## Exercise 8: Plotting an Analitycal Function

Create a basic plot using Matplotlib libraries, which visualizes two functions of two variables. Use MALTAB way of plotting.

$$f_1(x) = 4x^2$$
$$f_2(x) = x^4 - 12x^2 + 20$$
$$x \in \langle -4, 4 \rangle$$

**Hint:** use `plt.plot(x, y)` method.

## Exercise 9: Creating a Graph

There are several ways to create a graph with Matplotlib. The first one is closely related to the MATLAB way of doing it, called Pyplot. Pyplot is an API that is a state-based interface for Matplotlib, meaning that it keeps the configurations and other parameters in the object itself. Pyplot is intended as a simple case.

1. Import required libraries

2. Use the second API, called object-oriented API – intended for more complex plots. It allows more flexibility and configuration. Accessing this API you can create figure and axes using plt.subplot module:

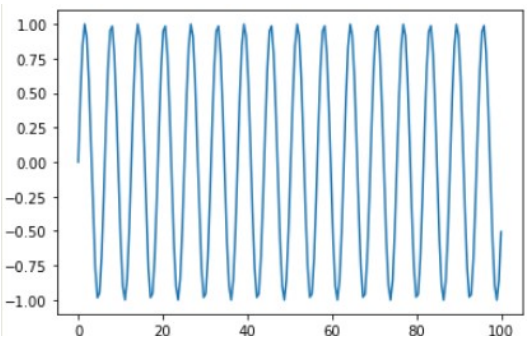
```
fig, ax = plt.subplots()
```

3. To add a plot to a graph created using the object-oriented API, use the following command:

```
x = np.linspace(0,100,500)
y = np.sin(2*np.pi*x/100)
ax.plot(x, y)
```

## Exercise 10: Creating a Graph for a Mathematical Function

Create a plot using the object-oriented API with the NumPy sin() function on the interval [0,100]



## Exercise 11: Creating Line Grapgh Using Different Libraries

Let's compare the creation process between Matplotlib, Pandas, and Seaborn. We will create a Pandas DataFrame with random values and plot it using various methods.

1. Create a dataset with random values:

```
import numpy as np
X = np.arange(0,100)
Y = np.random.randint(0,200, size=X.shape[0])
```

2. Plot the data using the **Matplotlib Pyplot** interface:

```
import matplotlib.pyplot as plt
plt.plot(X, Y)
```

3. Now, create a **Pandas DataFrame** with the created values:

```
import pandas as pd
df = pd.DataFrame({'x':X, 'y_col':Y})
```

4. You can plot t using the **Pyplot** interface, but with the data argument:

```
plt.plot('x', 'y_col', data=df)
```

5. And you can also plot it from Pandas DataFrame:

```
df.plot('x', 'y_col')
```

6. What about Seaborn? Let's create the same line plot with Seaborn:

```
import seaborn as sns
sns.lineplot(X, Y)
sns.lineplot('x', 'y_col', data=df)
```

We can see that, in this case, the interface used by Matplotlib and Seaborn is quite similar.

## Activity 4: Line Graphs with the Object-Oriented API and Pandas DataFrames

In this activity, we will create a time series line graph from the „cars.data” file dataset as a first example of plotting using pandas and the object-oriented API. This kind of graph is common in analysis and helps to answer questions such as "is the average horsepower increasing or decreasing with time?"

Now, follow these procedures to plot a graph of average horsepower per year using pandas and while using the object-oriented API.

1.Import the required libraries in the Jupyter notebook and read the dataset from the cars.data dataset repository: `matplotlib, numpy, pandas`

Load data to pandas dataframe from file: `cars.data`

See how your dataframe looks like...

```
df.head()
```

| | 18.0 8 307.0 130.0 3504. 12.0 70 1\t"chevrolet chevelle malibu" |
|---|---|
| **0** | 15.0 8 350.0 165.0 3693. 11... |
| **1** | 18.0 8 318.0 150.0 3436. 11... |
| **2** | 16.0 8 304.0 150.0 3433. 12... |
| **3** | 17.0 8 302.0 140.0 3449. 10... |

...better provide column names

2. Provide the column names to simplify the dataset, as illustrated here:

```
column_names = ['mpg', 'cylinders', 'displacement [cu]',
'horsepower','weight', 'acceleration', 'year', 'origin', 'name']
```

3. Now read the new dataset again with column names and display it.

| | mpg | cylinders | displacement [cu] | horsepower | weight | acceleration | year | origin | name |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130.0 | 3504.0 | 12.0 | 70 | 1 | chevrolet chevelle malibu |
| **1** | 15.0 | 8 | 350.0 | 165.0 | 3693.0 | 11.5 | 70 | 1 | buick skylark 320 |
| **2** | 18.0 | 8 | 318.0 | 150.0 | 3436.0 | 11.0 | 70 | 1 | plymouth satellite |
| **3** | 16.0 | 8 | 304.0 | 150.0 | 3433.0 | 12.0 | 70 | 1 | amc rebel sst |

4.There are some horsepower values set to '?'. Change them to NaN.

```
df.loc[df.horsepower=='?']
```

| | mpg | cylinders | displacement [cu] | horsepower | weight | acceleration | year | origin | name |
|---|---|---|---|---|---|---|---|---|---|
| **32** | 25.0 | 4 | 98.0 | ? | 2046.0 | 19.0 | 71 | 1 | ford pinto |
| **126** | 21.0 | 6 | 200.0 | ? | 2875.0 | 17.0 | 74 | 1 | ford maverick |
| **330** | 40.9 | 4 | 85.0 | ? | 1835.0 | 17.3 | 80 | 2 | renault lecar deluxe |
| **336** | 23.6 | 4 | 140.0 | ? | 2905.0 | 14.3 | 80 | 1 | ford mustang cobra |
| **354** | 34.5 | 4 | 100.0 | ? | 2320.0 | 15.8 | 81 | 2 | renault 18i |

5. Convert horsepower values to numeric

6. Look at the year value. it is given partially without the thousands and hundreds. Convert them to full year e.g: 70 to 1970, 82 to 1982 etc.

Create new column 'full_date' and fill it on the basis of the value of 'year' column:

```
df['full_date'] = pd.to_datetime(df.year, format='%y')
```

Fill 'year' column with year from full date created before:

```
df['year'] = df['full_date'].dt.year
```

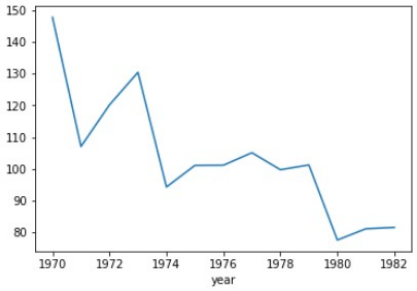
7. Check your changes. Yoy should get:

| | mpg | cylinders | displacement [cu] | horsepower | weight | acceleration | year | origin | name | full_date |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130.0 | 3504.0 | 12.0 | 1970 | 1 | chevrolet chevelle malibu | 1970-01-01 |
| **1** | 15.0 | 8 | 350.0 | 165.0 | 3693.0 | 11.5 | 1970 | 1 | buick skylark 320 | 1970-01-01 |
| **2** | 18.0 | 8 | 318.0 | 150.0 | 3436.0 | 11.0 | 1970 | 1 | plymouth satellite | 1970-01-01 |

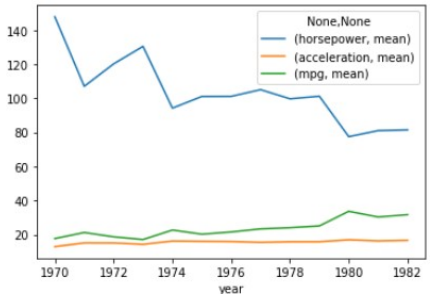
```
mpg                        float64
cylinders                    int64
displacement [cu]          float64
horsepower                 float64
weight                     float64
acceleration               float64
year                         int64
origin                       int64
name                        object
full_date          datetime64[ns]
dtype: object
```

8. Now the dataset is ready to create plots. Plot the average horsepower per year using the following command:

```
df.groupby('year')['horsepower'].mean().plot()
```



9. Try to plot average horsepower, acceleration and mpg per year. Show it on one plot. Use groupby. This should be only one line of code.



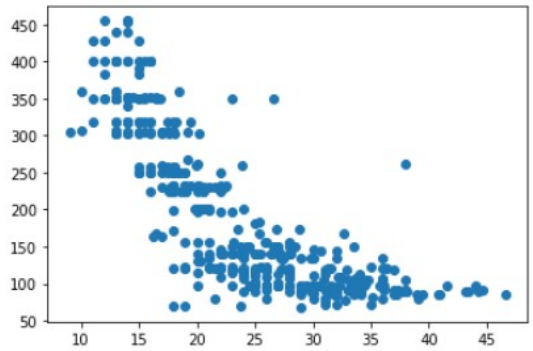

Scatter plots

Scatter plotsc help us  understand the correlation between two variables. They allow the distribution of points to be seen. Samples of creating scatter plots with different libraries:

1. Matplotlib

```
fig, ax = plt.subplots()
ax.scatter(x = df['mpg'], y=df['displacement [cu]'])
```

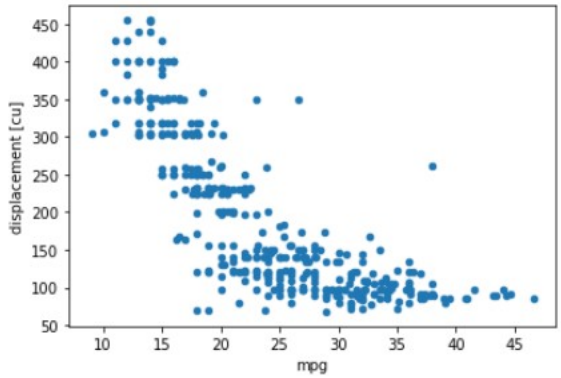
```
<matplotlib.collections.PathCollection at 0x7fdd546f2b80>
```




2. Pandas Data Frame

```
fig, ax = plt.subplots()
df.plot(kind='scatter', x='mpg', y='displacement [cu]', ax =ax)
```

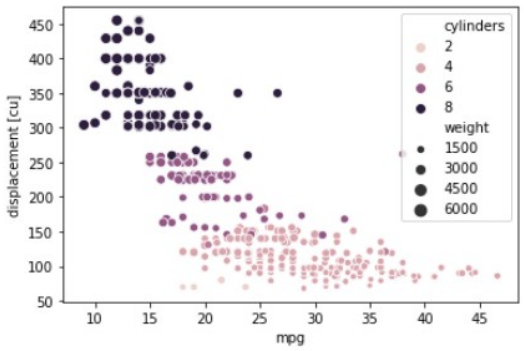
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdd546cfee0>
```



2. Seaborn

```
import seaborn as sns
sns.scatterplot(data=df, x='mpg', y='displacement [cu]',
                hue='cylinders',size='weight')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdd545bc280>
```




## Activity 5: Understanding Relationships of Variables Using Scatter Plots

To continue our data analysis and learn how to plot data, let's look at a situation where a scatter plot can help. For example, let's use a scatter plot to answer the following question:

Is there a relationship between **horsepower** and **weight**?

To answer this question, we need to create a scatter plot with the data from `cars.data` file.

1. Use the `cars.data` dataset, already ingested.

2. Create scatter plots using: Matplotlib, Pandas and Seaborn.

When it’s done: we can identify a roughly linear relationship between horsepower and weight, with some outliers with higher horsepower and lower weight. This is the kind of graph that would help an analyst interpret the data's behavior.

## Exercise 12: Creating a Histogram of Horsepower Distribution

1. Use the `cars.data` dataset, already ingested.

2. Create histogram

```
df.horsepower.plot(kind='hist')
```

3. Identify the horsepower concentration:

```
sns.distplot(df['horsepower'], bins=10)
```

We can see in this graph (with concentraion) that the value distribution is skewed to the left, with more cars with horsepower of between 50 and 100 than greater than 200, for example. This could be quite useful in understanding how some data varies in an analysis

4. Try with other columns e.g. „displacement”.

## Exercise 13: Analyzing the Behavior of the Number of Cylinders and Horsepower Using a Boxplot

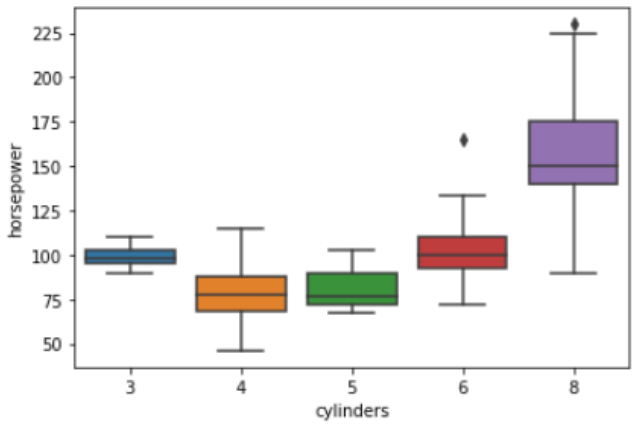
Sometimes we want not only to see the distribution of each variable, but also to see the variation of the variable of interest with respect to another attribute. We would like to know, for instance, how the horsepower varies given the number of cylinders. Let's create a boxplot with Seaborn, comparing the horsepower distribution to the number of cylinders.

1. Use the `cars.data` dataset, already ingested.

2. Create a boxplot using the Seaborn boxplot function:

```
sns.boxplot(data=df, x="cylinders", y="horsepower")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdd5285c880>
```

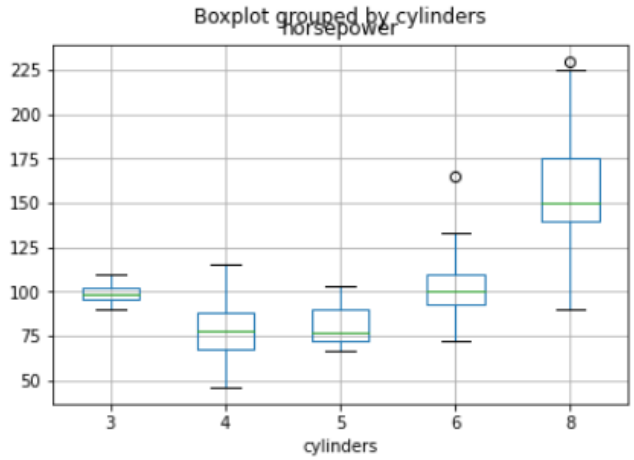



3. Now, just for comparison purposes, create the same boxplot using pandas directly:

df.boxplot(column='horsepower', by='cylinders')

```
df.boxplot(column='horsepower', by='cylinders')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdd52
```



On the analysis side, we can see that the variation range from 3 cylinders is smaller thanfor 8 cylinders for horsepower. We can also see that 6 and 8 cylinders have outliers in the data. As for the plotting, the Seaborn function is more complete, showing different colors automatically for different numbers of cylinders, and including the name of the DataFrame columns as labels in the graph.

## Exercise 14: Configuring a Title and Labels for Axis Objects

1.Set the title, the x-axis label, and the y-axis label by calling the set method:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.set(title="Graph title", xlabel="Label of x axis (units)",
ylabel="Label of y axis (units)")
ax.plot()
```

2. The legends for the plot can be either passed externally, when using only Matplotlib, or can be set on the Pandas plot and plotted with the axes. Use the following command to plot the legends:

```
fig, ax = plt.subplots()
df.groupby('year')['horsepower'].mean().plot(ax=ax,
label='horsepower')
ax.legend()
```

3. For line graphs, the color, the weight, markers, and the style of the lines can be configured with the ls, lw, marker, and color parameters:

```
df.groupby('year')['horsepower'].mean().plot(ls='-.', color='r', lw=3)
```

4. We can also configure the size of the figure. The figsize parameter can be passed to all plot functions as a tuple (x-axis, y-axis) with the size in inches:

```
df.plot(kind='scatter', x='weight', y='horsepower',
figsize=(20,10)
```

## Exercise 15: Working with Matplotlib Style Sheets

1.Let's first print the list of available styles using the following command:

```
import matplotlib.pyplot as plt
print(plt.style.available)
```

2. Now, let's create a scatter plot with the style as classic.

```
plt.style.use(['classic'])
df.plot(kind='scatter', x='weight', y='horsepower')
```

## Activity 6: Exporting a Graph to a File on Disk

Saving our work to a file is a good way to enable sharing the results in different media.

It also helps if we want to keep it for future reference. Let's create a graph and save it to disk.

1. Create any kind of graph using the Matplotlib object-oriented API. For example, a histogram on weight.

2. Export this to a PNG file using the savefig function:

```
fig.savefig('weight_hist.png')
```

## Activity 7: Complete Plot Design

As an analyst, we want to understand whether the average miles per year increased or not, and we want to group by number of cylinders. For example, what is the behavior, in fuel consumption, of a car with three cylinders, over time? Is it higher or lower than a car with four cylinders?

Make a plot like this below, and save it to the file.