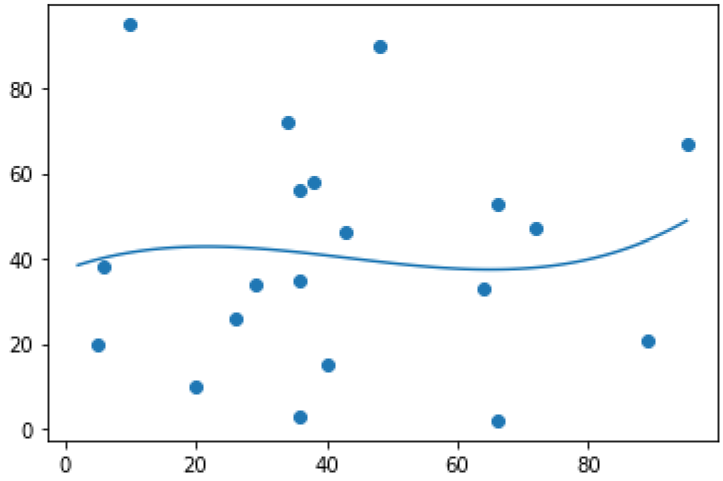
Muhammad Arham Adeel

Polynomial Regression

Practice and task given in the video.

In [1]:

```
#bad fit

import numpy
import matplotlib.pyplot as plt

x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]

mymodel = numpy.poly1d(numpy.polyfit(x,y,3))

myline = numpy.linspace(2,95,100)

plt.scatter(x,y)
plt.plot(myline,mymodel(myline))
plt.show()
```



R-Square for Bad fit

In [2]:

```
from sklearn.metrics import r2_score

x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]

model = numpy.poly1d(numpy.polyfit(x,y,3))

print(r2_score(y,model(x)))
```

```
0.009952707566680652
```

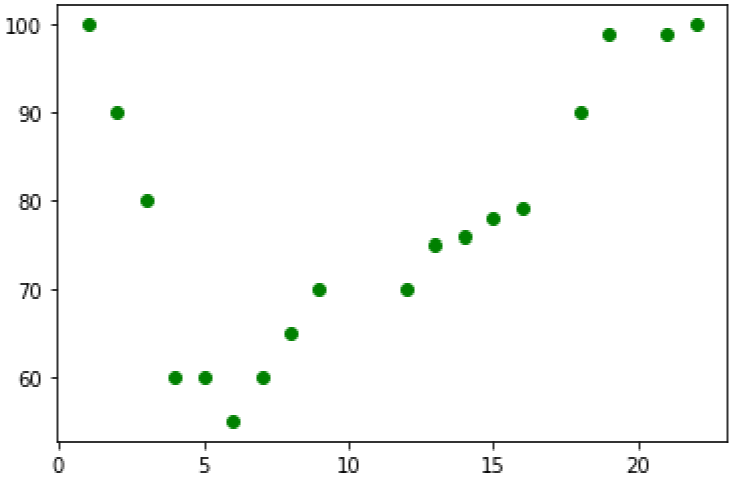
In [3]:

```
#Step-1 Data

x = [1,2,3,4,5,6,7,8,9,12,13,14,15,16,18,19,21,22]
```

```
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]
# Len(x)
plt.scatter(x,y, color="green")
```

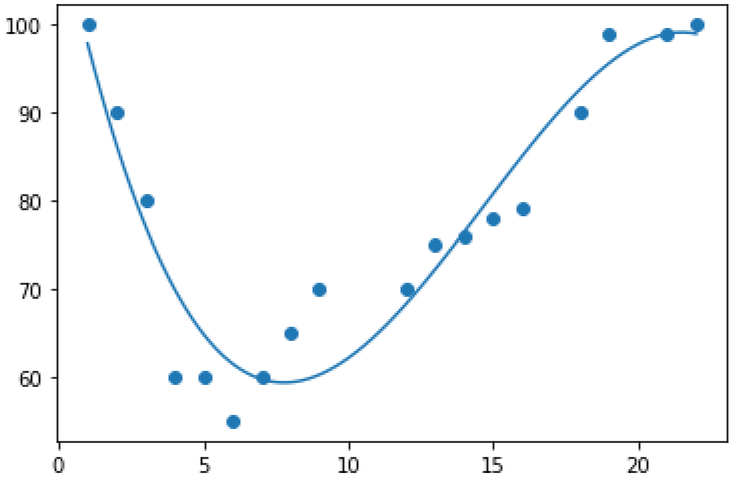
Out[3]:

```
<matplotlib.collections.PathCollection at 0x1f162b36e80>
```



In [4]:

```
mymodel = numpy.poly1d(numpy.polyfit(x,y,3))

myline = numpy.linspace(1,22,100)

plt.scatter(x,y)
plt.plot(myline,mymodel(myline))
plt.show()
```



In [5]:

```
# Stpe-3: R_Squared

from sklearn.metrics import r2_score

x = [1,2,3,4,5,6,7,8,9,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

model = numpy.poly1d(numpy.polyfit(x,y,3))

print(r2_score(y,model(x)))
```

```
0.9003852716313046
```

In [6]:

```
from sklearn.metrics import r2_score

x = [1,2,3,4,5,6,7,8,9,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

model = numpy.poly1d(numpy.polyfit(x,y,3))

speed = mymodel(18)
speed
```

Out[6]:

```
92.66670110727381
```

How do you determine the degree of a polynomial fit?

We can choose the degree of polynomial based on the relationship between target and predictor. The 1-degree polynomial is a simple linear regression; therefore, the value of degree must be greater than 1. With the increasing degree of the polynomial, the complexity of the model also increases.

Hands on Example

In [7]:

```
# importing libraries

import numpy
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv("https://s3.us-west-2.amazonaws.com/public.gamelab.fun/dataset/position_salaries.
df.head()
```

Out[7]:

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

In [8]:

```
x= df.iloc[:,1:2].values
y = df.iloc[:,2].values
```

In [9]:

```
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) #here i can s
```

In [10]:

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(x,y)
```

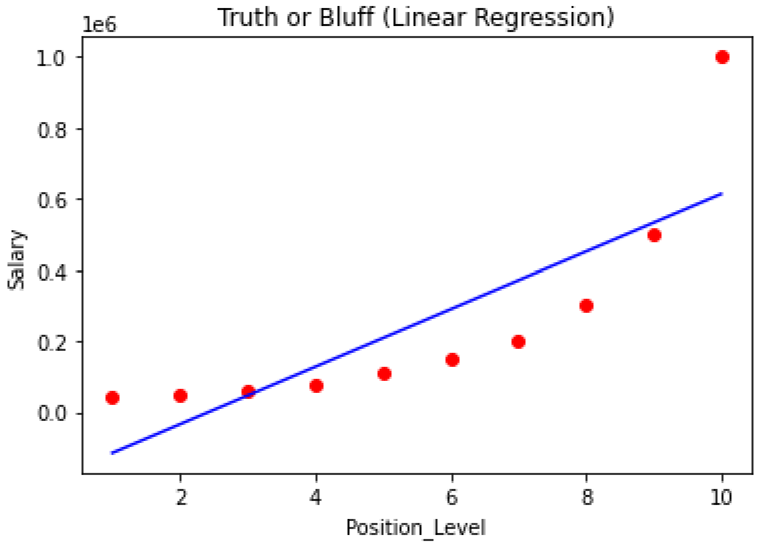
Out[10]:

```
LinearRegression()
```

Visualize the Linear Regression Result

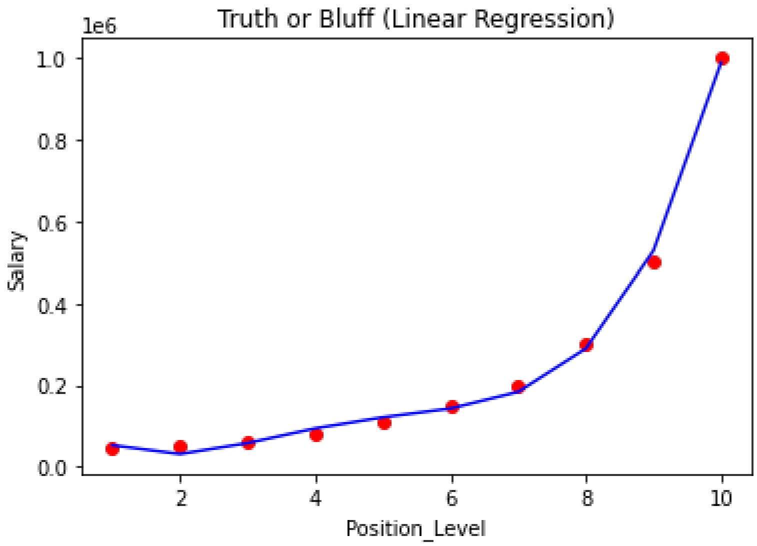
In [47]:

```
def viz_linear():
    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()
    return
viz_linear()
```



In [53]:

```
# Fitting polynomial regression to the dateset

from sklearn.preprocessing import PolynomialFeatures

poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(x)

pol_reg = LinearRegression()
pol_reg.fit(X_poly,y)

# Visulaize the polynomial Results

def viz_polynomial():
    plt.scatter(x,y,color="red")
    plt.plot(x,pol_reg.predict(poly_reg.fit_transform(x)), color="blue")
    plt.title("Truth or Bluff (Linear Regression)")
    plt.xlabel("Position_Level")
    plt.ylabel("Salary")
    plt.show()
    return
viz_polynomial()
```



In [49]:

```
# predicting a new result with linerRegresion


pred_linear = lin_reg.predict([[11]])
pred_linear
```

Out[49]:

```
array([694333.33333333])
```

In [54]:

```
#predicting a new result with polynomial regression

pred_polynomial = pol_reg.predict(poly_reg.fit_transform([[11]]))
```

In [55]:

```
print("Linear Regression Result",pred_linear)
```

```
Linear Regression Result [694333.33333333]
```

In [56]:

```
print("Polynomial Regression Result = ",pred_polynomial)
```

```
Polynomial Regression Result =  [1780833.33333322]
```

In [57]:

```
print("The difference is = ",pred_linear - pred_polynomial)
```

```
The difference is =  [-1086499.99999989]
```

In [ ]: