Fitting different models

In [26]:

```
import numpy as np
import matplotlib.pyplot as plt
np.set_printoptions(precision=3)
#generate the data
x = np.arange(100)
y = 150 + 3*x + 0.03*x**2 + 5*np.random.randn(len(x))

```

line fit

In [27]:

```

#Design the design matrices
M1 = np.vstack( (np.ones_like(x), x) ).T
#print(M1)
# Solve the equations
p1 = np.linalg.lstsq(M1, y)
#estimated parameters
print('The coefficients from the linear fit: {0}'.format(p1[0]))
#line fit
y_pred1=  p1[0][1] * x  + p1[0][0]
```

```
The coefficients from the linear fit: [102.542   5.948]
```

```
C:\Users\jyostna\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: FutureWa
rning: `rcond` parameter will change to the default of machine precision times
``max(M, N)`` where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=No
ne`, to keep using the old, explicitly pass `rcond=-1`.
```

quadratic fit

In [28]:

```
#Design the design matrices
M2 = np.vstack( (np.ones_like(x), x, x**2) ).T
#print(M1)
# Solve the equations
p2 = np.linalg.lstsq(M2, y)
#estimated parameters
print('The coefficients from the quadratic fit: {0}'.format(p2[0]))


#quadratic  fit
y_pred2= p2[0][0] + p2[0][1] * x + p2[0][2] * x *x  #quadratic fit

```

```
The coefficients from the quadratic fit: [1.522e+02 2.907e+00 3.072e-02]

C:\Users\jyostna\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: FutureWa
rning: `rcond` parameter will change to the default of machine precision times
``max(M, N)`` where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=No
ne`, to keep using the old, explicitly pass `rcond=-1`.
  """
```

Cubic fit

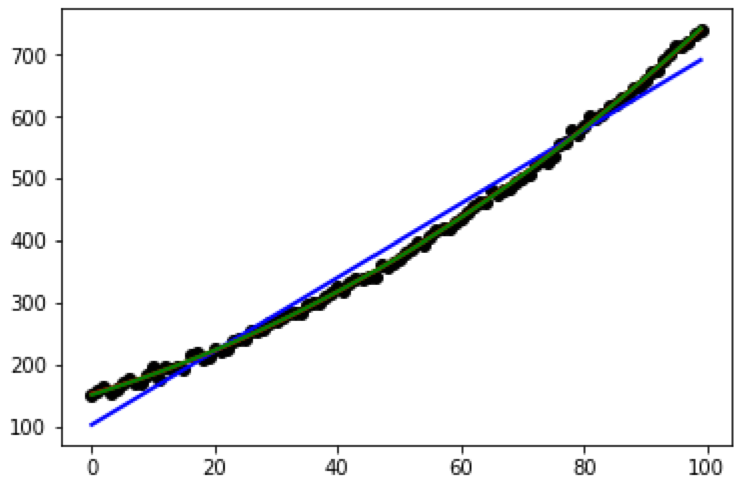
In [29]:

```
#Design the design matrices
M3 = np.vstack( (np.ones_like(x), x, x**2, x**3) ).T
# Solve the equations
p3 = np.linalg.lstsq(M3, y)
#estimated parameters
print('The coefficients from the quadratic fit: {0}'.format(p3[0]))
#cubic  fit
y_pred3= p3[0][0] + p3[0][1] * x + p3[0][2] * x *x + p3[0][3] * x * x * x   #c


```

```
The coefficients from the quadratic fit: [1.511e+02 3.048e+00 2.714e-02 2.410e
-05]

C:\Users\jyostna\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWa
rning: `rcond` parameter will change to the default of machine precision times
``max(M, N)`` where M and N are the input matrix dimensions.
To use the future default and silence this warning we advise to pass `rcond=No
ne`, to keep using the old, explicitly pass `rcond=-1`.
  after removing the cwd from sys.path.
```

In [30]:
```
plt.scatter(x, y,  color='black')
plt.plot(x, y_pred1, color='blue', linewidth=2)

plt.plot(x, y_pred2, color='red' , linewidth=2 )

plt.plot(x, y_pred3, color='green' , linewidth=2 )
```

Out[30]:
```
[<matplotlib.lines.Line2D at 0x2851db92f28>]
```



Fit models USING STATS MODELS

In [36]:
```
import statsmodels.api as sm
Res1 = sm.OLS(y, M1).fit()
Res2 = sm.OLS(y, M2).fit()
Res3 = sm.OLS(y, M3).fit()
print(Res1.summary())
```

```
Model:                            OLS   Adj. R-squared:                  0.9
82
Method:                 Least Squares   F-statistic:                     526
4.
Date:                Tue, 16 Apr 2019   Prob (F-statistic):           5.52e-
87
Time:                        08:59:13   Log-Likelihood:                -457.
29
No. Observations:                 100   AIC:                             91
8.6
Df Residuals:                      98   BIC:                             92
3.8
Df Model:                           1
Covariance Type:            nonrobust
============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------
```

the AIC-value,

The Akaike Information Criterion, which can be used to assess the quality of the model: the lower the AIC value, the better the model. We see that the quadratic model has the lowest AIC value and therefore is the best model: it provides the same quality of fit as the cubic model, but uses fewer parameters to achieve that quality.

In [30]:
```
print('The AIC-value is \n {0:4.1f} for the linear fit,\n {1:4.1f} for the qua
Res3.aic))
```

```
The AIC-value is
 909.5 for the linear fit,
 647.4 for the quadratic fit, and
 648.8 for the cubic fit
```

The degrees of freedom (Df)

df of the model are the number of predictors, or explanatory, variables.

The Df of the residuals is the number of observations minus the degrees of freedom of the model, minus one (for the offset).

In [40]:
```
print( Res1.df_model, Res2.df_model, Res3.df_model)
```

```
1.0 2.0 3.0
```

R-Squared

R2 value indicates the proportion of variation in the y-variable that is due to variation in the x-variables.

For simple linear regression, the R2 value is the square of the sample correlation r_xy.

For multiple linear regression with intercept (which includes simple linear regression), the R2 value is defined as

R^2 = SS_mod / SS_tot

R^2 is close to 1 means the model is better

In [41]:
```
print( Res1.rsquared, Res2.rsquared, Res3.rsquared)
```

```
0.9835452518838422 0.9988266561367175 0.9988334443953123
```

Definitions for Regression with Intercept

The Sum-of Squares variables SSxx

n is the number of observations,

and k is the number of regression parameters.

DF_mod = k - 1 is the (Corrected) Model Degrees of Freedom.

DF_res = n - k is the Residuals Degrees of Freedom DF_tot = n - 1 is the (Corrected) Total Degrees of Freedom.

For multiple regression models with intercept, DF_mod + DF_res = DF_tot.

MS_mod = SS_mod/DFmod : Model Mean of Squares

MS_res = SS_res/DF_res : Residuals Mean of Squares.

MS_tot = SS_tot/DF_tot : Total Mean of Squares, which is the sample variance of the y-variable.

In [ ]:

```
1
```