PROBLEM SET 1 MRIDUL HARISH, CED18I034

Question 1 - 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.

In [ ]:
```
from random import randint
import collections
```

In [ ]:
```
def generate_data():
    return[randint(1,10) for i in range(randint(100,200))]
```

In [ ]:
```
data = generate_data()
counter = dict(sorted(collections.Counter(data).items()))
```

In [ ]:
```
counter
```

Out[ ]:
```
{1: 15, 2: 9, 3: 20, 4: 18, 5: 12, 6: 17, 7: 17, 8: 13, 9: 5, 10: 14}
```

In [ ]:
```
print("{:<7} {:<30} {:<7}".format('Score', 'Tally', 'Frequency'))

for key, value in counter.items():
    tally = "".join("|" if i % 5 != 4 else "\ " for i in range(value))
    print("{:<7} {:<30} {:<7}".format(key, tally, value))
```

```
Score   Tally                          Frequency
1       |||\ |||\ |||\                 15
2       |||\ ||||                      9
3       |||\ |||\ |||\ |||\            20
4       |||\ |||\ |||\ |||             18
5       |||\ |||\ ||                   12
6       |||\ |||\ |||\ ||              17
7       |||\ |||\ |||\ ||              17
8       |||\ |||\ |||                  13
9       |||\                           5
10      |||\ |||\ ||||                 14
```

Question 2 - 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.

In [ ]:
```
import numpy as np
```

In [ ]:
```
def generate_marks(n):
    marks = {}
    for i in range(1, n+1):
        marks[f"CSE20D{i:02d}"] = 25+(i+8)%10 if i % 2 == 0 else 25+(i+7)%10
    return marks
```

In [ ]:
```
data = generate_marks(18)

print("{:<10} {:<5}".format('Roll number', 'Marks'))

for key, value in data.items():
    print("{:<10} {:<5}".format(key, value))
```

```
Roll number Marks
CSE20D01   33
CSE20D02   25
CSE20D03   25
CSE20D04   27
CSE20D05   27
CSE20D06   29
CSE20D07   29
CSE20D08   31
CSE20D09   31
CSE20D10   33
CSE20D11   33
CSE20D12   25
CSE20D13   25
CSE20D14   27
CSE20D15   27
CSE20D16   29
CSE20D17   29
CSE20D18   31
```

In [ ]:
```
marks = [*data.values()]
print(f"Mean of the marks = {np.mean(marks)}")
print(f"Median of the marks = {np.median(marks)}")
```

```
Mean of the marks = 28.666666666666668
Median of the marks = 29.0
```

Question 3 - 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).

In [ ]:
```
from random import randint
import numpy as np
```

In [ ]:
```
def generate_data(n):
    def generate_x():
        x = []
        while len(x) != 20:
            x.append(randint(6, 200))
            x = list(set(x))
        return x

    list_x = generate_x()
    return[(x, 2*x + 3) for x in list_x]
```

In [ ]:
```
data = generate_data(20)
data
```

Out[ ]:
```
[(131, 265),
 (138, 279),
 (142, 287),
 (14, 31),
 (18, 39),
 (153, 309),
 (25, 53),
 (28, 59),
 (30, 63),
 (158, 319),
 (166, 335),
 (168, 339),
 (177, 357),
 (50, 103),
 (186, 375),
 (86, 175),
 (93, 189),
 (96, 195),
 (110, 223),
 (127, 257)]
```

In [ ]:
```
x, y = zip(*data)
print(f"Standard deviation of the sample space is : {np.std(y)}")
```

```
Standard deviation of the sample space is : 114.54710821317141
```

Question 4 - For a given data of heights of a class, theheights 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 calculations 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]

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [ ]:
```
height = [167.65, 167, 172, 175, 165, 167, 168, 167, 167.3, 170, 167.5, 170, 1
x = list(range(1, len(height)+1))
```

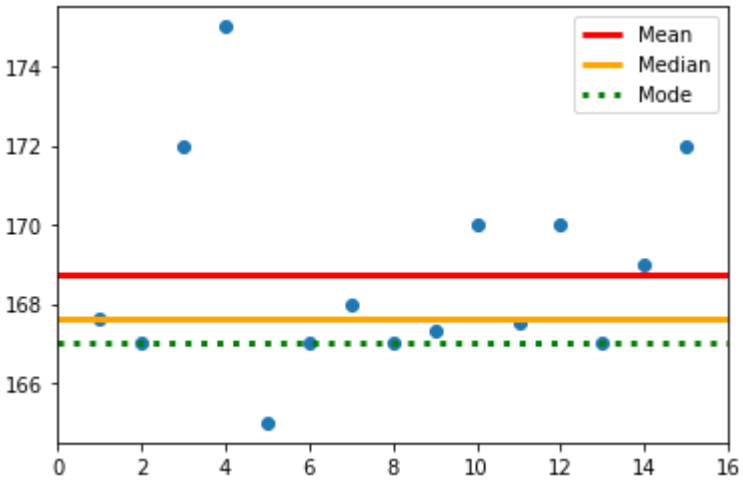
In [ ]:
```
data = pd.DataFrame({"x" : x, "height" : height})
data
```

Out[ ]:

	x	height
0	1	167.65
1	2	167.00
2	3	172.00
3	4	175.00
4	5	165.00
5	6	167.00
6	7	168.00
7	8	167.00
8	9	167.30
9	10	170.00
10	11	167.50
11	12	170.00
12	13	167.00
13	14	169.00
14	15	172.00

In [ ]:

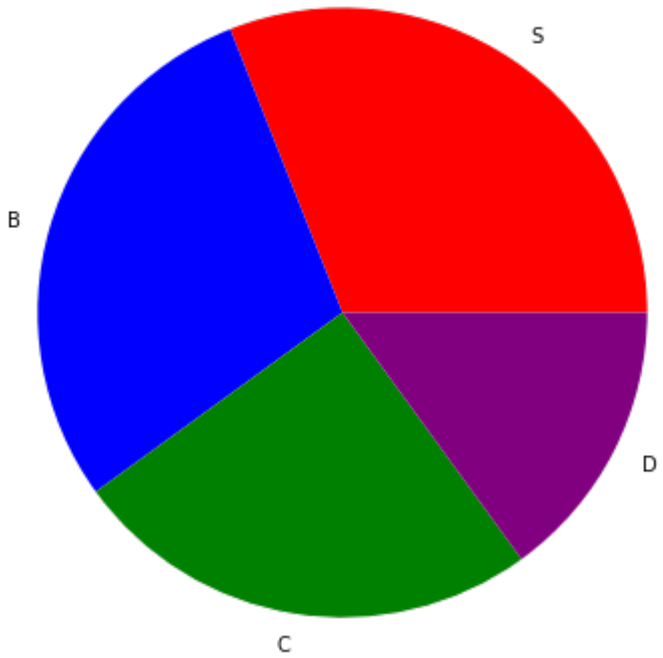
```
plt.xlim(0, len(height)+1)
plt.scatter(data['x'], data['height'])
Mean = np.mean(height)
Median = np.median(height)
Mode = data['height'].mode()[0]
Stddev = data['height'].std()
Skew = (Mean - Mode)/Stddev

plt.hlines(Mean, 0, 16, color = 'red', linestyle = 'solid', label = 'Mean', li
plt.hlines(Median, 0, 16, color = 'orange', linestyle = 'solid', label = 'Medi
plt.hlines(Mode, 0, 16, color = 'green', linestyle = 'dotted', label = 'Mode',
#plt.text(0, 160, "Mean = {:.2f}" "\nMedian = {:.2f}" "\nMode = {:.2f}" "\nSt
plt.legend()
plt.show()
```



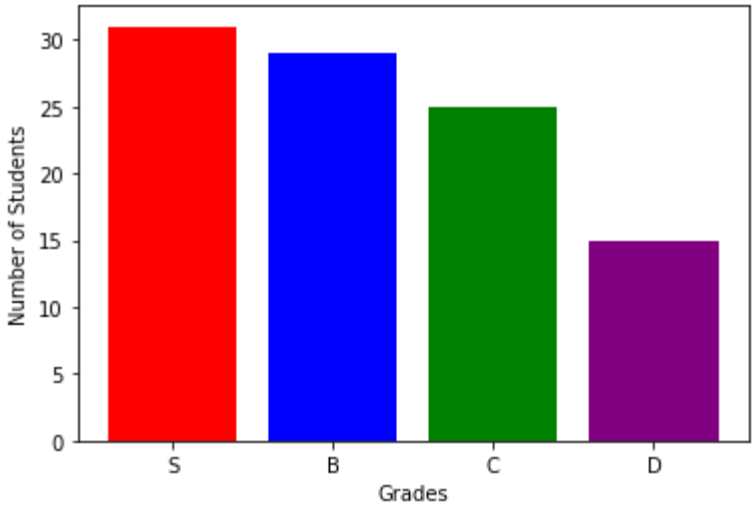
Question 5 - In Analytics and Systems of Bigdata course, for a class of 100 students, around 31 students secured 'S' grade, 29 secured 'B' grade, 25 'C' grades, and rest of them secured 'D' grades. If the range of each grade is 15 marks. (S for 85 to 100 marks, A for 70 to 85 ...). Develop an application that represents the above data: using Pie and Bar graphs.

In [ ]:
```
import matplotlib.pyplot as plt
```

In [ ]:
```
Grades = ["S", "B", "C", "D"]
Students = [31, 29, 25, 15]
Figure = plt.figure(figsize =(10, 7))
plt.pie(Students, labels = Grades, colors=["red", "blue", "green", "purple"])
plt.show()
```



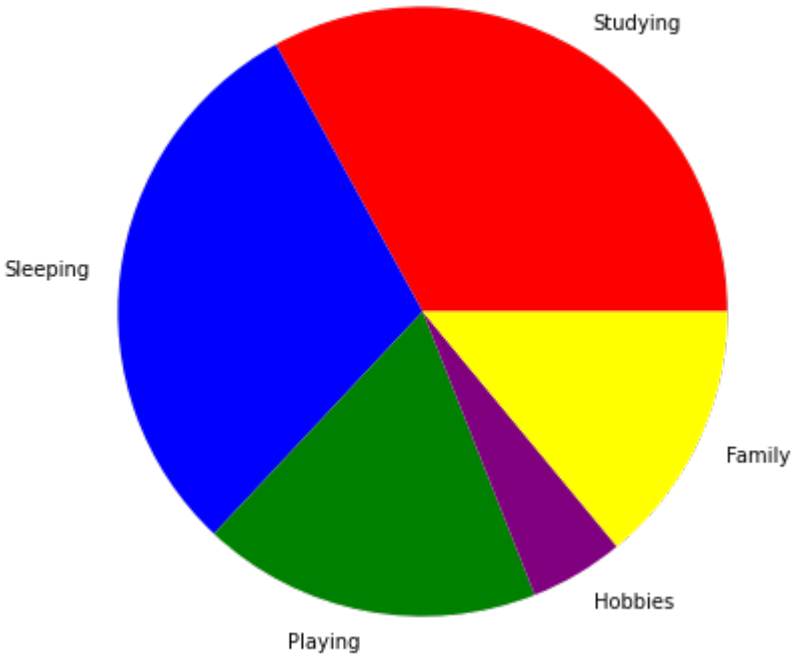
In [ ]:
```
plt.bar(Grades, Students, color = ["red", "blue", "green", "purple"])
plt.xlabel("Grades")
plt.ylabel("Number of Students")
plt.show()
```


30
25
20
15
10
5
0
Number of Students
S
B
C
D
Grades

Question 6 - 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.

In [ ]:
```
import matplotlib.pyplot as plt
```

In [ ]:
```
Activities = ["Studying", "Sleeping", "Playing", "Hobbies", "Family"]
Time_Distribution = [33, 30, 18, 5, 14]
Figure = plt.figure(figsize =(10, 7))
plt.pie(Time_Distribution, labels = Activities, colors=["red", "blue", "green"
plt.show()
```



Question 7 - 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). Maroon shows failure (" U grade "); similar to a heatmap...cold to warm. The color scheme is automatic and the least grade ends up with red-maroon shade

In [ ]:

```
import random
from random import randint
import numpy as np
import pandas as pd
```

In [ ]:

```
MidSem = [random.randint(0, 30) for i in range(20)]
EndSem = [random.randint(0, 50) for i in range(20)]
Assignments = [random.randint(0, 20) for i in range(20)]
TotalMarks = np.array([MidSem[i] + EndSem[i] + Assignments[i] for i in range(2
Grades = ['S', 'A', 'B', 'C', 'D', 'E', 'U']

def AbsoluteGrading(TotalMarks, average):
    if(TotalMarks >= 90):
        return('S')
    elif(TotalMarks >= 80):
        return('A')
    elif(TotalMarks >= 70):
        return('B')
    elif(TotalMarks >= 60):
        return('C')
    elif(TotalMarks >= 50):
        return('D')
    elif(TotalMarks >= (average/2)):
        return('E')
    else:
        return('U')

Grade = []
for i in range(20):
    Grade.append(AbsoluteGrading(TotalMarks[i], TotalMarks.mean()))

frequency = {}
frequency['S'] = 0
frequency['A'] = 0
frequency['B'] = 0
frequency['C'] = 0
frequency['D'] = 0
frequency['E'] = 0
frequency['U'] = 0

for i in Grade:
    if i in Grades:
        frequency[str(i)] += 1
```

In [ ]:
```
MidSem
```

Out[ ]:
```
[17, 0, 8, 19, 20, 18, 9, 12, 29, 9, 22, 25, 21, 11, 2, 17, 17, 17, 24, 8]
```

In [ ]:
```
EndSem
```

Out[ ]:
```
[30, 23, 46, 22, 16, 10, 29, 19, 50, 8, 8, 39, 7, 0, 47, 38, 16, 27, 11, 48]
```

In [ ]:
```
Assignments
```

Out[ ]:
```
[4, 5, 8, 19, 1, 11, 4, 6, 14, 3, 3, 17, 9, 15, 6, 7, 3, 15, 0, 17]
```

In [ ]:
```
TotalMarks
```

Out[ ]:
```
array([51, 28, 62, 60, 37, 39, 42, 37, 93, 20, 33, 81, 37, 26, 55, 62, 36,
       59, 35, 73])
```

In [ ]:
```
frequency
```

Out[ ]:
```
{'S': 1, 'A': 1, 'B': 1, 'C': 3, 'D': 3, 'E': 10, 'U': 1}
```

In [ ]:
```
data = pd.DataFrame(frequency.items(), columns = ['Grade', 'Frequency'])
data
```

Out[ ]:

	Grade	Frequency
0	S	1
1	A	1
2	B	1
3	C	3
4	D	3
5	E	10
6	U	1

In [ ]:
```
marks_list = pd.DataFrame({"Marks":TotalMarks, "Grade":Grade})
marks_list.style.background_gradient(cmap = 'Spectral')
```

Out[ ]:

	Marks	Grade
0	51	D
1	28	E
2	62	C

	Marks	Grade
3	60	C
4	37	E
5	39	E
6	42	E
7	37	E
8	93	S
9	20	U
10	33	E
11	81	A
12	37	E
13	26	E
14	55	D
15	62	C
16	36	E
17	59	D

Question 8 - 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. Formulas Required for Relative Grading: Passing Minimum: 50% of class average. (Minimum marks for passing) X= Passing Students' Mean- Passing Minimum. S_cutoff = Max_Mark – 0.1 *(Max_Mark-Passing Students Mean) Y= S_cutoff – Passing Students Mean A_cutoff = Passing Students Mean + Y* (5/8) B_cutoff = Passing Students Mean + Y *(2/8) C_cutoff = Passing Students Mean - X* (2/8) D_cutoff = Passing Students Mean - X * (5/8) E_cutoff = Passing Minimum Maroon shows failure (" U grade "); similar to a heatmap...cold to warm. The color scheme is automatic and the least grade ends up with red-maroon shade

In [ ]:
```
import random
from random import randint
import numpy as np
import pandas as pd
```

In [ ]:
```
MidSem = [random.randint(0, 30) for i in range(20)]
EndSem = [random.randint(0, 50) for i in range(20)]
Assignments = [random.randint(0, 20) for i in range(20)]
TotalMarks = np.array([MidSem[i] + EndSem[i] + Assignments[i] for i in range(
Grades = ['S', 'A', 'B', 'C', 'D', 'E', 'U']
```

In [ ]:
```
Mean = TotalMarks.mean()
PassingMinimum = Mean/2
PassingMarks = TotalMarks[TotalMarks > PassingMinimum]
PassingMean = PassingMarks.mean()
MaximumMarks = TotalMarks.max()
```

In [ ]:
```
Mean
```

Out[ ]:
```
48.0
```

In [ ]:
```
PassingMinimum
```

Out[ ]:
```
24.0
```

In [ ]:
```
PassingMarks
```

Out[ ]:
```
array([80, 67, 33, 42, 30, 42, 48, 59, 30, 53, 39, 82, 59, 32, 65, 67, 88])
```

In [ ]:
```
PassingMean
```

Out[ ]: 53.88235294117647

In [ ]:
```
MaximumMarks
```

Out[ ]: 88

In [ ]:
```
X = PassingMean - PassingMinimum
S_cutoff = MaximumMarks - 0.1*(MaximumMarks - PassingMean)
Y = S_cutoff - PassingMean
A_cutoff = PassingMean + Y*(5/8)
B_cutoff = PassingMean + Y*(2/8)
C_cutoff = PassingMean - X*(2/8)
D_cutoff = PassingMean - X*(5/8)
E_cutoff = PassingMinimum
```

In [ ]:
```
S_cutoff
```

Out[ ]: 84.58823529411765

In [ ]:
```
A_cutoff
```

Out[ ]: 73.07352941176471

In [ ]:
```
B_cutoff
```

Out[ ]: 61.55882352941177

In [ ]:
```
C_cutoff
```

Out[ ]: 46.411764705882355

In [ ]:
```
D_cutoff
```

Out[ ]: 35.205882352941174

In [ ]:
```
E_cutoff
```

Out[ ]: 24.0

In [ ]:
```
def RelativeGrading(TotalMarks):
    if(TotalMarks >= S_cutoff):
        return('S')
    elif(TotalMarks >= A_cutoff):
        return('A')
    elif(TotalMarks >= B_cutoff):
        return('B')
    elif(TotalMarks >= C_cutoff):
        return('C')
    elif(TotalMarks >= D_cutoff):
        return('D')
    elif(TotalMarks >= E_cutoff):
        return('E')
    else:
        return('U')

Grade = []
for i in range(20):
    Grade.append(RelativeGrading(TotalMarks[i]))
```

In [ ]:
```
frequency = {}
frequency['S'] = 0
frequency['A'] = 0
frequency['B'] = 0
frequency['C'] = 0
frequency['D'] = 0
frequency['E'] = 0
frequency['U'] = 0

for i in Grade:
    if i in Grades:
        frequency[str(i)] += 1
```

In [ ]:
```
frequency
```

Out[ ]:
```
{'S': 1, 'A': 2, 'B': 3, 'C': 4, 'D': 3, 'E': 4, 'U': 3}
```

In [ ]:
```
data = pd.DataFrame(frequency.items(), columns = ['Grade', 'Frequency'])
data
```

Out[ ]:

	Grade	Frequency
0	S	1
1	A	2
2	B	3
3	C	4
4	D	3
5	E	4
6	U	3

In [ ]:

```
marks_list = pd.DataFrame({"Marks":TotalMarks, "Grade":Grade})
marks_list.style.background_gradient(cmap = 'Spectral')
```

Out[ ]:

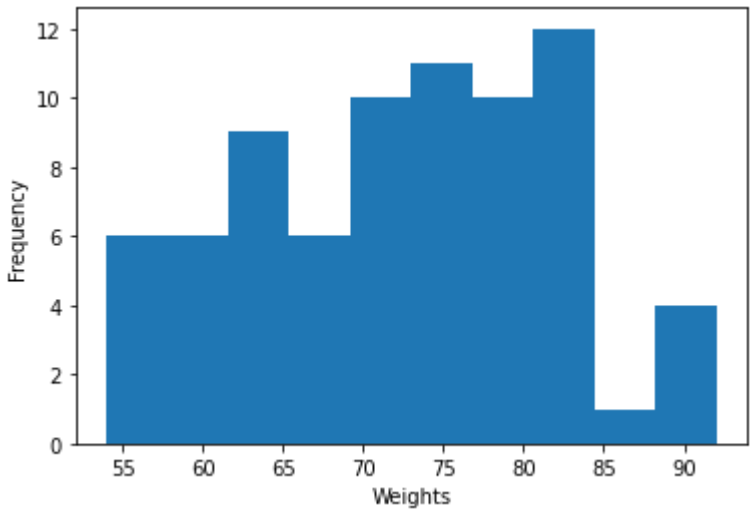
	Marks	Grade
0	17	U
1	80	A
2	67	B
3	33	E
4	13	U
5	42	D
6	30	E
7	42	D
8	48	C
9	14	U
10	59	C
11	30	E
12	53	C
13	39	D
14	82	A
15	59	C
16	32	E
17	65	B
18	67	B
19	88	S

Question 9 - Consider the following sample of weights for 45 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

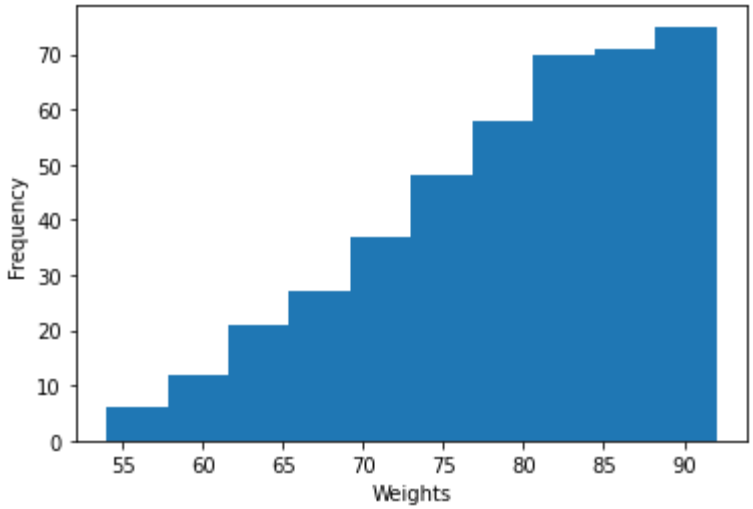
In [ ]:
```
import matplotlib.pyplot as plt
```

In [ ]:
```
Weights = [79, 71, 89, 57, 76, 64, 82, 82, 67, 80, 81, 65, 73, 79, 79, 60, 58,
81, 76, 65, 70, 76, 58, 82, 59, 73, 72, 79, 87, 63, 74, 90, 69, 70, 83, 76, 61
65, 81, 78, 77, 81, 81, 63, 71, 66, 56, 62, 75, 64, 74, 74, 70, 71, 56, 69, 63
```

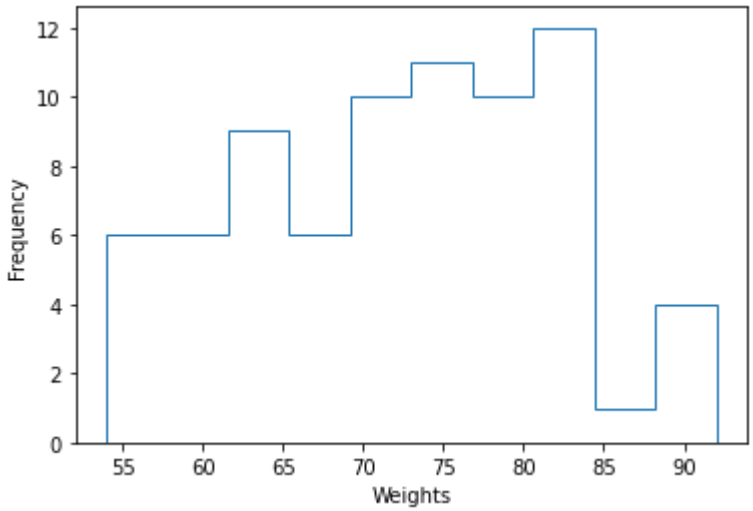
In [ ]:
```
plt.hist(Weights)
plt.xlabel("Weights")
plt.ylabel("Frequency")
plt.show()
```



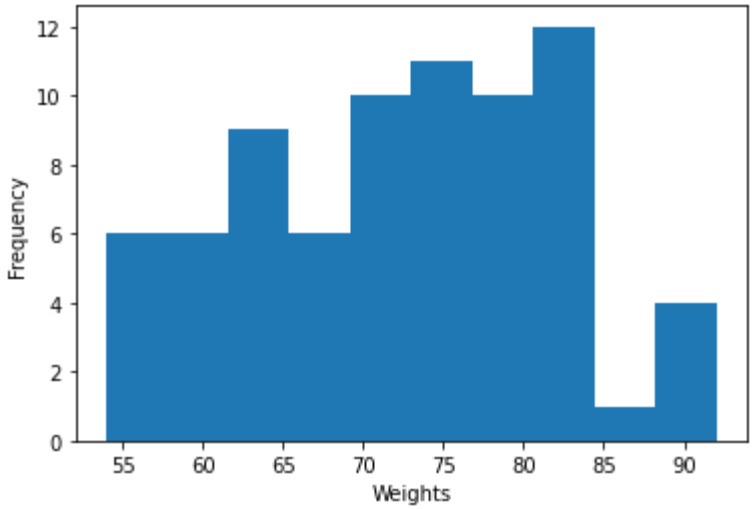
In [ ]:
```
plt.hist(Weights, cumulative=True)
plt.xlabel("Weights")
plt.ylabel("Frequency")
plt.show()
```



In [ ]:
```
plt.hist(Weights, histtype="step")
plt.xlabel("Weights")
plt.ylabel("Frequency")
plt.show()
```



In [ ]:
```
plt.hist(Weights, histtype="barstacked")
plt.xlabel("Weights")
plt.ylabel("Frequency")
plt.show()
```



In [ ]:

```
plt.hist(Weights, histtype="stepfilled", orientation="horizontal")
plt.xlabel("Weights")
plt.ylabel("Frequency")
plt.show()
```