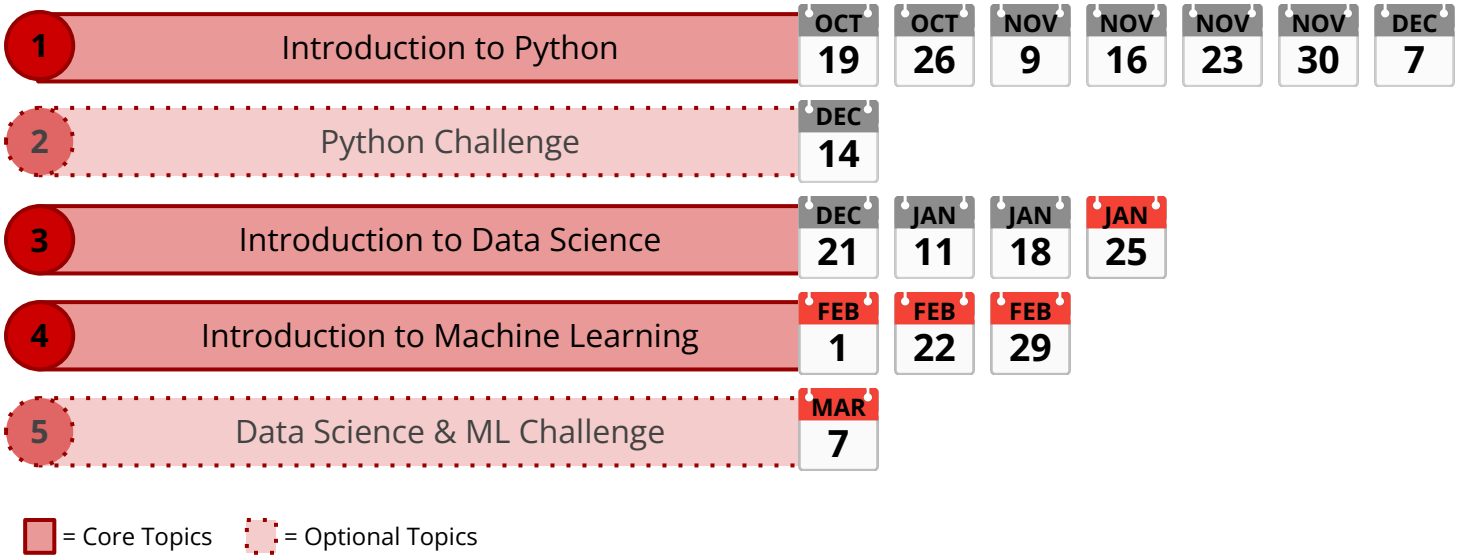
# Python for Data Science and Machine Learning

School Year 2023-2024

IST

# Course Structure



# Jupyter Notebook Setup


In a browser:

```
192.168.10.4:8888
```

Password: **ist**

# Recap: Pandas

Pandas is a powerful Python data analysis toolkit.

It provides flexible data structures like **Series** and **DataFrame**.

Widely used in data science, finance, and many other fields.

**12.0**

```
import pandas as pd
import numpy as np
```

# Recap: DataFrame

A **DataFrame** is a two-dimensional data structure with labeled axes (rows and columns).

**12.1**

```
df = pd.read_csv("titanic_dataset.csv")
df
```

# Recap: DataFrame

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **886** | 887 | 0 | 2 | Montvila, Rev. Juozas | male | 27.0 | 0 | 0 | 211536 | 13.0000 | NaN | S |
| **887** | 888 | 1 | 1 | Graham, Miss. Margaret Edith | female | 19.0 | 0 | 0 | 112053 | 30.0000 | B42 | S |
| **888** | 889 | 0 | 3 | Johnston, Miss. Catherine Helen "Carrie" | female | NaN | 1 | 2 | W./C. 6607 | 23.4500 | NaN | S |
| **889** | 890 | 1 | 1 | Behr, Mr. Karl Howell | male | 26.0 | 0 | 0 | 111369 | 30.0000 | C148 | C |
| **890** | 891 | 0 | 3 | Dooley, Mr. Patrick | male | 32.0 | 0 | 0 | 370376 | 7.7500 | NaN | Q |

