DELHI SKILL AND ENTREPRENEURSHIP UNIVERSITY

DSEU
Crafting Excellence

SESSION 2023-24

Data Science

(Data Science using Python)

Lab File

COURSE:- BCA

ROLL NO :- 41221139

SUBMITTED BY :-

Sachin Rajbhar

SUBMITTED TO:-

Ms. Bushra Jamal Ma'am

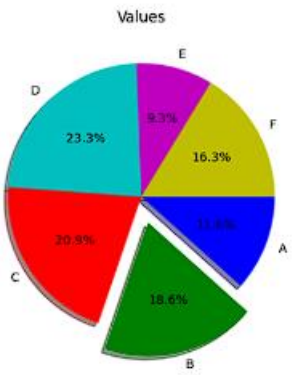
INDEX

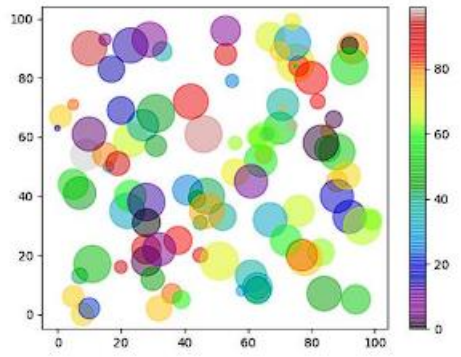
S.no	Name of practical	Date	Sign
1.	python code to print Fibonacci series		
2.	python code to perform arithmetic operations		
3.	python code to find area of triangle		
4.	python code to solve quadratic equation		
5.	python code for Swapping of 2 variables		
6.	python code to Check prime number		
7.	python code to perform following operations on strings: • Create string type variable • Slicing of strings • Concatenation of strings • Comparison of strings • Formatting of strings		
8.	Python code to illustrate use of • Arbitrary arguments • Keyword arguments • Arbitrary keyword arguments		
9.	Installation of Jupyter/ Google collab, and • Survey of Jupyter Dashboard • Adding new notebook		
10	Python codes to demonstrate usage of NumPy library to: • Install NumPy and show its version • Create array using NumPy • Use of arange() • zeros • ones • full • random 2D array • shaping and reshaping of an array • linspace • identity matrix • create and load a csv file using NumPy (Hint: loadtxt())		
11	Working with NumPy arrays • Create and find Max value in a 4 X 4 array • Find axis-wise max value from the above created array • Transpose of above created array matrix • Use of NumPy attributes and methods (e.g. : dtype(int 8, int 16, int 32, int 64), shape, size, itemsize, ndim, empty, empty_like, ravel, argmax(), argmin(), argsort(), sqrt(), where(), count_nonzero())		

12	Jupyter console - create a notebook and show use of various types of cells • Add code cell • Add mark down cells, and perform following: ➢ add levels of headers/headings ➢ make text bold and italics ➢ create ordered list ➢ create bullet list ➢ create hyperlinks ➢ create a table ➢ insert an image • Demonstrate usage of kernel menu in Jupyter. • Demonstrate usage of checkpoint in Jupyter.		
13	Working with Pandas Series and DataFrame • Install Pandas and show its version • Create Series in Pandas using list(and also demonstrate adding your own label rather than default) • Sort values of Series • Create a DataFrame using list • Create a DataFrame from given 2 Series: "calories": [420, 380, 390], "duration": [50, 40, 45] ➢ and add your own labels to it as - day1,day2, day3. ➢ Print only index and columns in this DataFrame • Convert a Pandas dictionary to a Pandas Series. • Create a DataFrame from a dictionary and print, & perform Introspection of data in above created DataFrame with: ➢ head(), tail(), renaming of columns ➢ use of shape, info(), index and describe() ➢ filter out only 1 column from a DataFrame ➢ slice out one element from the DataFrame ➢ slice out specific rows and columns from the DataFrame ➢ drop a column from a DataFrame • Create 2 DataFrames from the given dictionaries and then perform merging of DataFrames (left, right, inner, outer) d1 = {'Name': ['Pankaj', 'Meghna', 'Lisa'], 'Country': ['India', 'India', 'USA'], 'Role': ['CEO', 'CTO', 'CTO']} d2 = {'ID': [1, 2, 3], 'Name': ['Pankaj', 'Anupam', 'Amit']} • Create 2 DataFrames from the dictionaries given below and show their concatenation.		

	'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'], 'Age':[27, 24, 22, 32], 'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'], Qualification':['Msc', 'MA', 'MCA', 'Phd'] 'Name':['Abhi', 'Ayushi', 'Dhiraj', 'Hitesh'], 'Age':[17, 14, 12, 52], 'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'], 'Qualification':['Btech', 'B.A', 'Bcom', 'B.hons'] • Load a csv file in Jupyter/ Google Collab and display ➢ its values, & ➢ DataFrame • Create a DataFrame in Jupyter and convert it to .csv file.		
14	Use of help module: • ? • object ?(hint: df?) • object ?? • help() - help mode and interactive help		
15	Use of Magic functions in Jupyter or Ipython • %autocall • %cd • %automagic • %dhist • %env • %lsmagic • %matplotlib –list • %who • %pwd • %notebook • %run • %time (Hint: via comparison of NumPy array and a list) • %load • %quickref • %pdoc, etc.		
16	Working with Matplotlib • Install Matplotlib and show its version • Plot a single line using data as ([1,2,3], [2,5,7]) and also add title and labels to the plot. • Draw 2 lines showing comparison in a single plot, & also ➢ Change the default style ➢ Label the lines,& ➢ Change default width of lines • Demonstrate plotting a basic pie-chart		

17	Write a python code to display following pie-chart in clockwise direction with shadow.		

			
18	Read data.csv file from Kaggle and perform cleaning of bad data such as: • Empty cells ➢ Detect no of missing values in a column(using isna()) ➢ Remove rows containing empty cells(using dropna()) ➢ Replace empty values in a column using any value(using fillna()) ➢ Replace empty values in a column using mean, median or mode • Find the correlation between columns of the given dataset and visualize this dataset. • Show correlation between Duration and Calories columns using scatter plot. • Draw a histogram to show how many workouts lasted between 50 and 60 minutes?		
19	Write a python code to read any .csv file and increase the maximum number of rows from default to a large size so as to display the entire DataFrame.		
20	Write a python program to: • Read contents of a JSON file in Jupyter Notebook with or without Pandas. • Create JSON data in a dictionary and convert it to a pandas DataFrame. • Parse a JSON file(Hint: convert from JSON to Python)		
21	Create and parse a .xml file to a DataFrame in python.		
22	Write short python codes to demonstrate: • Uploading • Streaming • Sampling		
23	Write python codes to access data in structured flat-file form by: • parsing a .txt file • parsing a .csv file • reading different sheets from an excel file		
24	Write python codes to access data in unstructured flat-file form to: • Read an image from a public domain and render it on gray scale on screen. • Discover the image type and size • Resize the image by- ➢ Manipulating image array, or ➢ Use resize(), and then ➢ Convert the resized image to a dataset format for further analysis.		
25	Write python codes to demonstrate usage of following functions: • bar(), barh() while changing their default color, width, and height • hist() for normal data distribution for height of 250 people • pie() with labels, legends with header, shadow, change start angle and default colors, explode all wedges.		
26	Write a python program to draw a line with y points only(i.e. default x points) having following properties: • Dash-dot line sof Magenta color, size -10 and Hexagon marker(of size 10, edge color- green, face color - yellow) • Get the default axes and set ticks as [0,1,2,3,4,5,6,7,8,9,10] • Save the figure to your current working directory or anywhere else on your system • Add colored horizontal grid(y-axis), and also change default linewidth and linestyle of grid. • Annotate first and last points of line, and		

	• Add title, labels and legends to the graph(change the font, color and size of labels and title).		
27	Write python codes for following: • Plot only first and last markers of a line(i.e draw a line without line) • Display multiple plots in a single figure.		
28	Draw a scatter plot to show relationship between speed and age of 15 cars in movement with following properties: • set your own color for each dot in scatter plot • set the color of each dot for autumn color map, also show the autumn colorbar on side • set your own size for dots and also adjust the transparency of the dots • Comparison of speed and age of 15 cars for at least 3 days.		
29	Draw the following graph along with its colormap  		
30	Students can make assumptions for this question. • Create a DataFrame using a Dictionary/any csv file/ any json file etc. • Display all the column labels of your dataset • Sort the DataFrame based on a particular numerical column in descending order • Display the mean of any numerical column A in your DataFrame • Fill the missing values in column A with this mean value assuming no outliers are present in that column. • Remove the column having more than 4 null values.		
31	Assume that you have a DataFrame as DataFrame([["A", 1], ["B", 2], ["C", 3], ["D", 4]],columns = ["Col_A", "Col_B"]). • Insert a column at a specific location in a DataFrame. • Select columns based on the column's Data Type • Count the number of Non-NaN cells for each column • Split DataFrame into equal parts • Reverse DataFrame row-wise or column-wise		
32	Read company_sales_data.csv(657 B) on Kaggle and perform the following: • Read Total profit of all months and show it using a line plot • Get Total profit of all months and show line plot with the following Style properties ➢ Generated line plot must include following Style properties: - ➢ Line Style dotted and Line-color should be red ➢ Show legend at the lower right location. ➢ X label name = Month Number ➢ Y label name = Sold units number ➢ Add a circle marker. ➢ Line marker color as read ➢ Line width should be 3 • Read all product sales data and show it using a multiline plot. • Read toothpaste sales data of each month and show it using a scatter plot. • Read face cream and facewash product sales data and show it using the bar chart. Read sales data of bathing soap of all months and show it using a bar chart. Save this plot to your hard disk. • Read the total profit of each month and show it using the histogram to see most common profit ranges. • Calculate total sale data for last year for each product and show it using		

	a Pie chart. • Read Bathing soap facewash of all months and display it using the Subplot.		

PRACTICAL 1

Q: python code to print Fibonacci series

CODE:

```
def fib(n):
    a = 0
    b = 1
    print("Fibonacci Series:- ")
    print(a)
    print(b)
    for i in range (0,n-2):
        c=a+b
        print(c)
        a=b
        b=c

fib(6)
```

OUTPUT:

```
Fibonacci Series:-
0
1
1
2
3
5
```

PRACTICAL 2

Q: python code to perform arithmetic operations

CODE:

```
def arith(a,b):
    print("Arithmetic Operations:- a=",a,"and b=",b)
    print("a+b = ",a+b)
    print("a-b = ",a-b)
    print("a*b = ",a*b)
    print("a/b = ",a/b)
    print("a%b = ",a%b)

arith(3,6)
```

OUTPUT:

```
Arithmetic Operations:- a= 3 and b= 6
a+b =  9
a-b =  -3
a*b =  18
a/b =  0.5
a%b =  3
```

PRACTICAL 3

Q: python code to find area of triangle

CODE:

```
def triArea(b,h):
    return (1/2*b*h)

x = int(input("Enter base value:- "))
y = int(input("Enter height value:- "))
print("Area of given triangle =",triArea(x,y))
```

OUTPUT:

```
Enter base value:- 3
Enter height value:- 6
Area of given triangle = 9.0
```

PRACTICAL 4

Q: python code to solve quadratic equation

CODE:

```
import math
def quadEq(a,b,c):
    if a==0:
        print("Invalid Eqn")
    else:
        D = b*b-4*a*c
        if D>0:
            print("Two real solutions:-")
            one = ( -b + math.sqrt(D) )/2*a
            two = ( -b - math.sqrt(D) )/2*a
            print(one)
            print(two)
        elif D==0:
            print("Just one solution:-")
            one = (-b)/2*a
            print(one)

        else:
            absD = abs(D)
            print("Complex Solutions(No Real Solution):-")
            print((-b)/2*a,"+",math.sqrt(absD)/2*a,"i")
            print((-b)/2*a,"-",math.sqrt(absD)/2*a,"i")

quadEq(1,2,2)
```

OUTPUT:

```
Complex Solutions(No Real Solution)
-1.0 + 1.0 i
-1.0 - 1.0 i
```

PRACTICAL 5

Q: python code for Swapping of 2 variables

CODE:

```
def swap2(a,b):
        print("Before Swapping a=",a,"b=",b)
```

```
    temp=a
    a=b
    b=temp
    print("After Swapping a=",a,"b=",b)

swap2(1,-1)
```

OUTPUT:

```
Before Swapping a= 1 b= -1
After Swapping a= -1 b= 1
```

PRACTICAL 6

Q: python code to Check prime number

CODE:

```
def chkPrime(n):
   if(n<0):
      print("Not a Prime \nNegatives Cannot be prime")
   elif n==0:
      print("Not a Prime \nZero is divisible by all")
   else:
      # check factors
      isPrime = 1
      for i in range(2,int(n/2)):
         if n%i == 0:
            isPrime = 0
            break
      # use flag isPrime
      if isPrime:
         print(n,"is a prime number")
      else:
         print(n,"is a not a prime number")

chkPrime(15)
```

OUTPUT:

```
15 is a not a prime number
```

PRACTICAL 7

Q: python code to perform following operations on strings:

CODE:

```
# Create a string type variable
str_var = "Hello, world!"
```

```
# Slicing of strings
sliced_str = str_var[0:5]  # Slices the first 5 characters
print("Sliced string:", sliced_str)

# Concatenation of strings
str1 = "Hello"
str2 = "world"
concatenated_str = str1 + " " + str2 + "!"
print("Concatenated string:", concatenated_str)

# Comparison of strings
str3 = "hello"
if str1.lower() == str3.lower():
    print("The strings are equal (ignoring case)")
else:
    print("The strings are not equal (ignoring case)")

if str1 == str3:
    print("The strings are equal (case-sensitive)")
else:
    print("The strings are not equal (case-sensitive)")

# Formatting of strings
name = "John"
age = 30
formatted_str = "My name is {} and I am {} years old".format(name, age)
print("Formatted string:", formatted_str)
```

OUTPUT:

```
Sliced string: Hello
Concatenated string: Hello world!
The strings are equal (ignoring case)
The strings are not equal (case-sensitive)
Formatted string: My name is John and I am 30 years old
```

PRACTICAL 8

Q: Python code to illustrate use of

CODE:

```
# Example function that uses arbitrary arguments, keyword arguments, and arbitrary keyword
arguments
def my_function(arg1, arg2, *args, **kwargs):
    print("arg1:", arg1)
```

```
    print("arg2:", arg2)
    print("args:", args)
    print("kwargs:", kwargs)
    print()

# Call the function with various arguments
my_function(1, 2, 3, 4, 5, name="John", age=30)
my_function("Hello", "world", 1, 2, 3, key1="value1", key2="value2")
my_function(True, None)
```

OUTPUT:

```
arg1: 1
arg2: 2
args: (3, 4, 5)
kwargs: {'name': 'John', 'age': 30}

arg1: Hello
arg2: world
args: (1, 2, 3)
kwargs: {'key1': 'value1', 'key2': 'value2'}

arg1: True
arg2: None
args: ()
kwargs: {}
```

PRACTICAL 9

Q: Installation of Jupyter/ Google collab, and
- Survey of Jupyter Dashboard
- Adding new notebook

Install the classic Jupyter Notebook with:

```
pip install notebook
```

To run the notebook:

```
jupyter notebook
```

The main elements of a Jupyter Notebook include:

1. Notebook Dashboard: This is the landing page of Jupyter Notebook, where you can navigate through your directory and files, and create or open notebooks.

2. Menu Bar: The menu bar at the top of the notebook interface contains various menu options for opening, saving, and creating new notebooks, changing the kernel, and accessing other Jupyter Notebook features.
3. Toolbar: The toolbar contains icons for common notebook actions, such as saving the notebook, adding new cells, cutting/copying/pasting cells, and running cells.
4. Code Cells: Code cells are where you write and execute code in the notebook. Each code cell has an input area where you type code, and an output area where the results of running that code are displayed.
5. Markdown Cells: Markdown cells are used to write formatted text, such as headings, lists, tables, and links, in the notebook. You can also embed images and videos in Markdown cells.
6. Output Cells: Output cells are where the results of running code in a code cell are displayed. Depending on the code, the output could be text, images, tables, or charts.
7. Kernel: The kernel is the computational engine that executes the code in the notebook. You can change the kernel to use different programming languages or environments.
8. Cell Types: Cells can be classified as code cells or markdown cells, and each cell type has a different format for input and output.

Adding a new notebook in Jupyter Notebook is a straightforward process. Here are the steps:

1. Launch Jupyter Notebook by opening the Anaconda Navigator or by typing `jupyter notebook` in the command line.
2. Once Jupyter Notebook opens in your web browser, navigate to the directory where you want to create the new notebook.
3. Click on the "New" button in the top-right corner of the Jupyter Notebook interface.
4. From the dropdown menu, select "Python 3" (or another language of your choice) to create a new notebook with a Python kernel.
5. A new notebook will open in a new browser tab, and you can start typing code in the first cell.
6. To add additional cells, you can click the "+" button in the toolbar above the notebook or press the "B" key on your keyboard.
7. To save the notebook, click on the "Save and Checkpoint" button in the toolbar or go to "File" and select "Save and Checkpoint" from the dropdown menu.

PRACTICAL 10

Q: Python codes to demonstrate usage of NumPy library to:

- Create array using numpy

- Use of arange()
- zeros
- ones
- full
- random 2D array
- shaping and reshaping of an array
- linspace
- identity matrix

create and load a csv file

CODE:

```
import numpy as np
print(np.__version__)
arr = np.array([1, 2, 3, 4, 5])
print(arr)

print(type(arr))
print(arr.dtype)
np.array([[1,2],[3,4]])
```

OUTPUT:

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
int32

array([[1, 2],
       [3, 4]])
```

CODE:

```
np.arange(1,5)
x=np.arange(0,11,2)
print(x)
```

OUTPUT:

```
np.arange(1,5)
```

```
array([1, 2, 3, 4])
```

```
x=np.arange(0,11,2)
print(x)
```

```
[ 0  2  4  6  8 10]
```

CODE:

```
np.zeros((3,2))
```

OUTPUT:

```
array([[0.],
       [0.],
       [0.]])
```

CODE:

```
np.ones((3,5))
```

OUTPUT:

```
array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

CODE:

```
np.full((2,3),8)
```

OUTPUT:

```
array([[4, 4, 4],
       [4, 4, 4]])
```

CODE:

```
np.random.rand(2,3)
```

OUTPUT:

```
array([[0.97672616, 0.64471945, 0.38506868],
       [0.57354404, 0.14834056, 0.63658139]])
```

CODE:

```
arr = np.arange(1, 10)
print(arr, '\n')
arr = arr.reshape(3, 3)
print(arr, '\n')
print("minimum element is:", arr.min())
print("maximum element is:", arr.max())

# Reshape back to the original size
arr = arr.reshape(9)
print(arr)
```

OUTPUT:

```
[1 2 3 4 5 6 7 8 9]

[[1 2 3]
 [4 5 6]
 [7 8 9]]

minimum element is: 1
maximum element is: 9
[1 2 3 4 5 6 7 8 9]
```

CODE:

```
np.linspace(0,10,5)
```

OUTPUT:

```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

CODE:

```
np.eye(3)
```

OUTPUT:

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

CODE:

```
arr = np.loadtxt("C:\\Users\\athar123\\Desktop\\test_bca.csv", delimiter=",", dtype=str)
display(arr)
display(arr[2])
```

OUTPUT:

```
array([['a', 'b', 'c'],
       ['d', 'e', 'f'],
       ['g', 'h', 'i'],
       ['1', '2', '3']], dtype='<U1')

array(['g', 'h', 'i'], dtype='<U1')
```

PRACTICAL 11

Q: Working with NumPy arrays

- Create and find Max value in a 4 X 4 array
- Find axis-wise max value from the above created array
- Transpose of above created array matrix
- Use of NumPy attributes and methods (e.g. : dtype(int 8, int 16, int 32, int 64), shape, size, itemsize, ndim, empty, empty_like, ravel, argmax(), argmin(), argsort(), sqrt(), where(), count_nonzero())
- Convert NumPy array to list and check

CODE:

```
arraxis = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]])
print(np.max(arraxis))
```

OUTPUT:

```
16
```

CODE:

