PRACTISE SESSION 5 SOLUTIONS

Write a Python module to display the largest number from a list. The input list is [4,6,2,4,8,12,7], hence the output should be 12. Print the index of the largest number. Your output will be: 12 and 5. Write your own module and function(s) first, then find in-built or third party modules for solving the problem.

```
def my_max(numbers):
  maximum = 0
  maximum_index=0
  for i in range(len(numbers)):
    if maximum < numbers[i]:
      maximum = numbers[i]
      maximum_index = i
  return maximum, maximum_index

input_numbers = [4,6,2,4,8,12,7]
print(my_max(input_numbers))


maximum_2 = max(input_numbers)
maximum_index_2 = input_numbers.index(maximum_2)
print(maximum_2, maximum_index_2)
```

Define a function sum_of_reciprocals() which takes a positive integer N as an input and returns the sum of the reciprocals for values 1 to N (inclusive), i.e. the sum of the values: 1/1 + 1/2 + 1/3 + … + 1/N. E.g, a function call sum_of_reciprocals(2) should return 1.5 as its result. Be careful to avoid division-by-zero errors.

```
def sum_of_reciprocals(n):
  sum_of_values = 0
  for i in range(1,n+1):
    print(i)
    sum_of_values = sum_of_values + 1/i
  return sum_of_values

print(sum_of_reciprocals(2))
```

PRACTISE SESSION 6 SOLUTIONS

Two asteroids are on a near collision course with each other. Your task is to estimate their positions over time and plot the distance between them. We can model their movements in two dimensions (as they are travelling in the same plane). Their initial positions and velocities are as follows:

Asteroid 1 position = (150.4, 200.5) and velocity = (4.9, 7.1)
Asteroid 2 position = (122.6, 64.0) and velocity =(5.2, 2.95)

You must read the numbers from the provided text file. The text file contains some additional information which can be ignored.

Asteroid 1 has initial position x=150.4; y=200.5, relative to the origin, that has been chosen to be close to where the paths of the two asteroids will cross. The units here are unimportant (but each unit might correspond to 1000km, perhaps). The velocity of the asteroid 1 is 4.9, 7.1, given as units per hour, with separate components for x and y directions, i.e. it is moving at 4.9 units per hour in the x direction, and at 7.1 units per hour in the y direction. Your task is to compute the positions of the two asteroids at one hour intervals across a 50 hour period, and to estimate the distance between them against time. The distance between positions (x1, y1) and (x2, y2) can be computed using Pythagoras' theorem.

```
file = open("asteroid.txt","r")

x=[]
y=[]
for line in file:
   line = line.rstrip('\n')
   line = line.split(' ')
   if len(line) == 2:
      x.append(line[0])
      y.append(line[1])
file.close()

a1_p_x = x[0]
a1_v_x = x[1]
a1_p_y = y[0]
a1_v_y = y[1]

a2_p_x = x[2]
a2_v_x = x[3]
a2_p_y = y[2]
a2_v_y = y[3]
```

```
import numpy as np

def distance(a1x, a1y, a2x, a2y):
   return np.sqrt((a1x - a2x) ** 2+ (a1y - a2y) ** 2)

result = []

for i in range(50):
   a1x = a1_p_x + a1_v_x * i
   a1y = a1_p_y + a1_v_y * i

   a2x = a2_p_x + a2_v_x * i
   a2y = a2_p_y + a2_v_y * i

   result.append(distance(a1x, a1y, a2x, a2y))

print(result)
```

MATPLOTLIB

A numerical plotting library

Matplotlib is a Python plotting library which produces publication quality figures.

You can generate plots, histograms, power spectra, bar charts, scatter plots, etc., with just a few lines of code.

