In [5]:
```
#4
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+=i
a=t
print("\n"+str(a))
print("Time Taken by iterative sum:",end="")
%timeit a
```

```
6
Time taken  to vectorized sum:
6.7 µs ± 484 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)

6
Time Taken by iterative sum:25.1 ns ± 3.17 ns per loop (mean ± std. dev. of 7 runs, 10,000,000 loops each)
```

In [7]:
```
# program 5
import numpy as np
da=[60,8,7,5,34,78]
d=np.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))
```

```
[np.int64(3600), np.int64(64), np.int64(49), np.int64(25), np.int64(1156), np.int64(6084)]
[np.int64(60), np.int64(8), np.int64(7), np.int64(5), np.int64(34), np.int64(78)]
192
```

In [9]:
```
# program 6
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
```

In [11]:
```
# program 7
import pandas as py

t={
    'Course':["PY","JV","DBMS","MMA","MMA"],
    'Fee':[300,600,21,350,67],
    'Complexity':[100,56,32,10,67]
}

d=py.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
```

In [13]:
```python
# program 8
import pandas as py

t={
    'Course':["PY","JV","DBMS","MMA","MMA"],
    'Fee':[300,600,21,350,67],
    'Complexity':[100,56,32,10,67]
}

d=py.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
```

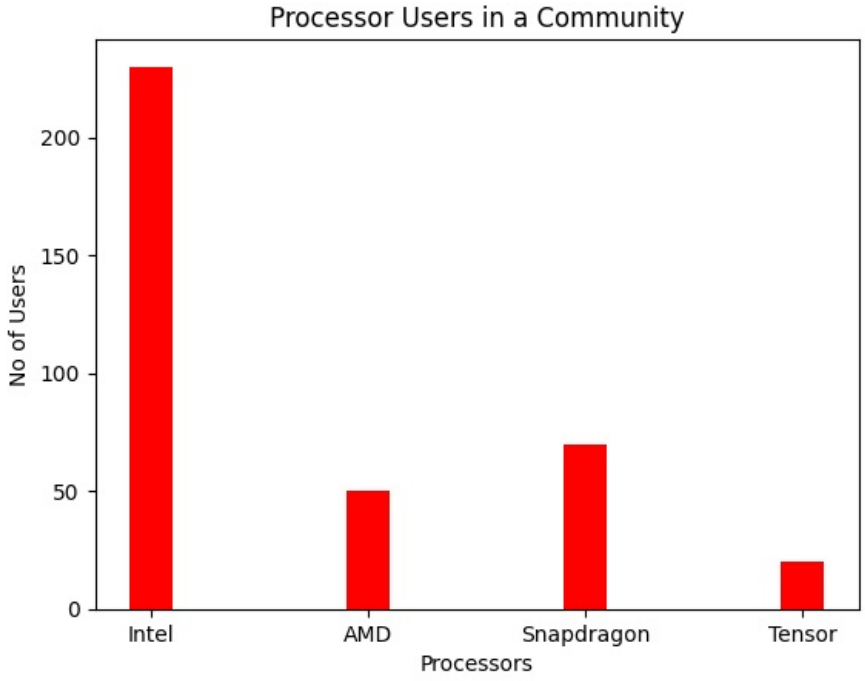
In [15]:
```python
# program 9
import pandas as pd
from functools import reduce

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
```

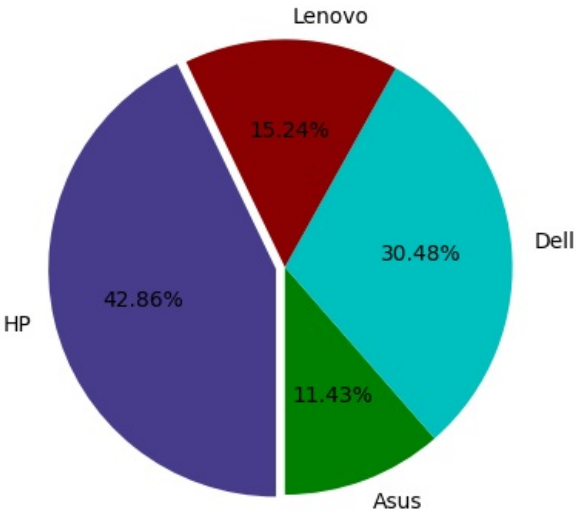
In [19]:
```python
# program 10
import numpy as n
import pandas as p

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    18   NaN
1 2023-09-08    26  18.0
2 2023-09-09    18  26.0
3 2023-09-10    23  18.0
4 2023-09-11    22  23.0

Resampling:
             temp      f
date
2023-09-30  21.4  21.25
```

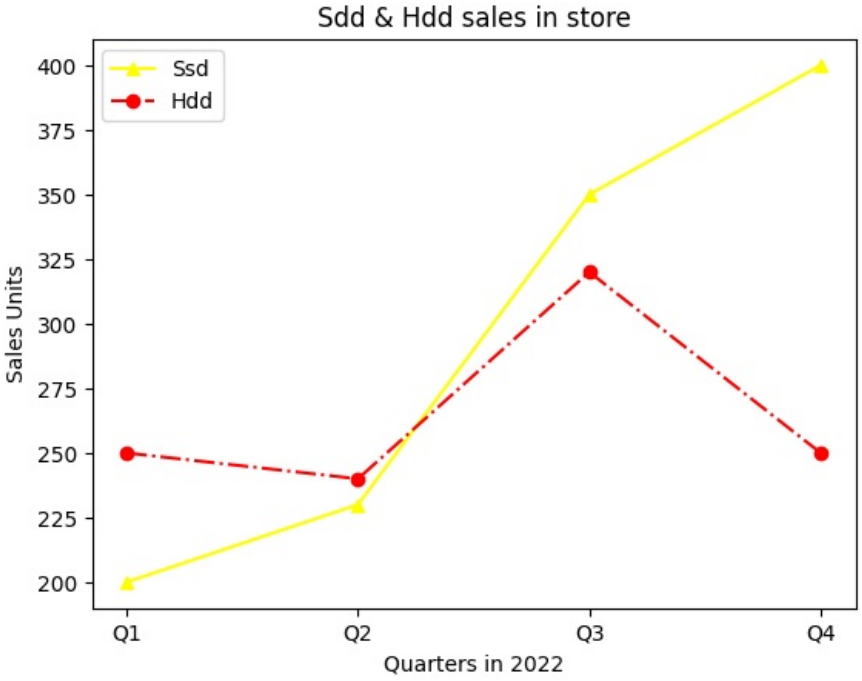
In [2]:
```python
# program 11
from matplotlib import pyplot as plt

pro_na=["Intel","AMD","Snapdragon","Tensor"]
use=[230,50,70,20]

plt.bar(pro_na,use,color='red',width=0.2)
plt.xlabel("Processors"),plt.ylabel("No of Users")
plt.title("Processor Users in a Community")
plt.show()
```



In [4]:
```
from matplotlib import pyplot as plt
us=[12,32,16,45]
la=["Asus","Dell","Lenovo","HP"]
e=[0,0,0,0.04]
c=["g","c","#8B0000","#473C8B"]
plt.pie(us,labels=la,startangle=270,
explode=e,colors=c,autopct='%1.2f%%')

plt.show()
```

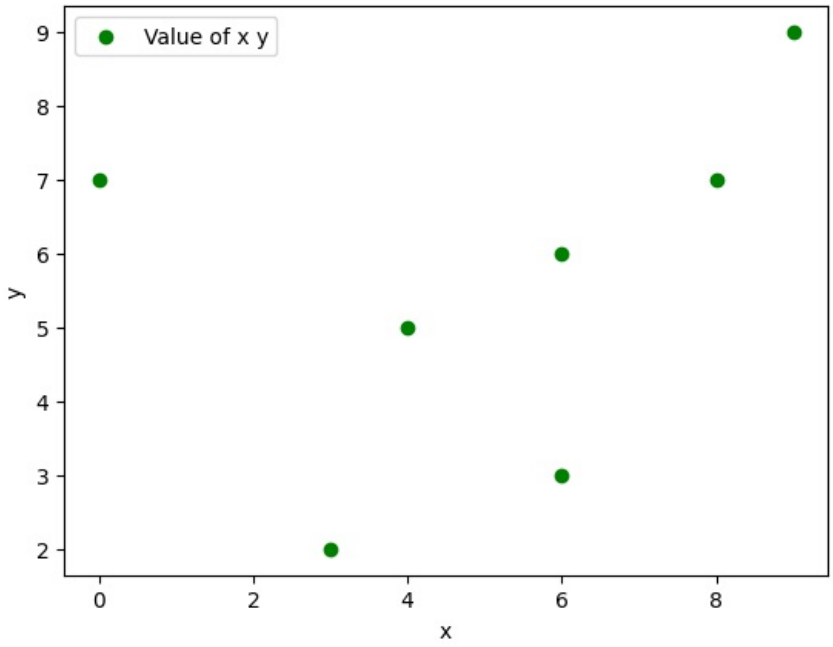


In [1]:
```
# 11
from matplotlib import pyplot as p
Q=["Q1","Q2","Q3","Q4"]
ssd=[200,230,350,400]
hdd=[250,240,320,250]
p.plot(Q,ssd,'^-',color='yellow')
p.plot(Q,hdd,'o-.r')
p.xlabel("Quarters in 2022"),p.ylabel("Sales Units")
p.title("Sdd & Hdd sales in store")
p.legend(['Ssd','Hdd'])
p.show()
```



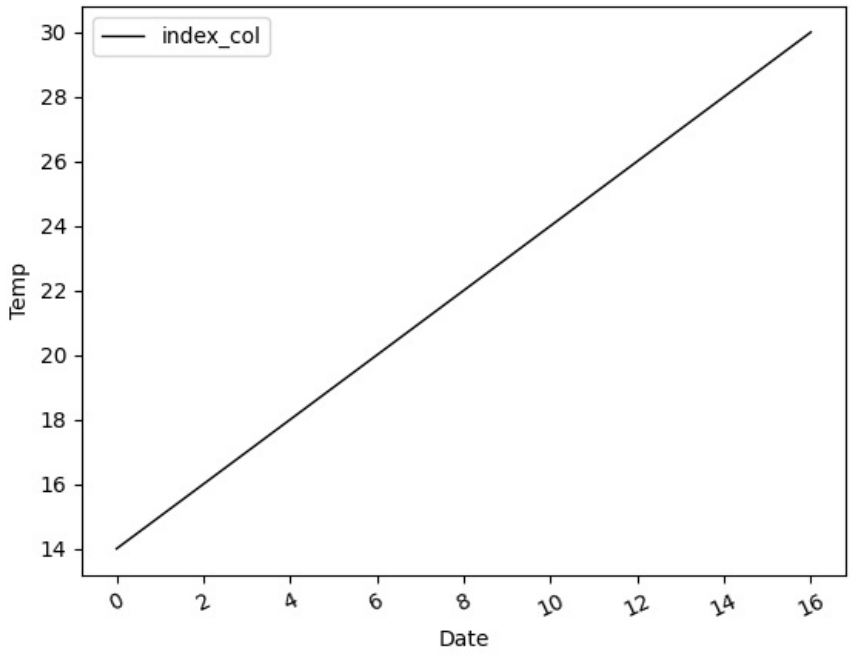
In [10]:
```
# histogram 11
from matplotlib import pyplot as p
import numpy as n
x=n.random.normal(180,5,200)
p.hist(x,color='k')
p.xlabel("Height in cm"),p.ylabel("People")
p.title("Height of 200 people")
p.show()
```

In [14]:
```
# scatter plot 11
from matplotlib import pyplot as p
x=[6,8,9,0,4,6,3]
y=[6,7,9,7,5,3,2]
c=['k','b']
p.scatter(x,y,label='Value of x y',color='g')
p.xlabel('x')
p.ylabel('y')
p.legend()
p.show()
```



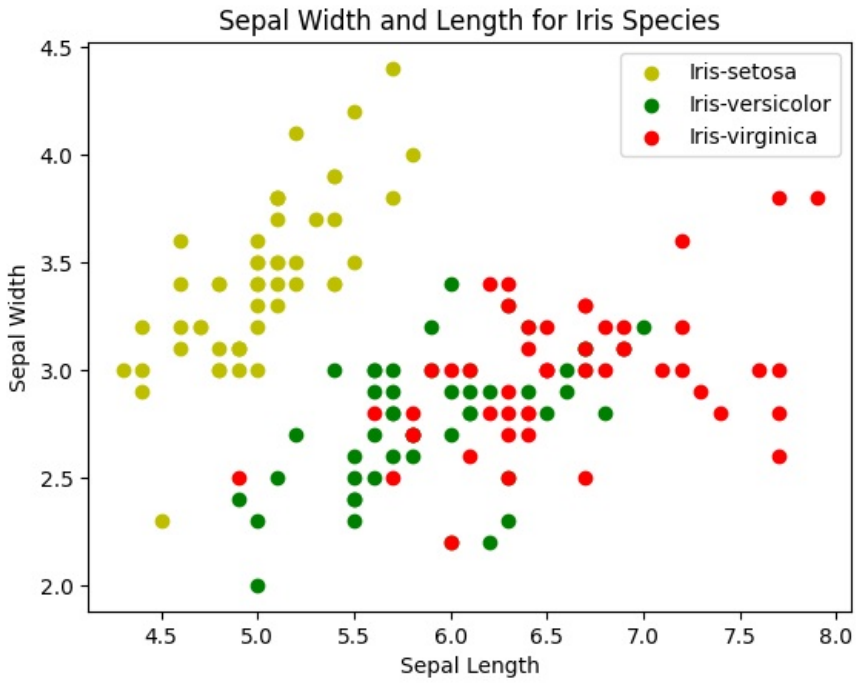
In [44]:
```
# program 12
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

t = pd.read_csv('sandya1.csv')
a = t.plot(color='k', linewidth=1)

plt.xticks(rotation=25)

a.set_ylabel('Temp')

plt.xlabel('Date')
```

```
plt.show()
```



In [56]:

```
# program 13
import pandas as pd
from matplotlib import pyplot as plt
t = pd.read_csv("IRIS.csv")

species_colors = {
    'Iris-setosa': 'y','Iris-versicolor': 'g','Iris-virginica': 'r'
}

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()
```

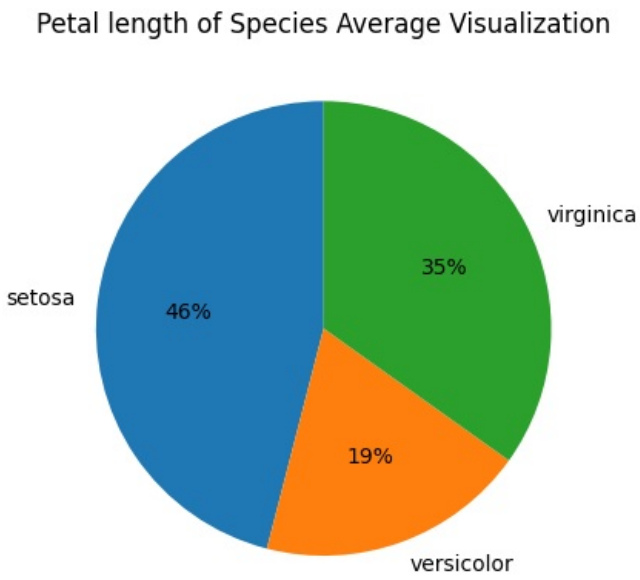


In [32]:

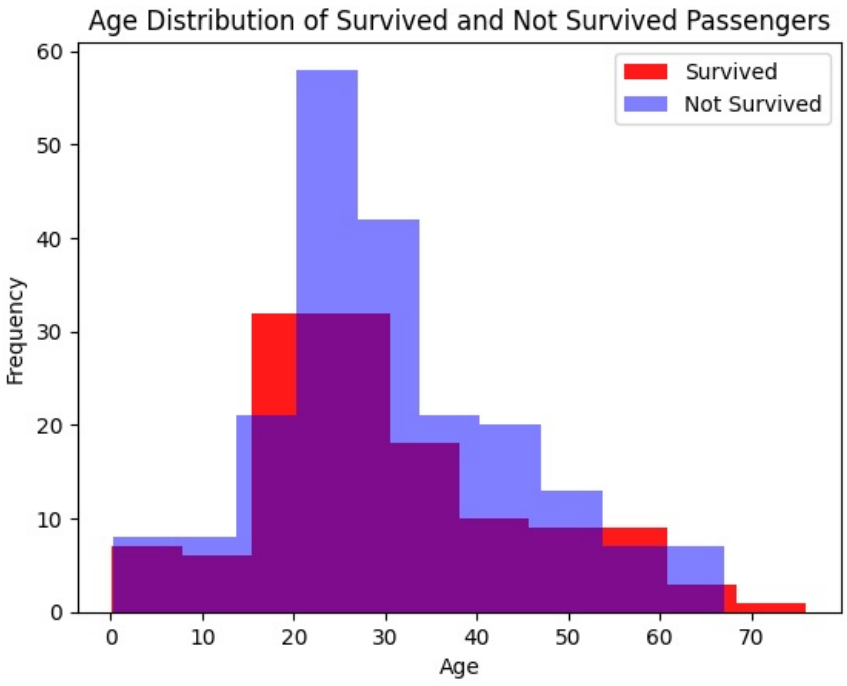
```
# program 14
#14. Visualization of Iris-dataset using Pie Chart

import pandas as pd
from matplotlib import pyplot as plt

t = pd.read_csv("iris14.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
```

Out[32]: <function matplotlib.pyplot.show(close=None, block=None)>

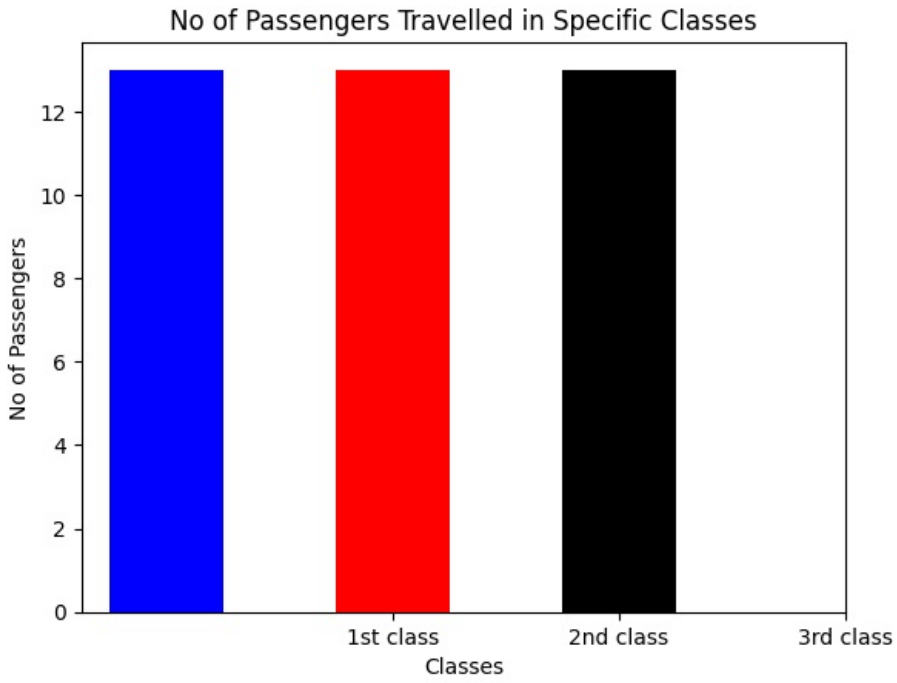


In [7]:
```
#  15. Visualization of Titanic-dataset using Histogram
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv('ve.csv.csv')
age_survived = data[data['Survived'] == 1]['Age']
age_not_survived = data[data['Survived'] == 0]['Age']

plt.hist(age_survived,  color='r', alpha=0.9, label='Survived')
plt.hist(age_not_survived, color='b', 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()
```

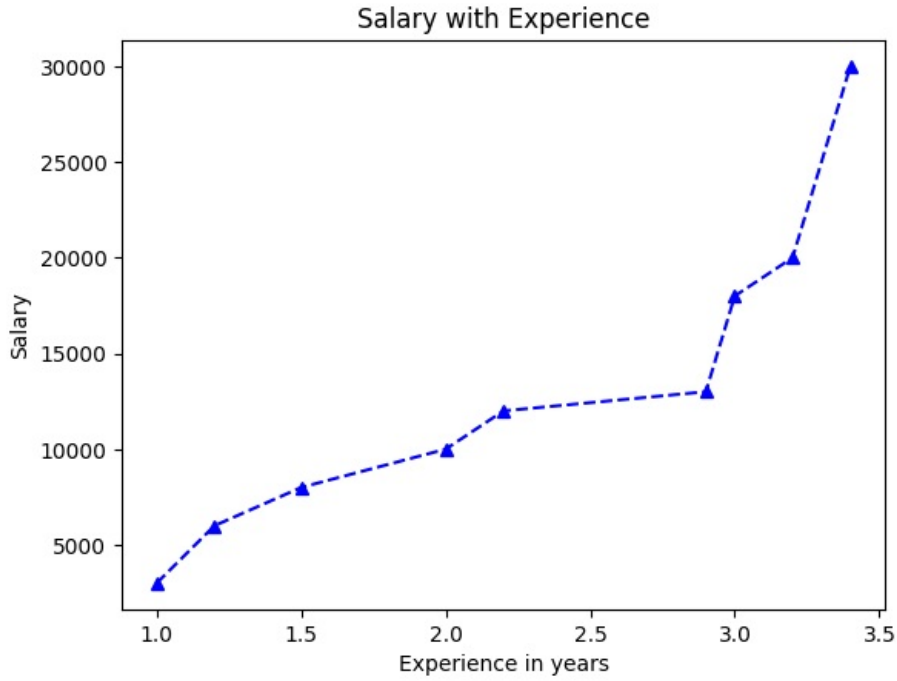


In [2]:
```
# program 16
import pandas as p
import matplotlib.pyplot as m
d=p.read_csv("pa16.csv")

c=d["Pclass"].value_counts()
```

```
co=['b','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()
```



In [40]:
```
# program 17
import pandas as p
import matplotlib.pyplot as m
d={"Ex":[1,1.2,1.5,2,2.2,2.9,3,3.2,3.4],
   "Salary":[3000,6000,8000,10000,12000,13000,18000,20000,30000]}

df=p.DataFrame(d)
m.plot(df["Ex"],df["Salary"],'^--',color='b')
m.xlabel("Experience in years");m.ylabel("Salary");m.title("Salary with Experience")
m.show()
```



In [88]:
```
# program 18
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()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[88], line 2
      1 # program 18
----> 2 import seaborn as s
      3 import matplotlib.pyplot as p
      4 d=s.load_dataset('iris')

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\__init__.py:9
      7 from .categorical import *  # noqa: F401,F403
      8 from .distributions import *  # noqa: F401,F403
----> 9 from .matrix import *  # noqa: F401,F403
     10 from .miscplot import *  # noqa: F401,F403
     11 from .axisgrid import *  # noqa: F401,F403

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\matrix.py:11
      9 import pandas as pd
     10 try:
---> 11     from scipy.cluster import hierarchy
     12     _no_scipy = False
     13 except ImportError:

File ~\anaconda3\Lib\site-packages\scipy\cluster\__init__.py:27
      1 """
      2 =========================================
      3 Clustering package (:mod:`scipy.cluster`)
   (...)
     23
     24 """
     25 __all__ = ['vq', 'hierarchy']
---> 27 from . import vq, hierarchy
     29 from scipy._lib._testutils import PytestTester
     30 test = PytestTester(__name__)

File ~\anaconda3\Lib\site-packages\scipy\cluster\vq.py:74
     70 from scipy._lib._array_api import (
     71     _asarray, array_namespace, size, atleast_nd, copy, cov
     72 )
     73 from scipy._lib._util import check_random_state, rng_integers
---> 74 from scipy.spatial.distance import cdist
     76 from . import _vq
     78 __docformat__ = 'restructuredtext'

File ~\anaconda3\Lib\site-packages\scipy\spatial\__init__.py:110
      1 """
      2 =============================================================
      3 Spatial algorithms and data structures (:mod:`scipy.spatial`)
   (...)
    107    QhullError
    108 """  # noqa: E501
--> 110 from ._kdtree import *
    111 from ._ckdtree import *
    112 from ._qhull import *

File ~\anaconda3\Lib\site-packages\scipy\spatial\_kdtree.py:4
      1 # Copyright Anne M. Archibald 2008
      2 # Released under the scipy license
      3 import numpy as np
----> 4 from ._ckdtree import cKDTree, cKDTreeNode
      6 __all__ = ['minkowski_distance_p', 'minkowski_distance',
      7            'distance_matrix',
      8            'Rectangle', 'KDTree']
     11 def minkowski_distance_p(x, y, p=2):

File _ckdtree.pyx:1, in init scipy.spatial._ckdtree()

ValueError: numpy.dtype size changed, may indicate binary incompatibility. Expected 96 from C header, got 88 fro
m PyObject
```

In [86]:

```
# program 19
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":[30000,8000,10000,10000,50000]
}

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("Sum and Mean of:",sm)

st=df.pivot_table(values='Salary', index='Type',aggfunc='std')
print("Standard Deviation:",st)
```

```
Average Salary from ecah dept:
                            Salary
Dept           Type
Administration Full-Time  20000.0
Management     Full-Time  50000.0
Technical      Itern       8000.0
               Part-Time  10000.0
Sum and Mean of:            Total Salary  Mean Salary  Number of Employees
Type
Full-Time         90000      30000.0                    3
Itern              8000       8000.0                    1
Part-Time         10000      10000.0                    1
Standard Deviation:             Salary
Type
Full-Time  20000.0
```

In [3]:
```
# program 20
import pandas as pd
a = pd.Series([10, 20, 30, 40, 50])
b = pd.Series([40, 50, 60, 70, 80])
print("Series A:",a)
print("\nSeries B:",b)

non_com = a[~a.isin(b)].tolist() + b[~b.isin(a)].tolist()
print("Items not common to both Series:")
print(non_com)

print("Smallest element in Series A:", a.min())
print("Largest element in Series A:",a.max())

print("Sum of Series B:", b.sum())
print("Average of Series A:",a.mean())
print("Median of Series B:", 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
```

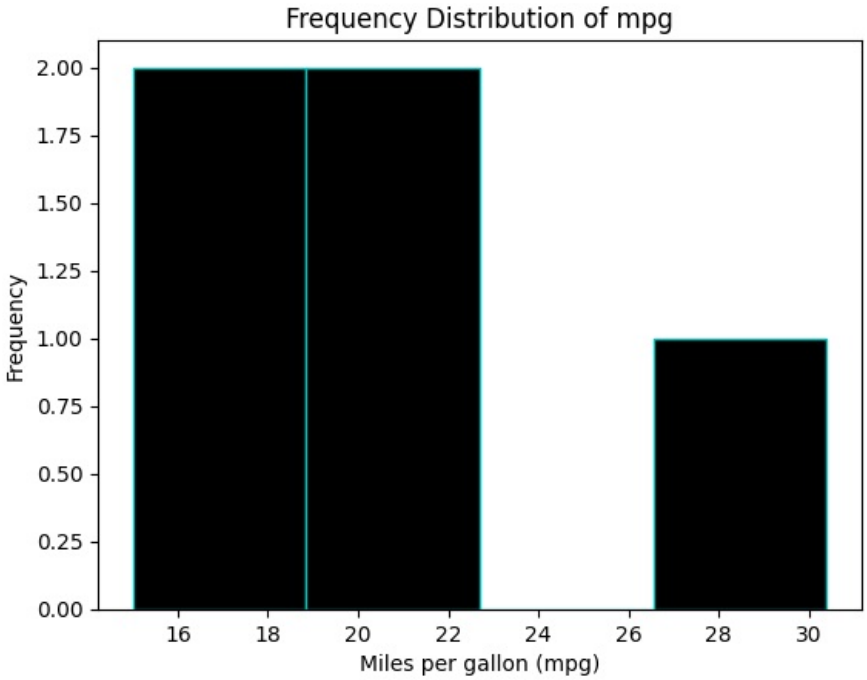
In [37]:
```
# program 21
import pandas as pd
da={
    "mpg":[18,15,18,16,17],"cylinders":[8,8,6,4,8],"displacement":[307,350,318,
 304,302],
    "horsepower":[130,165,150,150,140],"weigth":[3504,3693,3436,3433,3449],
    "acceleration":[12.0,11.5,11.0,12.0,10.5],"model year":[70,71,70,80,70],
    "origin":[1,1,1,1,1],
    "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("8 cylinders:",ei)
```

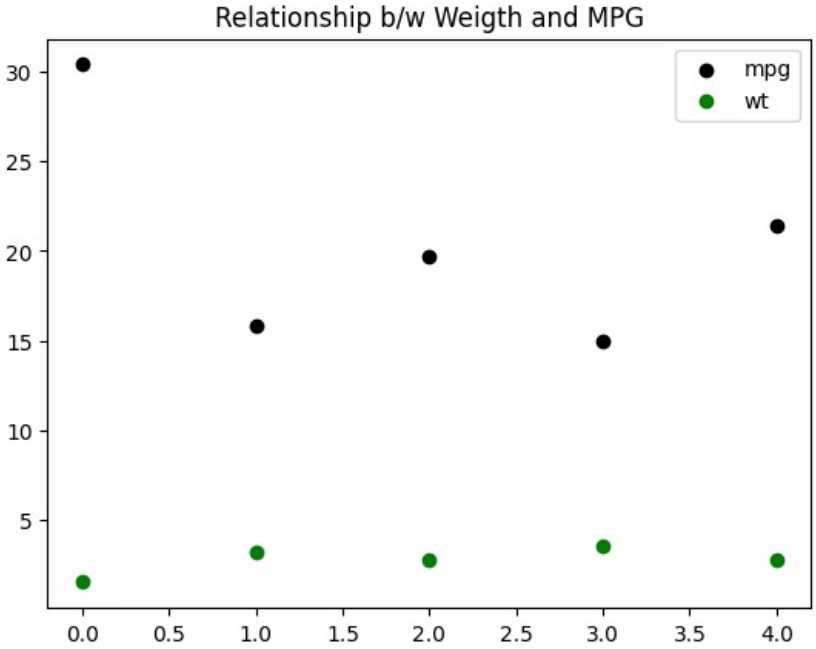
```
print("By year:",ye)
```

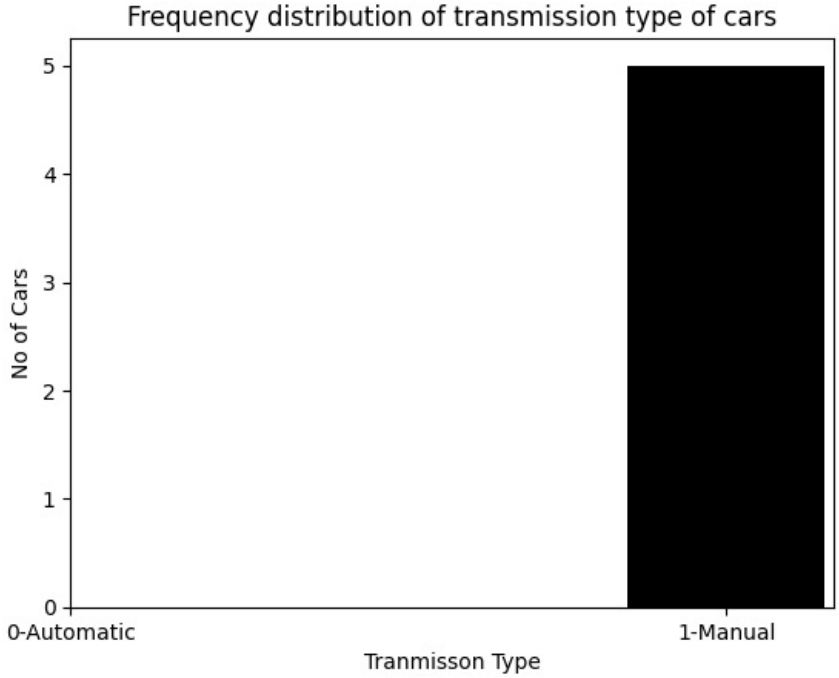
```
Satistical:
             mpg  cylinders  displacement  horsepower       weigth  \
count   5.00000   5.000000      5.000000    5.000000     5.000000
mean   16.80000   6.800000    316.200000  147.000000  3503.000000
std     1.30384   1.788854     19.879638   13.038405   110.006818
min    15.00000   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
```

In [70]:

```
# program 22
import pandas as p
import matplotlib.pyplot as m
#import seaborn as b
# data as 32 Elements
data=p.read_csv("emty.csv")

# HISTOGRAM
mpg=data['mpg']
m.hist(mpg,bins='auto',color='k',edgecolor='c')
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='k',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
b.boxplot(mpg,color='c')
m.xlabel("MPG");m.ylabel("Values")
m.title("BOX plot of MPG Vlues")
m.show()
```


Frequency Distribution of mpg
Frequency
Miles per gallon (mpg)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
16
18
20
22
24
26
28
30


Relationship b/w Weigth and MPG
mpg
wt
5
10
15
20
25
30
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Frequency distribution of transmission type of cars
No of Cars
Tranmisson Type
0
1
2
3
4
5
0-Automatic
1-Manual

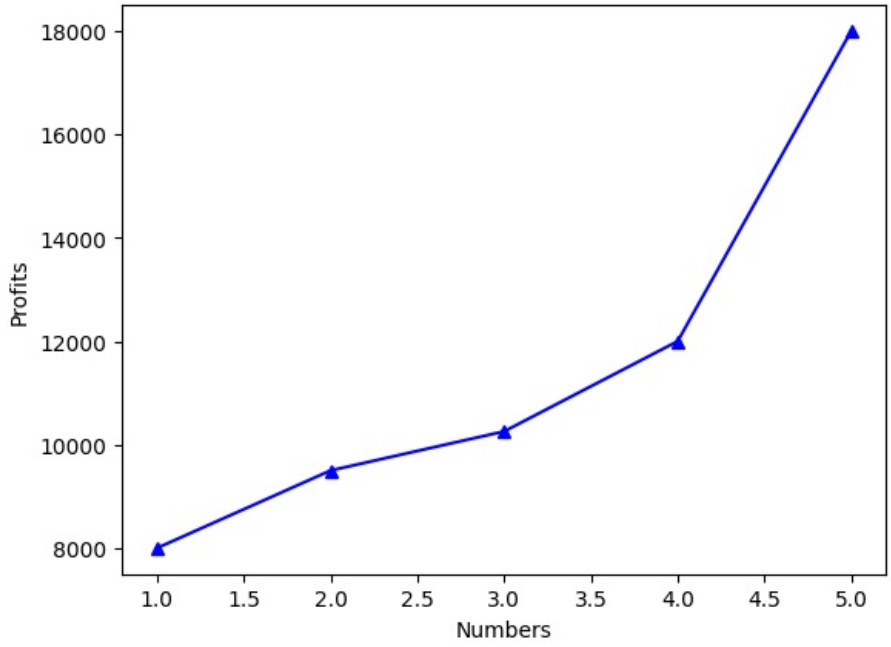
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12660\1487509867.py in ?()
     30 m.title("Frequency distribution of transmission type of cars")
     31 m.show()
     32
     33 # BOX PLOT
---> 34 b.boxplot(mpg,color='c')
     35 m.xlabel("MPG");m.ylabel("Values")
     36 m.title("BOX plot of MPG Vlues")
     37 m.show()

~\AppData\Roaming\Python\Python312\site-packages\pandas\core\generic.py in ?(self, name)
   6295             and name not in self._accessors
   6296             and self._info_axis._can_hold_identifiers_and_holds_name(name)
   6297         ):
   6298             return self[name]
-> 6299         return object.__getattribute__(self, name)

AttributeError: 'Series' object has no attribute 'boxplot'
```

In [74]:
```
# program 23
import pandas as p
import numpy as n

d={"Day":[1,2,3,4,5,6,7,8,9,10],
   "Steps":[4665,9552,7332,4904,5335,7852,8332,6504,8065,7689]}

dp=p.DataFrame(d)
dp["+1000 Steps"]=dp["Steps"]+1000

fi=dp[dp["+1000 Steps"]>7000]["Day"]

print("DataFrame:\n",dp)
print("Days on which Steps were >7000:",fi)
```

```
DataFrame:
    Day  Steps  +1000 Steps
0    1   4665         5665
1    2   9552        10552
2    3   7332         8332
3    4   4904         5904
4    5   5335         6335
5    6   7852         8852
6    7   8332         9332
7    8   6504         7504
8    9   8065         9065
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
```

In [84]:
```
# program 24
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("Sum of Profits:",su)

mi=df.isna()
print("Missing values:",mi)

print("Maximum Value:",df['Draw'].max())

m.plot(df['n'],df['Profits'],'^-',color='b')
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 [ ]: