# Prediction using Supervised ML

***Import necessary libraries***

In [12]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression

%matplotlib inline
```

***Load the Dataset***

In [6]:
```
df = pd.read_csv('Dataset.csv')
```

In [7]:
```
df.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 |

***Information about dataset***

In [8]:
```
df.describe()
```

Out[8]:

| | Hours | Scores |
|---|---|---|
| **count** | 25.000000 | 25.000000 |
| **mean** | 5.012000 | 51.480000 |
| **std** | 2.525094 | 25.286887 |
| **min** | 1.100000 | 17.000000 |
| **25%** | 2.700000 | 30.000000 |
| **50%** | 4.800000 | 47.000000 |
| **75%** | 7.400000 | 75.000000 |
| **max** | 9.200000 | 95.000000 |

***Training the dataset***

In [18]:
```
#Differentiating between given train data and outcome data
X = df['Hours'].values.reshape(-1,1)
y = df['Scores'].values.reshape(-1,1)
```

In [19]:
```
#Split the data into 70% train and 30% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

***Model - Linear Regression***

In [20]:
```
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
```

In [23]:
```
y_pred
```

Out[23]:
```
array([[46.0069352 ],
       [61.88185566],
       [12.27272925],
       [26.16328464],
       [92.63951403],
       [77.75677611],
       [20.21018947],
       [16.24145936]])
```

***Accuracy scores of predicted value***

In [21]:
```
print('Mean Squarred Error: ', mean_squared_error(y_test,y_pred))
print('Root Mean Squarred Error: ', np.sqrt(mean_squared_error(y_test,y_pred)))
print('R2 Score: ' , r2_score(y_test,y_pred) * 100)
```

```
Mean Squarred Error:  25.343759142910855
Root Mean Squarred Error:  5.034258549469907
R2 Score:  96.8190453508535
```

***Creating a new dataframe with the actual and predicted values***

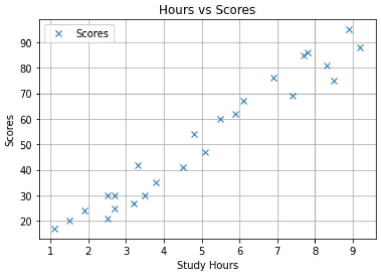
In [22]:
```
df_new = pd.DataFrame({'Actual':y_test.flatten(), 'Predicted':y_pred.flatten()})
df_new.head()
```

Out[22]:

| | Actual | Predicted |
|---|---|---|
| **0** | 41 | 46.006935 |
| **1** | 67 | 61.881856 |
| **2** | 17 | 12.272729 |
| **3** | 21 | 26.163285 |
| **4** | 88 | 92.639514 |

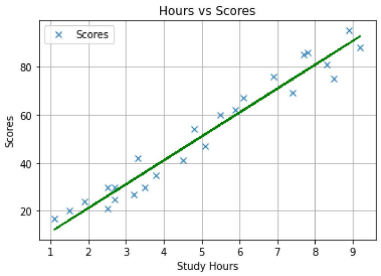
***Visualizing the data***

In [38]:
```
#Actual Data Points
df.plot(x = 'Hours', y = 'Scores', style = 'x')
plt.title('Hours vs Scores')
plt.xlabel('Study Hours')
plt.ylabel('Scores')
plt.grid()
plt.show()
```




In [51]:
```
#Plotting the linear regression line
line = lr.coef_*X + lr.intercept_

df.plot(x = 'Hours', y = 'Scores', style = 'x')
plt.plot(X, line, color = 'green');
plt.title('Hours vs Scores')
plt.xlabel('Study Hours')
plt.ylabel('Scores')
plt.grid()
plt.show()
```




***Predicted score***

In [74]:
```
lrCoef = list(lr.coef_)
lrIntr = list(lr.intercept_)
regLine = 'scores = ' + str(lrCoef[0][0]) + ' * hours + ' + str(lrIntr[0])
print(f'The regression line for the above scenario is: {regLine}.')
```

```
The regression line for the above scenario is: scores = 9.92182528203645 * hours + 1.3587214347700183.
```

In [73]:
```
givenHours = 9.25
```

In [75]:
```
predScore = lrCoef[0][0] * givenHours + lrIntr[0]
print(f'The predicted score for study of {givenHours} hours is: {predScore}.')
```

```
The predicted score for study of 9.25 hours is: 93.13560529360717.
```

---

Done By:
Varun Bhattacharya