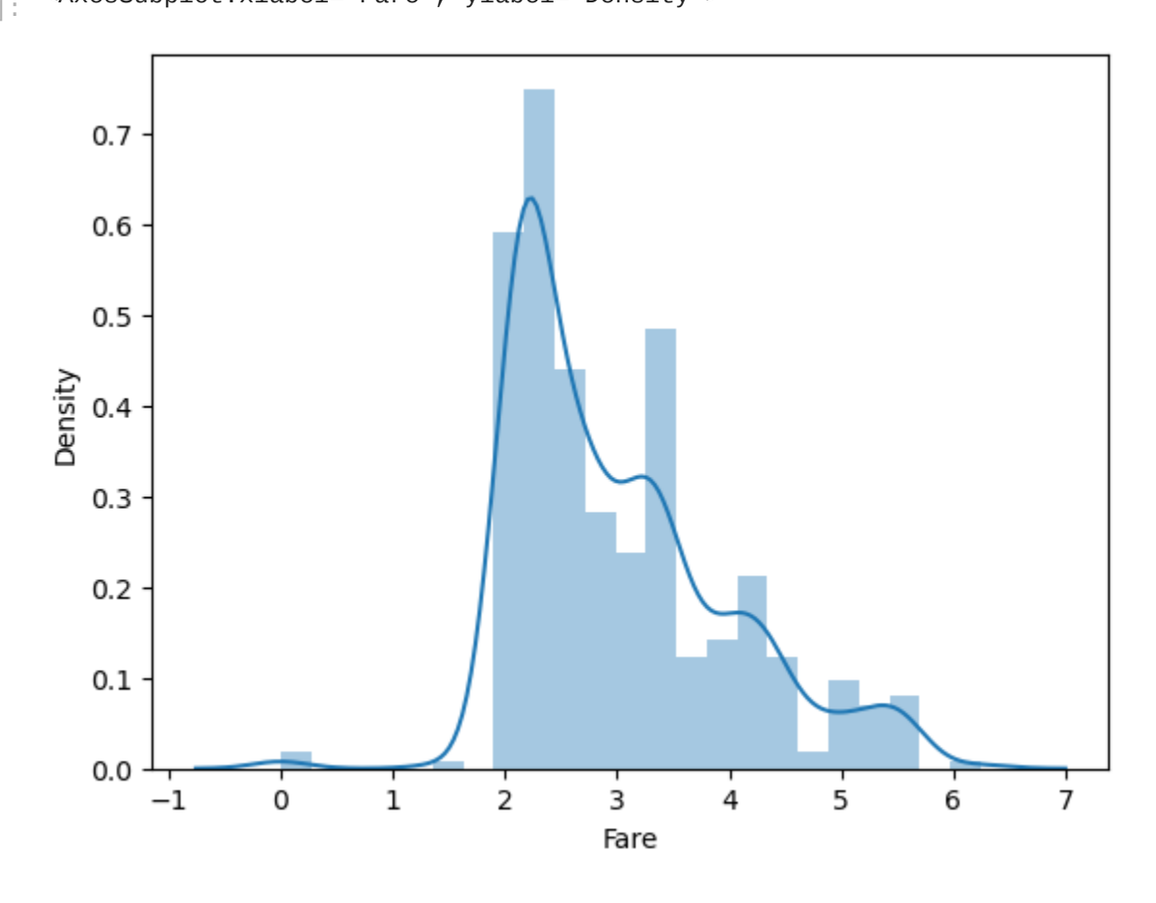
In [25]:
```
sns.distplot(df['Fare'])
```

Out[25]: <AxesSubplot:xlabel='Fare', ylabel='Density'>



In [26]:
```
df['Fare'] = np.log(df['Fare']+1)
sns.distplot(df['Fare'])
```

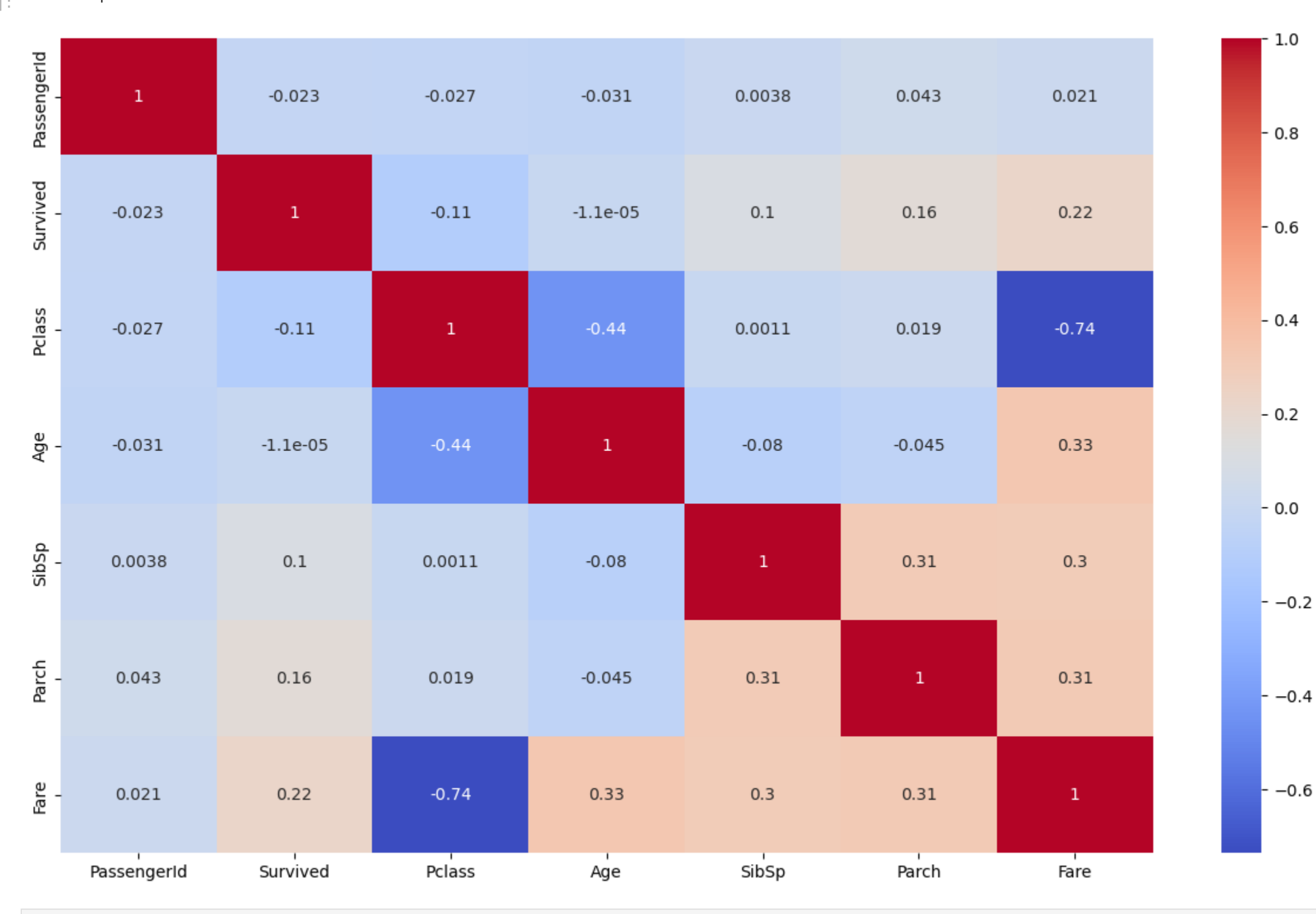
Out[26]: <AxesSubplot:xlabel='Fare', ylabel='Density'>



Correlation Matrix

In [27]:
```
corr = df.corr()
plt.figure(figsize=(15, 9))
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

Out[27]: <AxesSubplot:>



In [28]:
```
df.head()
```

Out[28]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	2.178064	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	2.079442	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	2.369075	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	2.268252	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	2.586824	S

In [29]:
```
## drop unnecessary columns
df = df.drop(columns=['Name', 'Ticket'], axis=1)
df.head()
```

Out[29]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	892	0	3	male	34.5	0	0	2.178064	Q
1	893	1	3	female	47.0	1	0	2.079442	S
2	894	0	2	male	62.0	0	0	2.369075	Q
3	895	0	3	male	27.0	0	0	2.268252	S
4	896	1	3	female	22.0	1	1	2.586824	S

Label Encoding

In [30]:
```
from sklearn.preprocessing import LabelEncoder
cols = ['Sex', 'Embarked']
le = LabelEncoder()

for col in cols:
    df[col] = le.fit_transform(df[col])
df.head()
```

Out[30]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	892	0	3	1	34.5	0	0	2.178064	1
1	893	1	3	0	47.0	1	0	2.079442	2
2	894	0	2	1	62.0	0	0	2.369075	1
3	895	0	3	1	27.0	0	0	2.268252	2
4	896	1	3	0	22.0	1	1	2.586824	2

Train-Test Split

In [31]:
```
train = df.iloc[:train_len, :]
test = df.iloc[train_len:, :]
train.head()
```

Out[31]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	892	0	3	1	34.5	0	0	2.178064	1
1	893	1	3	0	47.0	1	0	2.079442	2
2	894	0	2	1	62.0	0	0	2.369075	1
3	895	0	3	1	27.0	0	0	2.268252	2
4	896	1	3	0	22.0	1	1	2.586824	2

In [32]:
```
test.head()
```

Out[32]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
418	892	0	3	1	34.5	0	0	2.178064	1
419	893	1	3	0	47.0	1	0	2.079442	2
420	894	0	2	1	62.0	0	0	2.369075	1
421	895	0	3	1	27.0	0	0	2.268252	2
422	896	1	3	0	22.0	1	1	2.586824	2

In [33]:
```
# input split
X = train.drop(columns=['PassengerId', 'Survived'], axis=1)
y = train['Survived']
X.head()
```

Out[33]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	34.5	0	0	2.178064	1
1	3	0	47.0	1	0	2.079442	2
2	2	1	62.0	0	0	2.369075	1
3	3	1	27.0	0	0	2.268252	2
4	3	0	22.0	1	1	2.586824	2

Model Training

In [34]:
```
from sklearn.model_selection import train_test_split, cross_val_score
# classify column
def classify(model):
    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
    model.fit(x_train, y_train)
    print('Accuracy:', model.score(x_test, y_test))

    score = cross_val_score(model, X, y, cv=5)
    print('CV Score:', np.mean(score))
```

In [35]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
classify(model)
```

```
Accuracy: 1.0
CV Score: 1.0
```

In [36]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
classify(model)
```

```
Accuracy: 1.0
CV Score: 1.0
```

In [37]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
classify(model)
```

```
Accuracy: 1.0
CV Score: 1.0
```

In [38]:
```
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier()
classify(model)
```

```
Accuracy: 1.0
CV Score: 1.0
```

Complete Model Training with Full Data

In [49]:
```
model = RandomForestClassifier()
model.fit(X, y)
```

Out[49]: RandomForestClassifier()

In [50]:
```
test.head()
```

Out[50]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
418	892	0	3	1	34.5	0	0	2.178064	1
419	893	1	3	0	47.0	1	0	2.079442	2
420	894	0	2	1	62.0	0	0	2.369075	1
421	895	0	3	1	27.0	0	0	2.268252	2
422	896	1	3	0	22.0	1	1	2.586824	2

In [51]:
```
# input split for test data
X_test = test.drop(columns=['PassengerId', 'Survived'], axis=1)
```

In [52]:
```
X_test.head()
```

Out[52]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
418	3	1	34.5	0	0	2.178064	1
419	3	0	47.0	1	0	2.079442	2
420	2	1	62.0	0	0	2.369075	1
421	3	1	27.0	0	0	2.268252	2
422	3	0	22.0	1	1	2.586824	2

In [53]:
```
pred = model.predict(X_test)
pred
```

Out[53]:
```
array([0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0],
      dtype=int64)
```

In [ ]:

In [ ]: