Hrishikesh Rajesh Menon COE18B024 Problem-Set 1

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.

Modules Used:-

Numpy

Pandas

Approach:-

- Taking a random dataset with more than 50 entries
- Preprocessing data
- Using a loop to calculate Tally Values
- Data is added to a Pandas Dataframe and printed

In [341]:
```
import pandas as pan
import numpy as nps

def Grader(marks :float):
    if(marks>=90):
        return('A+')
    elif(marks>=80):
        return('A')
    elif(marks>=70):
        return('B')
    elif(marks>=60):
        return('C')
    elif(marks>=50):
        return('D')
    elif(marks>=40):
        return('E')
    return 'U'

Student_file=open("class-grades.csv")

Marks=[]
Grades=[]

chk=True
for row in Student_file:
    rw=row.split(',')

    if chk:
        chk=False
        continue

    if(rw[5]!=''):
        x=float(rw[5])
    Marks.append(x)
    Grades.append(Grader(x))

stud=pan.DataFrame({"Marks":Marks,"Grades":Grades})
counts=stud.iloc[:,1].value_counts()
display(counts)
```

```
C     21
A+    16
D     16
A     15
B     12
E     12
U      7
Name: Grades, dtype: int64
```

```
In [342]: grades = list(counts.index.values)
          freq = list(counts.values)
          print("Grades",grades)
          print("Frequencies",freq)
```

```
Grades ['C', 'A+', 'D', 'A', 'B', 'E', 'U']
Frequencies [21, 16, 16, 15, 12, 12, 7]
```

```
In [343]: tally = []
          for i in freq:
              tmp = ""
              for j in range(1,i+1):
                  if (j%5 != 0):
                      tmp = tmp+"|"
                  else:
                      tmp = tmp+"/ "
              tally.append(tmp)
          print("Tally",tally)
```

```
Tally ['||||/ ||||/ ||||/ ||||/ |', '||||/ ||||/ ||||/ |', '||||/ ||||/ ||||/ |', '||||/ ||||/ ||||/ ', '|
|||/ ||||/ ||', '||||/ ||||/ ||', '||||/ ||']
```

```
In [344]: grade = pan.DataFrame({"Grades":grades,"Tally":tally,"Frequency":freq})
          display(grade)
```

	Grades	Tally	Frequency
0	C	||||/ ||||/ ||||/ ||||/ |	21
1	A+	||||/ ||||/ ||||/ |	16
2	D	||||/ ||||/ ||||/ |	16
3	A	||||/ ||||/ ||||/	15
4	B	||||/ ||||/ ||	12
5	E	||||/ ||||/ ||	12
6	U	||||/ ||	7

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.

Modules Used:-

Numpy

Pandas

Approach:-

- Creating the Dataset and assigning marks based on formula
- Create a Pandas Dataframe for the given data
- Using mean and median functions to calculate and print it out

In [345]:

```
#Creating the dataset and dataframe
rollno=[]
marks=[]

for i in range(1,19):
    rollno.append("CSE20D"+ (str(i).zfill(2)) )
    if (i%2==0):
        marks.append(25+ ((i+8)%10) )
    else :
        marks.append(25+ ((i+7)%10) )

stud = pan.DataFrame({"Rollno":rollno,"Marks":marks})
stud
```

Out[345]:

	Rollno	Marks
0	CSE20D01	33
1	CSE20D02	25
2	CSE20D03	25
3	CSE20D04	27
4	CSE20D05	27
5	CSE20D06	29
6	CSE20D07	29
7	CSE20D08	31
8	CSE20D09	31
9	CSE20D10	33
10	CSE20D11	33
11	CSE20D12	25
12	CSE20D13	25
13	CSE20D14	27
14	CSE20D15	27
15	CSE20D16	29
16	CSE20D17	29
17	CSE20D18	31

In [346]:

```
#Printing mean and median

mean=stud['Marks'].mean()
median=stud['Marks'].median()
print("Mean:-",mean)
print("\nMedian:-",median)
```

```
Mean:- 28.666666666666668

Median:- 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.

Modules Used:-

Numpy

Pandas

Random

Approach:-

- Creating the X Dataset and assigning the respective Y using the line equation
- Create a Pandas Dataframe for the given data
- Using the standard deviation functions to calculate and print it out

In [347]:
```
import random
X_list=[]
Y_list=[]

for i in range(20):
    temp=random.randrange(6,1000)
    X_list.append(temp)
    Y_list.append((temp*2)+3)

line_df= pan.DataFrame({"X":X_list,"Y":Y_list})
line_df
```

Out[347]:

	X	Y
0	819	1641
1	135	273
2	425	853
3	65	133
4	393	789
5	172	347
6	628	1259
7	552	1107
8	141	285
9	559	1121
10	81	165
11	18	39
12	100	203
13	889	1781
14	46	95
15	971	1945
16	272	547
17	229	461
18	975	1953
19	990	1983

```
In [348]: #Finding a Standard Devation

          print("Standard Deviation:-")
          line_df.std()

          Standard Deviation:-

Out[348]: X    349.439551
          Y    698.879103
          dtype: float64
```

QUESTION 4

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.

1. Mean height of the student
2. Median and Mode of the sample space
3. Standard deviation
4. Measure of skewness. **{(Mean-Mode)/standard deviation}**

Modules Used:-

Numpy

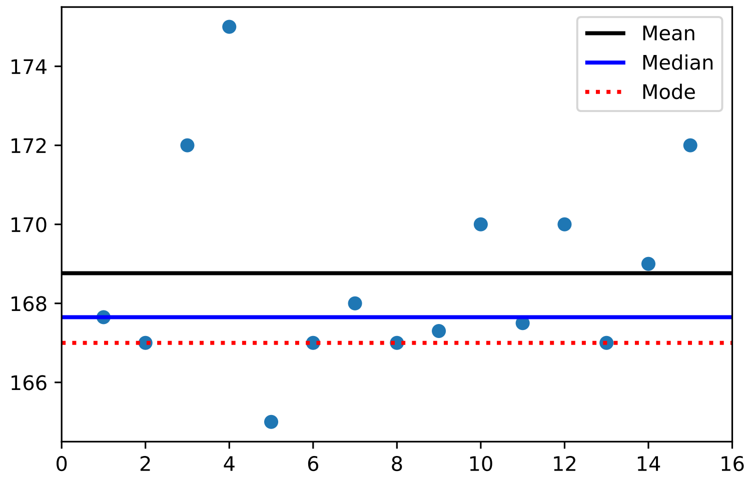
Pandas

Matplotlib

Approach:-

- Creating the X Dataset and assigning the respective Y using the given data
- Create a Pandas Dataframe for the given data
- Using the numpy functions to calculate mean,median,mode,standard
- Using Mathplotlib to express the data in an easy to read visual form

In [349]:
```
import matplotlib.pyplot as plt

X_list=list(range(1,16))
Y_list=[167.65, 167, 172, 175, 165, 167, 168, 167, 167.3, 170, 167.5, 170, 167, 169,172]

scat= pan.DataFrame({"X":X_list,"Y":Y_list})
scat.head()
plt.xlim(0,16)
plt.scatter(scat['X'],scat['Y'])
mean=nps.mean(Y_list)
median=nps.median(Y_list)
mode=scat['Y'].mode()[0]
stan_dev=scat['Y'].std()
skew=(mean-mode)/stan_dev

plt.hlines(mean,0,16,colors='black',linestyles='solid',label='Mean',linewidth=2)
plt.hlines(median,0,16,colors='blue',linestyles='solid',label='Median',linewidth=2)
plt.hlines(mode,0,16,color='red',linestyles='dotted',label='Mode',linewidth=2)
plt.text(0,160,"Mean:- {:.2f} \nMedian:- {:.2f} \nMode:- {:.2f} \nStandard Deviation:-  {:.2f} \nMeasure o
f skewness:- {:.2f}".format(mean,median,mode,stan_dev,skew))
plt.legend()
plt.show()
```



```
Mean:- 168.76
Median:- 167.65
Mode:- 167.00
Standard Deviation:-  2.61
Measure of skewness:- 0.68
```

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.

Modules Used:-

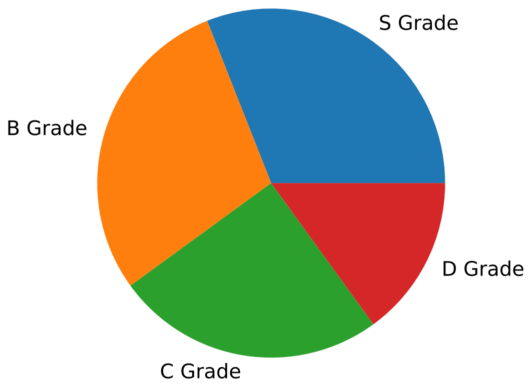
Numpy

Matplotlib

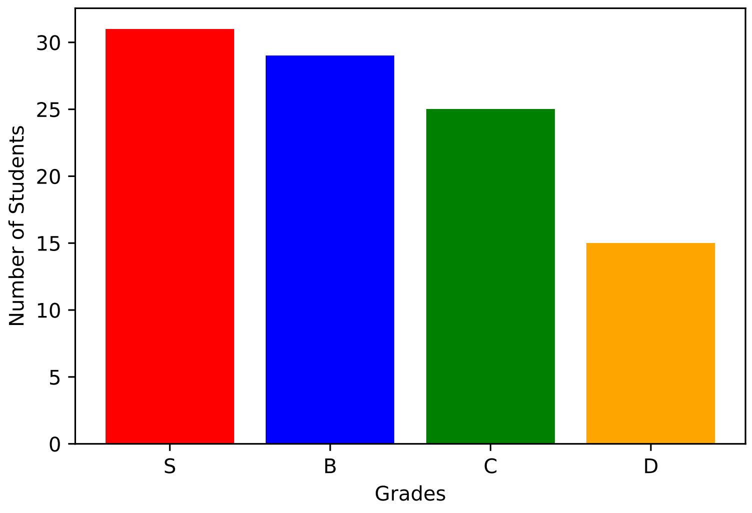
Approach:-

- Create a Dictionary to hold the respective frequency of Grades
- Using Mathplotlib to express the data in an easy to read visual form

In [350]:
```
grades = ["S Grade","B Grade","C Grade","D Grade"]
Stud = nps.array([31,29,25,15])

#Pie
plt.pie(Stud,labels=grades)
plt.show()
```



In [351]:
```
#bar graph
grades = ["S","B","C","D"]
plt.bar(grades,Stud,color=["red","blue","green","orange"])
plt.xlabel("Grades")
plt.ylabel("Number of Students")
plt.show()
```



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.

Modules Used:-

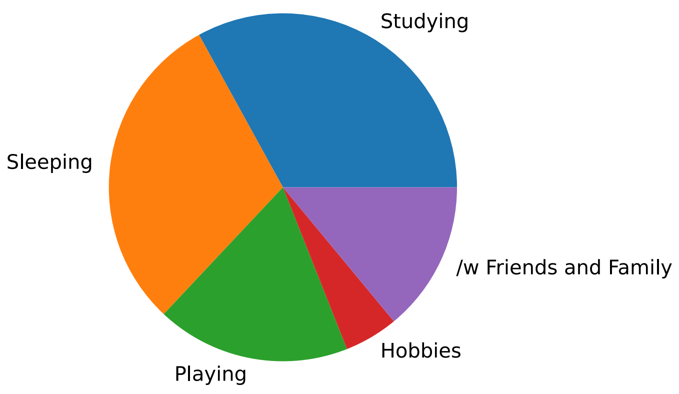
Numpy

Matplotlib

Approach:-

- Create a Dictionary to hold the respective frequency of time
- Using Mathplotlib to express the data in an easy to read visual form

In [352]:
```python
time = [33,30,18,5,14]
activities = ["Studying","Sleeping","Playing","Hobbies","/w Friends and Family"]
plt.pie(time,labels=activities)
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).

Modules Used:-

Numpy

Matplotlib

Random

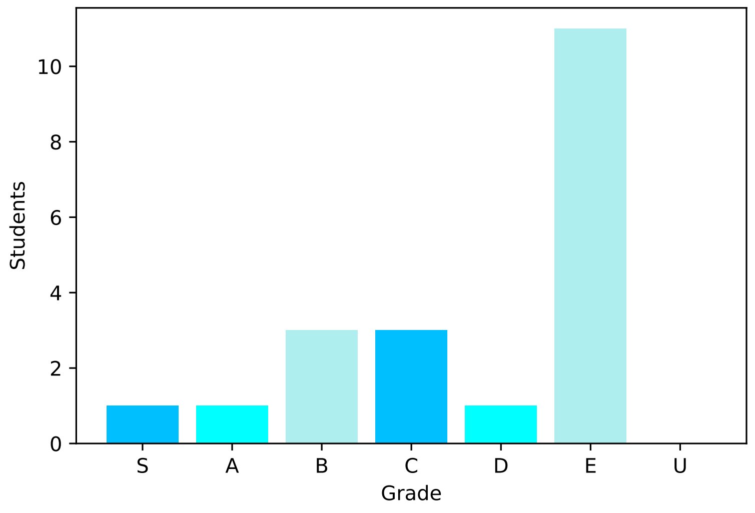
Approach:-

- Generate a random Dataset for Marks
- Assign the grader using a function
- Calculate frequency of grades and store in a DataFrame
- Using Mathplotlib to express the data in an easy to read visual form

In [484]:
```
Mid = [random.randint(0,30) for i in range(20)]
End = [random.randint(0,50) for i in range(20)]
Assign = [random.randint(0,20) for i in range(20)]

Marks = nps.array([Mid[i]+End[i]+Assign[i] for i in range(20)])

def Grader(marks :int,mean : int):
    if(marks>=90):
        return('S')
    elif(marks>=80):
        return('A')
    elif(marks>=70):
        return('B')
    elif(marks>=60):
        return('C')
    elif(marks>=50):
        return('D')
    elif(marks>=mean/2):
        return('E')
    return 'U'

Grades=[]
for i in range(20):
    Grades.append(Grader(Marks[i],Marks[i].mean()))

stud=pan.DataFrame({"Marks":Marks,"Grades":Grades})
counts=stud.iloc[:,1].value_counts()

freq=[]
grad=['S','A','B','C','D','E','U']
for i in range(len(grad)):
    try:
        cout=counts[grad[i]]
    except:
        cout=0
    freq.append(cout)

Freq_table = pan.DataFrame({"Grades":grad,"Freq":freq})

Freq_table
```

Out[484]:

	Grades	Freq
0	S	1
1	A	1
2	B	3
3	C	3
4	D	1
5	E	11
6	U	0

In [488]:
```
#Bar graph of Grade Frequency
plt.bar(grad,freq,color=["deepskyblue","cyan","paleturquoise"])
plt.xlabel("Grade")
plt.ylabel('Students')
plt.plot()
```
Out[488]:
```
[]
```


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.

Modules Used:-

Numpy

Matplotlib

Random

Approach:-

- Use the same dataset of marks as used in QUESTION 7
- Find Mean and Passing Marks
- Calculate the CutOffs using Relative Grading Formula to assign grades
- Calculate frequency of grades and store in a DataFrame
- Using Mathplotlib to express the data in an easy to read visual form

In [492]:
```
#Using np array "Marks" from Question 7

Mean = Marks.mean()
PassingMinimum = Mean/2

PassingMarks = Marks[Marks>PassingMinimum]
PassingMean = PassingMarks.mean()

MaxMark = Marks.max()
```

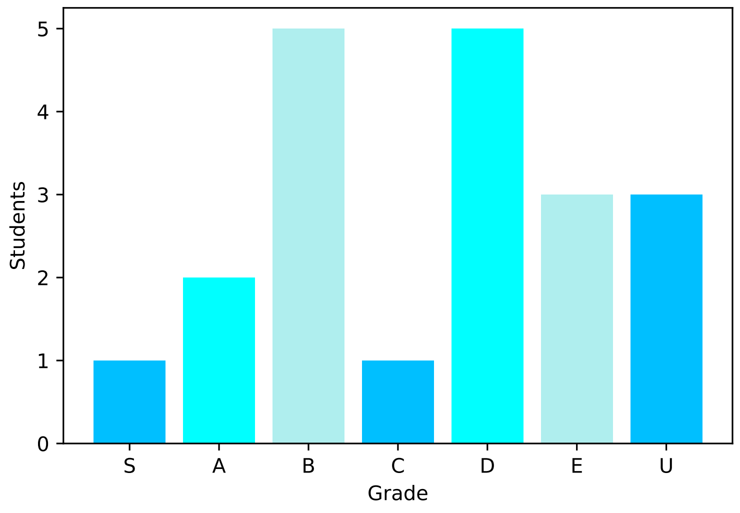
In [495]:
```
#Calculating Cutoff

X = PassingMean - PassingMinimum

Scoff = MaxMark - 0.1*(MaxMark-PassingMean)

Y = Scoff - PassingMean

Acoff = PassingMean+Y*5/8
Bcoff = PassingMean+Y*2/8
Ccoff = PassingMean-X*2/8
Dcoff = PassingMean-X*5/8
Ecoff = PassingMinimum

#Grader function
def Grader(marks :int):
    if(marks>=Scoff):
        return('S')
    elif(marks>=Acoff):
        return('A')
    elif(marks>=Bcoff):
        return('B')
    elif(marks>=Ccoff):
        return('C')
    elif(marks>=Dcoff):
        return('D')
    elif(marks>=Ecoff):
        return('E')
    return 'U'


#Assigning Grades along with Frequency
Grades=[]
for i in range(20):
    Grades.append(Grader(Marks[i]))

stud=pan.DataFrame({"Marks":Marks,"Grades":Grades})
counts=stud.iloc[:,1].value_counts()

freq=[]
grad=['S','A','B','C','D','E','U']
for i in range(len(grad)):
    try:
        cout=counts[grad[i]]
    except:
        cout=0
    freq.append(cout)

Freq_table = pan.DataFrame({"Grades":grad,"Freq":freq})

Freq_table
```

Out[495]:

	Grades	Freq
0	S	1
1	A	2
2	B	5
3	C	1
4	D	5
5	E	3
6	U	3

In [498]:
```
#Bar graph of Grade Frequency
plt.bar(grad,freq,color=["deepskyblue","cyan","paleturquoise"])
plt.xlabel("Grade")
plt.ylabel('Students')
plt.plot()
```

Out[498]:
```
[]
```



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.

Modules Used:-

Matplotlib

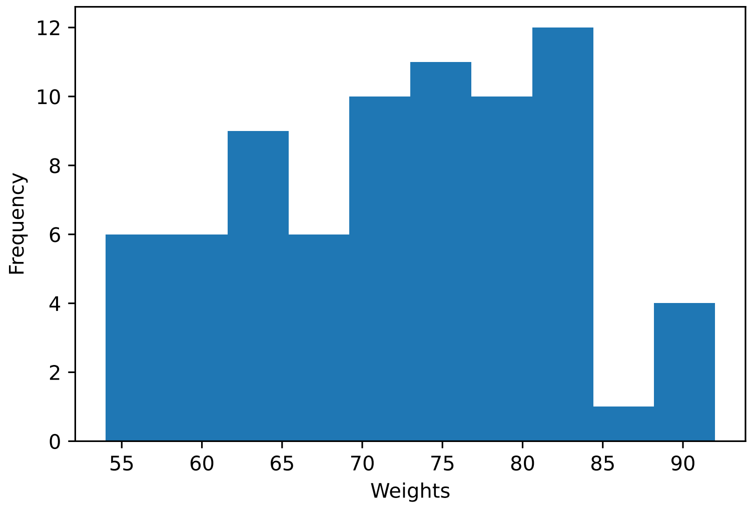
Approach:-

- Split the string into a list using split()
- Using Mathplotlib to depict the different types of histograms

In [500]:
```
Weight_samps=("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 5
9 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".split(" "))

Weight_samps=[int(Weight_samps[i]) for i in range(len(Weight_samps))]
```

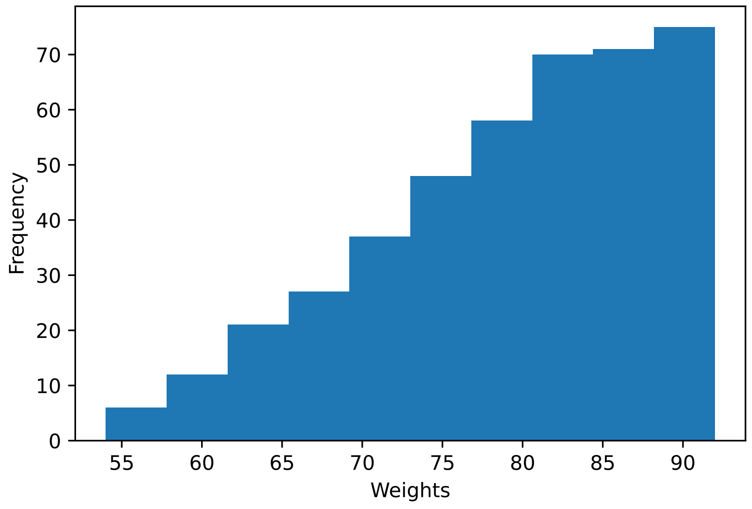
Default Bar Histogram

In [502]:
```python
# Using Matplotlib default histogram

plt.hist(Weight_samps)
plt.xlabel("Weights")
plt.ylabel("Frequency")
plt.show()
```



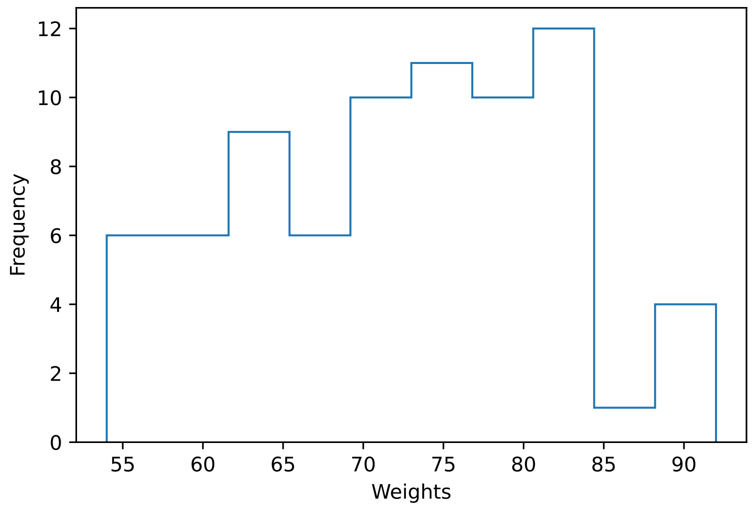
Cumulative Histogram

In [504]:
```python
#Using Matplotlib cumulative Histogram (each bin gives the counts in that bin plus all bins for smaller va
lues. The last bin gives the total number of datapoints)
plt.hist(Weight_samps, cumulative=True)
plt.xlabel("Weights")
plt.ylabel("Frequency")
plt.show()
```



Step Histogram

```
In [505]: # Using Matplotlib Step Unfilled histogram

          plt.hist(Weight_samps,histtype="step")
          plt.xlabel("Weights")
          plt.ylabel("Frequency")
          plt.show()
```


Frequency
Weights