At the beginning we import the libraries that are required for the implementation of the code operation. Here we import the Pandas to import and analyze data, NumPy to perform the multi-dimensional operation, and matplotlib to perform graphical plot into the context.

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

The next phase is to load the data into the program to perform the desired operation. Here we use the pandas to load the excel data and when data is successfully loaded we print a statement to get confirmation.

In [5]:
```
url ='https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_score
```

In [6]:
```
data=pd.read_csv(url)
```

Next is to view the data and so we are using the head() function. The head function default view is the top five, but again whatever you want to be in the number of views you can do it as entered, in this case, I have entered six views.

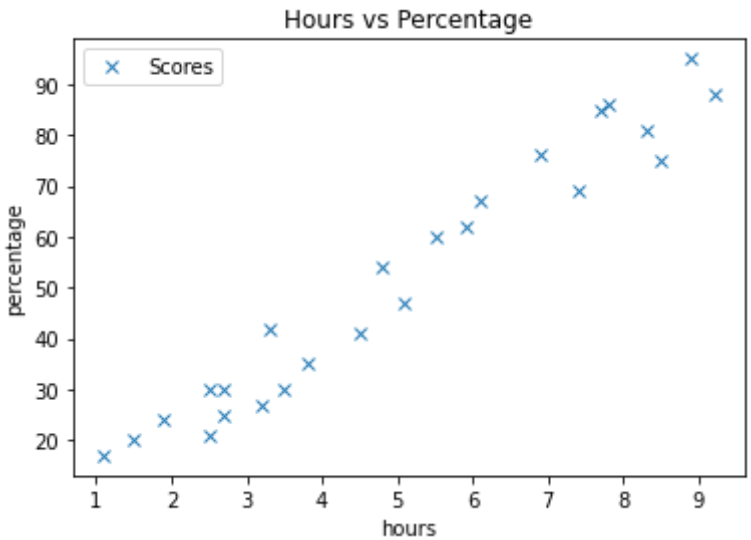
In [7]:
```
data.head()
```

Out[7]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

The next phase is to enter distribution scores and plot them according to the requirement, here we are going to enter the title, x_label, and y_label, and show it according to the desired result.

In [15]:
```
data.plot(x='Hours',y='Scores',style='x')
plt.title('Hours vs Percentage')
plt.xlabel('hours')
plt.ylabel('percentage')
plt.show()
```



The process of dividing the data into attributes and labels is our next task, so we implement the same as below.

In [17]:
```
X = data.iloc[:, :-1].values
y = data.iloc[:, 1].values
```

The split of data into the training and test sets is very important as in this time we will be using Scikit Learn's builtin method of train_test_split(), as below:

In [18]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2, random_state
```

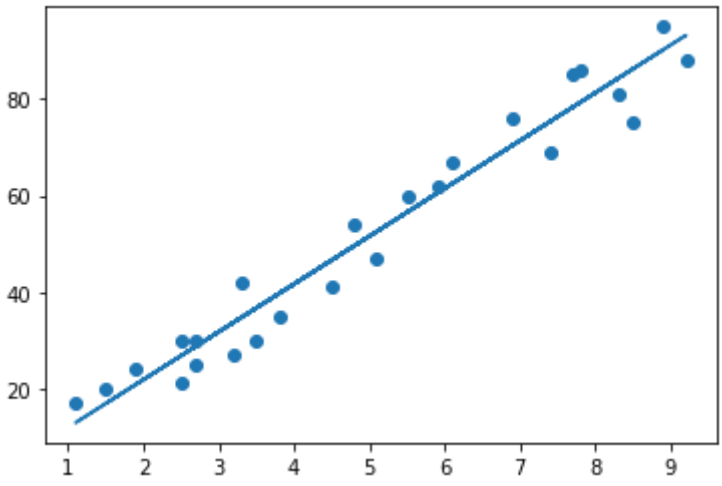
The very next process is to train the algorithm, thus the step include the following:

In [19]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
print("Training ... Completed !.")
```

```
Training ... Completed !.
```

The very next phase is to implement the plotting test data using the previously trained test data:

In [20]:
```
line = regressor.coef_*X+regressor.intercept_
plt.scatter(X, y)
plt.plot(X, line);
plt.show()
```



Predicting the scores for the model is the next important step towards knowing our model, as we proceed as follows,

In [21]:
```
print(X_test)
y_pred = regressor.predict(X_test)
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
```

Comparing the actual versus predicted model to understand our model fitting,

In [22]:
```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[22]:

	Actual	Predicted
0	20	16.884145

	Actual	Predicted
1	27	33.732261
2	69	75.357018
3	30	26.794801
4	62	60.491033

Now it's time to test our model with sample testing hours, so in this case, we take 9.2 hours, i.e, if a student studies for seven hours, approximately how many marks he can get based on the data we received and the model we applied.

In [24]:
```
hours = [[9.2]]
own_pred = regressor.predict(hours)
print("Number of hours = {}".format(hours))
print("Prediction Score = {}".format(own_pred[0]))
```

```
Number of hours = [[9.2]]
Prediction Score = 93.19619966334325
```

In [ ]:
```

```