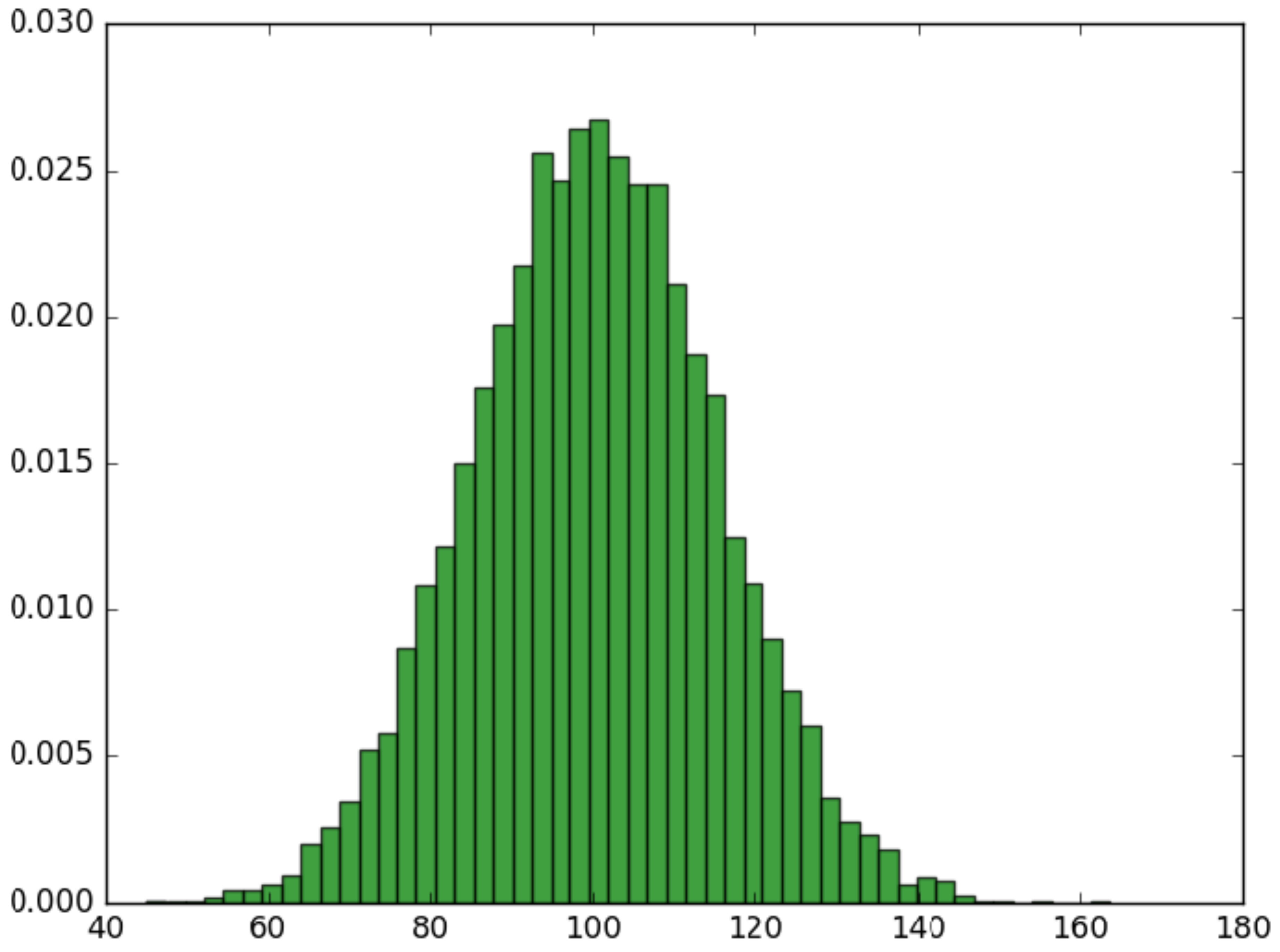
1D DATASET

Histogram

```
# Import matplotlib
import matplotlib.pyplot as plt

# Generate some data
data = [0,1,2,2,2,3,4]

# the histogram of the data
plt.hist(data)

# Show the histogram
plt.show()
```

Read the hist.txt file and create a histogram. Calculate the mean and the standard deviation of the data.

```
import matplotlib.pyplot as plt
import numpy as np

file = open('hist.txt','r')

data=[]
for line in file:
  line = line.rstrip('\n')
  data.append(float(line))
file.close()

# the histogram of the data
plt.hist(data)

#Show the histogram
plt.show()

# Mean and standard deviation
print(np.mean(data), np.std(data))
```



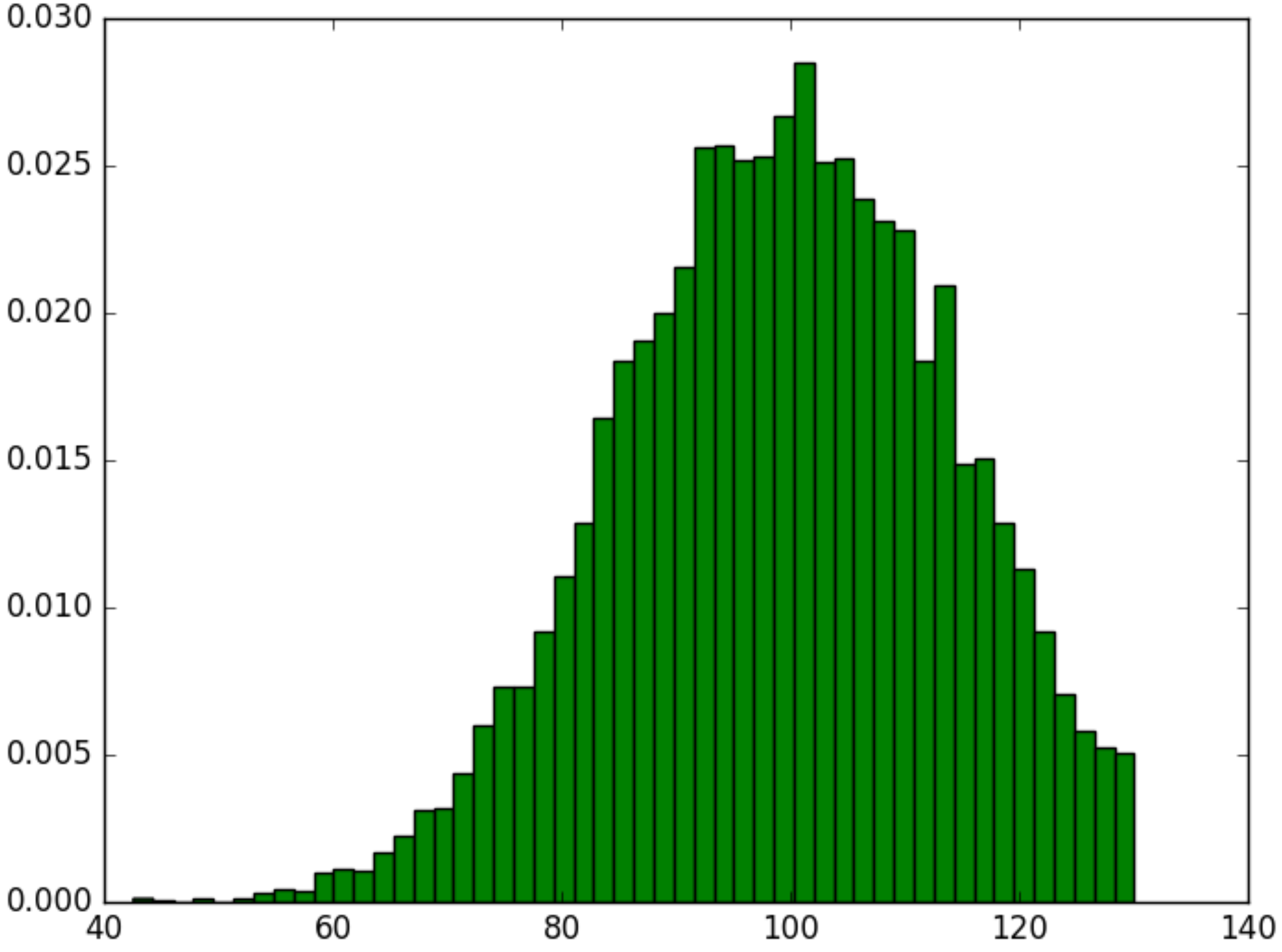
```
plt.hist(x,50,normed=1,facecolor='green')
```

Read the hist.txt file and create a histogram. Calculate the mean and the standard deviation of the data.

Set the threshold to 2 sigmas and delete the data above the threshold!

th = np.mean(data) + 2 * np.std(data)

```
import matplotlib.pyplot as plt
import numpy as np

file = open('hist.txt','r')

data=[]
filtered=[]
for line in file:
   line = line.rstrip('\n')
   data.append(float(line))
file.close()

TH = np.mean(data) + 2 * np.std(data)

for item in data:
   if item < TH:
      filtered.append(item)

# the histogram of the data
plt.hist(filtered,50,normed=1,facecolor='green')

#Show the histogram
plt.show()
```


0.030
0.025
0.020
0.015
0.010
0.005
0.000
40
60
80
100
120
140

2D DATASET

Lineplot

```
# Import marplot
import matplotlib.pyplot as plt

# Generate some data
x = [1,2,3,4,5,6,7]
y = [0,2,1,3,4,3,4]

# the histogram of the data
plt.plot(x,y)

# Show the histogram
plt.show()
```

2D DATASET

Scatter plot

```
# Import marplot
import matplotlib.pyplot as plt

# Generate some data
x = [1,2,3,4,5,6,7]
y = [0,2,1,3,4,3,4]

# the histogram of the data
plt.scatter(x,y)

# Show the histogram
plt.show()
```

2D DATASET

Least squares polynomial fit

Fit a polynomial p(x) = p[0] * x**deg + ... + p[deg] of degree *deg* to points *(x, y)*. Returns a vector of coefficients *p* that minimises the squared error.

```
z = np.polyfit(x, y, deg)
```

If deg == 1: p(x) = p[0] + p[1] * x ———————————Linear: f(x) = a + b*x

If deg == 2: p(x) = p[0] + p[1] * x + p[2] * x ** 2 ———Parabolic: f(x) = a + b*x + c**x

https://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html

2D DATASET

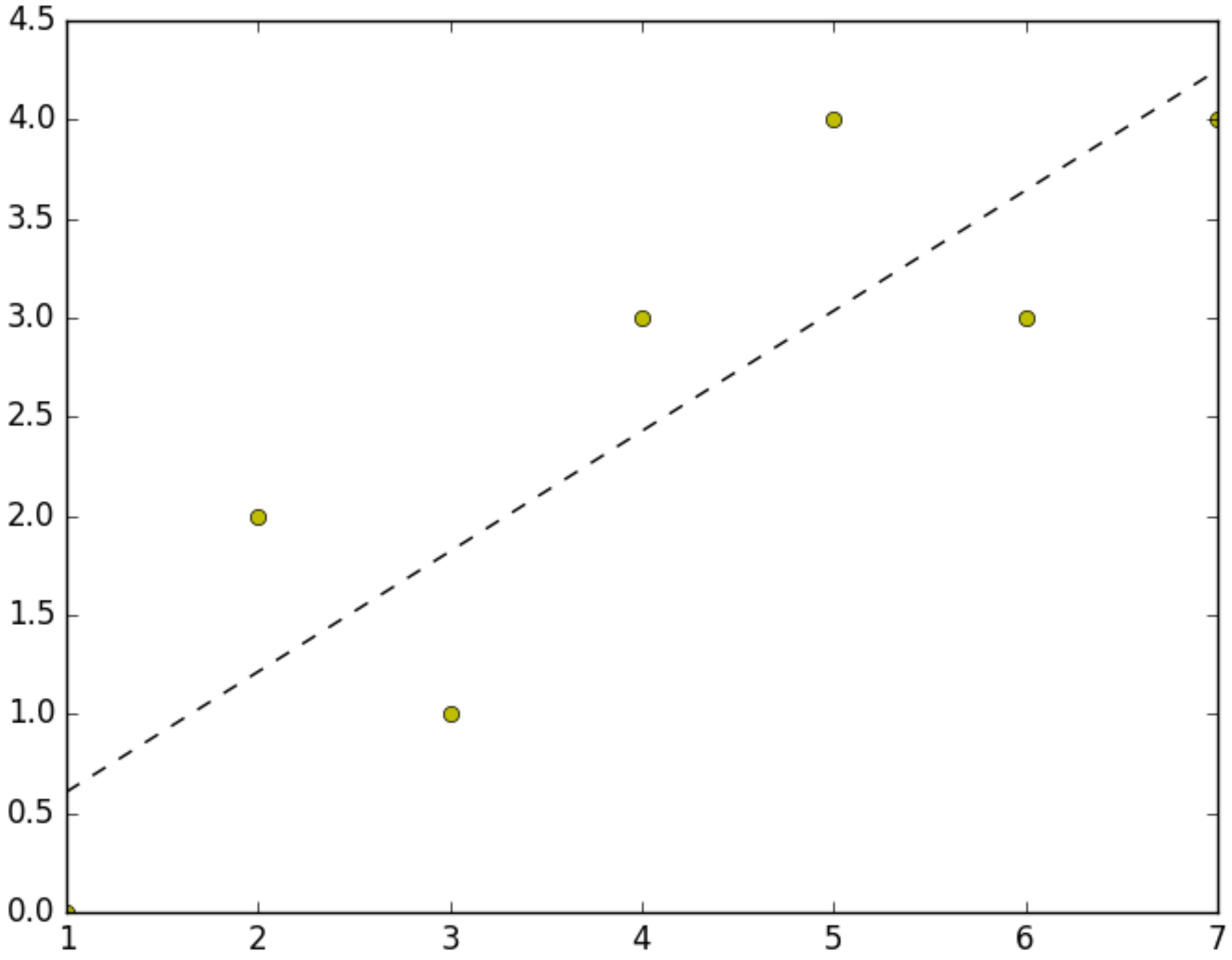
Least squares polynomial fit.

```
# Import marplot
import matplotlib.pyplot as plt

# Generate some data
x = [1,2,3,4,5,6,7]
y = [0,2,1,3,4,3,4]

# Regression line
z = np.polyfit(x, y, 1)
fit = np.poly1d(z)

plt.plot(x, y, 'yo', x, fit(x), ‘--k')
plt.show()
```

https://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html

2D DATASET

Linear regression



2D DATASET

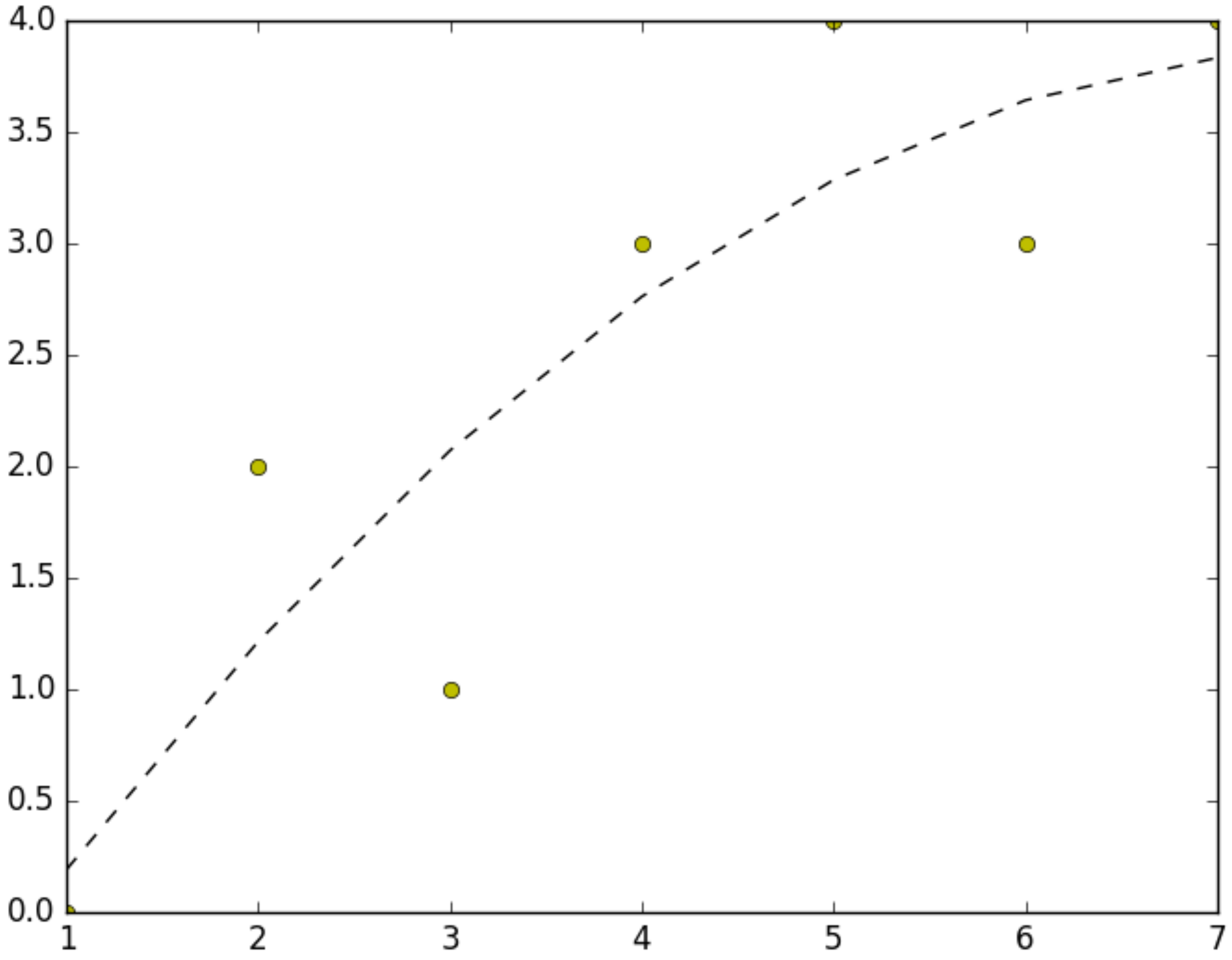
Least squares polynomial fit.

```
# Import marplot
import matplotlib.pyplot as plt

# Generate some data
x = [1,2,3,4,5,6,7]
y = [0,2,1,3,4,3,4]

# Regression line
z = np.polyfit(x, y, 2)
fit = np.poly1d(z)

plt.plot(x, y, 'yo', x, fit(x), ‘--k')
plt.show()
```

https://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html

2D DATASET

Parabolic regression



PRACTISE SESSION 7

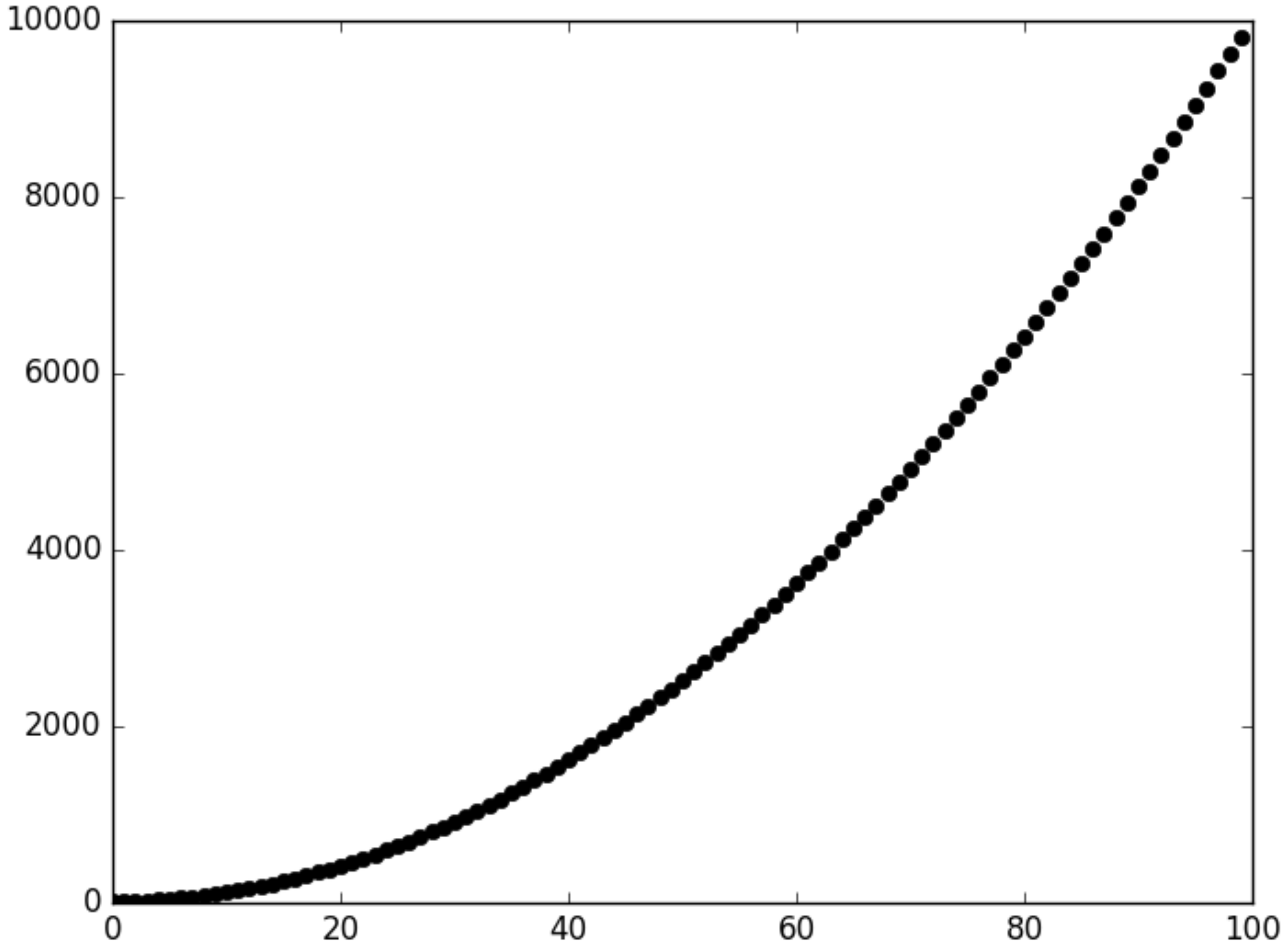
PRACTISE SESSION 7 SOLUTIONS

Plot a graph of the function f(x) = x**2 + 20, for integer values of x in the range 0-100. We can build the lists of x and y values with a for loop. Build the list of y values by starting with an empty list and appending values to it.

- Having created the lists of x and y values, plot the graph.
- Generate noise with the random module. The function is f(x) = x**2 + 20 + R, where R is the noise term, an integer between 0 and 1000. Plot the graph.
- The noise represents the observational error of the investigated system. Try to restore the noiseless system by applying a polynomial fit. Try to change the degree of the polynomial fit. Find the best fit.

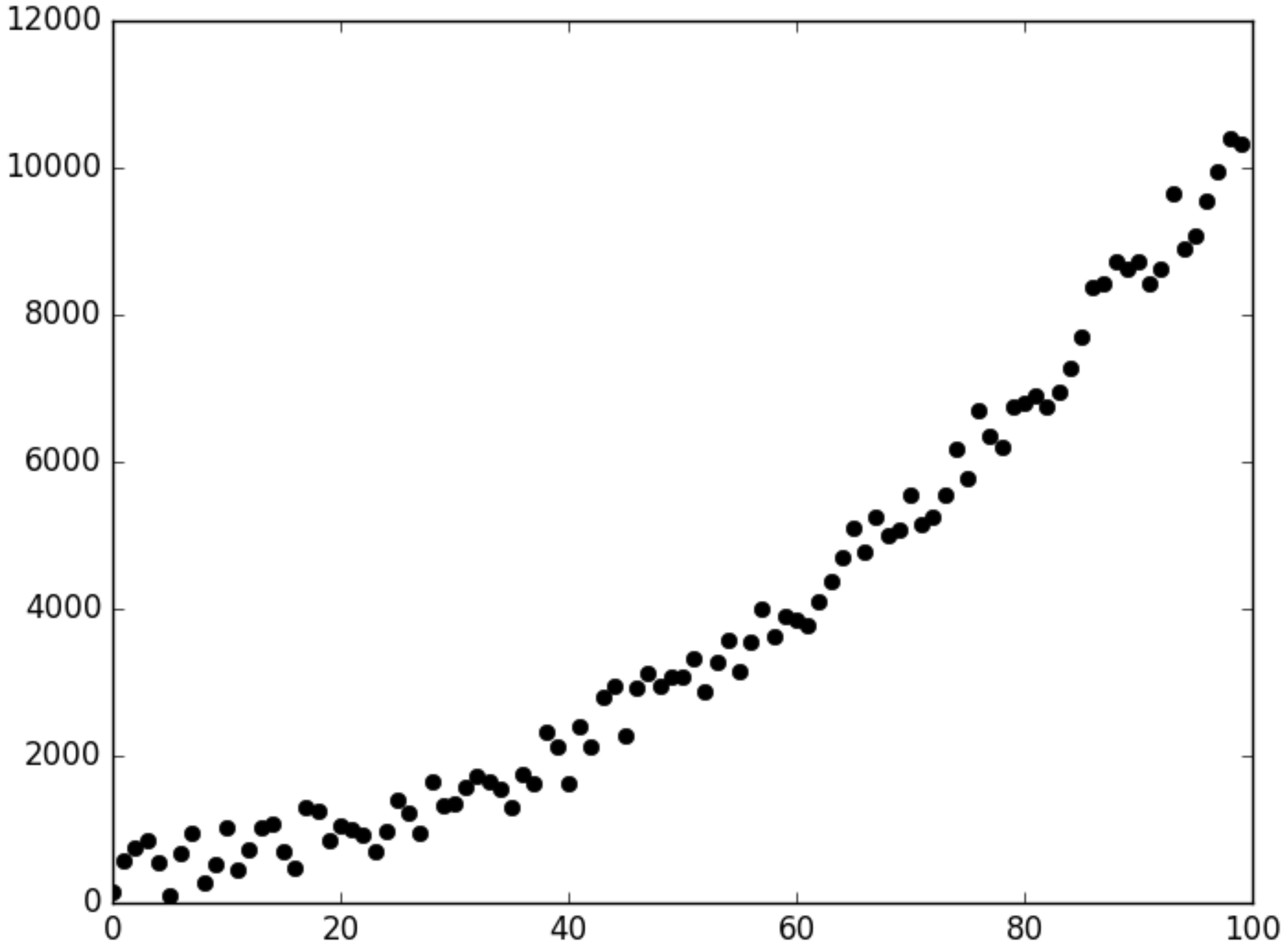
Plot the graph

```
import matplotlib.pyplot as plt

x = []
y = []

for i in range(100):
  x.append(i)
  y.append(i**2 + 20)

plt.plot(x,y,’ko’)
plt.show()
```


10000
8000
6000
4000
2000
0
0
20
40
60
80
100

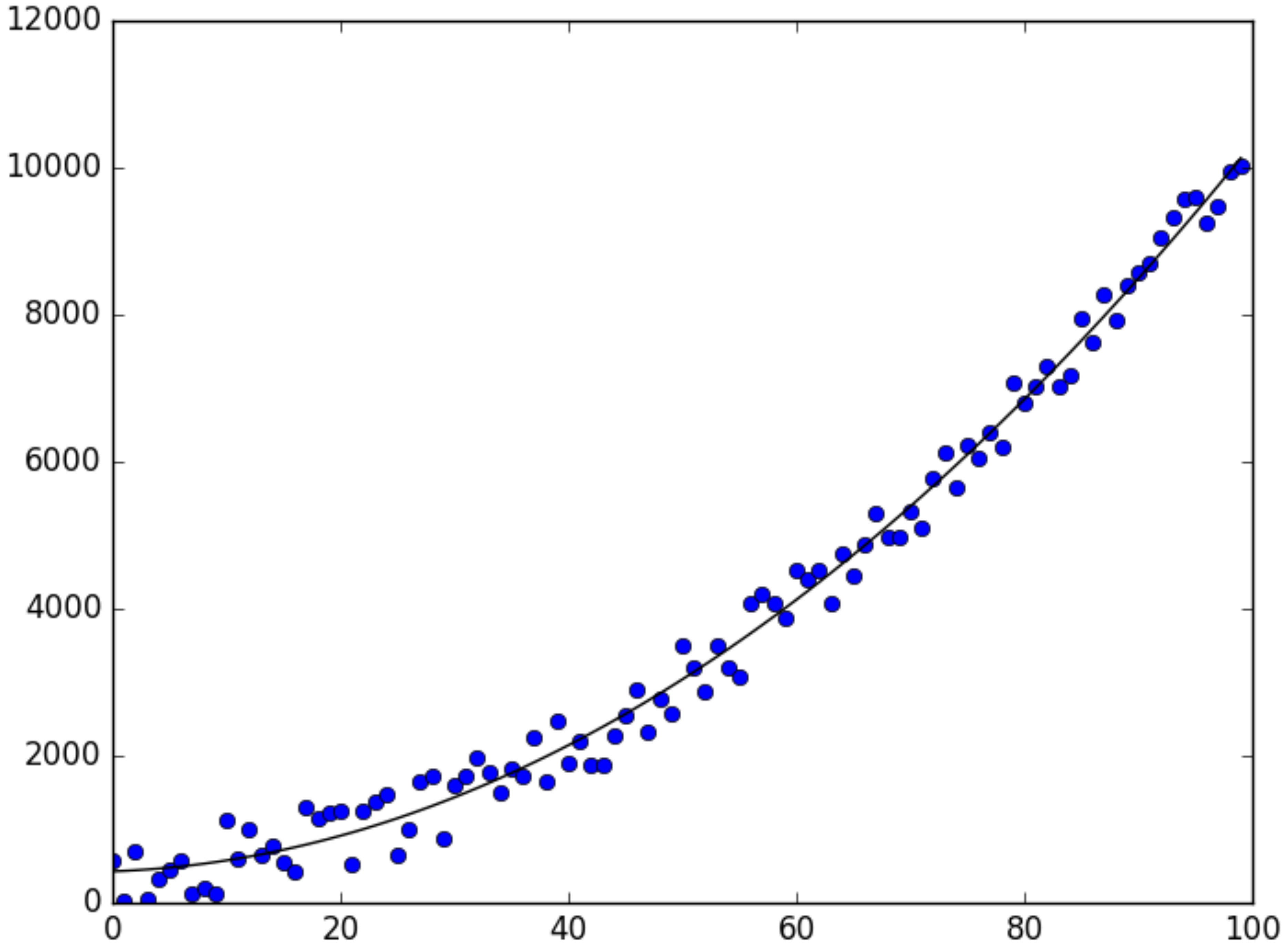
Generate noise

```
import matplotlib.pyplot as plt
import random as r

x = []
y = []

for i in range(100):
  x.append(i)
  y.append(i**2 + 20 + r.randint(0,1000))

plt.plot(x,y,'ko')
plt.show()
```


12000
10000
8000
6000
4000
2000
0
0
20
40
60
80
100

Polynomial fit

```
import matplotlib.pyplot as plt
import random as r
import numpy as np

x = []
y = []

for i in range(100):
  x.append(i)
  y.append(i**2 + 20 + r.randint(0,1000))

z = np.polyfit(x, y, 2)
fit = np.poly1d(z)

plt.plot(x, y, ’bo’)
plt.plot(x, fit(x), ’k-’)
plt.show()
```



12000
10000
8000
6000
4000
2000
0
0
20
40
60
80
100

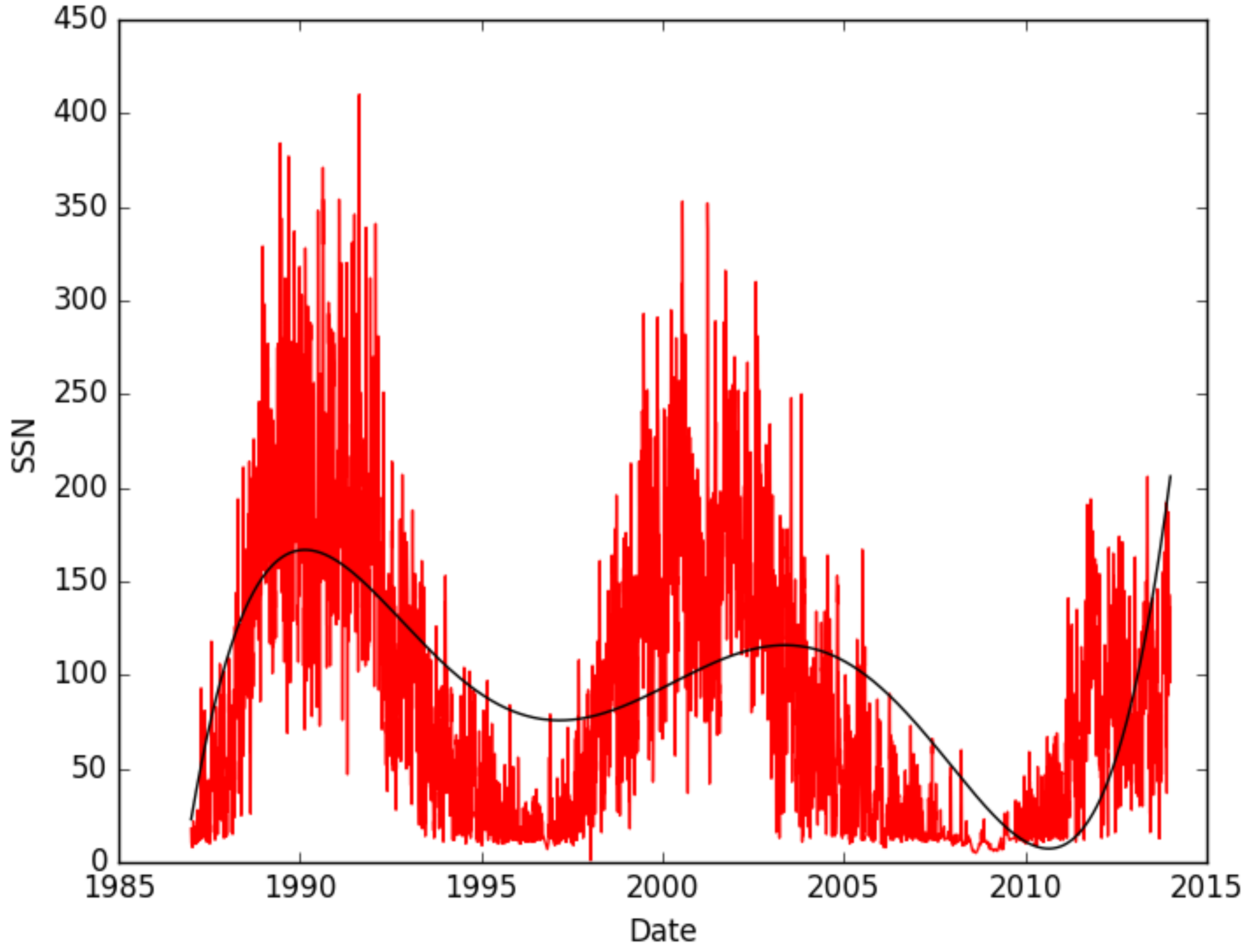
Your supervisor is interested in solar physics, hence he asks you to create a fancy chart about the variation of the sunspot numbers between the years 1987 and 2014.

Download the data and read it

```
file = open("SN_d_tot_V2.0.csv","r")

date=[]
sunspot_number=[]
for line in file:
  line = line.rstrip('\n')
  line = line.split(';')
  if float(line[3]) > 1987 and float(line[3]) < 2014:
    if float(line[4]) > 0:
      date.append(float(line[3]))
      sunspot_number.append(float(line[4]))
file.close()
```

Plot and fit the data

```
import numpy as np
import matplotlib.pyplot as plt

z = np.polyfit(date, sunspot_number, 20)
fit = np.poly1d(z)

plt.plot(date, sunspot_number, 'r-')
plt.plot(date, fit(date), 'k-')
plt.xlabel('Date')
plt.ylabel('SSN')
plt.show()
```


450
400
350
300
250
200
150
100
50
0
SSN
1985
1990
1995
2000
2005
2010
2015
Date