3. Array Aggregation Functions [NUMPY]

In [1]:
```
import numpy as n
a=n.array([54,78,32,46,89,76])
print("\nAn Array:",a)
print("\nSum:",n.sum(a))
print("Product:",n.prod(a))
print("Mean:",n.mean(a))
print("Standard Deviation:",n.std(a))
print("Variance",n.var(a))
print("Minimum Value:",n.min(a))
print("Max:",n.max(a))
print("Min Index:",n.argmin(a))
print("Max Index:",n.argmax(a))
print("Median:",n.median(a))
print("Product:",n.prod(a))
```

```
An Array: [54 78 32 46 89 76]

Sum: 375
Product: -1012440064
Mean: 62.5
Standard Deviation: 19.997916558148418
Variance 399.9166666666667
Minimum Value: 32
Max: 89
Min Index: 2
Max Index: 4
Median: 65.0
Product: -1012440064
```

4. Vectorized Operations using NUMPY

Vectorized Sum and Multiplication product:

In [2]:
```
import numpy as np
import timeit
np.a=[4,5,1]
print(np.prod(np.a))
print("Time taken by vectorized product : ",end= "")
%timeit np.prod(np.a)
total=1
for item in np.a:
 total=total*item
t=total
print(t)
print("Time taken by iterative multiplication : ",end= "")
%timeit t
```

```
20
Time taken by vectorized product : 12.4 µs ± 1.19 µs per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
20
Time taken by iterative multiplication : 39.4 ns ± 4.22 ns per loop (mean ± std. dev. of 7 runs, 10,000,000 loop
s each)
```

SUM:

In [3]:
```
import numpy as n
import timeit
print(n.sum(n.arange(4)))
print("Time taken to vectorized sum:")
%timeit n.sum(n.arange(4))
t=0
for i in range(0,4):
 t=t+i
a=t
print("\n"+str(a))
print("Time Taken by iterative sum:",end="")
%timeit a
```

```
6
Time taken to vectorized sum:
9.6 µs ± 1.65 µs per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
str 6
Time Taken by iterative sum:41.1 ns ± 3.45 ns per loop (mean ± std. dev. of 7 runs, 10,000,000 loops each)
```

5. Use Map, Filter, Reduce and Lambda Functions on List using Numpy

In [4]:
```
import numpy as n
```

```
da=[60,8,7,5,34,78]
d=n.array(da)
from functools import reduce as r
print(list(map(lambda num:num**2,d)))
print(list(filter(lambda num:num>2,d)))
print(r(lambda x,y:x+y,d))
```

```
[3600, 64, 49, 25, 1156, 6084]
[60, 8, 7, 5, 34, 78]
192
```

6. Using aggregation functions on a Data Frame

In [5]:
```
import pandas as p
d=p.DataFrame([[2,5,6],
[4,6,3],
[5,7,8]],
columns=["Maths","Java","Py"])
print(d)
c=d.agg(['sum','min','max','count','mean','median','std','size',])
print()
print(c)
```

```
   Maths  Java  Py
0      2     5   6
1      4     6   3
2      5     7   8

            Maths  Java         Py
sum     11.000000  18.0  17.000000
min      2.000000   5.0   3.000000
max      5.000000   7.0   8.000000
count    3.000000   3.0   3.000000
mean     3.666667   6.0   5.666667
median   4.000000   6.0   6.000000
std      1.527525   1.0   2.516611
size     3.000000   3.0   3.000000
```

7. Grouping using Pandas on a Dataframe

In [6]:
```
import pandas as p
t={
'Course':["PY","JV","DBMS","MMA","MMA"],
'Fee':[300,600,21,350,67],
'Complexity':[100,56,32,10,67]
}
d=p.DataFrame(t)
print(d)
c=d.groupby('Course').agg({'Fee':'min'})
print("\n",c)
```

```
  Course  Fee  Complexity
0     PY  300         100
1     JV  600          56
2   DBMS   21          32
3    MMA  350          10
4    MMA   67          67

         Fee
Course
DBMS      21
JV       600
MMA       67
PY       300
```

8. Pivot and melt functions using Pandas

In [7]:
```
import pandas as p
t={
'Course':["PY","JV","DBMS","MMA","MMA"],
'Fee':[300,600,21,350,67],
'Complexity':[100,56,32,10,67]
}
d=p.DataFrame(t)
print(d)
print("\n",d.pivot(columns='Course',values='Complexity'))
print("\n",d.melt())
```

```
  Course  Fee  Complexity
0     PY  300         100
1     JV  600          56
2   DBMS   21          32
3    MMA  350          10
4    MMA   67          67

 Course  DBMS    JV   MMA     PY
0        NaN   NaN   NaN  100.0
1        NaN  56.0   NaN    NaN
2       32.0   NaN   NaN    NaN
3        NaN   NaN  10.0    NaN
4        NaN   NaN  67.0    NaN

       variable value
0       Course    PY
1       Course    JV
2       Course  DBMS
3       Course   MMA
4       Course   MMA
5          Fee   300
6          Fee   600
7          Fee    21
8          Fee   350
9          Fee    67
10  Complexity   100
11  Complexity    56
12  Complexity    32
13  Complexity    10
14  Complexity    67
```

9. Use Map, Filter and Reduce, Lambda functions using Pandas [Data Frame]

In [8]:
```
import pandas as pd
from functools import reduce
data = {
'Numbers': [1, 2, 3, 4, 5],
'Letters': ['A', 'B', 'C', 'D', 'E']
}
df = pd.DataFrame(data)
sq=df['Numbers'].map(lambda x: x**2)
ev=list(filter(lambda x: x % 2 == 0, df['Numbers']))
po = reduce(lambda x, y: x * y, df['Numbers'])
print("Dataframe:\n",df)
print("\nMap for Squaring:\n",sq)
print("\nReduce for product:\n", po)
```

```
Dataframe:
    Numbers Letters
0        1       A
1        2       B
2        3       C
3        4       D
4        5       E

Map for Squaring:
 0     1
1     4
2     9
3    16
4    25
Name: Numbers, dtype: int64

Reduce for product:
 120
```

10. Time series using Pandas (resample, shift operations)

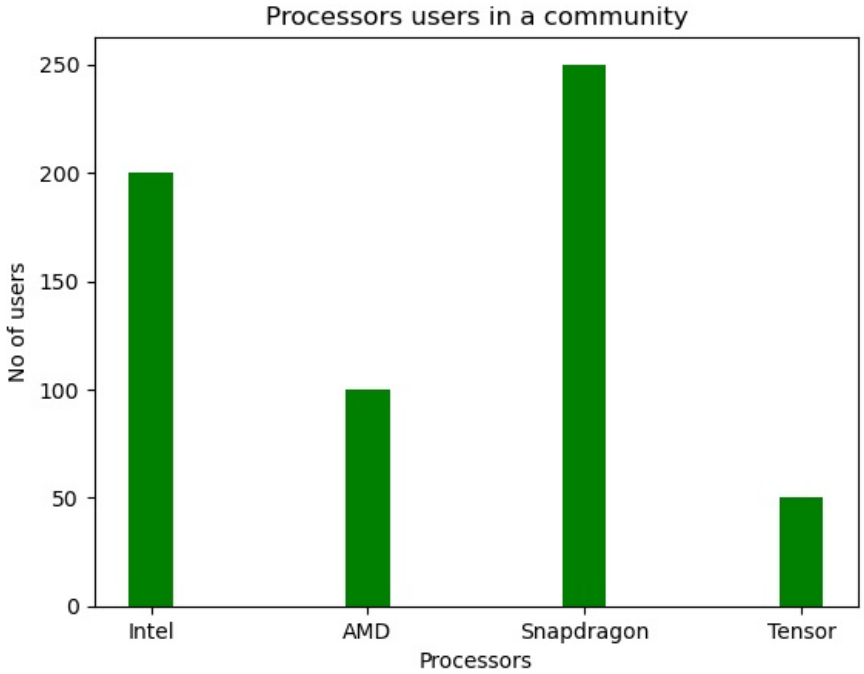
In [9]:
```
import numpy as n
import pandas as p
d=p.DataFrame(
{"date":p.date_range(start="2023-09-07",periods=5,freq="D"),"temp":n.random.randint(18,30,size=5)}
)
d["f"]=d["temp"].shift(1)
print("Shift:\n",d)
dfw=d.resample("ME",on="date").mean()
print("\nResampling:\n",dfw)
```

```
Shift:
         date  temp     f
0 2023-09-07    28   NaN
1 2023-09-08    25  28.0
2 2023-09-09    23  25.0
3 2023-09-10    26  23.0
4 2023-09-11    23  26.0

Resampling:
             temp     f
date
2023-09-30  25.0  25.5
```

11. Data visualization using Matplotlib (Bar chart,pie,line ,histogram,scatter)

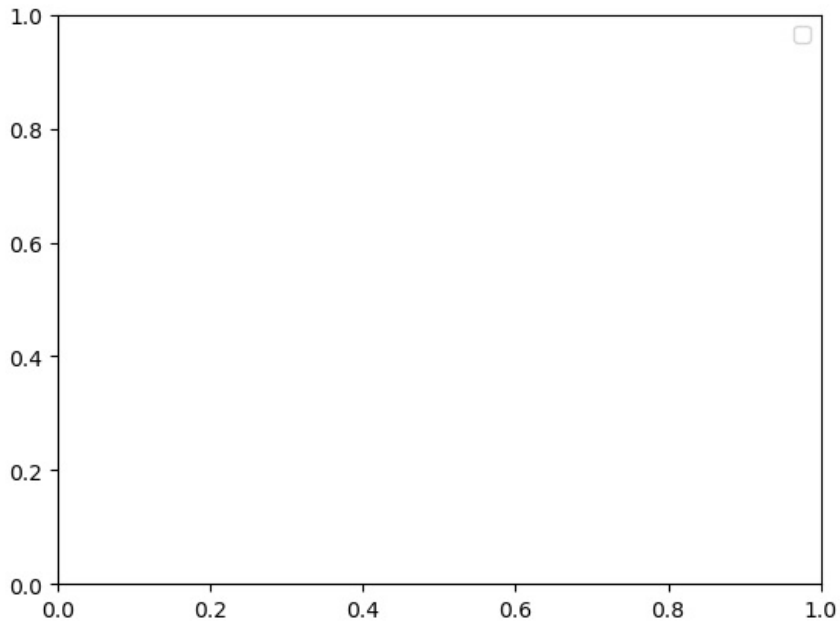
BAR CHART:

In [1]:
```
from matplotlib import pyplot as p
pro_na=["Intel","AMD","Snapdragon","Tensor"]
use=[200,100,250,50]
p.bar(pro_na,use,color='green',width=0.2)
p.xlabel("Processors"),p.ylabel("No of users")
p.title("Processors users in a community")
p.show()
```



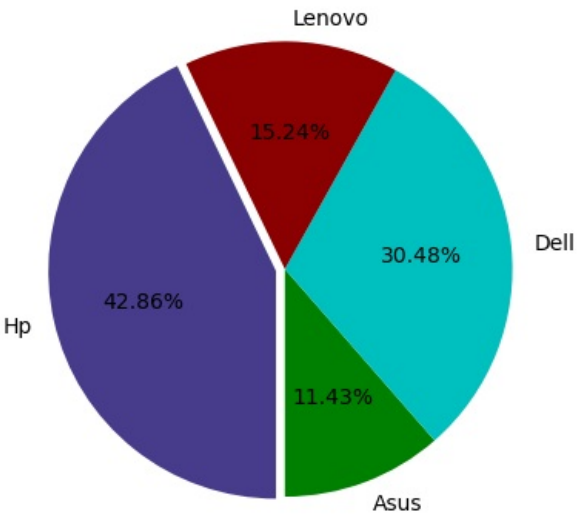

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignor
ed when legend() is called with no argument.
```

Out[1]:
```
<matplotlib.legend.Legend at 0x2852ba1c770>
```



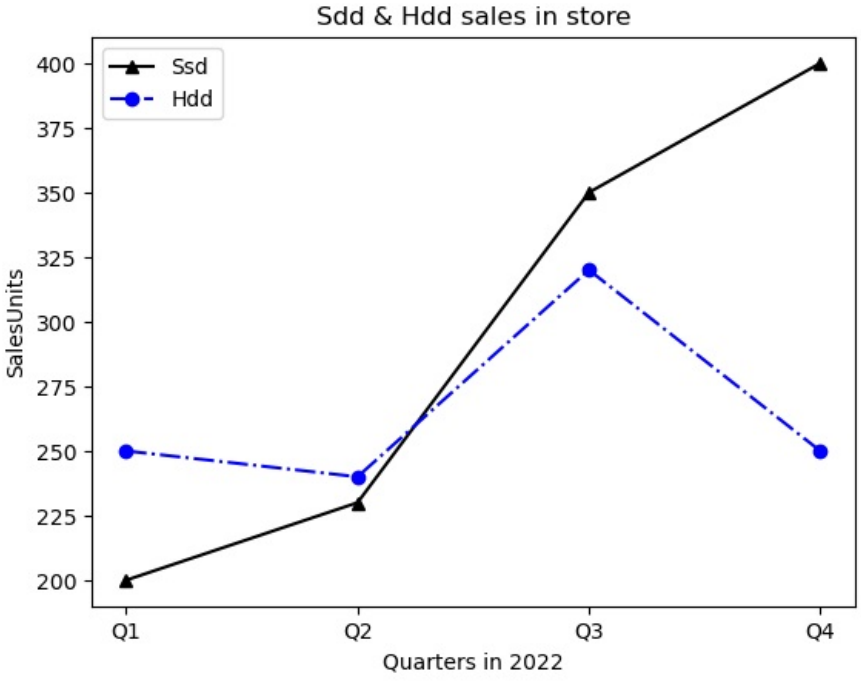
PIE CHART:

In [11]:
```
from matplotlib import pyplot as pi
us=[12,32,16,45]
la=["Asus","Dell","Lenovo","Hp"]
e=[0,0,0,0.04]
c=["g","c","#8B0000","#473C8B"]
pi.pie(us,labels=la,startangle=270,
      explode=e,colors=c,autopct='%1.2f%%')
pi.show()
```



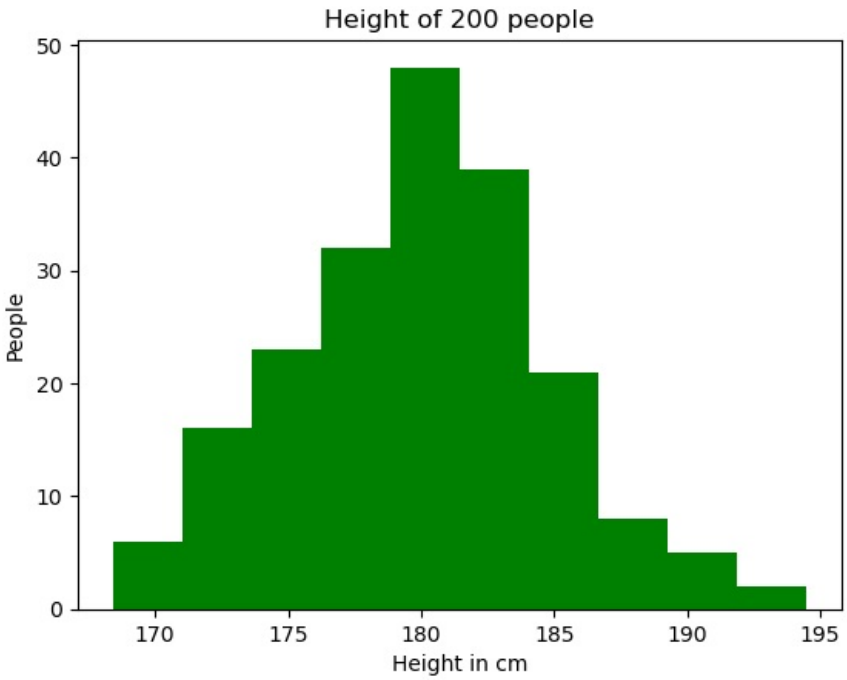
LINE GRAPH

In [12]:
```
from matplotlib import pyplot as p
Q=["Q1","Q2","Q3","Q4"]
ssd=[200,230,350,400]
hdd=[250,240,320,250]
p.plot(Q,ssd,'^-',color='black')
p.plot(Q,hdd,'o-.b')
p.xlabel("Quarters in 2022"),p.ylabel("SalesUnits")
p.title("Sdd & Hdd sales in store")
p.legend(['Ssd','Hdd'])
p.show()
```



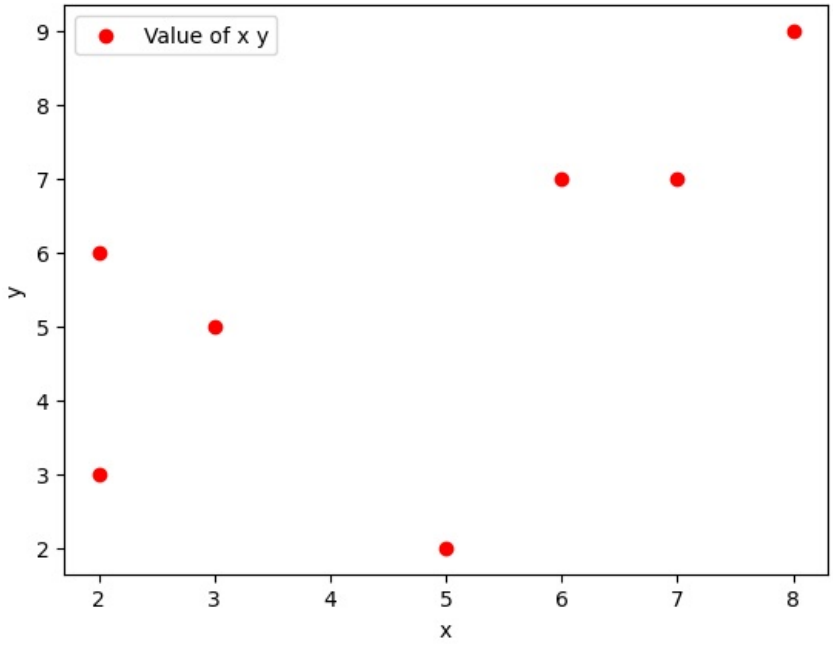
HISTOGRAM

In [13]:
```
from matplotlib import pyplot as p
import numpy as n
x=n.random.normal(180,5,200)
p.hist(x,color='g')
p.xlabel("Height in cm"),p.ylabel("People")
p.title("Height of 200 people")
p.show()
```



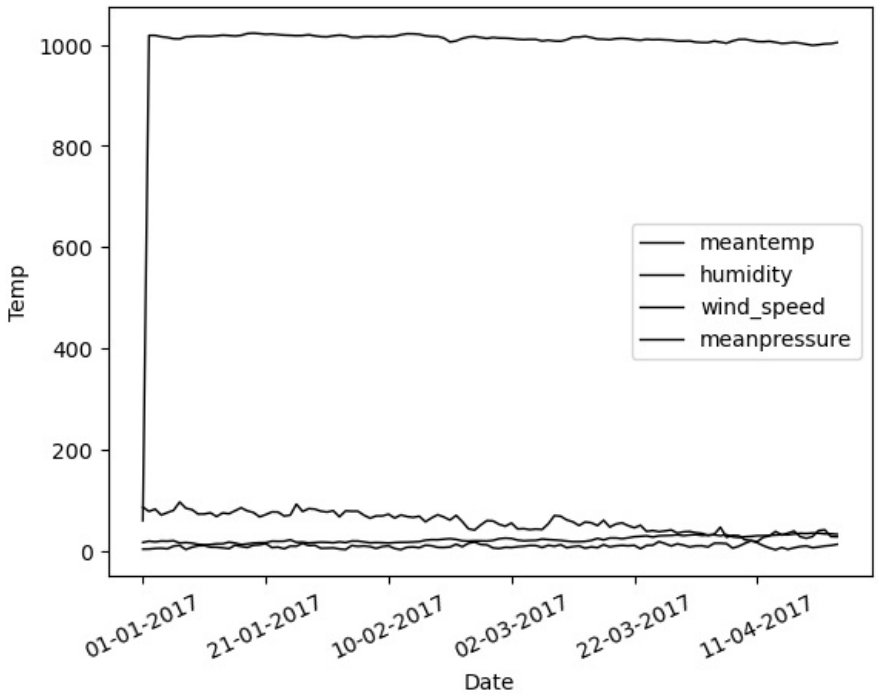
In [ ]:
```
SCATTER PLOT:
```

In [14]:
```
from matplotlib import pyplot as p
x=[2,6,8,7,3,2,5]
y=[6,7,9,7,5,3,2]
c=['k','b']
p.scatter(x,y,label='Value of x y',color='r')
p.xlabel('x')
p.ylabel('y')
p.legend()
p.show()
```



12. Visualization of time series data using temperature on different days

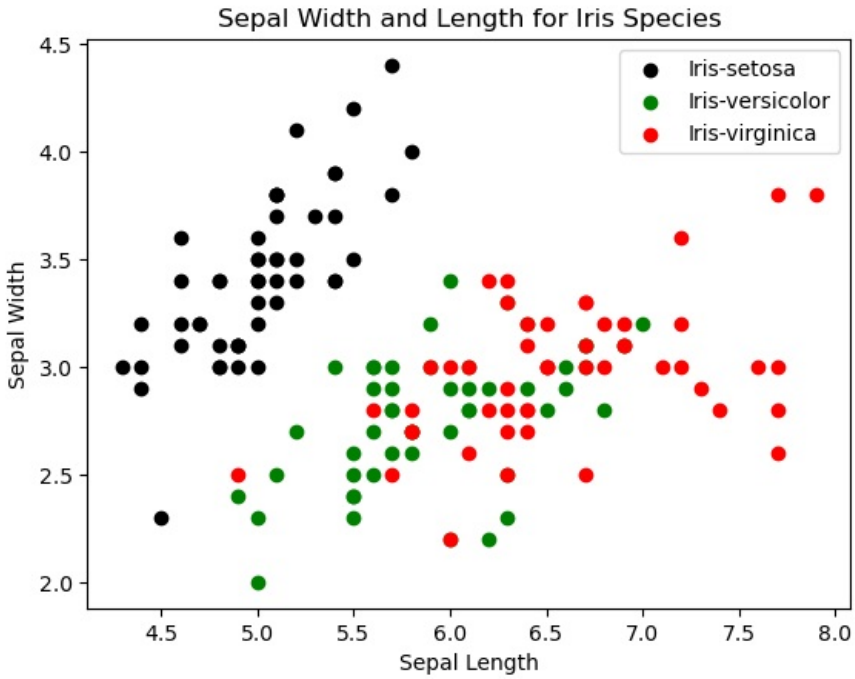
In [15]:
```
import pandas as pd
import matplotlib.pyplot as plt
t = pd.read_csv('D:\\DailyDelhiClimateTest.csv', parse_dates=['day'],
index_col='day')
a = t.plot(color='k', linewidth=1)
plt.xticks(rotation=25)
a.set_ylabel('Temp')
plt.xlabel('Date')
plt.show()
```



13. Visualization of Iris-dataset using Scatter Plot

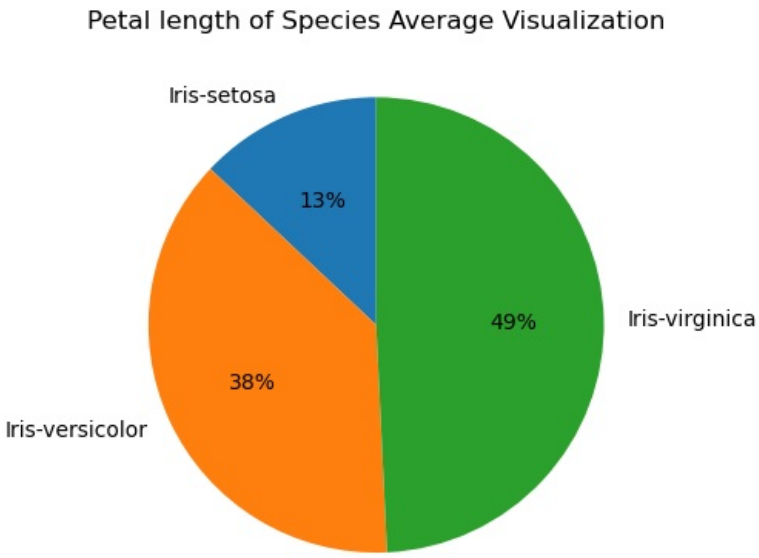
In [16]:

```
import pandas as pd
from matplotlib import pyplot as plt

# Correct the file path using raw string notation
file_path = r"C:\Users\DELL\OneDrive\Desktop\IRIS.csv"

# Read the CSV file
t = pd.read_csv(file_path)

# Define colors for species
species_colors = {
    'Iris-setosa': 'k',
    'Iris-versicolor': 'g',
    'Iris-virginica': 'r'
}

# Plotting
for species, color in species_colors.items():
    sl = t[t['species'] == species]['sepal_length']
    sw = t[t['species'] == species]['sepal_width']
    plt.scatter(sl, sw, color=color, label=species)

plt.legend()
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('Sepal Width and Length for Iris Species')
plt.show()
```



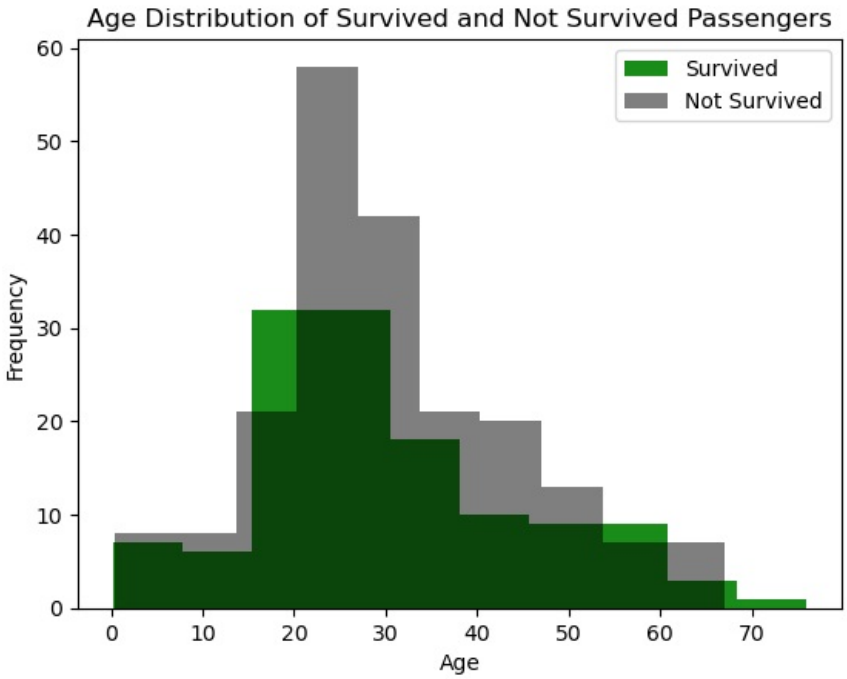
14. Visualization of Iris-dataset using Pie Chart

In [17]:
```
import pandas as pd
from matplotlib import pyplot as plt
t = pd.read_csv("C:\\Users\\DELL\\OneDrive\\Desktop\\IRIS.csv")
sv=t.groupby("species")["petal_length"].mean()
plt.pie(sv,labels=sv.index,startangle=90,autopct="%1.0f%%")
plt.title("Petal length of Species Average Visualization")
plt.show()
```



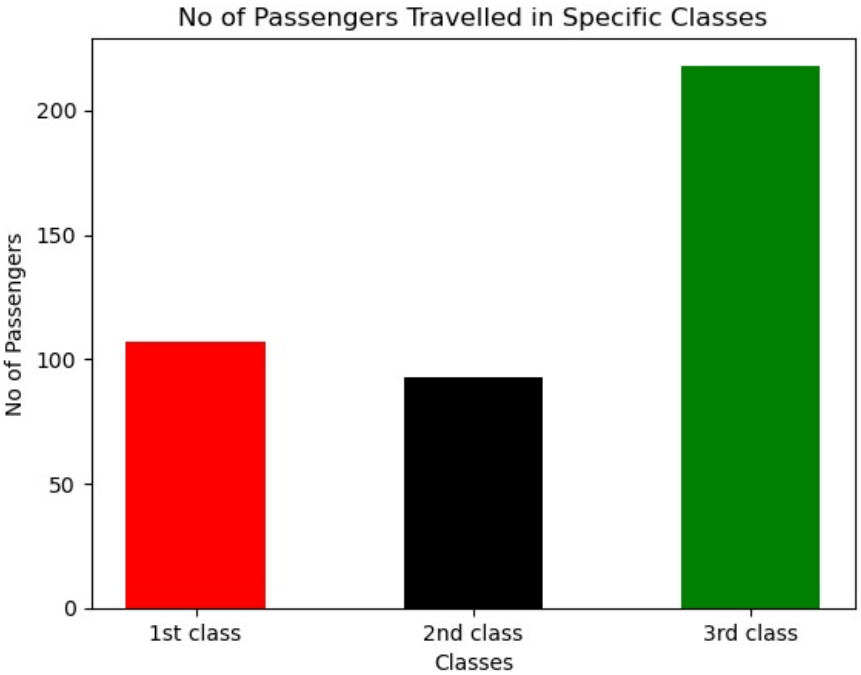
15. Visualization of Titanic-dataset using Histogram

In [18]:
```
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_csv('C:\\Users\\DELL\\OneDrive\\Desktop\\15data.csv')
age_survived = data[data['Survived'] == 1]['Age']
age_not_survived = data[data['Survived'] == 0]['Age']
plt.hist(age_survived, color='g', alpha=0.9, label='Survived')
plt.hist(age_not_survived, color='k', alpha=0.5,label='Not Survived')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Age Distribution of Survived and Not Survived Passengers')
plt.legend()
plt.show()
```



16. Visualization of Titanic-dataset using bar chart

In [19]:
```
import pandas as p
import matplotlib.pyplot as m
d=p.read_csv("C:\\Users\\DELL\\OneDrive\\Desktop\\15data.csv")
c=d["Pclass"].value_counts()
co=['g','r','k']
m.bar(c.index,c.values,color=co,width=0.5)
m.xticks([1,2,3],["1st class","2nd class","3rd class"])
m.xlabel("Classes");m.ylabel("No of Passengers");
m.title("No of Passengers Travelled in Specific Classes")
m.show()
```

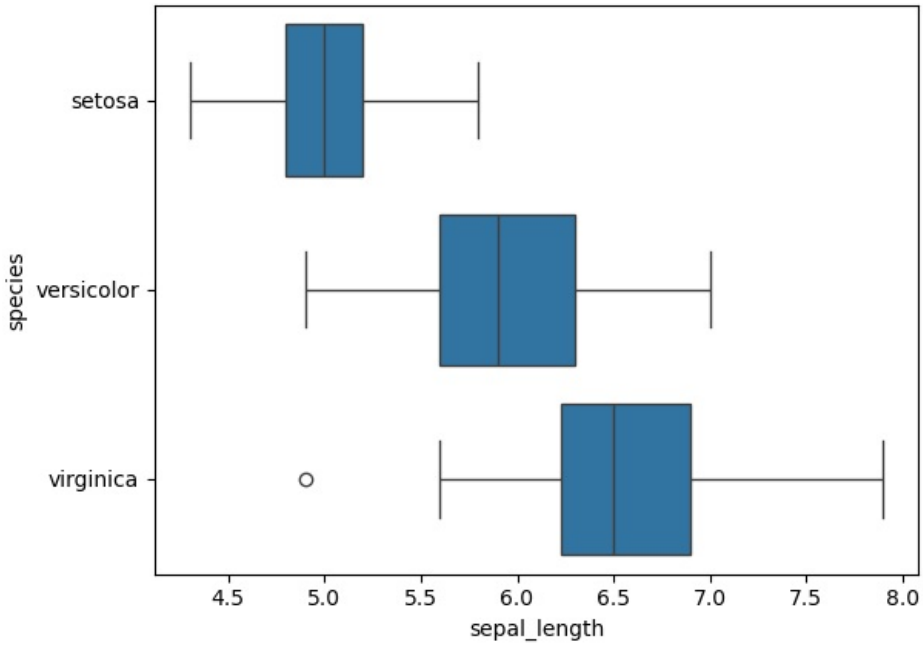


17. Visualize Employee dataset using Line graph [Represent Salary and Experience]

In [20]:
```
import pandas as p
import matplotlib.pyplot as m
d={"Ex":[1,1.3,1.5,2,2.2,2.9,3,3.2,3.2],"Salary":[1000,3000,6000,8000,10000,12000,18000,20000,30000]}
df=p.DataFrame(d)
m.plot(df["Ex"],df["Salary"],'^--',color='k')
m.xlabel("Experience in years");m.ylabel("Salary");m.title("Salary with Experience")
m.show()
```

18. Visualize Iris dataset using Box-Plot

In [22]:
```
import seaborn as s
import matplotlib.pyplot as p
d=s.load_dataset('iris')
s.boxplot(x=d['sepal_length'],y=d['species'])
p.show()
```



19 EXP

In [23]:
```
import pandas as p
import matplotlib.pyplot as m
d={
"First_name":["Aryan","Rohan","Riya","Yash","Siddhant"],
"Last_name":["Singh","Agarwal","Shah","Bhatia","Khanna"],
"Type":["Full-Time","Itern","Full-Time","Part-Time","Full-Time"],
"Dept":["Administration","Technical","Administration","Technical","Management"],
'YoE':[2,3,5,7,6],"Salary":[20000,5000,10000,10000,20000]
}
df=p.DataFrame(d)
av=df.pivot_table(index=['Dept', 'Type'], values='Salary', aggfunc='mean')
print("Average Salary from ecah dept:\n",av)
sm=df.pivot_table(index=['Type'], values='Salary', aggfunc=['sum', 'mean',
'count'])
sm.columns=['Total Salary', 'Mean Salary', 'Number of Employees']
print("\nSum and Mean of:\n",sm)
st=df.pivot_table(values='Salary', index='Type',aggfunc='std')
print("\nStandard Deviation:\n",st)
```

```
Average Salary from ecah dept:
                           Salary
Dept           Type
Administration Full-Time  15000.0
Management     Full-Time  20000.0
Technical      Itern       5000.0
               Part-Time  10000.0

Sum and Mean of:
            Total Salary   Mean Salary  Number of Employees
Type
Full-Time         50000  16666.666667                    3
Itern              5000   5000.000000                    1
Part-Time         10000  10000.000000                    1

Standard Deviation:
                 Salary
Type
Full-Time  5773.502692
```

20 EXP

In [24]:
```
import pandas as pd
a = pd.Series([10, 20, 30, 40, 50])
b = pd.Series([40, 50, 60, 70, 80])
print("Series A:")
print(a)
print("\nSeries B:")
print(b)
non_com = a[~a.isin(b)].tolist() + b[~b.isin(a)].tolist()
print("Items not common to both Series:")
print(non_com)
print("\nSmallest element in Series A:\n", a.min())
print("\nLargest element in Series A:\n",a.max())
print("\nSum of Series B:\n", b.sum())
print("\nAverage of Series A:\n",a.mean())
print("\nMedian of Series B:\n", a.median())
```

```
Series A:
0    10
1    20
2    30
3    40
4    50
dtype: int64

Series B:
0    40
1    50
2    60
3    70
4    80
dtype: int64
Items not common to both Series:
[10, 20, 30, 60, 70, 80]

Smallest element in Series A:
 10

Largest element in Series A:
 50

Sum of Series B:
 300

Average of Series A:
 30.0

Median of Series B:
 30.0
```

21 EXP

In [25]:
```
import pandas as pd
da={
"mpg":[18,15,18,16,17],"cylinders":[8,8,6,4,8],"displacement":[307,350,318,
304,302],
"horsepower":[130,165,150,150,140],"weigth":[3504,3693,3436,3433,3449],
"acceleration":[12.0,11.5,11.0,12.0,10.5],"model year":[70,71,70,80,70],
"origin":[1,1,1,1,1],"car name":["cheverlot","buick","plymoth","amc","ford"]
}
df=pd.DataFrame(da)
sa=df.describe()
```

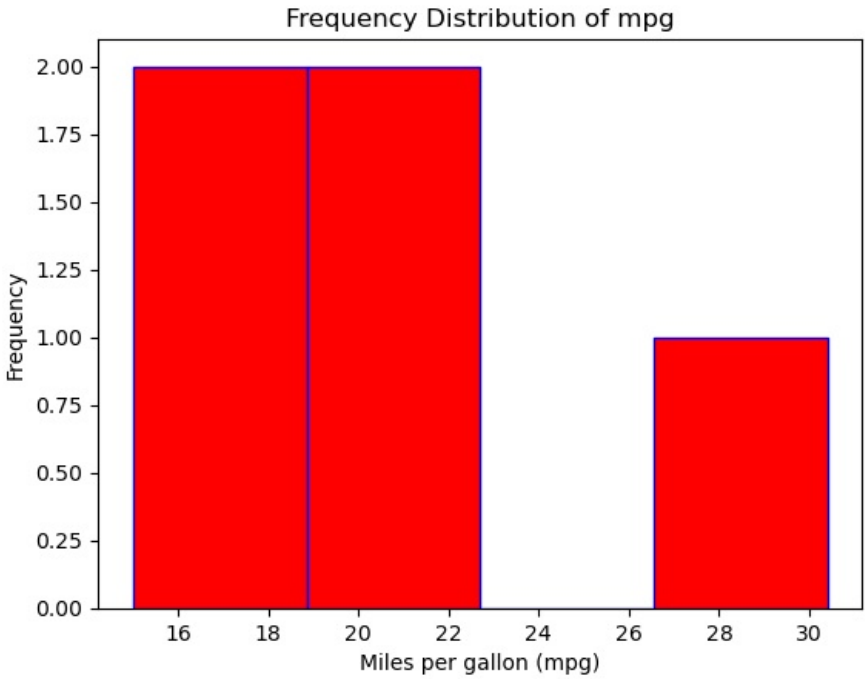
```
ei=df[df["cylinders"]==8]
ye = df.groupby('model year')["model year"].count()
print("Satistical:\n",sa)
print("\n8 cylinders:\n",ei)
print("\nBy year:\n",ye)
```

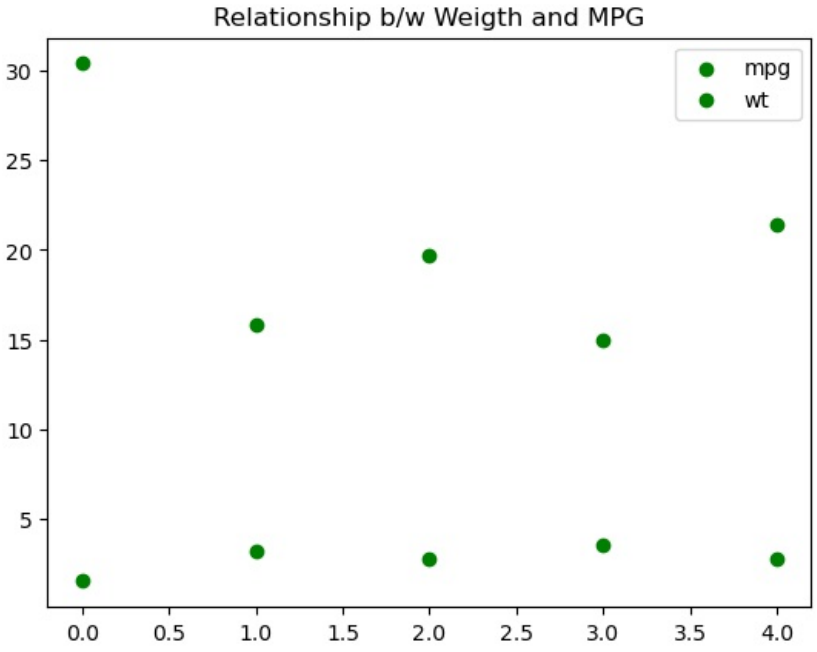
```
Satistical:
             mpg  cylinders  displacement  horsepower       weigth  \
count   5.00000   5.000000      5.000000    5.000000     5.000000
mean   16.80000   6.800000    316.200000  147.000000  3503.000000
std     1.30384   1.788854     19.879638   13.038405   110.006818
min    15.00000   4.000000    302.000000  130.000000  3433.000000
25%    16.00000   6.000000    304.000000  140.000000  3436.000000
50%    17.00000   8.000000    307.000000  150.000000  3449.000000
75%    18.00000   8.000000    318.000000  150.000000  3504.000000
max    18.00000   8.000000    350.000000  165.000000  3693.000000

       acceleration  model year  origin
count       5.00000     5.00000     5.0
mean       11.40000    72.20000     1.0
std         0.65192     4.38178     0.0
min        10.50000    70.00000     1.0
25%        11.00000    70.00000     1.0
50%        11.50000    70.00000     1.0
75%        12.00000    71.00000     1.0
max        12.00000    80.00000     1.0

8 cylinders:
    mpg  cylinders  displacement  horsepower  weigth  acceleration  model year  \
0   18           8           307         130    3504          12.0          70
1   15           8           350         165    3693          11.5          71
4   17           8           302         140    3449          10.5          70

   origin   car name
0       1  cheverlot
1       1      buick
4       1       ford

By year:
 model year
70    3
71    1
80    1
Name: model year, dtype: int64
```

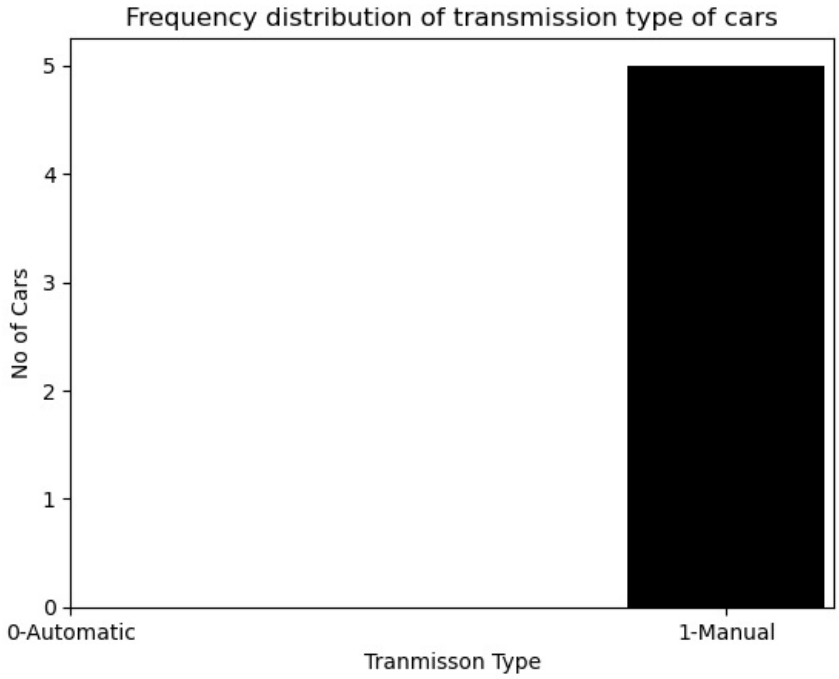
22. Data from an online platform has been collected. This data contains fuel consumption

and 11 aspects of automobile design and performance for 32 automobiles. Variable description is given below.Dataset - 'mtcars.csv

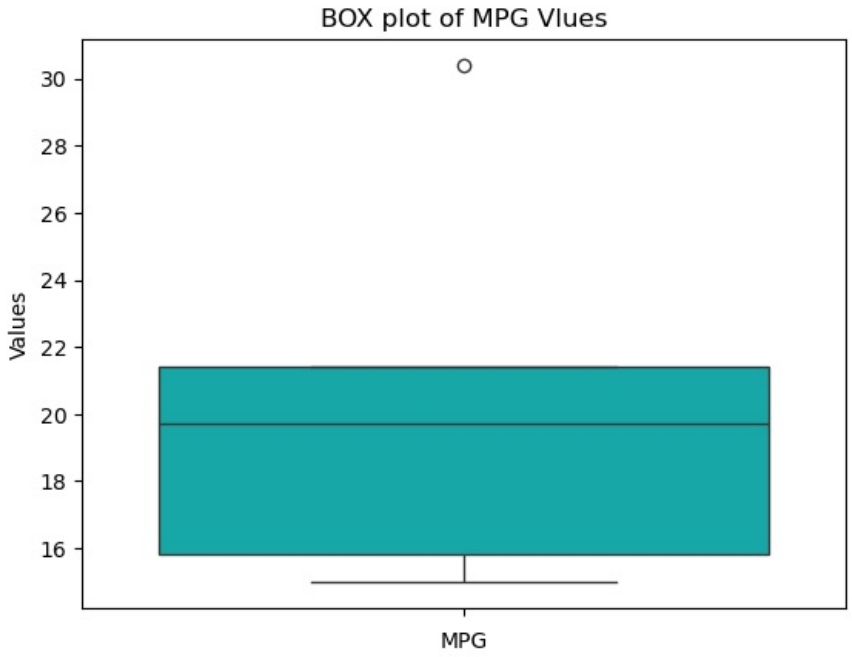
In [2]:
```
import pandas as p
import matplotlib.pyplot as m
import seaborn as s
# data as 32 Elements
data=p.read_csv("D:\\emty.csv")
# HISTOGRAM
mpg=data['mpg']
m.hist(mpg,bins='auto',color='r',edgecolor='b')
m.xlabel('Miles per gallon (mpg)');m.ylabel('Frequency')
m.title('Frequency Distribution of mpg')
m.show()
# SCATTER
wt=data['wt']
iv=range(len(data))
m.scatter(iv,mpg,color='g',label='mpg')
m.scatter(iv,wt,color='g',label='wt')
m.title("Relationship b/w Weigth and MPG")
m.legend()
m.show()
# BAR PLOT
c=data['am'].value_counts()
co=['k','g']
m.bar(c.index,c.values,color=co,width=0.3)
m.xticks([0,1],['0-Automatic','1-Manual'])
m.xlabel("Tranmisson Type");m.ylabel("No of Cars")
m.title("Frequency distribution of transmission type of cars")
m.show()
# BOX PLOT
s.boxplot(mpg,color='c')
m.xlabel("MPG");m.ylabel("Values")
m.title("BOX plot of MPG Vlues")
```







Out[2]: Text(0.5, 1.0, 'BOX plot of MPG Vlues')



23. Ramesh decides to walk 10000 steps every day to combat the effect that lockdown

has had on his body's agility, mobility, flexibility and strength. Consider the following data from fitness tracker over a period of 10 days
1.Code to add 1000 steps to all the observations 2.Code to find out the days on which Ramesh walked more than 7000 steps

In [27]:
```
import pandas as p
import numpy as n
d={"Day":[1,2,3,4,5,6,7,8,9,10],
"Steps":[4335,9552,7332,4504,5335,7552,8332,6504,8965,7689]}
dp=p.DataFrame(d)
dp["+1000 Steps"]=dp["Steps"]+1000
fi=dp[dp["+1000 Steps"]>7000]["Day"]
print("DataFrame:\n",dp)
print("\nDays on which Steps were >7000:\n",fi)
```

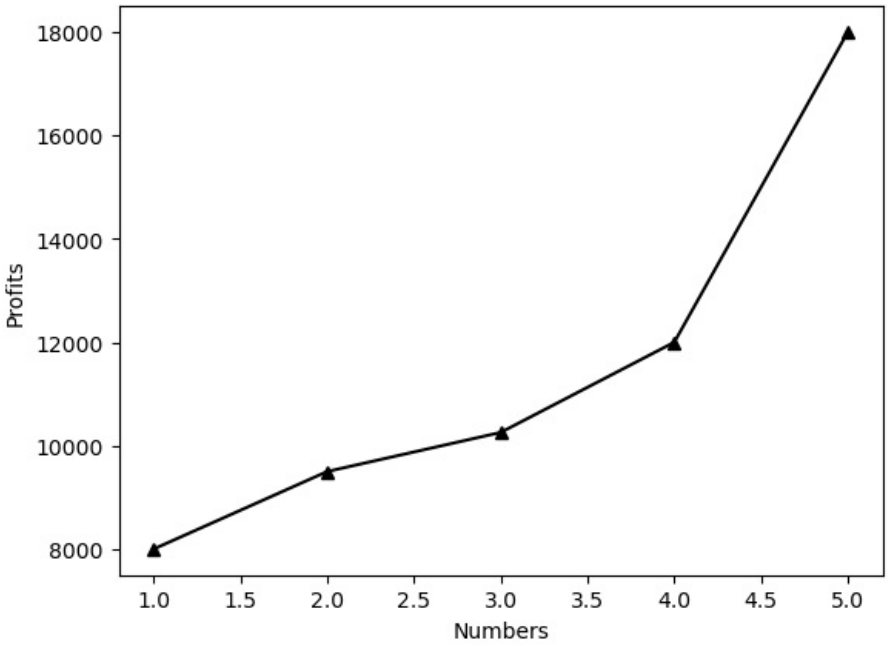
```
DataFrame:
    Day  Steps  +1000 Steps
0    1   4335         5335
1    2   9552        10552
2    3   7332         8332
3    4   4504         5504
4    5   5335         6335
5    6   7552         8552
6    7   8332         9332
7    8   6504         7504
8    9   8965         9965
9   10   7689         8689

Days on which Steps were >7000:
 1     2
2     3
5     6
6     7
7     8
8     9
9    10
Name: Day, dtype: int64
```

24 EXP

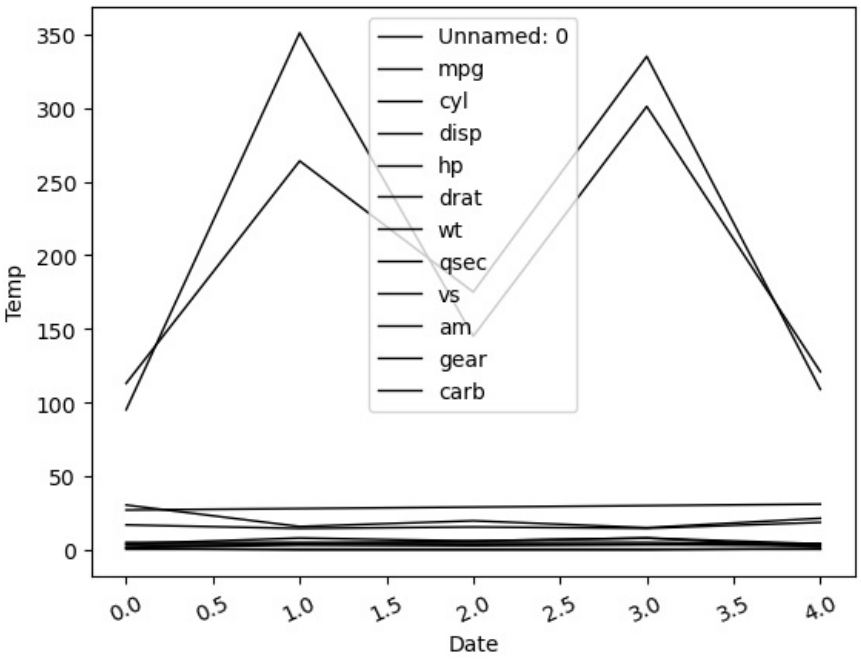
In [28]:
```
import numpy as n
import pandas as p
import matplotlib.pyplot as m
da={
'n':[1,2,3,4,5],'Pencil':[300,350,400,500,520],'TextBooks':[250,350,400,420,500],
'Draw':[100,200,200,250,300],'Total':[800,1000,1320,1510,2000],"Profits":[8000,9500,10256,12000,18000]
}
df=p.DataFrame(da)
sta=df.describe()
print("Statistics:\n",sta)
su=df['Profits'].sum()
print("\nSum of Profits:\n",su)
mi=df.isna()
print("\nMissing values:\n",mi)
print("\nMaximum Value:\n",df['Draw'].max())
```

```
m.plot(df['n'],df['Profits'],'^-',color='k')
m.xlabel("Numbers");m.ylabel("Profits")
m.show()
```

```
Statistics:
              n      Pencil   TextBooks        Draw         Total       Profits
count  5.000000    5.000000    5.000000    5.000000     5.000000      5.000000
mean   3.000000  414.000000  384.000000  210.000000  1326.000000  11551.200000
std    1.581139   94.762862   92.357999   74.161985   466.669048   3882.152393
min    1.000000  300.000000  250.000000  100.000000   800.000000   8000.000000
25%    2.000000  350.000000  350.000000  200.000000  1000.000000   9500.000000
50%    3.000000  400.000000  400.000000  200.000000  1320.000000  10256.000000
75%    4.000000  500.000000  420.000000  250.000000  1510.000000  12000.000000
max    5.000000  520.000000  500.000000  300.000000  2000.000000  18000.000000

Sum of Profits:
 57756

Missing values:
       n  Pencil  TextBooks   Draw  Total  Profits
0  False   False      False  False  False    False
1  False   False      False  False  False    False
2  False   False      False  False  False    False
3  False   False      False  False  False    False
4  False   False      False  False  False    False

Maximum Value:
 300
```



In [1]:
```
# program 12
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

t = pd.read_csv('D:\\emty.csv')
a = t.plot(color='k', linewidth=1)

plt.xticks(rotation=25)

a.set_ylabel('Temp')

plt.xlabel('Date')

plt.show()
```



In [ ]: