1. Construct the n x 4 design matrix, D.

Solution. The design matrix D, along the system we will be solving is as follows:

$$Da = y$$
$$\begin{bmatrix} 1 & t_i & \sin(2\pi t_i) & \cos(2\pi t_i) \end{bmatrix} a_i = y_i$$
$$\begin{bmatrix} 1 & 0 & \sin(2\pi(0)) & \cos(2\pi(0)) \\ 1 & .028 & \sin(2\pi(.028)) & \cos(2\pi(.028)) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 3.57 & \sin(2\pi(3.57)) & \cos(2\pi(3.57)) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{99} \end{bmatrix}$$

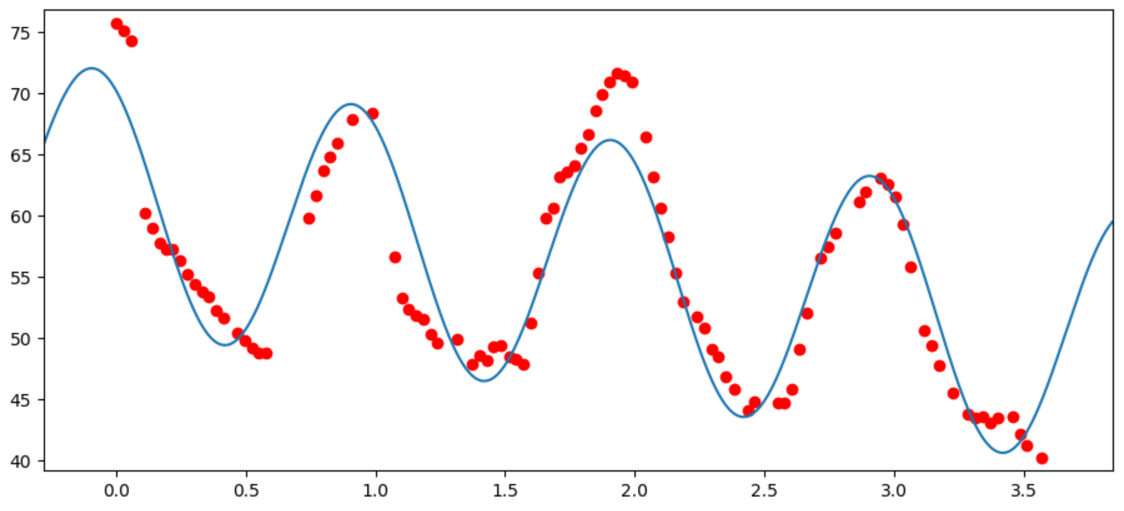
2. To solve this overdetermined system, we will let $A = D^T D$ and then solve the system, $Aa = D^T y$. The values of A, as solved by our program are:

$$A = D^T D = \begin{bmatrix} 100 & 176.697 & 11.2464 & -2.19107 \\ 176.697 & 418.5 & 14.2034 & -9.11544 \\ 11.2464 & 14.2034 & 49.6226 & 0.499124 \\ -2.19107 & -9.11544 & 0.499124 & 50.3774 \end{bmatrix}$$

3. To actually solve our system $Aa = D^T y$, we use the linear algebra solver in the numpy package. Our results a are

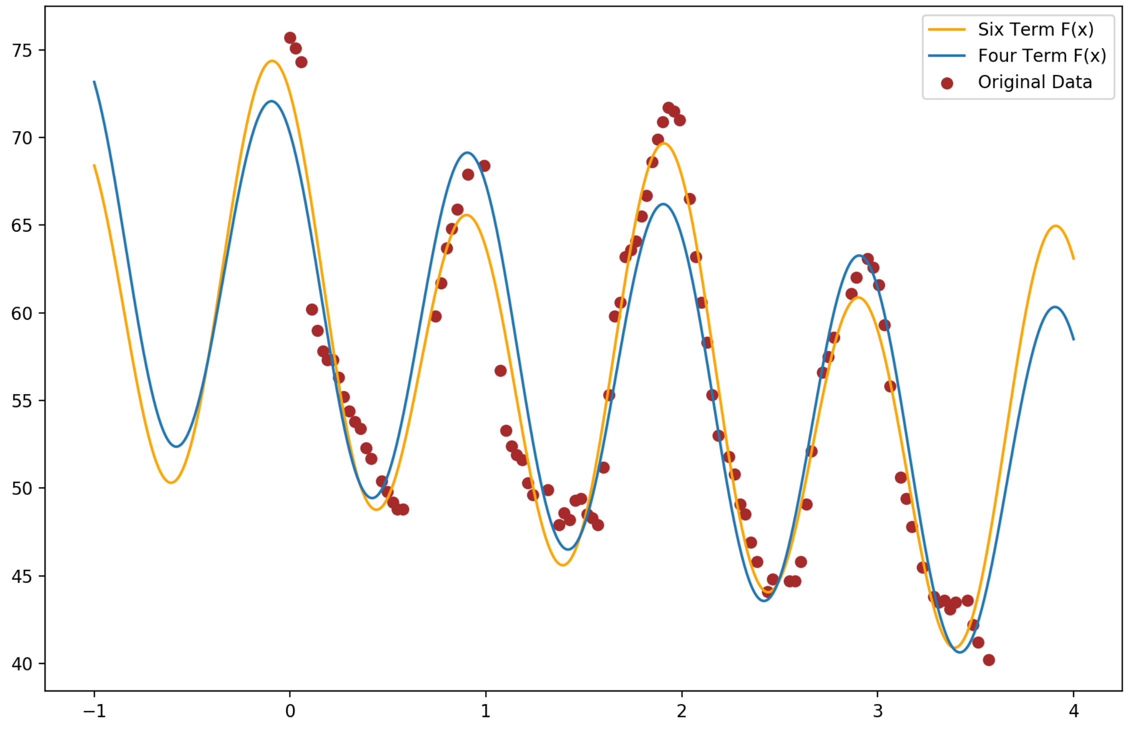
$$a = \begin{bmatrix} 62.222 \\ -2.93388 \\ -5.54473 \\ 9.00424 \end{bmatrix}$$

Here is the graph of our approximation function $f(x) = 62.222 - 2.93388x - 5.54473\sin(2\pi x) + 9.00424\cos(2\pi x)$ plot on top on the original data points.



4. When repeating the process of creating the design matrix from earlier but with two extra terms, $a_4 \sin(\pi x)$ and $a_5 \cos(\pi x)$, we find that our matrix A is now six by six matrix.

5. We see from this graph that the original approximation seems to fit the data better.



6.(a) The average value $\hat{F}$ of the first approximation function is as follows:

$$\begin{aligned}\hat{F} = \frac{1}{3-1}\int_1^3 f(x)dx &= \frac{1}{2}\int_1^3 62.222 - 2.93388x - 5.54473\sin(2\pi x) + 9.00424\cos(2\pi x)dx \\ &= \frac{1}{2}\left[62.222x - \frac{2.93388}{2}x^2 + \frac{5.54473}{2\pi}\cos(2\pi x) + \frac{9.00424}{2\pi}\sin(2\pi x)\right]_1^3 \\ &= 55.35419553385757\end{aligned}$$

(b) A few thing we can do with the function that we cannot do with the data is take the first and second derivatives to find the rate of change and concavity of the function. We can also use the function to interpolate missing data values within the current domain of our independent variable. We can also use it to extrapolate values beyond the current domain of our data.

7. Some other strategies that we may use to approximate this data are splines. We could easily fits cubic splines (or some other degree) with knots between each of the peaks and valleys.

Also, if we wanted to capture the long term trend of the data, we could use linear regression.

```
import random
import sys
import os   #
import math
import numpy as np     # Use every time
import matplotlib.pyplot as plt    # plot package
import IPython       # Interactive python-
import scipy, pylab # scientific python
import matplotlib
import sympy   # symbolic
import pandas as pd


print("go")
print("  ")

weather = r'weather.txt'

df = pd.read_csv(weather)
df = pd.DataFrame(df)

print('ijij', len(df.Time))

m = np.array(pd.read_csv(weather))
y = m[:,1]
y = np.array(y)
x = m[:,0]

D = np.zeros((len(x),4))
a = np.zeros(4)



for j in range(0,4):
    for i in range(0,len(x)):
        if(j==0):
         D[i,j]=1
        elif(j==1):
         D[i,j]= x[i]
        elif(j==2):
         D[i,j]= np.sin(2*np.pi*x[i])
        else:
         D[i,j]= np.cos(2*np.pi*x[i])

print()

a = np.linalg.solve(np.matmul(np.transpose(D),D), np.matmul(np.transpose(D),y))


D2 = np.zeros((len(x),6))
a2 = np.zeros(6)
```

```
for j in range(0,6):
    for i in range(0,len(x)):
        if(j==0):
         D2[i,j]=1
        elif(j==1):
         D2[i,j]= x[i]
        elif(j==2):
         D2[i,j]= np.sin(2*np.pi*x[i])
        elif(j==3):
         D2[i,j]= np.cos(2*np.pi*x[i])
        elif(j==4):
         D2[i,j] = np.sin(np.pi*x[i])
        else:
         D2[i,j] = np.cos(np.pi*x[i])

a2 = np.linalg.solve(np.matmul(np.transpose(D2),D2), np.matmul(np.transpose(D2),y))

xxx = np.arange(-1, 4, .0001)

def f(x):
    return a[0]+a[1]*x + a[2]*np.sin(2*np.pi*x) + a[3]*np.cos(2*np.pi*x)

def f2(x):
    return a2[0]+a2[1]*x + a2[2]*np.sin(2*np.pi*x) + a2[3]*np.cos(2*np.pi*x)
    +a2[4]*np.sin(np.pi*x[i])+a2[5]*np.cos(np.pi*x[i])

A = np.matmul(np.transpose(D),D)

A2 = np.matmul(np.transpose(D2),D2)


F = scipy.integrate.quad(f,1,3)

print(F[0]*(1/2))
print(np.mean(y))

plt.scatter(x, y, color = 'brown')
plt.plot(xxx, f2(xxx), color='orange')
plt.plot(xxx,f(xxx) )
plt.show()
```