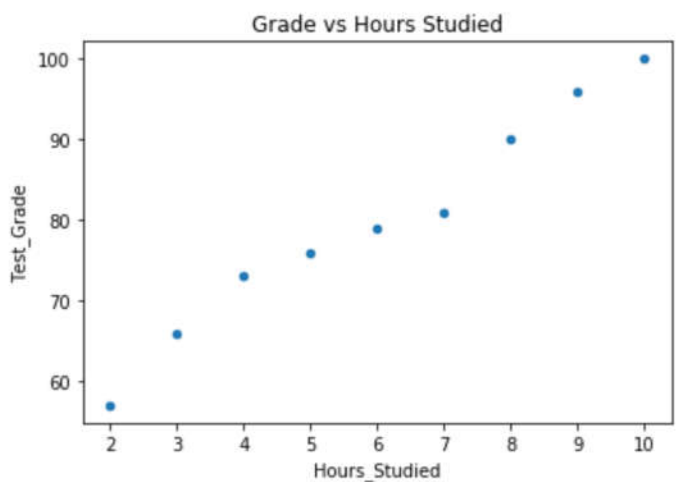
In [21]:
```python
# Linear Realtion between Dependent and Independent Variable

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

# importing linear regression function
import sklearn.linear_model as lm

# function to calculate r-squared, MAE, RMSE
from sklearn.metrics import r2_score , mean_absolute_error, mean_squared_error

%matplotlib inline


# Load data
df = pd.read_csv('Grade_Set_1.csv')
#print (df)

# Simple scatter plot
df.plot(kind='scatter', x='Hours_Studied', y='Test_Grade', title='Grade vs Hours St
udied')
plt.show()
# check the correlation between variables
print("Correlation Matrix: ")
print(df.corr())

df.head()
```



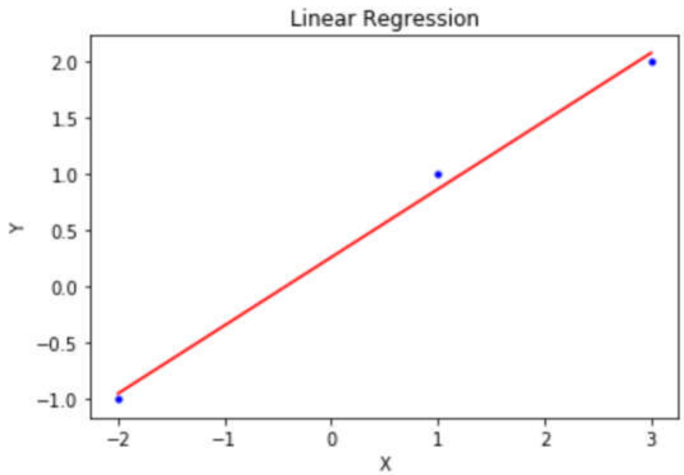

```
Correlation Matrix:
               Hours_Studied  Test_Grade
Hours_Studied       1.000000    0.987797
Test_Grade          0.987797    1.000000
```

Out[21]:

	Hours_Studied	Test_Grade
0	2	57
1	3	66
2	4	73
3	5	76
4	6	79

In [22]:
```
# imports
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

x = np.array([-2, 1, 3]).reshape(3,1)
y = np.array([-1, 1, 2]).reshape(3,1)


# sckit-learn implementation

# Model initialization
regression_model = LinearRegression()
# Fit the data(train the model)
regression_model.fit(x, y)
# Predict
y_predicted = regression_model.predict(x)

# model evaluation
rmse = mean_squared_error(y, y_predicted)


# printing values
print('Slope:' ,regression_model.coef_)
print('Intercept:', regression_model.intercept_)
print('mean squared error: ', rmse)


# plotting values


# Visualising the Linear Regression results

# data points
plt.scatter(x, y, color='blue', s=10)

plt.title('Linear Regression')


# predicted values
plt.plot(x, y_predicted, color='r')
plt.xlabel("X")
plt.ylabel("Y")

plt.show()


# Predicting a new result with Linear Regression
print('Using sklearn', regression_model.predict([[110.0]]))

print('Manual calculation', 0.263115789 + 110.0*0.60526316)
```

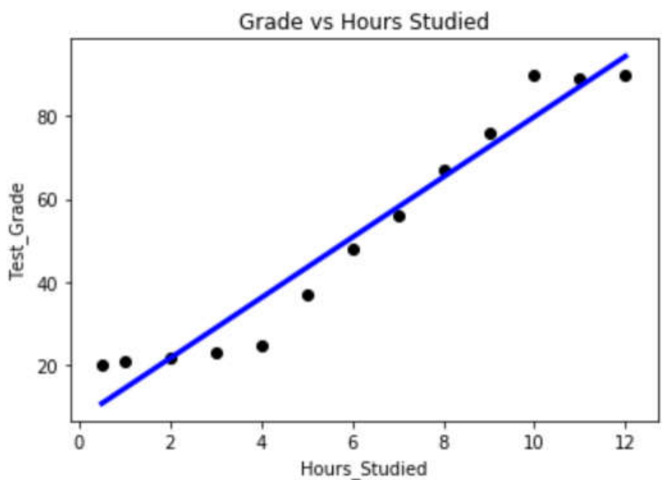
```
Slope: [[0.60526316]]
Intercept: [0.26315789]
mean squared error:  0.008771929824561398
```




```
Using sklearn [[66.84210526]]
Manual calculation 66.842063389
```

In [23]:
```
# Load data
df = pd.read_csv('Grade_Set_2.csv')
print(df)

# Simple scatter plot
df.plot(kind='scatter', x='Hours_Studied', y='Test_Grade', title='Grade vs Hours St
udied')

# check the correlation between variables
print("Correlation Matrix: ")
df.corr()


# Create linear regression object
lr = lm.LinearRegression()

x= df.Hours_Studied[:, np.newaxis]           # independent variable
y= df.Test_Grade                             # dependent variable

# Train the model using the training sets
lr.fit(x, y)
print ("Intercept: ", lr.intercept_)
print ("Coefficient: ", lr.coef_)

# plotting fitted line
plt.scatter(x, y,  color='black')
plt.plot(x, lr.predict(x), color='blue', linewidth=3)
plt.title('Grade vs Hours Studied')
plt.ylabel('Test_Grade')
plt.xlabel('Hours_Studied')

print ("R Squared: ", r2_score(y, lr.predict(x)))
```

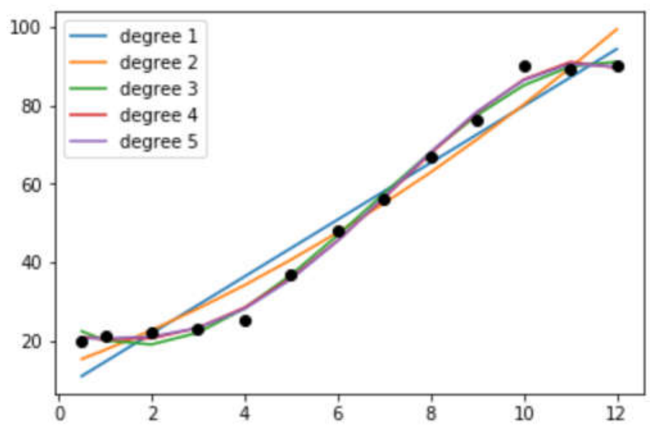
```
    Hours_Studied  Test_Grade
0             0.5          20
1             1.0          21
2             2.0          22
3             3.0          23
4             4.0          25
5             5.0          37
6             6.0          48
7             7.0          56
8             8.0          67
9             9.0          76
10           10.0          90
11           11.0          89
12           12.0          90
Correlation Matrix:
Intercept:  7.27106067219556
Coefficient:  [7.25447403]
R Squared:  0.9503677766997879
```



In [24]:
```
lr = lm.LinearRegression()

x= df.Hours_Studied         # independent variable
y= df.Test_Grade            # dependent variable

# NumPy's vander function will return powers of the input vector
for deg in [1, 2, 3, 4, 5]:
    lr.fit(np.vander(x, deg + 1), y);
    y_lr = lr.predict(np.vander(x, deg + 1))
    plt.plot(x, y_lr, label='degree ' + str(deg));
    plt.legend(loc=2);
    print ("R-squared for degree " + str(deg) + " = ",  r2_score(y, y_lr))
plt.plot(x, y, 'ok')
```

```
R-squared for degree 1 =  0.9503677766997879
R-squared for degree 2 =  0.9608726568678714
R-squared for degree 3 =  0.9938323120374665
R-squared for degree 4 =  0.9955000184096712
R-squared for degree 5 =  0.9956204913897356
```

Out[24]: [<matplotlib.lines.Line2D at 0x1b30c00940>]



In [25]:

```
import warnings
warnings.filterwarnings('ignore')

import statsmodels.api as sm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn import metrics
from statsmodels.stats.outliers_influence import variance_inflation_factor, OLSInfl
uence
from sklearn.model_selection import train_test_split

import warnings
warnings.filterwarnings('ignore')

import statsmodels.api as sm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn import metrics
from statsmodels.stats.outliers_influence import variance_inflation_factor, OLSInfl
uence
from sklearn.model_selection import train_test_split

%matplotlib inline

from sklearn import linear_model

# Load data
df = pd.read_csv('Grade_Set_2.csv')
df.columns = ['x','y']

for i in range(2,50):                # power of 1 is already there
    colname = 'x_%d'%i                # new var will be x_power
    df[colname] = df['x']**i

independent_variables = list(df.columns)
independent_variables.remove('y')

X= df[independent_variables]         # independent variable
y= df.y                              # dependent variable

# split data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=.80, random_st
ate=1)

# Ridge regression
lr = sm.OLS(y_train, X_train).fit()
y_train_pred = lr.predict(X_train)
y_test_pred = lr.predict(X_test)

print ("Train MAE: ", metrics.mean_absolute_error(y_train, y_train_pred))
print ("Train RMSE: ", np.sqrt(metrics.mean_squared_error(y_train, y_train_pred)))

print ("Test MAE: ", metrics.mean_absolute_error(y_test, y_test_pred))
print ("Test RMSE: ", np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

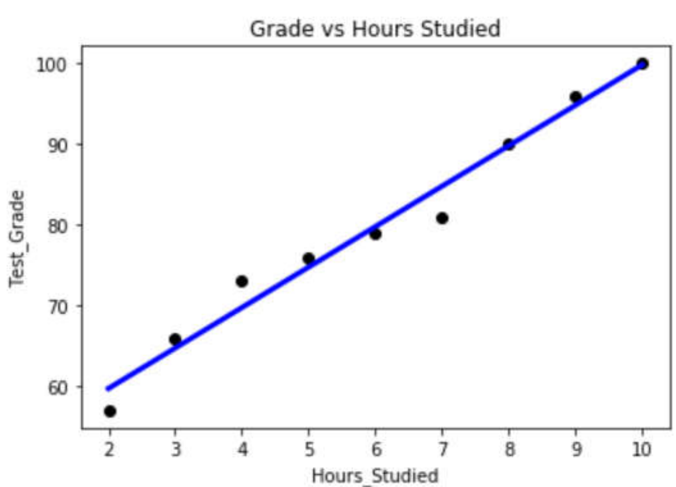
```
Train MAE:  18.20045122478613
Train RMSE:  27.491829932639963
Test MAE:  23.33333333328628
Test RMSE:  23.366642891045505
```

In [26]:
```
import warnings
warnings.filterwarnings('ignore')

import statsmodels.api as sm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn import metrics
from statsmodels.stats.outliers_influence import variance_inflation_factor, OLSInfl
uence
from sklearn.model_selection import train_test_split

%matplotlib inline

from sklearn import linear_model

# Load data
df = pd.read_csv('Grade_Set_2.csv')
df.columns = ['x','y']

for i in range(2,50):              # power of 1 is already there
    colname = 'x_%d'%i              # new var will be x_power
    df[colname] = df['x']**i

independent_variables = list(df.columns)
independent_variables.remove('y')

X= df[independent_variables]       # independent variable
y= df.y                            # dependent variable

# split data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=.80, random_st
ate=1)

# Ridge regression
lr = linear_model.Ridge(alpha=0.001)
lr.fit(X_train, y_train)
y_train_pred = lr.predict(X_train)
y_test_pred = lr.predict(X_test)

print("------ Ridge Regression ------")
print ("Train MAE: ", metrics.mean_absolute_error(y_train, y_train_pred))
print ("Train RMSE: ", np.sqrt(metrics.mean_squared_error(y_train, y_train_pred)))

print ("Test MAE: ", metrics.mean_absolute_error(y_test, y_test_pred))
print ("Test RMSE: ", np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))


# LASSO regression
lr = linear_model.Lasso(alpha=0.001)
lr.fit(X_train, y_train)
y_train_pred = lr.predict(X_train)
y_test_pred = lr.predict(X_test)

print("----- LASSO Regression -----")
print ("Train MAE: ", metrics.mean_absolute_error(y_train, y_train_pred))
print ("Train RMSE: ", np.sqrt(metrics.mean_squared_error(y_train, y_train_pred)))

print ("Test MAE: ", metrics.mean_absolute_error(y_test, y_test_pred))
print ("Test RMSE: ", np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
------ Ridge Regression ------
Train MAE:  12.775326528414379
Train RMSE:  16.72063936357992
Test MAE:  22.397943556789926
Test RMSE:  22.432642089791898
----- LASSO Regression -----
Train MAE:  0.8423742988874519
Train RMSE:  1.219129185560593
Test MAE:  4.32364759404346
Test RMSE:  4.872324349696696
------ Elastic Net Regression ------
Train MAE:  0.8822280533191137
Train RMSE:  1.2664062087497077
Test MAE:  3.5541770995212745
Test RMSE:  4.138557504736036
```

In [27]:
```
# Demo for outlier

# Load data
df = pd.read_csv('Grade_Set_1.csv')

#df.loc[9] = np.array([5, 100])

x= df.Hours_Studied[:, np.newaxis] # independent variable
y= df.Test_Grade.values            # dependent variable

# Train the model using the training sets
lr.fit(x, y)
print ("Intercept: ", lr.intercept_)
print ("Coefficient: ", lr.coef_)


# plotting fitted line
plt.scatter(x, y,  color='black')
plt.plot(x, lr.predict(x), color='blue', linewidth=3)

residuals = lr.predict(x) - y

plt.title('Grade vs Hours Studied')
plt.ylabel('Test_Grade')
plt.xlabel('Hours_Studied')

# add predict value to the data frame
df['Test_Grade_Pred'] = lr.predict(x)

# Using built-in function
print ("R Squared : ", r2_score(df.Test_Grade,  df.Test_Grade_Pred))
print ("Mean Absolute Error: ", mean_absolute_error(df.Test_Grade, df.Test_Grade_Pr
ed))
print ("Root Mean Squared Error: ", np.sqrt(mean_squared_error(df.Test_Grade, df.Te
st_Grade_Pred)))

print(residuals)
```

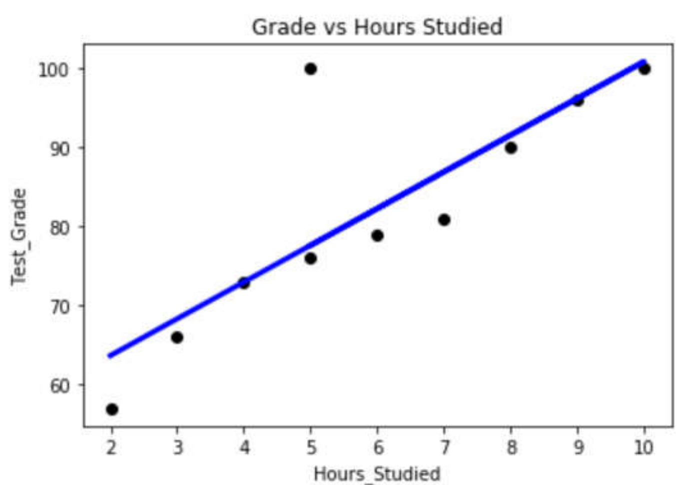
```
Intercept:  49.67867777777778
Coefficient:  [5.01651667]
R Squared :  0.9757431065371913
Mean Absolute Error:  1.6186185185185167
Root Mean Squared Error:  2.042299636273036
[ 2.71171111 -1.27177222 -3.25525556 -1.23873889  0.77777778  3.79429444
 -0.18918889 -1.17267222 -0.15615556]
```



In [ ]:

In [28]:
```
# Load data
df = pd.read_csv('Grade_Set_1.csv')

df.loc[9] = np.array([5, 100])

x= df.Hours_Studied[:, np.newaxis] # independent variable
y= df.Test_Grade.values            # dependent variable

# Train the model using the training sets
lr.fit(x, y)
print ("Intercept: ", lr.intercept_)
print ("Coefficient: ", lr.coef_)



# plotting fitted line
plt.scatter(x, y,  color='black')
plt.plot(x, lr.predict(x), color='blue', linewidth=3)
plt.title('Grade vs Hours Studied')
plt.ylabel('Test_Grade')
plt.xlabel('Hours_Studied')

# add predict value to the data frame
df['Test_Grade_Pred'] = lr.predict(x)

# Using built-in function
print ("R Squared : ", r2_score(df.Test_Grade,  df.Test_Grade_Pred))
print ("Mean Absolute Error: ", mean_absolute_error(df.Test_Grade, df.Test_Grade_Pr
ed))
print ("Root Mean Squared Error: ", np.sqrt(mean_squared_error(df.Test_Grade, df.Te
st_Grade_Pred)))
```

```
Intercept:  54.40326765188834
Coefficient:  [4.64351396]
R Squared :  0.685546138163505
Mean Absolute Error:  4.480367816091954
Root Mean Squared Error:  7.761235840599034
```



In [ ]: