Basics of Data Visualization

Honours* in Data Science Third Year of Engineering (Semester V)

Data Science and Visualization

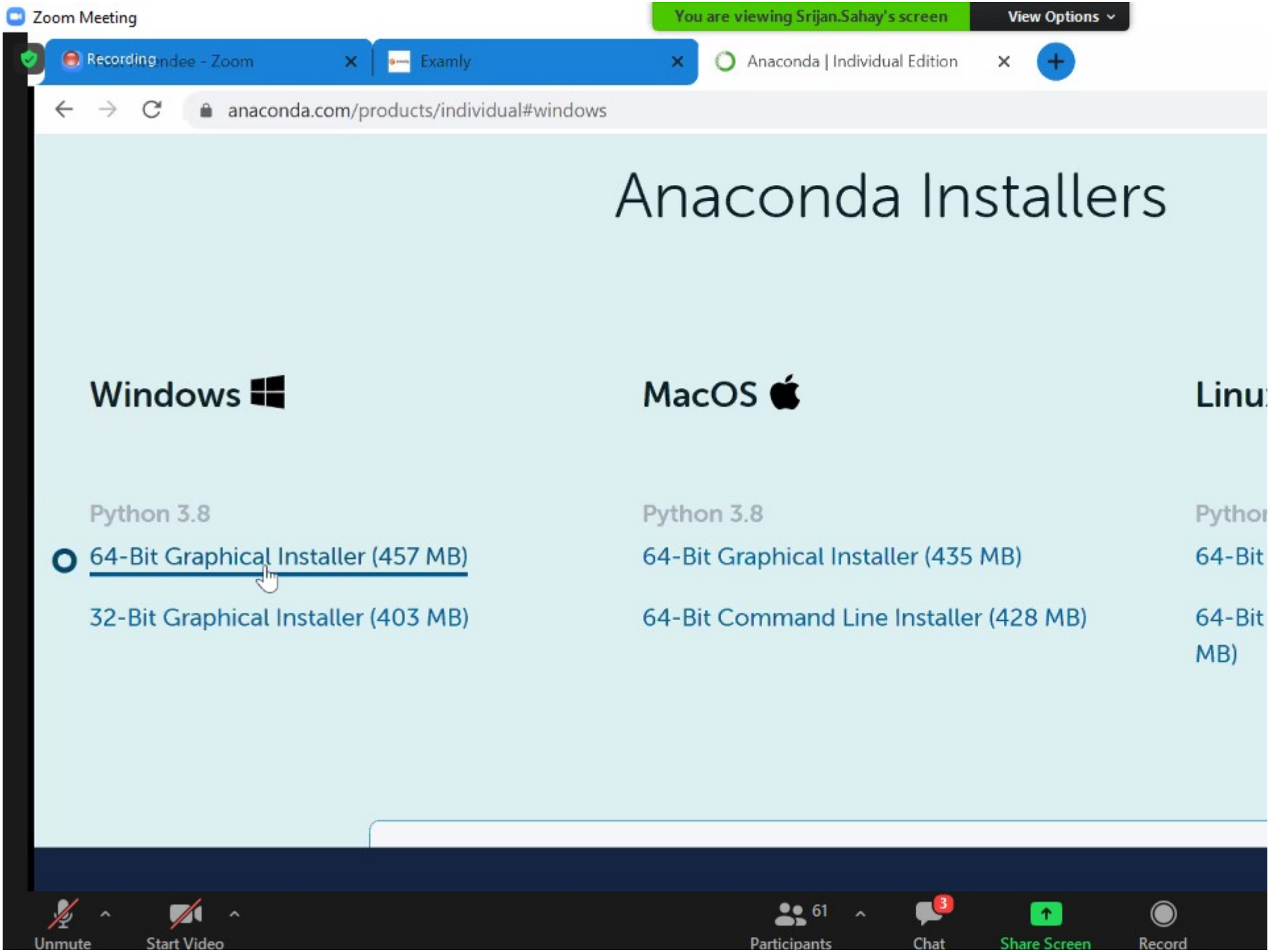
What is Python

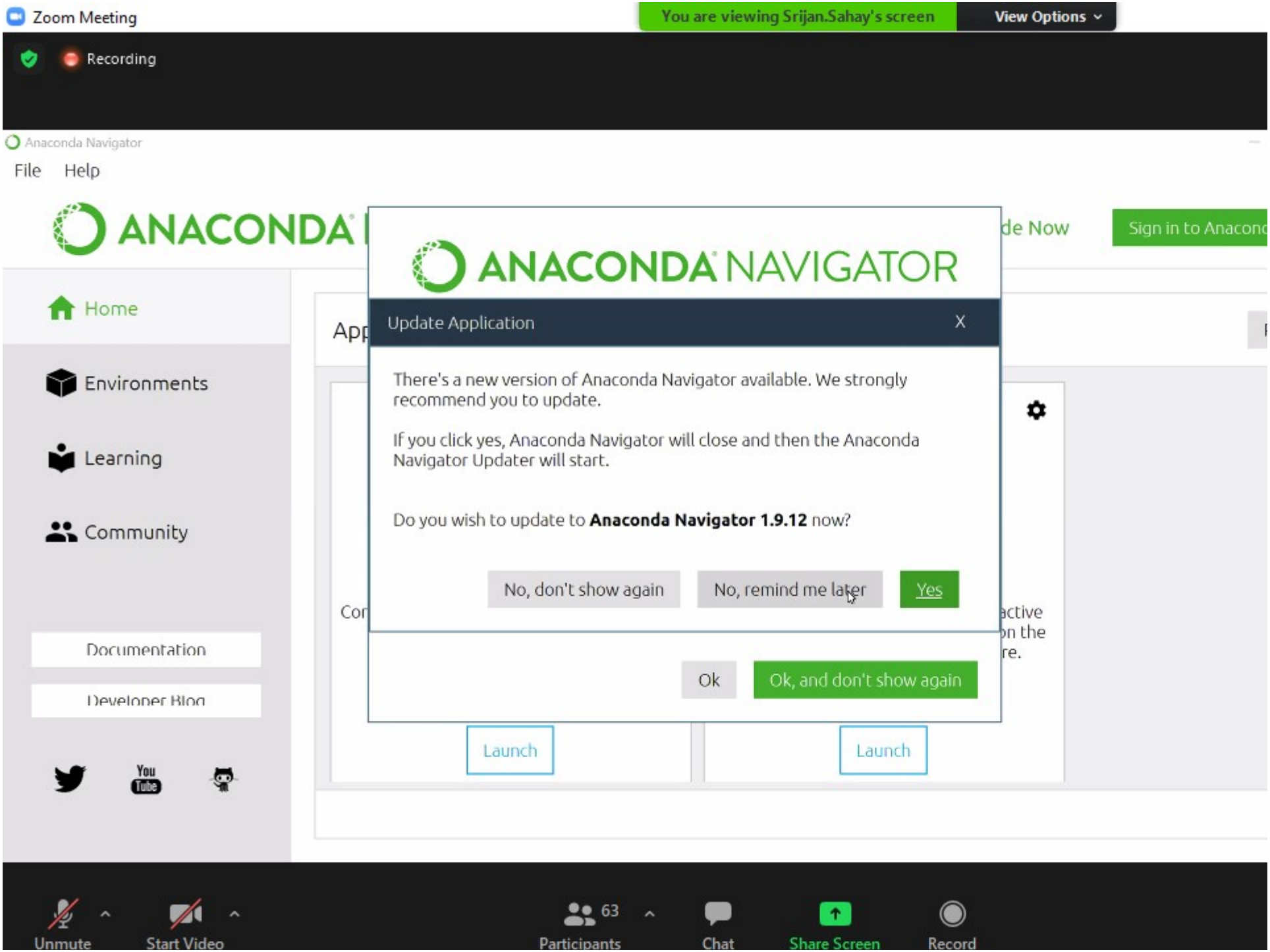
Introduction

What is Python?
Python is an interpreted, high-level and general-purpose programming language.
Python is programming language as well as scripting language.

History
Invented in the Netherlands, early 90s by Guido van Rossum.
Python was conceived in the late 1980s and its implementation was started in December 1989.
Guido Van Rossum is fan of 'Monty Python's Flying Circus', this is a famous TV show in Netherlands.
Named after Monty Python.
Open sourced from the beginning.
Python 2.0 was released on 16 October 2000.
Python 3.0 was released on 3 December 2008.
Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020.

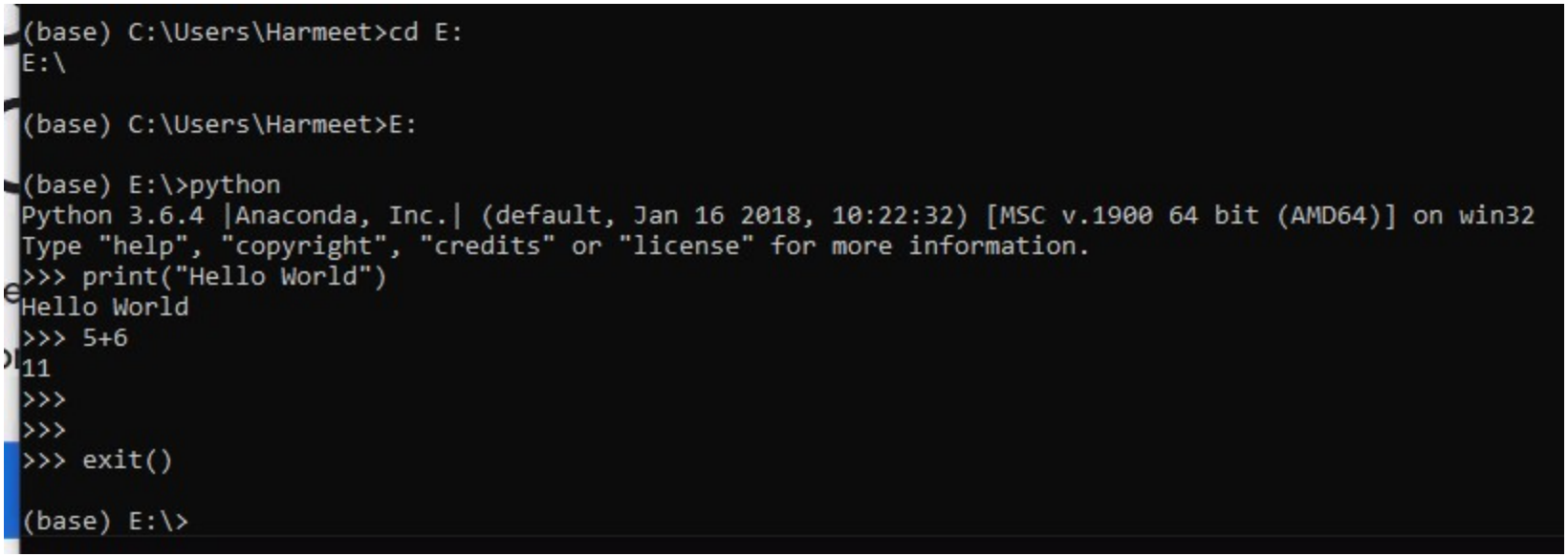
IDE: To write codes: Anaconda, Spyder, Jupyter id
Latest vesrion of Anaconada


Zoom Meeting
You are viewing Srijan.Sahay's screen
View Options
Recording
Examly
Anaconda | Individual Edition
anaconda.com/products/individual#windows
Anaconda Installers
Windows
MacOS
Python 3.8
64-Bit Graphical Installer (457 MB)
32-Bit Graphical Installer (403 MB)
Python 3.8
64-Bit Graphical Installer (435 MB)
64-Bit Command Line Installer (428 MB)
Unmute
Start Video
61
Participants
3
Chat
Share Screen
Record

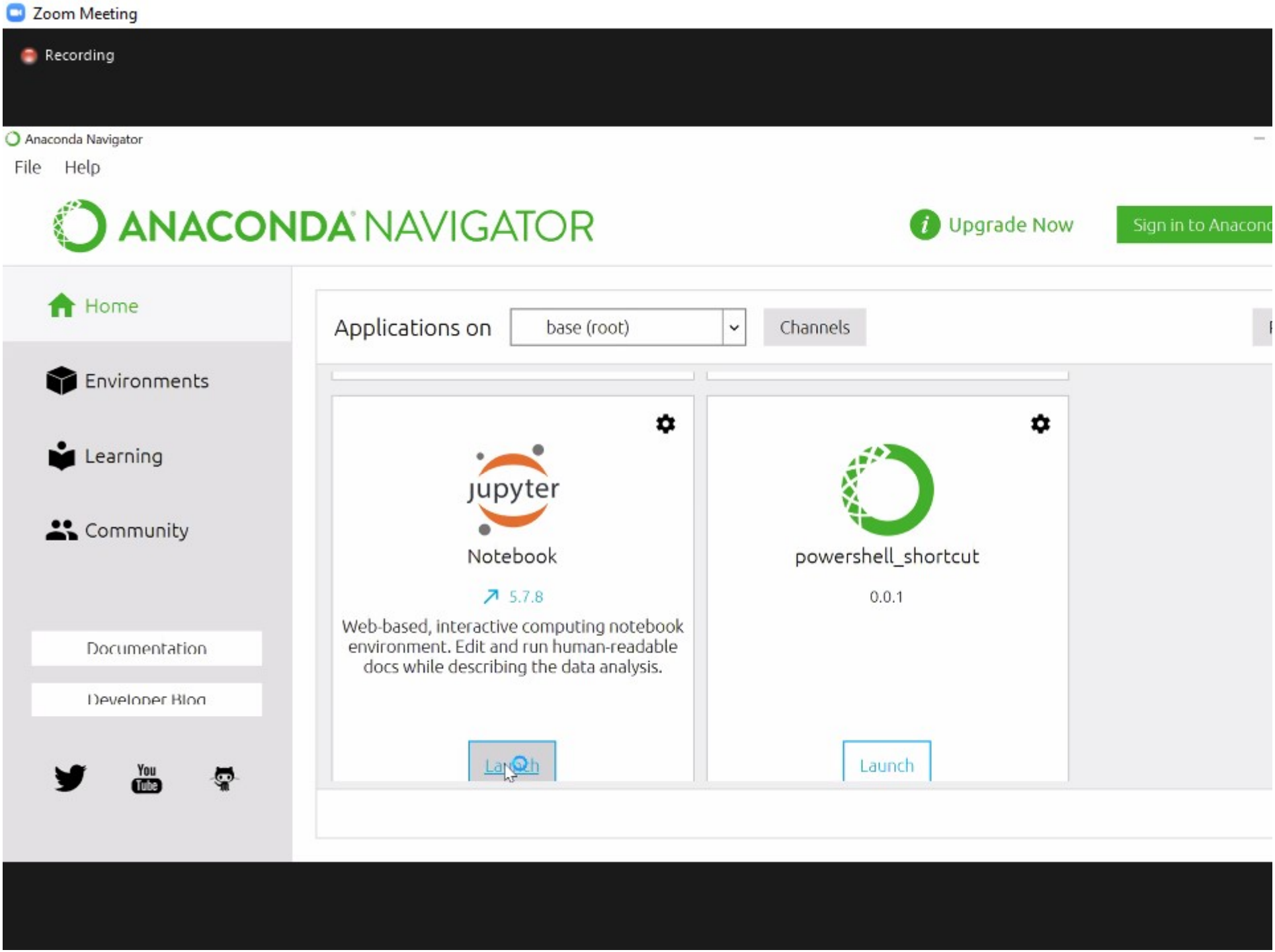

Zoom Meeting
You are viewing Srijan.Sahay's screen
View Options
Recording
Anaconda Navigator
File Help
ANACONDA
Sign in to Anaconc
Home
Environments
Learning
Community
Documentation
Developer Blog
ANACONDA NAVIGATOR
Update Application
X
There's a new version of Anaconda Navigator available. We strongly recommend you to update.
If you click yes, Anaconda Navigator will close and then the Anaconda Navigator Updater will start.
Do you wish to update to Anaconda Navigator 1.9.12 now?
No, don't show again
No, remind me later
Yes
Ok
Ok, and don't show again
Launch
Launch
Unmute
Start Video
63
Participants
Chat
Share Screen
Record

Anaconda IDE Repository

- Install it on D drive
- Anaconda Navigator
- Anaconda Prompt
- It is started with Monty Python others kept maintaining snake names however Guerdo meant a TV Show
- https://towardsdatascience.com/optimizing-jupyter-notebook-tips-tricks-and-nbextensions-26d75d502663
- All additional information, used to specify how a particular notebook/cell/output will be represented, when converted, is stored in the metadata
- Conda list



Recent version have some pending


Zoom Meeting
Recording
Anaconda Navigator
File
Help
ANACONDA NAVIGATOR
Upgrade Now
Sign in to Anacond
Home
Environments
Learning
Community
Documentation
Developer Blog
Applications on
base (root)
Channels
jupyter
Notebook
5.7.8
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
powershell_shortcut
0.0.1
Launch

Data Visualization

- Data visualization allows us to quickly interpret the data and adjust different variables to see their effect
- Technology is increasingly making it easier for us to do so

Why visualize data?

  - Observe the patterns
  - Identify extreme values that could be anomalies
  - Easy interpretation

Popular plotting libraries in Python

Python offers multiple graphing libraries that offers diverse features

• ***matplotlib***	• to create 2D graphs and plots
• ***pandas visualization***	• easy to use interface, built on Matplotlib
• ***seaborn***	• provides a high-level interface for drawing attractive and informative statistical graphics
• ***ggplot***	• based on R's ggplot2, uses Grammar of Graphics
• ***plotly***	• can create interactive plots

Matplotlib

- Matplotlib is a 2D plotting library which produces good quality figures
- Although it has its origins in emulating the MATLAB graphics commands, it is independent of MATLAB
- It makes heavy use of NumPy and other extension code to provide good performance even for large arrays

Scatter Plot

What is a scatter plot?

- A scatter plot is a set of points that represents the values obtained for two different variables plotted on a horizontal and vertical axes

When to use scatter plots?

- Scatter plots are used to convey the relationship between two numerical variables
- Scatter plots are sometimes called correlation plots because they show how two variables are correlated

Importing necessary libraries

```
import pandas as pd
```

'pandas' library to work with dataframes

```
import numpy as np
```

'numpy' library to do numerical operations

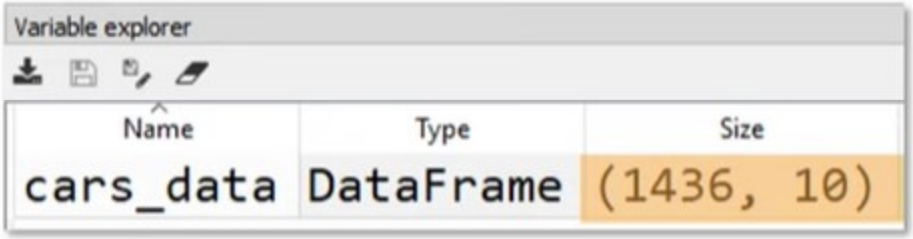
```
import matplotlib.pyplot as plt
```

'matplotlib' library to do visualization

Importing Data

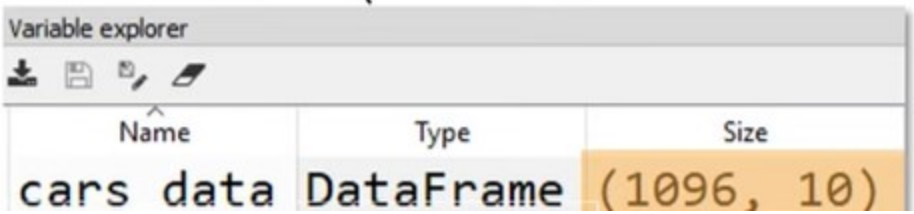
- Importing data

```
cars_data = pd.read_csv('Toyota.csv',index_col=0,
                        na_values=["??","????"])
```



- Removing missing values from the dataframe

```
cars_data.dropna(axis = 0, inplace=True)
```



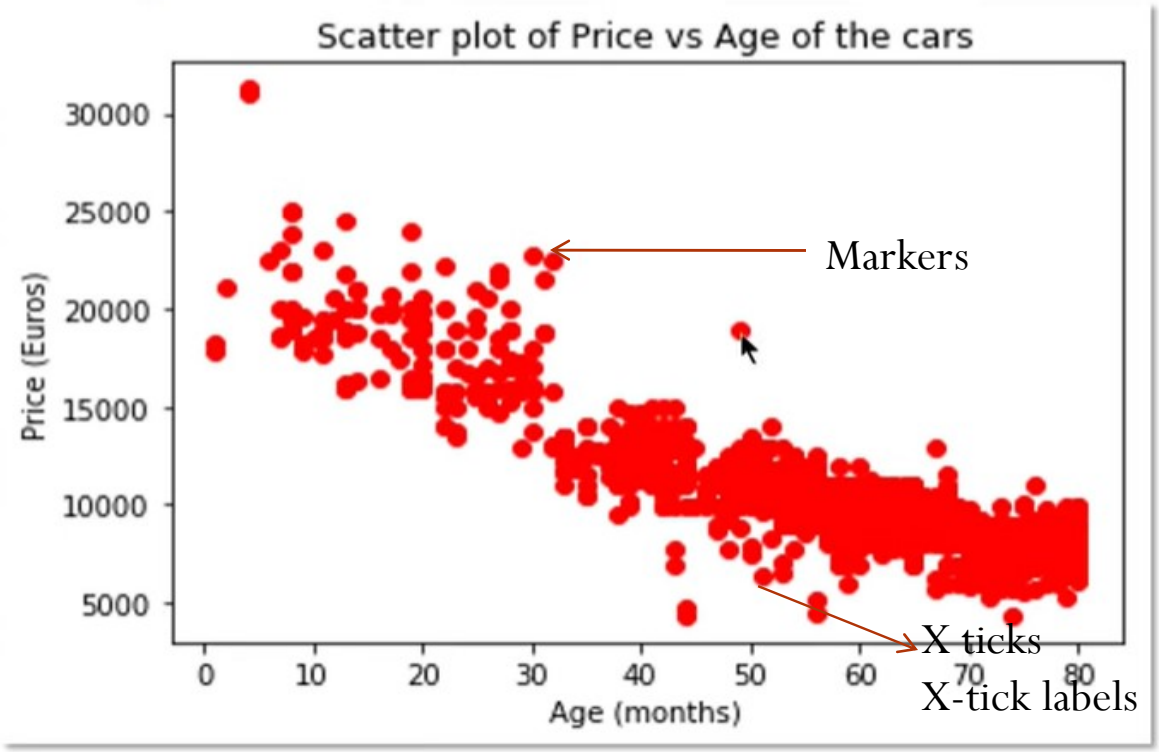
Scatter Plot

x y

```
plt.scatter(cars_data['Age'], cars_data['Price'], c='red')
plt.title('Scatter plot of Price vs Age of the cars')
plt.xlabel('Age (months)')
plt.ylabel('Price (Euros)')
plt.show()
```

Scatter Plot

- The price of the car decreases as age of the car increases



Histogram

What is a histogram?

- It is a graphical representation of data using bars of different heights
- Histogram groups numbers into ranges and the height of each bar depicts the frequency of each range or bin

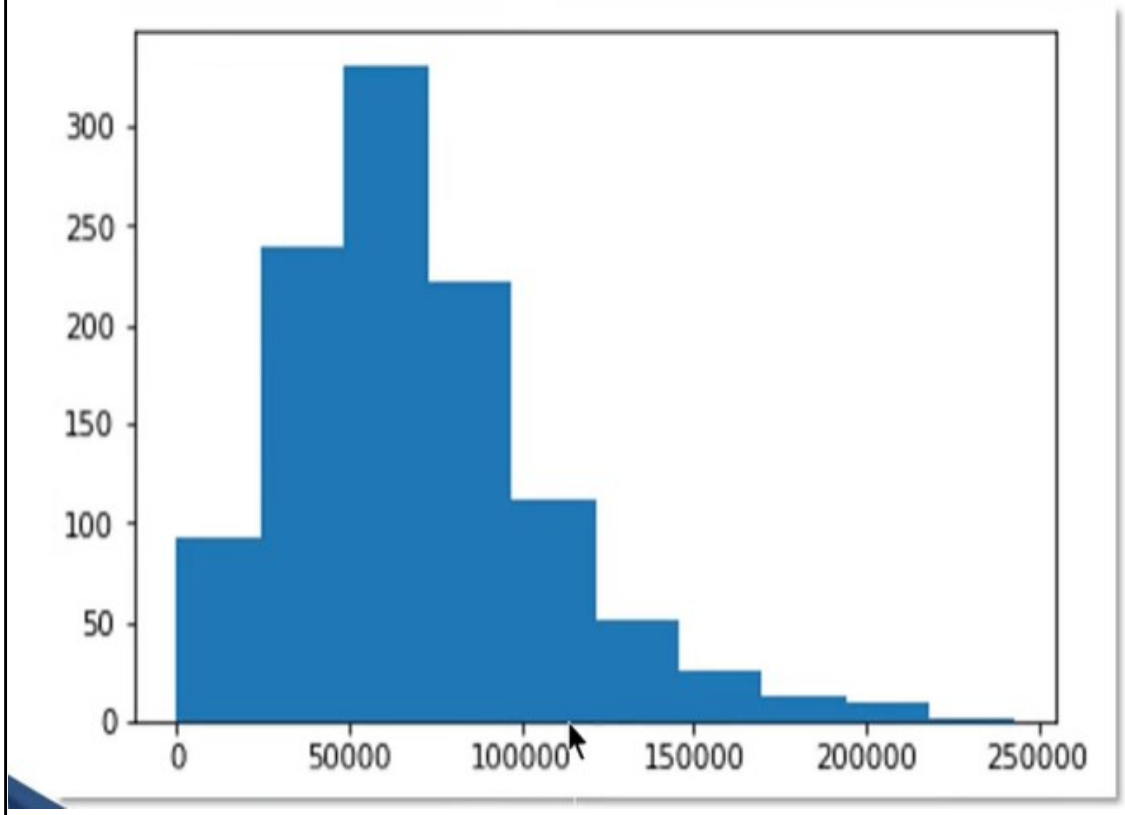
When to use histograms?

- To represent the frequency distribution of numeical variables

x

```
plt.hist(cars_data['KM'])
```

⟶ Histogram with default arguments



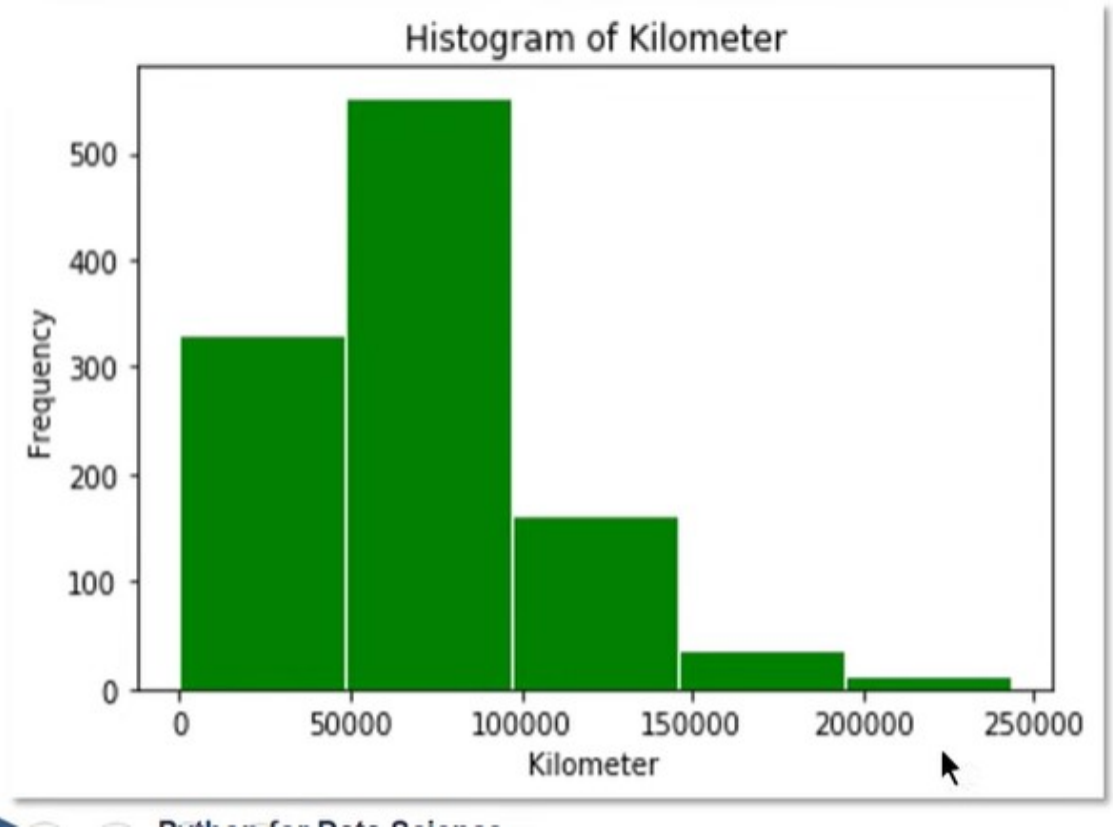
Histogram

```
plt.hist(cars_data['KM'],
         color     = 'green',
         edgecolor = 'white',
         bins      = 5)
plt.title('Histogram of Kilometer')
plt.xlabel('Kilometer')
plt.ylabel('Frequency')

plt.show()
```

Histogram

- Frequency distribution of kilometre of the cars shows that most of the cars have travelled between 50000 – 100000 km and there are only few cars with more distance travelled



Bar plot

What is a bar plot?

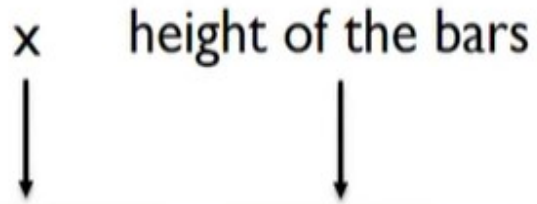
- A bar plot is a plot that presents categorical data with rectangular bars with lengths proportional to the counts that they represent

When to use bar plot?

- To represent the frequency distribution of categorical variables
- A bar diagram makes it easy to compare sets of data between different groups

Bar plot

```
counts    = [979, 120, 12]
fuelType = ('Petrol', 'Diesel', 'CNG')
index     = np.arange(len(fuelType))
```




```
plt.bar(index, counts, color=['red', 'blue', 'cyan'])
plt.title('Bar plot of fuel types')
plt.xlabel('Fuel Types')
plt.ylabel('Frequency')
plt.show()
```

Bar plot

```
counts    = [979, 120, 12]
fuelType = ('Petrol', 'Diesel', 'CNG')
index     = np.arange(len(fuelType))
```

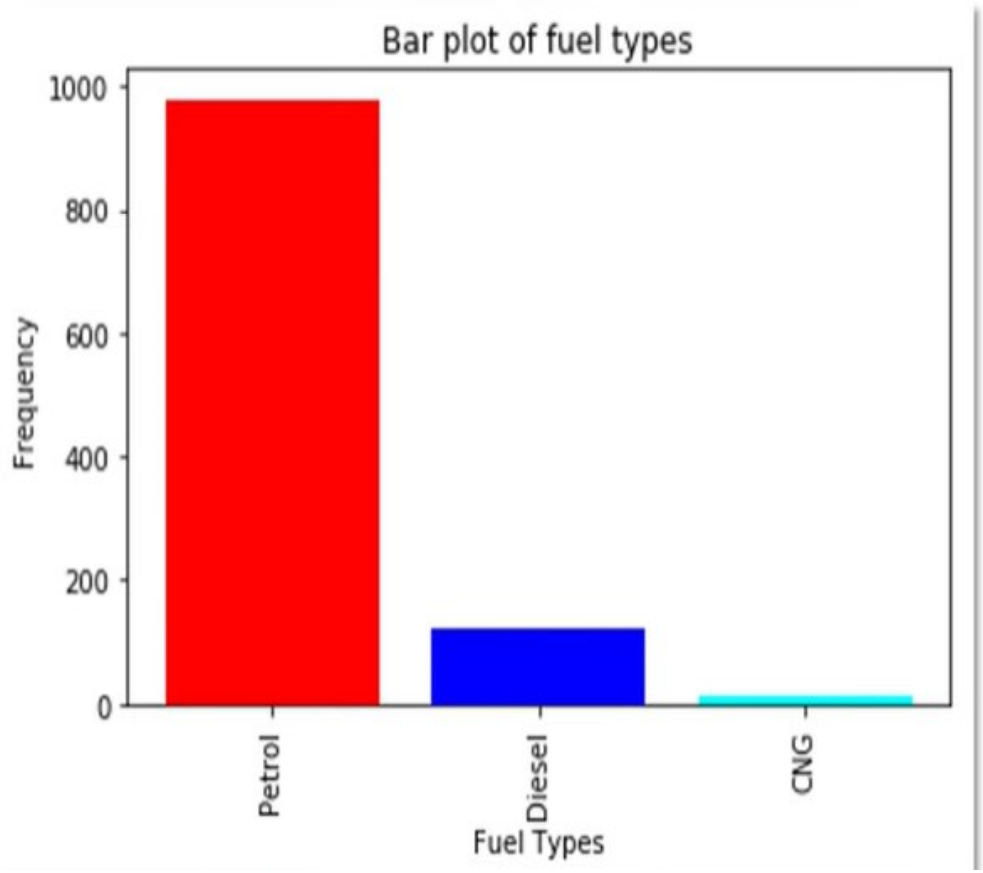
x height of the bars

```
plt.bar(index, counts, color=['red', 'blue', 'cyan'])
plt.title('Bar plot of fuel types')
plt.xlabel('Fuel Types')
plt.ylabel('Frequency')
plt.xticks(index, fuelType,rotation = 90)
plt.show()
```

Set the labels of the xticks

Bar plot

- Bar plot of fuel type shows that most of the cars have petrol as fuel type



Summary

We have learnt how to create basic plots using *matplotlib* library

- Scatter plot
- Histogram
- Bar plot

We will learn how to create basic plots using *seaborn* library:

- Scatter plot
- Histogram
- Bar plot
- Box and whiskers plot
- Pairwise plots

Seaborn

- Seaborn is a Python data visualization library based on matplotlib
- It provides a high-level interface for drawing attractive and informative statistical graphics

Scatter Plot

Importing libraries

- Importing necessary libraries

```
import pandas as pd
```
→ 'pandas' library to work with dataframes

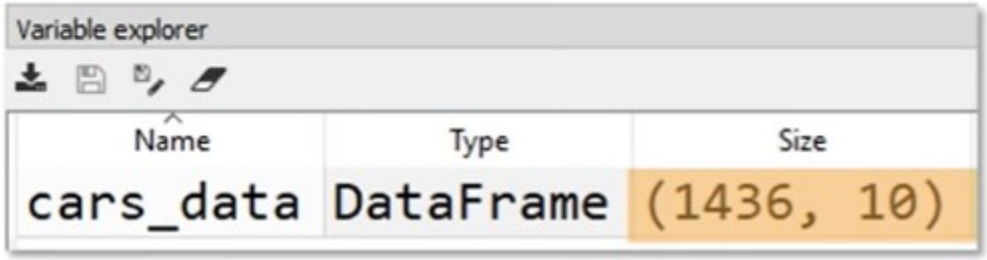
```
import numpy as np
```
→ 'numpy' library to do numerical operations

```
import matplotlib.pyplot as plt
```
→ 'matplotlib' library to do visualization

```
import seaborn as sns
```
→ 'seaborn' library to do visualization

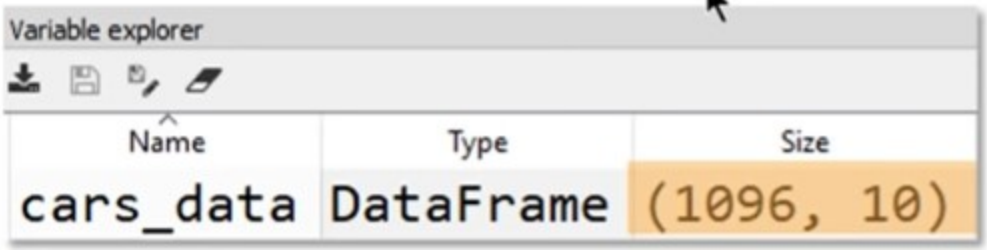
- Importing data

```
cars_data = pd.read_csv('Toyota.csv',index_col=0,
                        na_values=["??","????"])
```



- Removing missing values from the dataframe

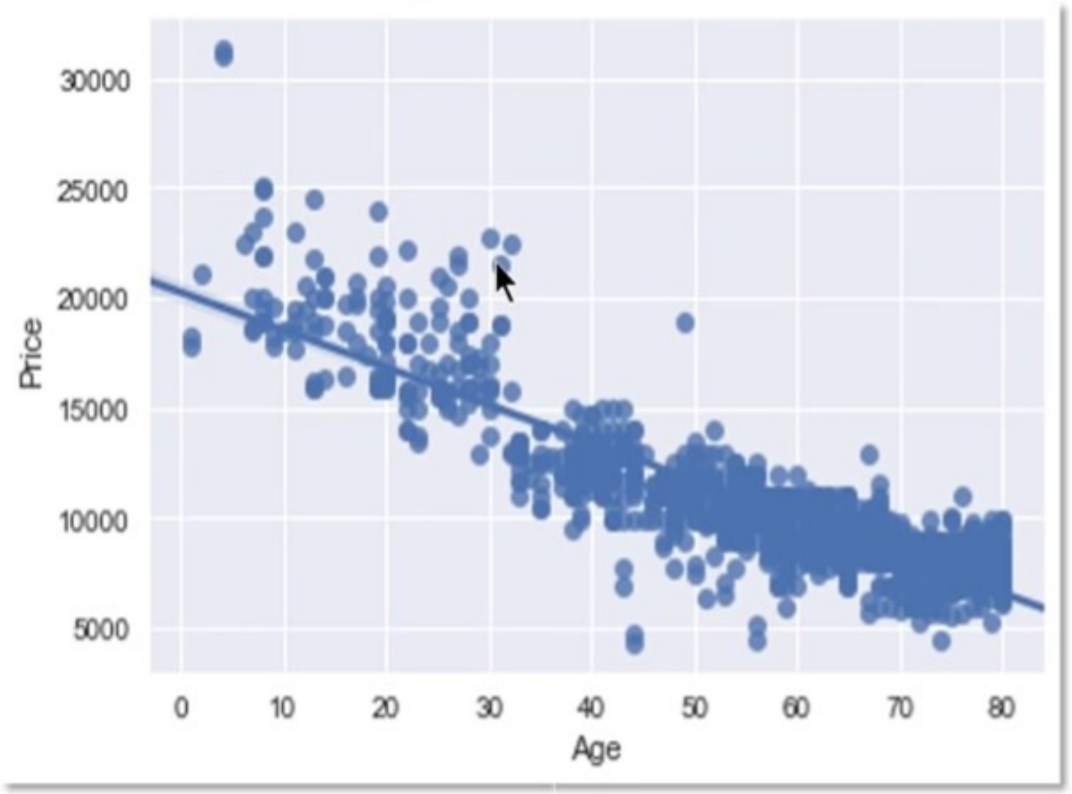
```
cars_data.dropna(axis = 0, inplace=True)
```



Scatter plot

- Scatter plot of *Price vs Age* with default arguments

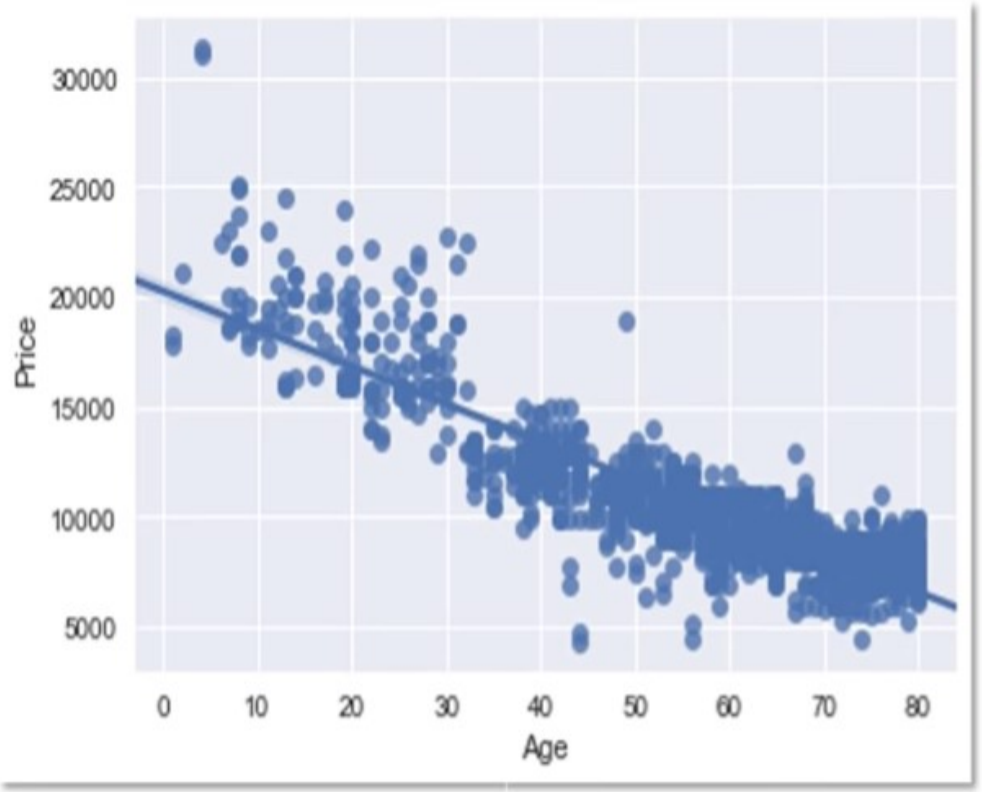
```
sns.set(style="darkgrid")
sns.regplot(x=cars_data['Age'], y=cars_data['Price'])
```



Scatter plot

- Scatter plot of *Price vs Age* with default arguments

```
sns.set(style="darkgrid")
sns.regplot(x=cars_data['Age'], y=cars_data['Price'])
```

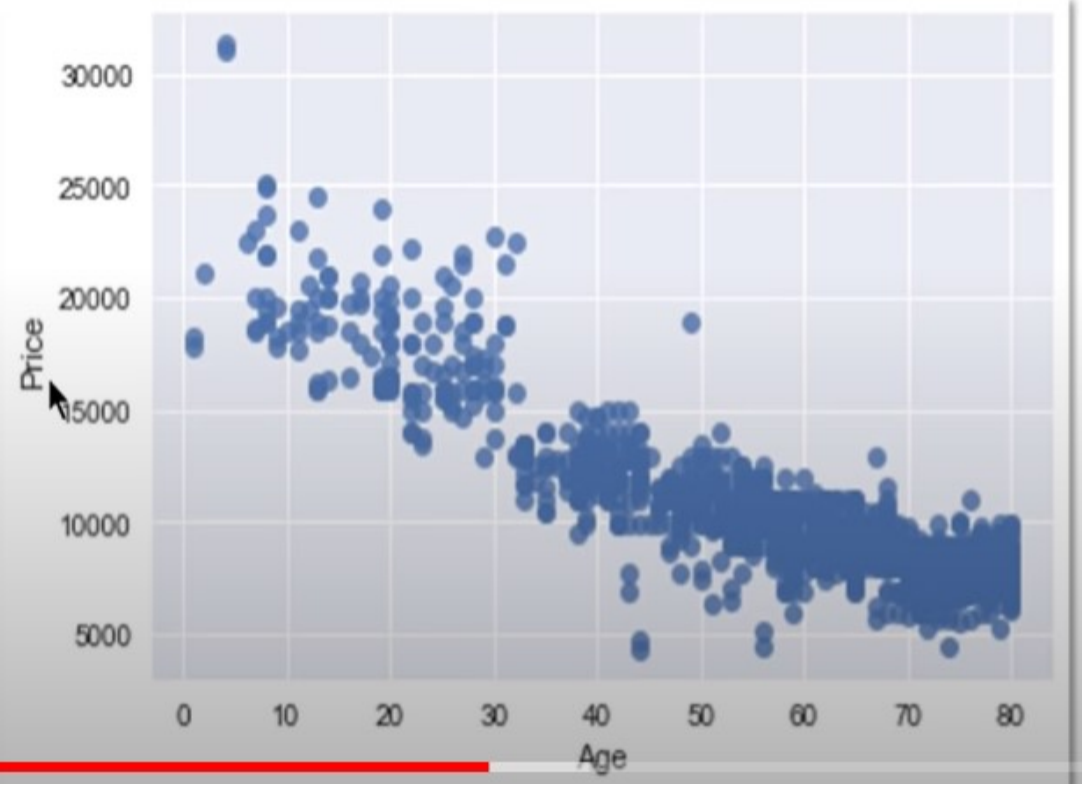


- By default, `fit_reg = True`
- It estimates and plots a regression model relating the x and y variables

Scatter plot

- Scatter plot of *Price vs Age* without the regression fit line

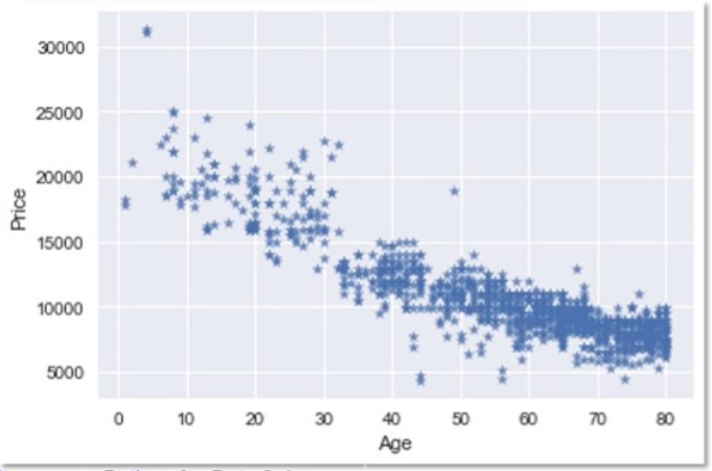
```
sns.regplot(x=cars_data['Age'], y=cars_data['Price'],
            fit_reg=False)
```



Scatter plot

- Scatter plot of *Price vs Age* by customizing the appearance of markers

```
sns.regplot(x=cars_data['Age'], y=cars_data['Price'],
            marker="*", fit_reg=False)
```



Scatter plot

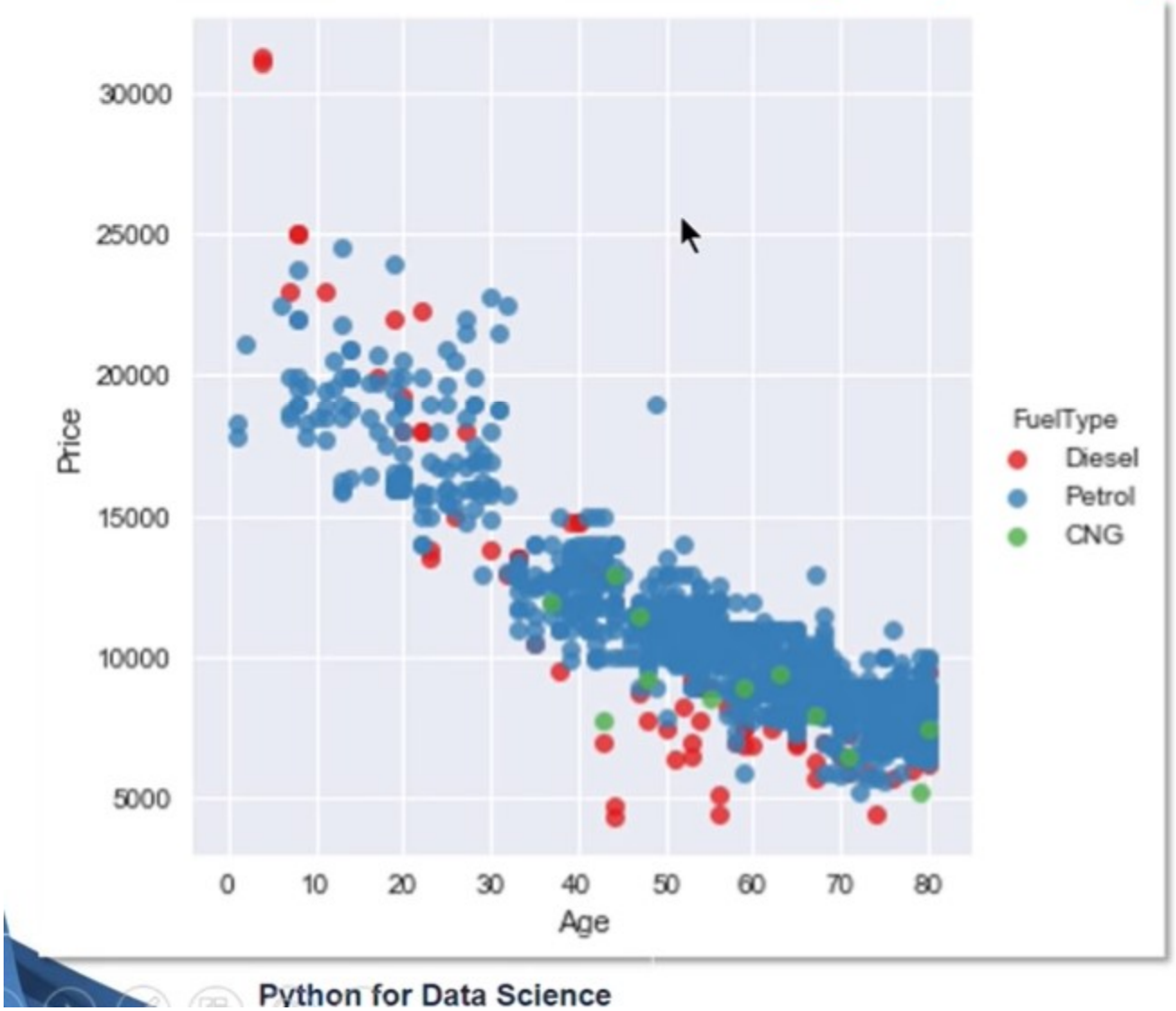
- Scatter plot of *Price vs Age by FuelType*

- Using hue parameter, including another variable to show the fuel types categories with different colors

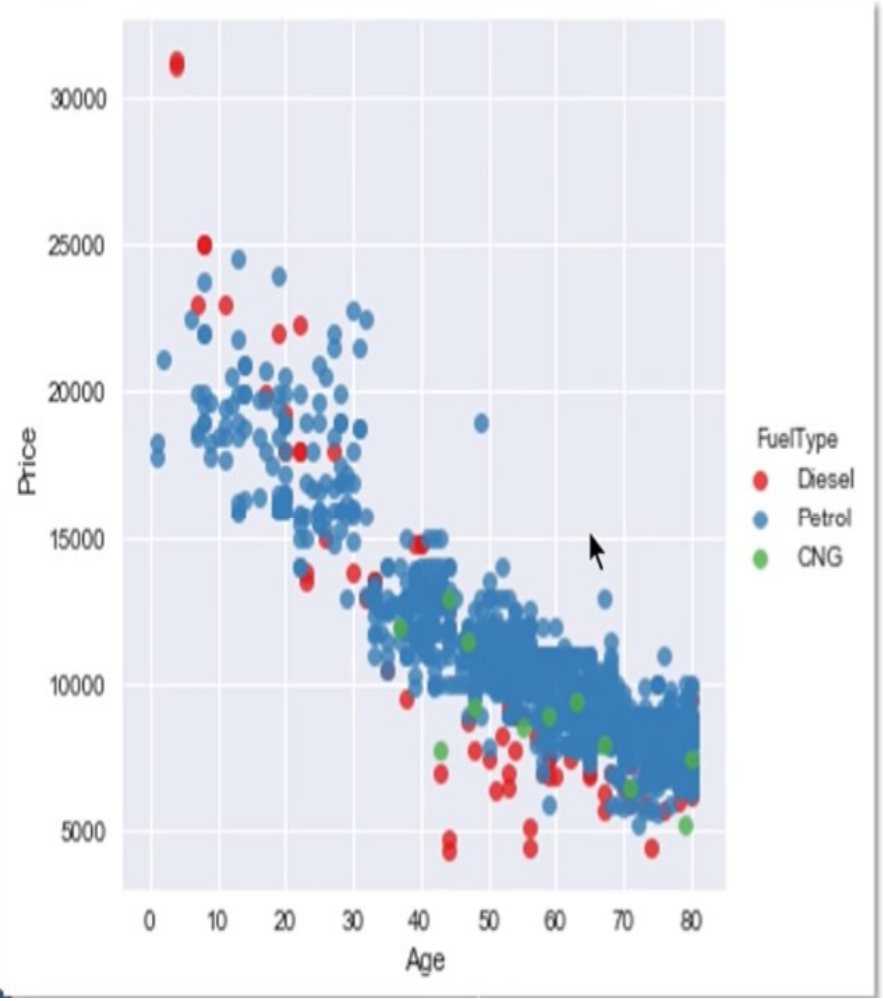
```
sns.lmplot(x='Age', y='Price', data=cars_data,
           fit_reg=False, hue='FuelType',
           legend=True, palette="Set1")
```

Scatter plot

- Scatter plot of *Price vs Age by FuelType*



- Scatter plot of *Price vs Age by FuelType*



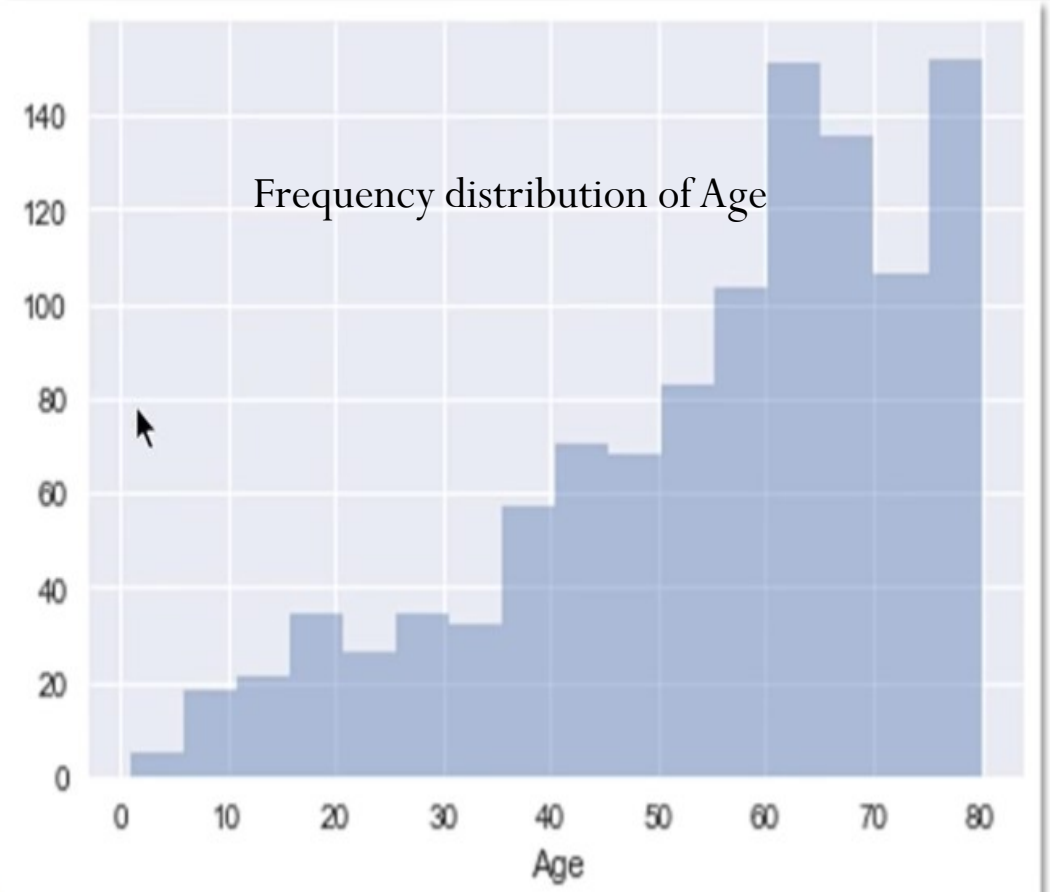
Similarly, custom the appearance of the markers using

- transparency
- shape
- size

Histogram

- Histogram without kernel density estimate

```
sns.distplot(cars_data['Age'],kde=False)
```



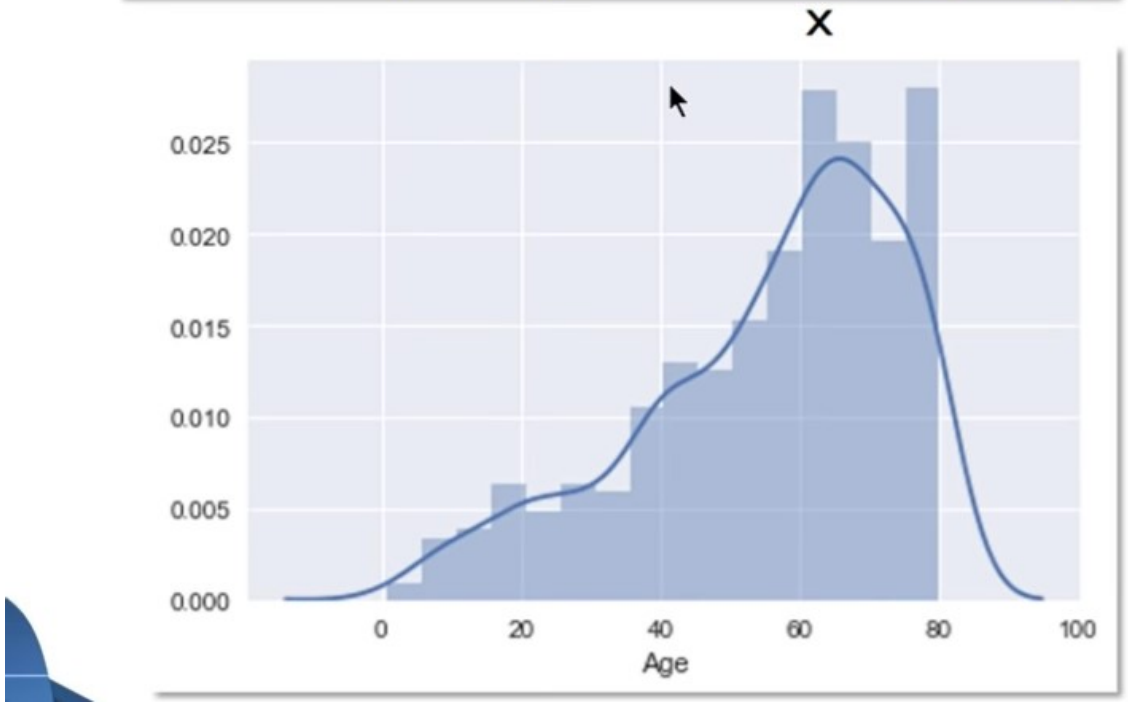
A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analagous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.

KDE===Distribution of variable age

Histogram

- Histogram with default kernel density estimate

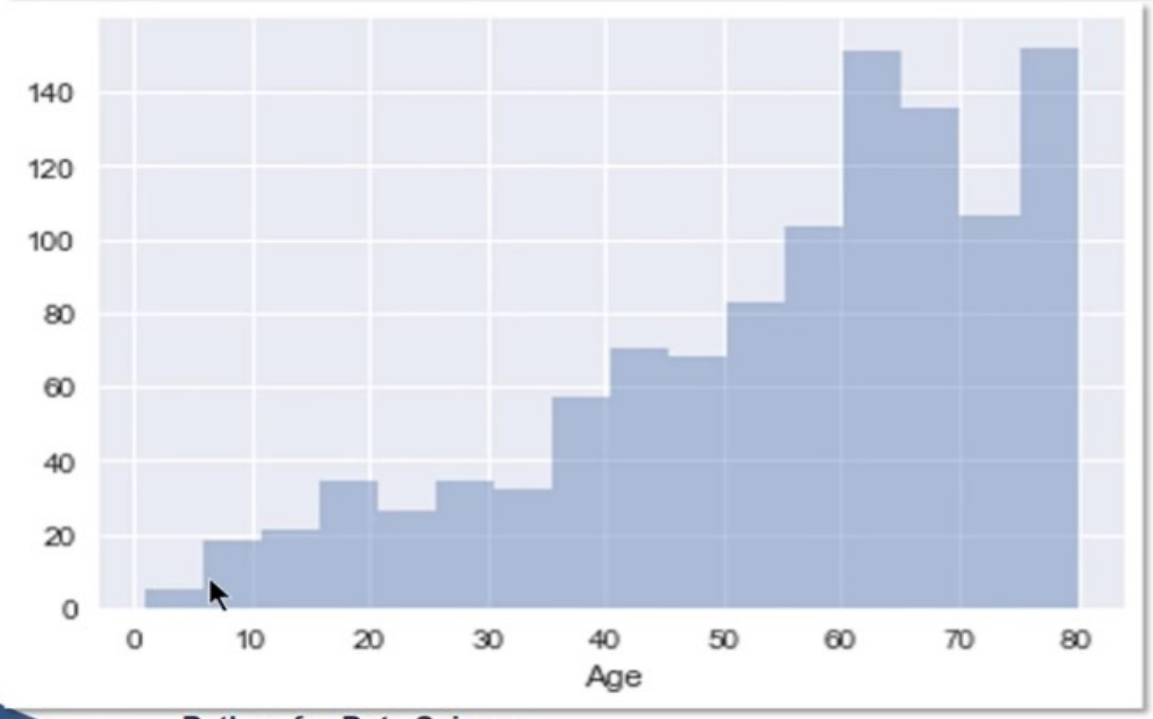
```
sns.distplot(cars_data['Age'] )
```



Histogram

- Histogram without kernel density estimate

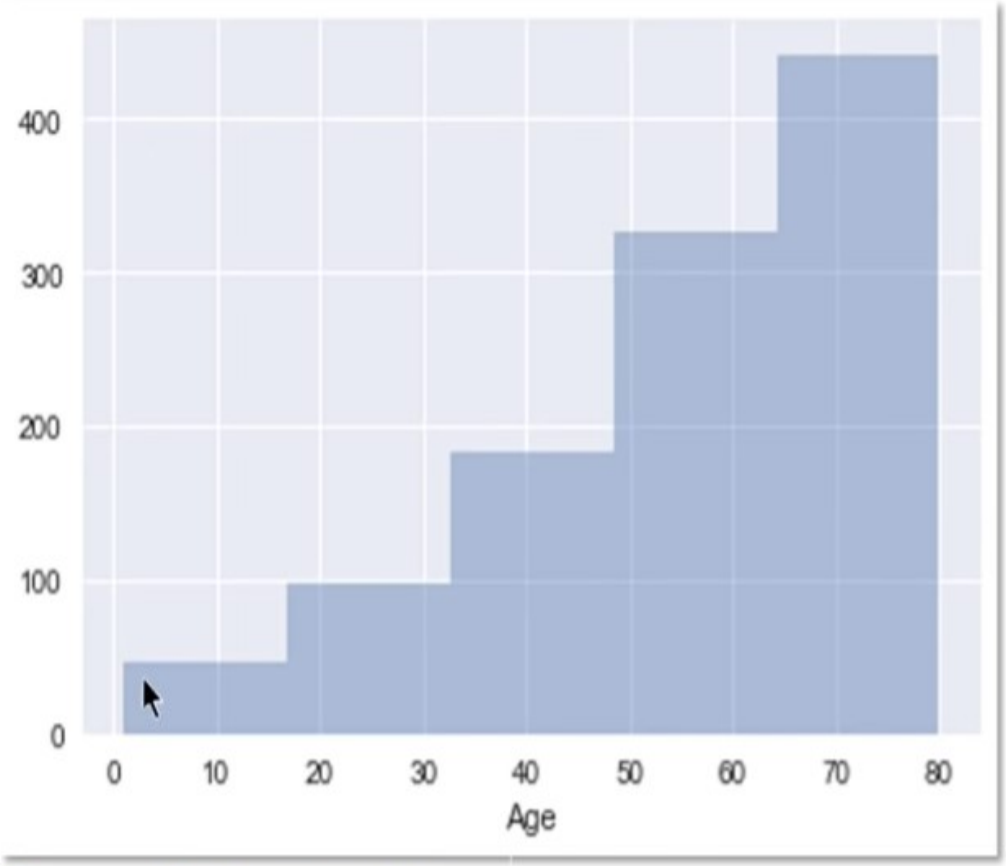
```
sns.distplot(cars_data['Age'],kde=False)
```



Histogram

- Histogram with fixed no. of bins

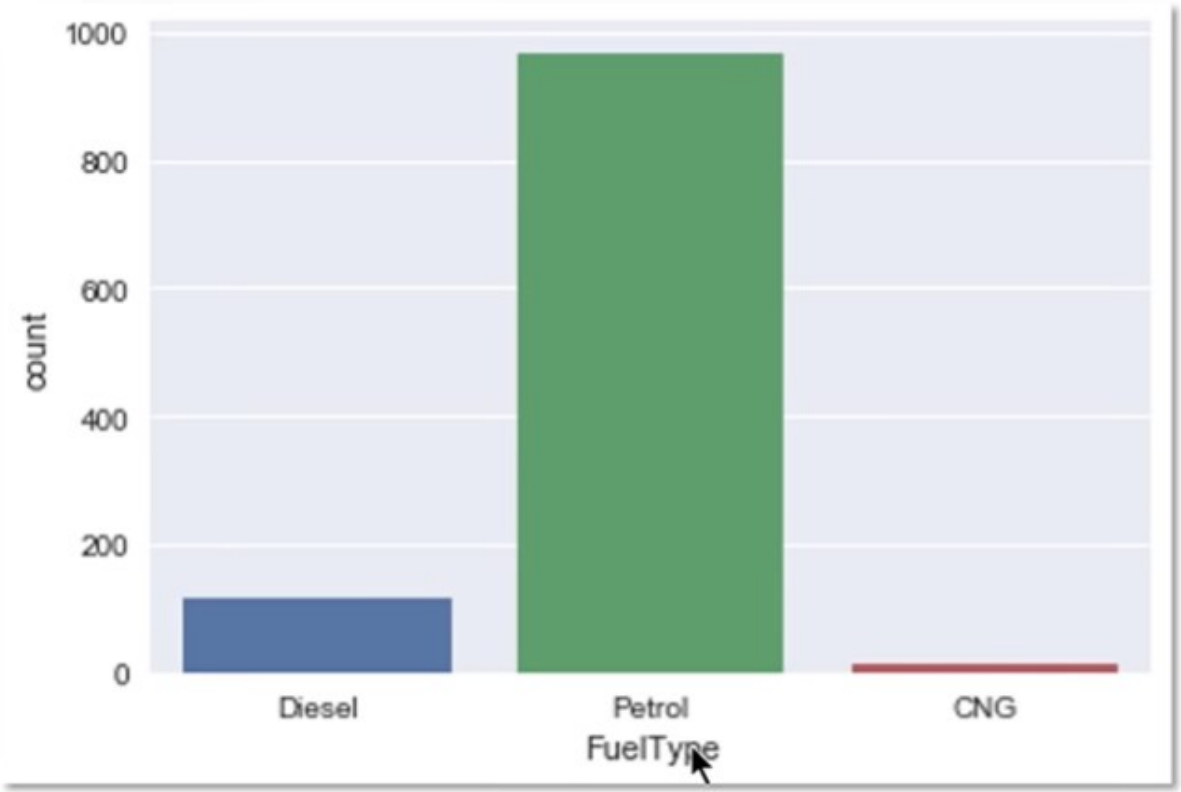
```
sns.distplot(cars_data['Age'], kde = False, bins=5 )
```



Bar plot

- Frequency distribution of fuel type of the cars

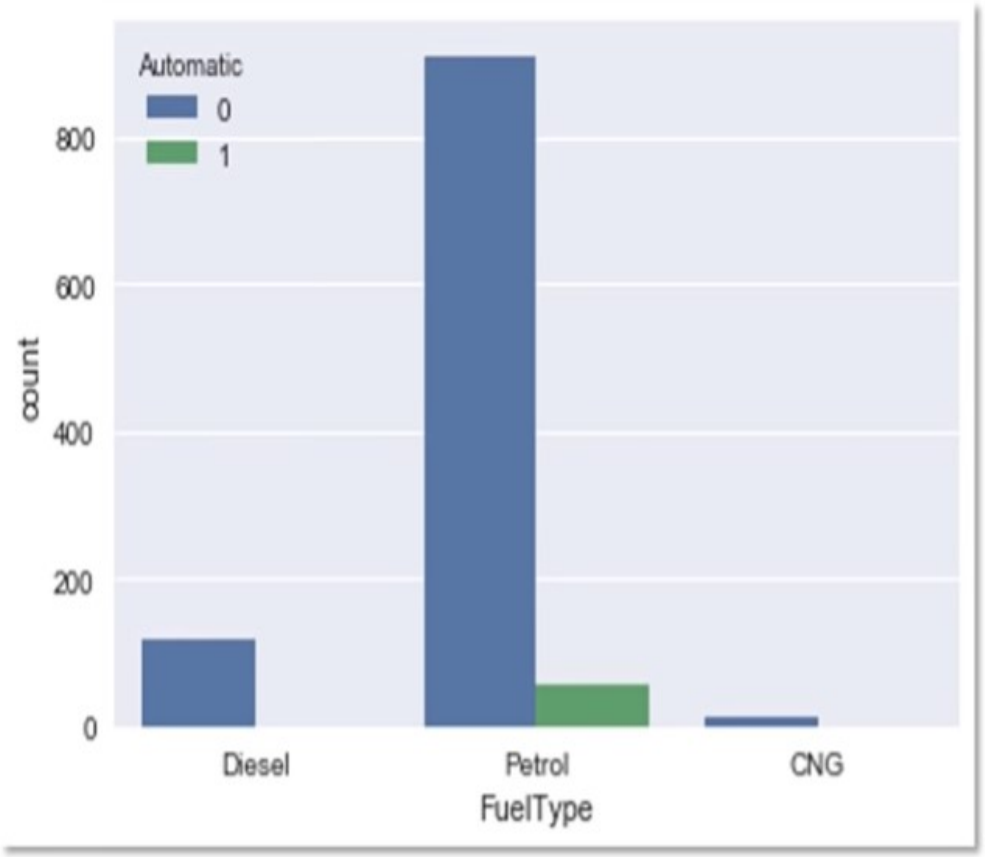
```
sns.countplot(x="FuelType", data=cars_data)
```



Grouped bar plot

- Grouped bar plot of *FuelType* and *Automatic*

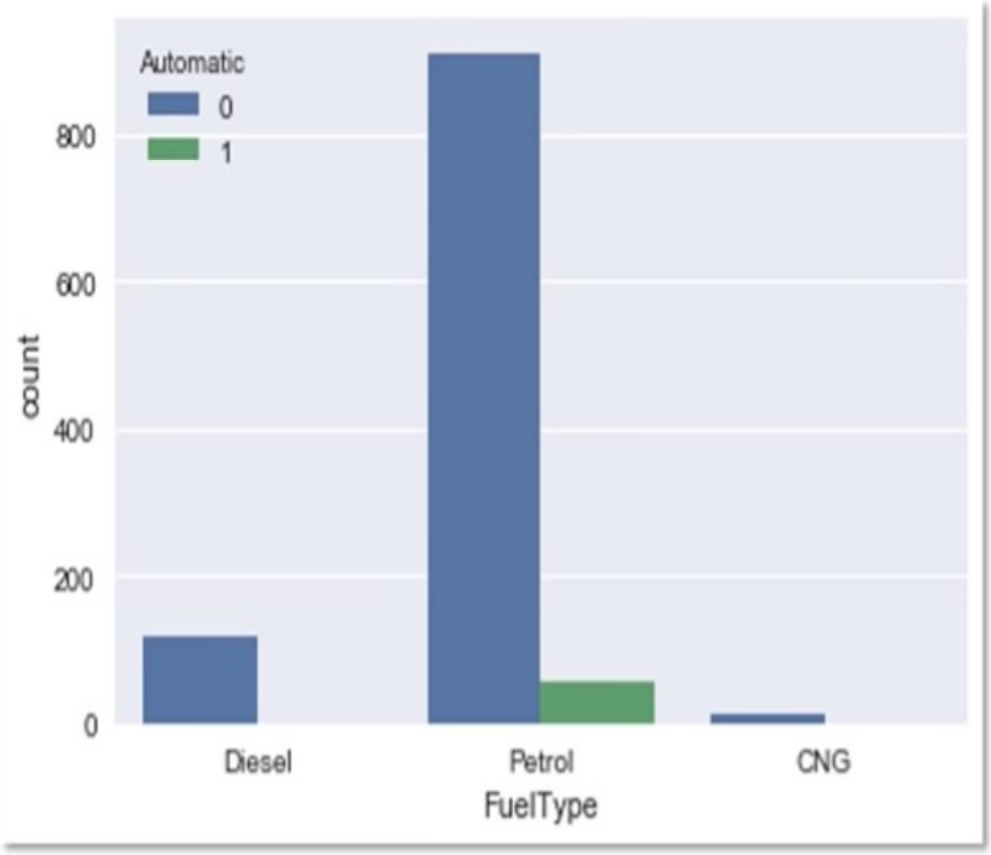
```
sns.countplot(x="FuelType", data=cars_data, hue = "Automatic")
```



Grouped bar plot

- Grouped bar plot of *FuelType* and *Automatic*

```
sns.countplot(x="FuelType", data=cars_data, hue = "Automatic")
```




```
pd.crosstab(index   = cars_data['Automatic'],
            columns = cars_data2['FuelType'],
            dropna  = True)
```

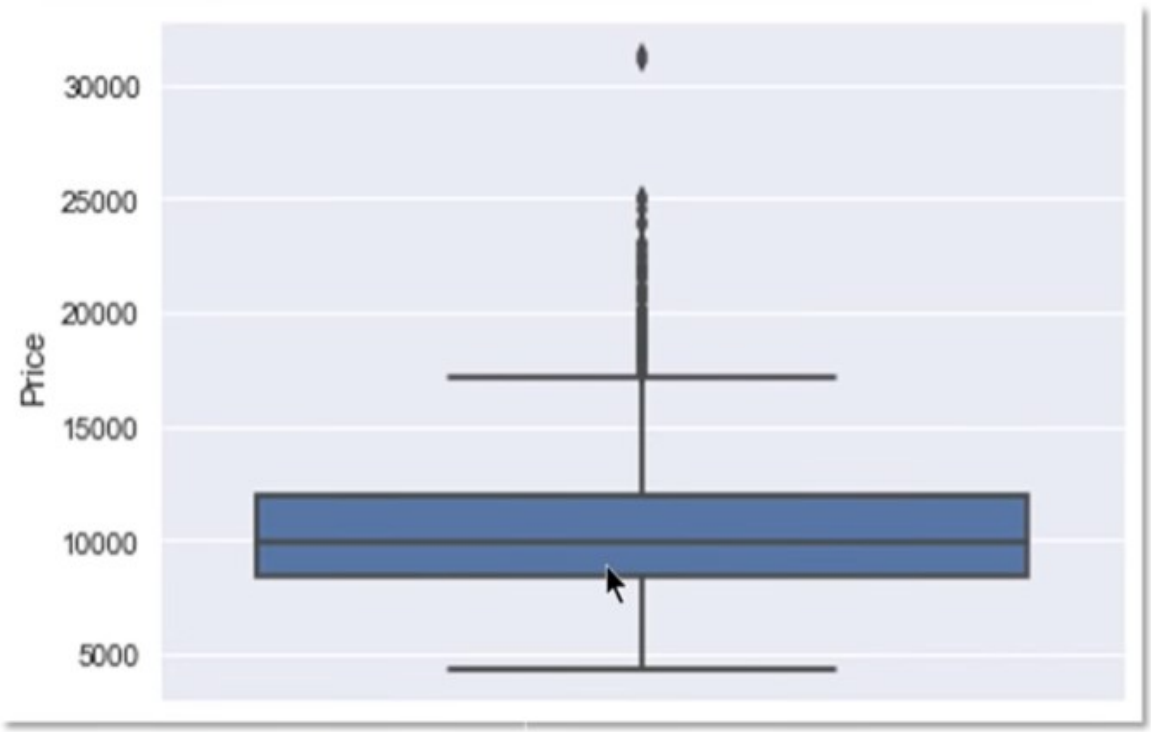
Out[5]:

FuelType	CNG	Diesel	Petrol
Automatic			
0	15	144	1104
1	0	0	73

Box and whiskers plot – numerical variable

- Box and whiskers plot of *Price* to visually interpret the five-number summary

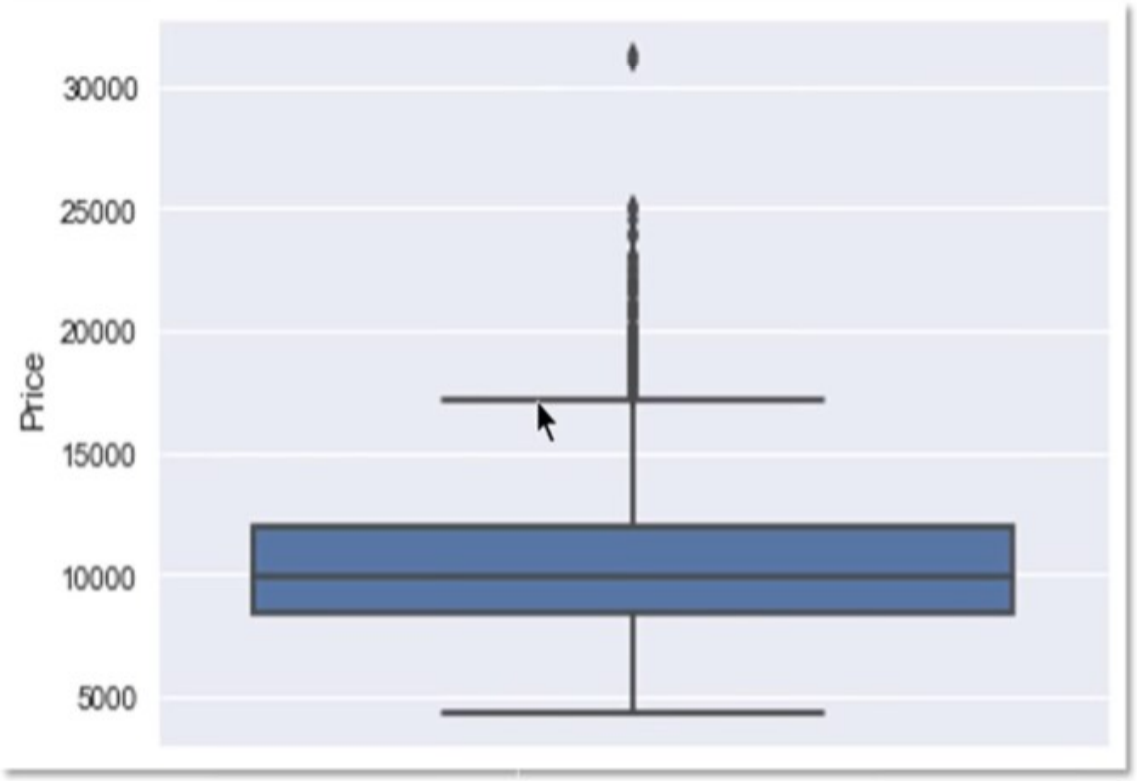
```
sns.boxplot(y=cars_data["Price"] )
```



Box and whiskers plot – numerical variable

- Box and whiskers plot of *Price* to visually interpret the five-number summary

```
sns.boxplot(y=cars_data["Price"] )
```



BOXPLOT

A boxplot is a powerful data visualization tool used to understand the distribution of data. It splits the data into quartiles, and summarises it based on five numbers derived from these quartiles:

- **median:** the middle value of data. marked as Q2, portrays the 50th percentile.
- **first quartile:** the middle value between "minimum non-outlier" and median. marked as Q1, portrays the 25th percentile.
- **third quartile:** the middle value between "maximum non-outlier" and median. marked as Q3, portrays the 75th percentile.
- **"maximum non-outlier":** calculated by (Q3 + 1.5*IQR). All values above this are considered outliers.
- **"minimum non-outlier":** calculated by (Q1 – 1.5*IQR). All values below this are considered outliers.

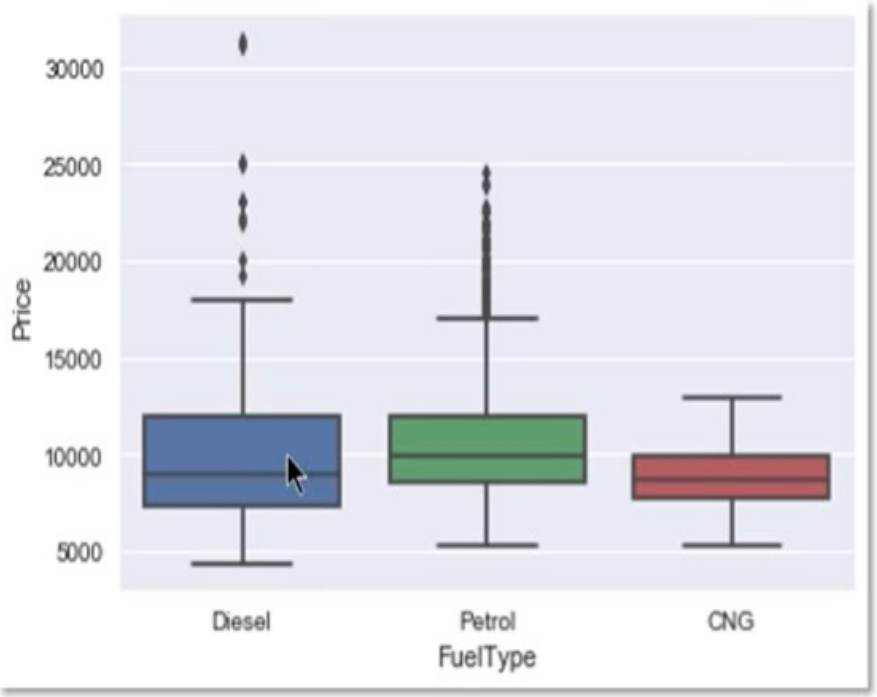
IQR (Inter Quartile Range)

In Python 3, We can graph a boxplot using three methods, using matplotlib, using pandas, or using seaborn. Here, we will use seaborn, which is a matplotlib wrapper that provides close integration with pandas data structures and better palette options than matplotlib. We will use seaborn.boxplot() method, and then we will learn how to show mean on boxplot.

Box and whiskers plot

- Box and whiskers plot for numerical vs categorical variable
- Price of the cars for various fuel types

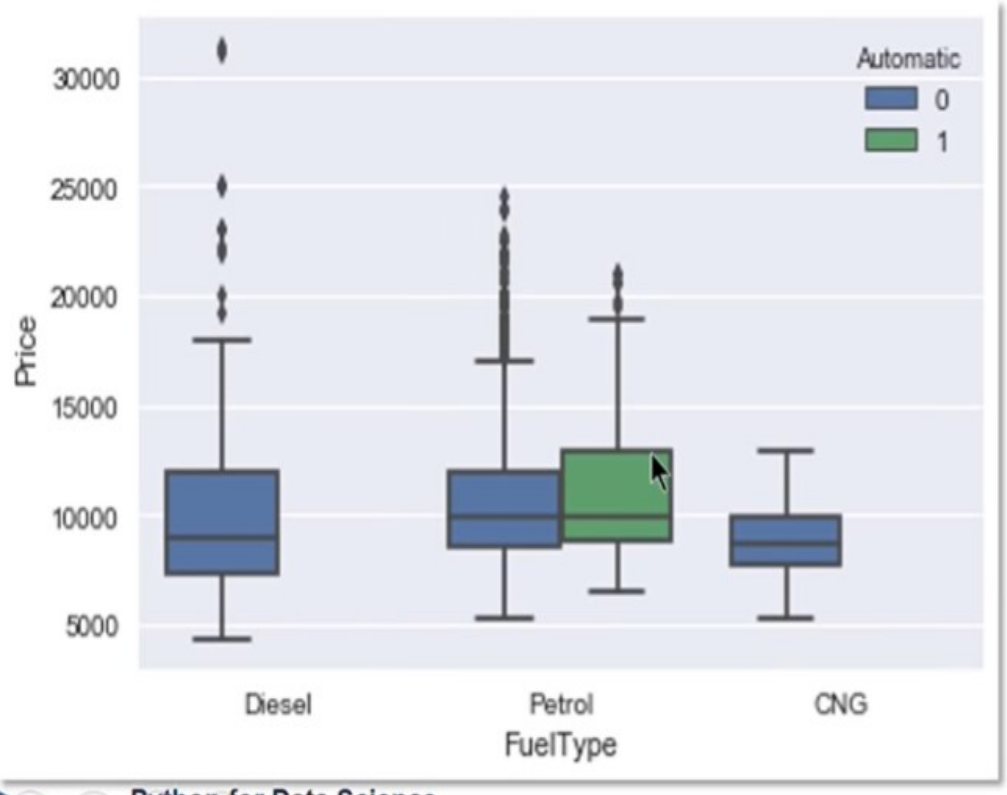
```
sns.boxplot(x = cars_data['FuelType'], y = cars_data["Price"])
```



Grouped box and whiskers plot

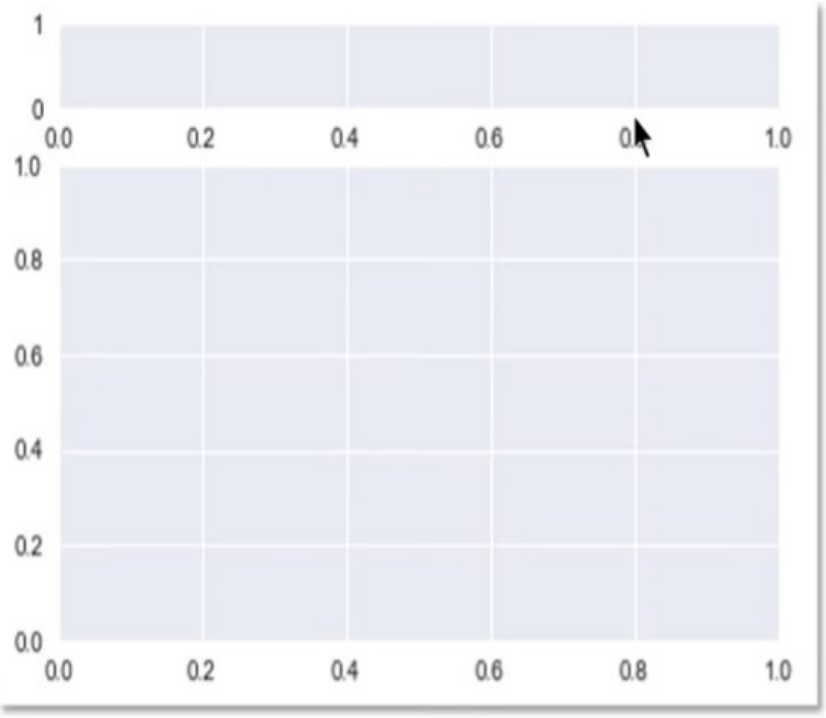
- Grouped box and whiskers plot of *Price* vs *FuelType* and *Automatic*

```
sns.boxplot(x = "FuelType",  y = cars_data["Price"],
            hue = "Automatic", data = cars_data)
```



Box-whiskers plot and Histogram

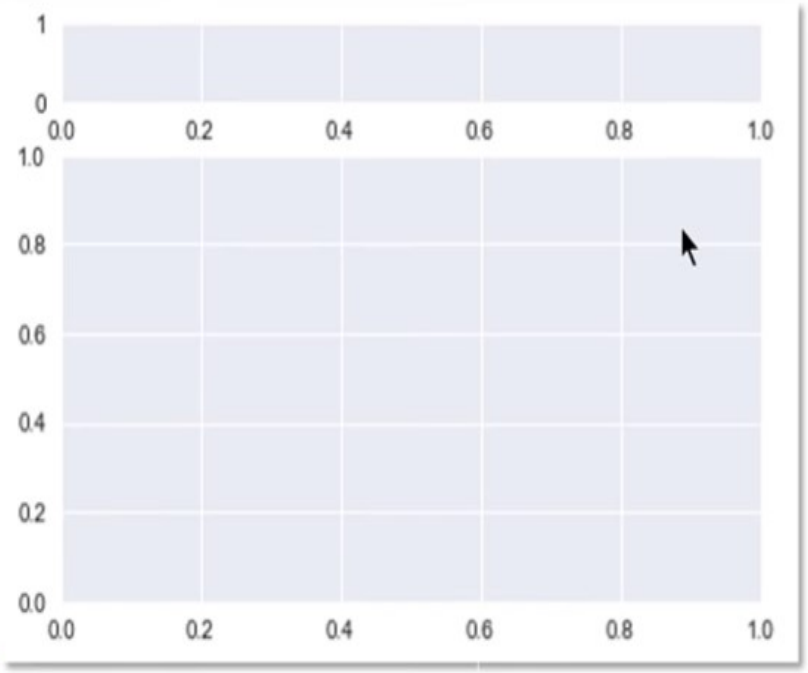
- Let's plot box-whiskers plot and histogram on the same window
- Split the plotting window into 2 parts



Box-whiskers plot and Histogram

- Let's plot box-whiskers plot and histogram on the same window
- Split the plotting window into 2 parts

```
f,(ax_box, ax_hist)=plt.subplots(2, gridspec_kw={"height_ratios": (.15, .85)}
```

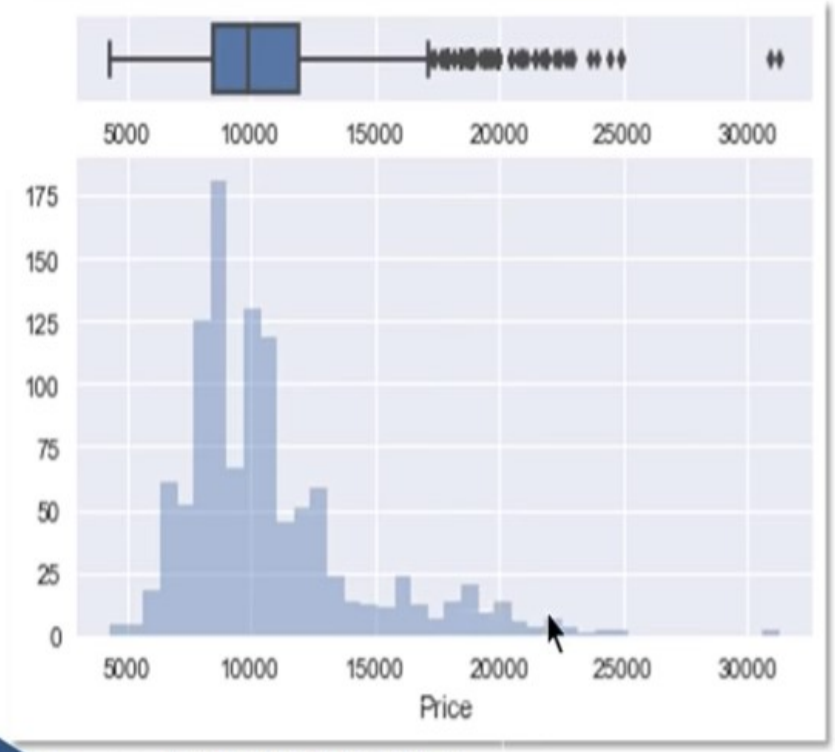


Box-whiskers plot and Histogram

- Now, add create two plots

```
sns.boxplot(cars_data["Price"] , ax=ax_box)

sns.distplot(cars_data["Price"], ax=ax_hist, kde = False)
```

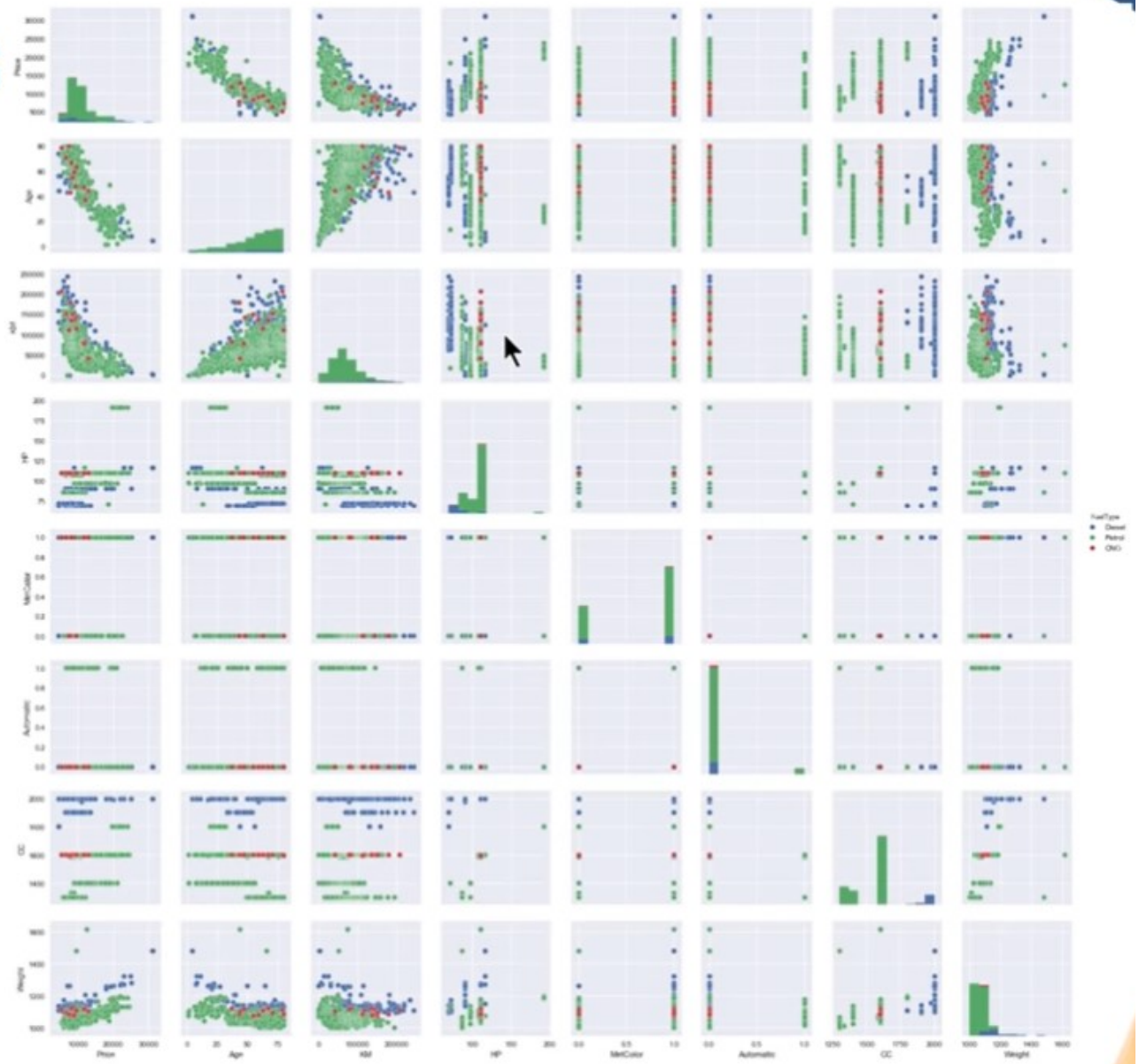


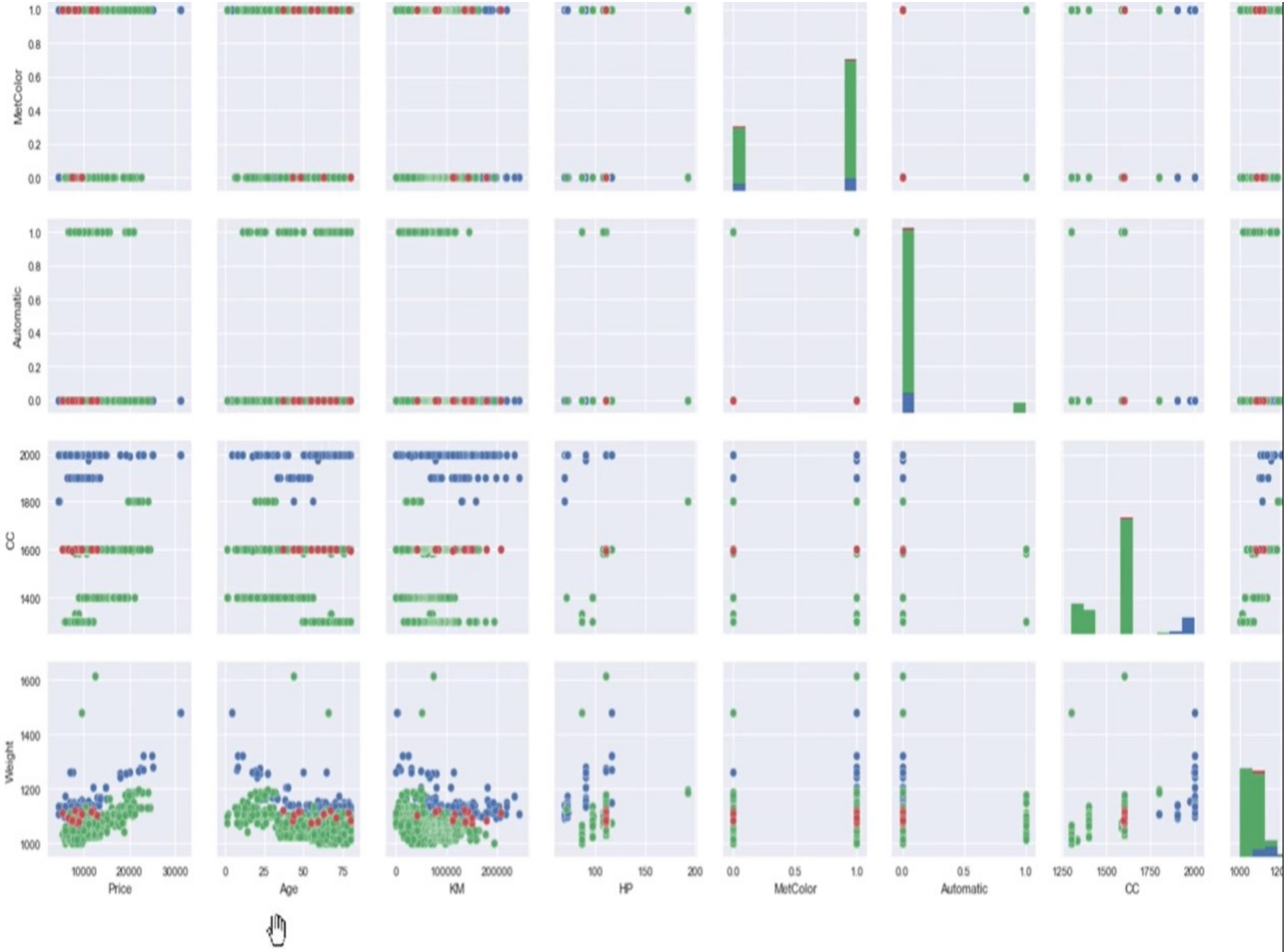
Pairwise plots

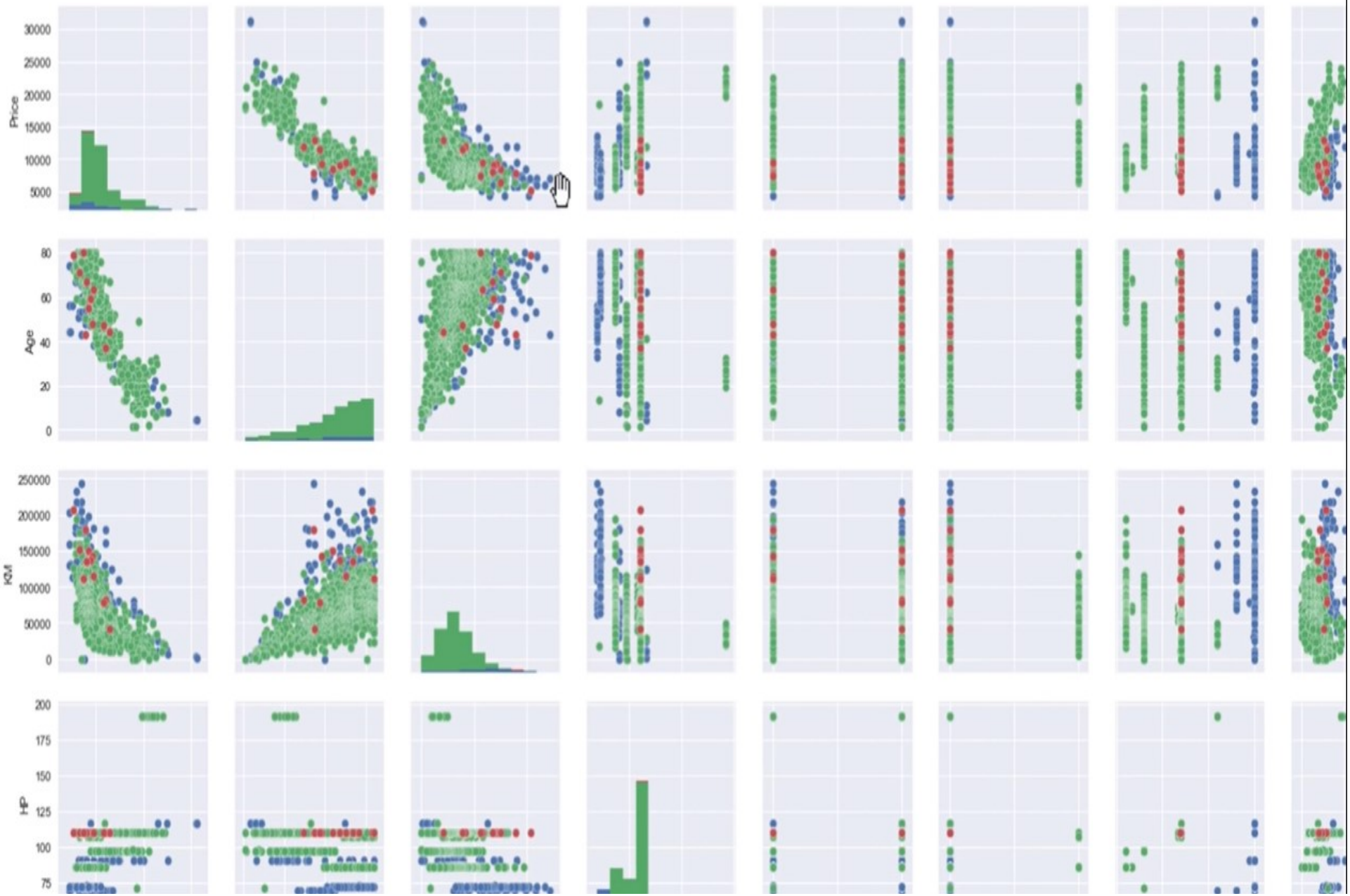
- It is used to plot pairwise relationships in a dataset
- Creates scatterplots for joint relationships and histograms for univariate distributions

```
sns.pairplot(cars_data, kind="scatter", hue="FuelType")
plt.show()
```

Pairwise plots




MetColor
Automatic
CC
Weight
Price
Age
KM
HP
MetColor
Automatic
CC


Price
30000
25000
20000
15000
10000
5000
Age
80
60
40
20
0
KM
250000
200000
150000
100000
50000
0
HP
200
175
150
125
100
75

Summary

We have learnt how to create basic plots using *seaborn* library:

- Scatter plot
- Histogram
- Bar plot
  - Grouped bar plot
- Box and whiskers plot
  - Grouped box and whiskers plot
- Pairwise plots