Polynomial Regression

Importing the Libraries

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [2]:
```
dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values
```

Splitting the dataset into the Training set and Test set

Since we need the entire dataset for training (for this particular problem), we will not split the data into training and test sets

In [ ]:
```
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 = 0)
```

Fitting Linear Regression to the dataset

In [3]:
```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
```

Out[3]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Fitting Polynomial Regression to the dataset

In [4]:
```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X)
poly_reg.fit(X_poly, y)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
```

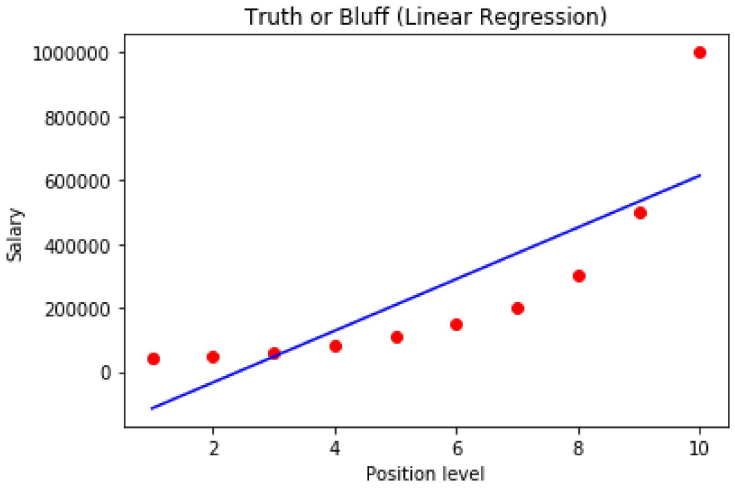
Out[4]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

As we can see above, the higher degrees of the expression leads to better fit. This however, can also culminate into a over-fitted model.

Please ensure that the regressor is not overfitted to suit the training data

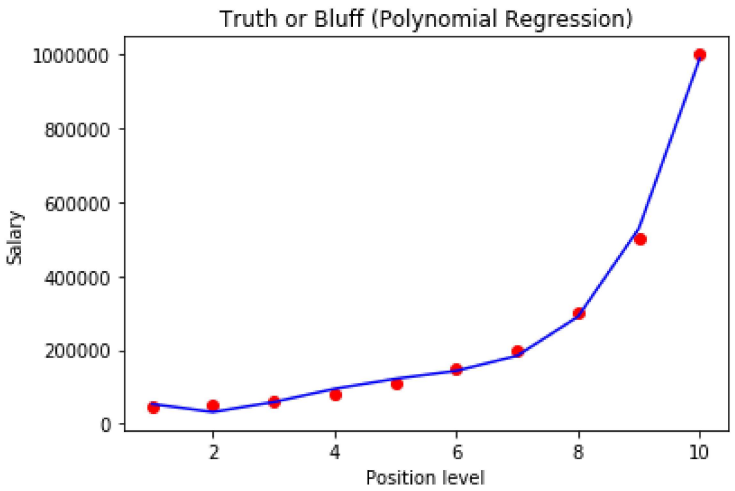
Visualising the Linear Regression results

In [5]:
```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg.predict(X), color = 'blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



Visualising the Polynomial Regression results

In [6]:
```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



Predicting a new result with Linear Regression

In [7]:
```
lin_reg.predict(6.5)
```

Out[7]:
```
array([330378.78787879])
```

Predicting a new result with Polynomial Regression

In [8]:
```
lin_reg_2.predict(poly_reg.fit_transform(6.5))
```

Out[8]:
```
array([158862.4526516])
```