891 rows × 12 columns

# Recap: Selecting DataFrame Data

- The **loc** method in Pandas can be used for selecting rows but also for columns.
- By specifying the row and column labels, you can access specific portions of the dataset.

```
df.loc[0, "Name"]
```

```
df.loc[0:4, "Name"]
```

```
df.loc[:4, "Name"]
```

```
df.loc[4, ["Name", "Age"]]
```

```
df.loc[0:4, ["Name", "Age"]]
```

```
df.loc[:, ["Name", "Age"]]
```

# Recap: Boolean Indexing

- **Boolean indexing** in Pandas allows you to select data subsets based on the actual values in the data.

```
df[df.loc[0:9, "Age"] > 30]
```

- **SHORTHAND:** If you wish to **select specific columns** across **all rows** you can use the following:

```
df.loc[:, 'Age']
```

```
df['Age']
```

```
df[df.loc[:, "Age"] > 30]
```

```
df[df["Age"] > 30]
```

# Recap: Chaining Indexing

You can **chain** multiple boolean indexing operations by using:

- | for “or”
- **&** for “and”

**IMPORTANT!** You must use **brackets**!

```
df[(df["Pclass"] == 1) | (df["Pclass"] == 2)]
```

```
df[(df["Pclass"] == 1) & (df["Age"] < 18)]
```

# Recap: Data Analysis

We can use the **.mean()**, **.count()**, **.max()** and **.min()** functions to analyse our data.

```
df["Age"].mean()
```

```
df["Fare"].max()
```

```
df[df["Survived"] == 1]["Age"].min()
```

# Recap: Grouping

Before we analyse our data we can group pieces of information together. We use the **.groupby()** function.

We pass in the **column** to group the data with.

```
df.groupby("Embarked")["Name"].count()
```

```
df.groupby("Pclass")["Survived"].mean()
```

# Recap: Indexing, Grouping & Analysis

When using them all together, in order we:

1. First use boolean indexing
2. Secondly use grouping
3. Finally we select the analysis function we'd like

```
df[df["Age"] < 18].groupby("Pclass")["Survived"].count()
```

**Indexing** **Grouping** **Data Analysis**

# Recap Exercise

Complete the **12.2** , **12.3**, **12.4**, **12.5** & extension programs.

- **12.2**: Select passengers who are females *and* traveled in first class.
- **12.3**: Find the average fare paid by passengers in each class.
- **12.4**: Find the maximum age of male passengers who perished in the Titanic disaster.
- **12.5**: Calculate the count of passengers in each class who did not survive.

# Solution 12.2

```
df[(df['Sex'] == 'female') & (df['Pclass'] == 1)]
```

# Solution 12.3

```
df.groupby('Pclass')['Fare'].mean()
```

# Solution 12.4

```
df[(df['Sex'] == 'male') & (df['Survived'] == 0)]['Age'].max()
```

# Solution 12.5

```
df[df['Survived'] == 0].groupby('Pclass')['Name'].count()
```

# Solution 12.5.1 & 12.5.2

```
df[(df['Sex'] == 'female') & (df['Survived'] == 0)].groupby('Pclass')['Fare'].mean()
```

```
df[df['Age'] < 30].groupby('Survived')['Fare'].sum()
```

# Plotting: Bar Chart

Pandas can use a library called **matplotlib** to plot the data we have extracted into charts.

**12.06**

```
df.groupby('Pclass')['Fare'].mean()
```

Let's turn it into a chart:

**12.07**

```
df.groupby('Pclass')['Fare'].mean().plot(kind='bar')
```

# Plotting: Bar Chart

**12.06**

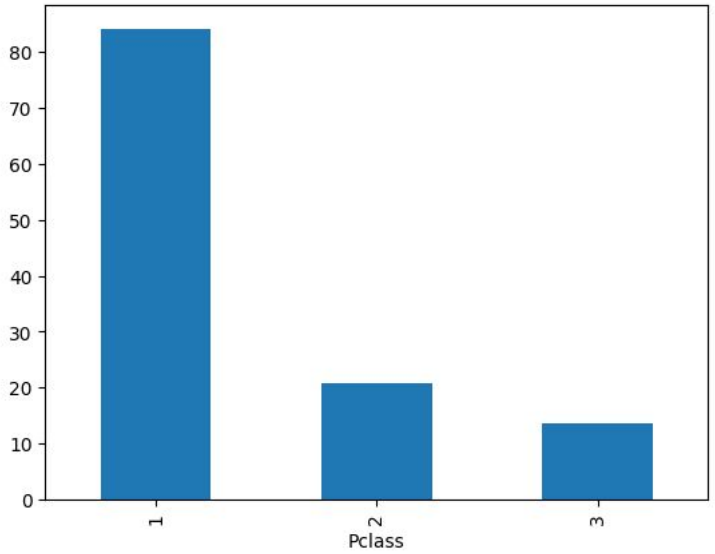
```
df.groupby('Pclass')['Fare'].mean()
```

```
Pclass
1    84.154687
2    20.662183
3    13.675550
Name: Fare, dtype: float64
```

**12.07**

```
df.groupby('Pclass')['Fare'].mean().plot(kind='bar')
```

```
<Axes: xlabel='Pclass'>
```



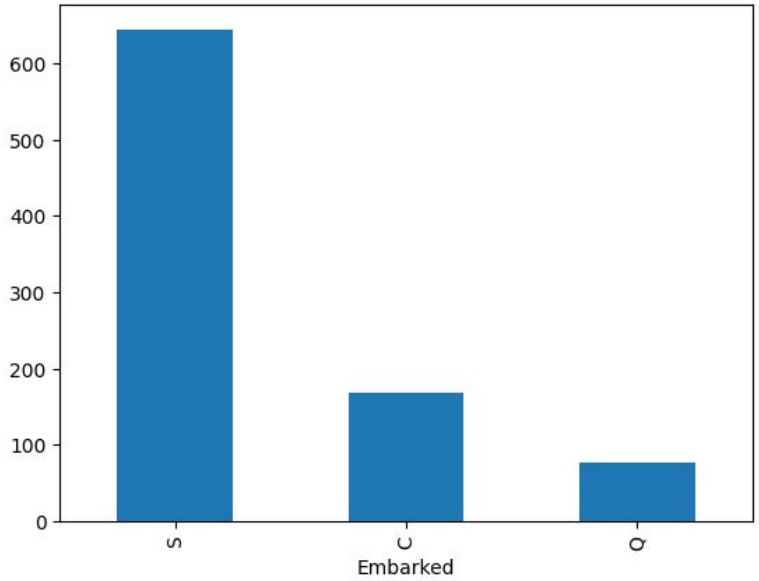
# Exercise

Complete the **12.8**, **12.9** & extension programs.

- **12.8**: Plot the total number of Passengers by Embarkation Point
- **12.9**: Plot the total Fare collected from each Passenger Class

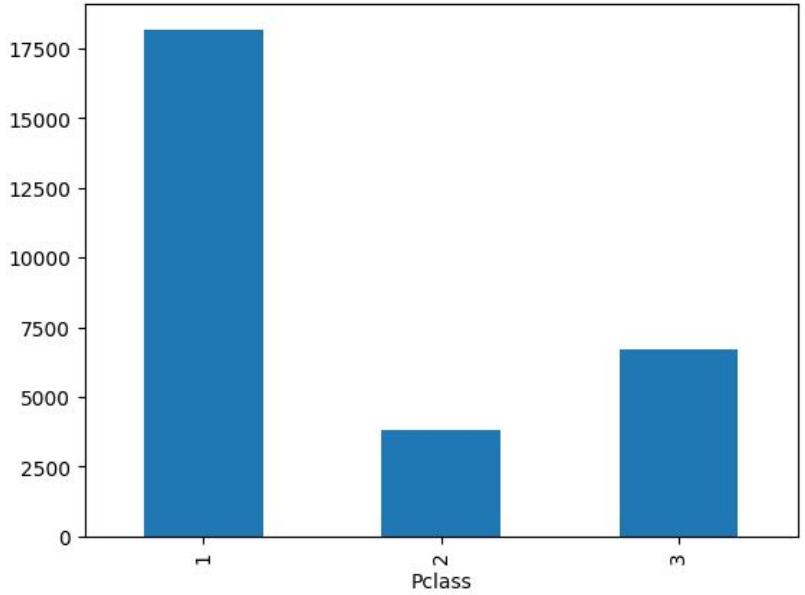
# Solution 12.8

```
df['Embarked'].value_counts().plot(kind='bar')
```



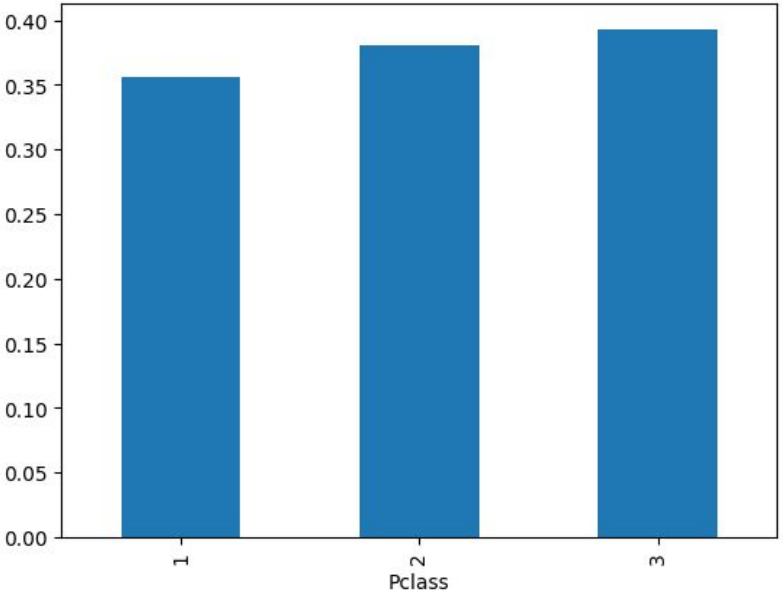
# Solution 12.9

```
df.groupby('Pclass')['Fare'].sum().plot(kind='bar')
```

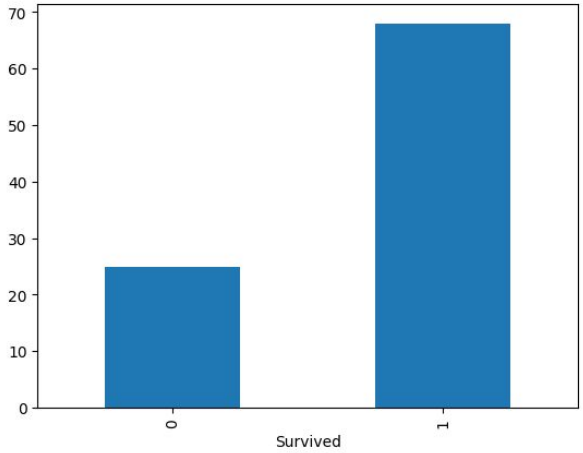


# Solution 12.9.1 & 12.9.2

```
df.groupby('Pclass')['Parch'].mean().plot(kind='bar')
```




```
df[df["Age"] > 30]\
    .groupby('Survived')['Fare']\
    .mean()\
    .plot(kind='bar')
```



# Plotting: Line Chart

We can also generate other types of chart:

**12.10**

```
df.groupby("Age")[["SibSp", "Parch"]].mean()
```

**12.11**

```
df.groupby('Age')[['SibSp', 'Parch']].mean().plot(kind='line')
```

Experiment changing the group column and the selection

# Plotting: Bar Chart

**12.10**

```
df.groupby('Age')[['SibSp', 'Parch']].mean()
```

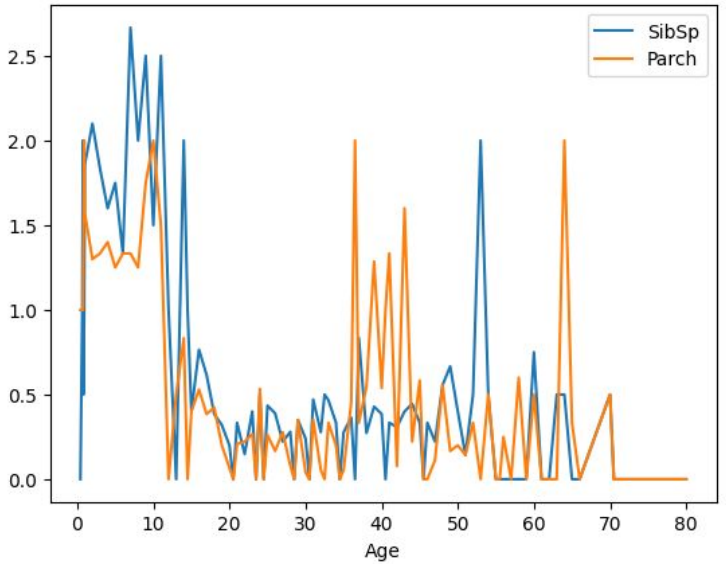
| Age | SibSp | Parch |
|---|---|---|
| **0.42** | 0.0 | 1.0 |
| **0.67** | 1.0 | 1.0 |
| **0.75** | 2.0 | 1.0 |
| **0.83** | 0.5 | 1.5 |
| **0.92** | 1.0 | 2.0 |
| **...** | ... | ... |
| **70.00** | 0.5 | 0.5 |
| **70.50** | 0.0 | 0.0 |
| **71.00** | 0.0 | 0.0 |
| **74.00** | 0.0 | 0.0 |
| **80.00** | 0.0 | 0.0 |

88 rows × 2 columns

**12.11**

```
df.groupby('Age')[['SibSp', 'Parch']].mean().plot(kind='line')
```

```
<Axes: xlabel='Age'>
```



# Plotting: Scatter Chart

Let's generate Scatter charts:

**12.12**

```
df[df["Survived"] == 0].plot(kind='scatter', x='Age', y='Fare')
```

**12.13**

```
df[df["Survived"] == 2].plot(kind='scatter', x='Age', y='Fare')
```

We can merge the two together!

**12.14**

```
df.plot(kind='scatter', x='Age', y='Fare', c='Survived', colormap='viridis')
```

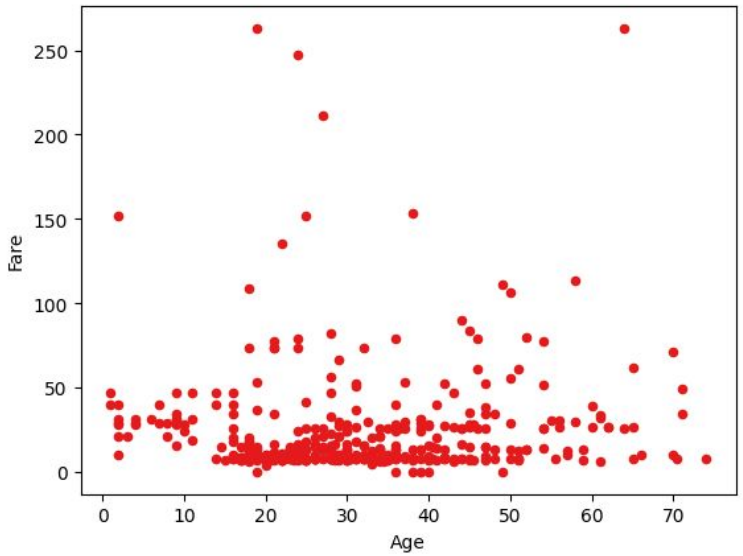
Experiment changing the **x**, **y** and **c** parameters

# Plotting: Bar Chart

**12.12**

```
df[df["Survived"] == 0].plot(kind='scatter', x='Age', y='Fare')
```

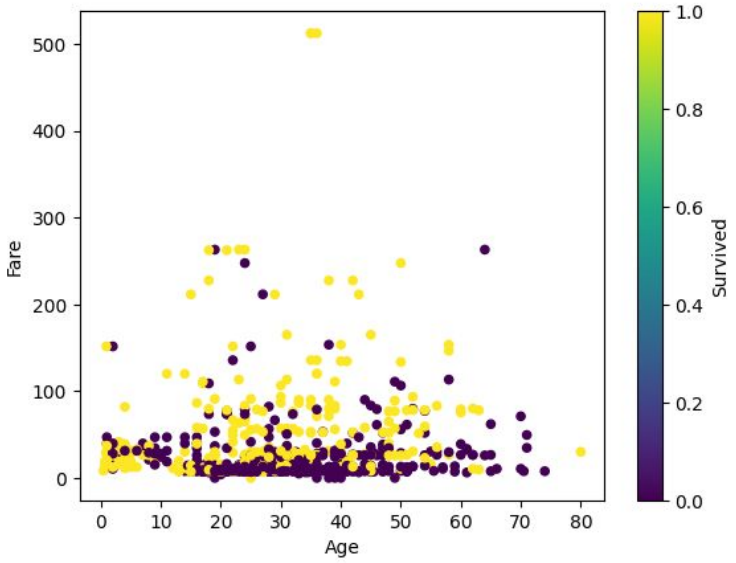
```
<Axes: xlabel='Age', ylabel='Fare'>
```




**12.14**

```
df.plot(kind='scatter', x='Age', y='Fare', c='Survived', colormap='viridis')
```

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



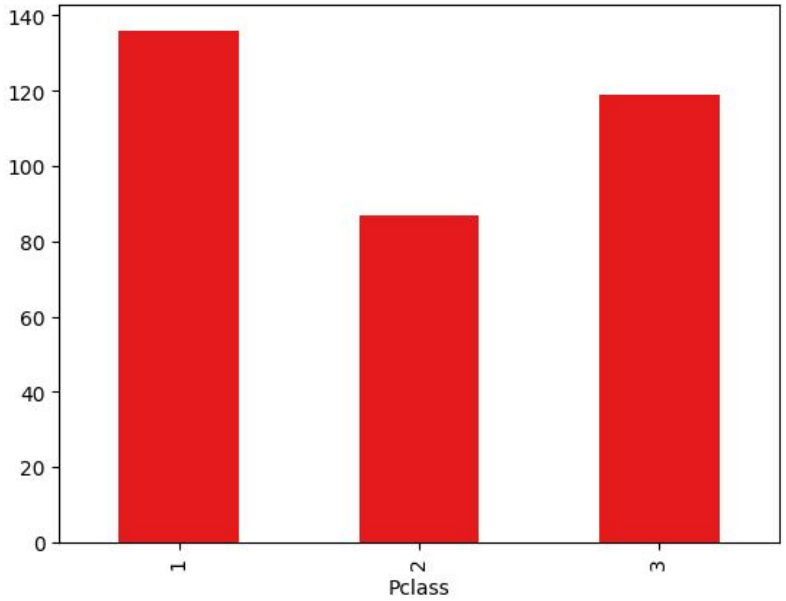
# Recap Exercise

Complete the **12.15** , **12.16** & extension programs.

- **12.15**: Display the total number of survivors for each passenger class
- **12.16**: Find the average fare paid by passengers in each class.

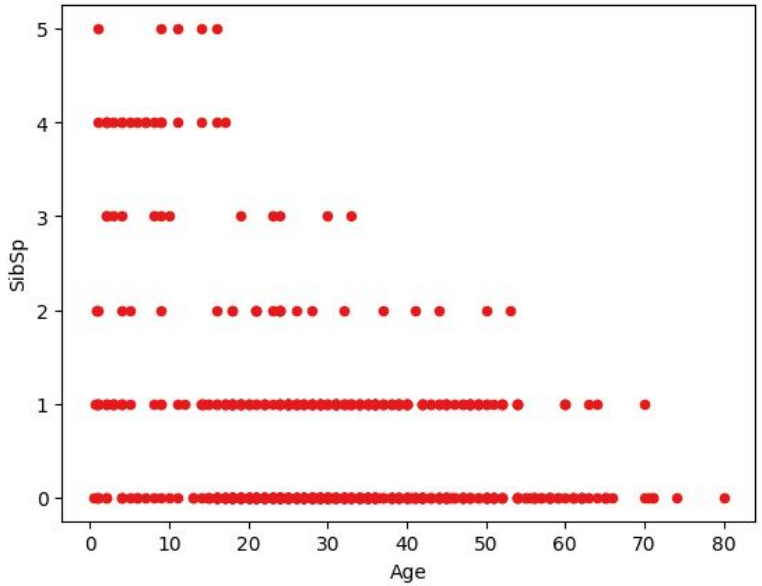
# Solution 12.15

```
df.groupby('Pclass')['Survived'].sum().plot(kind='bar')
```



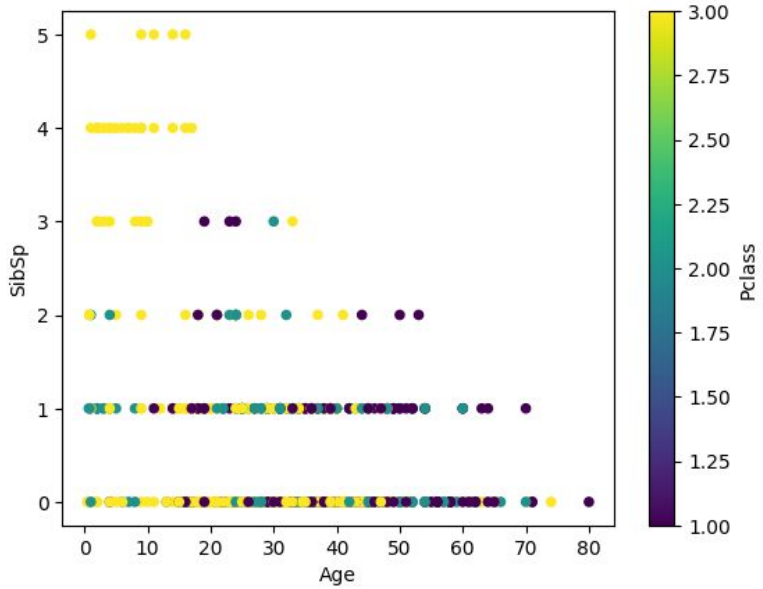
# Solution 12.16

```
df.plot(kind='scatter', x='Age', y='SibSp')
```



# Solution 12.16.1

```
df.plot(kind='scatter', x='Age', y='SibSp', c='Pclass', colormap='viridis')
```



# Quiz Time!

https://ahaslides.com/HFHY2

# End of Class

See you all next week!