```
print("Sum of arr(axis = 0) : ", np.sum(arraxis, axis = 0)) #0 is column
print("Sum of arr(axis = 1) : ", np.sum(arraxis, axis = 1)) #1 is row
```

OUTPUT:

```
Sum of arr(axis = 0) :  [28 32 36 40]
Sum of arr(axis = 1) :  [10 26 42 58]
```

CODE:

```
print("\n",arraxis.T)
```

OUTPUT:

```
[[ 1  5  9 13]
 [ 2  6 10 14]
 [ 3  7 11 15]
 [ 4  8 12 16]]
```

CODE:

```
myarray = np.array([[3,6,777779,7]],np.int8)
print(myarray.dtype)
print(myarray.size)
print(myarray[0,3])
print(myarray[0,2]) # garbage value bcoz not in range of int8

la = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(la.shape)
print(la.dtype)
print(la.size)

ar = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(ar.ndim)
print(ar.size)
print(ar.itemsize)
print(ar.dtype)
print(ar.nbytes)

x = np.arange(99) # o to n-1 elements
print(x)
y = np.reshape(x,(3,33))
print("\n",y)
z = np.ravel(y)
print("\n",z)

emp = np.empty((4,6))
print(emp)
e = np.empty(2)
print("\n",e)

emp = np.empty_like(zer)
print(emp)

vikas=np.array([[1,2,3],[4,5,6],[7,8,9]])
argu=np.array([[6,3,1],[5,4,6],[7,9,8]])
print(vikas.argmax())
print(vikas.argmin())
print(vikas.argsort())
print("\n")
print(argu.argmax())
print(argu.argmin())
print(argu.argsort())

print(arr1)
print("\n",np.where(arr1>2)) #Dimaag kehraabh krr diya ishne (corresponding r & c)
print("\n",ar)
print(np.where(ar>3))
print("\n")
print(np.nonzero(arr1),np.count_nonzero(arr1),sep="\n")

print(arr1)
print("\n",np.where(arr1>2)) #Dimaag kehraabh krr diya ishne (corresponding r & c)
print("\n",ar)
print(np.where(ar>3))
print("\n")
print(np.nonzero(arr1),np.count_nonzero(arr1),sep="\n")
```

OUTPUT:

```
int8    int16       int32       int64
4       4           4           4
7       7           7           7
51      -8653       777779      777779
```

```
(3, 3)
int32
9
```

```
2
9
4
int32
36
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
 96 97 98]

 [[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 2
3
  24 25 26 27 28 29 30 31 32]
 [33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
  57 58 59 60 61 62 63 64 65]
 [66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89
  90 91 92 93 94 95 96 97 98]]

 [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
 96 97 98]
```

```
[[3.56043053e-307 1.60219306e-306 7.56571288e-307 1.89146896e-307
  1.37961302e-306 1.05699242e-307]
 [8.01097889e-307 1.78020169e-306 7.56601165e-307 1.02359984e-306
  1.33510679e-306 2.22522597e-306]
 [1.33511018e-306 6.23057689e-307 1.33511290e-306 1.78019082e-306
  8.45559303e-307 8.06613465e-308]
 [6.89810244e-307 1.22387550e-307 2.22522596e-306 8.34423917e-308
  9.79107193e-307 3.33509775e-317]]

 [ 1.32631977e-29 -2.54876250e-75]
```

```
[[[9.61061338e-312 9.61028475e-312 2.51438828e-312 0.00000000e+000]
  [1.69484279e-319 2.51438828e-312 0.00000000e+000 9.61063136e-312]
  [0.00000000e+000 0.00000000e+000 0.00000000e+000 9.61063369e-312]
  [0.00000000e+000 2.47032823e-323 6.95266271e-310 2.02369289e-320]
  [2.47032823e-323 6.95266272e-310 9.88131292e-324 0.00000000e+000]
  [0.00000000e+000 1.33563730e-318 0.00000000e+000             nan]]

 [[0.00000000e+000 3.56043053e-307 1.60219306e-306 7.56571288e-307]
  [1.89146896e-307 1.37961302e-306 1.05699242e-307 8.01097889e-307]
  [1.78020169e-306 7.56601165e-307 1.02359984e-306 1.33510679e-306]
  [2.22522597e-306 1.33511018e-306 6.23057689e-307 1.86921279e-306]
  [8.90098127e-307 1.78020848e-306 1.60219035e-306 1.42418172e-306]
  [2.04712906e-306 7.56589622e-307 1.11258277e-307 8.90111708e-307]]]
```

```
8               7
0               2
[[0 1 2]        [[2 1 0]
 [0 1 2]         [1 0 2]
 [0 1 2]]        [0 2 1]]
```

```
[[1 2 1]
 [4 0 6]
 [8 1 0]]

 (array([1, 1, 2], dtype=int64), array([0, 2, 0], dtype=int64))

 [[1 2 3]
 [4 5 6]
 [7 8 9]]
(array([1, 1, 1, 2, 2, 2], dtype=int64), array([0, 1, 2, 0, 1, 2], dtype=
int64))


(array([0, 0, 0, 1, 1, 2, 2], dtype=int64), array([0, 1, 2, 0, 2, 0, 1],
dtype=int64))
7
```

CODE:

```
import numpy as np
# create a NumPy array
x = np.array([1, 2, 3, 4, 5])
print(x)
# convert the array to a list
lst = x.tolist()
# print the list
print(lst)
```

OUTPUT:

```
[1 2 3 4 5]
[1, 2, 3, 4, 5]
```

PRACTICAL 12

Q: Jupyter console - create a notebook and show use of various types of cells
• Add code cell
• Add mark down cells, and perform following:
➢ add levels of headers/headings
➢ make text bold and italics
➢ create ordered list
➢ create bullet list
➢ create hyperlinks
➢ create a table
➢ insert an image
• Demonstrate usage of kernel menu in Jupyter.
• Demonstrate usage of checkpoint in Jupyter.

CODE:

```
for i in range(1, 6):
    print(i ** 2)
```

OUTPUT:

```
1
4
9
16
25
```

CODE:

```
# Heading 1
## Heading 2
### Heading 3

**This text is bold.** *This text is italic.*

- Item 1
- Item 2
- Item 3

1. First item
2. Second item
3. Third item

[Click here to go to Google](https://www.google.com)

| Name | Age | Gender |
|------|-----|--------|
| John | 30  | Male   |
| Jane | 25  | Female |

![Create](Save013.jpeg)
```

OUTPUT:

Heading 1

Heading 2

Heading 3

This text is bold. *This text is italic.*

- Item 1
- Item 2
- Item 3

1. First item
2. Second item
3. Third item

Click here to go to Google

Name	Age	Gender
John	30	Male
Jane	25	Female

Kernel menu

The kernel menu in Jupyter provides several options for managing and interacting with the Python kernel running in the background. You can access the kernel menu by clicking on "Kernel" in the top menu bar. Some common options include:

- Restart: Restarts the Python kernel
- Interrupt: Interrupts the execution of the current cell
- Change kernel: Allows you to switch to a different Python environment or kernel
- Shutdown: Shuts down the current kernel

Checkpoints

Checkpoints allow you to save a snapshot of your notebook's current state. To create a checkpoint, go to the "File" menu and click "Save and Checkpoint". This will create a new checkpoint that you can revert to later if needed. You can also access and manage checkpoints under the "File" menu.

PRACTICAL 13

Q: Working with Pandas Series and DataFrame

CODE:

```
!pip install pandas
import pandas as pd
print(pd.__version__)
```

OUTPUT:

```
1.4.4
```

CODE:

```
import pandas as pd
# Creating a Series using a list
fruits = pd.Series(['Apple', 'Banana', 'Cherry', 'Dates'])
# Printing the Series
print(fruits)
# Adding our own labels to the Series
fruits = pd.Series(['Apple', 'Banana', 'Cherry', 'Dates'], index=['a', 'b', 'c', 'd'])
print(fruits)
```

OUTPUT:

```
0     Apple
1    Banana
2    Cherry
3     Dates
dtype: object
a     Apple
b    Banana
c    Cherry
d     Dates
dtype: object
```

CODE:

```
import pandas as pd
# Creating a Series using a list
fruits = pd.Series(['Apple', 'Banana', 'Cherry', 'Dates'])
# Sorting the Series
sorted_fruits = fruits.sort_values()
# Printing the sorted Series
print(sorted_fruits)
```

OUTPUT:

```
0     Apple
1    Banana
2    Cherry
3     Dates
dtype: object
```

CODE:

```
import pandas as pd
# Creating a DataFrame using a list
data = [['Alice', 25], ['Bob', 30], ['Charlie', 35], ['David', 40]]
df = pd.DataFrame(data, columns=['Name', 'Age'])
# Printing the DataFrame
print(df)
```

OUTPUT:

```
      Name  Age
0    Alice   25
1      Bob   30
2  Charlie   35
3    David   40
```

CODE:

```
import pandas as pd
# Creating two Series
calories = pd.Series([420, 380, 390], index=['day1', 'day2', 'day3'])
duration = pd.Series([50, 40, 45], index=['day1', 'day2', 'day3'])
# Creating a DataFrame from the two Series
df = pd.DataFrame({'calories': calories, 'duration': duration})
# Adding our own labels to the DataFrame
df.index = ['day1', 'day2', 'day3']
# Printing only index and columns in this DataFrame
print(df.index)
print(df.columns)
```

OUTPUT:

```
Index(['day1', 'day2', 'day3'], dtype='object')
Index(['calories', 'duration'], dtype='object')
```

CODE:

```
import pandas as pd
# Creating a dictionary
data = {'apple': 2, 'banana': 3, 'cherry': 4}
# Converting the dictionary to a Series
fruits = pd.Series(data)
# Printing the Series
print(fruits)
```

OUTPUT:

```
apple     2
banana    3
cherry    4
dtype: int64
```

CODE:

```
import pandas as pd
# create a dictionary with some data
data = {'Name': ['John', 'Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 20, 35],
        'Gender': ['M', 'F', 'M', 'M'],
```

```
    'City': ['New York', 'Paris', 'London', 'San Francisco']}
# create a DataFrame from the dictionary
df = pd.DataFrame(data)
# print the first 5 rows of the DataFrame
print(df.head())
# print the last 5 rows of the DataFrame
print(df.tail())
# rename the 'City' column to 'Location'
df.rename(columns={'City': 'Location'}, inplace=True)
# print the shape of the DataFrame
print(df.shape)
# print information about the DataFrame
print(df.info())
# print the index of the DataFrame
print(df.index)
# print the description of the DataFrame
print(df.describe())
# filter out only the 'Age' column
age_col = df['Age']
print(age_col)
# slice out one element from the DataFrame
elem = df.loc[1, 'Name']
print(elem)
# slice out specific rows and columns from the DataFrame
slice_df = df.loc[[0, 2], ['Name', 'Location']]
print(slice_df)
# drop the 'Gender' column from the DataFrame
df.drop(columns=['Gender'], inplace=True)
print(df)
```

OUTPUT:

```
      Name  Age Gender           City
0     John   25      M       New York
1    Alice   30      F          Paris
2      Bob   20      M         London
3  Charlie   35      M  San Francisco
      Name  Age Gender           City
0     John   25      M       New York
1    Alice   30      F          Paris
2      Bob   20      M         London
3  Charlie   35      M  San Francisco
(4, 4)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      4 non-null      object
 1   Age       4 non-null      int64
 2   Gender    4 non-null      object
 3   Location  4 non-null      object
dtypes: int64(1), object(3)
memory usage: 256.0+ bytes
None
RangeIndex(start=0, stop=4, step=1)
             Age
count   4.000000
mean   27.500000
std     6.454972
min    20.000000
25%    23.750000
50%    27.500000
75%    31.250000
max    35.000000
0    25
1    30
2    20
3    35
Name: Age, dtype: int64
Alice
```

```
   Name  Location
0  John  New York
2   Bob    London
      Name  Age       Location
0     John   25       New York
1    Alice   30          Paris
2      Bob   20         London
3  Charlie   35  San Francisco
```

CODE:

```
import pandas as pd
d1 = {'Name': ['Pankaj', 'Meghna', 'Lisa'], 'Country': ['India', 'India', 'USA'], 'Role': ['CEO', 'CTO', 'CTO']}
d2 = {'ID': [1, 2, 3], 'Name': ['Pankaj', 'Anupam', 'Amit']}
# create dataframes from dictionaries
df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)
# merge dataframes
# left merge
df_left = pd.merge(df1, df2, on='Name', how='left')
print("Left merge:")
print(df_left)
# right merge
df_right = pd.merge(df1, df2, on='Name', how='right')
print("\nRight merge:")
print(df_right)
# inner merge
df_inner = pd.merge(df1, df2, on='Name', how='inner')
print("\nInner merge:")
print(df_inner)
# outer merge
df_outer = pd.merge(df1, df2, on='Name', how='outer')
print("\nOuter merge:")
print(df_outer)
```

OUTPUT:

```
Left merge:
     Name Country Role   ID
0  Pankaj   India  CEO  1.0
1  Meghna   India  CTO  NaN
2    Lisa     USA  CTO  NaN

Right merge:
     Name Country Role  ID
0  Pankaj   India  CEO   1
1  Anupam     NaN  NaN   2
2    Amit     NaN  NaN   3

Inner merge:
     Name Country Role  ID
0  Pankaj   India  CEO   1

Outer merge:
     Name Country Role   ID
0  Pankaj   India  CEO  1.0
1  Meghna   India  CTO  NaN
2    Lisa     USA  CTO  NaN
3  Anupam     NaN  NaN  2.0
4    Amit     NaN  NaN  3.0
```

CODE:

```
import pandas as pd
d1 = {'Name': ['Jai', 'Princi', 'Gaurav', 'Anuj'], 'Age': [27, 24, 22, 32], 'Address': ['Nagpur', 'Kanpur', 'Allahabad',
'Kannuaj'], 'Qualification': ['Msc', 'MA', 'MCA', 'Phd']}
d2 = {'Name': ['Abhi', 'Ayushi', 'Dhiraj', 'Hitesh'], 'Age': [17, 14, 12, 52], 'Address': ['Nagpur', 'Kanpur',
'Allahabad', 'Kannuaj'], 'Qualification': ['Btech', 'B.A', 'Bcom', 'B.hons']}
df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)
df_concatenated = pd.concat([df1, df2])
print(df_concatenated)
```

OUTPUT:

```
      Name  Age    Address Qualification
0      Jai   27     Nagpur           Msc
1   Princi   24     Kanpur            MA
2   Gaurav   22  Allahabad           MCA
3     Anuj   32    Kannuaj           Phd
0     Abhi   17     Nagpur         Btech
1   Ayushi   14     Kanpur           B.A
2   Dhiraj   12  Allahabad          Bcom
3   Hitesh   52    Kannuaj        B.hons
```

CODE:

```
import pandas as pd
# Load the csv file
df = pd.read_csv('df_to_file.csv')
```

```
# Display the values
print(df.values)
# Display the DataFrame
print(df)
```

OUTPUT:

```
[['John' 25 'New York']
 ['Emily' 27 'Paris']
 ['Michael' 30 'London']
 ['Jessica' 22 'Los Angeles']]
      Name  Age         City
0     John   25     New York
1    Emily   27        Paris
2  Michael   30       London
3  Jessica   22  Los Angeles
```

CODE:

```
import pandas as pd
# create a sample DataFrame
data = {
   'Name': ['John', 'Emily', 'Michael', 'Jessica'],
   'Age': [25, 27, 30, 22],
   'City': ['New York', 'Paris', 'London', 'Los Angeles']
}
df = pd.DataFrame(data)
# print the DataFrame
print(df)
# save the DataFrame to a CSV file
df.to_csv('df_to_file.csv', index=False)
```

OUTPUT:

```
      Name  Age         City
0     John   25     New York
1    Emily   27        Paris
2  Michael   30       London
3  Jessica   22  Los Angeles
```

PRACTICAL 14

Q: Use of help module:

- ?
- object ?(hint: df?)
- object ??

• help() - help mode and interactive help

The `help` module in Python provides a way to access documentation and help for objects and modules within Python. There are several ways to use the `help` module, including the following:

- `?`: The `?` character can be used to get help for any object or module. For example, typing `list?` in a Jupyter Notebook code cell will display the help information for the `list` type. Similarly, `numpy.arange?` will display the help information for the `numpy.arange()` function.
- `object?`: Adding a `?` after an object name, such as a variable or function, will display its docstring in a Jupyter Notebook. For example, typing `df?` will display the docstring for the `df` variable, assuming it has one.
- `object??`: Adding a `??` after an object name will display the source code for the object, if available. For example, typing `numpy.arange??` in a Jupyter Notebook code cell will display the source code for the `numpy.arange()` function.
- `help()`: The `help()` function can be used to enter the interactive help mode. In this mode, you can type the name of an object or module to display its help information. You can exit the interactive help mode by typing `quit`.

CODE:

```
# enter help mode
help()

# display help information for the list type
help(list)

# exit help mode
quit()
```

OUTPUT:

```
Welcome to Python 3.9's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.9/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help>
```

PRACTICAL 15

Q: Use of Magic functions in Jupyter or Ipython

```
# With autocall set to 1, parentheses are optional for function calls
%autocall 1
```

```
# This function call doesn't require parentheses
print "Hello world!"

# Change to a different directory
%cd /path/to/directory

# With automagic on, you don't need to prefix magics with %
%automagic on
# This cell will execute as if you typed %pwd
Pwd

# Display the history of commands entered in the current session
%dhist

# Set an environment variable
%env MYVAR=hello
# Access the environment variable
print(os.environ['MYVAR'])

# List all available magics
%lsmagic

# List all available backends for matplotlib
%matplotlib -list

# Define some variables
x = 5
y = 'hello'
z = [1, 2, 3]
# List all defined variables
%who

# Display the current working directory
%pwd

# Convert the current notebook to a standalone executable script
%notebook myscript.py

# Run a Python script in the current namespace
%run myscript.py

import numpy as np
# Create a NumPy array and a list
arr = np.arange(1000000)
lst = list(range(1000000))
# Time the execution of summing the array and the list
%time arr_sum = np.sum(arr)
%time lst_sum = sum(lst)

# Load code from a file into a code cell
%load myscript.py

# Display a quick reference guide for Jupyter Notebook
%quickref

# Display the docstring for a function
%pdoc numpy.arange
```

PRACTICAL 16

Q: Working with Matplotlib

CODE:

```
!pip install matplotlib
import matplotlib
print(matplotlib.__version__)
```

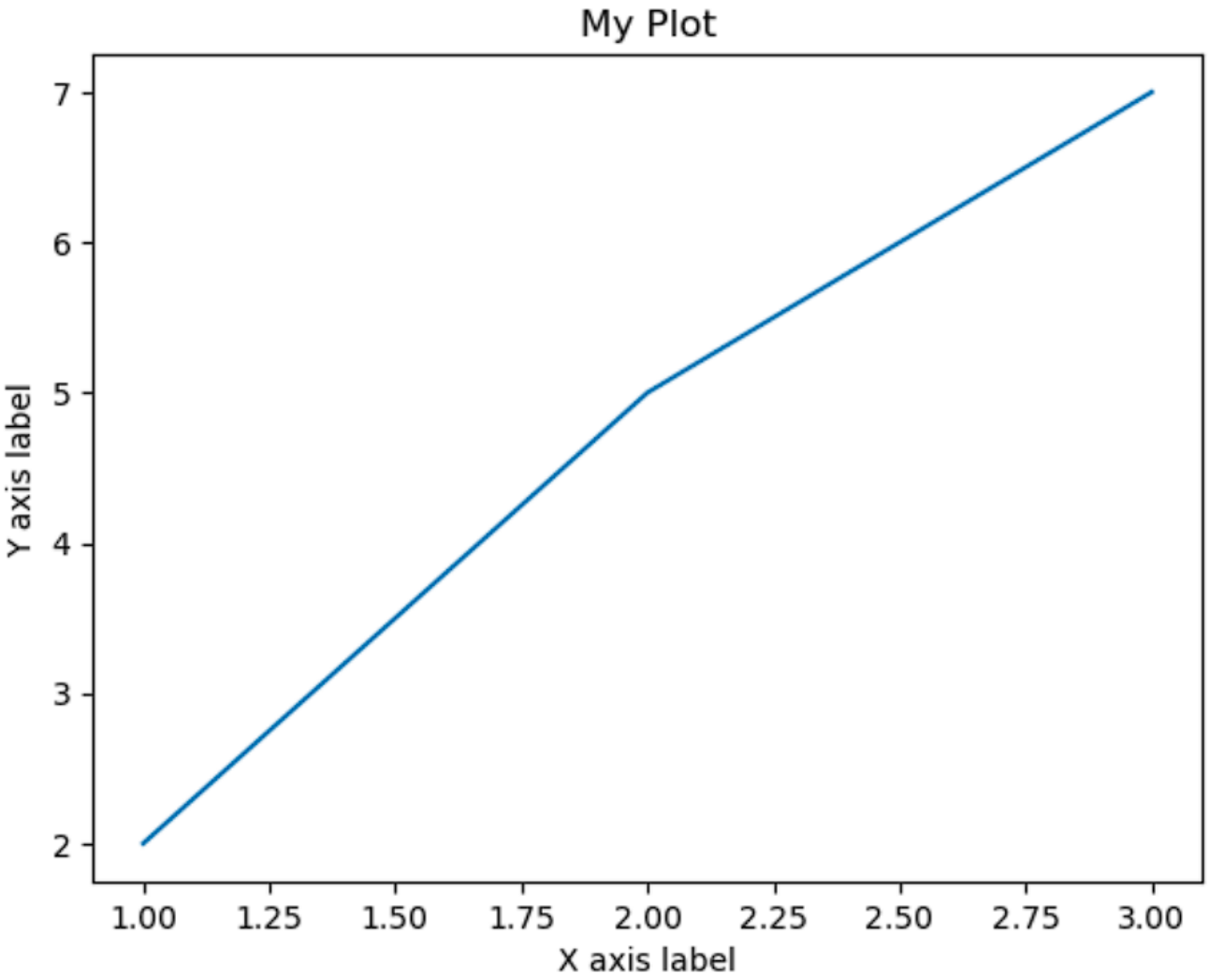
OUTPUT:

```
Defaulting to user installation because normal site-packages is not wri
teable
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\l
ib\site-packages (3.5.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\a
naconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anac
onda3\lib\site-packages (from matplotlib) (1.4.2)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda
3\lib\site-packages (from matplotlib) (9.2.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anac
onda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anacon
da3\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: numpy>=1.17 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib) (1.21.5)
Requirement already satisfied: pyparsing>=2.2.1 in c:\programdata\anaco
nda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib
\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
3.5.2
```

CODE:

```
import matplotlib.pyplot as plt
# Define the data to plot
x = [1, 2, 3]
y = [2, 5, 7]
# Plot the data
plt.plot(x, y)
# Add a title and labels
plt.title("My Plot")
plt.xlabel("X axis label")
plt.ylabel("Y axis label")
# Show the plot
plt.show()
```

OUTPUT:



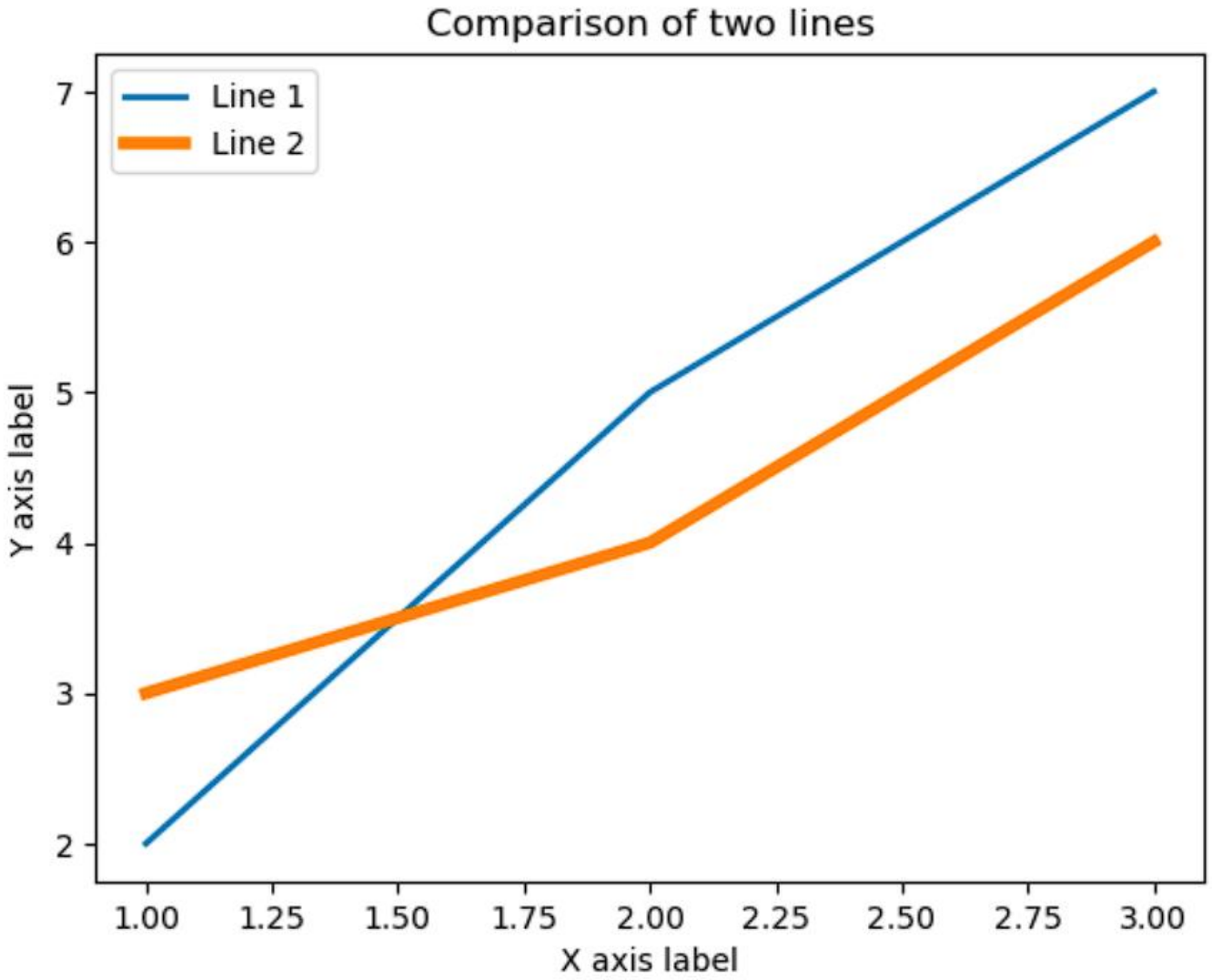
CODE:

```
# Define the data for the two lines
x = [1, 2, 3]
y1 = [2, 5, 7]
y2 = [3, 4, 6]
# Plot the two lines
plt.plot(x, y1, label='Line 1', linewidth=2)
plt.plot(x, y2, label='Line 2', linewidth=4)
# Add a title and labels
plt.title("Comparison of two lines")
plt.xlabel("X axis label")
plt.ylabel("Y axis label")
# Add a legend
plt.legend()
# Show the plot
plt.show()
```

OUTPUT:

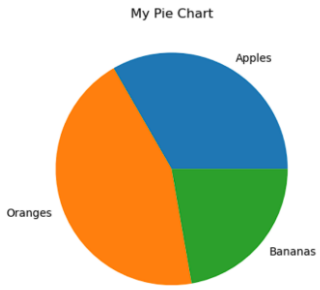
CODE:

```
# Define the data for the pie chart
labels = ['Apples', 'Oranges', 'Bananas']
sizes = [30, 40, 20]
# Plot the pie chart
```



plt.pie(sizes, labels=labels)
 # Add a title
plt.title("My Pie Chart")
 # Show the plot
plt.show()

OUTPUT:



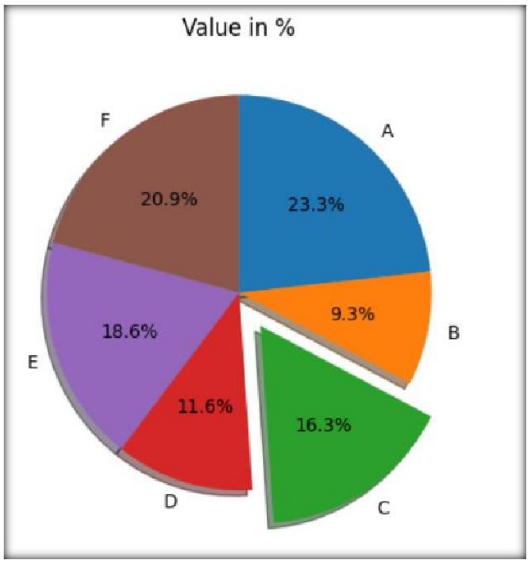
PRACTICAL 17

Q: Installation of Jupyter/ Google collab, and
- Survey of Jupyter Dashboard
- Adding new notebook

CODE:

```
import matplotlib.pyplot as plt

values =[23.3,9.3,16.3,11.6,18.6,20.9]

label=['A','B','C','D','E','F']

explode=[0,0,0.2,0,0,0]


plt.pie(values ,labels=label,autopct='%1.1f%%',explode=explode ,startangle=90 ,
counterclock=False,shadow=True)
plt.title("Value in %")plt.show()
```

OUTPUT:



PRACTICAL 18

Q: 1Read data.csv file from Kaggle and perform cleaning of bad data such as:

• Empty cells

➢ Detect no of missing values in a column(using isna())

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


# load the data csv file
df = pd.read_csv('data_new2.csv')#
missing_values = df.isna().sum()
print(missing_values)
```

```
Missing values:
 Duration      0
Pulse          0
Maxpulse       9
Calories      12
date           0
age          168
dtype: int64
```

➢ Remove rows containing empty cells(using dropna())

```
# Remove rows containing empty cells
df = df.dropna()
print(df)
```

```
Empty DataFrame
Columns: [Duration, Pulse, Maxpulse, Calories, date, age ]
Index: []
```

➢ Replace empty values in a column using any value(using fillna()) .

Replace empty values in a column using any value

df['Maxpulse'] = df['Maxpulse'].fillna('20')

print(df.head(10))

	Duration	Pulse	Maxpulse	Calories	date	age
0	60	110	20	NaN	23-12-1980	12.0
1	60	117	20	479.0	24-12-1980	NaN
2	60	103	20	340.0	25-12-1980	NaN
3	45	109	20	NaN	26-12-1980	NaN
4	45	117	20	NaN	27-12-1980	NaN
5	60	102	127.0	300.0	28-12-1980	NaN
6	60	110	20	374.0	29-12-1980	NaN
7	45	104	134.0	253.3	30-12-1980	NaN
8	30	109	133.0	NaN	31-12-1980	NaN
9	60	98	124.0	269.0	01-01-1981	NaN

➢ Replace empty values in a column using mean, median or mode

replacaing values using mean

column_m= df['Maxpulse'].median() print("Median of

maxpulse: ",column_m) df['Maxpulse'] =

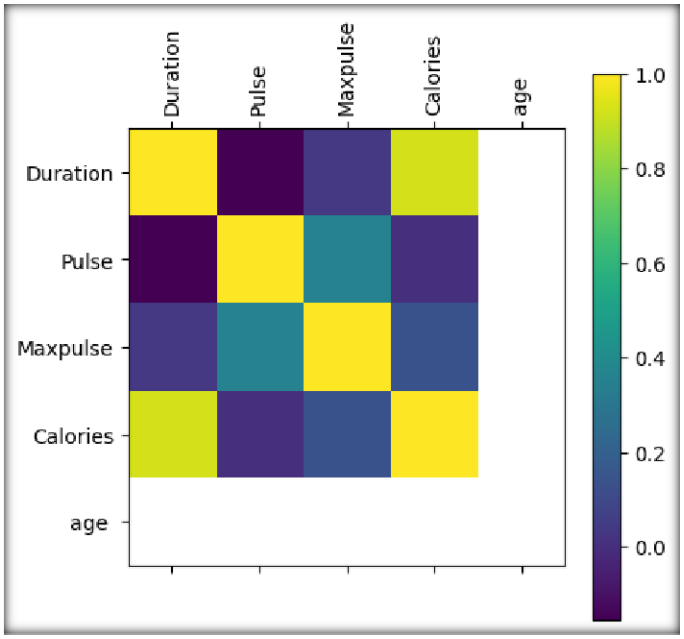
df['Maxpulse'].fillna(column_m)print(df.head(5))

```
Median of maxpulse:  130.5
   Duration  Pulse  Maxpulse    Calories        date  age
0        60    110     130.5  107.461538  23-12-1980  12.0
1        60    117     130.5  479.000000  24-12-1980   NaN
2        60    103     130.5  340.000000  25-12-1980   NaN
3        45    109     130.5  107.461538  26-12-1980   NaN
4        45    117     130.5  107.461538  27-12-1980   NaN
```

• Find the correlation between columns of the given dataset and visualize this dataset.

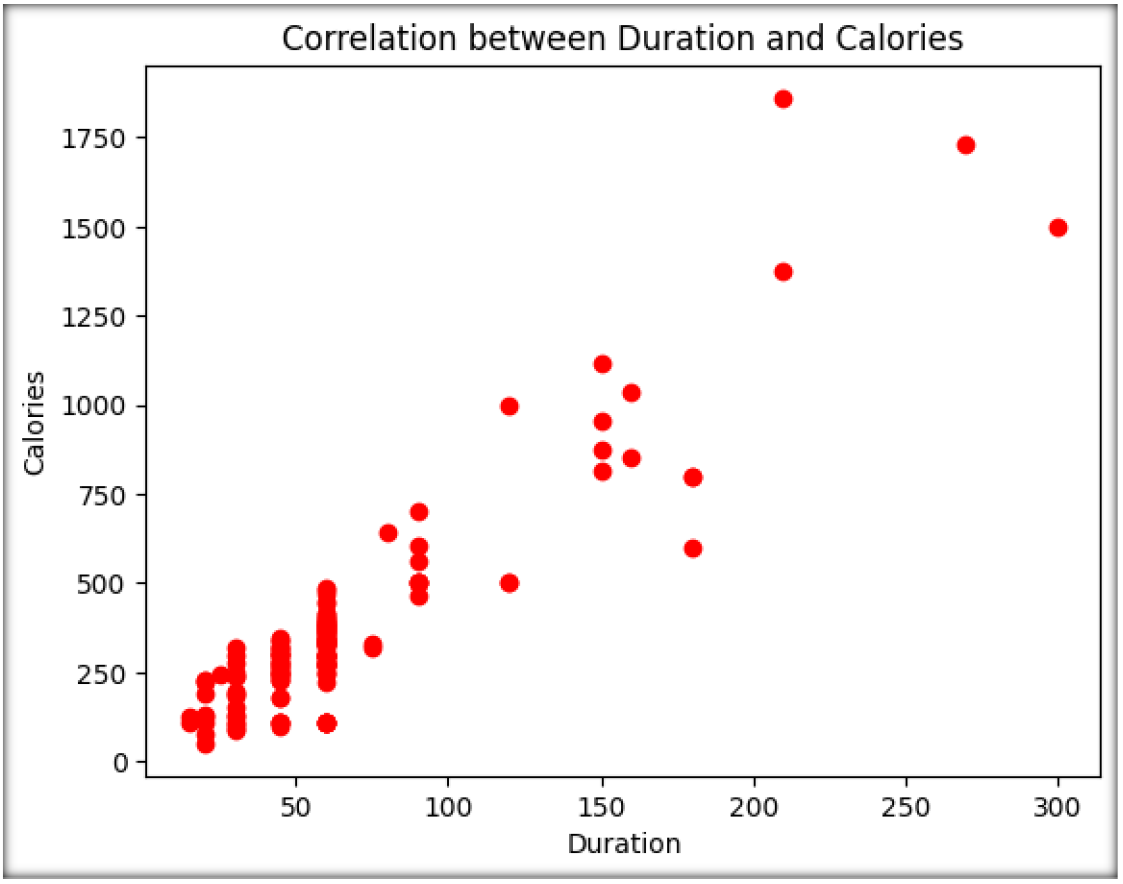
```
#corelation correlation =
df.corr()
print("Correlation:\n", correlation)#
Visualize the dataset
plt.matshow(correlation)
plt.xticks(range(len(correlation.columns)), correlation.columns, rotation=90)
plt.yticks(range(len(correlation.columns)), correlation.columns) plt.colorbar()
plt.show()
```

```
Correlation:
             Duration     Pulse  Maxpulse  Calories  age
Duration     1.000000 -0.155408  0.036115  0.924001   NaN
Pulse       -0.155408  1.000000  0.358985  0.004998   NaN
Maxpulse     0.036115  0.358985  1.000000  0.136753   NaN
Calories     0.924001  0.004998  0.136753  1.000000   NaN
age               NaN       NaN       NaN       NaN   NaN
```



• Show correlation between Duration and Calories columns using scatter plot.

```
# Show correlation between Duration and Calories columns using scatter plot
plt.scatter(df['Duration'] ,df['Calories'],color='red')
plt.xlabel('Duration')
plt.ylabel('Calories')
plt.title('Correlation between Duration and Calories')plt.show()
```

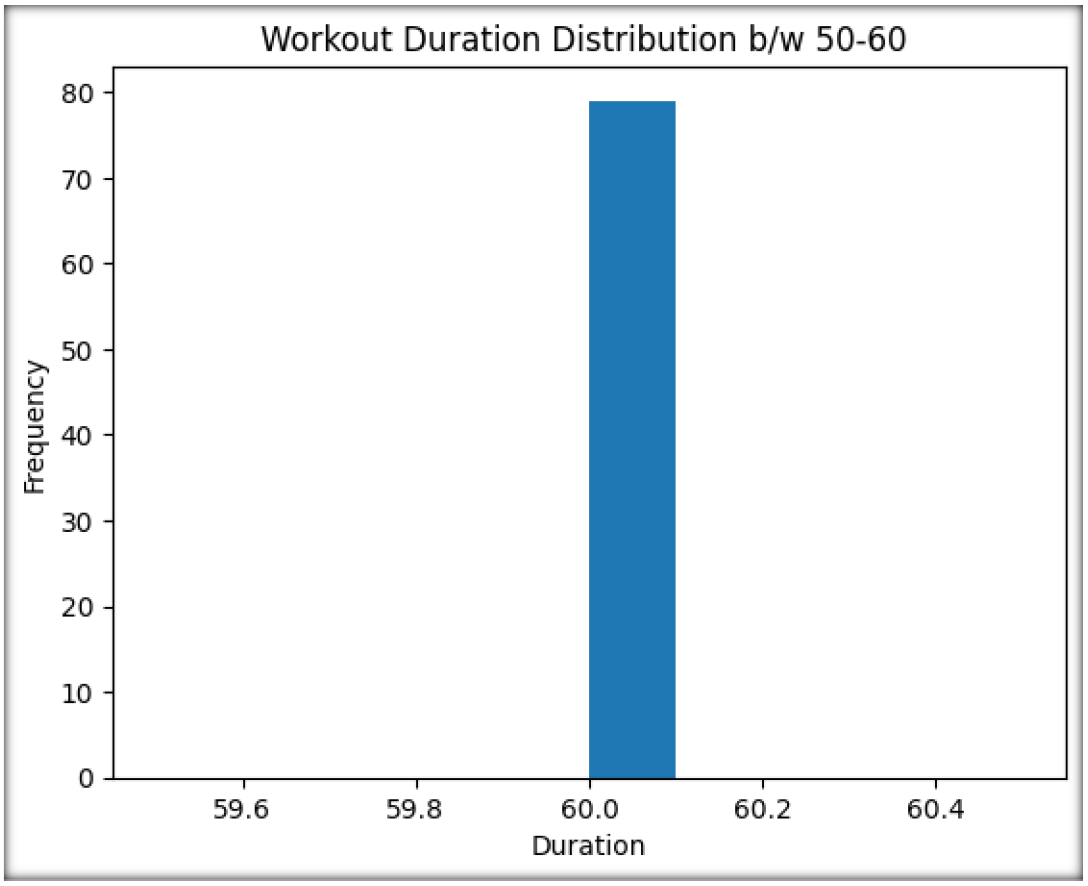


• Draw a histogram to show how many workouts lasted between 50 and 60 minutes?

Code:

```
# Draw a histogram to show how many workouts lasted between 50 and 60 minutes
filtered_df = df[(df['Duration'] >= 50) & (df['Duration'] <= 60)]
plt.hist(filtered_df['Duration'], bins=10)
plt.xlabel('Duration')
plt.ylabel('Frequency')
plt.title('Workout Duration Distribution b/w 50-60 ')plt.show()
```

OUTPUT:



PRACTICAL 19

Q:

```
import pandas as pd

# Set the maximum number of rows to display
pd.set_option('display.max_rows',None)


# Read the CSV file
file_path = 'Titanic-Dataset.csv' # Replace 'data.csv' with the actual path to your .csvfile
df = pd.read_csv(file_path)


# Display the entire DataFrame
print(df)
```

OUTPUT:

	PassengerId	Survived	Pclass	\
0	1	0	3	
1	2	1	1	
2	3	1	3	
3	4	1	1	
4	5	0	3	
5	6	0	3	
6	7	0	1	
7	8	0	3	
8	9	1	3	
9	10	1	2	
10	11	1	3	
11	12	1	1	
12	13	0	3	
13	14	0	3	
14	15	0	3	
15	16	1	2	
16	17	0	3	
17	18	1	2	

	Name	Sex	Age	SibSp	\
0	Braund, Mr. Owen Harris	male	22.00	1	
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.00	1	
2	Heikkinen, Miss. Laina	female	26.00	0	
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.00	1	
4	Allen, Mr. William Henry	male	35.00	0	
5	Moran, Mr. James	male	NaN	0	
6	McCarthy, Mr. Timothy J	male	54.00	0	
7	Palsson, Master. Gosta Leonard	male	2.00	3	
8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.00	0	
9	Nasser, Mrs. Nicholas (Adele Achem)	female	14.00	1	
10	Sandstrom, Miss. Marguerite Rut	female	4.00	1	
11	Bonnell, Miss. Elizabeth	female	58.00	0	
12	Saundercock, Mr. William Henry	male	20.00	0	

	Parch	Ticket	Fare	Cabin	Embarked
0	0	A/5 21171	7.2500	NaN	S
1	0	PC 17599	71.2833	C85	C
2	0	STON/O2. 3101282	7.9250	NaN	S
3	0	113803	53.1000	C123	S
4	0	373450	8.0500	NaN	S
5	0	330877	8.4583	NaN	Q
6	0	17463	51.8625	E46	S
7	1	349909	21.0750	NaN	S
8	2	347742	11.1333	NaN	S
9	0	237736	30.0708	NaN	C
10	1	PP 9549	16.7000	G6	S
11	0	113783	26.5500	C103	S
12	0	A/5. 2151	8.0500	NaN	S
13	5	347082	31.2750	NaN	S
14	0	350406	7.8542	NaN	S
15	0	248706	16.0000	NaN	S
16	1	382652	29.1250	NaN	Q
17	0	244373	13.0000	NaN	S

PRACTICAL 20

Q: Write a python program to:

- Read contents of a JSON file in Jupyter Notebook with or withoutPandas.
- Create JSON data in a dictionary and convert it to a pandas DataFrame.
- Parse a JSON file(Hint: convert from JSON to Python)

Code:

```
import pandas as pd# with
pandas print("with panda ")
dat=pd.read_json('example2.json')print(data)
# withoutpandas
print("without panda ") filepath="D:\DATA_SCIENCE_USING_PYTHON\example2.json"with
open(filepath,'r')as file:
    data=json.load(file)print(data)
```

```
with panda
[{'Name': 'Ashish', 'Age': 20, 'City': 'Patna'}, {'Name': 'Khushi', 'Age': 19, 'City': 'Dwarka'}, {'Name': 'Aman', 'Age': 29,
'City': 'Vasantkunj'}]
without panda
[{'Name': 'Ashish', 'Age': 20, 'City': 'Patna'}, {'Name': 'Khushi', 'Age': 19, 'City': 'Dwarka'}, {'Name': 'Aman', 'Age': 29,
'City': 'Vasantkunj'}]
```

```
# create a json data and read it ina dataframe
import pandas as pd

data = {
    "Name": ["akansha","ritik","komal","sachin"],"Age": [19,21,18,20],
    "City": ["Delhi","Azadpur","Bindapur","Sakarpur"]
}
df = pd.DataFrame(data)
```

Print the DataFrameprint(df)

```
      Name  Age      City
0  akansha   19     Delhi
1    ritik   21   Azadpur
2    komal   18  Bindapur
3   sachin   20  Sakarpur
```

```
# parse a json file
```

```
import json
file_path = "D:\DATA_SCIENCE_USING_PYTHON\example2.json"with open(file_path, "r") as file:
    json_content = json.load(file)


for item in json_content: print("Name:", item["Name"])
    print("Age:", item["Age"])
    print("City:", item["City"])
```

OUTPUT:

```
Name: Ashish
Age: 20
City: Patna
Name: Khushi
Age: 19
City: Dwarka
Name: Aman
Age: 29
City: Vasantkunj
```

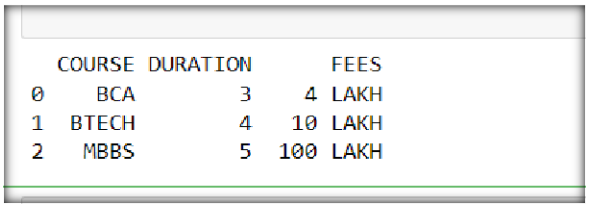
PRACTICAL 21

Q: Create and parse a .xml file to a DataFrame in python.

Code:

```
#  CREATE and parse a xml file to data frame
from lxml import objectifyimport pandas as pd

xml = objectify.parse(open("MYDOC.XML"))root = xml.getroot()

df = pd.DataFrame()

for i in range(0,3):
    obj = root.getchildren()[i].getchildren()
    row = dict(zip(['COURSE', 'DURATION', 'FEES'],
                   [obj[0].text, obj[1].text, obj[2].text]))row_s = pd.Series(row)
    row_s.name = i
    df = pd.concat([df, row_s], axis=1)

df = df.T print(df)
```

OUTPUT:



	COURSE	DURATION	FEES
0	BCA	3	4 LAKH
1	BTECH	4	10 LAKH
2	MBBS	5	100 LAKH

PRACTICAL 22

Q: Write short python codes to demonstrate:

- Uploading
- Streaming
- Sampling

Code:

```
# uploading small amount of data

print("uploading small amount of data :")

with open ("color1.txt",'r') as open_file:
    print('color1.txt content:\n' + open_file.read())
```

```
uploading small amount of data :
color1.txt content:

color  value
red     1
pink    2
orange  3
green   4
black   5
white   6
grey    7
blue    8
```

```
# streaming data into memory

print("Data streaming example : ")

with open("color1.txt", 'r') as open_file:
    for observation in open_file:
        print('Reading Data: ' + observation)
```

```
Data streaming example :
Reading Data:

Reading Data: color  value

Reading Data: red     1

Reading Data: pink      2

Reading Data: orange  3

Reading Data: green   4

Reading Data: black   5

Reading Data: white   6

Reading Data: grey    7

Reading Data: blue    8
```

 # sampling data in different ways

```
n = 2
with open("color1.txt", 'r') as open_file:
   for j, observation in enumerate(open_file):
   if j % n==0:
        print('Reading Line: ' + str(j) +' Content: ' +
        observation)
```

```
Reading Line: 0 Content:

Reading Line: 2 Content: red     1

Reading Line: 4 Content: orange  3

Reading Line: 6 Content: black   5

Reading Line: 8 Content: grey    7
```

PRACTICAL 23

Q: Installation of Jupyter/ Google collab, and
- Survey of Jupyter Dashboard
- Adding new notebook

 # parsing atext file

```
def  parse_txt_file(file_path):

with open(file_path, 'r') as file:
        lines = file.readlines()
        for line in lines:
            # Process each line of the text file
            # Here, you can perform any required operations on the data

            print(line.strip())  # Example: print the stripped line
parse_txt_file('myfile.txt')
```

```
In conclusion, deploying a website through Azure offers significant advantages and opportunities for organizations seeking a re
liable and scalable cloud platform. Throughout this report, we have explored the concept of deploying a website through Azure,
including understanding Azure itself, the benefits it provides, and the various Azure services relevant to website deployment.
By leveraging Azure, organizations can take advantage of its robust infrastructure, global reach, and extensive range of servic
es. Azure Web Apps and Azure Virtual Machines offer flexible deployment options to accommodate different website requirements,
while services like Azure SQL Database, Azure Cosmos DB, and Azure Active Directory enable seamless integration of
databases, user authentication, and data storage.
```

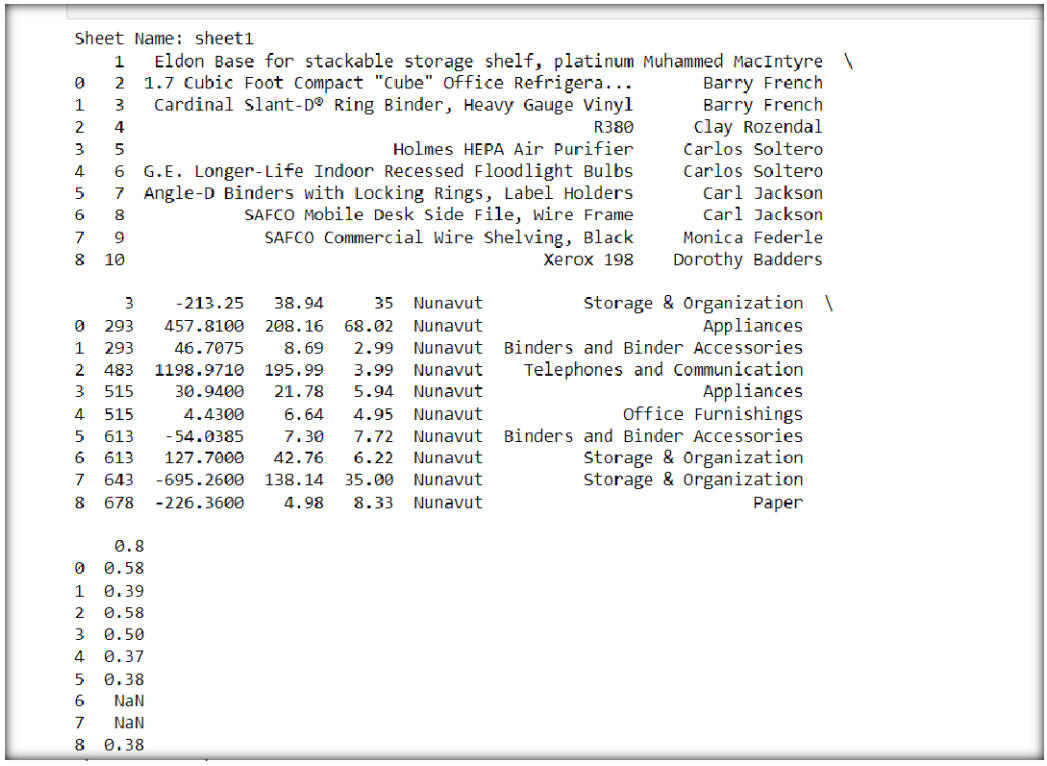
parsing a csv file

```
import csv
def parse_csv_file(file_path):

with open(file_path, 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            print(row)


file_path = 'myfile.csv'
parse_csv_file(file_path)
```

```
['A', 'B', 'C', 'D']
['1', '2', '3', '4']
['5', '6', '7', '8']
['9', '10', '11', '12']
```

reading from a multiple excel sheets

```
import pandas as pd


def read_excel_file(file_path):
    excel_data = pd.read_excel(file_path, sheet_name=None)

    for sheet_name, sheet_data in excel_data.items():
        # Process each sheet in the Excel file
        # Here, you can perform any required operations on the data

        print(f"Sheet Name: {sheet_name}")
        print(sheet_data)  # Example: print the sheet data as DataFrame


# Usage
file_path = 'multiple.xlsx' # Replace 'data.xlsx' with the actual path to your Excel file

read_excel_file(file_path)
```



```
Sheet Name: sheet1
    1   Eldon Base for stackable storage shelf, platinum Muhammed MacIntyre  \
0   2  1.7 Cubic Foot Compact "Cube" Office Refrigera...       Barry French
1   3   Cardinal Slant-D® Ring Binder, Heavy Gauge Vinyl       Barry French
2   4                                               R380      Clay Rozendal
3   5                           Holmes HEPA Air Purifier     Carlos Soltero
4   6  G.E. Longer-Life Indoor Recessed Floodlight Bulbs     Carlos Soltero
5   7  Angle-D Binders with Locking Rings, Label Holders       Carl Jackson
6   8            SAFCO Mobile Desk Side File, Wire Frame       Carl Jackson
7   9              SAFCO Commercial Wire Shelving, Black     Monica Federle
8  10                                         Xerox 198    Dorothy Badders

     3    -213.25   38.94     35  Nunavut          Storage & Organization  \
0  293   457.8100  208.16  68.02  Nunavut                      Appliances
1  293    46.7075    8.69   2.99  Nunavut  Binders and Binder Accessories
2  483  1198.9710  195.99   3.99  Nunavut    Telephones and Communication
3  515    30.9400   21.78   5.94  Nunavut                      Appliances
4  515     4.4300    6.64   4.95  Nunavut             Office Furnishings
5  613   -54.0385    7.30   7.72  Nunavut  Binders and Binder Accessories
6  613   127.7000   42.76   6.22  Nunavut          Storage & Organization
7  643  -695.2600  138.14  35.00  Nunavut          Storage & Organization
8  678  -226.3600    4.98   8.33  Nunavut                           Paper

    0.8
0  0.58
1  0.39
2  0.58
3  0.50
4  0.37
5  0.38
6   NaN
7   NaN
8  0.38
```

```
Sheet Name: sheet2
    1                                          fruit jam Muhammed MacIntyre  \
0   2                                Office Refrigerators       Barry French
1   3                                               table       Barry French
2   4                                            Computer       Clay Rozendal
3   5                                phlipis Air Purifier     Carlos Soltero
4   6                                          Ring lights     Carlos Soltero
5   7  Angle-D Binders with Locking Rings, Label Holders       Carl Jackson
6   8            SAFCO Mobile Desk Side File, Wire Frame       Carl Jackson
7   9              SAFCO Commercial Wire Shelving, Black     Monica Federle
8  10                                            Xerox 198    Dorothy Badders

     3  -213.25       38.94     35 Unnamed: 7          Storage & Organization  \
0  293  -212.25  208.160000  68.02    nothing                      Appliances
1  293  -211.25    8.690000   2.99    nothing  Binders and Binder Accessories
2  483  -210.25  195.990000   3.99        NaN    Telephones and Communication
3  515  -209.25   21.780000   5.94    nothing                      Appliances
4  515  -208.25    6.640000   4.95        NaN             Office Furnishings
5  613  -207.25   26.699333   7.72        NaN  Binders and Binder Accessories
6  613  -206.25   11.461048   6.22    nothing          Storage & Organization
7  643  -205.25   -3.777238  35.00    nothing          Storage & Organization
8  678  -204.25  -19.015524   8.33    nothing                           Paper

    0.8
0  0.58
1  0.39
2  0.58
3  0.50
4  0.37
5  0.38
6   NaN
7   NaN
8  0.38
```

PRACTICAL 24

Q: Write python codes to access data in unstructured flat-file form to:

- Read an image from a public domain and render it on gray scale onscreen.
- Discover the image type and size
- Resize the image by-

➢ Manipulating image array, or

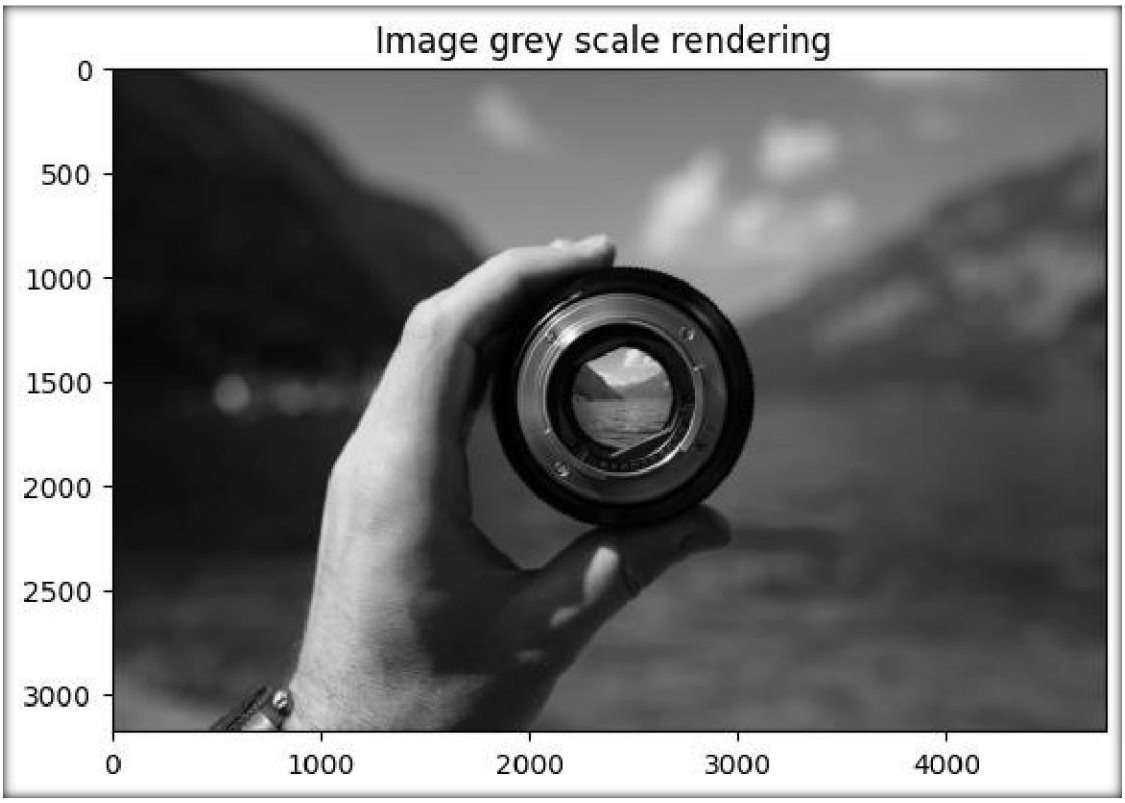
➢ Use resize(), and then

Convert the resized image to a dataset format for further analysis

Code:

```
# gray scaling
from skimage.io import imread
from skimage.transform import resizefrom matplotlib import
pyplot as plt import matplotlib.cm as cm
example_file=("https://www.seiu1000.org/sites/main/files/main-
images/camera_lense_0.jpeg")
image1 = imread(example_file ,as_gray=True)plt.imshow(image1,
cmap=cm.gray)
plt.show()
# type and size of image
print("Data type of image is : %s, shape:%s " %(type(image1),image1.shape))
```

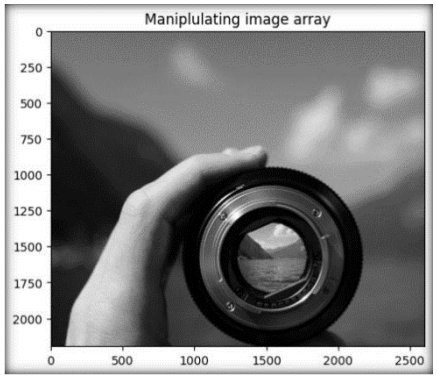
```
Data type of image is : <class 'numpy.ndarray'>, shape:(3176, 4764)
```



 # manipulating image array

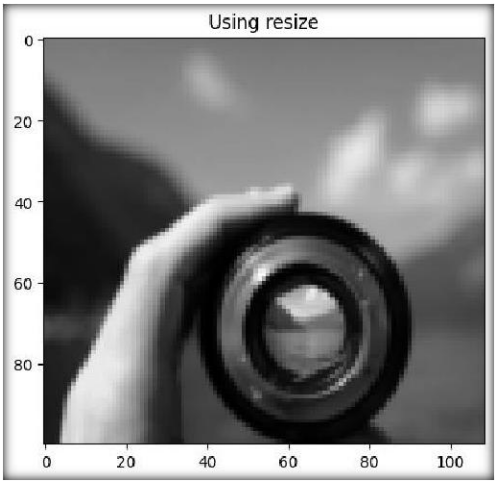
```
image2=image1[5:2200 ,900:3500]
plt.imshow(image2,cmap=cm.gray)
plt.title("Maniplulating image array")
plt.show()
```

OUTPUT:



 # using resize() function

```
image3=resize(image2,(100,109),mode='symmetric')
plt.imshow(image3,cmap=cm.gray)
plt.title("Using resize")plt.show()
```



convert to dataset

```
image_row = image3.flatten()
df = pd.DataFrame(image_row, columns=["Pixel Intensity"])
csv_filename = "image3_dataset.csv" # Replace with the desired file namedf.to_csv(csv_filename, index=False)
print(df.head(89))
```

	Pixel Intensity
0	0.445125
1	0.445158
2	0.445851
3	0.445763
4	0.446011
..	...
84	0.432439
85	0.431630
86	0.430670
87	0.430156
88	0.430938

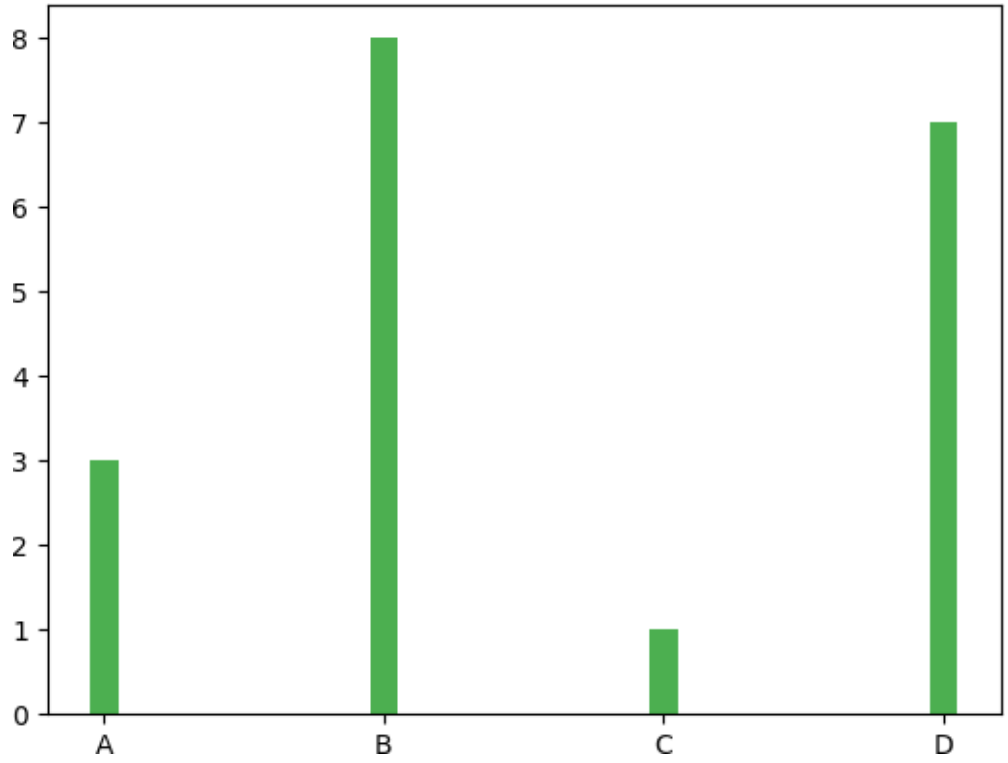
[89 rows x 1 columns]

PRACTICAL 25

Q: Write python codes to demonstrate usage of following functions:

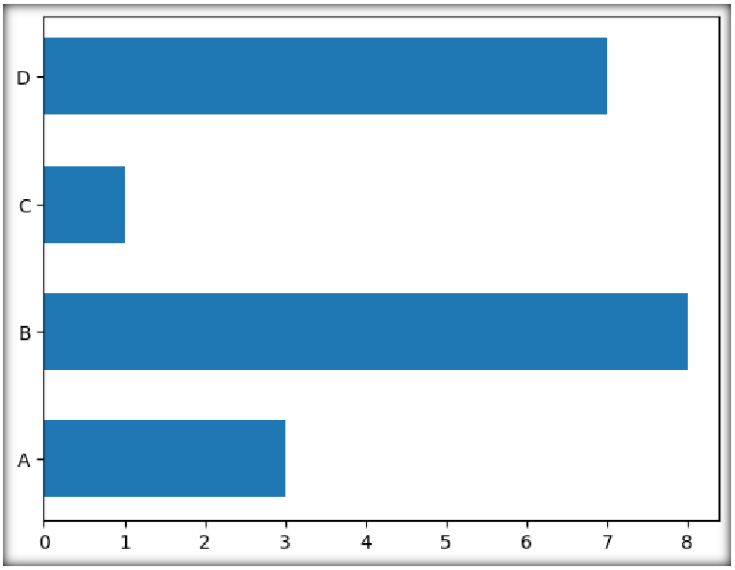
- bar(), barh() while changing their default color, width, and height .

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


x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 7])


plt.bar(x,y,color = "#4CAF50",width = 0.1)
plt.show()
```




```
#bar h
import matplotlib.pyplot as plt

import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 7])
```

```
plt.barh(x, y, height = 0.6 )
plt.show()
```



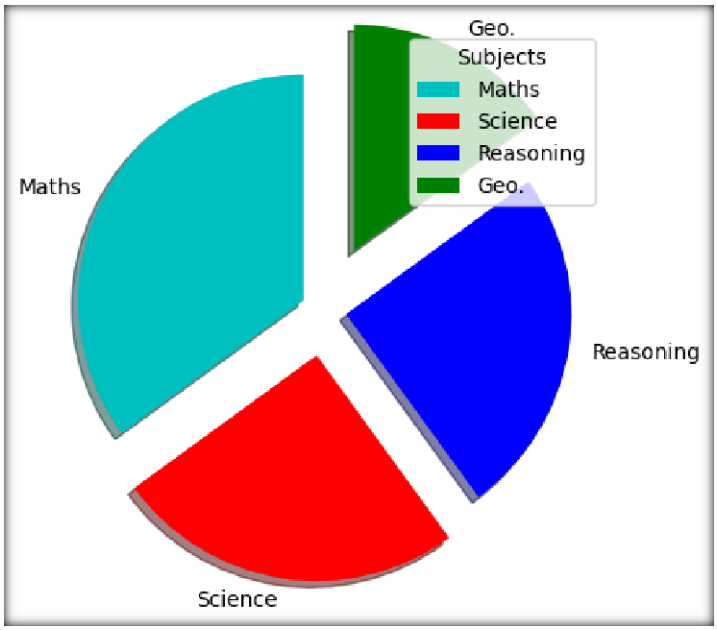
• hist() for normal data distribution for height of 250 people

• pie() with labels, legends with header, shadow, change start angle anddefault colors, explode all wedges.

Code:

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


Marks = np.array([99,98,78,69])
subject= ['Maths','Science','Reasoning','Geo.']myexplode = [0.1, 0.2,
0.1, 0.3]
mycolors = ['c','r','b','g']
plt.pie(y, labels = subject, startangle = 90, explode = myexplode, shadow = True,colors = mycolors)
plt.legend(title = "Subjects",loc=1)

plt.show()
```

OUTPUT:

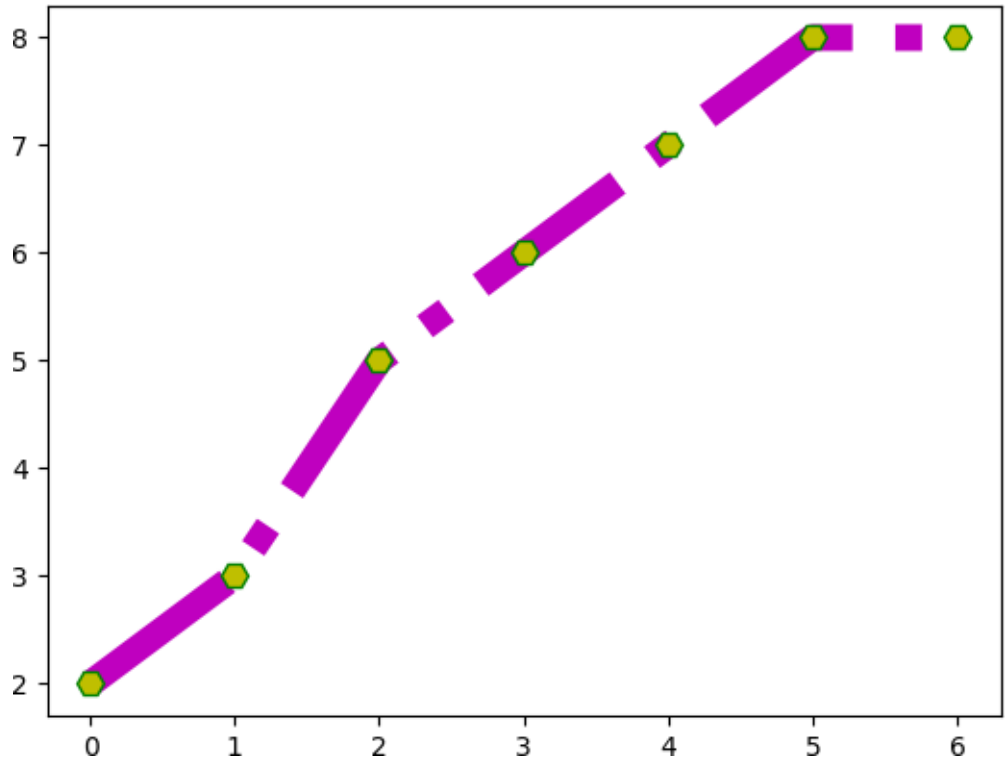


PRACTICAL 26

Q: Write a python program to draw a line with y points only(i.e. default xpoints) having following properties:

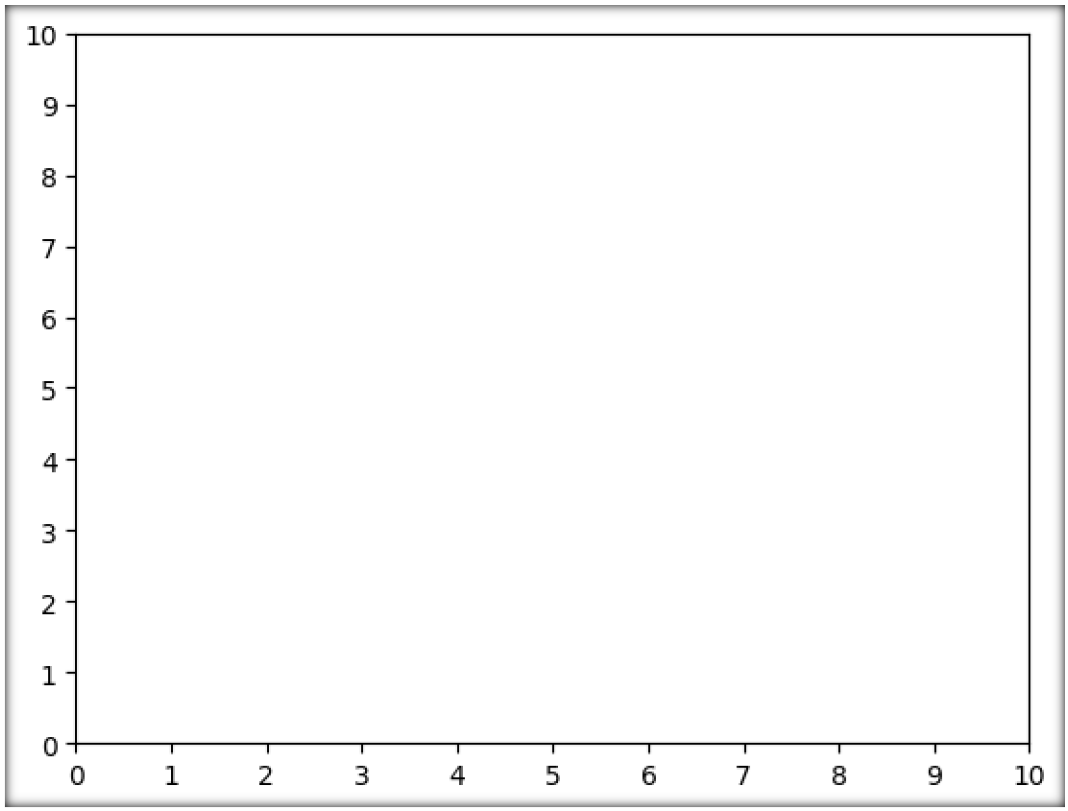
- Dash-dot lines of Magenta color, size -10 and Hexagon marker(of size10, edge color- green, face color - yellow).

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

ypoints = np.array([2,3,5,6,7,8,8])

plt.plot(ypoints, ls = '-.', c = 'm', lw = 10, marker = 'H', ms = 10, mec = 'g', mfc = 'y')plt.show()
```



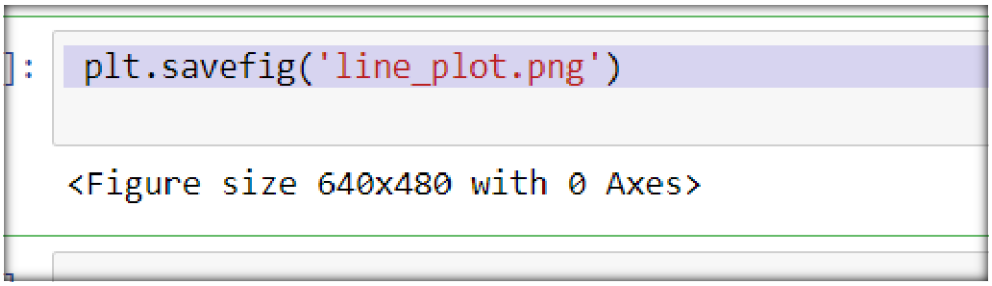
• Get the default axes and set ticks as [0,1,2,3,4,5,6,7,8,9,10]

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.set_xticks([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
ax.set_xticklabels([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

ax.set_yticks([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
ax.set_yticklabels([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

# Display the plot
plt.show()
```



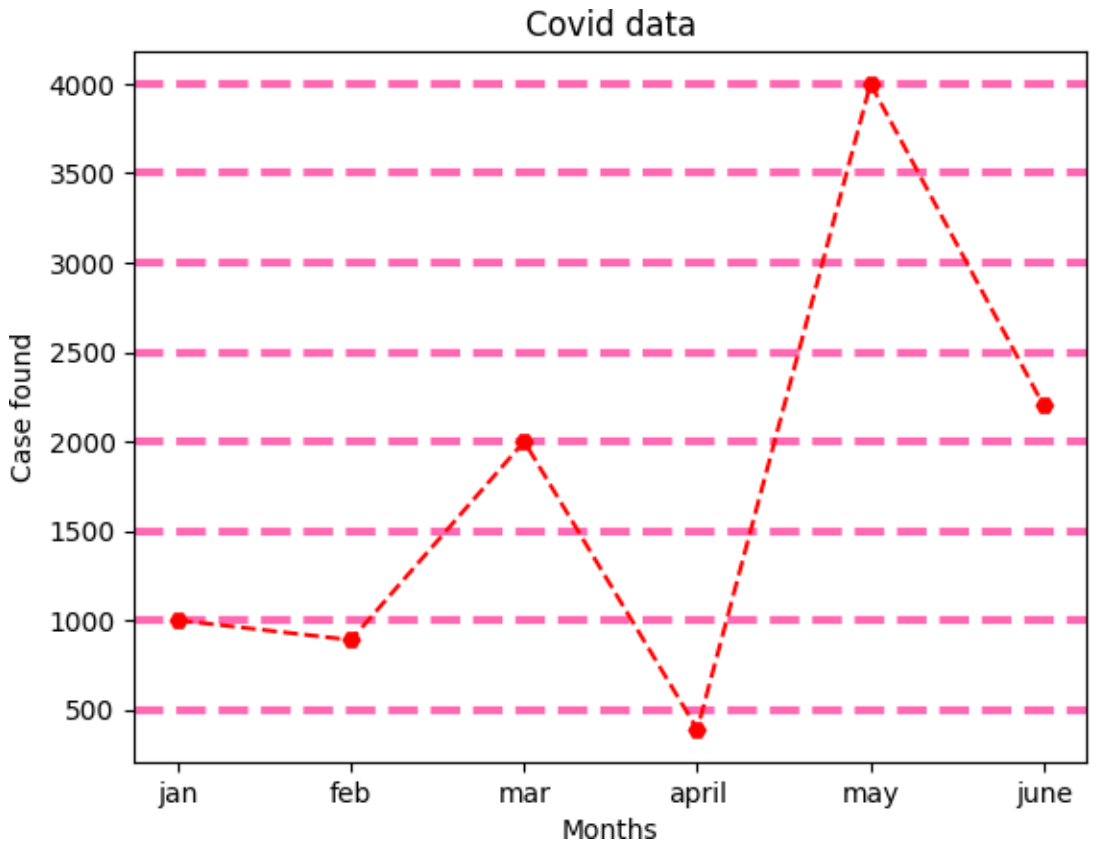
• Save the figure to your current working directory or anywhere else on your system.

‘plt.savefig('line_plot.png')



• Add colored horizontal grid(y-axis), and also change default linewidthand linestyle of grid.

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

months= np.array(['jan','feb','mar','april','may','june'])

cases = np.array([1000,890,2000,390,4000,2200])

plt.title("Covid data")

plt.xlabel("Months")

plt.ylabel("Case found")

plt.plot(months,cases,'--rH')

plt.grid(axis = 'y' , color = 'hotpink', linestyle = '--', linewidth = 3)

plt.show()
```

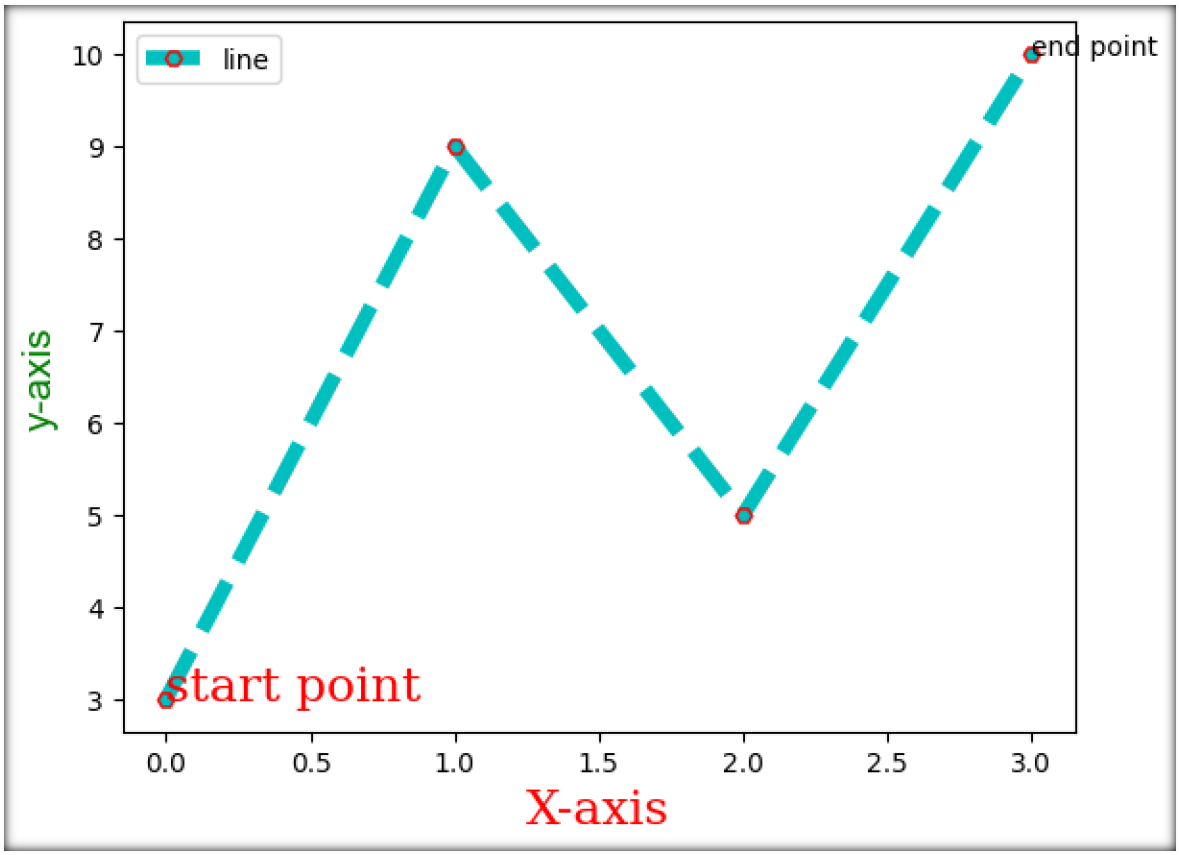


• Annotate first and last points of line, and Add title, labels and legends to thegraph(change the font, color and size of labels and title).

Code:

```
# E
import matplotlib.pyplot as plt

import numpy as np


ypoints = np.array([3, 9, 5, 10])
font1 = {'family':'serif','color':'red','size':18}
font2 = {'family':'arial','color':'green','size':15}


plt.plot(ypoints, "Hc--" ,linewidth = '5.5',mec='r')
plt.xlabel("X-axis",font1)
plt.ylabel("y-axis",font2)
plt.text(0,3,"start point",font1)
plt.text(3,10,"end point")
plt.legend(['line'])
plt.show()
```

OUTPUT:

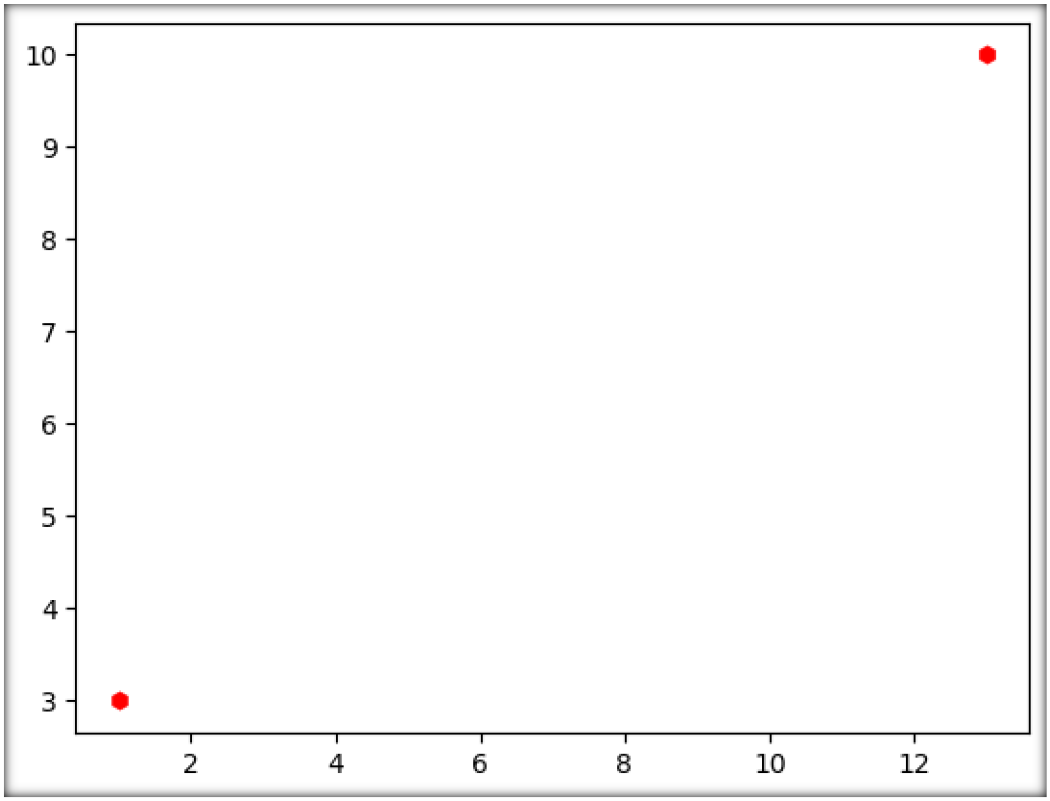


PRACTICAL 27

Q: Write python codes for following:

• Plot only first and last markers of a line(i.e draw a line without line).import matplotlib.pyplot as plt

```
import numpy as np


xpoints = np.array([1, 13])
ypoints = np.array([3, 10])


plt.plot(xpoints, ypoints, 'hr')

plt.show()
```



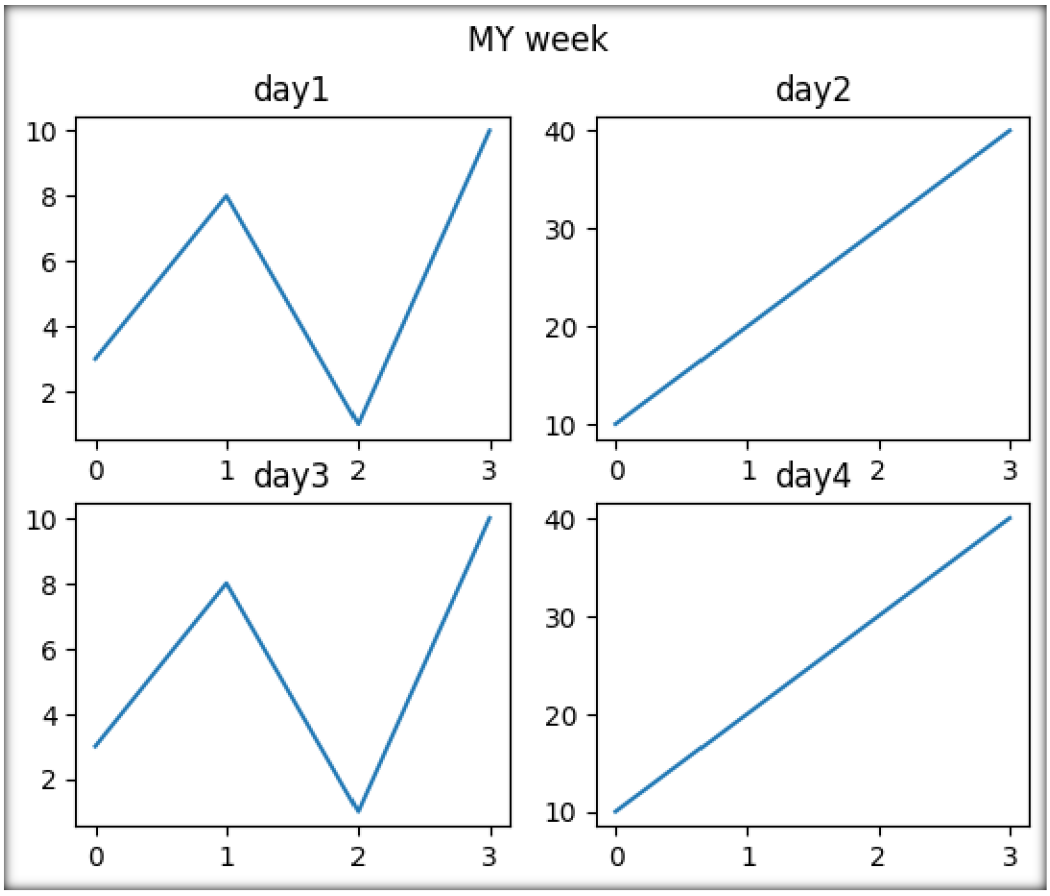
• Display multiple plots in a single figure.

Code:

```
import matplotlib.pyplot as plt

import numpy as np


x = np.array([0, 1, 2, 3])

y = np.array([3, 8, 1, 10])


plt.subplot(2, 2, 1)

plt.plot(x,y)

plt.title("day1")


x = np.array([0, 1, 2, 3])

y = np.array([10, 20, 30, 40])


plt.subplot(2, 2, 2)

plt.plot(x,y)

plt.title("day2")
```

```
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 2, 3)
plt.plot(x,y)
plt.title("day3")

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 2, 4)
plt.plot(x,y) plt.title("day4")

plt.suptitle("MY week")

plt.show()
```

OUTPUT:

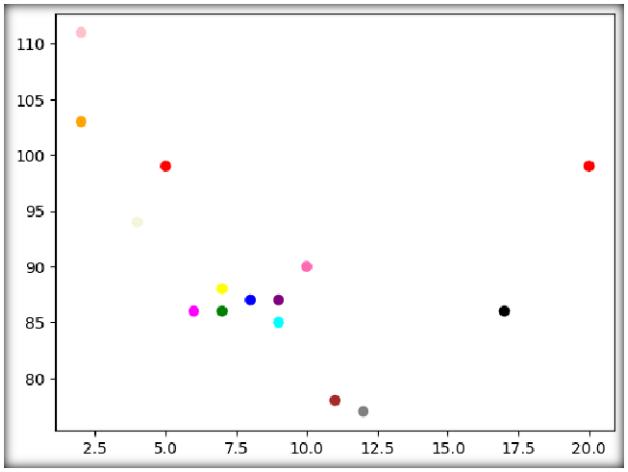


PRACTICAL 28

Q: Draw a scatter plot to show relationship between speed and age of 15 carsin movement with following properties:

• set your own color for each dot in scatter plot.

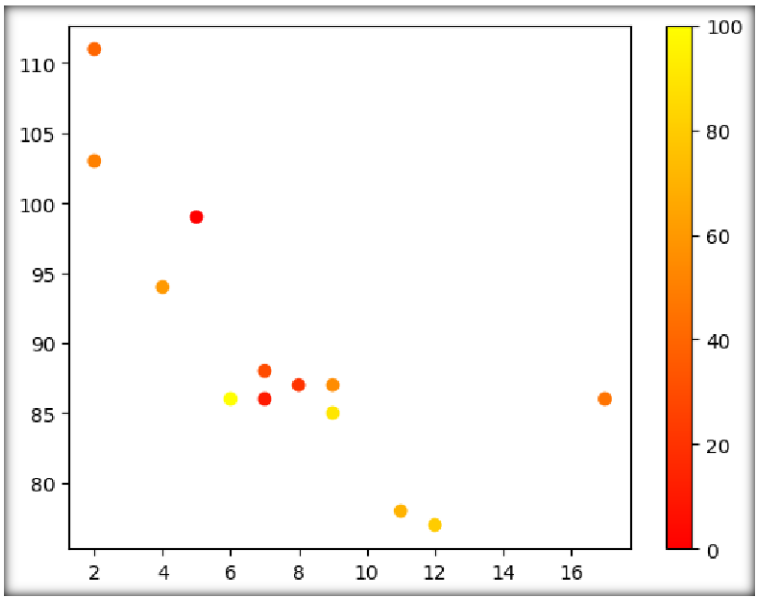
```
import matplotlib.pyplot as plt
import numpy as np

# the age and speed of 15 cars:
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6,20,10])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86,99,90])
colors = np.array(["red","green","blue","yellow","pink","black","orange","purple","beige","brown
","gray","cyan","magenta",'red','hotpink'])

plt.scatter(x, y, c=colors)

plt.show()
```



• set the color of each dot for autumn color map, also show the autumn colorbar on side .

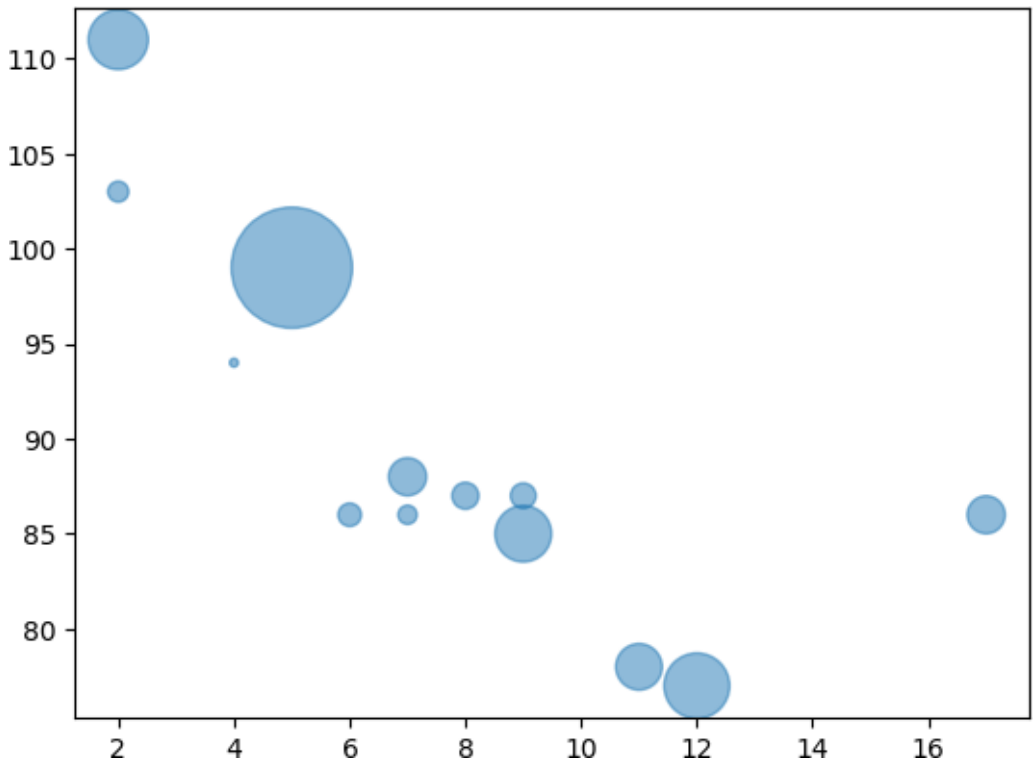
```
# b
import matplotlib.pyplot as plt
import numpy as np
```

```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x, y, c=colors, cmap='autumn')

plt.colorbar()

plt.show()
```



• set your own size for dots and also adjust the transparency of the dots .import matplotlib.pyplot as plt

```
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
sizes = np.array([2044,50,100,200,500,200,60,90,10,300,600,445,75])

plt.scatter(x, y, s = sizes, alpha=0.5)
```



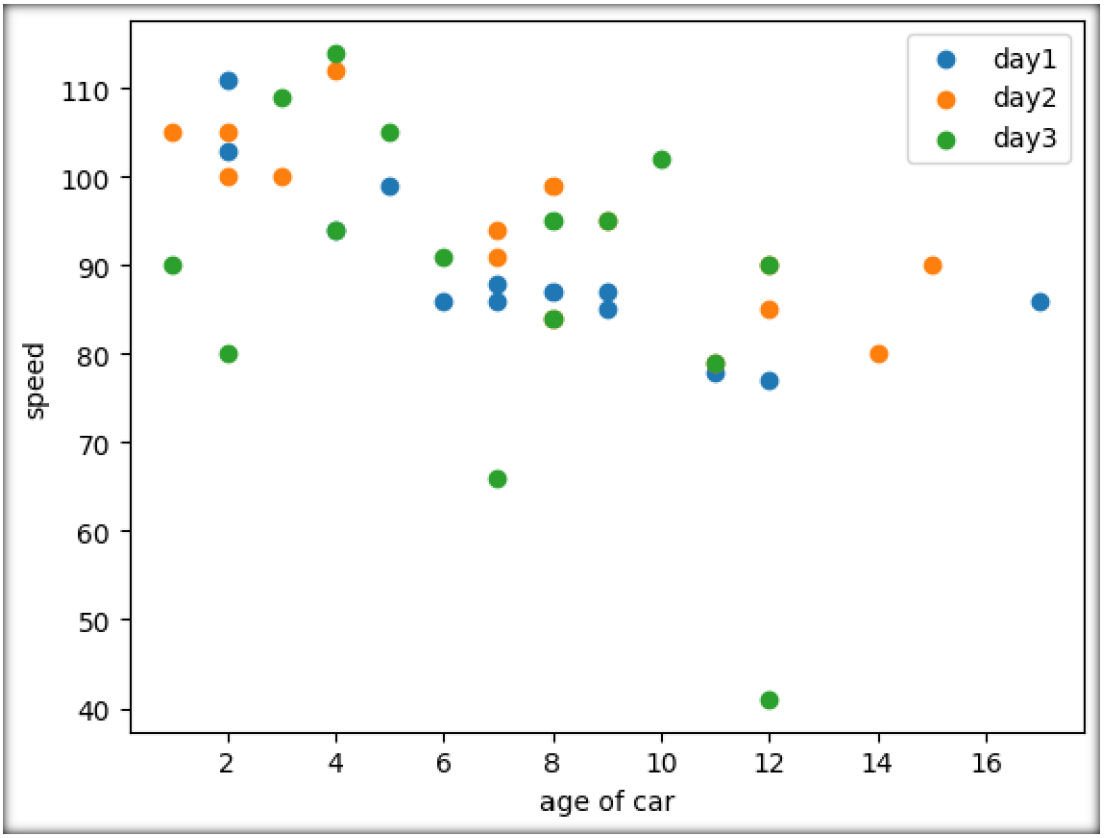
• Comparison of speed and age of 15 cars for at least 3 days.

Code:

```
# d

import matplotlib.pyplot as plt

import numpy as np


#day one, the age and speed of 15 cars:

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)


#day two, the age and speed of 15 cars:

x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])

y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])

plt.scatter(x, y)


#day three, the age and speed of 15 cars:

x = np.array([2,5,8,10,12,8,1,9,4,3,11,4,6,7,12])

y = np.array([80,105,84,102,90,95,90,95,94,109,79,114,91,66,41])
```

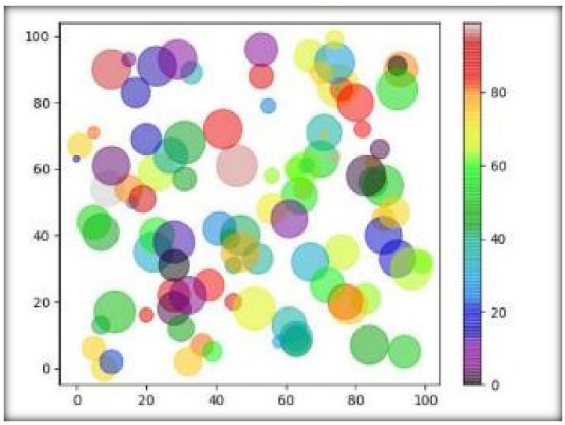
```
plt.scatter(x, y) plt.xlabel("age of car")
plt.ylabel("speed")

plt.legend(['day1','day2','day3'])plt.show()
```

OUTPUT:



PRACTICAL 29

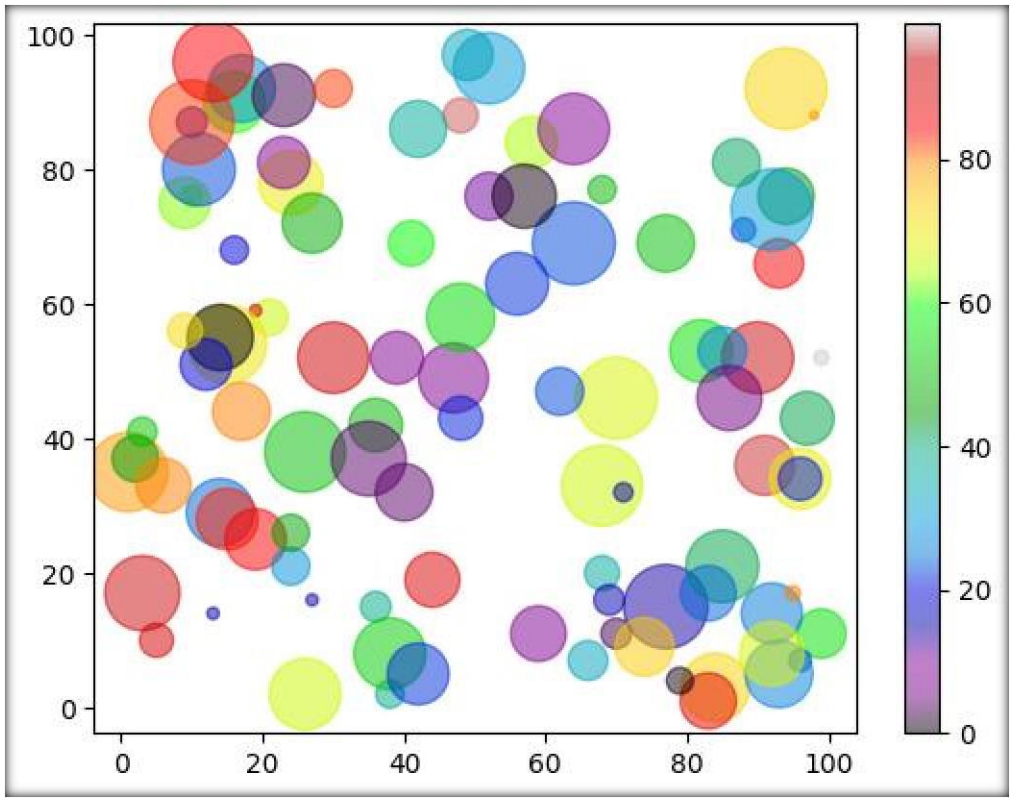
Q: Draw the following graph along with its colormap



Code:

```
import matplotlib.pyplot as plt

import numpy as np


x = np.random.randint(100, size=(100))y =
np.random.randint(100, size=(100))
colors = np.random.randint(100, size=(100)) sizes = 10 *
np.random.randint(100, size=(100))


plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='nipy_spectral')


plt.colorbar()
plt.show()
```

OUTPUT:



PRACTICAL 30

Q: Students can make assumptions for this question.

• Create a DataFrame using a Dictionary/any csv file/ any json file etc.import pandas as pd

```
# Read the CSV file into a DataFrame
df = pd.read_csv('data_new2.csv')
print(df.head(2))
```

	Duration	Pulse	Maxpulse	Calories	date
0	60	110.0	NaN	NaN	23-12-1980
1	60	117.0	NaN	NaN	24-12-1980

• Display all the column labels of your dataset .

```
print("Column labels:")
print(df.columns)
```

```
Column labels:
Index(['Duration', 'Pulse', 'Maxpulse', 'Calories', 'date'], dtype='object')
```

• Sort the DataFrame based on a particular numerical column indescending order .

```
sorted_df = df.sort_values(by='Maxpulse', ascending=False)
print("\nSorted DataFrame:")
print(sorted_df.head(10))
```

```
Sorted DataFrame:
     Duration  Pulse  Maxpulse  Calories        date
109       210  137.0     184.0    1860.4  11-04-1981
80         30  159.0     182.0     319.2  13-03-1981
54         30  136.0     175.0     238.0  15-02-1981
3          45  109.0     175.0       NaN  26-12-1980
58         20  153.0     172.0     226.4  19-02-1981
94         20  150.0     171.0     127.4  27-03-1981
144        60  136.0     170.0     470.2  16-05-1981
85         30  151.0     170.0     300.0  18-03-1981
122        60  119.0     169.0     336.7  24-04-1981
81         45  149.0     169.0     344.0  14-03-1981
```

• Display the mean of any numerical column A in your DataFrame .

```
column_mean = df['Pulse'].mean()
print("\n Mean of Pulse :", column_mean)
```

```
 Mean of Pulse : 107.52380952380952
```

• Fill the missing values in column A with this mean value assuming nooutliers are present in that column.

```
df['Pulse'].fillna(column_mean, inplace=True)
print("\nDataFrame after filling missing values:")
print(df.to_string())
```

```
     71         60  109.0  04-03-1981
     72         90  100.0  05-03-1981
     73        150    NaN  06-03-1981
     74         45  114.0  07-03-1981
     75         90   98.0  08-03-1981
     76         45  105.0  09-03-1981
     77         45  110.0  10-03-1981
     78        120  100.0  11-03-1981
     79        270  100.0  12-03-1981
     80         30  159.0  13-03-1981
```

```
70          150   97.00000  03-03-1981
71           60  109.00000  04-03-1981
72           90  100.00000  05-03-1981
73          150  107.52381  06-03-1981
74           45  114.00000  07-03-1981
75           90   98.00000  08-03-1981
76           45  105.00000  09-03-1981
77           45  110.00000  10-03-1981
78          120  100.00000  11-03-1981
79          270  100.00000  12-03-1981
80           30  159.00000  13-03-1981
```

- Remove the column having more than 4 null values.

Code:

```
df.dropna(thresh=len(df) - 4, axis=1, inplace=True)

print("\nDataFrame after removing column with more than 4 null values:")

print(df.head(5))
```

OUTPUT:

```
DataFrame after removing column with more than 4 null values:
   Duration  Pulse        date
0        60  110.0  23-12-1980
1        60  117.0  24-12-1980
2        60  103.0  25-12-1980
3        45  109.0  26-12-1980
4        45  117.0  27-12-1980
```

PRACTICAL 31

Q: Assume that you have a DataFrame as DataFrame([["A", 1], ["B", 2], ["C",3], ["D", 4]],columns = ["Col_A", "Col_B"]).

```
import pandas as pd
import numpy as np
df = pd.DataFrame([["A", 1], ["B", 2], ["C", 3], ["D", 4]], columns=["Col_A", "Col_B"])
print(df)
```

```
  Col_A  Col_B
0     A      1
1     B      2
2     C      3
3     D      4
```

• Insert a column at a specific location in a DataFrame.

```
df.insert(loc=1, column='Col_K', value=[10,20,30,40])
print("DataFrame after inserting 'Col_K' column:")
print(df)
```

```
DataFrame after inserting 'Col_K' column:
  Col_A  Col_K  Col_B
0     A     10      1
1     B     20      2
2     C     30      3
3     D     40      4
```

• Select columns based on the column's Data Type ..

```
numeric_columns = df.select_dtypes(include='number')
print("\nNumeric columns:")
print(numeric_columns)
```

```
Numeric columns:
   Col_h  Col_K  Col_B
0     10     10      1
1     20     20      2
2     30     30      3
3     40     40      4
```

• Count the number of Non-NaN cells for each column .

non_nan_counts = df.count()

print("\nNon-NaN counts for each column:")print(non_nan_counts)

```
Non-NaN counts for each column:
Col_A    4
Col_h    4
Col_K    4
Col_B    4
dtype: int64
```

• Split DataFrame into equal parts .

num_parts = 2

df_parts = np.array_split(df, num_parts)

print("\nDataFrame split into", num_parts, "parts:","\n")

for part in df_parts:

    print(part)

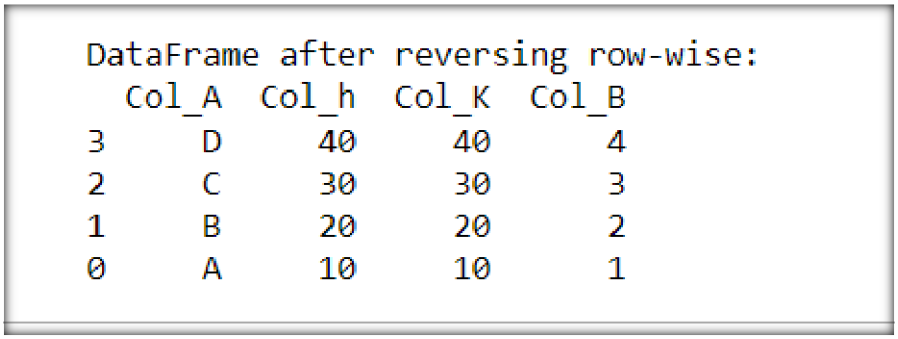
```
DataFrame split into 2 parts:

  Col_A  Col_h  Col_K  Col_B
0     A     10     10      1
1     B     20     20      2
  Col_A  Col_h  Col_K  Col_B
2     C     30     30      3
3     D     40     40      4
```

• Reverse DataFrame row-wise or column-wise

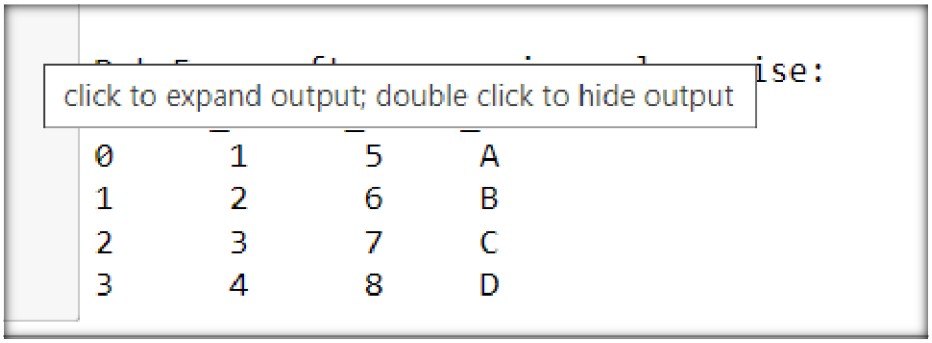
Code:

```
#rowwise
df_reverse_row = df[::-1]
print("\nDataFrame after reversing row-wise:")
print(df_reverse_row)
#columnwise
df_reverse_column = df.iloc[:, ::-1]
print("\nDataFrame after reversing column-wise:")
print(df_reverse_column)
```

OUTPUT:



```
DataFrame after reversing row-wise:
  Col_A  Col_h  Col_K  Col_B
3     D     40     40      4
2     C     30     30      3
1     B     20     20      2
0     A     10     10      1
```



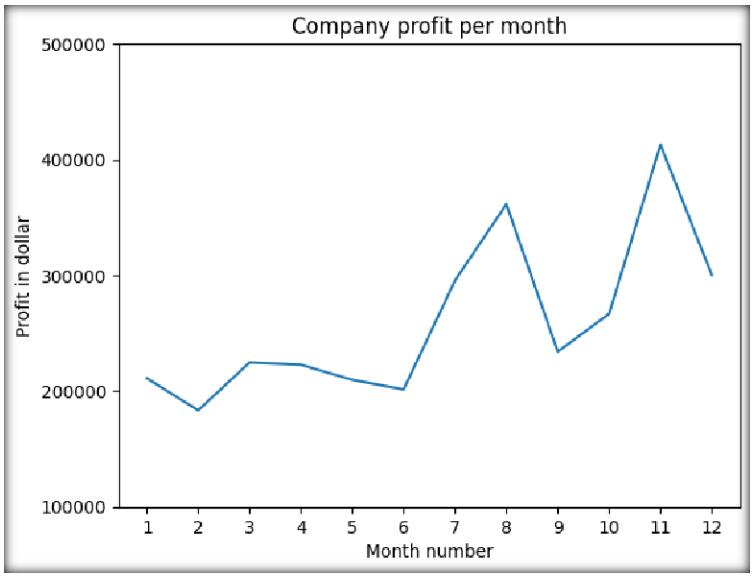
```
click to expand output; double click to hide output
0      1      5      A
1      2      6      B
2      3      7      C
3      4      8      D
```

PRACTICAL 32

Q: Read company_sales_data.csv(657 B) on Kaggle and perform thefollowing:

• Read Total profit of all months and show it using a line plot.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('company_sales_data.csv')profitList = df
['total_profit'].tolist()
monthList  = df ['month_number'].tolist()
plt.plot(monthList, profitList, label = 'Month-wise Profit data of last year')plt.xlabel('Month number')
plt.ylabel('Profit in dollar') plt.xticks(monthList)
plt.title('Company profit per month')
plt.yticks([100000, 200000, 300000, 400000, 500000])
plt.show()
```



• Get Total profit of all months and show line plot with the followingStyle properties

➢ Generated line plot must include following Style properties: -

➢ Line Style dotted and Line-color should be red

➢ Show legend at the lower right location.

➢ X label name = Month Number

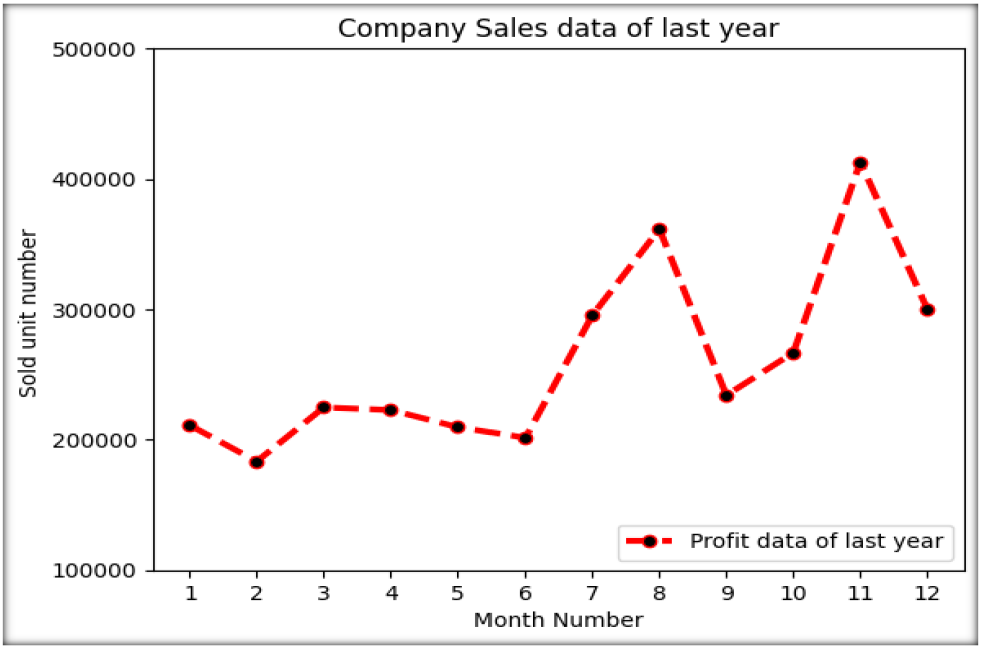
➢ Y label name = Sold units number

➢ Add a circle marker.

➢ Line marker color as read

➢ Line width should be 3.

```
df = pd.read_csv('company_sales_data.csv')

profitList = df ['total_profit'].tolist()
monthList  = df ['month_number'].tolist()


plt.plot(monthList, profitList, label = 'Profit data of last year',
      color='r', marker='o', markerfacecolor='k',
      linestyle='--', linewidth=3)


plt.xlabel('Month Number')
plt.ylabel('Sold unit number ')
plt.legend(loc='lower right')
plt.title('Company Sales data of last year')
plt.xticks(monthList)
plt.yticks([100000, 200000, 300000, 400000, 500000])
plt.show()
```



• Read all product sales data and show it using a multiline plot.# 3

monthList = df ['month_number'].tolist() faceCremSalesData = df ['facecream'].tolist()

faceWashSalesData = df ['facewash'].tolist() toothPasteSalesData = df

['toothpaste'].tolist() bathingsoapSalesData = df ['bathingsoap'].tolist()

shampooSalesData = df ['shampoo'].tolist() moisturizerSalesData = df

['moisturizer'].tolist()

plt.plot(monthList, faceCremSalesData, label = 'Face cream Sales Data',marker='o', linewidth=3)

plt.plot(monthList, faceWashSalesData, label = 'Face Wash Sales Data',marker='o', linewidth=3)

plt.plot(monthList, toothPasteSalesData, label = 'ToothPaste Sales Data', marker='o',linewidth=3)

plt.plot(monthList, bathingsoapSalesData, label = 'ToothPaste Sales Data',marker='o', linewidth=3)

plt.plot(monthList, shampooSalesData, label = 'ToothPaste Sales Data', marker='o',linewidth=3)

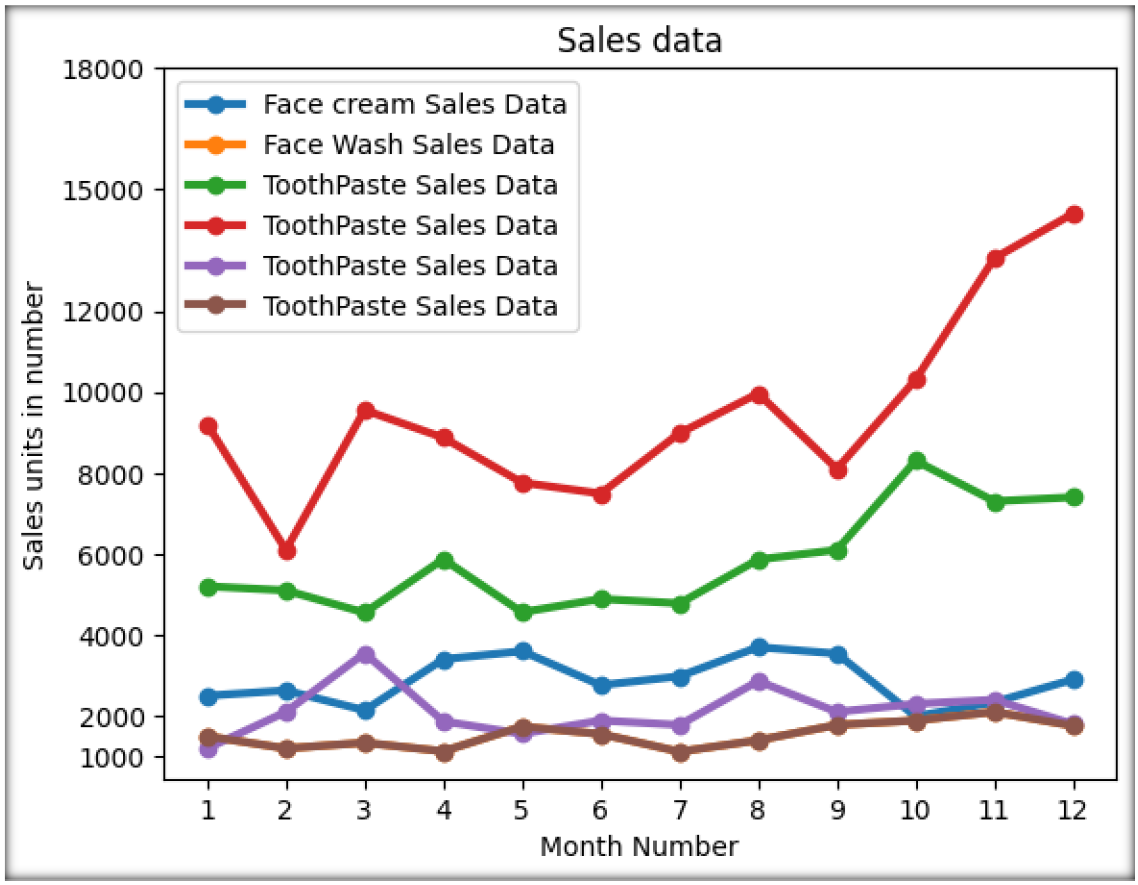
plt.plot(monthList, moisturizerSalesData, label = 'ToothPaste Sales Data',marker='o', linewidth=3)

plt.xlabel('Month Number') plt.ylabel('Sales units in

number')plt.legend(loc='upper left') plt.xticks(monthList)

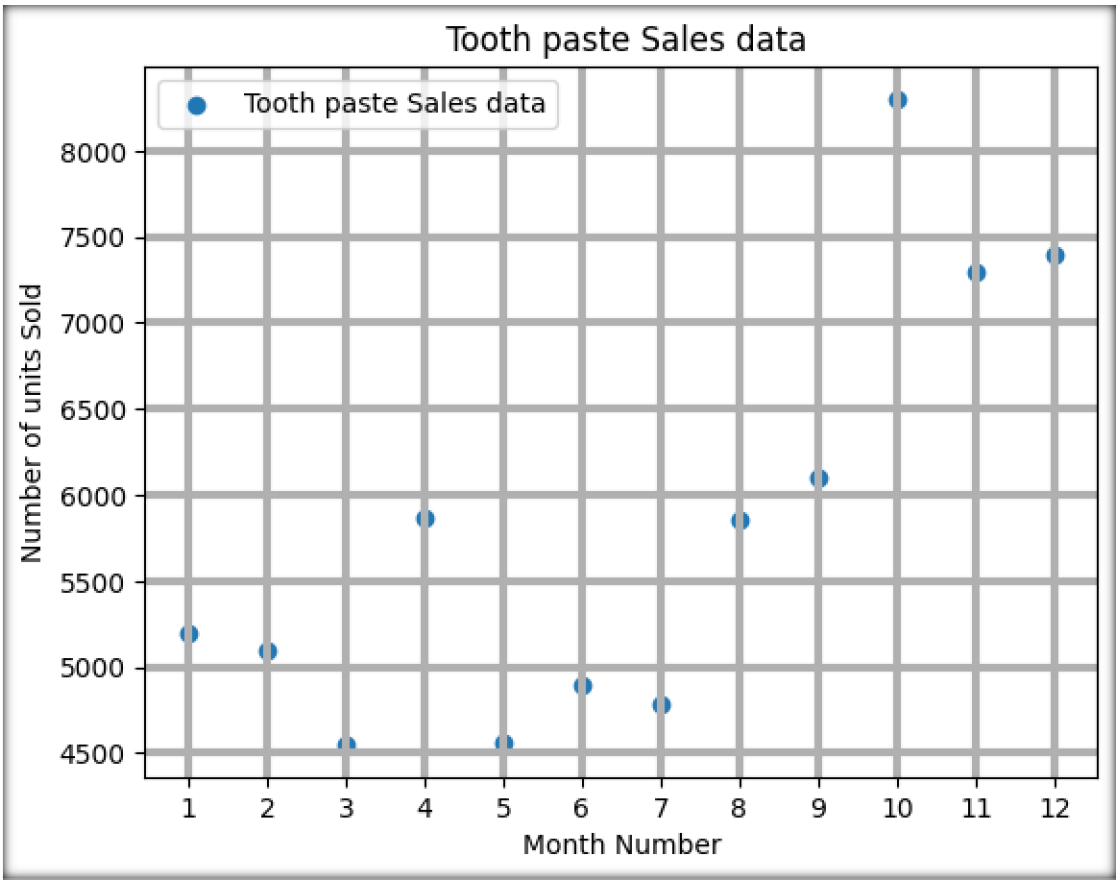
plt.yticks([1000, 2000, 4000, 6000, 8000, 10000, 12000, 15000, 18000])

plt.title('Sales data')plt.show()



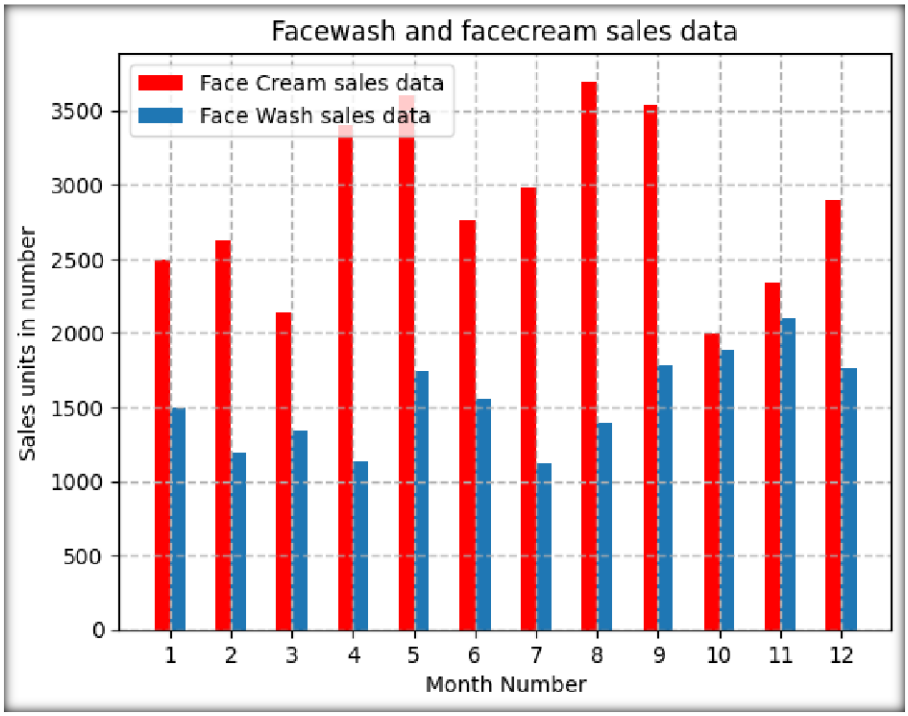
• Read toothpaste sales data of each month and show it using a scatter plot

```
monthList = df ['month_number'].tolist() toothPasteSalesData = df ['toothpaste'].tolist()

plt.scatter(monthList, toothPasteSalesData, label = 'Tooth paste Sales data')plt.xlabel('Month Number')

plt.ylabel('Number of units Sold')plt.legend(loc='upper

left') plt.title(' Tooth paste Sales data')

plt.xticks(monthList)

plt.grid(True, linewidth= 3, linestyle="-")plt.show()
```



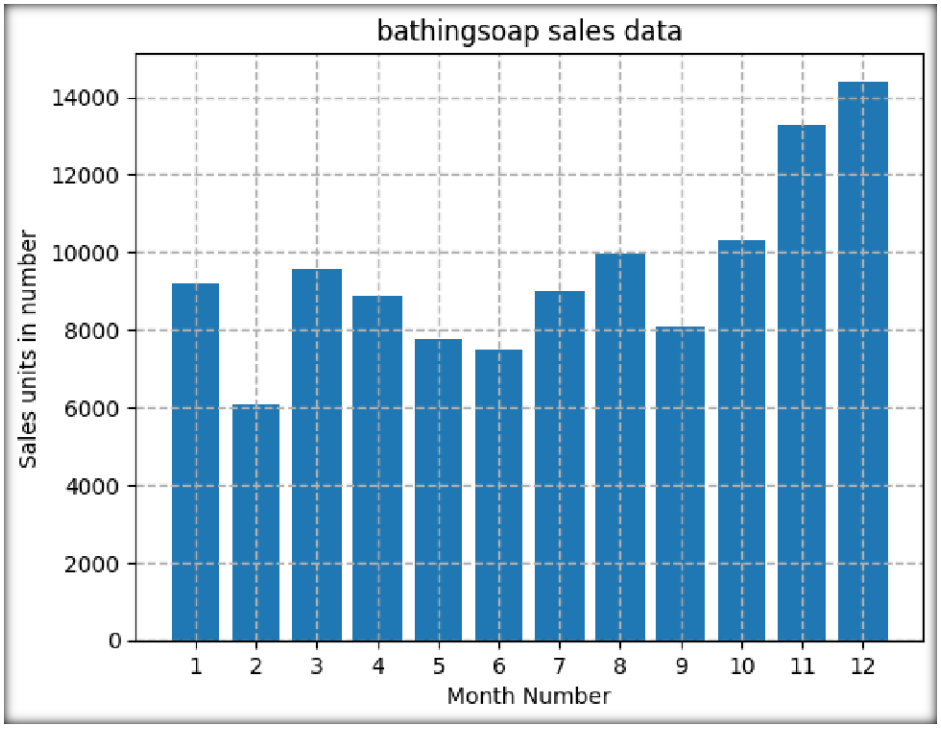
• Read face cream and facewash product sales data and show it using thebar chart.

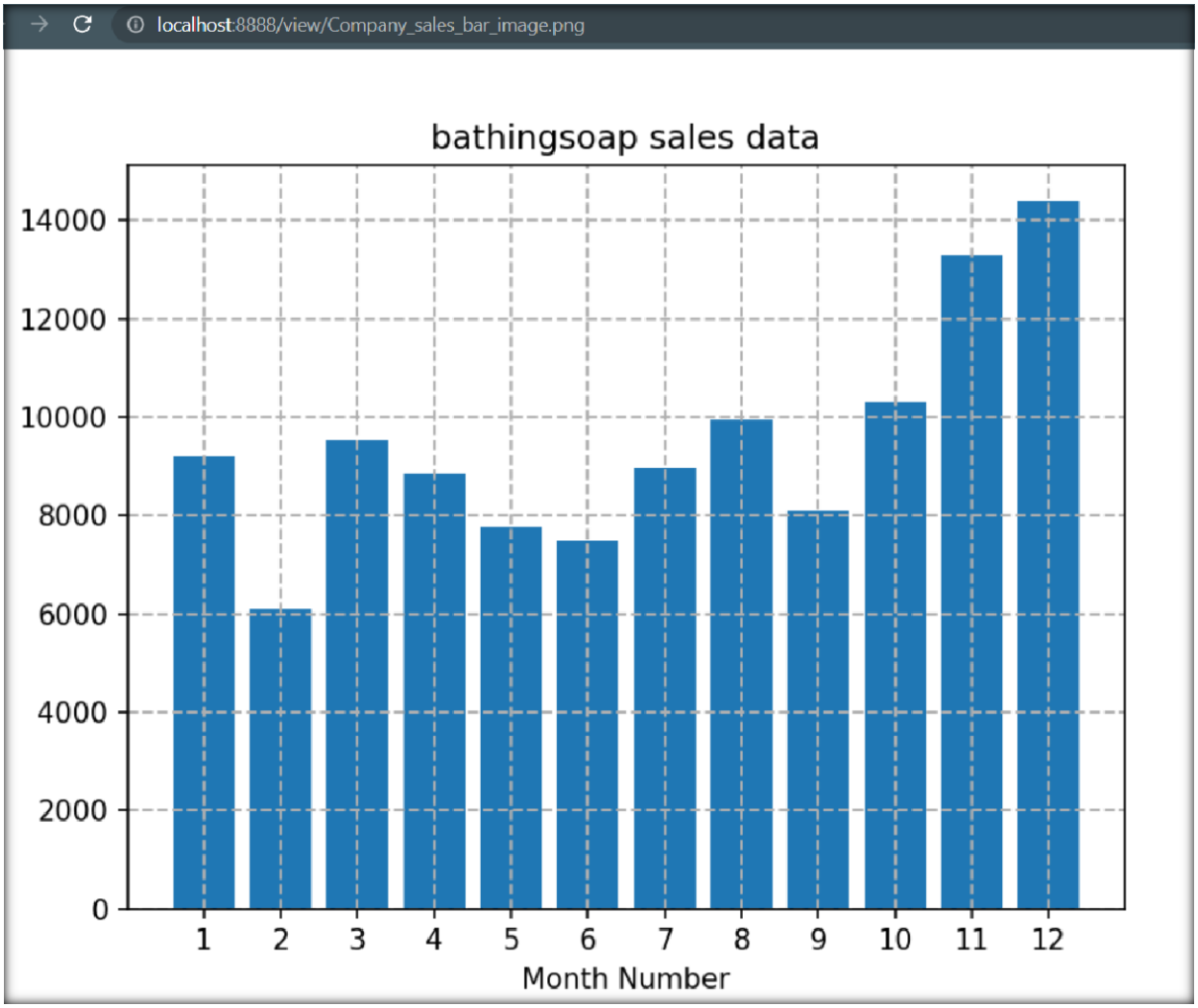
```
monthList = df ['month_number'].tolist() faceCremSalesData = df ['facecream'].tolist()

faceWashSalesData  = df ['facewash'].tolist()

plt.bar([a-0.25 for a in monthList], faceCremSalesData, width= 0.25, label = 'FaceCream sales data',
align='edge',color='red')

plt.bar([a+0.25 for a in monthList], faceWashSalesData, width= -0.25, label = 'FaceWash sales data', align='edge')

plt.xlabel('Month Number') plt.ylabel('Sales units in

number')plt.legend(loc='upper left') plt.title(' Sales data')

plt.xticks(monthList)

plt.grid(True, linewidth= 1, linestyle="--") plt.title('Facewash and

facecream sales data')plt.show()
```



• Read sales data of bathing soap of all months and show it using a barchart. Save this plot to your hard disk.

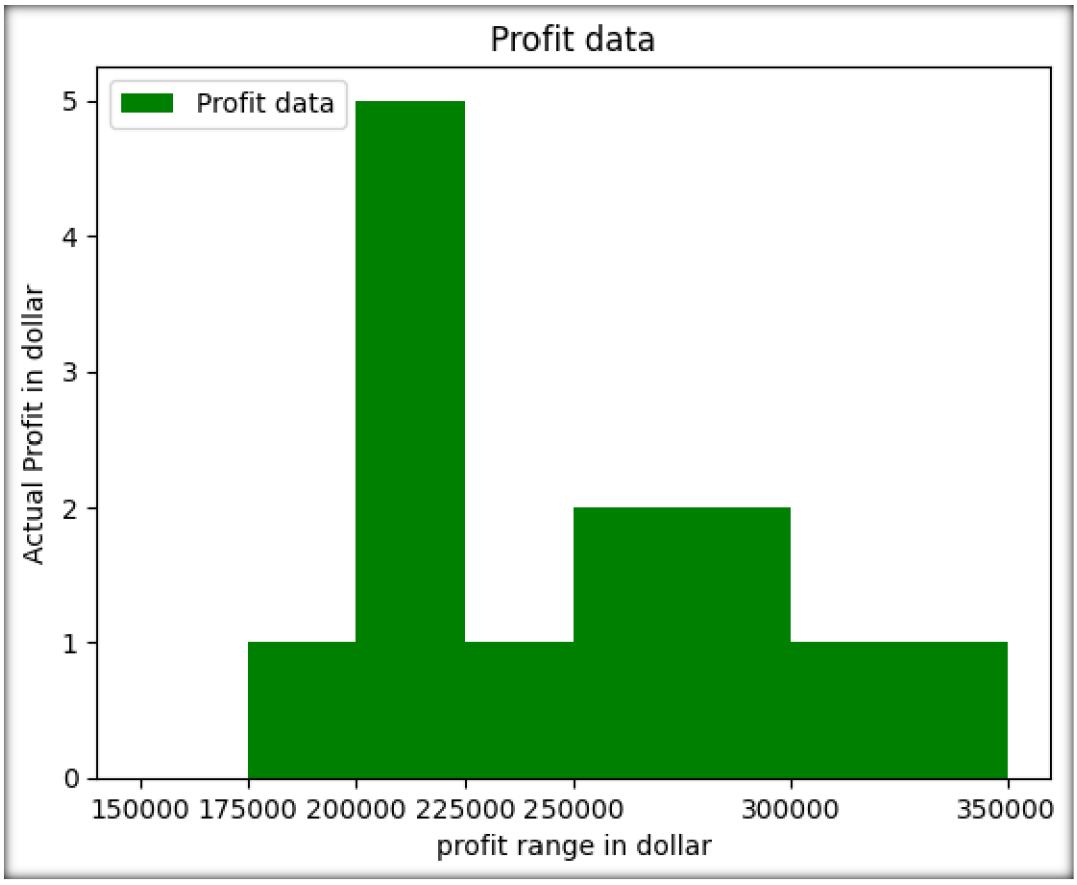
```
monthList = df ['month_number'].tolist()

bathingsoapSalesData= df ['bathingsoap'].tolist()

plt.bar(monthList, bathingsoapSalesData)

plt.xlabel('Month Number')
plt.ylabel('Sales units in number')
plt.title(' Sales data')
plt.xticks(monthList)
plt.grid(True, linewidth= 1, linestyle="--")
plt.title('bathingsoap sales data')
plt.savefig(r'D:\DATA_SCIENCE_USING_PYTHON\Company_sales_bar_image.png ', dpi=150)
plt.show()
```


bathingsoap sales data
Sales units in number
Month Number
0
2000
4000
6000
8000
10000
12000
14000
1
2
3
4
5
6
7
8
9
10
11
12


localhost:8888/view/Company_sales_bar_image.png
bathingsoap sales data
Month Number
0
2000
4000
6000
8000
10000
12000
14000
1
2
3
4
5
6
7
8
9
10
11
12

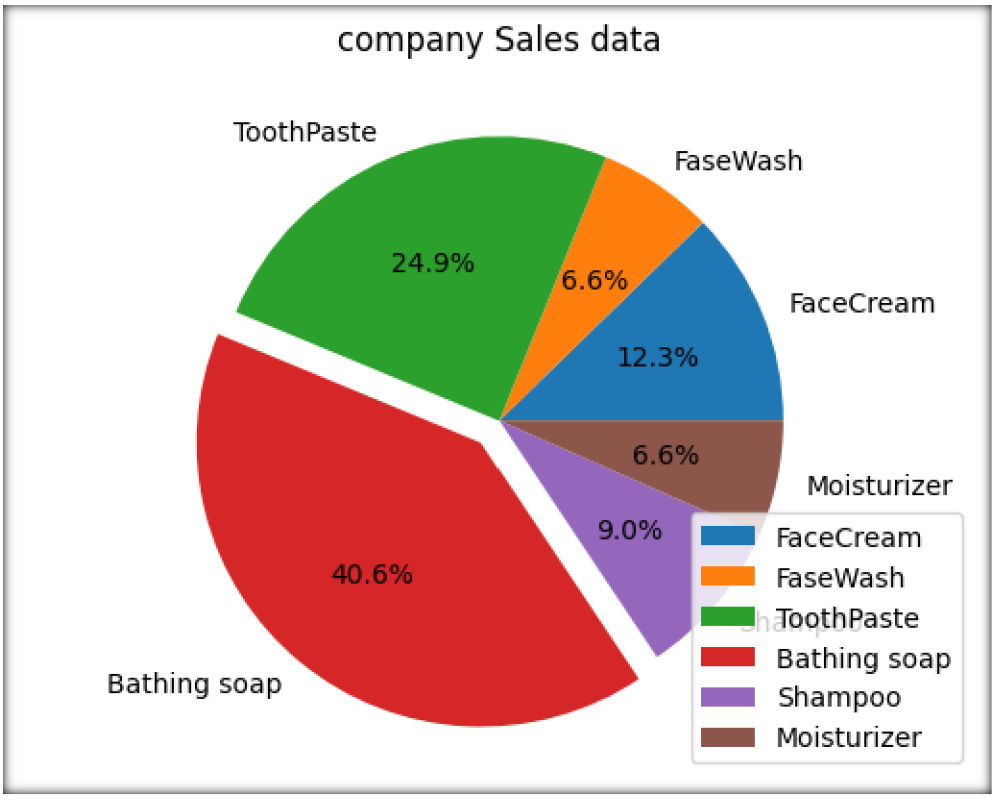
Read the total profit of each month and show it using the histogram tosee most common profit ranges

```
profitList = df ['total_profit'].tolist()

labels = ['low', 'average', 'Good', 'Best']

profit_range = [150000, 175000, 200000, 225000, 250000, 300000, 350000]

plt.hist(profitList, profit_range, label = 'Profit data',color='green')

plt.xlabel('profit range in dollar')

plt.ylabel('Actual Profit in dollar')

plt.legend(loc='upper left')

plt.xticks(profit_range)

plt.title('Profit data')

plt.show()
```



• Calculate total sale data for last year for each product and show it usinga Pie chart.

```
monthList  = df ['month_number'].tolist()
labels = ['FaceCream', 'FaseWash', 'ToothPaste', 'Bathing soap', 'Shampoo','Moisturizer']
salesData = [df ['facecream'].sum(), df ['facewash'].sum(), df ['toothpaste'].sum(),
df ['bathingsoap'].sum(), df ['shampoo'].sum(), df ['moisturizer'].sum()]gap=[0,0,0,0.1,0,0]
plt.axis("equal")
plt.pie(salesData, labels=labels, autopct='%1.1f%%',explode=gap)
plt.legend(loc=4)
plt.title('company Sales data')
plt.show()
```



• Read Bathing soap facewash of all months and display it using theSubplot.

Code:

```
monthList= df ['month_number'].tolist()
bathingsoap = df ['bathingsoap'].tolist()
faceWashSalesData = df ['facewash'].tolist()

f, axarr = plt.subplots(2, sharex=True)
axarr[0].plot(monthList, bathingsoap, label = 'Bathingsoap Sales Data', color='c',marker='D', linewidth=3)
axarr[0].set_title('Sales data of  a Bathingsoap')
axarr[1].plot(monthList, faceWashSalesData, label = 'Face Wash Sales Data',color='r', marker='D',
linewidth=3)
axarr[1].set_title('Sales data of  a facewash')

plt.xticks(monthList)
plt.xlabel('Month Number')
plt.ylabel('Sales units in number')
plt.show()
```

OUTPUT: