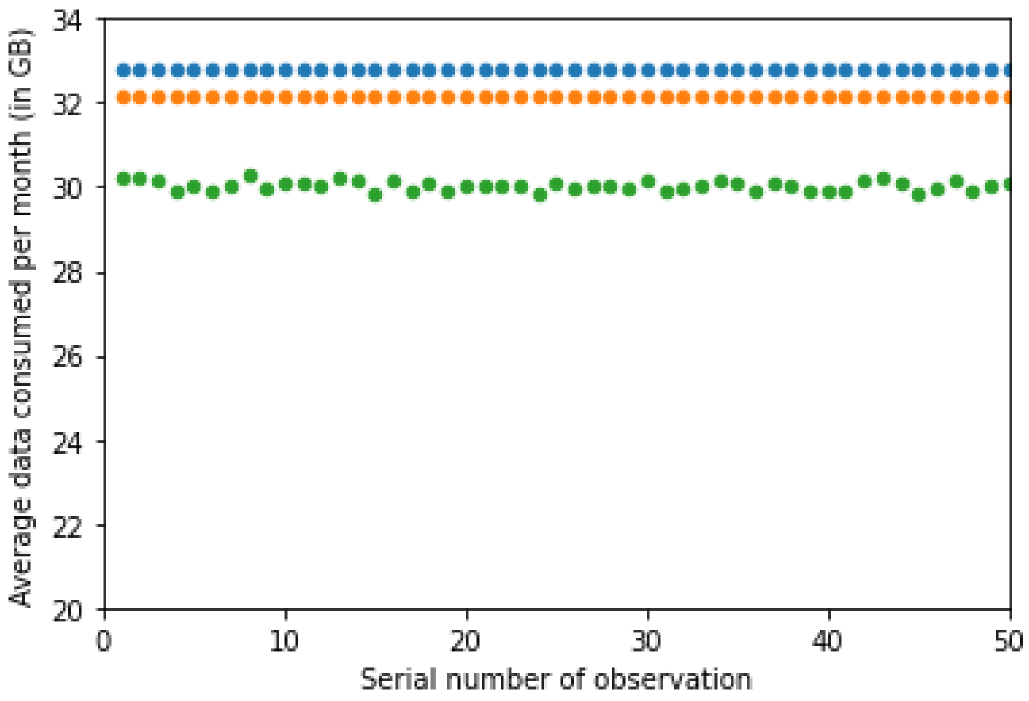
Problem 1

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

import pandas as pd
df = pd.read_csv('hw1a.csv')
list1 = df['datalist'].tolist()
#print(list1)

K = int(input("\nPlease enter the desired value of K: "))

# number of iterations
n = 50

# standard deviation
std_deviation = 0

  # for case(a)
print("\n(a) Ask the first K students that you find as soon as you
enter the campus:")
p = 0
y_values_1 = []

for j in range (0 , n):
 for i in range (0 , K):
  p += list1[i]
 p = p/K
 y_values_1.append(p)
 p = 0

#print(str(x_values_1))

#p = p/K
print("The average monthly data consumption is expected to be: " +
str(sum(y_values_1)/n) + " GB.")
print("The standard deviation of the statistical data is: " +
str(std_deviation) +" GB.")

  # for case(b)
print("\n(b) Choose an arbitrary point in the campus and ask K
students from there:")

q = 0
index_list = []
list2 = []
y_values_2 = []
```

```
for i in range (0 , n):
 index_list.append(i)

for i in range(0 , n):
 random_index = random.choice(index_list)
 if(random_index + K <= len(index_list)):
  list2 = list1[random_index : K + random_index - 1 : 1]
  q = sum(list2)
 else:
  list2 = list1[random_index : n-1 : 1]
  for j in range (0 , K - (n-random_index)):
   list2.append(list1[j])
  q = sum(list2)
 q = q/K
 y_values_2.append(q)
 q = 0

mean_2 = sum(y_values_2)/n
for i in range(0 , n):
  std_deviation += y_values_2[i]**2 + mean_2**2 - 2 * mean_2 *
y_values_2[i]

print("The average monthly data consumption is expected to be: " +
str(sum(y_values_2)/n) + " GB.")
print("The standard deviation of the statistical data is: " +
str(math.sqrt(std_deviation/K)) +" GB.")

std_deviation = 0

  # for case(c)
print("\n(c) Randomly select K from the 10,000 people in the colony,
i.e., let's say we put all the 10,000 names in a pot, mix the pot
thoroughly and pick a name. Repeat K times:")
r = 0
empty_list_3 = [K]
y_values_3 = []

for j in range (0 , n):
 for i in range (0 , n):
  empty_list_3 = random.choices(list1 , k = K)
  r += sum(empty_list_3)
 r = r/(K * n)
 y_values_3.append(r)
 r = 0

mean_3 = sum(y_values_3)/n
for i in range(0 , n):
  std_deviation += y_values_3[i]**2 + mean_3**2 - 2 * mean_3 *
```

```
y_values_3[i]

print("The average monthly data consumption is expected to be: " +
str(sum(y_values_3)/n) + " GB.")
print("The standard deviation of the statistical data is: " +
str(math.sqrt(std_deviation/K)) +" GB.")

x_values = []
for i in range (0 , n):
  x_values.append(i+1)

plt.axis([0, 50, 20, 34])

plt.scatter(x_values , y_values_1 , s=20)
plt.scatter(x_values , y_values_2 , s=20)
plt.scatter(x_values , y_values_3 , s=20)

plt.xlabel('Serial number of observation')
plt.ylabel('Average data consumed per month (in GB)')

print("\nN.B. : The blue, orange and green curves correspond to cases
(a), (b) and (c), respectively.\n")
```

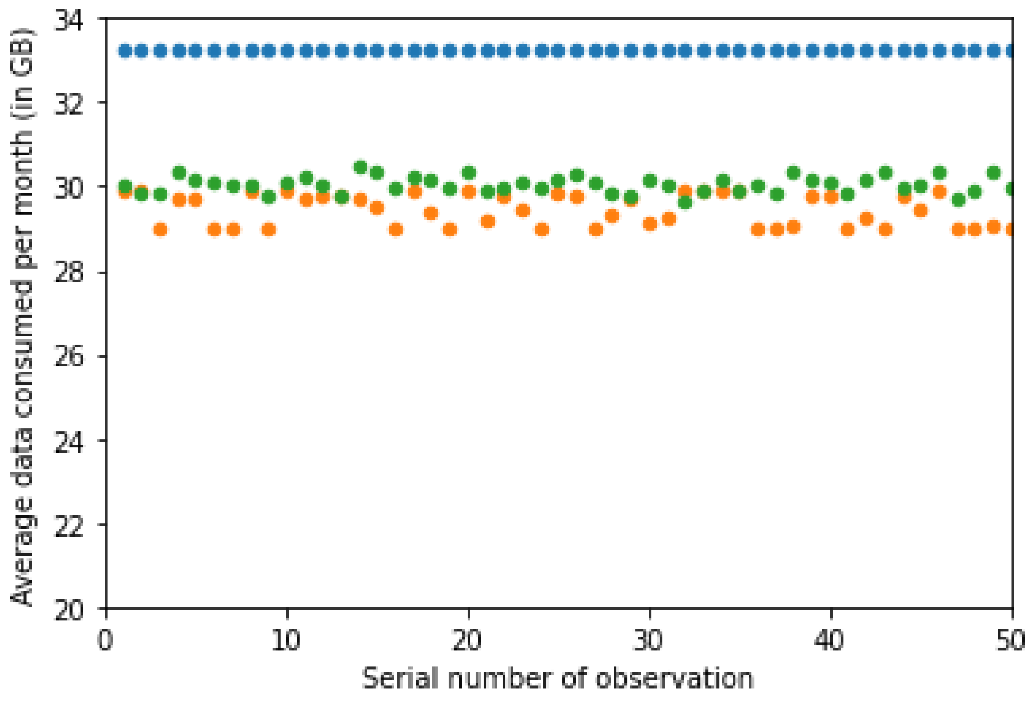
```
Please enter the desired value of K: 50

