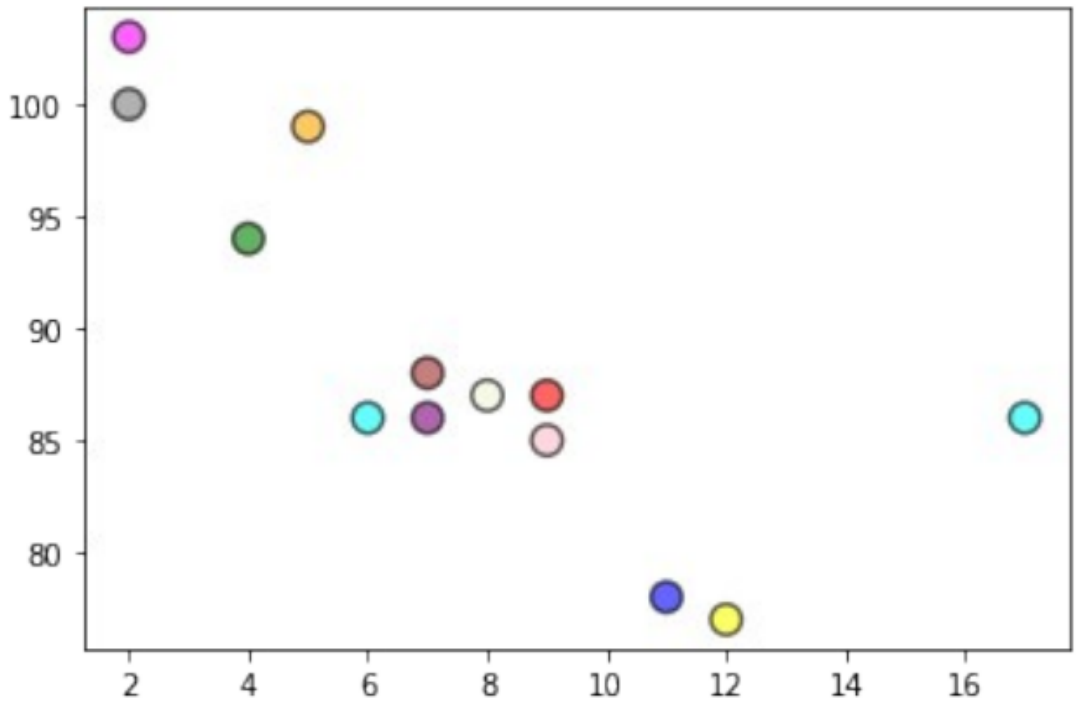
MINI PROJECT

NAME-Ayush Srivastav
Roll No-2193069
Class-CSE Core 2

Assignment No.1

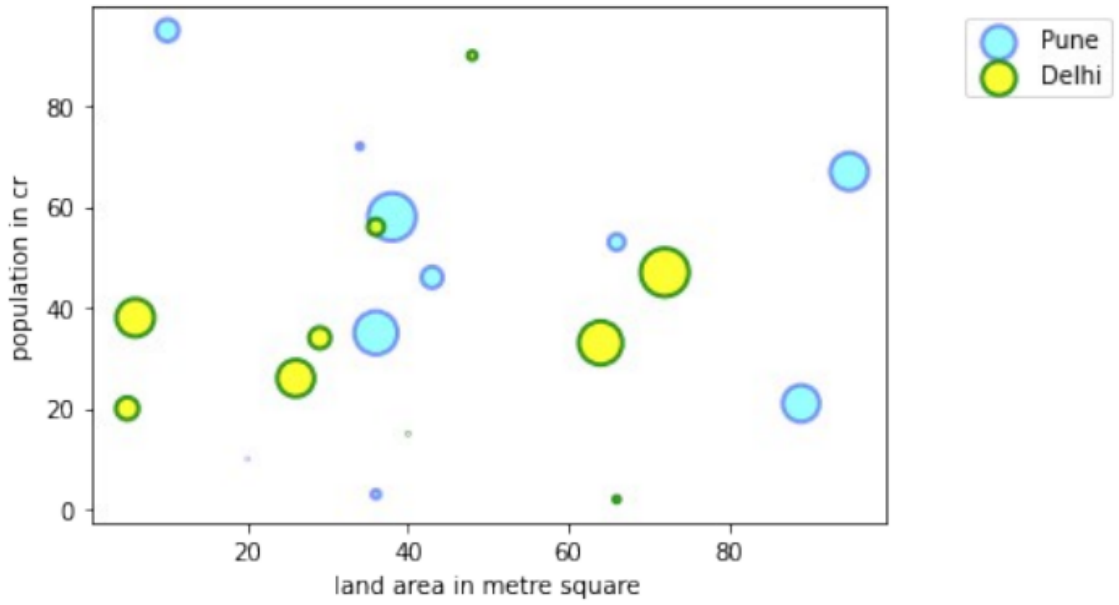
AIM: Scatter plot
Code with output:

```
import matplotlib.pyplot as plt x =[5, 7, 8, 7, 2, 17, 2, 9, 4, 11, 12, 9, 6] y =[99, 86, 87, 88, 100, 86, 103, 87, 94, 78, 77, 85, 86] color_set = ["orange","purple","beige","brown","gray","cyan","magenta","red","green","blue","yellow","pink","cyan"] plt.scatter(x , y , color = color_set , s = 110 , alpha = 0.6 , linewidth = 1.5 , edgecolor = "black") plt.show()
```



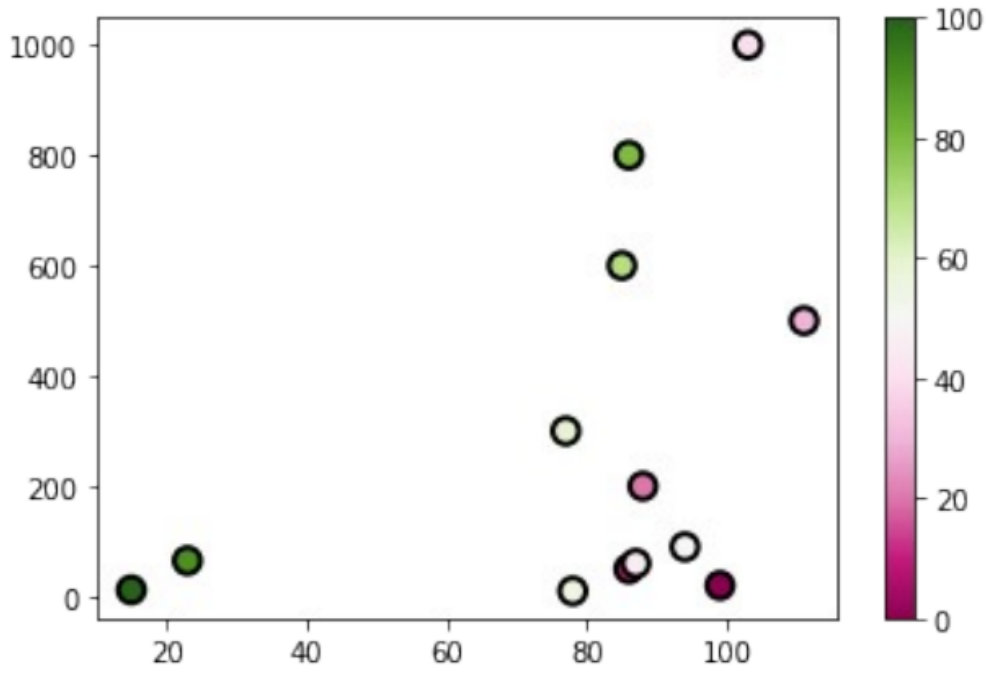
#AYUSH SRIVASTAV - 2193069

```
import matplotlib.pyplot as plt import numpy as np # dataset-1 x1 = [89, 43, 36, 36, 95, 10, 66, 34, 38, 20] y1 = [21, 46, 3, 35, 67, 95, 53, 72, 58, 10] # dataset2 x2 = [26, 29, 48, 64, 6, 5, 36, 66, 72, 40] y2 = [26, 34, 90, 33, 38, 20, 56, 2, 47, 15] sizes = (np.random.sample(size = 10) * 22) ** 2 plt.scatter(x1, y1, color = "cyan" , s=sizes , linewidth = 2 , marker = "o" , edgecolor = "blue" , alpha = 0.4) plt.scatter(x2, y2, color = "yellow" , s=sizes , linewidth = 2 , marker = "o" , edgecolor = "green" , alpha = 0.8)

plt.xlabel("land area in metre square") plt.ylabel("population in cr") plt.legend(["Pune" , "Delhi"] , bbox_to_anchor = (1.3, 1)) plt.show()
```



#AYUSH SRIVASTAV - 2193069

```
import matplotlib.pyplot as plt import numpy as np # Define Data x = np.array([99,86,88,111,103,87,94,78,77,85,86,23,15]) y = np.array([20,50,200,500,1000,60,90,10,300,600,800,65,12]) colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100]) plt.scatter(x, y, c = colors , cmap= 'PiYG' , s = 90 , linewidth = 2 , edgecolor = "black") plt.colorbar() plt.show()
```

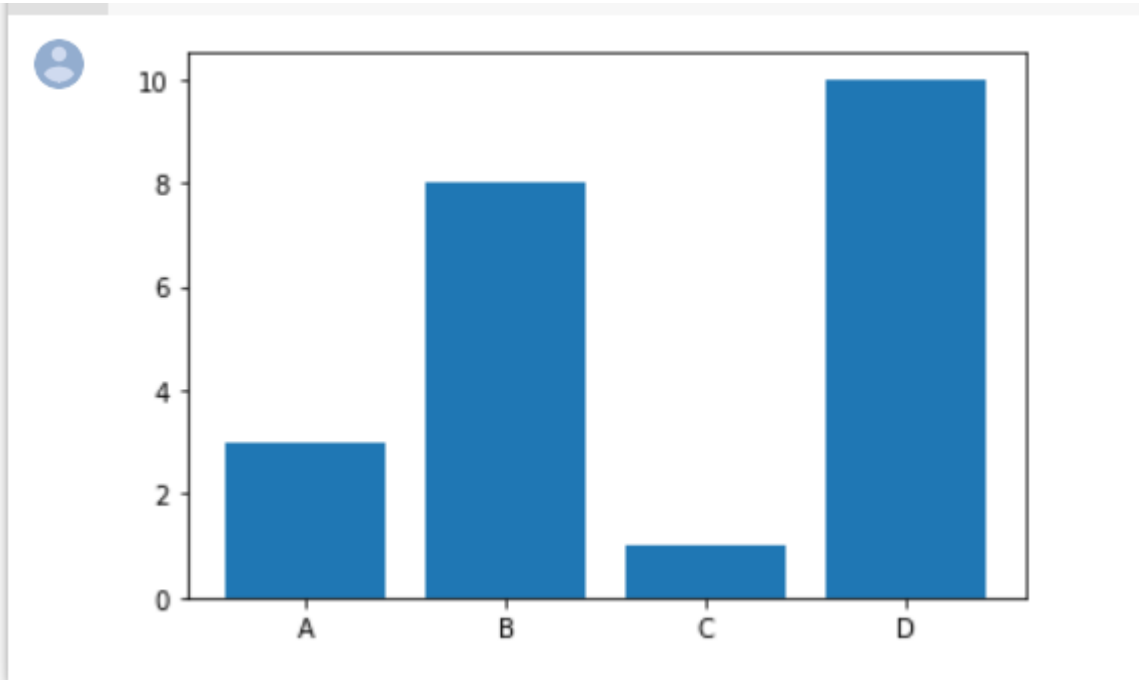


Assignment No.2

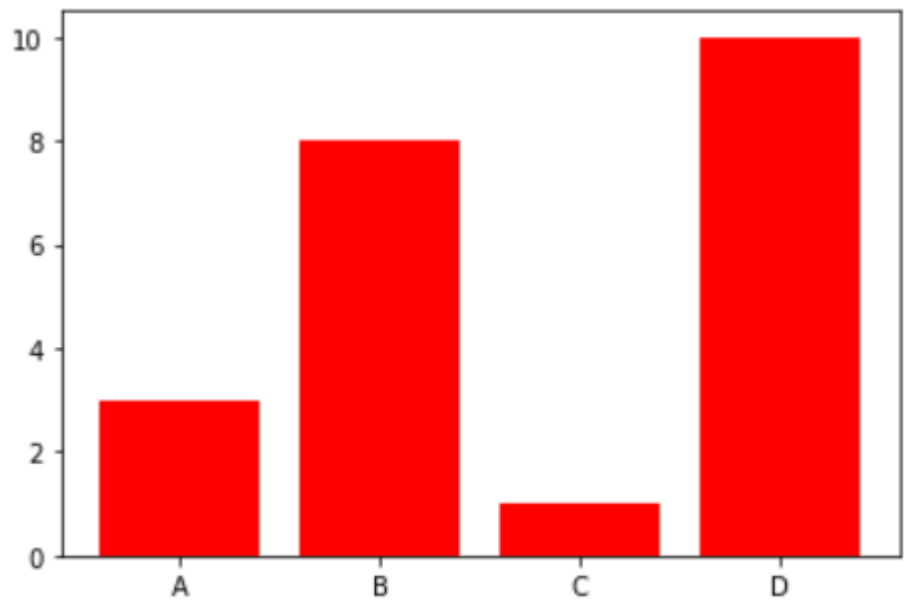
AIM: Different types of bar plot

Code with output

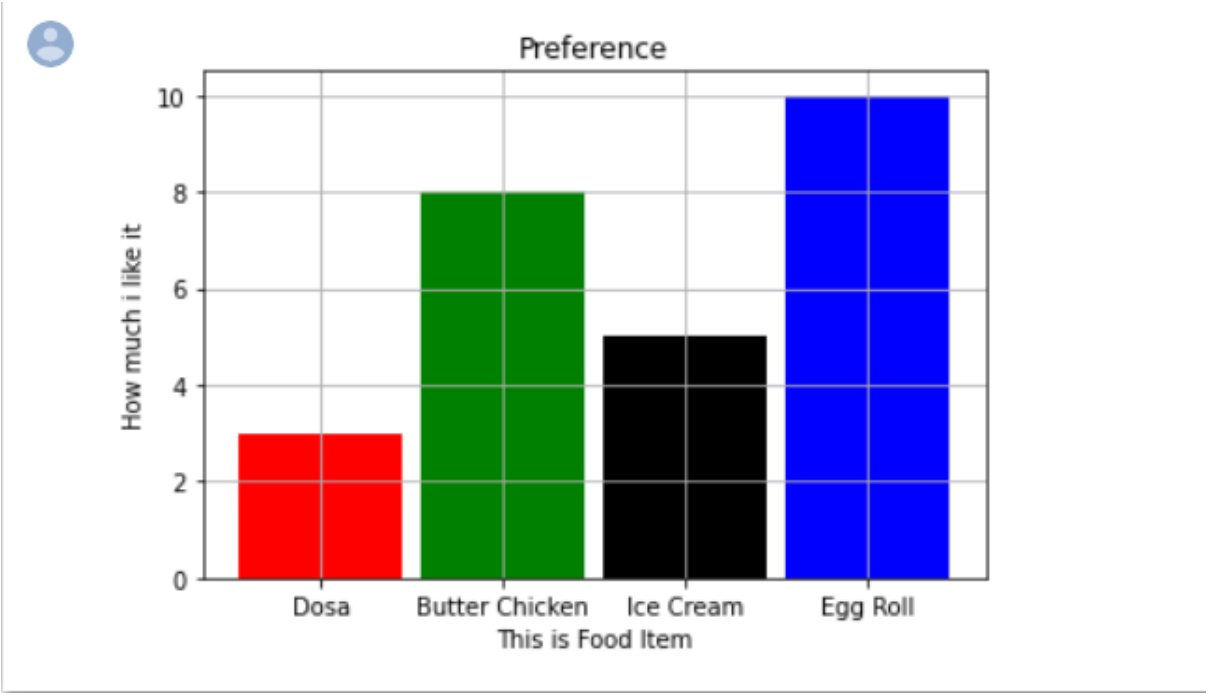
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x, y)
plt.show()
```




```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x, y,color="red")
plt.show()
```

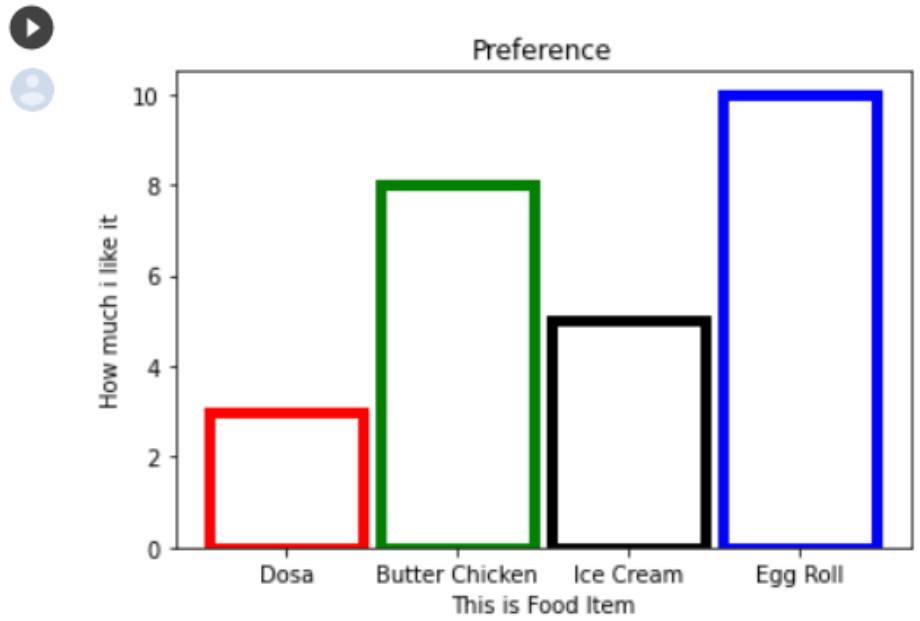


```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(["Dosa", "Butter Chicken", "Ice Cream", "Egg
Roll"])
y = np.array([3, 8, 5, 10])
z = np.array(["red","green","black","blue"])

plt.bar(x, y, width=0.9,color=z)
plt.xlabel("This is Food Item")
plt.ylabel("How much i like it")
plt.title("Preference")
plt.grid()
plt.show()
```




```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(["Dosa", "Butter Chicken", "Ice Cream", "Egg
Roll"])
y = np.array([3, 8, 5, 10])
z = ["red","green","black","blue"]
```

```
plt.bar(x, y,
width=0.9,fill=False,edgecolor=z,linewidth=5)
plt.xlabel("This is Food Item")
plt.ylabel("How much i like it")
plt.title("Preference")
plt.show()
```



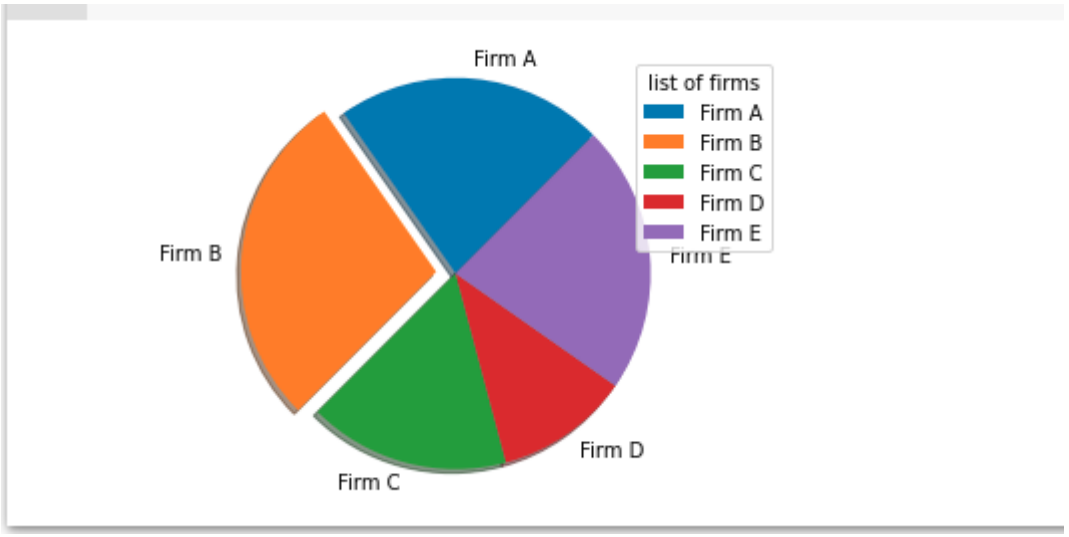
Assignment No.3

AIM: Pichart

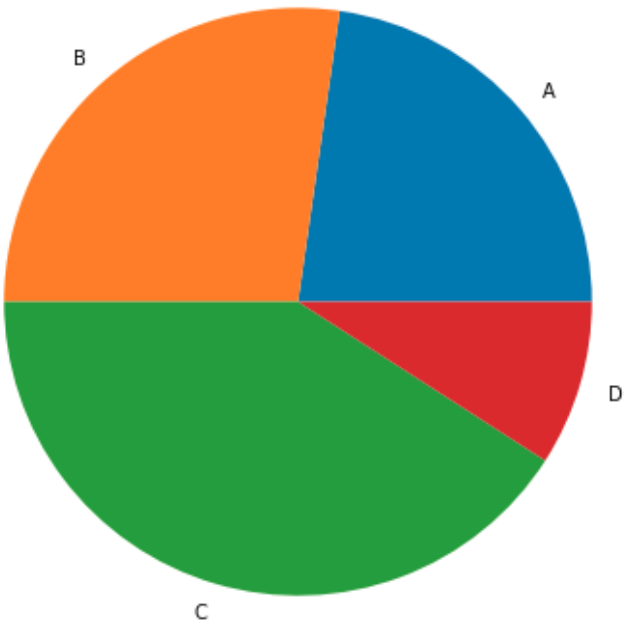
Code with output

```
from pandas import DataFrame
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

firms=["Firm A","Firm B","Firm C","Firm D","Firm E"]
market_share=[20,25,15,10,20]
explode=[0,0.1,0,0,0]
plt.pie(market_share,explode=explode,labels=firms,shadow=True,startangle=45)
```

```
plt.axis("equal")
plt.legend(title="list of firms")
plt.show()
```



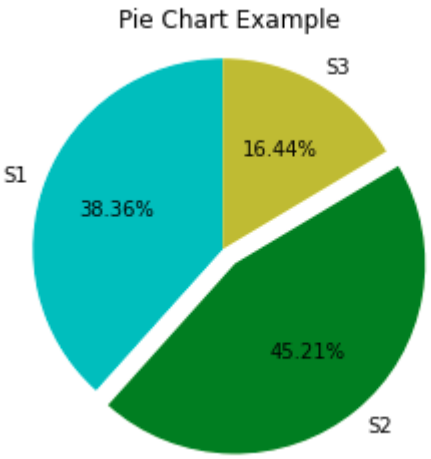

```
plt.figure(figsize=(7,7))
x = [25,30,45,10]
#labels of the pie chart
labels = ['A','B','C','D']
plt.pie(x, labels=labels)
plt.show()
```




```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
```

```
mycolors = ["black", "hotpink", "b", "#4CAF50"]

plt.pie(y, labels = mylabels, colors = mycolors)
plt.show()
```


```
labels = 'S1', 'S2', 'S3'
sections = [56, 66, 24]
colors = ['c', 'g', 'y']

plt.pie(sections, labels=labels, colors=colors,
        startangle=90,
        explode = (0, 0.1, 0),
        autopct = '%1.2f%%')

plt.axis('equal') # Try commenting this out.
plt.title('Pie Chart Example')
plt.show()
```

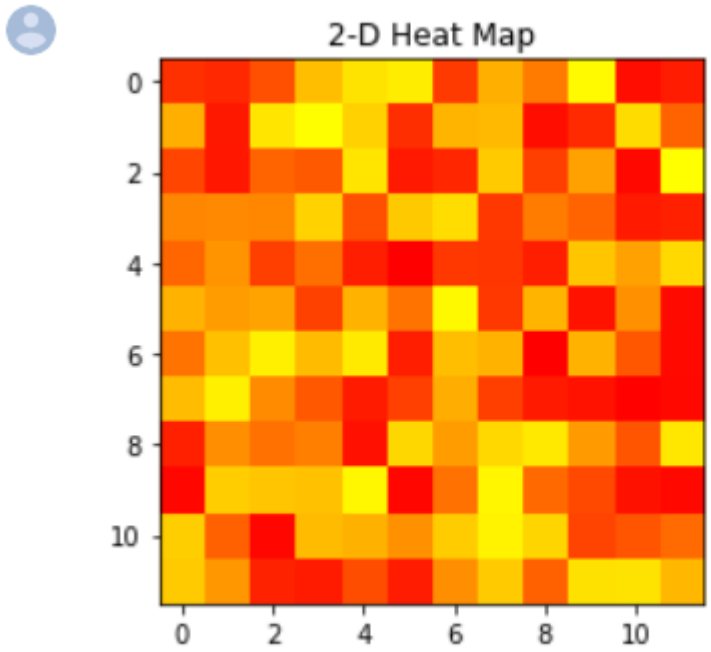


Assignment No.4

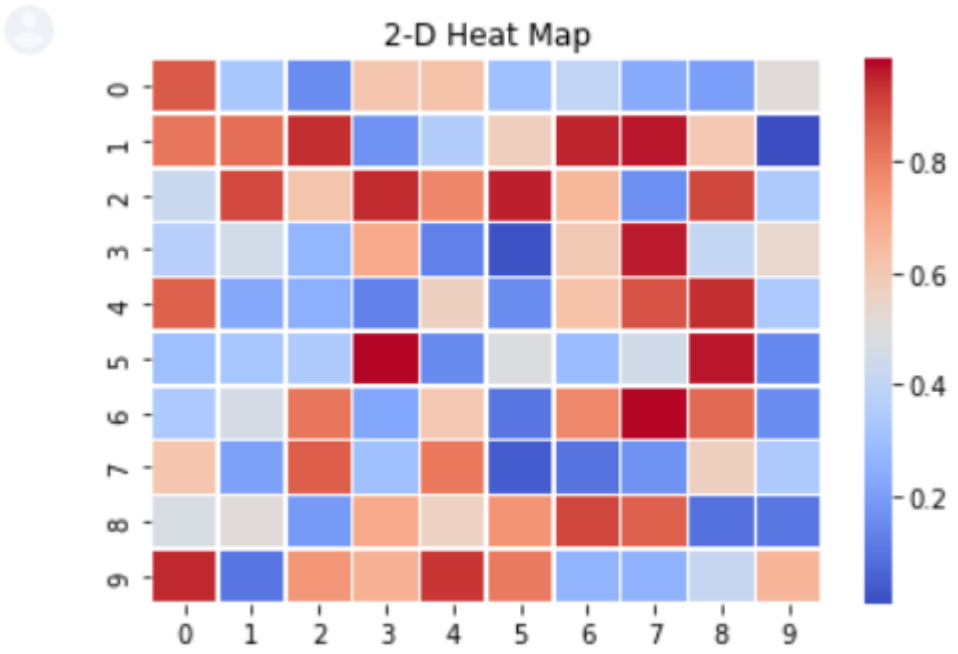
AIM: Heat Map

Code with output

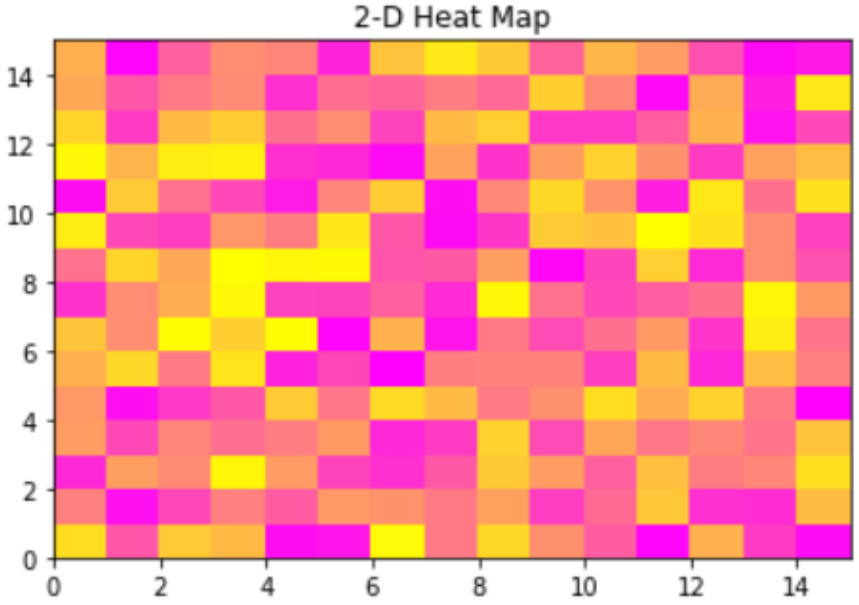
```
data = np.random.random(( 12 , 12 ))
plt.imshow( data , cmap = 'autumn' , interpolation = 'nearest' )

plt.title( "2-D Heat Map" )
plt.show()
```



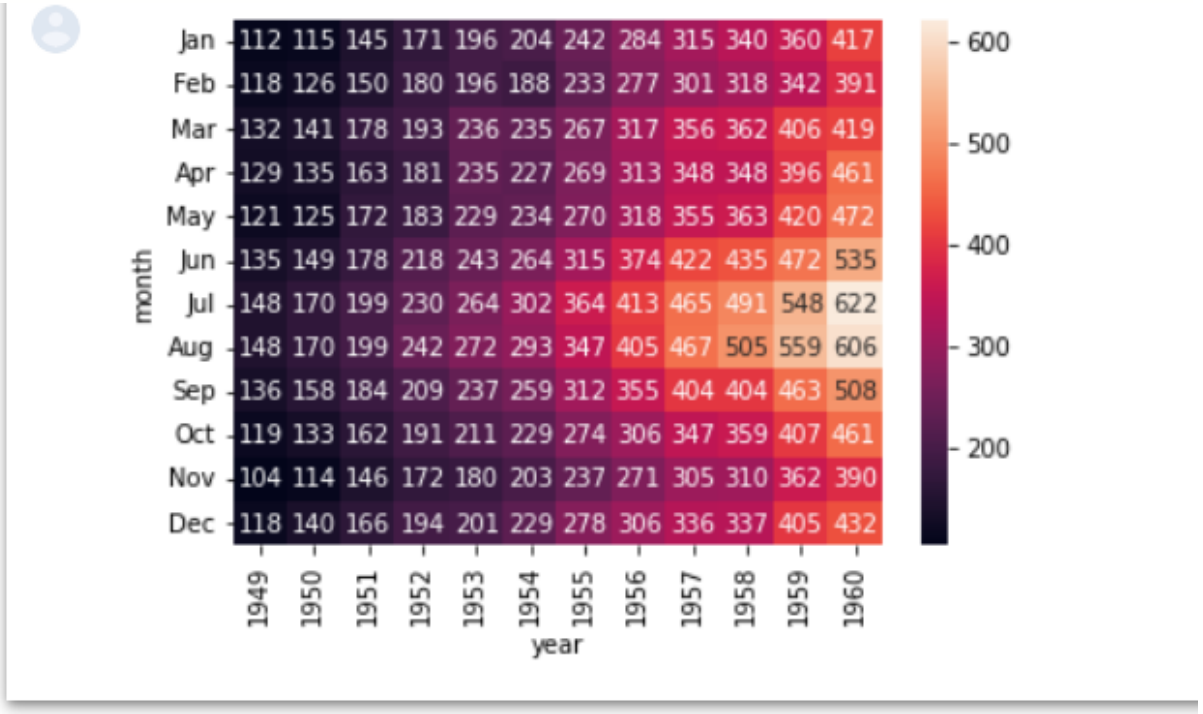

```
data_set = np.random.rand( 32, 5 )
ax = sns.heatmap( data_set , linewidth = 1 , cmap = 'coolwarm' )

plt.title( "2-D Heat Map" )
plt.show()
```



```
Z = np.random.rand( 15 , 15 )

plt.pcolormesh( Z , cmap = 'spring' )

plt.title( '2-D Heat Map' )
plt.show()
```




```
ax = sns.heatmap(flights, annot=True, fmt="d")
```

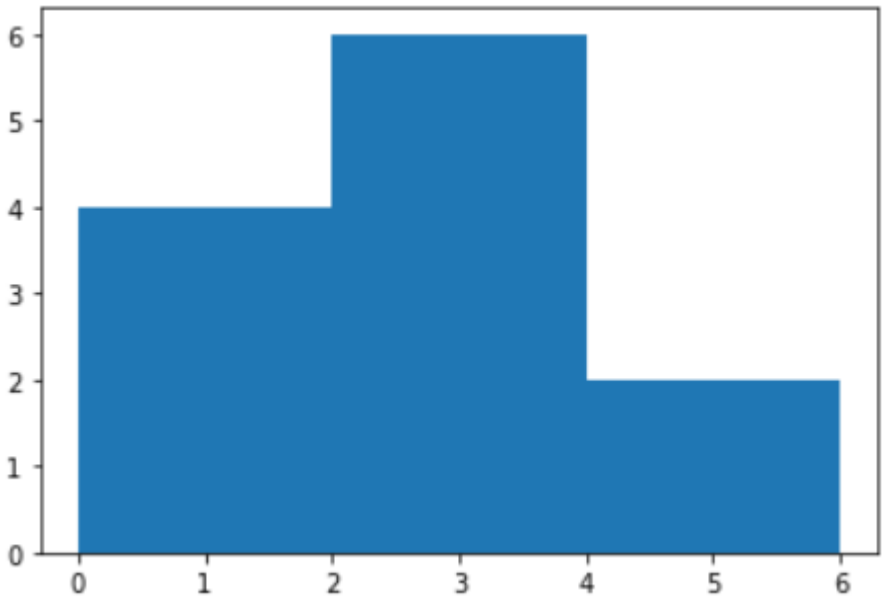


Assignment No.5

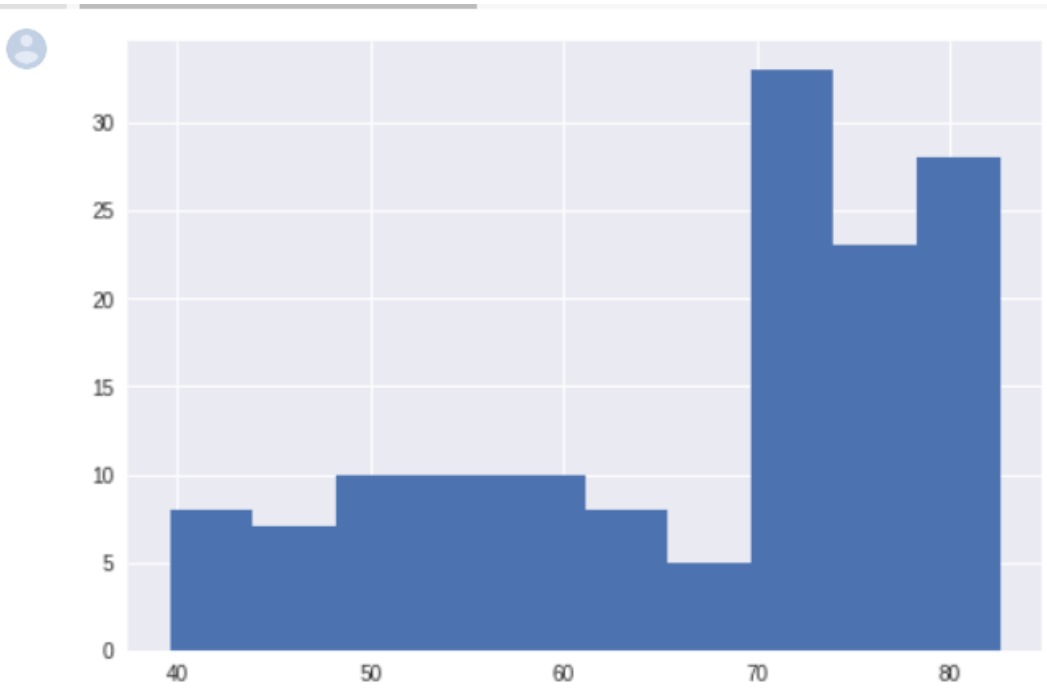
AIM: Histogram

Code with output

```
import matplotlib.pyplot as plt

values = [0, 0.6, 1.4, 1.6, 2.2, 2.5, 2.6, 3.2, 3.5, 3.9,
4.2, 6]

# default bins = 10
plt.hist(values, bins=3)

plt.show()
```




```
life_exp = [43.828, 76.423, 72.301, 42.731, 75.32, 81.235, 79.829,
75.635, 64.062, 79.441, 56.728, 65.554, 74.852, 50.728, 72.39, 73.005,
52.295, 49.58, 59.723, 50.43, 80.653, 44.74100000000001, 50.651,
78.553, 72.961, 72.889, 65.152, 46.462, 55.322, 78.782, 48.328, 75.748,
78.273, 76.486, 78.332, 54.791, 72.235, 74.994, 71.33800000000002,
71.878, 51.57899999999999, 58.04, 52.947, 79.313, 80.657, 56.735,
59.448, 79.406, 60.022, 79.483, 70.259, 56.007, 46.38800000000001,
60.916, 70.19800000000001, 82.208, 73.33800000000002, 81.757,
64.69800000000001, 70.65, 70.964, 59.545, 78.885, 80.745, 80.546,
72.567, 82.603, 72.535, 54.11, 67.297, 78.623, 77.58800000000002,
71.993, 42.592, 45.678, 73.952, 59.44300000000001, 48.303, 74.241,
54.467, 64.164, 72.801, 76.195, 66.803, 74.543, 71.164, 42.082, 62.069,
52.90600000000001, 63.785, 79.762, 80.204, 72.899, 56.867, 46.859,
80.196, 75.64, 65.483, 75.53699999999998, 71.752, 71.421, 71.688,
75.563, 78.098, 78.74600000000002, 76.442, 72.476, 46.242, 65.528,
72.777, 63.062, 74.002, 42.56800000000001, 79.972, 74.663, 77.926,
48.159, 49.339, 80.941, 72.396, 58.556, 39.613, 80.884,
81.70100000000002, 74.143, 78.4, 52.517, 70.616, 58.42, 69.819, 73.923,
71.777, 51.542, 79.425, 78.242, 76.384, 73.747, 74.249, 73.422, 62.698,
42.38399999999999, 43.487]

plt.hist(life_exp)
plt.show()
```



Assignment No.6

AIM: Introduction to Pandas

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.

- High performance merging and joining of data.
- Time Series functionality.

Pandas read csv

A simple way to store big data sets is to use CSV files (comma separated files).

CSV files contains plain text and is a well know format that can be read by everyone including Pandas.

Pandas - Analysing Dataframes

One of the most used method for getting a quick overview of the DataFrame, is the head() method.

The head() method returns the headers and a specified number of rows, starting from the top.

Pandas - Cleaning Data

Data cleaning means fixing bad data in your data set.

Bad data could be:

- Empty cells
- Data in wrong format
- Wrong data
- Duplicates

Pandas - Cleaning Empty Cells

Empty Cells

Empty cells can potentially give you a wrong result when you analyze data.

Remove Rows

One way to deal with empty cells is to remove rows that contain empty cells.

Pandas - Cleaning Data of Wrong Format

Data of Wrong Format

Cells with data of wrong format can make it difficult, or even impossible, to analyze data.

To fix it, you have two options: remove the rows, or convert all cells in the columns into the same format.

Assignment No.7

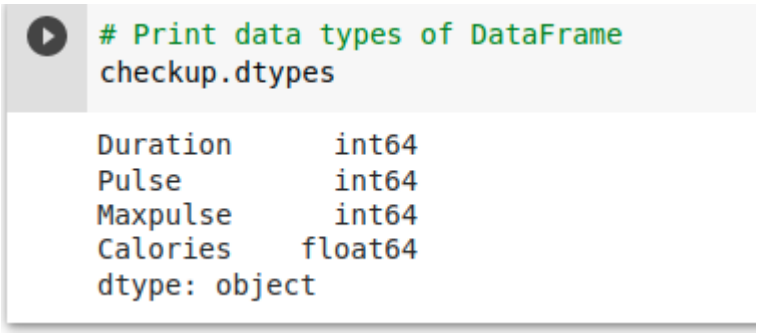
AIM: EDA(Exploratory Data Analysis)

Code with output

```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import missingno as msno
import datetime as dt
# Read in the dataset
checkup = pd.read_csv('/content/data.csv')
# Print the header of the DataFrame
checkup.head()
```

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0

```
# Print data types of DataFrame
checkup.dtypes
```




```
# Print info of DataFrame
checkup.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Duration  169 non-null    int64  
 1   Pulse     169 non-null    int64  
 2   Maxpulse  169 non-null    int64  
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
```

```
# Print number of missing values
checkup.isna().sum()
```

```
Duration    0
Pulse       0
Maxpulse    0
Calories    5
dtype: int64
```

```
# Print description of DataFrame
checkup.describe()
```

	Duration	Pulse	Maxpulse	Calories
count	169.000000	169.000000	169.000000	164.000000
mean	63.846154	107.461538	134.047337	375.790244
std	42.299949	14.510259	16.450434	266.379919
min	15.000000	80.000000	100.000000	50.300000
25%	45.000000	100.000000	124.000000	250.925000
50%	60.000000	105.000000	131.000000	318.600000
75%	60.000000	111.000000	141.000000	387.600000
max	300.000000	159.000000	184.000000	1860.400000

```
checkup.shape
```

```
(169, 4)
```

```
checkup.describe(include='all')
```

	Duration	Pulse	Maxpulse	Calories
count	169.000000	169.000000	169.000000	164.000000
mean	63.846154	107.461538	134.047337	375.790244
std	42.299949	14.510259	16.450434	266.379919
min	15.000000	80.000000	100.000000	50.300000
25%	45.000000	100.000000	124.000000	250.925000
50%	60.000000	105.000000	131.000000	318.600000
75%	60.000000	111.000000	141.000000	387.600000
max	300.000000	159.000000	184.000000	1860.400000

checkup.tail()

	Duration	Pulse	Maxpulse	Calories
164	60	105	140	290.8
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

checkup.Pulse.unique()

```
array([110, 117, 103, 109, 102, 104,  98, 100, 106,  90,  97, 108, 130,
       105,  92, 101,  93, 107, 114, 111,  99, 123, 118, 136, 121, 115,
       153, 159, 149, 151, 129,  83,  80, 150,  95, 152, 137, 124, 116,
       112, 119, 113, 141, 122,  85, 120, 125])
```

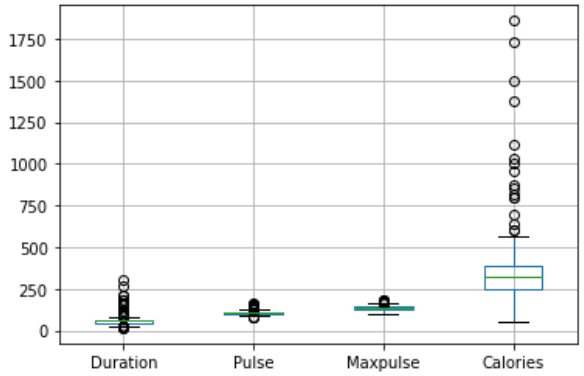
checkup.Pulse.value_counts()

```
100    19
90     12
103     9
109     9
107     8
108     7
97      7
110     7
106     6
111     6
98      6
105     6
102     6
104     4
114     4
95      3
115     3
117     3
118     3
136     3
93      3
92      3
99      2
151     2
112     2
123     2
80      2
```

```
80      2
150     2
101     2
149     1
116     1
120     1
85      1
122     1
141     1
113     1
119     1
124     1
159     1
137     1
152     1
130     1
121     1
153     1
83      1
129     1
125     1
Name: Pulse, dtype: int64
```

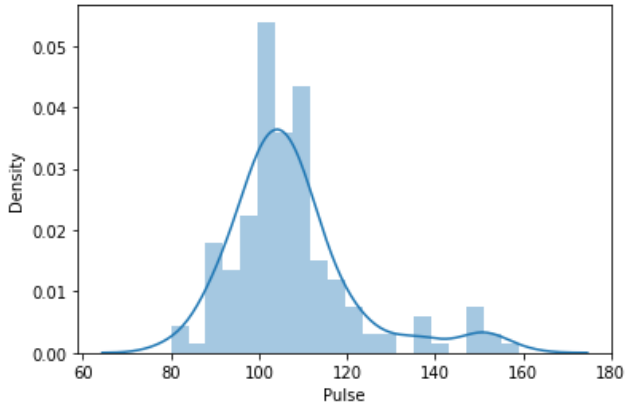
```
checkup.boxplot()
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:1376: VisibleDep
  X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else np.asarray(X))
<matplotlib.axes._subplots.AxesSubplot at 0x7fde205b7e10>
```

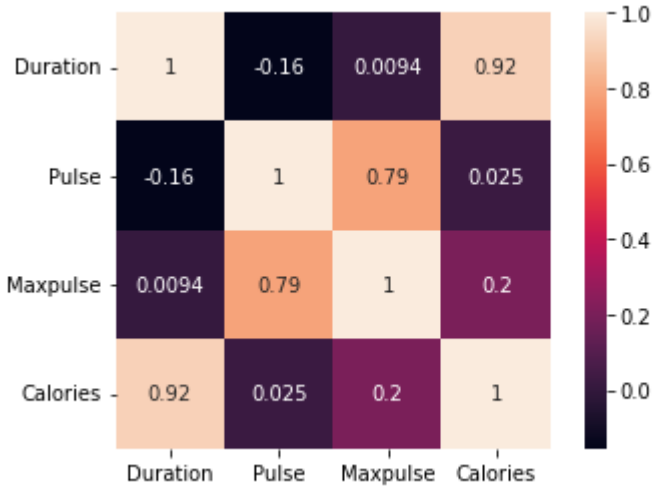



```
sns.distplot(checkup['Pulse'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fde1fa9fb10>
```

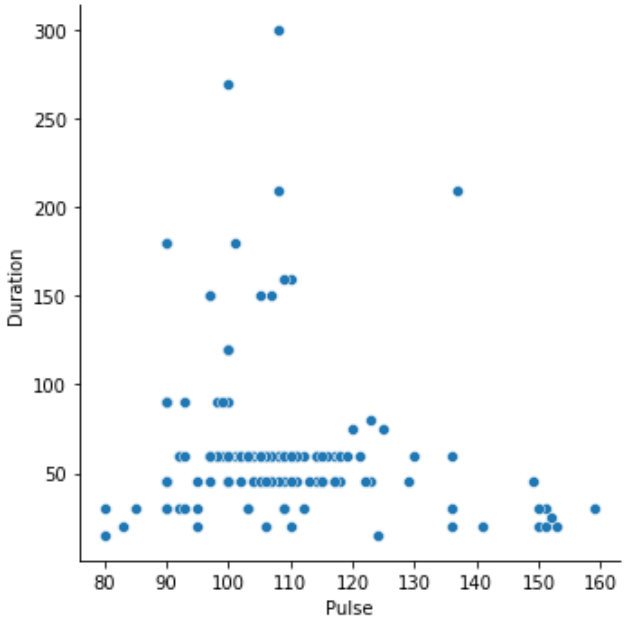



```
corr = checkup.corr()
sns.heatmap(corr, annot=True, square=True)
plt.yticks(rotation=0)
plt.show()
```




```
sns.relplot(x='Pulse', y='Duration', data=checkup)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fde1b0f6d50>
```




```
x = checkup["Calories"].mean()

checkup["Calories"].fillna(x, inplace = True)
# Print number of missing values
checkup.isna().sum()
```

```
Duration    0
Pulse       0
Maxpulse    0
Calories    0
dtype: int64
```

Assignment No.8

AIM: Scrapping

Code with output

```
import bs4
from bs4 import BeautifulSoup as bs
import requests
!pip install bs4
!pip install requests
```

```
Requirement already satisfied: bs4 in /usr/local/lib/python3.7/dist-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.7/dist-packages (from bs4) (4.6.3)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (2.23.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests) (2021.10.8)
```

```
link='https://www.flipkart.com/search?q=tv&as=on&as-show=on&otracker=AS_Query_TrendingAutoSuggest_8_0_na_na_na&otracker1=AS_Query_TrendingAutoSuggest_8_0_na_na_na&as-pos=8&as-type=TRENDING&suggestionId=tv&requestId=9c9fa553-b7e5-454b-a65b-bbb7a9c74a29'
page = requests.get(link)
page.content
soup = bs(page.content, 'html.parser')
#it gives us the visual representation of data
print(soup.prettify())
name=soup.find('div',class_="_4rR01T")
print(name)
```

```
<div class="_4rR01T">Nokia 109 cm (43 inch) Full HD LED Smart Android TV</div>
```

```
# to get just the name we will use the below code
name.text
```

```
'Nokia 109 cm (43 inch) Full HD LED Smart Android TV'
```

```
#get rating of a product
rating=soup.find('div',class_="_3LWZlK")
```

```
print(rating)
rating.text
#get other details and specifications of the product
specification=soup.find('div',class_="fMghEO")
print(specification)
Specification.text

Netflix|Prime Video|Disney+Hotstar|YoutubeOperating System: AndroidFull HD 1920 x 1080 Pixels24 W Speaker Output60 Hz Refresh Rate3 x HDMI | 2 x USBVA Type Panel1 Year Warranty on Product
for each in specification:
    spec=each.find_all('li',class_='rgWa7D')
    print(spec[0].text)
    print(spec[1].text)
    print(spec[2].text)
    print(spec[3].text)
    print(spec[4].text)
    print(spec[5].text)
```

```
Netflix|Prime Video|Disney+Hotstar|Youtube
Operating System: Android
Full HD 1920 x 1080 Pixels
24 W Speaker Output
60 Hz Refresh Rate
3 x HDMI | 2 x USB
```

```
#get price of the product
price=soup.find('div',class_='_30jeq3 _1_WHN1')
print(price)
price.text
```

```
<div class="_30jeq3 _1_WHN1">₹24,990</div>
'₹24,990'
```

```
products=[]                    #List to store the name of the product
prices=[]                      #List to store price of the product
```

```
ratings=[]                    #List to store rating of the
product
apps = []                     #List to store supported apps
os = []                       #List to store operating system
hd = []                       #List to store resolution
sound = []                    #List to store sound output
for data in soup.findAll('div',class_='_3pLy-c row'):
        names=data.find('div', attrs={'class':'_4rR01T'})
        price=data.find('div', attrs={'class':'_30jeq3
_1_WHN1'})
        rating=data.find('div', attrs={'class':'_3LWZlK'})
        specification = data.find('div',
attrs={'class':'fMghEO'})

        for each in specification:
            col=each.find_all('li',
attrs={'class':'rgWa7D'})
            app =col[0].text
            os_ = col[1].text
            hd_ = col[2].text
            sound_ = col[3].text

        products.append(names.text) # Add product name to
list
        prices.append(price.text) # Add price to list
        ratings.append(rating)    #Add rating specifications
to list
        apps.append(app)# Add supported apps specifications
to list
        os.append(os_) # Add operating system
specifications to list
        hd.append(hd_) # Add resolution specifications to
list
        sound.append(sound_) # Add sound specifications to
list
```

```
#printing the length of list
print(len(products))
print(len(ratings))
print(len(prices))
print(len(apps))
print(len(sound))
print(len(os))
```

```
24
24
24
24
24
24
```

```
import pandas as pd
df=pd.DataFrame({'Product Name':products,'Supported_apps':apps,'sound_system':sound,'OS':os,"Resolution":hd,'Price':prices})
df.head(10)
```

	Product Name	Supported_apps	sound_system	OS	Resolution	Price
0	Nokia 109 cm (43 inch) Full HD LED Smart Andro...	Netflix|Prime Video|Disney+Hotstar|Youtube	24 W Speaker Output	Operating System: Android	Full HD 1920 x 1080 Pixels	₹24,990
1	Inno-Q Pro 80 cm (32 inch) HD Ready LED Smart ...	Netflix|Prime Video|Disney+Hotstar|Youtube	20 W Speaker Output	Operating System: Android	HD Ready 1366 x 768 Pixels	₹8,999
2	Xiaomi 5A 80 cm (32 inch) HD Ready LED Smart A...	Netflix|Prime Video|Disney+Hotstar|Youtube	20 W Speaker Output	Operating System: Android	HD Ready 1366 x 768 Pixels	₹15,499
3	realme 80 cm (32 inch) HD Ready LED Smart Andr...	Netflix|Prime Video|Disney+Hotstar|Youtube	24 W Speaker Output	Operating System: Android	HD Ready 1366 x 768 Pixels	₹15,999
4	SAMSUNG Crystal 4K Pro 108 cm (43 inch) Ultra ...	Netflix|Disney+Hotstar|Youtube	20 W Speaker Output	Operating System: Tizen	Ultra HD (4K) 3840 x 2160 Pixels	₹35,990
5	Vu Premium TV 80 cm (32 inch) HD Ready LED Sma...	Netflix|Prime Video|Youtube	20 W Speaker Output	Operating System: Linux	HD Ready 1366 x 768 Pixels	₹12,499
6	OnePlus Y1 80 cm (32 inch) HD Ready LED Smart ...	Netflix|Prime Video|Disney+Hotstar|Youtube	20 W Speaker Output	Operating System: Android	HD Ready 1366 x 768 Pixels	₹15,999
7	Adsun 80 cm (32 inch) HD Ready LED Smart TV	Netflix|Disney+Hotstar|Youtube	20 W Speaker Output	Operating System: Android Based	HD Ready 1366 x 768 Pixels	₹8,750
8	LG 108 cm (43 inch) Ultra HD (4K) LED Smart TV	Netflix|Prime Video|Disney+Hotstar|Youtube	20 W Speaker Output	Operating System: WebOS	Ultra HD (4K) 3840 x 2160 Pixels	₹32,990
9	LG 80 cm (32 inch) HD Ready LED Smart TV	Netflix|Prime Video|Disney+Hotstar|Youtube	10 W Speaker Output	Operating System: WebOS	HD Ready 1366 x 768 Pixels	₹17,499

```
for i in range(1, 10):      #Number of pages plus one
        url =
"https://www.flipkart.com/search?q=tv&as=on&as-show=on&otracker=AS_Query_TrendingAutoSuggest_8_0_na_na_na&otracker1=
```

```
AS_Query_TrendingAutoSuggest_8_0_na_na_na&as-pos=8&as-type=TRENDING&suggestionId=tv&requestId=9c9fa553-b7e5-454b-a65b-bbb7a9c74a29".format(i)
        r = requests.get(url)
        soup = bs(r.content, "html5lib")    #Can use whichever parser you prefer
```