Analytics and Systems of Big Data Practice

Problem Set - 1

Bhaavanaa Thumu

CED17I021

Problem 1

Problem statement :

Given the following setup {Class, Tally score, Frequency}, develop an application that generates the table shown; (you can populate the relevant data; minimum data size: 50 records). The table is only an illustration for a data of color scores, you are free to test the application over any data set with the application generating the tally and frequency scores.

Logic used :

Initialize 50 colors and generate their respective frequencies randomly. For getting the tally of each frequency, iterate over (1, freq) using j.

```
if(j%5 != 0):
        add '|' to the tally string;
else:
        add '\' to the tally string;
```

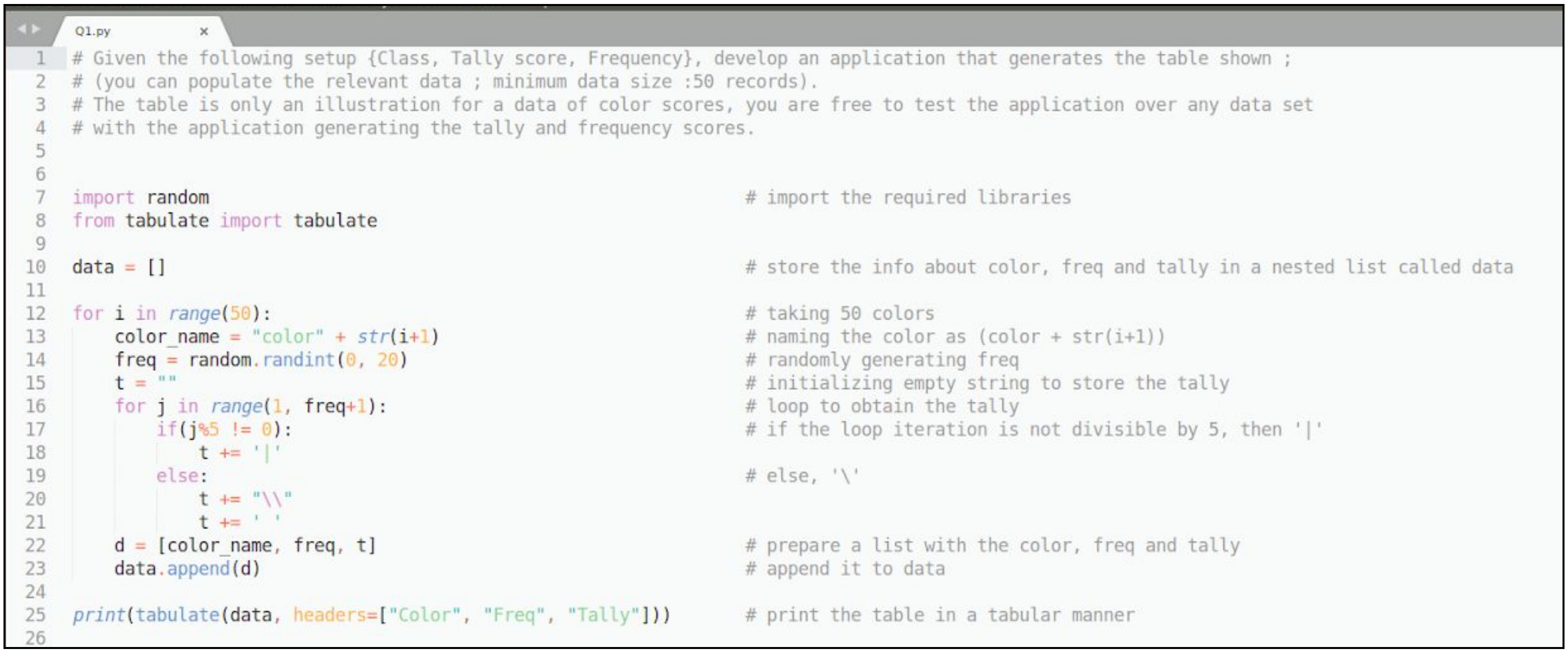
Tabulate and print the table with color, freq and tally.

Packages used :

random - to generate the random frequencies.

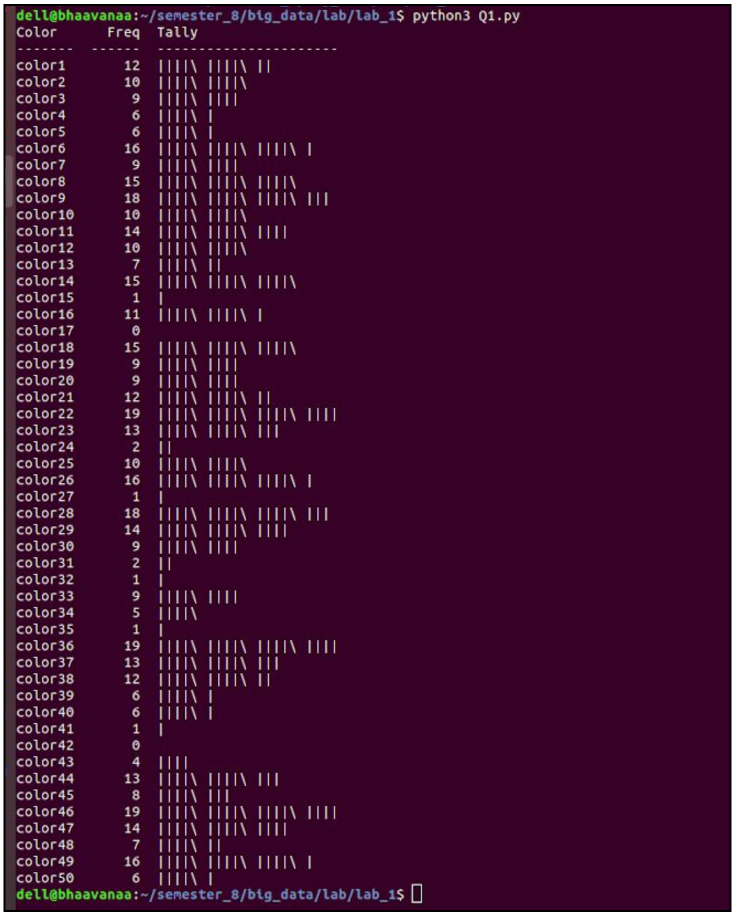
tabulate - to tabulate the contents - color, frequency and tally.

Code snippets :



```
# Given the following setup {Class, Tally score, Frequency}, develop an application that generates the table shown ;
# (you can populate the relevant data ; minimum data size :50 records).
# The table is only an illustration for a data of color scores, you are free to test the application over any data set
# with the application generating the tally and frequency scores.


import random                                                   # import the required libraries
from tabulate import tabulate

data = []                                                       # store the info about color, freq and tally in a nested list called data

for i in range(50):                                             # taking 50 colors
    color_name = "color" + str(i+1)                             # naming the color as (color + str(i+1))
    freq = random.randint(0, 20)                                # randomly generating freq
    t = ""                                                      # initializing empty string to store the tally
    for j in range(1, freq+1):                                  # loop to obtain the tally
        if(j%5 != 0):                                           # if the loop iteration is not divisible by 5, then '|'
            t += '|'
        else:                                                   # else, '\'
            t += "\\"
            t += ' '
    d = [color_name, freq, t]                                   # prepare a list with the color, freq and tally
    data.append(d)                                              # append it to data

print(tabulate(data, headers=["Color", "Freq", "Tally"]))       # print the table in a tabular manner

```

Detailed output :



```
dell@bhaavanaa:~/semester_8/big_data/lab/lab_1$ python3 Q1.py
Color      Freq  Tally
-------  ------  ----------------------
color1       12  ||||\ ||||\ ||
color2       10  ||||\ ||||\
color3        9  ||||\ ||||
color4        6  ||||\ |
color5        6  ||||\ |
color6       16  ||||\ ||||\ ||||\ |
color7        9  ||||\ ||||
color8       15  ||||\ ||||\ ||||\
color9       18  ||||\ ||||\ ||||\ |||
color10      10  ||||\ ||||\
color11      14  ||||\ ||||\ ||||
color12      10  ||||\ ||||\
color13       7  ||||\ ||
color14      15  ||||\ ||||\ ||||\
color15       1  |
color16      11  ||||\ ||||\ |
color17       0
color18      15  ||||\ ||||\ ||||\
color19       9  ||||\ ||||
color20       9  ||||\ ||||
color21      12  ||||\ ||||\ ||
color22      19  ||||\ ||||\ ||||\ ||||
color23      13  ||||\ ||||\ |||
color24       2  ||
color25      10  ||||\ ||||\
color26      16  ||||\ ||||\ ||||\ |
color27       1  |
color28      18  ||||\ ||||\ ||||\ |||
color29      14  ||||\ ||||\ ||||
color30       9  ||||\ ||||
color31       2  ||
color32       1  |
color33       9  ||||\ ||||
color34       5  ||||\
color35       1  |
color36      19  ||||\ ||||\ ||||\ ||||
color37      13  ||||\ ||||\ |||
color38      12  ||||\ ||||\ ||
color39       6  ||||\ |
color40       6  ||||\ |
color41       1  |
color42       0
color43       4  ||||
color44      13  ||||\ ||||\ |||
color45       8  ||||\ |||
color46      19  ||||\ ||||\ ||||\ ||||
color47      14  ||||\ ||||\ ||||
color48       7  ||||\ ||
color49      16  ||||\ ||||\ ||||\ |
color50       6  ||||\ |
dell@bhaavanaa:~/semester_8/big_data/lab/lab_1$
```

The output is the table consisting of the color, frequency and the tally.

Problem 2

Problem statement :

In a class of 18 students, assume marks distribution in an exam are as follows. Let the roll numbers start with CSE20D01 and all the odd roll numbers secure marks as follows: 25+((i+7)%10) and even roll numbers : 25+((i+8)%10). Develop an application that sets up the data and calculate the mean and median for the marks obtained using the platform support.

Logic used :

Initialize the roll nos from 1 to 18. For getting the marks obtained by each student, iterate over (1, 18) using i.

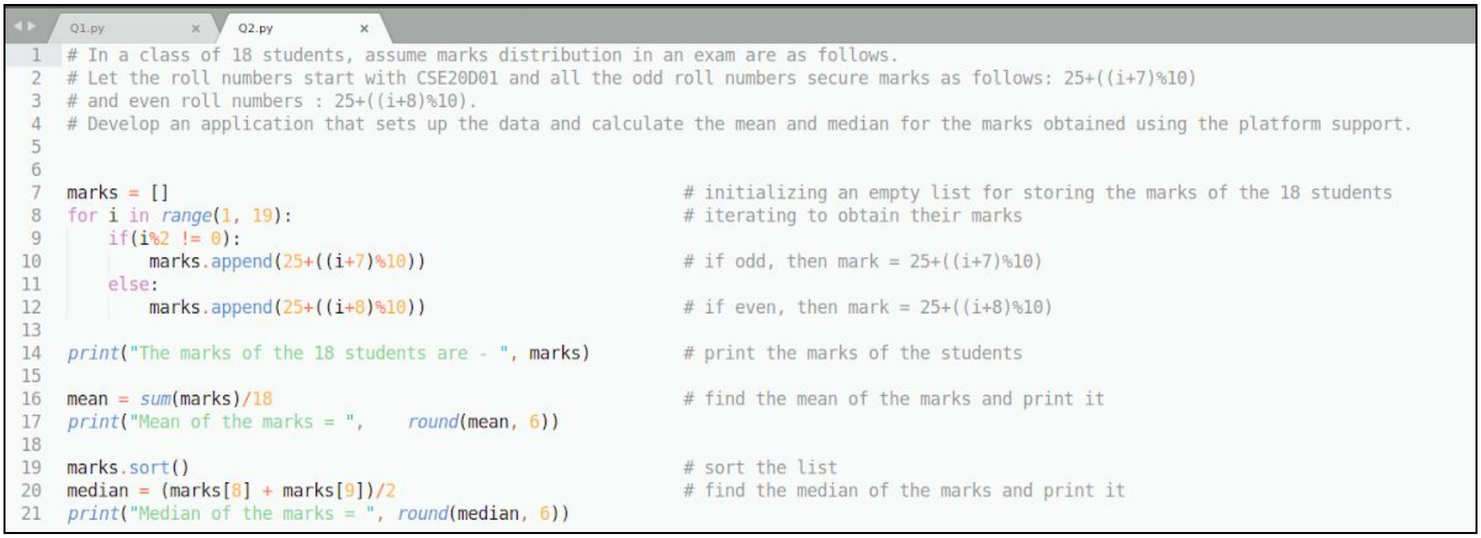
```
if(i%2 != 0):
        marks for that student = 25+((i+7)%10);
else:
        marks for that student = 25+((i+8)%10);
```

Next, print the marks obtained by the 18 students, along with the mean=(total sum of marks/18) and the median, obtained by sorting and finding the average of the middle two elements, since the total number of students is 18 and it is an even number.

Packages used :

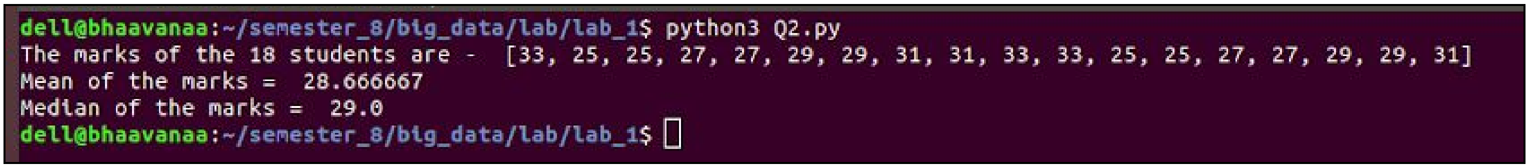
None.

Code snippets :



```python
# In a class of 18 students, assume marks distribution in an exam are as follows.
# Let the roll numbers start with CSE20D01 and all the odd roll numbers secure marks as follows: 25+((i+7)%10)
# and even roll numbers : 25+((i+8)%10).
# Develop an application that sets up the data and calculate the mean and median for the marks obtained using the platform support.


marks = []                                                  # initializing an empty list for storing the marks of the 18 students
for i in range(1, 19):                                      # iterating to obtain their marks
    if(i%2 != 0):
        marks.append(25+((i+7)%10))                         # if odd, then mark = 25+((i+7)%10)
    else:
        marks.append(25+((i+8)%10))                         # if even, then mark = 25+((i+8)%10)

print("The marks of the 18 students are - ", marks)         # print the marks of the students

mean = sum(marks)/18                                        # find the mean of the marks and print it
print("Mean of the marks = ",    round(mean, 6))

marks.sort()                                                # sort the list
median = (marks[8] + marks[9])/2                            # find the median of the marks and print it
print("Median of the marks = ", round(median, 6))
```

Detailed output :



```
dell@bhaavanaa:~/semester_8/big_data/lab/lab_1$ python3 Q2.py
The marks of the 18 students are -  [33, 25, 25, 27, 27, 29, 29, 31, 31, 33, 33, 25, 25, 27, 27, 29, 29, 31]
Mean of the marks =  28.666667
Median of the marks =  29.0
dell@bhaavanaa:~/semester_8/big_data/lab/lab_1$ 
```

The output consists of the marks of the 18 students along with their mean and median.

Problem 3

Problem statement :

For a sample space of 20 elements, the values are fitted to the line Y=2X+3, X>5. Develop an application that sets up the data and computes the standard deviation of this sample space. (use a random number generator supported in your development platform to generate values of X).

Logic used :

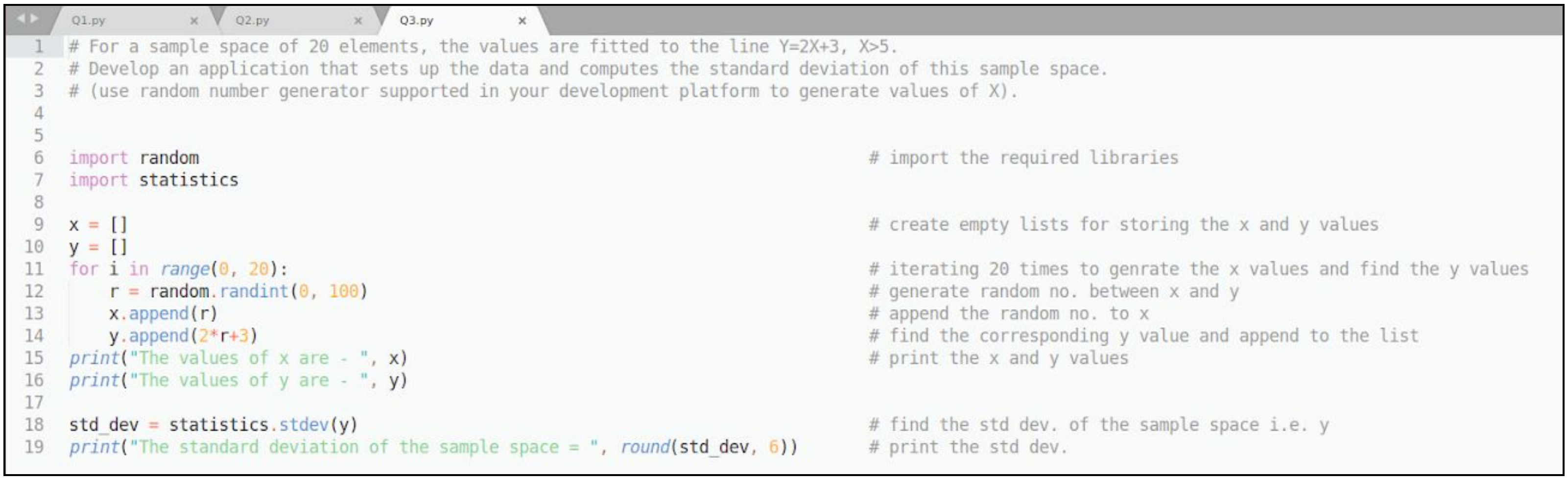
Randomly generate 20 values of x and obtain the respective values of (using y = 2x + 3). Then, compute the standard deviation of the y values obtained. Print the x values, y values and standard deviation obtained for the y values.

Packages used :

random - to generate the random numbers.

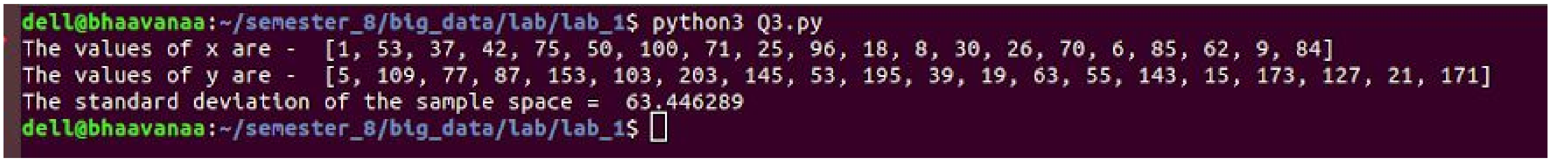
statistics - to compute the standard deviation.

Code snippets :



```
# For a sample space of 20 elements, the values are fitted to the line Y=2X+3, X>5.
# Develop an application that sets up the data and computes the standard deviation of this sample space.
# (use random number generator supported in your development platform to generate values of X).


import random                                                                   # import the required libraries
import statistics

x = []                                                                          # create empty lists for storing the x and y values
y = []
for i in range(0, 20):                                                          # iterating 20 times to genrate the x values and find the y values
    r = random.randint(0, 100)                                                  # generate random no. between x and y
    x.append(r)                                                                 # append the random no. to x
    y.append(2*r+3)                                                             # find the corresponding y value and append to the list
print("The values of x are - ", x)                                              # print the x and y values
print("The values of y are - ", y)

std_dev = statistics.stdev(y)                                                   # find the std dev. of the sample space i.e. y
print("The standard deviation of the sample space = ", round(std_dev, 6))       # print the std dev.
```

Detailed output :



```
dell@bhaavanaa:~/semester_8/big_data/lab/lab_1$ python3 Q3.py
The values of x are -  [1, 53, 37, 42, 75, 50, 100, 71, 25, 96, 18, 8, 30, 26, 70, 6, 85, 62, 9, 84]
The values of y are -  [5, 109, 77, 87, 153, 103, 203, 145, 53, 195, 39, 19, 63, 55, 143, 15, 173, 127, 21, 171]
The standard deviation of the sample space =  63.446289
dell@bhaavanaa:~/semester_8/big_data/lab/lab_1$ 
```

The output consists of the x and the respective y values, along with the standard deviation of the y values.

Problem 4

Problem statement :

For a given data of heights of a class, the heights of 15 students are recorded as 167.65, 167, 172, 175, 165, 167, 168, 167, 167.3, 170, 167.5, 170, 167, 169, and 172. Develop an application that computes; explore if there are any packages supported in your platform that depicts these measures / their calculation of central tendency in a visual form for ease of understanding.

a. Mean height of the student

b. Median and Mode of the sample space

c. Standard deviation

d. Measure of skewness. [(Mean-Mode)/standard deviation]

Logic used :

Compute the mean height, median, mode and standard deviation using the statistics library. Calculate the skewness using the formula, skewness = [(Mean-Mode)/standard deviation]. Plot the graph of the heights, and display the median, mean and mode using the lines on the graph.

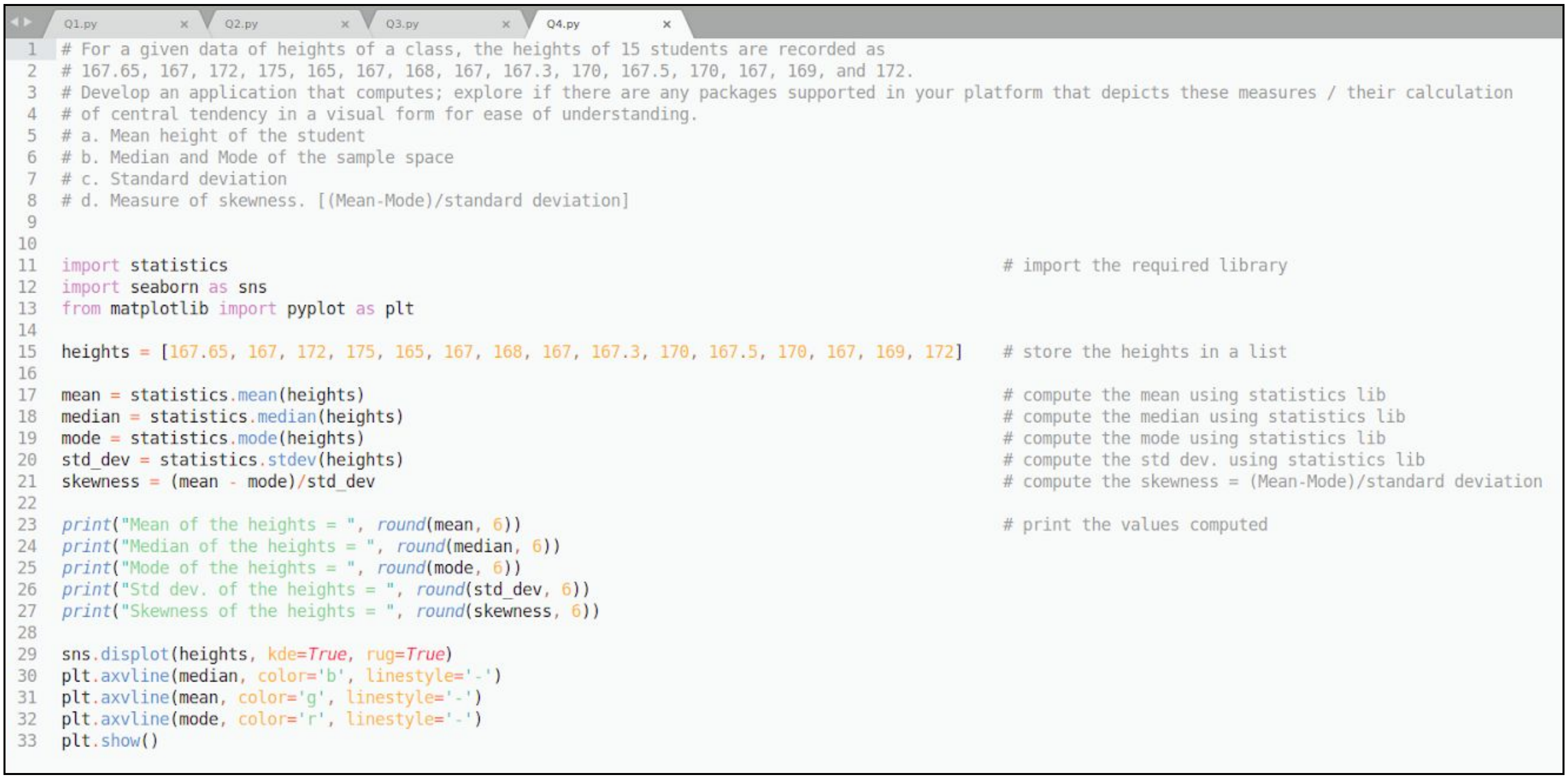
Packages used :

statistics - for obtaining the mean, median, mode, etc.

seaborn - helps display a line on the plot.

matplotlib.pyplot - for plotting the pie chart and bar graph.

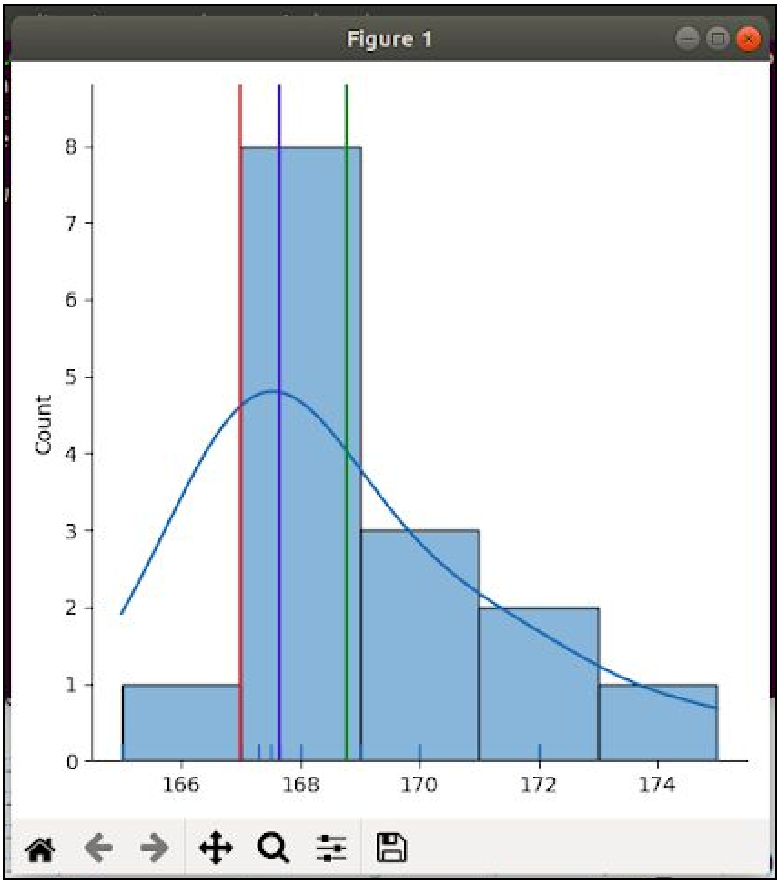
Code snippets :

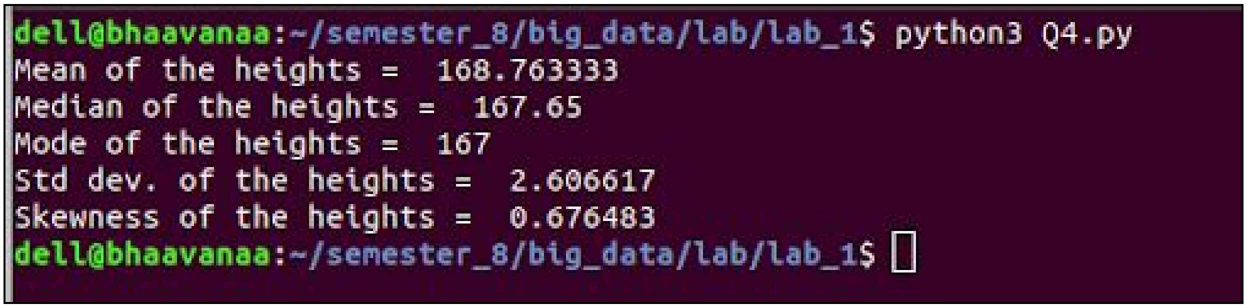


```python
# For a given data of heights of a class, the heights of 15 students are recorded as
# 167.65, 167, 172, 175, 165, 167, 168, 167, 167.3, 170, 167.5, 170, 167, 169, and 172.
# Develop an application that computes; explore if there are any packages supported in your platform that depicts these measures / their calculation
# of central tendency in a visual form for ease of understanding.
# a. Mean height of the student
# b. Median and Mode of the sample space
# c. Standard deviation
# d. Measure of skewness. [(Mean-Mode)/standard deviation]


import statistics                                                                               # import the required library
import seaborn as sns
from matplotlib import pyplot as plt

heights = [167.65, 167, 172, 175, 165, 167, 168, 167, 167.3, 170, 167.5, 170, 167, 169, 172]    # store the heights in a list

mean = statistics.mean(heights)                                                                 # compute the mean using statistics lib
median = statistics.median(heights)                                                             # compute the median using statistics lib
mode = statistics.mode(heights)                                                                 # compute the mode using statistics lib
std_dev = statistics.stdev(heights)                                                             # compute the std dev. using statistics lib
skewness = (mean - mode)/std_dev                                                                # compute the skewness = (Mean-Mode)/standard deviation

print("Mean of the heights = ", round(mean, 6))                                                 # print the values computed
print("Median of the heights = ", round(median, 6))
print("Mode of the heights = ", round(mode, 6))
print("Std dev. of the heights = ", round(std_dev, 6))
print("Skewness of the heights = ", round(skewness, 6))

sns.displot(heights, kde=True, rug=True)
plt.axvline(median, color='b', linestyle='-')
plt.axvline(mean, color='g', linestyle='-')
plt.axvline(mode, color='r', linestyle='-')
plt.show()
```

Detailed output :

The output consists of the x and the respective y values, along with the standard deviation of the y values.





```
dell@bhaavanaa:~/semester_8/big_data/lab/lab_1$ python3 Q4.py
Mean of the heights =  168.763333
Median of the heights =  167.65
Mode of the heights =  167
Std dev. of the heights =  2.606617
Skewness of the heights =  0.676483
dell@bhaavanaa:~/semester_8/big_data/lab/lab_1$ 
```

Problem 5

Problem statement :

In the Analytics and Systems of Big Data course, for a class of 100 students, around 31 students secured ‘S’ grade, 29 secured ‘B’ grade, 25 ‘C’ grades, and rest of them secured ‘D’ grades. If the range of each grade is 15 marks. (S for 85 to 100 marks, A for 70 to 85 ...). Develop an application that represents the above data : using Pie and Bar graphs.

Logic used :

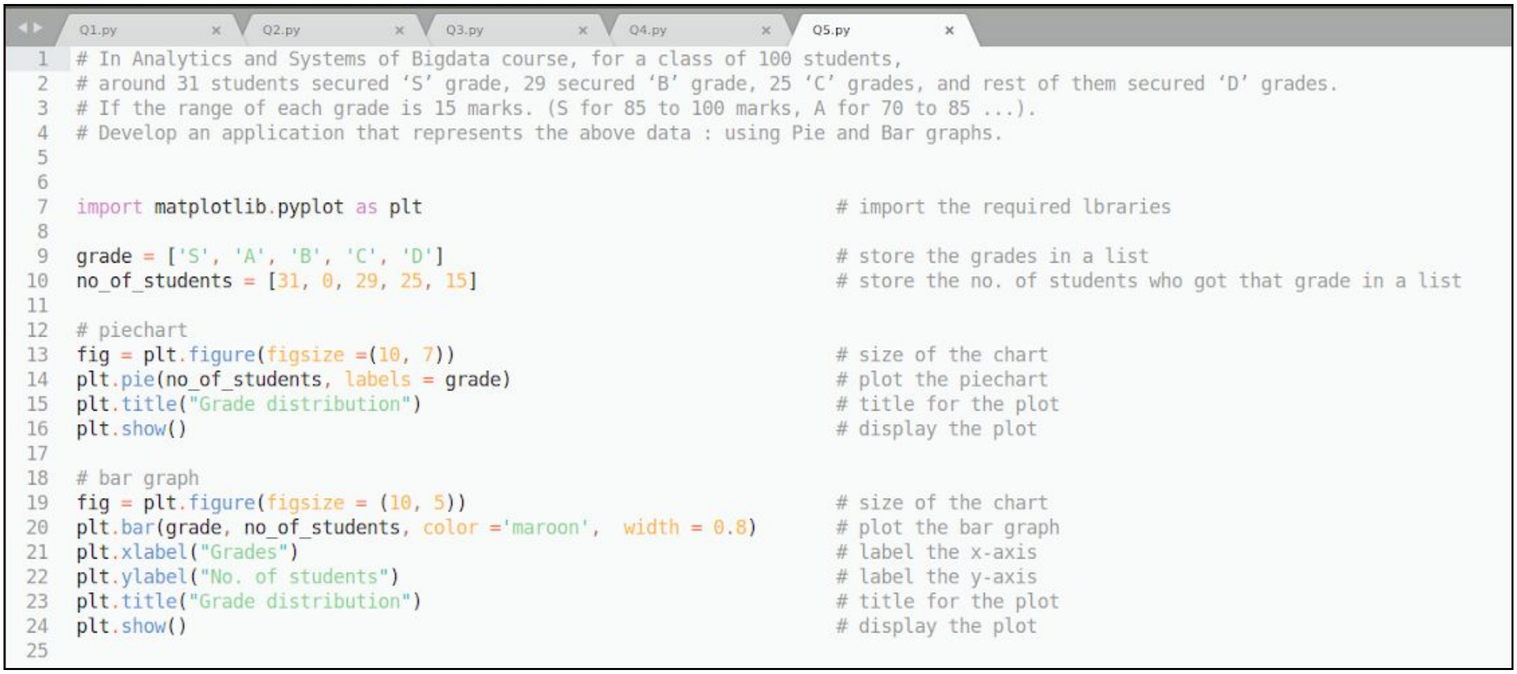
Given the grades and the number of students who obtained each grade, we can plot the pie chart and the bar graph using the matplotlib library.

Grade	S	A	B	C	D
No. of students	31	0	29	25	15

Packages used :

matplotlib.pyplot - for plotting the pie chart and bar graph.

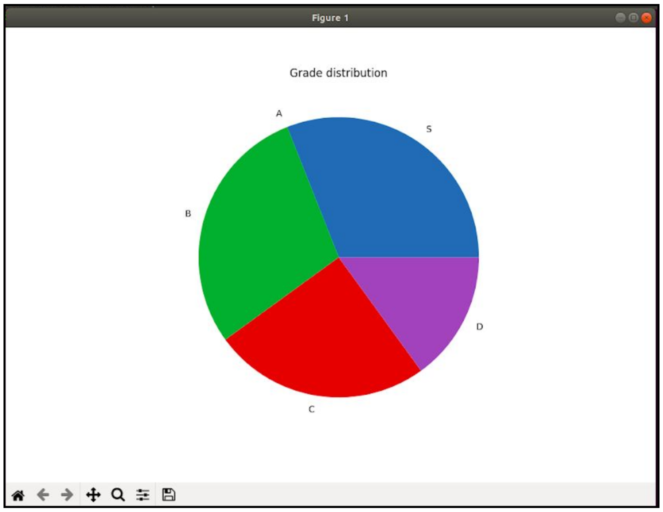
Code snippets :

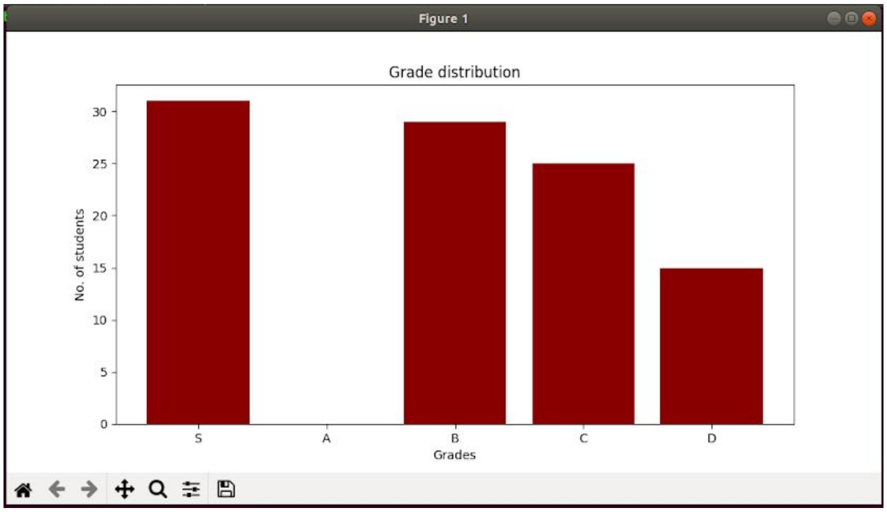


```
# In Analytics and Systems of Bigdata course, for a class of 100 students,
# around 31 students secured ‘S’ grade, 29 secured ‘B’ grade, 25 ‘C’ grades, and rest of them secured ‘D’ grades.
# If the range of each grade is 15 marks. (S for 85 to 100 marks, A for 70 to 85 ...).
# Develop an application that represents the above data : using Pie and Bar graphs.


import matplotlib.pyplot as plt                                     # import the required lbraries

grade = ['S', 'A', 'B', 'C', 'D']                                   # store the grades in a list
no_of_students = [31, 0, 29, 25, 15]                                # store the no. of students who got that grade in a list

# piechart
fig = plt.figure(figsize =(10, 7))                                  # size of the chart
plt.pie(no_of_students, labels = grade)                             # plot the piechart
plt.title("Grade distribution")                                     # title for the plot
plt.show()                                                          # display the plot

# bar graph
fig = plt.figure(figsize = (10, 5))                                 # size of the chart
plt.bar(grade, no_of_students, color ='maroon',  width = 0.8)       # plot the bar graph
plt.xlabel("Grades")                                                # label the x-axis
plt.ylabel("No. of students")                                       # label the y-axis
plt.title("Grade distribution")                                     # title for the plot
plt.show()                                                          # display the plot

```

Detailed output :

The output consists of the pie chart and the bar graph, given the grades and the number of students who obtained each grade.





Problem 6

Problem statement :

On a given day (average basis), a student is observed to spend 33% of time in studying, 30% in sleeping, 18% in playing, 5% for hobby activities, and rest for spending with friends and family. Plot a pie chart showing his daily activities.

Logic used :

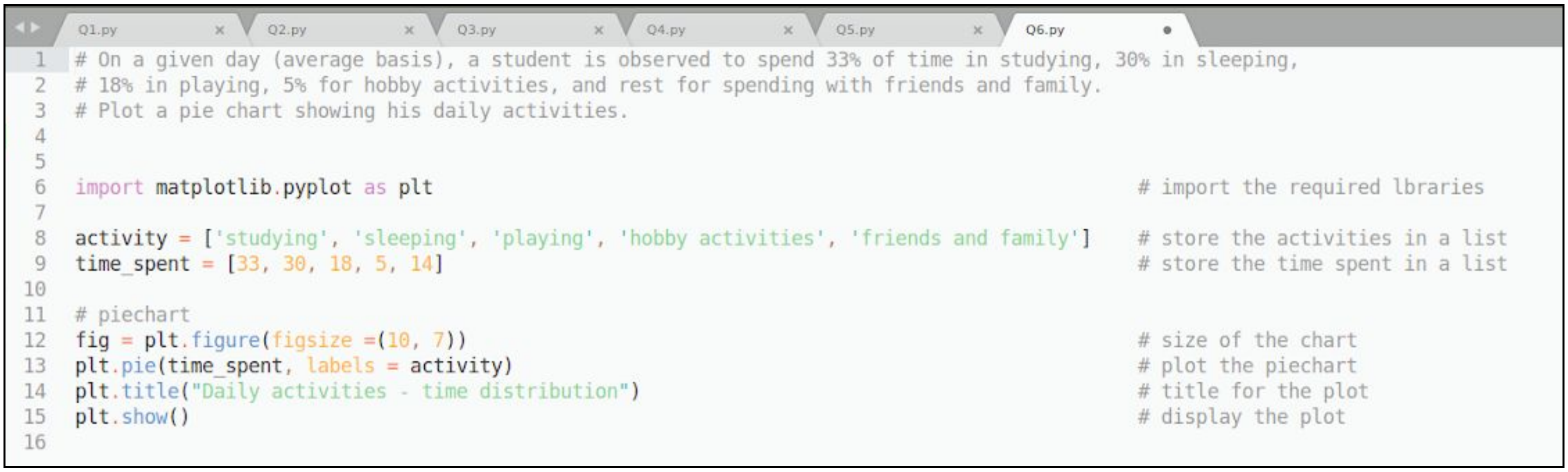
Given the daily activities and the amount of time spent on that activity, we can plot the pie chart using the matplotlib library.

Activity	studying	sleeping	playing	hobby activities	friends and family
Time spent	31	0	29	25	15

Packages used :

matplotlib.pyplot - for plotting the pie chart.

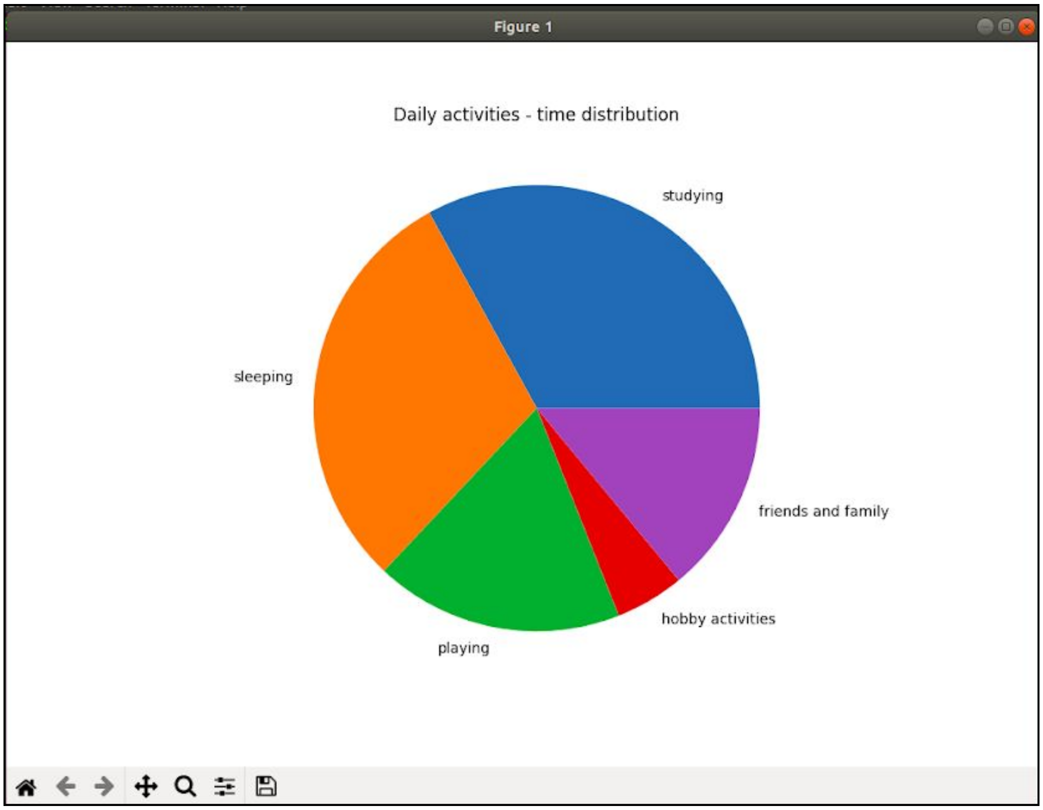
Code snippets :



```python
# On a given day (average basis), a student is observed to spend 33% of time in studying, 30% in sleeping,
# 18% in playing, 5% for hobby activities, and rest for spending with friends and family.
# Plot a pie chart showing his daily activities.


import matplotlib.pyplot as plt                                                            # import the required lbraries

activity = ['studying', 'sleeping', 'playing', 'hobby activities', 'friends and family']   # store the activities in a list
time_spent = [33, 30, 18, 5, 14]                                                           # store the time spent in a list

# piechart
fig = plt.figure(figsize =(10, 7))                                                         # size of the chart
plt.pie(time_spent, labels = activity)                                                     # plot the piechart
plt.title("Daily activities - time distribution")                                          # title for the plot
plt.show()                                                                                 # display the plot

```

Detailed output :

The output consists of the pie chart, given the activities and the percentage of time spent on each activity.



Problem 7

Problem statement :

Develop an application (absolute grader) that accepts marks scored by 20 students in ASBD course (as a split up of three : Mid Sem (30), End Sem(50) and Assignments(20). Compute the total and use it to grade the students following absolute grading : >=90 – S ; >=80 – A and so on till D. Compute the Class average for total marks in the course and 50% of class average would be fixed as the cut off for E. Generate a frequency table for the grades as well (Table displaying the grades and counts of them).

Logic used :

Iterate over (1, 20) for the students, and generate a random midsem score (out of 30), random endsem score (out of 50), and random assignment score (out of 20). We must then find the total score of each student, and find the class average. Then, set the range for grade E, i.e. 50% of class average. Since we have all the ranges for the grades, we can find the grade obtained by each student. Iterate over the total marks of each student, and assign the grade using the following conditions.

```
if(total_marks >= 90):
        grade = S;
elif(total_marks >= 80):
        grade = A;
elif(total_marks >= 70):
        grade = B;
```

```
elif(total_marks >= 60):
        grade = C;
elif(total_marks >= 50):
        grade = D;
elif(total_marks >= class_avg/2):
        grade = E;
else:
        grade = U;
```

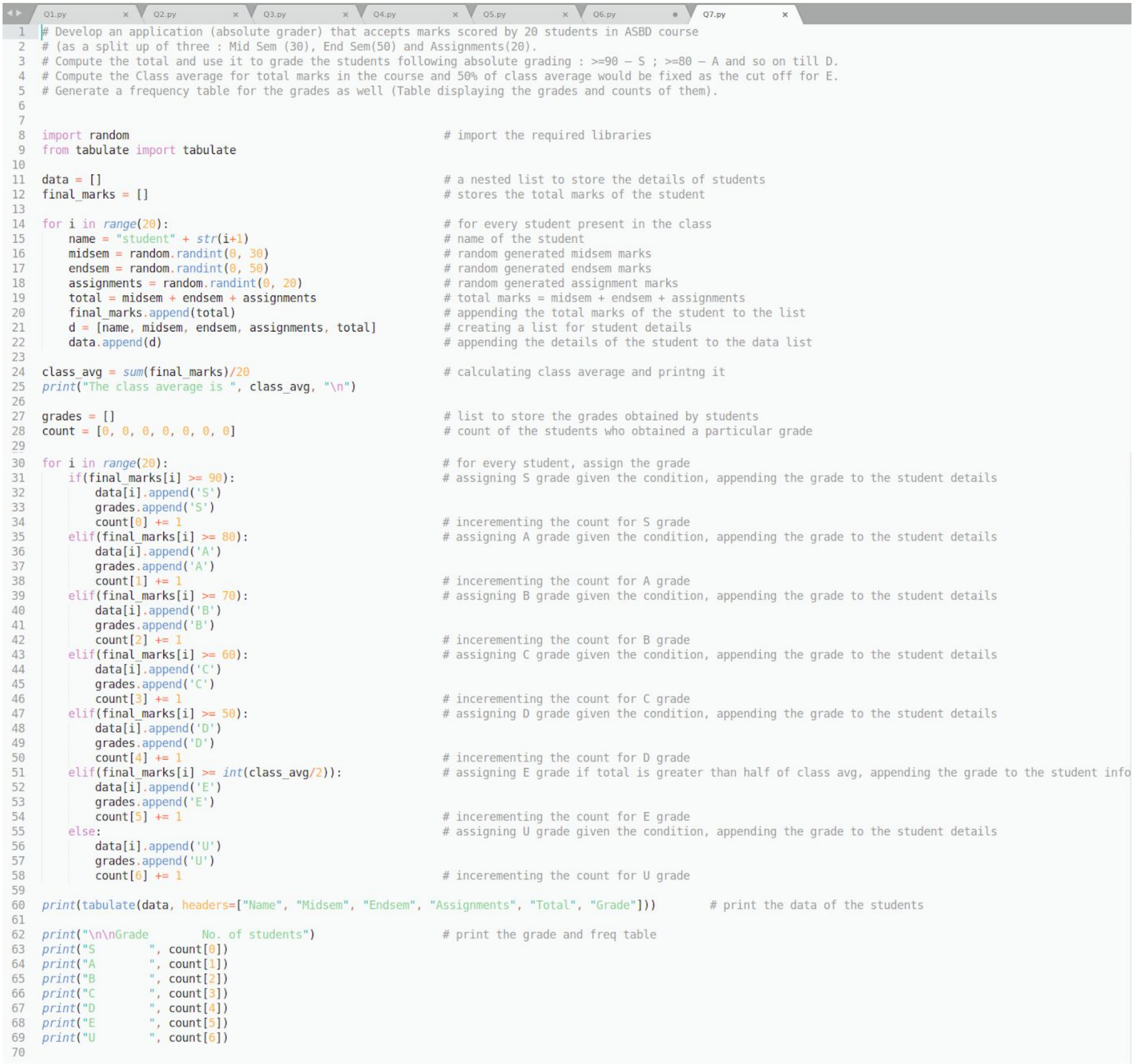
We must also keep track of the number of students who got a particular grade. We then print the marks table and the frequency table for the grade.

Packages used :

random - to generate the random frequencies.

tabulate - to tabulate the contents - color, frequency and tally.

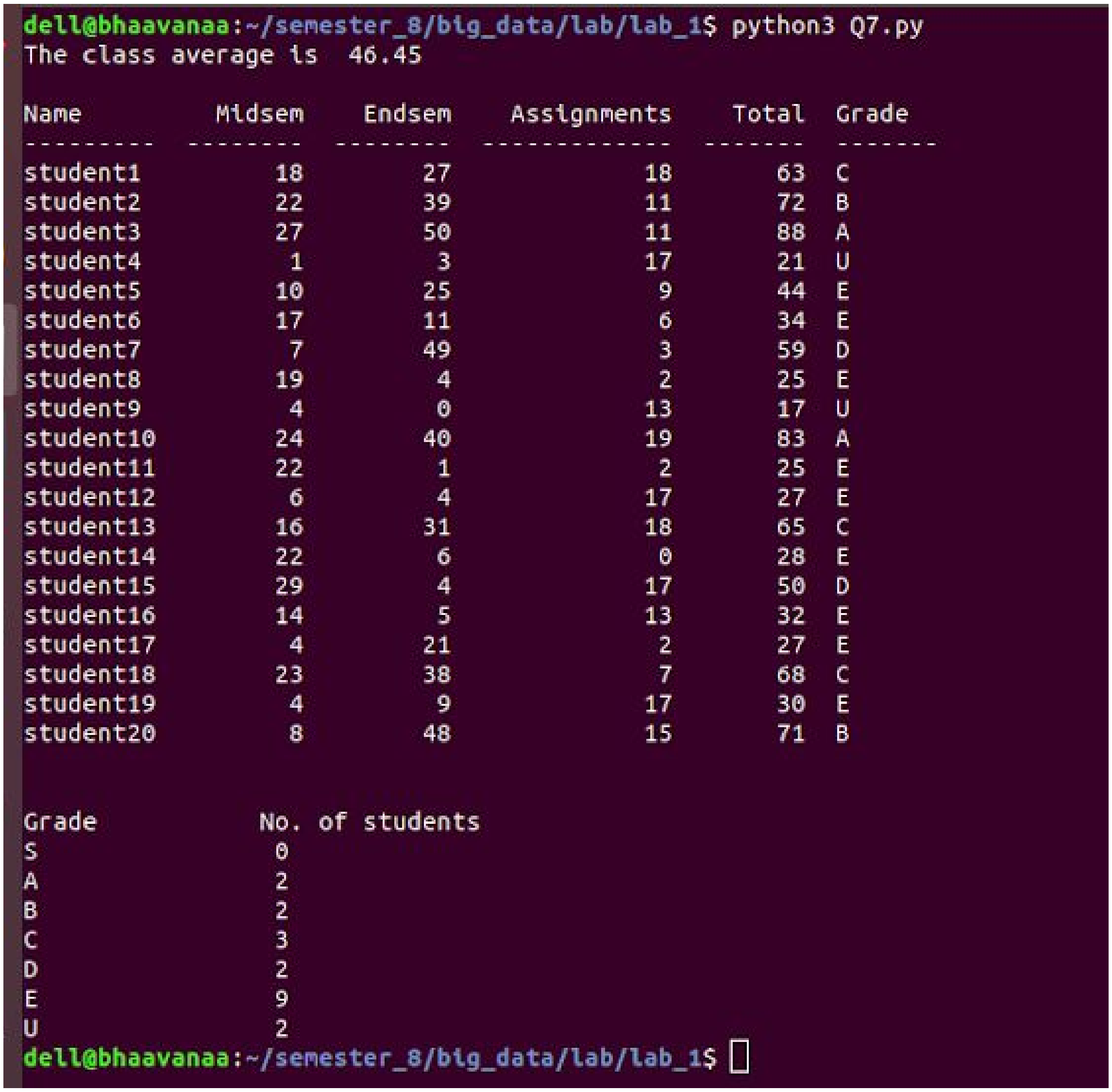
Code snippets :



```python
# Develop an application (absolute grader) that accepts marks scored by 20 students in ASBD course
# (as a split up of three : Mid Sem (30), End Sem(50) and Assignments(20).
# Compute the total and use it to grade the students following absolute grading : >=90 – S ; >=80 – A and so on till D.
# Compute the Class average for total marks in the course and 50% of class average would be fixed as the cut off for E.
# Generate a frequency table for the grades as well (Table displaying the grades and counts of them).


import random                                               # import the required libraries
from tabulate import tabulate

data = []                                                   # a nested list to store the details of students
final_marks = []                                            # stores the total marks of the student

for i in range(20):                                         # for every student present in the class
    name = "student" + str(i+1)                             # name of the student
    midsem = random.randint(0, 30)                          # random generated midsem marks
    endsem = random.randint(0, 50)                          # random generated endsem marks
    assignments = random.randint(0, 20)                     # random generated assignment marks
    total = midsem + endsem + assignments                   # total marks = midsem + endsem + assignments
    final_marks.append(total)                               # appending the total marks of the student to the list
    d = [name, midsem, endsem, assignments, total]          # creating a list for student details
    data.append(d)                                          # appending the details of the student to the data list

class_avg = sum(final_marks)/20                             # calculating class average and printng it
print("The class average is ", class_avg, "\n")

grades = []                                                 # list to store the grades obtained by students
count = [0, 0, 0, 0, 0, 0, 0]                               # count of the students who obtained a particular grade

for i in range(20):                                         # for every student, assign the grade
    if(final_marks[i] >= 90):                               # assigning S grade given the condition, appending the grade to the student details
        data[i].append('S')
        grades.append('S')
        count[0] += 1                                       # incerementing the count for S grade
    elif(final_marks[i] >= 80):                             # assigning A grade given the condition, appending the grade to the student details
        data[i].append('A')
        grades.append('A')
        count[1] += 1                                       # incerementing the count for A grade
    elif(final_marks[i] >= 70):                             # assigning B grade given the condition, appending the grade to the student details
        data[i].append('B')
        grades.append('B')
        count[2] += 1                                       # incerementing the count for B grade
    elif(final_marks[i] >= 60):                             # assigning C grade given the condition, appending the grade to the student details
        data[i].append('C')
        grades.append('C')
        count[3] += 1                                       # incerementing the count for C grade
    elif(final_marks[i] >= 50):                             # assigning D grade given the condition, appending the grade to the student details
        data[i].append('D')
        grades.append('D')
        count[4] += 1                                       # incerementing the count for D grade
    elif(final_marks[i] >= int(class_avg/2)):               # assigning E grade if total is greater than half of class avg, appending the grade to the student info
        data[i].append('E')
        grades.append('E')
        count[5] += 1                                       # incerementing the count for E grade
    else:                                                   # assigning U grade given the condition, appending the grade to the student details
        data[i].append('U')
        grades.append('U')
        count[6] += 1                                       # incerementing the count for U grade

print(tabulate(data, headers=["Name", "Midsem", "Endsem", "Assignments", "Total", "Grade"]))        # print the data of the students

print("\n\nGrade        No. of students")                   # print the grade and freq table
print("S        ", count[0])
print("A        ", count[1])
print("B        ", count[2])
print("C        ", count[3])
print("D        ", count[4])
print("E        ", count[5])
print("U        ", count[6])
```

Detailed output :

The output consists of the marks tables and the grade frequency table.



```
dell@bhaavanaa:~/semester_8/big_data/lab/lab_1$ python3 Q7.py
The class average is  46.45

Name             Midsem    Endsem    Assignments    Total   Grade
---------  --------  --------  -------------  -------  -------
student1         18        27             18       63  C
student2         22        39             11       72  B
student3         27        50             11       88  A
student4          1         3             17       21  U
student5         10        25              9       44  E
student6         17        11              6       34  E
student7          7        49              3       59  D
student8         19         4              2       25  E
student9          4         0             13       17  U
student10        24        40             19       83  A
student11        22         1              2       25  E
student12         6         4             17       27  E
student13        16        31             18       65  C
student14        22         6              0       28  E
student15        29         4             17       50  D
student16        14         5             13       32  E
student17         4        21              2       27  E
student18        23        38              7       68  C
student19         4         9             17       30  E
student20         8        48             15       71  B


Grade           No. of students
S                0
A                2
B                2
C                3
D                2
E                9
U                2
dell@bhaavanaa:~/semester_8/big_data/lab/lab_1$ 
```

Problem 8

Problem statement :

Extend the application developed in (7) to support relative grading which uses the class average (mean) and standard deviation to compute the cutoffs for various grades as opposed to fixing them statically; you can refer the sample grader (excel sheet) attached to understand the formulas for fixing the cutoffs; the grader would involve, mean, standard deviation, max mark, passed students data mean, etc. Understand the excel grader thoroughly before you try mimicking such an application in your development platform.

Logic used :

Randomly generate the midsem, endsem and assignment scores of the students, and compute the total marks of the student. Set the pass mark as half of the mean of the total marks. Compute the threshold values of the grades using the criteria mentioned in the excel sheet and assign the grades to the student.

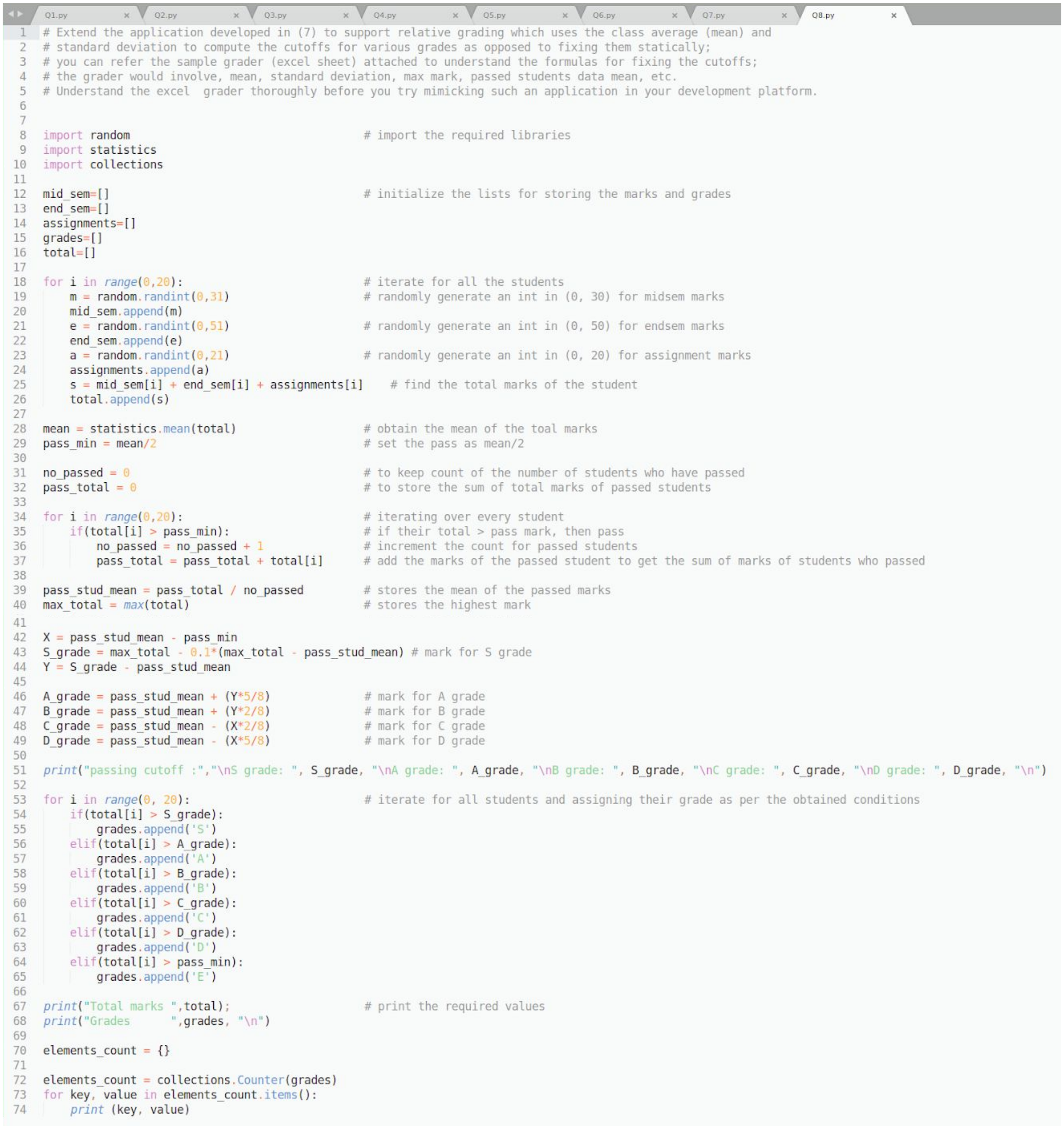
Packages used :

random - to generate the random numbers for the scores of the students.

statistics - for computing the quantities such as the mean, etc.

collections - to compute how many students got a particular grade.

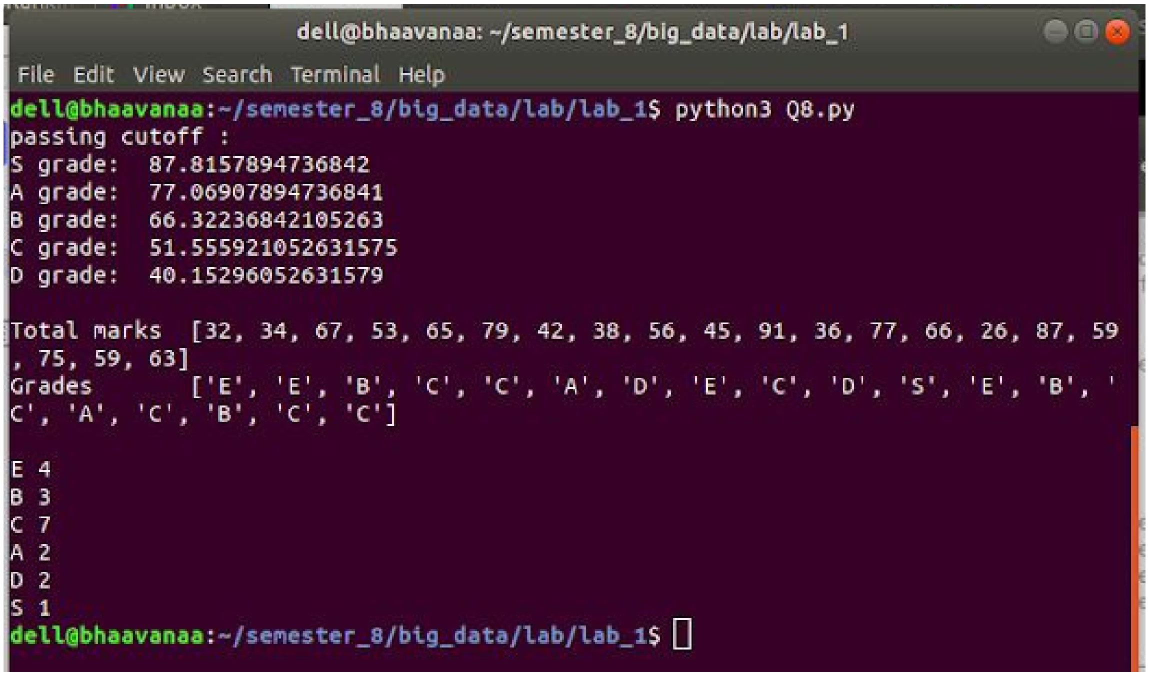
Code snippets :



```python
# Extend the application developed in (7) to support relative grading which uses the class average (mean) and
# standard deviation to compute the cutoffs for various grades as opposed to fixing them statically;
# you can refer the sample grader (excel sheet) attached to understand the formulas for fixing the cutoffs;
# the grader would involve, mean, standard deviation, max mark, passed students data mean, etc.
# Understand the excel  grader thoroughly before you try mimicking such an application in your development platform.


import random                                   # import the required libraries
import statistics
import collections

mid_sem=[]                                      # initialize the lists for storing the marks and grades
end_sem=[]
assignments=[]
grades=[]
total=[]

for i in range(0,20):                           # iterate for all the students
    m = random.randint(0,31)                    # randomly generate an int in (0, 30) for midsem marks
    mid_sem.append(m)
    e = random.randint(0,51)                    # randomly generate an int in (0, 50) for endsem marks
    end_sem.append(e)
    a = random.randint(0,21)                    # randomly generate an int in (0, 20) for assignment marks
    assignments.append(a)
    s = mid_sem[i] + end_sem[i] + assignments[i]    # find the total marks of the student
    total.append(s)

mean = statistics.mean(total)                   # obtain the mean of the toal marks
pass_min = mean/2                               # set the pass as mean/2

no_passed = 0                                   # to keep count of the number of students who have passed
pass_total = 0                                  # to store the sum of total marks of passed students

for i in range(0,20):                           # iterating over every student
    if(total[i] > pass_min):                    # if their total > pass mark, then pass
        no_passed = no_passed + 1               # increment the count for passed students
        pass_total = pass_total + total[i]      # add the marks of the passed student to get the sum of marks of students who passed

pass_stud_mean = pass_total / no_passed         # stores the mean of the passed marks
max_total = max(total)                          # stores the highest mark

X = pass_stud_mean - pass_min
S_grade = max_total - 0.1*(max_total - pass_stud_mean) # mark for S grade
Y = S_grade - pass_stud_mean

A_grade = pass_stud_mean + (Y*5/8)              # mark for A grade
B_grade = pass_stud_mean + (Y*2/8)              # mark for B grade
C_grade = pass_stud_mean - (X*2/8)              # mark for C grade
D_grade = pass_stud_mean - (X*5/8)              # mark for D grade

print("passing cutoff :","\nS grade: ", S_grade, "\nA grade: ", A_grade, "\nB grade: ", B_grade, "\nC grade: ", C_grade, "\nD grade: ", D_grade, "\n")

for i in range(0, 20):                          # iterate for all students and assigning their grade as per the obtained conditions
    if(total[i] > S_grade):
        grades.append('S')
    elif(total[i] > A_grade):
        grades.append('A')
    elif(total[i] > B_grade):
        grades.append('B')
    elif(total[i] > C_grade):
        grades.append('C')
    elif(total[i] > D_grade):
        grades.append('D')
    elif(total[i] > pass_min):
        grades.append('E')

print("Total marks ",total);                     # print the required values
print("Grades      ",grades, "\n")

elements_count = {}

elements_count = collections.Counter(grades)
for key, value in elements_count.items():
    print (key, value)
```

Detailed output :

The output consists of the grade threshold, grade obtained by each student, along with their total marks. It also consists of the number of students who have obtained a particular grade.



dell@bhaavanaa: ~/semester_8/big_data/lab/lab_1
File Edit View Search Terminal Help

```
dell@bhaavanaa:~/semester_8/big_data/lab/lab_1$ python3 Q8.py
passing cutoff :
S grade:  87.8157894736842
A grade:  77.06907894736841
B grade:  66.32236842105263
C grade:  51.555921052631575
D grade:  40.15296052631579

Total marks  [32, 34, 67, 53, 65, 79, 42, 38, 56, 45, 91, 36, 77, 66, 26, 87, 59
, 75, 59, 63]
Grades       ['E', 'E', 'B', 'C', 'C', 'A', 'D', 'E', 'C', 'D', 'S', 'E', 'B', '
C', 'A', 'C', 'B', 'C', 'C']

E 4
B 3
C 7
A 2
D 2
S 1
dell@bhaavanaa:~/semester_8/big_data/lab/lab_1$ 
```

Problem 9

Problem statement :

Consider the following sample of weights for 75 individuals: 79 71 89 57 76 64 82 82 67 80 81 65 73 79 79 60 58 83 74 68 78 80 78 81 76 65 70 76 58 82 59 73 72 79 87 63 74 90 69 70 83 76 61 66 71 60 57 81 57 65 81 78 77 81 81 63 71 66 56 62 75 64 74 74 70 71 56 69 63 72 81 54 72 91 92. For the above data generates histograms and depict them using packages in your platform. Explore the different types of histograms available and test drive the types supported in your platform.

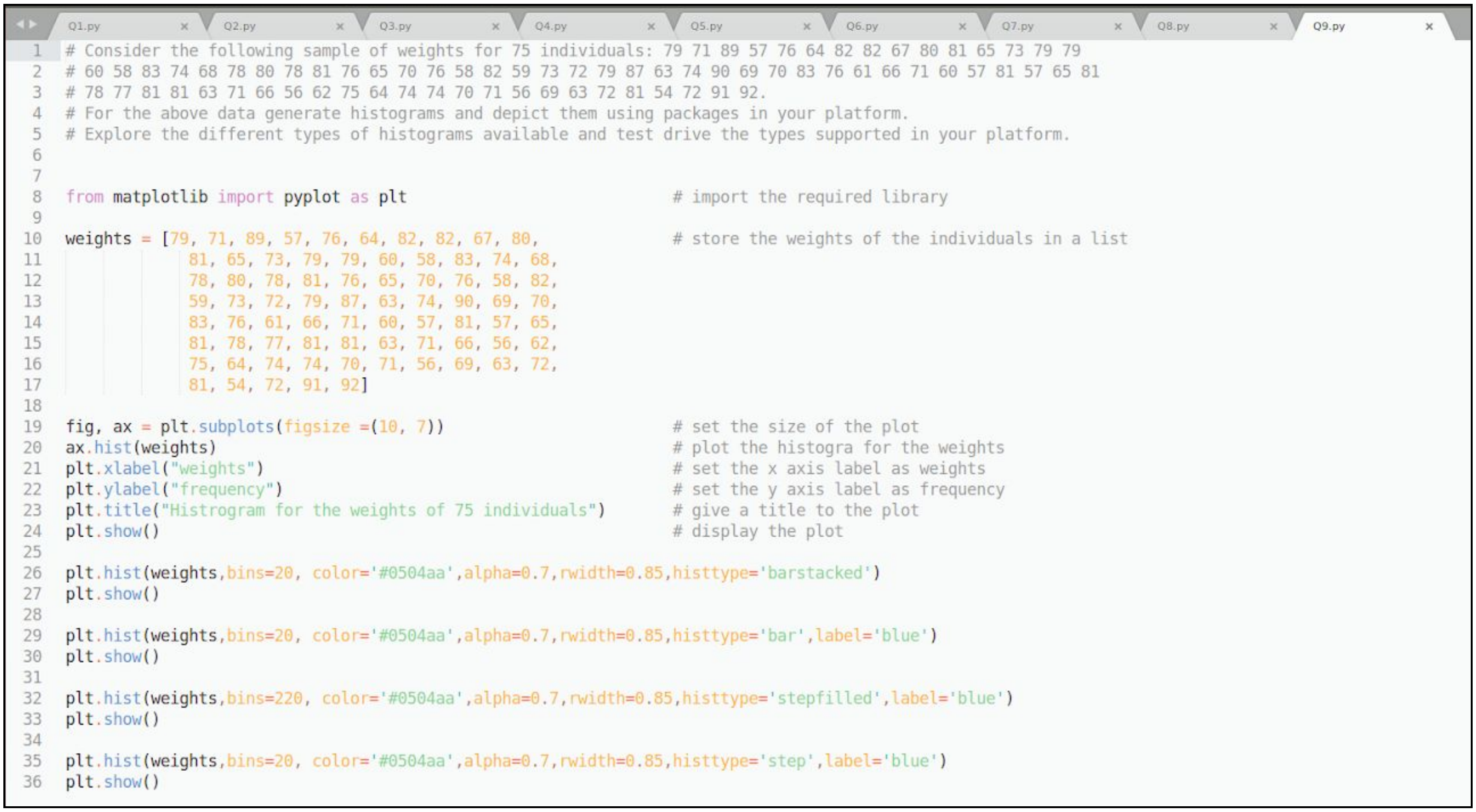
Logic used :

Given the samples of weights of 75 individuals, we can plot different types of histograms using the matplotlib library.

Packages used :

matplotlib.pyplot - for plotting the histograms.

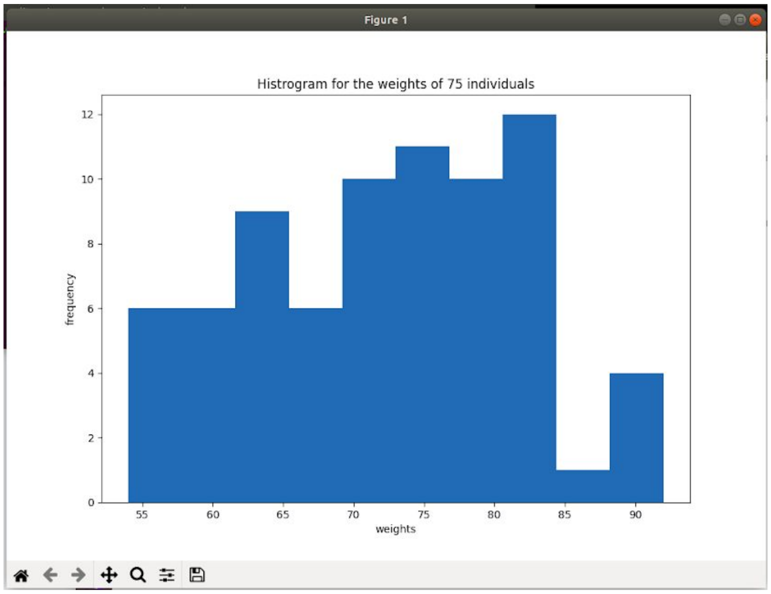
Code snippets :



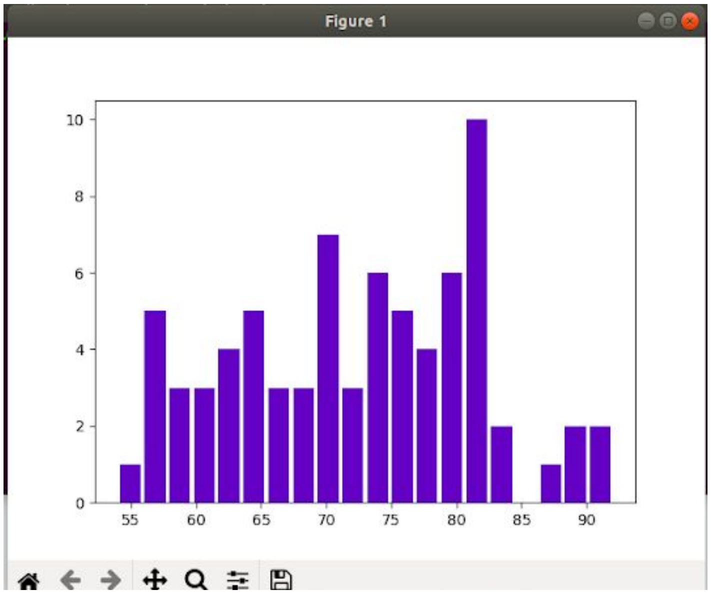
```python
# Consider the following sample of weights for 75 individuals: 79 71 89 57 76 64 82 82 67 80 81 65 73 79 79
# 60 58 83 74 68 78 80 78 81 76 65 70 76 58 82 59 73 72 79 87 63 74 90 69 70 83 76 61 66 71 60 57 81 57 65 81
# 78 77 81 81 63 71 66 56 62 75 64 74 74 70 71 56 69 63 72 81 54 72 91 92.
# For the above data generate histograms and depict them using packages in your platform.
# Explore the different types of histograms available and test drive the types supported in your platform.


from matplotlib import pyplot as plt                            # import the required library

weights = [79, 71, 89, 57, 76, 64, 82, 82, 67, 80,              # store the weights of the individuals in a list
             81, 65, 73, 79, 79, 60, 58, 83, 74, 68,
             78, 80, 78, 81, 76, 65, 70, 76, 58, 82,
             59, 73, 72, 79, 87, 63, 74, 90, 69, 70,
             83, 76, 61, 66, 71, 60, 57, 81, 57, 65,
             81, 78, 77, 81, 81, 63, 71, 66, 56, 62,
             75, 64, 74, 74, 70, 71, 56, 69, 63, 72,
             81, 54, 72, 91, 92]

fig, ax = plt.subplots(figsize =(10, 7))                        # set the size of the plot
ax.hist(weights)                                                # plot the histogra for the weights
plt.xlabel("weights")                                           # set the x axis label as weights
plt.ylabel("frequency")                                         # set the y axis label as frequency
plt.title("Histrogram for the weights of 75 individuals")       # give a title to the plot
plt.show()                                                      # display the plot

plt.hist(weights,bins=20, color='#0504aa',alpha=0.7,rwidth=0.85,histtype='barstacked')
plt.show()

plt.hist(weights,bins=20, color='#0504aa',alpha=0.7,rwidth=0.85,histtype='bar',label='blue')
plt.show()

plt.hist(weights,bins=220, color='#0504aa',alpha=0.7,rwidth=0.85,histtype='stepfilled',label='blue')
plt.show()

plt.hist(weights,bins=20, color='#0504aa',alpha=0.7,rwidth=0.85,histtype='step',label='blue')
plt.show()
```

Detailed output :

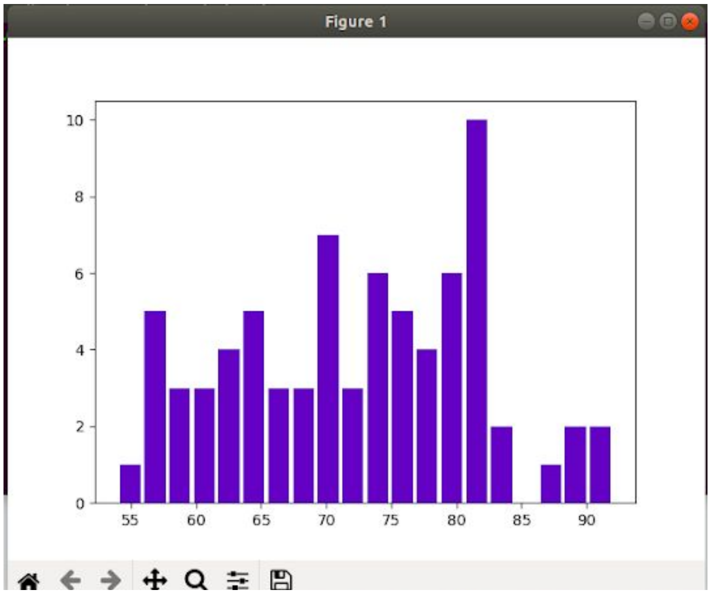
The output consists of the different types of histograms for the given data.



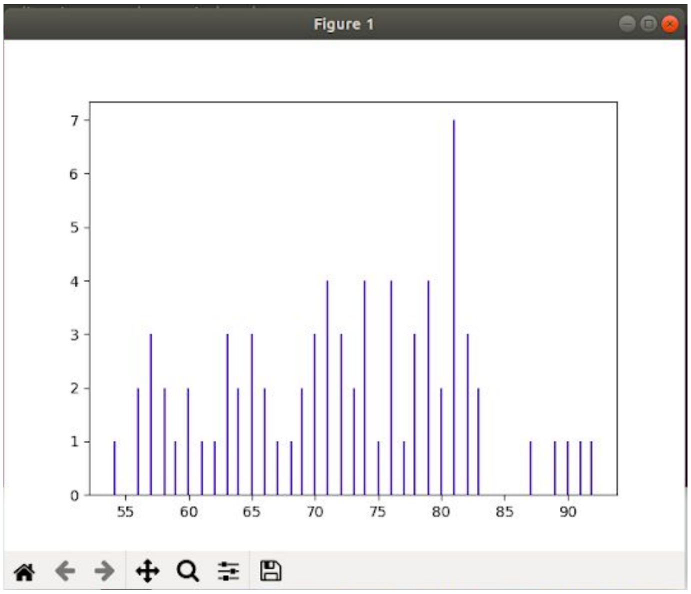
The normal histogram



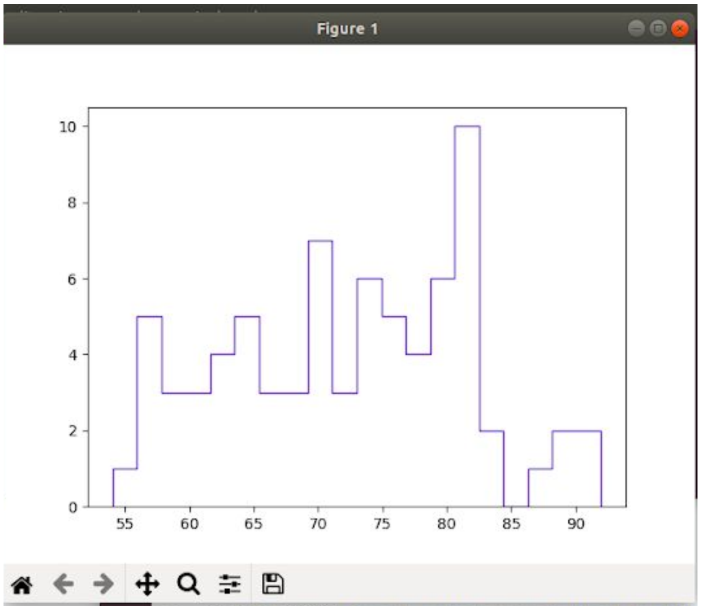
Barstacked histogram



Bar histogram



Stepfilled histogram



Step histogram