(a) Ask the first K students that you find as soon as you enter the
campus:
The average monthly data consumption is expected to be:
32.793037780776174 GB.
The standard deviation of the statistical data is: 0 GB.

(b) Choose an arbitrary point in the campus and ask K students from
there:
The average monthly data consumption is expected to be:
32.13294261927314 GB.
The standard deviation of the statistical data is: 0.0 GB.

(c) Randomly select K from the 10,000 people in the colony, i.e.,
let's say we put all the 10,000 names in a pot, mix the pot thoroughly
and pick a name. Repeat K times:
The average monthly data consumption is expected to be:
30.031890358037035 GB.
The standard deviation of the statistical data is: 0.10816692339742412
GB.

N.B. : The blue, orange and green curves correspond to cases (a), (b)
and (c), respectively.
```



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

import pandas as pd
df = pd.read_csv('hw1a.csv')
list1 = df['datalist'].tolist()
#print(list1)

K = int(input("\nPlease enter the desired value of K: "))

# number of iterations
n = 50

# standard deviation
std_deviation = 0

  # for case(a)
print("\n(a) Ask the first K students that you find as soon as you
enter the campus:")
p = 0
y_values_1 = []

for j in range (0 , n):
 for i in range (0 , K):
  p += list1[i]
 p = p/K
 y_values_1.append(p)
 p = 0

#print(str(x_values_1))

#p = p/K
print("The average monthly data consumption is expected to be: " +
str(sum(y_values_1)/n) + " GB.")
print("The standard deviation of the statistical data is: " +
str(std_deviation) +" GB.")

  # for case(b)
print("\n(b) Choose an arbitrary point in the campus and ask K
students from there:")

q = 0
index_list = []
list2 = []
y_values_2 = []

for i in range (0 , n):
 index_list.append(i)

for i in range(0 , n):
 random_index = random.choice(index_list)
 if(random_index + K <= len(index_list)):
  list2 = list1[random_index : K + random_index - 1 : 1]
  q = sum(list2)
 else:
  list2 = list1[random_index : n-1 : 1]
  for j in range (0 , K - (n-random_index)):
   list2.append(list1[j])
  q = sum(list2)
 q = q/K
 y_values_2.append(q)
 q = 0

mean_2 = sum(y_values_2)/n
for i in range(0 , n):
  std_deviation += y_values_2[i]**2 + mean_2**2 - 2 * mean_2 *
y_values_2[i]

print("The average monthly data consumption is expected to be: " +
str(sum(y_values_2)/n) + " GB.")
print("The standard deviation of the statistical data is: " +
str(math.sqrt(std_deviation/K)) +" GB.")

std_deviation = 0
```

```
  # for case(c)
print("\n(c) Randomly select K from the 10,000 people in the colony,
i.e., let's say we put all the 10,000 names in a pot, mix the pot
thoroughly and pick a name. Repeat K times:")
r = 0
empty_list_3 = [K]
y_values_3 = []

for j in range (0 , n):
 for i in range (0 , n):
  empty_list_3 = random.choices(list1 , k = K)
  r += sum(empty_list_3)
 r = r/(K * n)
 y_values_3.append(r)
 r = 0

mean_3 = sum(y_values_3)/n
for i in range(0 , n):
  std_deviation += y_values_3[i]**2 + mean_3**2 - 2 * mean_3 *
y_values_3[i]

print("The average monthly data consumption is expected to be: " +
str(sum(y_values_3)/n) + " GB.")
print("The standard deviation of the statistical data is: " +
str(math.sqrt(std_deviation/K)) +" GB.")

x_values = []
for i in range (0 , n):
  x_values.append(i+1)

plt.axis([0, 50, 20, 34])

plt.scatter(x_values , y_values_1 , s=20)
plt.scatter(x_values , y_values_2 , s=20)
plt.scatter(x_values , y_values_3 , s=20)

plt.xlabel('Serial number of observation')
plt.ylabel('Average data consumed per month (in GB)')

print("\nN.B. : The blue, orange and green curves correspond to cases
(a), (b) and (c), respectively.\n")


Please enter the desired value of K: 10

(a) Ask the first K students that you find as soon as you enter the
campus:
The average monthly data consumption is expected to be:
33.22836173233857 GB.
```

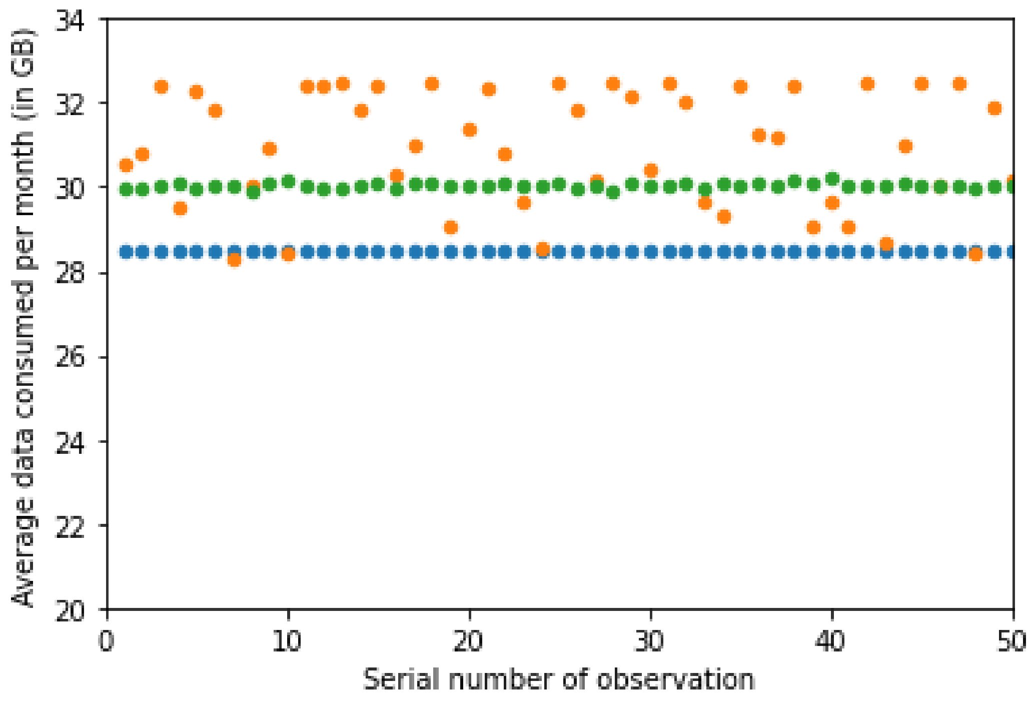
```
The standard deviation of the statistical data is: 0 GB.

(b) Choose an arbitrary point in the campus and ask K students from
there:
The average monthly data consumption is expected to be:
29.46609870883843 GB.
The standard deviation of the statistical data is: 0.8280417556088061
GB.

(c) Randomly select K from the 10,000 people in the colony, i.e.,
let's say we put all the 10,000 names in a pot, mix the pot thoroughly
and pick a name. Repeat K times:
The average monthly data consumption is expected to be:
30.053783431643023 GB.
The standard deviation of the statistical data is: 0.4317693615206274
GB.

N.B. : The blue, orange and green curves correspond to cases (a), (b)
and (c), respectively.
```



```python
import matplotlib.pyplot as plt
import numpy as np
import math
import random

import pandas as pd
df = pd.read_csv('hw1a.csv')
```

```
list1 = df['datalist'].tolist()
#print(list1)

K = int(input("\nPlease enter the desired value of K: "))

# number of iterations
n = 50

# standard deviation
std_deviation = 0

  # for case(a)
print("\n(a) Ask the first K students that you find as soon as you
enter the campus:")
p = 0
y_values_1 = []

for j in range (0 , n):
 for i in range (0 , K):
  p += list1[i]
 p = p/K
 y_values_1.append(p)
 p = 0

#print(str(x_values_1))

#p = p/K
print("The average monthly data consumption is expected to be: " +
str(sum(y_values_1)/n) + " GB.")
print("The standard deviation of the statistical data is: " +
str(std_deviation) +" GB.")

  # for case(b)
print("\n(b) Choose an arbitrary point in the campus and ask K
students from there:")

q = 0
index_list = []
list2 = []
y_values_2 = []

for i in range (0 , n):
 index_list.append(i)

for i in range(0 , n):
 random_index = random.choice(index_list)
 if(random_index + K <= len(index_list)):
  list2 = list1[random_index : K + random_index - 1 : 1]
  q = sum(list2)
 else:
  list2 = list1[random_index : n-1 : 1]
  for j in range (0 , K - (n-random_index)):
   list2.append(list1[j])
  q = sum(list2)
 q = q/K
 y_values_2.append(q)
 q = 0

mean_2 = sum(y_values_2)/n
for i in range(0 , n):
  std_deviation += y_values_2[i]**2 + mean_2**2 - 2 * mean_2 *
y_values_2[i]

print("The average monthly data consumption is expected to be: " +
str(sum(y_values_2)/n) + " GB.")
print("The standard deviation of the statistical data is: " +
str(math.sqrt(std_deviation/K)) +" GB.")

std_deviation = 0

  # for case(c)
print("\n(c) Randomly select K from the 10,000 people in the colony,
i.e., let's say we put all the 10,000 names in a pot, mix the pot
thoroughly and pick a name. Repeat K times:")
r = 0
empty_list_3 = [K]
y_values_3 = []

for j in range (0 , n):
 for i in range (0 , n):
  empty_list_3 = random.choices(list1 , k = K)
  r += sum(empty_list_3)
 r = r/(K * n)
 y_values_3.append(r)
 r = 0

mean_3 = sum(y_values_3)/n
for i in range(0 , n):
  std_deviation += y_values_3[i]**2 + mean_3**2 - 2 * mean_3 *
y_values_3[i]

print("The average monthly data consumption is expected to be: " +
str(sum(y_values_3)/n) + " GB.")
print("The standard deviation of the statistical data is: " +
str(math.sqrt(std_deviation/K)) +" GB.")

x_values = []
for i in range (0 , n):
  x_values.append(i+1)

plt.axis([0, 50, 20, 34])

plt.scatter(x_values , y_values_1 , s=20)
plt.scatter(x_values , y_values_2 , s=20)
plt.scatter(x_values , y_values_3 , s=20)

plt.xlabel('Serial number of observation')
plt.ylabel('Average data consumed per month (in GB)')

print("\nN.B. : The blue, orange and green curves correspond to cases
(a), (b) and (c), respectively.\n")
```

```
Please enter the desired value of K: 100

(a) Ask the first K students that you find as soon as you enter the
campus:
The average monthly data consumption is expected to be:
28.458208625606552 GB.
The standard deviation of the statistical data is: 0 GB.

(b) Choose an arbitrary point in the campus and ask K students from
there:
The average monthly data consumption is expected to be:
30.950632011417916 GB.
The standard deviation of the statistical data is: 0.9858944789190984
GB.

(c) Randomly select K from the 10,000 people in the colony, i.e.,
let's say we put all the 10,000 names in a pot, mix the pot thoroughly
and pick a name. Repeat K times:
The average monthly data consumption is expected to be:
30.039164652956813 GB.
The standard deviation of the statistical data is:
0.044577461772946324 GB.

N.B. : The blue, orange and green curves correspond to cases (a), (b)
and (c), respectively.
```

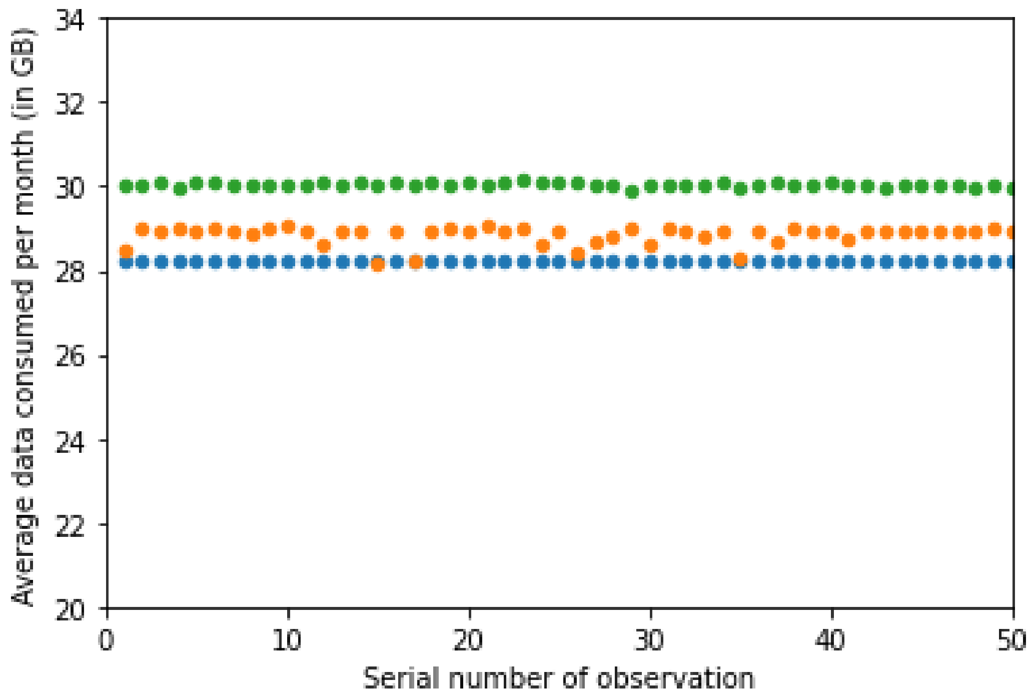


```
import matplotlib.pyplot as plt
import numpy as np
import math
import random

import pandas as pd
df = pd.read_csv('hw1a.csv')
list1 = df['datalist'].tolist()
#print(list1)

K = int(input("\nPlease enter the desired value of K: "))

# number of iterations
n = 50

# standard deviation
std_deviation = 0

  # for case(a)
print("\n(a) Ask the first K students that you find as soon as you
enter the campus:")
p = 0
y_values_1 = []

for j in range (0 , n):
 for i in range (0 , K):
  p += list1[i]
 p = p/K
 y_values_1.append(p)
 p = 0

#print(str(x_values_1))

#p = p/K
print("The average monthly data consumption is expected to be: " +
str(sum(y_values_1)/n) + " GB.")
print("The standard deviation of the statistical data is: " +
str(std_deviation) +" GB.")

  # for case(b)
print("\n(b) Choose an arbitrary point in the campus and ask K
students from there:")

q = 0
index_list = []
list2 = []
y_values_2 = []

for i in range (0 , n):
 index_list.append(i)

for i in range(0 , n):
 random_index = random.choice(index_list)
 if(random_index + K <= len(index_list)):
  list2 = list1[random_index : K + random_index - 1 : 1]
  q = sum(list2)
 else:
  list2 = list1[random_index : n-1 : 1]
  for j in range (0 , K - (n-random_index)):
   list2.append(list1[j])
  q = sum(list2)
 q = q/K
 y_values_2.append(q)
 q = 0

mean_2 = sum(y_values_2)/n
for i in range(0 , n):
  std_deviation += y_values_2[i]**2 + mean_2**2 - 2 * mean_2 *
y_values_2[i]

print("The average monthly data consumption is expected to be: " +
str(sum(y_values_2)/n) + " GB.")
print("The standard deviation of the statistical data is: " +
str(math.sqrt(std_deviation/K)) +" GB.")

std_deviation = 0
```

```
  # for case(c)
print("\n(c) Randomly select K from the 10,000 people in the colony,
i.e., let's say we put all the 10,000 names in a pot, mix the pot
thoroughly and pick a name. Repeat K times:")
r = 0
empty_list_3 = [K]
y_values_3 = []

for j in range (0 , n):
 for i in range (0 , n):
  empty_list_3 = random.choices(list1 , k = K)
  r += sum(empty_list_3)
 r = r/(K * n)
 y_values_3.append(r)
 r = 0

mean_3 = sum(y_values_3)/n
for i in range(0 , n):
  std_deviation += y_values_3[i]**2 + mean_3**2 - 2 * mean_3 *
y_values_3[i]

print("The average monthly data consumption is expected to be: " +
str(sum(y_values_3)/n) + " GB.")
print("The standard deviation of the statistical data is: " +
str(math.sqrt(std_deviation/K)) +" GB.")

x_values = []
for i in range (0 , n):
  x_values.append(i+1)

plt.axis([0, 50, 20, 34])

plt.scatter(x_values , y_values_1 , s=20)
plt.scatter(x_values , y_values_2 , s=20)
plt.scatter(x_values , y_values_3 , s=20)

plt.xlabel('Serial number of observation')
plt.ylabel('Average data consumed per month (in GB)')

print("\nN.B. : The blue, orange and green curves correspond to cases
(a), (b) and (c), respectively.\n")
```

```
Please enter the desired value of K: 200

(a) Ask the first K students that you find as soon as you enter the
campus:
The average monthly data consumption is expected to be:
28.260580340367138 GB.
```

```
The standard deviation of the statistical data is: 0 GB.

(b) Choose an arbitrary point in the campus and ask K students from
there:
The average monthly data consumption is expected to be:
28.855823733612507 GB.
The standard deviation of the statistical data is: 0.10644930012880942
GB.

(c) Randomly select K from the 10,000 people in the colony, i.e.,
let's say we put all the 10,000 names in a pot, mix the pot thoroughly
and pick a name. Repeat K times:
The average monthly data consumption is expected to be:
30.04437509602186 GB.
The standard deviation of the statistical data is: 0.02302855057752523
GB.

N.B. : The blue, orange and green curves correspond to cases (a), (b)
and (c), respectively.
```



1(b)(i) : My guess for the actual average data consumption is that it would be in the range of 29-32 GB/month. As regards the standard deviation, for K = 50 it should be very low while for lower values of K (e.g. 20, 30, 40,etc.) it will be significantly higher. An interesting, yet logically explainanble observation from the graphs is that as K increases, the values tend to get closer to a particular average value and the standard deviation also decreases.

The actual value of the average data (for K = 50) is nearly 30.043 GB/month. The actual value of the standard deviation (for K = 50) is 0.108 GB, which is quite negligible as compared to the average data consumption.

N.B. : The average and standard deviation values have been presented in the output of the code for all cases. Please look into the pictures of the output to see them.

1(b)(ii)(A) : The third scheme should be preferably used in practice to determine average data consumption. In the first scheme, the data available is not reliable since the students near the entrance would not ideally represent the population distribution of various student groups inside the campus. In the second scheme, the data available is too localized; since it is collected only around a single residential area. The data obtained from this scheme might very well not be applicable throughout the campus since some hostels might consume more or less data depending upon their requirements. So, overall, the third scheme is the best since its sample set is compltely randomized: it would take into consideration all possible student groups when we run multiple iterations of it.

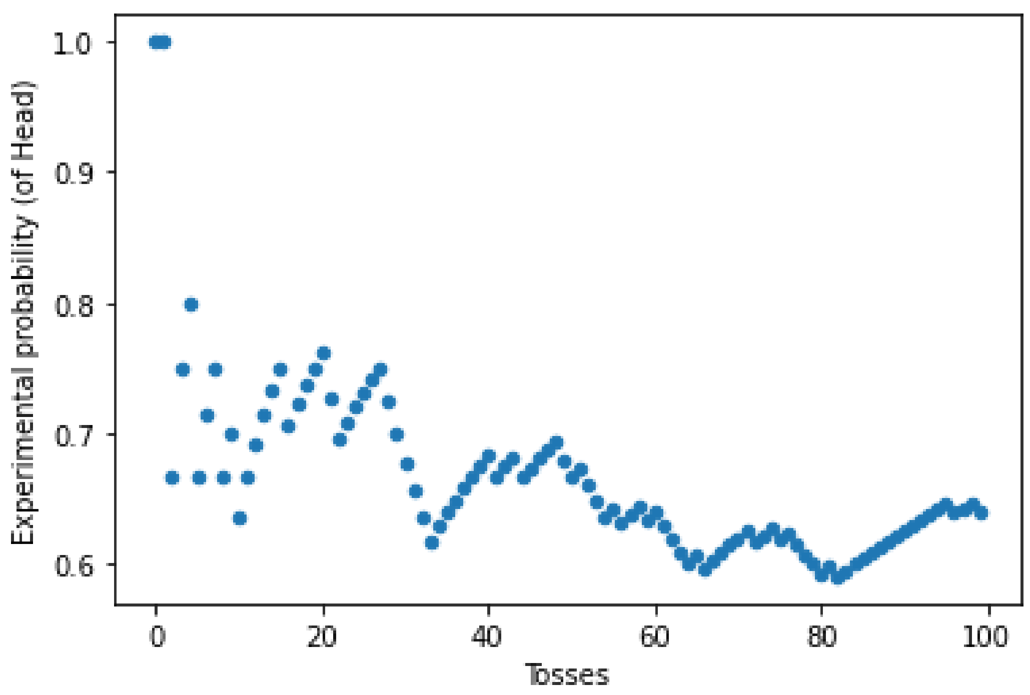
1(b)(ii)(B) : K should be chosen such that it is economical for measurement purposes as well as it represents a sufficiently large enough sample for determining the staistics of the situation correctly. In this case, I think that K = 50 is a good enough value, both economically and statistically. As you can observe from the graph, K = 50 is quite a unique case in the sense that the values deviate very little from the average in this case. The statistical data for the third scheme is very nearly constant, unlike other cases where it is in a highly scattered format. I believe that this graphical observation is sufficient enough to justify that the average that we obtain from this case is correct to a high degree of accuracy.

Problem 2

2(a)

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

b1=open('/content/hw1b1.txt','r')
b1_list = b1.readlines()
b1_list = list(map(int,b1_list))

# ASSUMPTION: Heads favoured

counts = [0,0]
probs = []

doubt_index = []
surety_index = []

for i in b1_list:
    if i == 0:
        counts[0] += 1
    if i == 1:
        counts[1] += 1

    probs.append(counts[1]/(counts[0]+counts[1]))

doubt = False
surety_against = False

for i in range(0,100):

    if probs[i] <= 0.56 and probs[i] > 0.40:
        if doubt == False:
            doubt = True
            doubt_index.append(i+1)

    if probs[i] <= 0.40:
        surety_against = True
        surety_index.append(i+1)

if not doubt_index and not surety_index:
 print("\nHe is convinced that his choice is correct when the first
coin is used for the experiment.")
else:
 print("\nThe points at which he suspects that he might be wrong are:
" + str(doubt_index))
 print("\nThe points at which he is convinced that he might be wrong
are: " + str(surety_index))

print("\nThe probability of getting a 'Head' from the first coin is: "
+ str(probs[-1]) + "\n")

x_values = range(0 , 100)
y_values = probs

plt.scatter(x_values , y_values , s=20)

plt.xlabel('Tosses')
plt.ylabel('Experimental probability (of Head)')
```

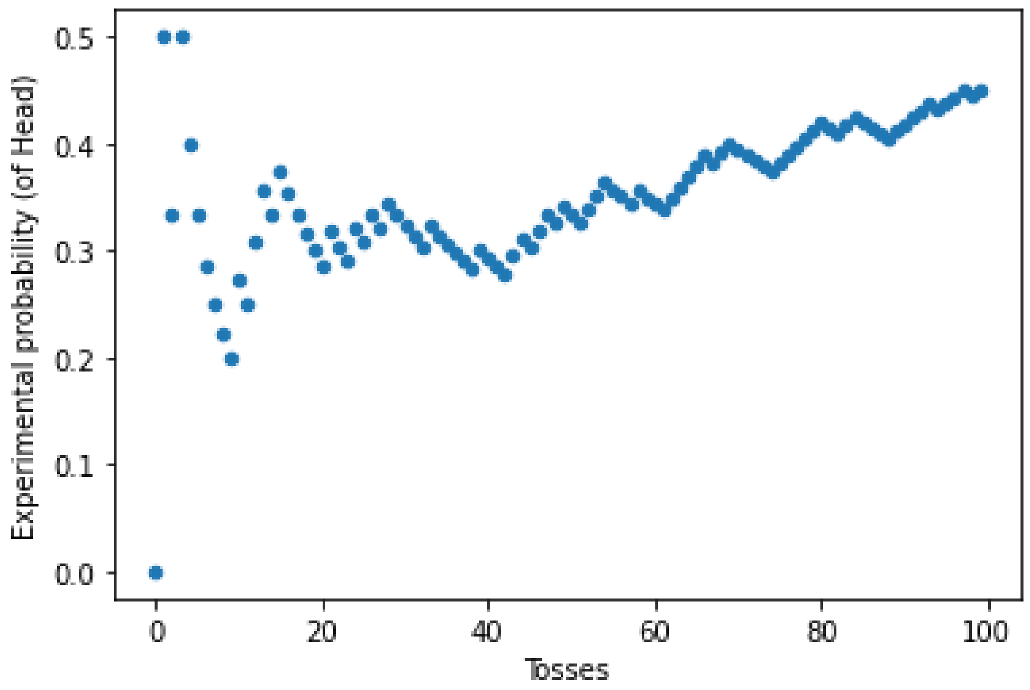
```
He is convinced that his choice is correct when the first coin is used
for the experiment.

The probability of getting a 'Head' from the first coin is: 0.64


Text(0, 0.5, 'Experimental probability (of Head)')
```



2(c)_hw1b2

```
b1=open('/content/hw1b2.txt','r')
b1_list = b1.readlines()
b1_list = list(map(int,b1_list))

# ASSUMPTION: Heads favoured

counts = [0,0]
probs = []

doubt_index = []
surety_index = []

for i in b1_list:
    if i == 0:
        counts[0] += 1
    if i == 1:
        counts[1] += 1
    probs.append(counts[1]/(counts[0]+counts[1]))

doubt = False
surety_against = False

for i in range(0,100):

    if probs[i] <= 0.56 and probs[i] > 0.40:
        if doubt == False:
            doubt = True
            doubt_index.append(i+1)

    if probs[i] <= 0.40:
        surety_against = True
        surety_index.append(i+1)

if not doubt_index and not surety_index:
 print("\nHe is convinced that his choice is correct when the second
coin is used for the experiment.")
else:
 print("\nThe points at which he suspects of being wrong are: " +
str(doubt_index))
 print("\nThe points at which he is convinced that he might be wrong
are: " + str(surety_index))

print("\nThe probability of getting a 'Head' from the second coin is:
" + str(probs[-1]) + "\n")

x_values = range(0 , 100)
y_values = probs

plt.scatter(x_values , y_values , s=20)

plt.xlabel('Tosses')
plt.ylabel('Experimental probability (of Head)')
```

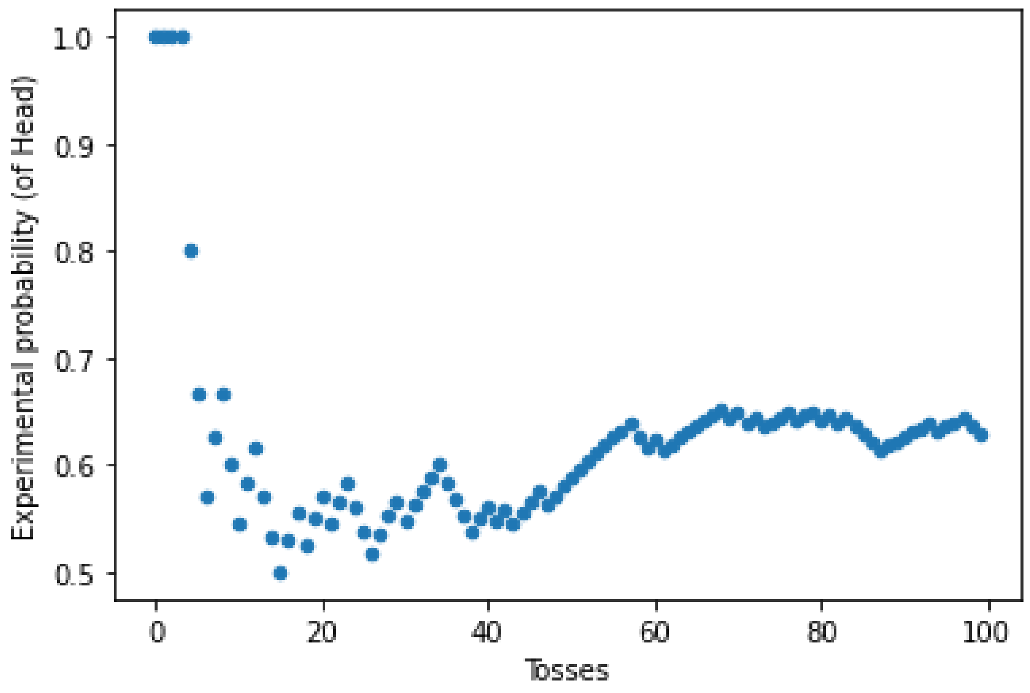
```
The points at which he suspects of being wrong are: [2]

The points at which he is convinced that he might be wrong are: [1, 3,
5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,
58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,
75, 76, 77, 78]

The probability of getting a 'Head' from the second coin is: 0.45


Text(0, 0.5, 'Experimental probability (of Head)')
```



2(c)_hw1b3

```
b1=open('/content/hw1b3.txt','r')
b1_list = b1.readlines()
b1_list = list(map(int,b1_list))

# ASSUMPTION: Heads favoured

counts = [0,0]
probs = []

doubt_index = []
surety_index = []

for i in b1_list:
    if i == 0:
        counts[0] += 1
    if i == 1:
        counts[1] += 1
    probs.append(counts[1]/(counts[0]+counts[1]))

doubt = False
surety_against = False

for i in range(0,100):

    if probs[i] <= 0.56 and probs[i] > 0.40:
        if doubt == False:
            doubt = True
            doubt_index.append(i+1)

    if probs[i] <= 0.40:
        surety_against = True
        surety_index.append(i+1)

if not doubt_index and not surety_index:
 print("\nHe is convinced that his choice is correct when the third
coin is used for the experiment.")
else:
 print("\nThe points at which he suspects of being wrong are: " +
str(doubt_index))
 print("\nThe points at which he is convinced that he might be wrong
are: " + str(surety_index))

print("\nThe probability of getting a 'Head' from the third coin is: "
+ str(probs[-1]) + "\n")

x_values = range(0 , 100)
y_values = probs

plt.scatter(x_values , y_values , s=20)

plt.xlabel('Tosses')
plt.ylabel('Experimental probability (of Head)')
```

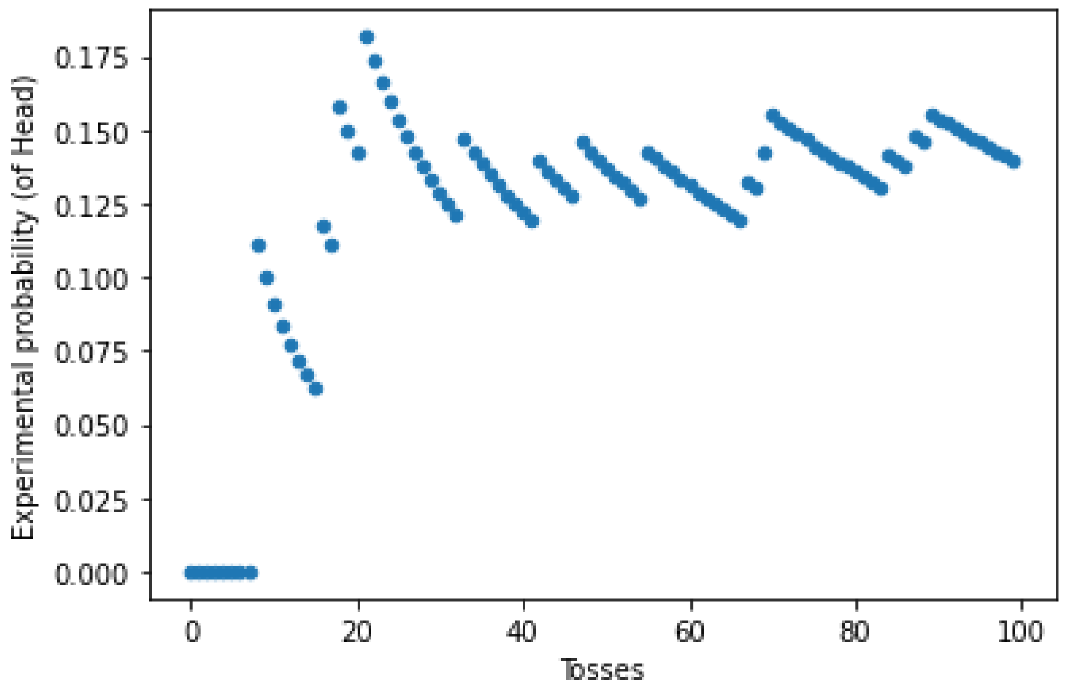
```
The points at which he suspects of being wrong are: [11]

The points at which he is convinced that he might be wrong are: []

The probability of getting a 'Head' from the third coin is: 0.63


Text(0, 0.5, 'Experimental probability (of Head)')
```



2(c)_hw1b4

```
b1=open('/content/hw1b4.txt','r')
b1_list = b1.readlines()
b1_list = list(map(int,b1_list))

# ASSUMPTION: Heads favoured

counts = [0,0]
probs = []

doubt_index = []
surety_index = []

for i in b1_list:
    if i == 0:
        counts[0] += 1
    if i == 1:
        counts[1] += 1
    probs.append(counts[1]/(counts[0]+counts[1]))

doubt = False
surety_against = False

for i in range(0,100):

    if probs[i] <= 0.56 and probs[i] > 0.40:
        if doubt == False:
            doubt = True
            doubt_index.append(i+1)

    if probs[i] <= 0.40:
        surety_against = True
        surety_index.append(i+1)

if not doubt_index and not surety_index:
 print("\nHe is convinced that his choice is correct when the first
coin is used for the experiment.")
else:
 print("\nThe points at which he suspects of being wrong are: " +
str(doubt_index))
 print("\nThe points at which he is convinced that he might be wrong
are: " + str(surety_index))

print("\nThe probability of getting a 'Head' from the fourth coin is:
" + str(probs[-1]) + "\n")

x_values = range(0 , 100)
y_values = probs

plt.scatter(x_values , y_values , s=20)

plt.xlabel('Tosses')
plt.ylabel('Experimental probability (of Head)')
```

```
The points at which he suspects of being wrong are: []

The points at which he is convinced that he might be wrong are: [1, 2,
3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72,
73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]

The probability of getting a 'Head' from the fourth coin is: 0.14


Text(0, 0.5, 'Experimental probability (of Head)')
```



2(a) : At any point, if the probability (of a Head) falls below 0.56 but is still larger than 0.40, he may begin to doubt his initial assumption.

2(b) : At any point, if the probability (of a Head) falls below 0.40, he may be convinced that his initial assumption was wrong.

2(c) : We apply the same boundary conditions as above for the (c) part also.

2(d) : At every point, we can check the difference between the number of tails and number of heads that have come up till that point. If it exceeds a particular fraction of the total number of tosses, we can say that the coin is not favourable to Head. This is also equivalent to setting up a limit as to how low the probability of a Head can be before Kohli begins to doubt his initial assumption.

Problem 3

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

heights = [161.8917578714209, 175.20557024887373, 164.2676553766112,
178.7222340341774,
178.00073820784604, 173.2772419681515, 160.59676770218624,
192.65051786762572,
165.16827896723976, 186.83322519192234, 158.3181312404598,
152.40314131522246,
176.96604318719778, 174.1476319971853, 165.64381551861507,
178.33112793512723,
167.63588343343207, 158.2518544667988,
162.39631147040325,164.93377167203346,
185.23966651457155, 174.00884801974885,
178.29163623554024,154.42607294132182,
152.02373388539078, 171.35451719883235, 172.46880217666438,
187.8259234866104,
165.8348672259168, 154.21715210274448, 183.99341334642781,
192.96899615398104,
193.24767230032984, 152.34085565439042, 177.86128609011809,
167.1786624519332,
183.32867104686477, 189.32630987297262, 171.8239506780645,
162.7533361998475,
181.8937717199068, 151.81355565372678, 175.70637064056112,
179.82951362312016,
179.39069884595247, 177.08397660941904,
182.1282175100166,175.2020315276609,
166.4430523185654, 157.22110851705304]

weights = [50.06513915417443, 57.08997118968728, 57.34038829783626,
56.88540143502565,
53.478696908960934, 56.768169670936345, 50.98936280545692,
62.35249986686582,
51.88319410967615, 52.674703814115304, 52.69023951205429,
51.27488019363186,
56.878634760999624, 60.283919884274155, 46.438146326369136,
57.504487839354574,
45.744829489560345, 51.17557752289022, 57.227062735277805,
53.41123363598972,
53.7655228010482, 48.77457857823918, 50.57295828987031,
49.109738459110076,
41.84216598976299, 55.871446827859636, 52.03636592100854,
63.228012473952845,
52.497642699508894, 43.42309786298475, 58.2807014841803,
69.4769468120694,
61.140432492527324, 51.89160705026108, 59.88218450922603,
55.475816112146205,
56.65606435614058, 62.99451248585423, 49.64889204896306,
54.28403891707874,
58.38375250582365, 46.20229138646405, 54.28590662636416,
54.51621196857059,
56.063543839297886, 63.33081820162695, 57.229870486518834,
58.40650863480111,
56.285176888103265, 55.73018000987019]

N = len(heights)

# p = sum(map(lambda i: i * i, heights))
# q = sum(map(lambda i: i * i, weights))
# Sum = [height * weight for (height,weight) in zip(heights ,
weights)]
# r = sum(map(lambda i: i * i, Sum))
```

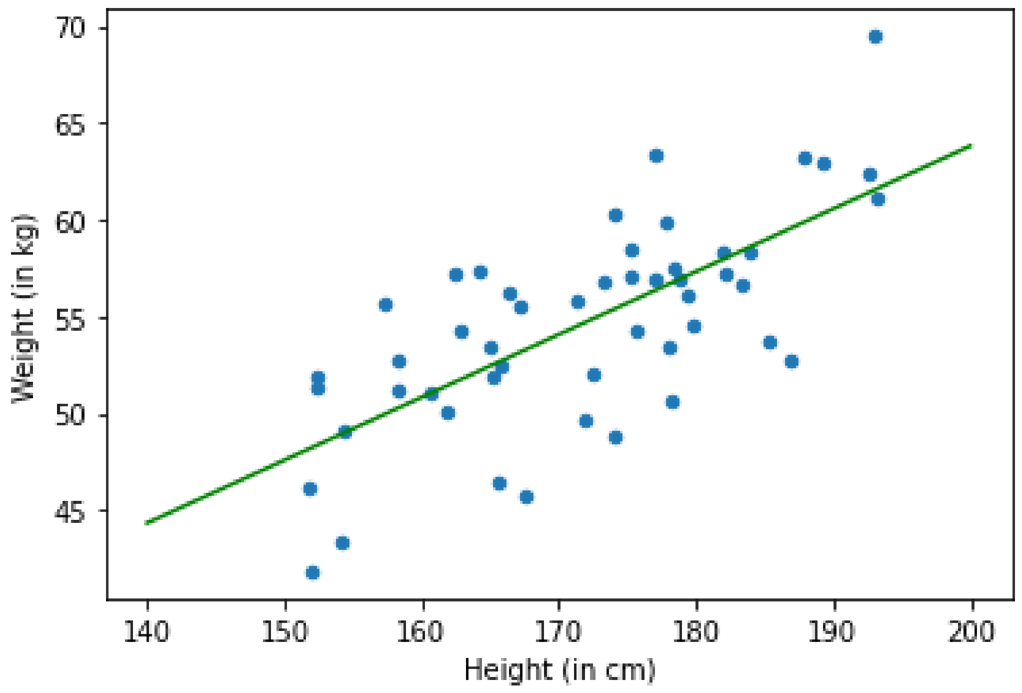
```python
s = sum(weights)

p = 0
for i in range (0 , N):
  p += heights[i]**2

q = sum(heights)

r = 0
for i in range (0 , N):
  r += heights[i] * weights[i]

t = 0
for i in range (0 , N):
  t += weights[i]**2

a = (N * r - q * s)/(N * p - q**2)

b = (p * s - q * r)/(N * p - q**2)

x_values = heights
y_values = weights

plt.scatter(x_values , y_values , s=20)

x = np.linspace(140,200,80)
y = a * x + b
plt.xlabel('Height (in cm)')
plt.ylabel('Weight (in kg)')
plt.plot(x, y, '-g', label='y=ax+b')

print("For 3(a):")
height = input("Please enter your height (in cm): ")
expected_weight = a * float(height) + b
print("\nYour expected weight is " + str(expected_weight) + " kg")

weight_variations = []

for i in range (0 , N):
  weight_variations.append(weights[i] - (a * heights[i] + b))
  max_var = max(weight_variations)
  min_var = min(weight_variations)

print("\nYour actual weight\x1B[3m might \x1B[0mlie between " +
str(expected_weight + max_var) +
      " kg and " + str(expected_weight + min_var) + " kg as per
statistical data available to us.")
```

```
h_1=155
weights_1 = [50.84083481424745, 50.014150611241, 47.434433994546275,
48.18509369914934,
49.47940437427491, 51.46395679542809, 50.92745554133917,
50.05313719206625,
44.32619446516681, 58.43082902641991, 51.78016307917405,
44.02198762936742,
50.72446115642888, 50.705517805027014, 57.59462421345118,
54.470110375852684,
53.22931789122948, 51.774125788427334, 54.474225400616,
48.6041391956155,
45.39272640839777, 44.650940250705666, 42.69887479467695,
47.003806810494126,
48.146952021872984]

h_2=165
weights_2 = [56.09178233874335, 53.45028077511345, 53.921575413701156,
48.85934258973829,
56.95307675637719, 58.34536050529028, 58.38396662842817,
59.53843135500536,
63.86284924117021, 50.37873251772672, 53.542546379774144,
55.65272988547241,
56.19834658523187, 49.931028628110056, 49.067210771486366,
57.081250056851644,
52.941790056077416, 50.11083820970033, 52.196245895572396,
52.86016951388828,
52.997389807842836, 50.77474626914866, 47.91793610117185,
51.317062516525056,
53.34045945347973]

h_3=175
weights_3 = [58.7729713699485, 57.52174253372618, 46.63896463839615,
60.08699337124898,
54.97048946487911, 50.89015721919731, 51.16430051196914,
60.44417409273054,
47.330206543798724, 58.62280875417217, 51.94075563649851,
53.85545926849789,
55.19225332108357, 57.316954222269885, 61.96906592544718,
64.45515173613902,
58.145794819284355, 52.00153353928704, 53.687784624881914,
54.467373628898336,
54.83437188336155, 58.39119923096617, 52.72810010849628,
53.5992234950328,
61.30297365392382]

n = len(weights_1)

w_1_expected = a * h_1 + b
w_2_expected = a * h_2 + b
w_3_expected = a * h_3 + b
```

```
x = 0
for i in range (0 , n):
  x += weights_1[i]**2

y = 0
for i in range (0 , n):
  y += weights_2[i]**2

z = 0
for i in range (0 , n):
  z += weights_3[i]**2

mean_error_1 = ((n * w_1_expected) - sum(weights_1))/n
mean_error_2 = ((n * w_2_expected) - sum(weights_2))/n
mean_error_3 = ((n * w_3_expected) - sum(weights_3))/n

std_deviation_1 = math.sqrt(x/n - (sum(weights_1)/n)**2)
std_deviation_2 = math.sqrt(y/n - (sum(weights_2)/n)**2)
std_deviation_3 = math.sqrt(z/n - (sum(weights_3)/n)**2)

print("\nFor 3(b):")
print("The average of the errors in weight for a height of " +
str(h_1) + " cm is given by: "
+ str(mean_error_1) + "kg and the standard deviation in the errors is
given by: " +
str(std_deviation_1) + " kg")

print("\nThe average of the errors in weight for a height of " +
str(h_2) + " cm is given by: "
+ str(mean_error_2) + " kg and the standard deviation in the errors is
given by: " +
str(std_deviation_2) + " kg")

print("\nThe average of the errors in weight for a height of " +
str(h_3) + " cm is given by: "
+ str(mean_error_3) + " kg and the standard deviation in the errors is
given by: " +
str(std_deviation_3) + " kg\n")
```

```
For 3(a):
Please enter your height (in cm): 180

Your expected weight is 57.31094492839616 kg

Your actual weight might lie between 65.2688418199921 kg and
49.75666664554784 kg as per statistical data available to us.

For 3(b):
```

```
The average of the errors in weight for a height of 155 cm is given
by: -0.6580092608288851kg and the standard deviation in the errors is
given by: 3.925858798008328 kg

The average of the errors in weight for a height of 165 cm is given
by: -1.3847743951587654 kg and the standard deviation in the errors is
given by: 3.758979908854162 kg

The average of the errors in weight for a height of 175 cm is given
by: 0.07534165346747614 kg and the standard deviation in the errors is
given by: 4.351940165945892 kg
```



3(a) : Based on the linear graph, we can find the maximum and minimum possible variations in the weight of students from the theoretical weight. Given this, we can say that the difference between true weight and actual weight *might* lie from -7.55427 kg to 7.95789 kg. Note that this is purely a statistical value and *need not* hold true at all in many circumstances.

3(b) : We would want the average of the errors $\frac{\sum_{i=1}^{N}(y_i - \hat{y})}{N}$ to be equal to zero. If so happens, then we can say that over a large number of observations recorded (of height), the average weight would tend to be nearly equal to the expected weight from the linear model. We would want the standard deviation to be at its minimum possible value. The standard deviation in this case will be given by (after taking average error to be zero) -

$$\sqrt{\frac{\sum_{i=1}^{N}\left(y_i - x_i\right)^2}{N} - b^2}.$$