

SESSION 2023-24

Data Science

(Data Science using Python)

Lab File

**COURSE:- BCA**

**ROLL NO :- 41221139**

**SUBMITTED BY :- SUBMITTED TO:-**

**Sachin Rajbhar 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() ) * Convert NumPy array to list and check. |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 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

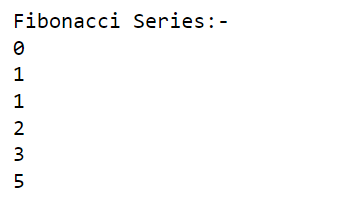
print(c)

a=b

b=c

fib(6)

**OUTPUT:**

****

**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)

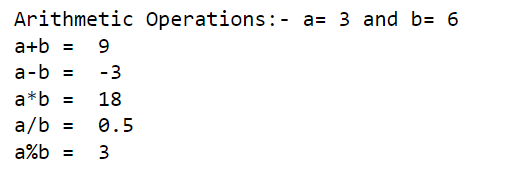
print("a\*b = ",a\*b)

print("a/b = ",a/b)

print("a%b = ",a%b)

arith(3,6)

**OUTPUT:**

****

**PRACTICAL 3**

Q: python code to find area of triangle

**CODE:**

def triArea(b,h):

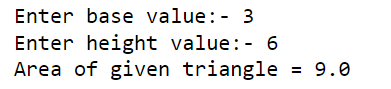
return (1/2\*b\*h)

x = int(input("Enter base value:- "))

y = int(input("Enter height value:- "))

print("Area of given triangle =",triArea(x,y))

**OUTPUT:**

****

**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)

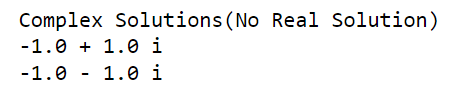
print("Complex Solutions(No Real Solution):-")

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:**

****

**PRACTICAL 5**

Q: python code for Swapping of 2 variables

**CODE:**

def swap2(a,b):

print("Before Swapping a=",a,"b=",b)

temp=a

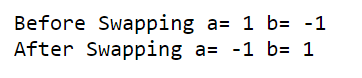
a=b

b=temp

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

swap2(1,-1)

**OUTPUT:**

****

**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:**

****

**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

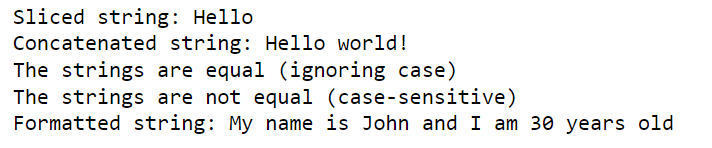
name = "John"

age = 30

formatted\_str = "My name is {} and I am {} years old".format(name, age)

print("Formatted string:", formatted\_str)

**OUTPUT:**

****

**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()

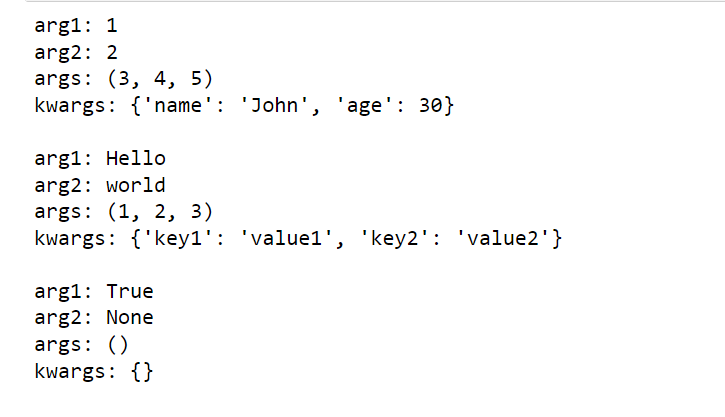
# Call the function with various arguments

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:**

****

**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])

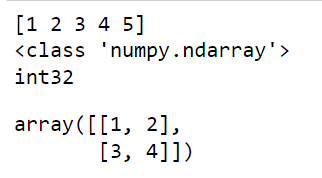
print(arr)

print(type(arr))

print(arr.dtype)

np.array([[1,2],[3,4]])

**OUTPUT:**

****

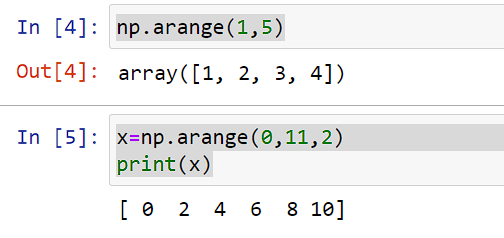
**CODE:**

np.arange(1,5)

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

print(x)

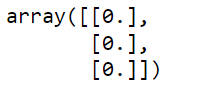
**OUTPUT:**

****

**CODE:**

np.zeros((3,2))

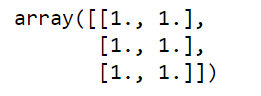
**OUTPUT:**

****

**CODE:**

np.ones((3,5))

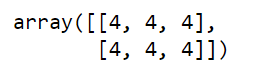
**OUTPUT:**

****

**CODE:**

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

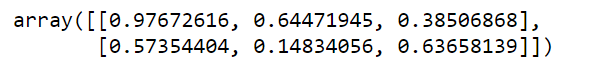
**OUTPUT:**

****

**CODE:**

np.random.rand(2,3)

**OUTPUT:**

****

**CODE:**

arr = np.arange(1, 10)

print(arr, '\n')

arr = arr.reshape(3, 3)

print(arr, '\n')

print("minimum element is:", arr.min())

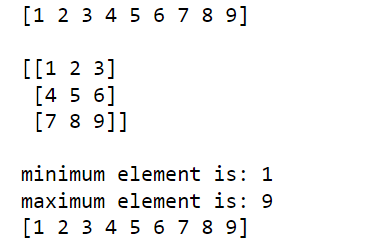
print("maximum element is:", arr.max())

# Reshape back to the original size

arr = arr.reshape(9)

print(arr)

**OUTPUT:**

****

**CODE:**

np.linspace(0,10,5)

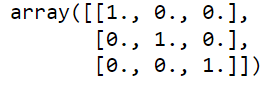
**OUTPUT:**

****

**CODE:**

np.eye(3)

**OUTPUT:**

****

**CODE:**

arr = np.loadtxt("C:\\Users\\athar123\\Desktop\\test\_bca.csv", delimiter=",", dtype=str)

display(arr)

display(arr[2])

**OUTPUT:**

****

**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:**

****

**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**

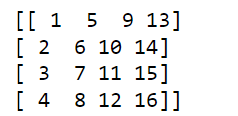
**OUTPUT:**

****

**CODE:**

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

**OUTPUT:**

****

**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)**

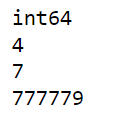
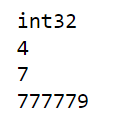
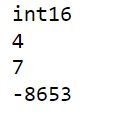
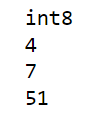
**print("\n",ar)**

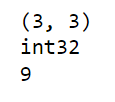
**print(np.where(ar>3))**

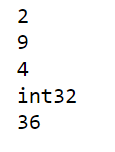
**print("\n")**

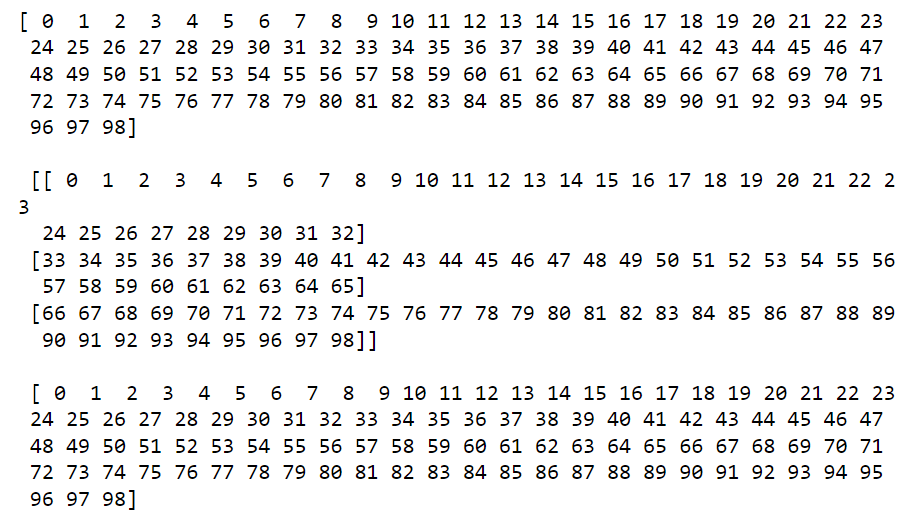
**print(np.nonzero(arr1),np.count\_nonzero(arr1),sep="\n")**

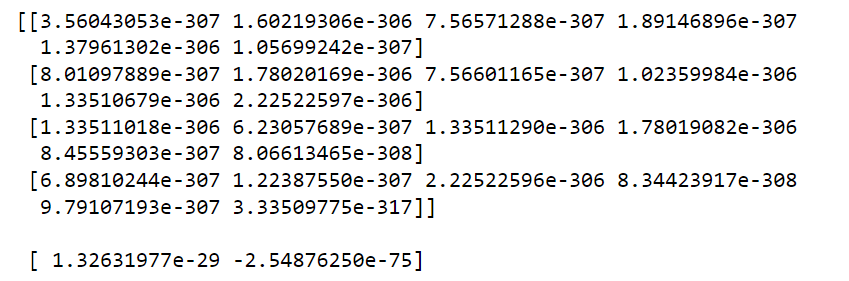
**OUTPUT:**

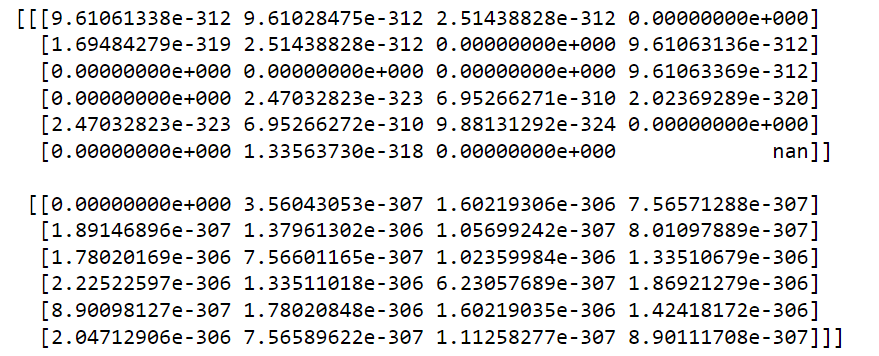
****

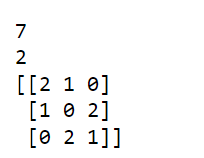
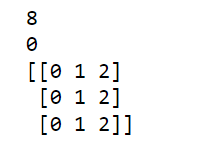
****

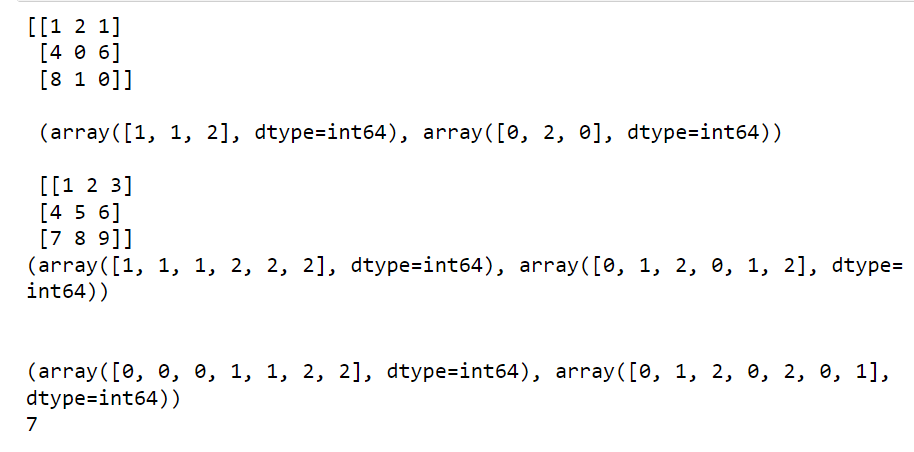
****

****

****

****

****

****

**CODE:**

**import numpy as np**

**# create a NumPy array**

**x = np.array([1, 2, 3, 4, 5])**

**print(x)**

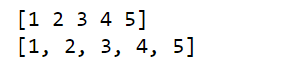
**# convert the array to a list**

**lst = x.tolist()**

**# print the list**

**print(lst)**

**OUTPUT:**

****

**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.

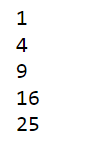
• Demonstrate usage of checkpoint in Jupyter.

**CODE:**

**for i in range(1, 6):**

**print(i \*\* 2)**

**OUTPUT:**

****

**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 |**

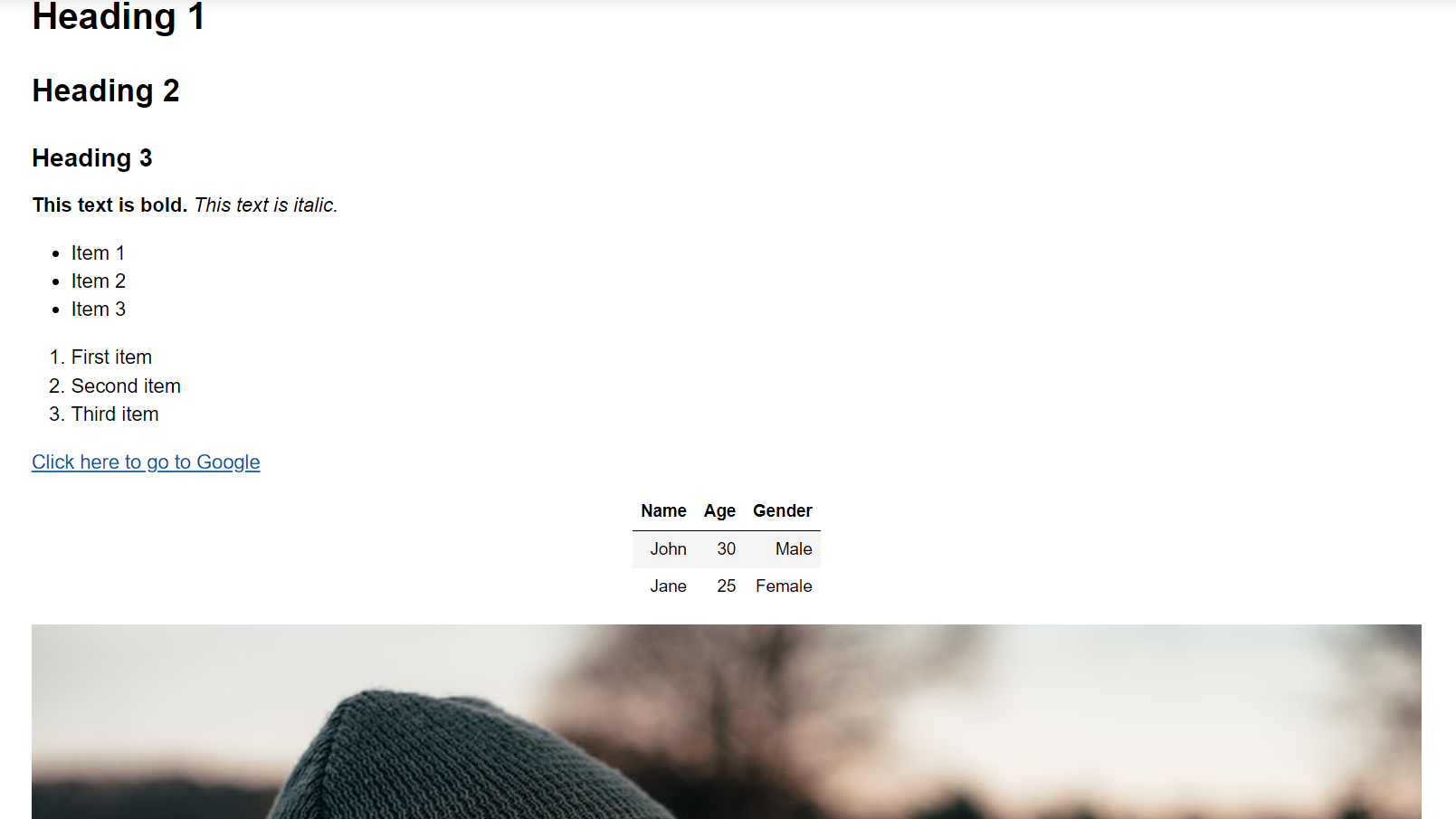
**|------|-----|--------|**

**| John | 30 | Male |**

**| Jane | 25 | Female |**

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

**OUTPUT:**

****

**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:**

****

**CODE:**

**import pandas as pd**

**# Creating a Series using a list**

**fruits = pd.Series(['Apple', 'Banana', 'Cherry', 'Dates'])**

**# Printing the Series**

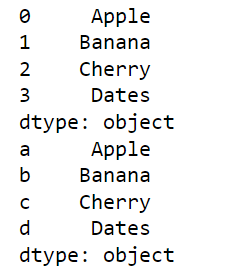
**print(fruits)**

**# Adding our own labels to the Series**

**fruits = pd.Series(['Apple', 'Banana', 'Cherry', 'Dates'], index=['a', 'b', 'c', 'd'])**

**print(fruits)**

**OUTPUT:**

****

**CODE:**

**import pandas as pd**

**# Creating a Series using a list**

**fruits = pd.Series(['Apple', 'Banana', 'Cherry', 'Dates'])**

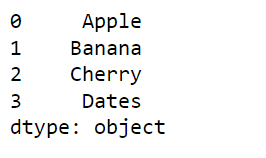
**# Sorting the Series**

**sorted\_fruits = fruits.sort\_values()**

**# Printing the sorted Series**

**print(sorted\_fruits)**

**OUTPUT:**

****

**CODE:**

**import pandas as pd**

**# Creating a DataFrame using a list**

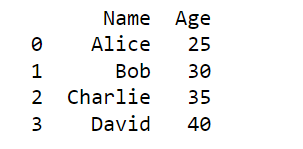
**data = [['Alice', 25], ['Bob', 30], ['Charlie', 35], ['David', 40]]**

**df = pd.DataFrame(data, columns=['Name', 'Age'])**

**# Printing the DataFrame**

**print(df)**

**OUTPUT:**

****

**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**

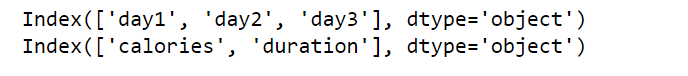
**df.index = ['day1', 'day2', 'day3']**

**# Printing only index and columns in this DataFrame**

**print(df.index)**

**print(df.columns)**

**OUTPUT:**

****

**CODE:**

**import pandas as pd**

**# Creating a dictionary**

**data = {'apple': 2, 'banana': 3, 'cherry': 4}**

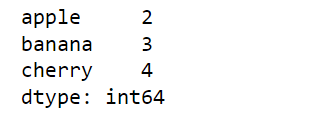
**# Converting the dictionary to a Series**

**fruits = pd.Series(data)**

**# Printing the Series**

**print(fruits)**

**OUTPUT:**

****

**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']]**

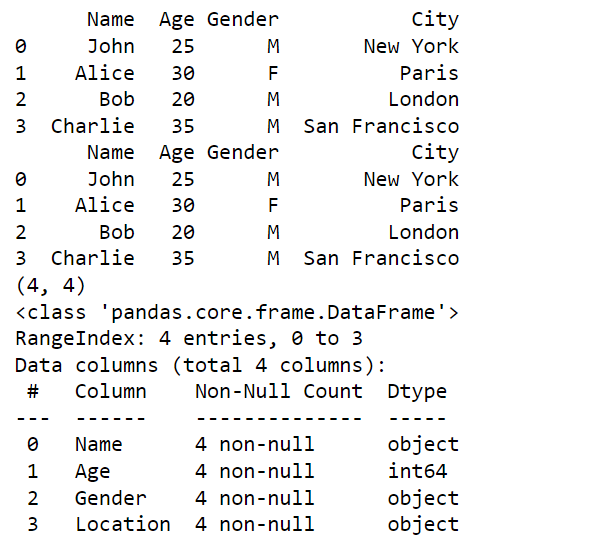
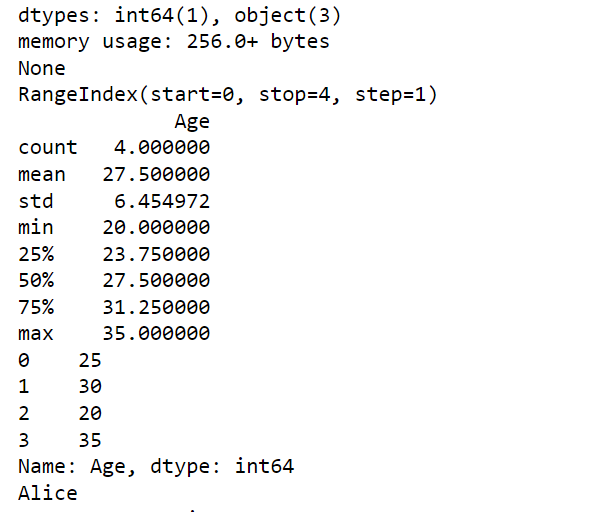
**print(slice\_df)**

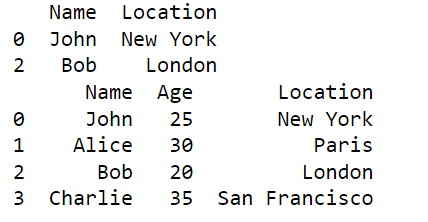
**# drop the 'Gender' column from the DataFrame**

**df.drop(columns=['Gender'], inplace=True)**

**print(df)**

**OUTPUT:**

**** ****



**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)**

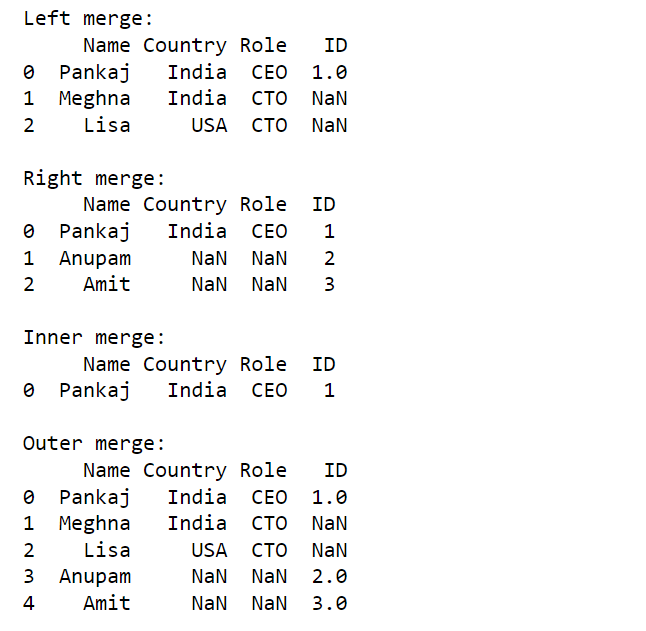
**# outer merge**

**df\_outer = pd.merge(df1, df2, on='Name', how='outer')**

**print("\nOuter merge:")**

**print(df\_outer)**

**OUTPUT:**

****

**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']}**

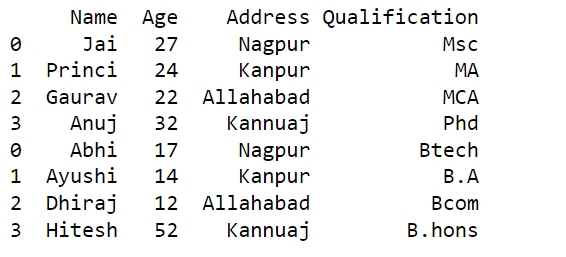
**df1 = pd.DataFrame(d1)**

**df2 = pd.DataFrame(d2)**

**df\_concatenated = pd.concat([df1, df2])**

**print(df\_concatenated)**

**OUTPUT:**

****

**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:**

****

**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)**

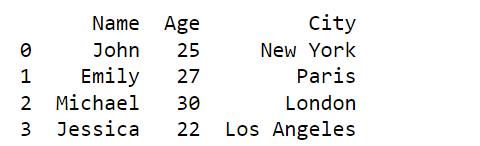
**# print the DataFrame**

**print(df)**

**# save the DataFrame to a CSV file**

**df.to\_csv('df\_to\_file.csv', index=False)**

**OUTPUT:**

****

**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()**

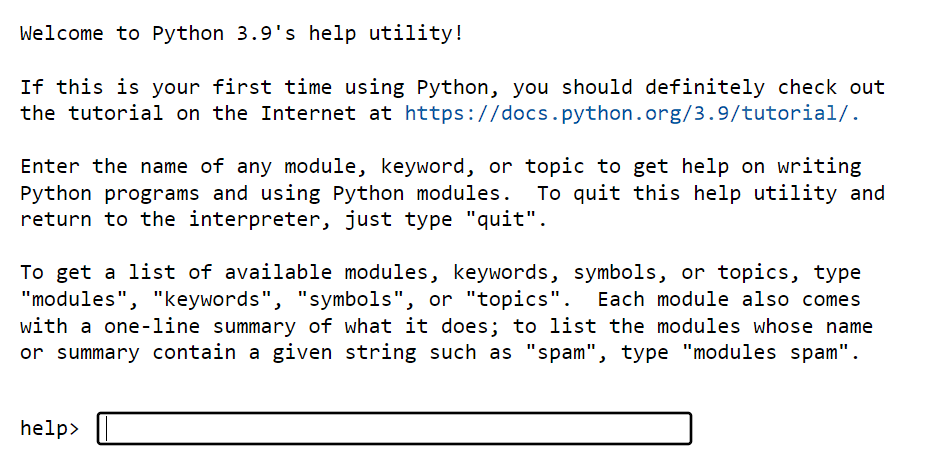
**# display help information for the list type**

**help(list)**

**# exit help mode**

**quit()**

**OUTPUT:**

****

**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**

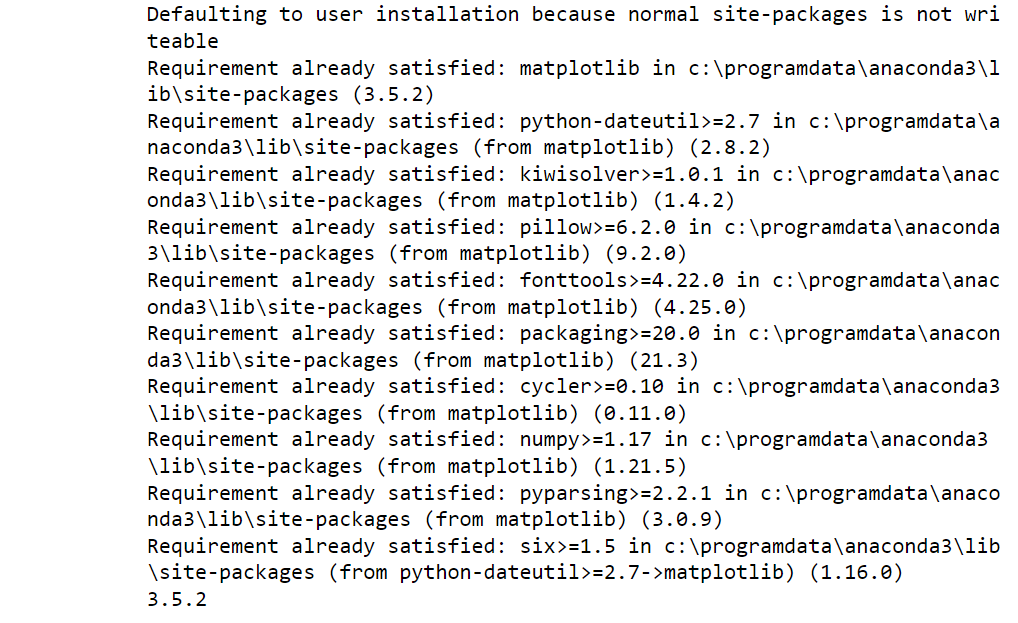
Q: Working with Matplotlib

**CODE:**

**!pip install matplotlib**

**import matplotlib**

**print(matplotlib.\_\_version\_\_)**

**OUTPUT:**

**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")**

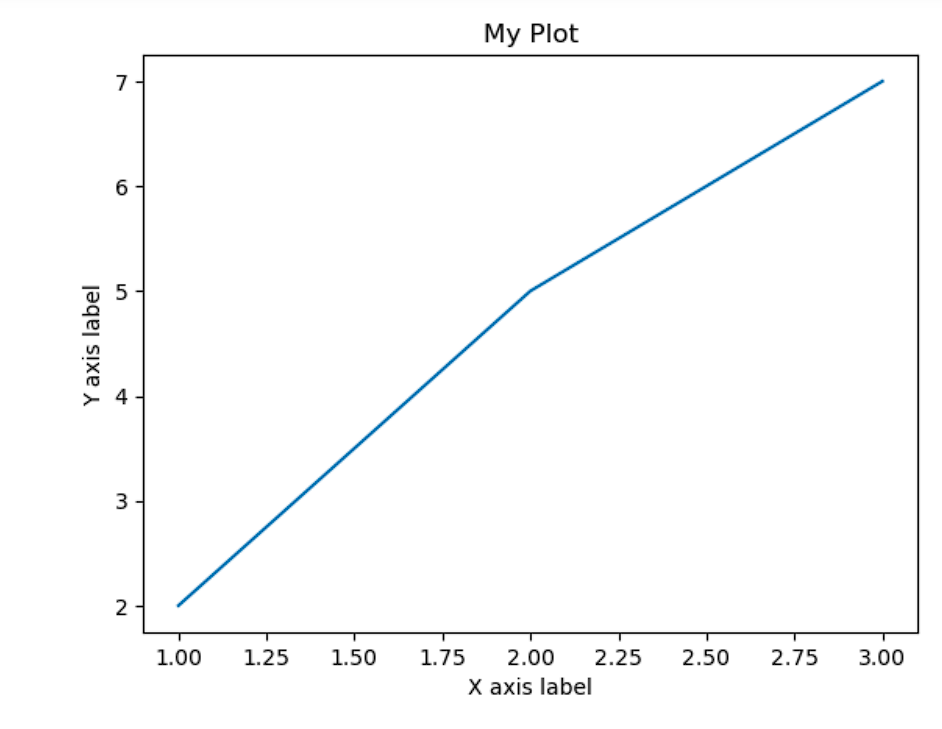
**plt.xlabel("X axis label")**

**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:**

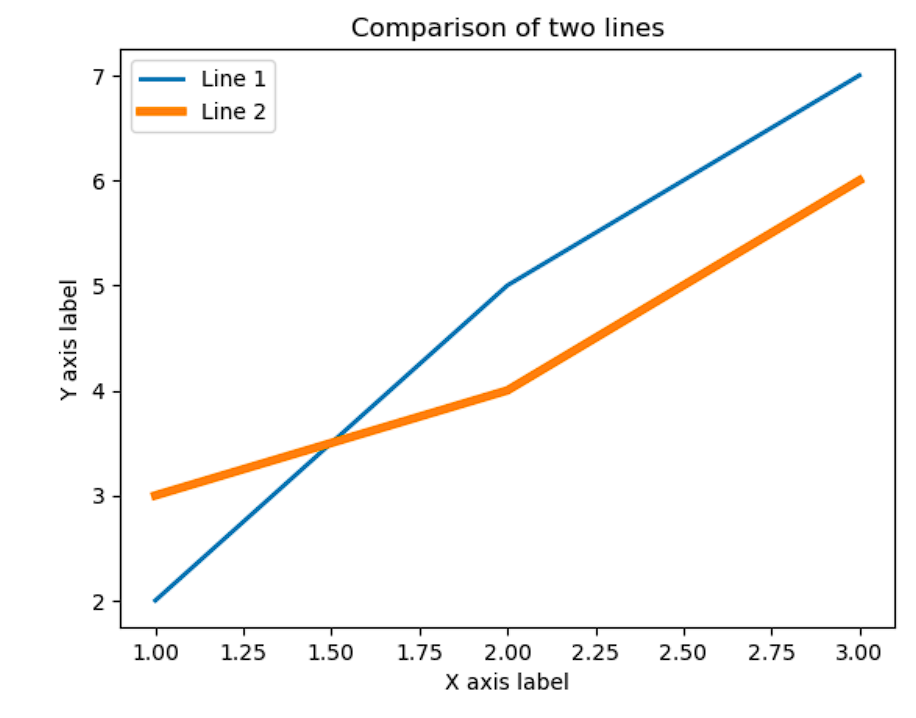
**CODE:**

**# Define the data for the pie chart**

**labels = ['Apples', 'Oranges', 'Bananas']**

**sizes = [30, 40, 20]**

**# Plot the pie chart**

**plt.pie(sizes, labels=labels)**

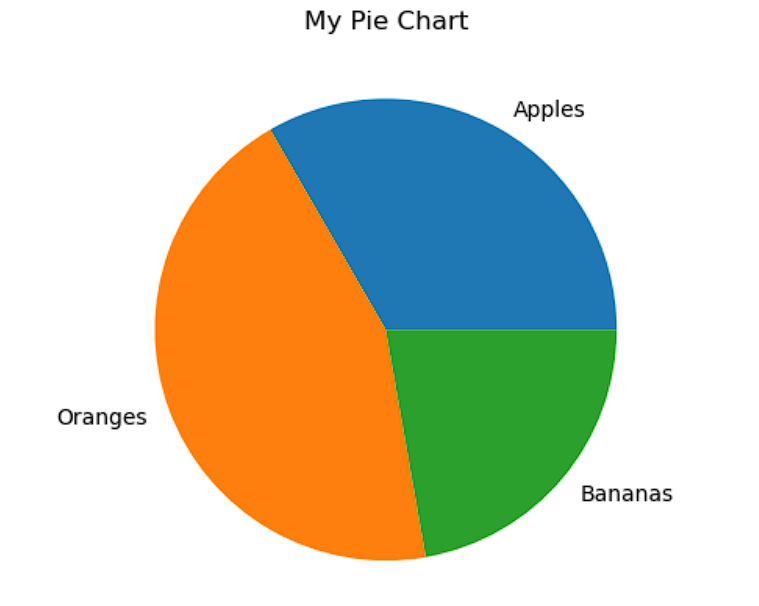
**# Add a title**

**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

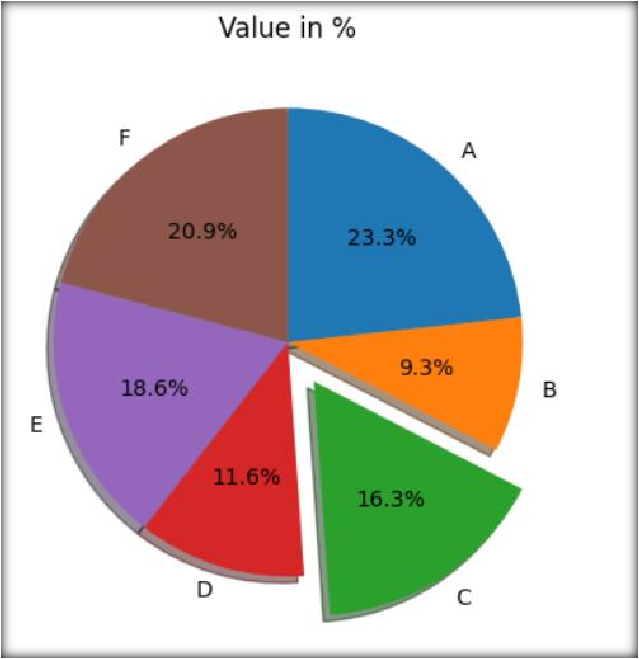
values =[23.3,9.3,16.3,11.6,18.6,20.9] label=['A','B','C','D','E','F']

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

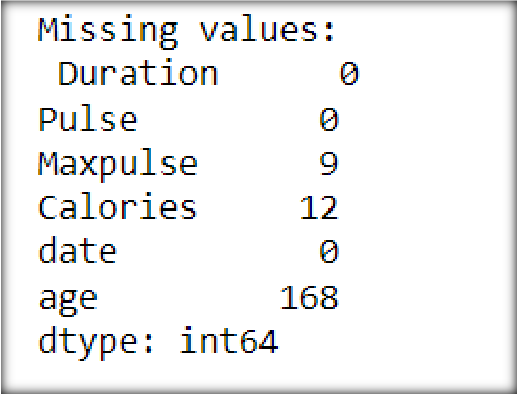
import pandas as pd import seaborn as sns

import matplotlib.pyplot as plt

# load the data csv file

df = pd.read\_csv('data\_new2.csv') #

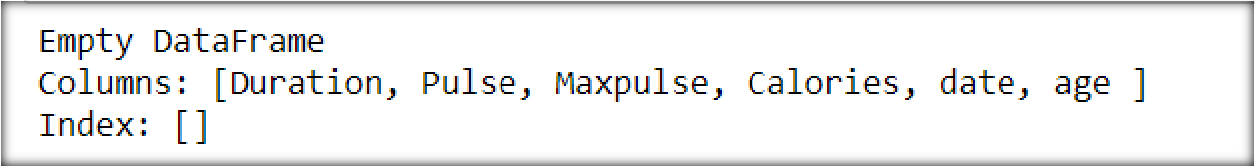
missing\_values = df.isna().sum() print(missing\_values)



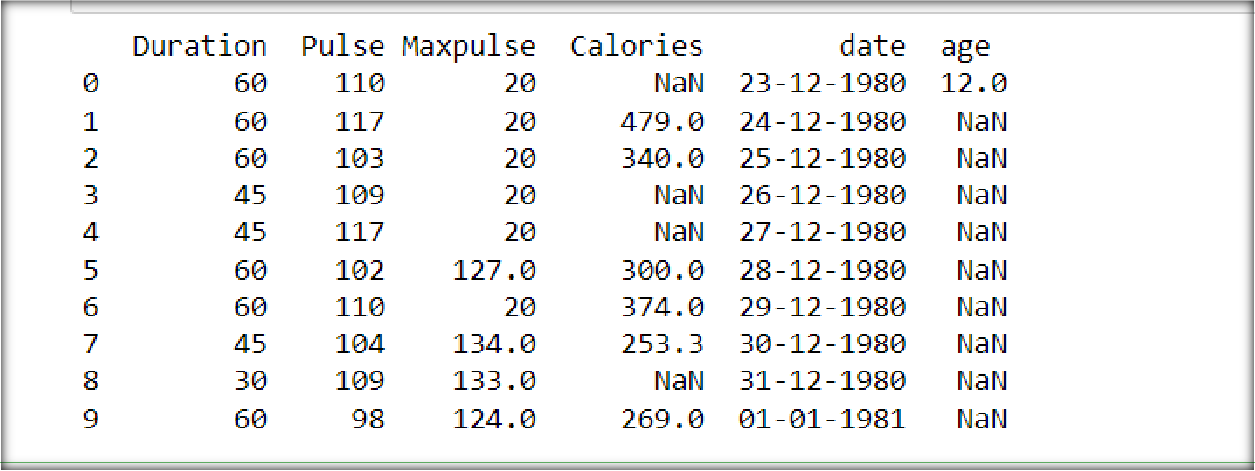
* Remove rows containing empty cells(using dropna()) # Remove rows containing empty cells

df = df.dropna() print(df)

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

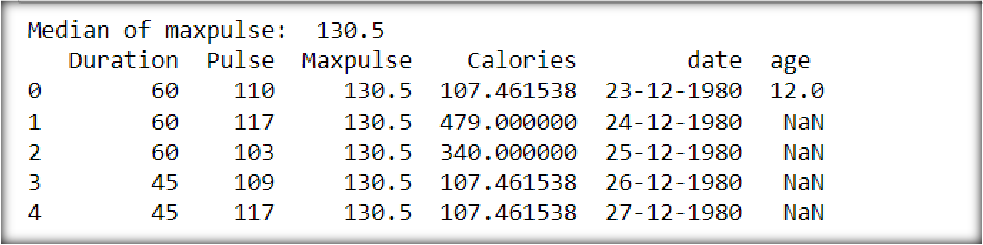


df['Maxpulse'] = df['Maxpulse'].fillna('20') print(df.head(10))



* 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))



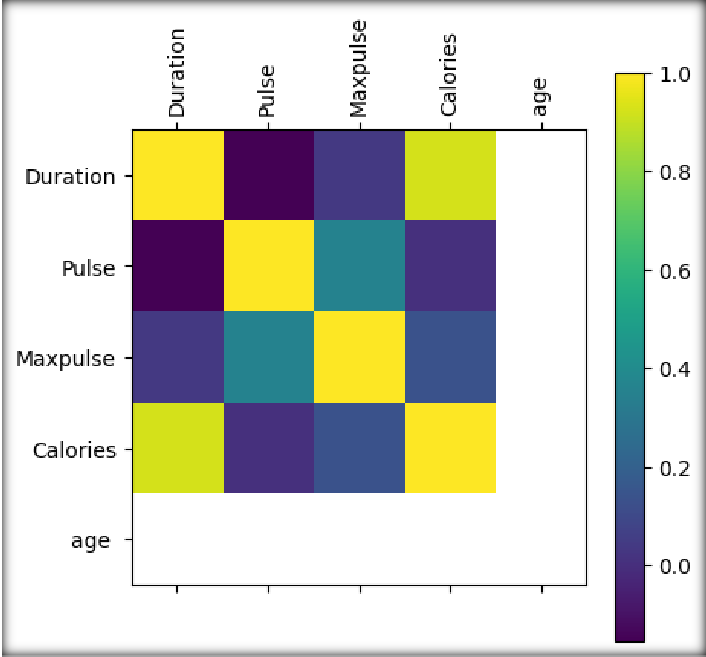
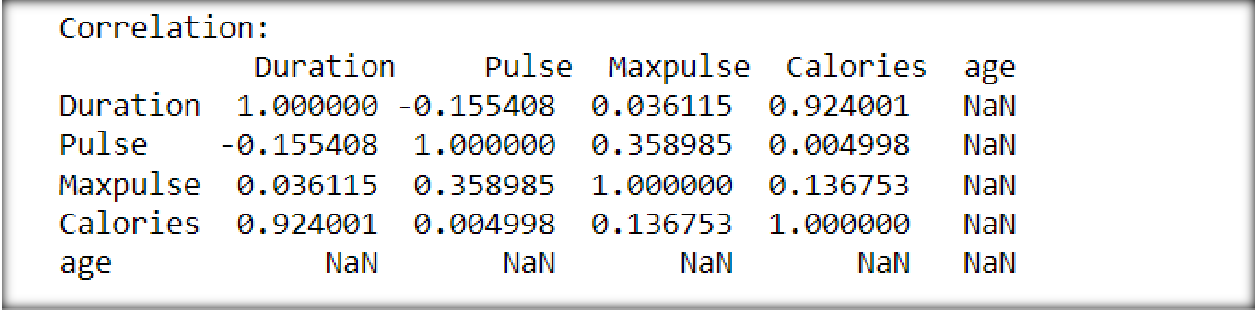
* Find the correlation between columns of the given dataset and visualize this dataset.

#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()

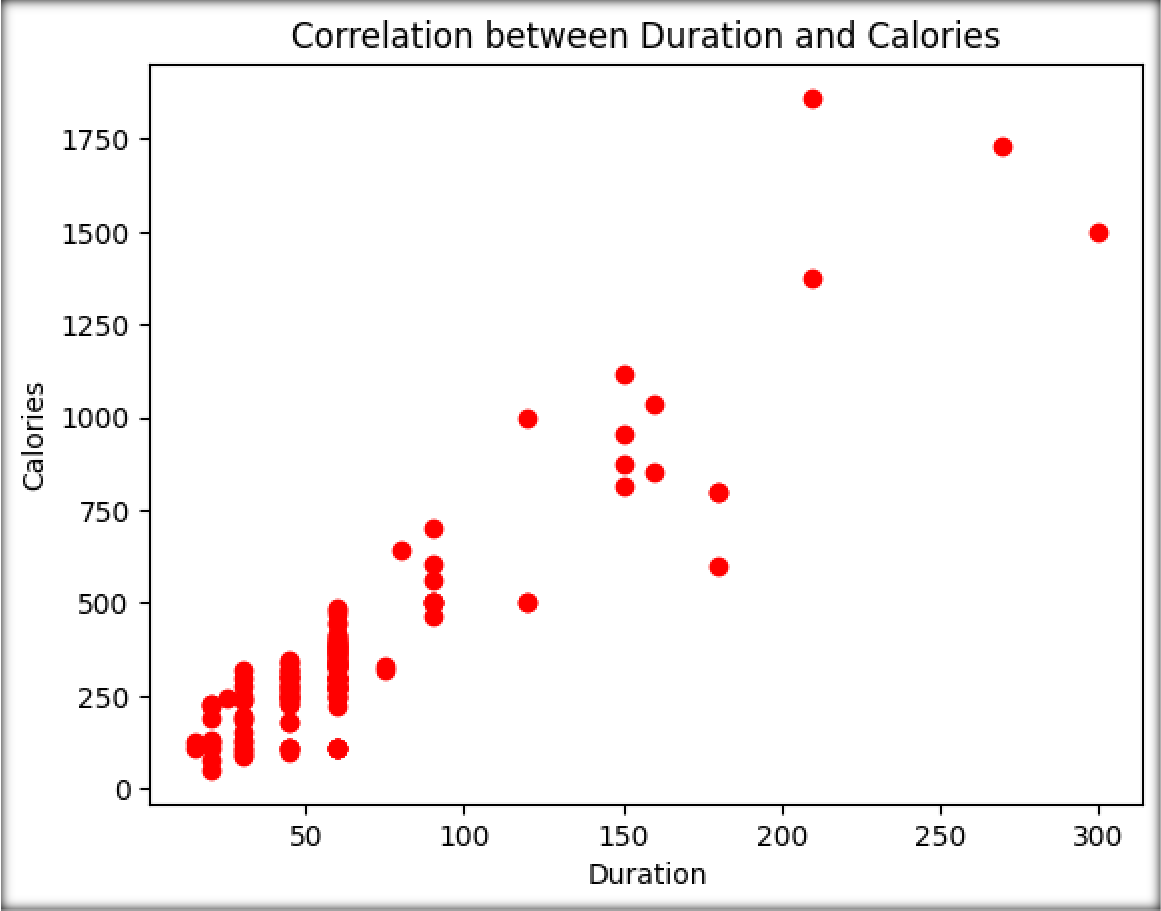


* Show correlation between Duration and Calories columns using scatter plot.

# 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:

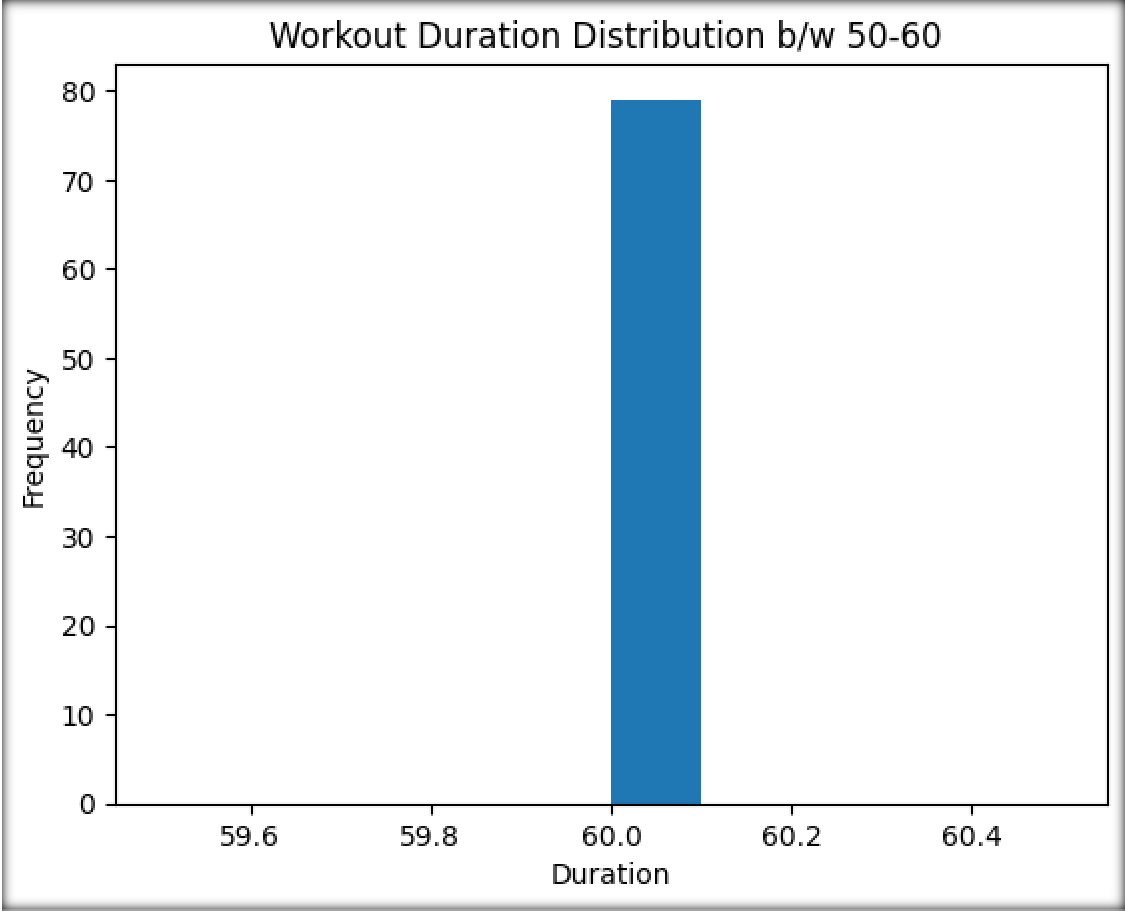
# Draw a histogram to show how many workouts lasted between 50 and 60 minutes

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)

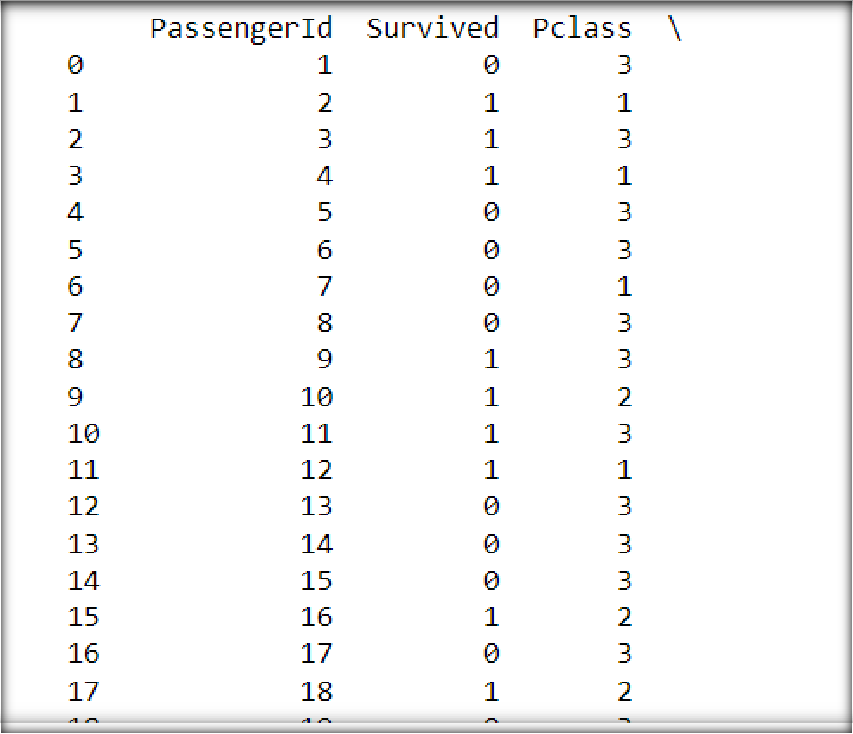
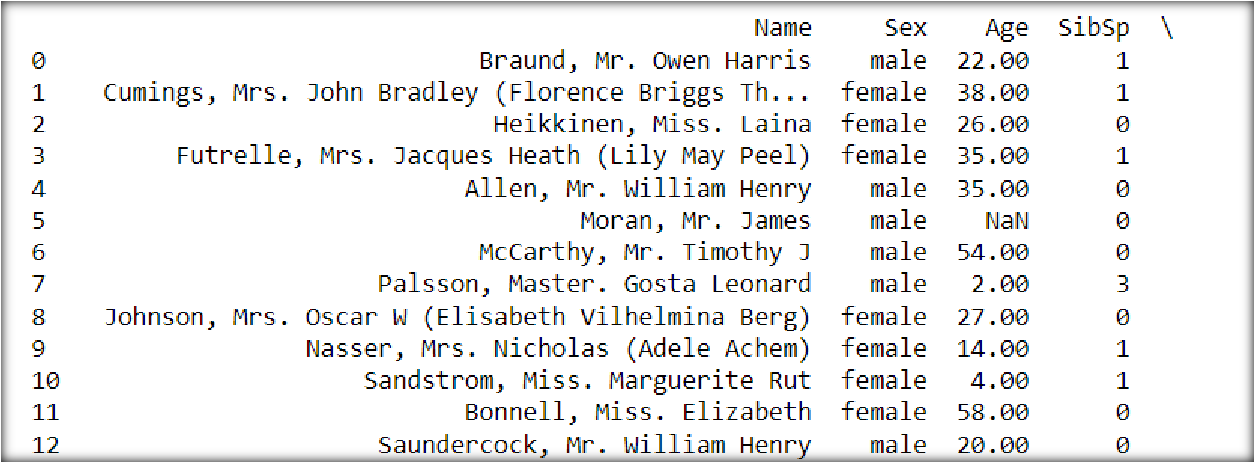
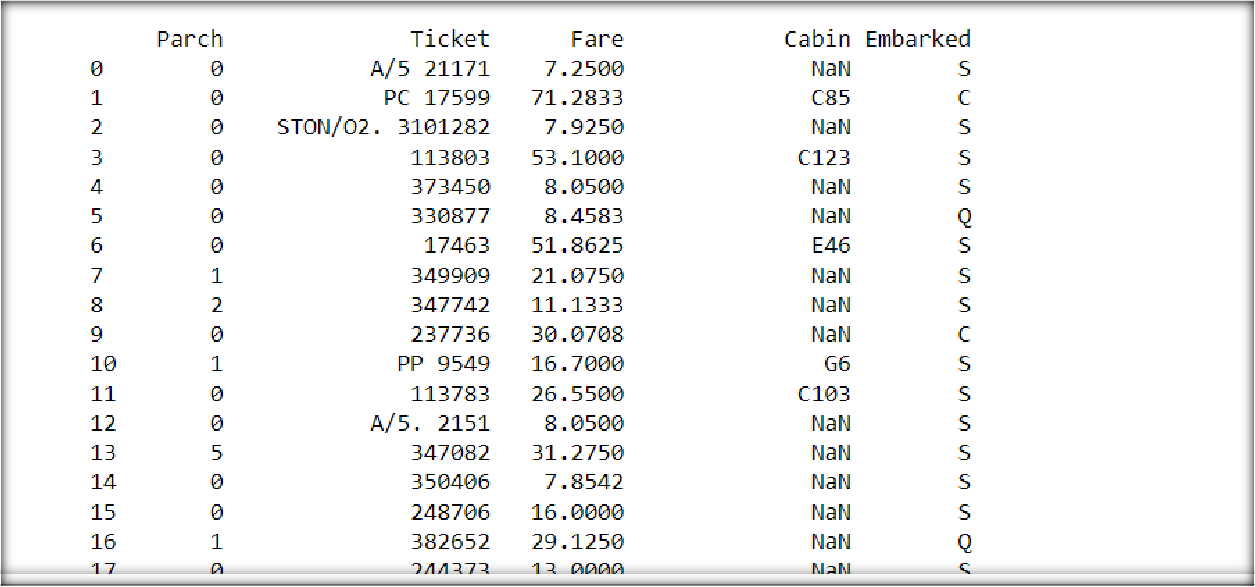
# Read the CSV file

file\_path = 'Titanic-Dataset.csv' # Replace 'data.csv' with the actual path to your .csv file

df = pd.read\_csv(file\_path)

# Display the entire DataFrame print(df)

**OUTPUT:**



**PRACTICAL 20**

Q: 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)

### Code:

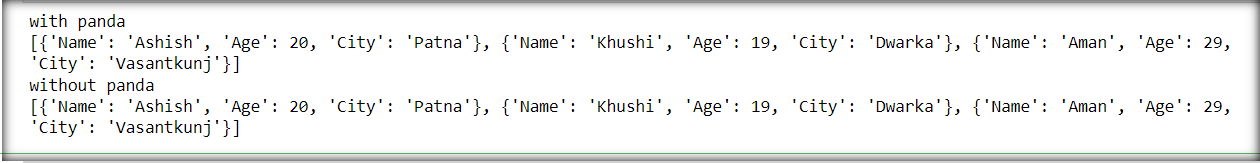
import pandas as pd **# with pandas** print("with panda ")

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)



### # create a json data and read it ina dataframe

import pandas as pd

data = {

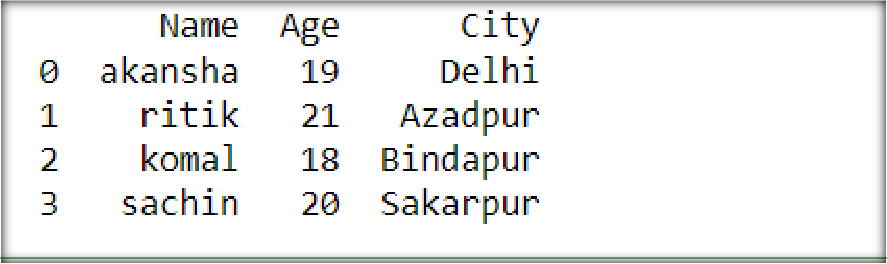
"Name": ["akansha","ritik","komal","sachin"], "Age": [19,21,18,20],

"City": ["Delhi","Azadpur","Bindapur","Sakarpur"]

}

df = pd.DataFrame(data)

# Print the DataFrame print(df)



### # parse a json file

import json

file\_path = "D:\DATA\_SCIENCE\_USING\_PYTHON\example2.json" with open(file\_path, "r") as file:

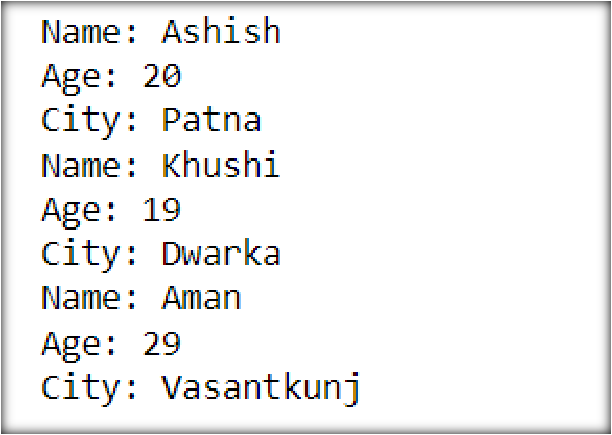
json\_content = json.load(file)

for item in json\_content: print("Name:", item["Name"])

print("Age:", item["Age"])

print("City:", item["City"])

**OUTPUT:**



**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 objectify import 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()

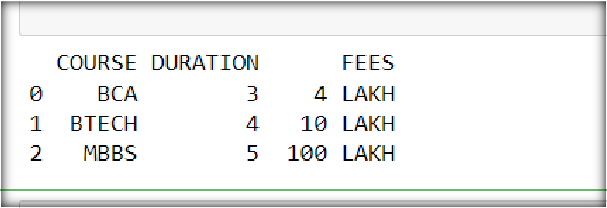
row = dict(zip(['COURSE', 'DURATION', 'FEES'],

[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:**



**PRACTICAL 22**

Q: Write short python codes to demonstrate:

* Uploading
* Streaming
* Sampling

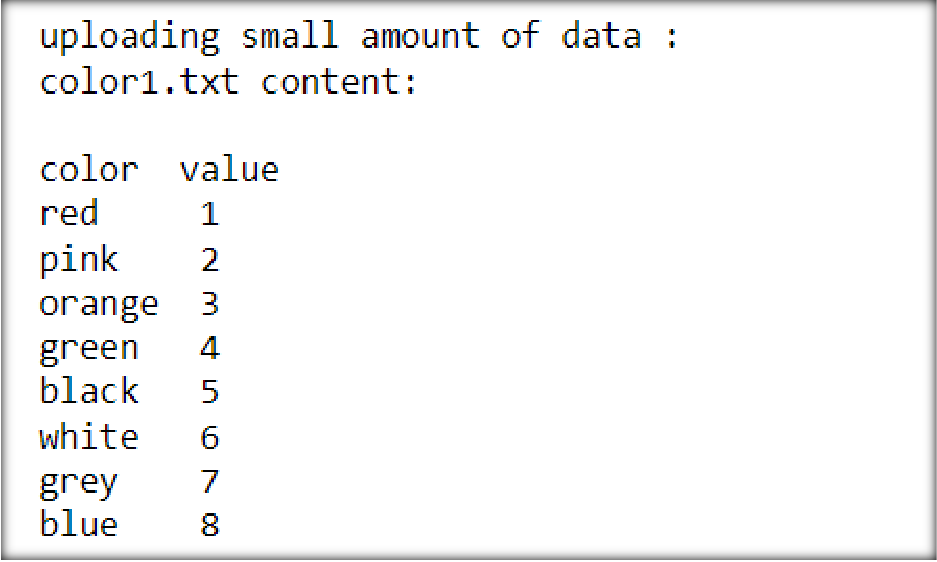
### Code:

**# 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())



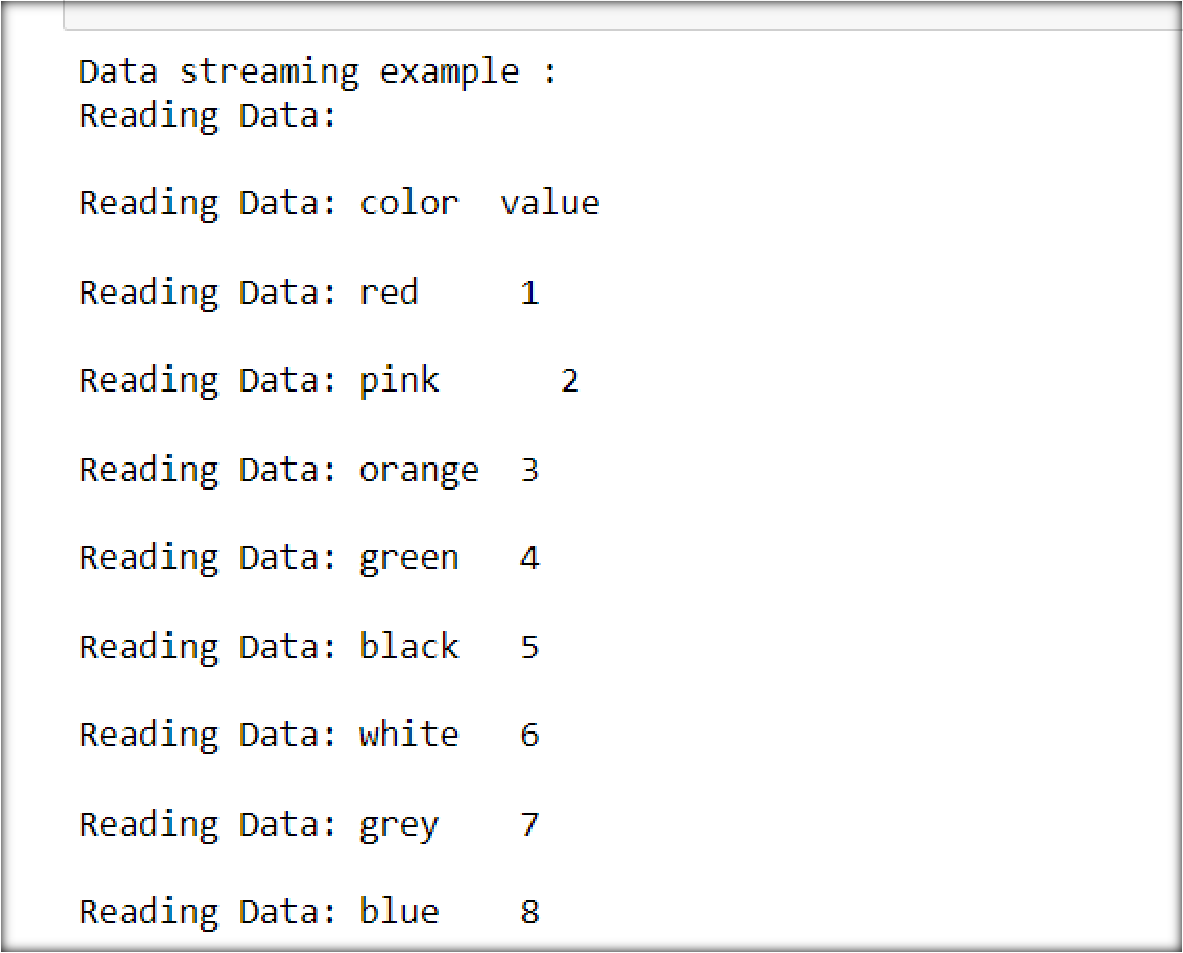
**# streaming data into memory**

print("Data streaming example : ")

with open("color1.txt", 'r') as open\_file:

for observation in open\_file:

print('Reading Data: ' + observation)



**# sampling data in different ways**

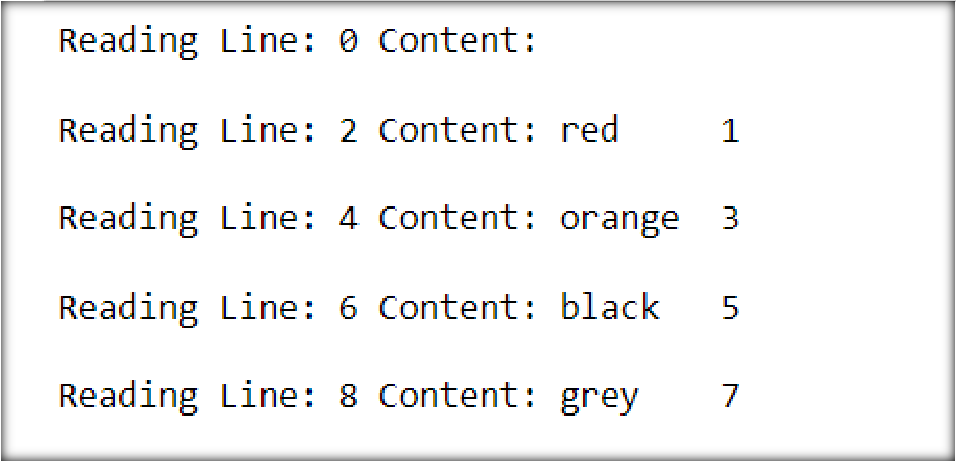
n = 2

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)



**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()

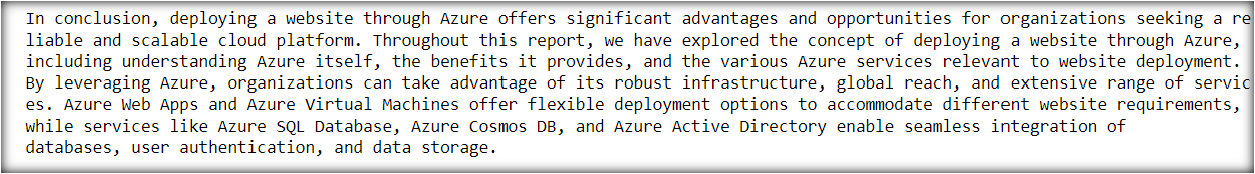
for line in lines:

# 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')



### # parsing a csv file

import csv

def parse\_csv\_file(file\_path):

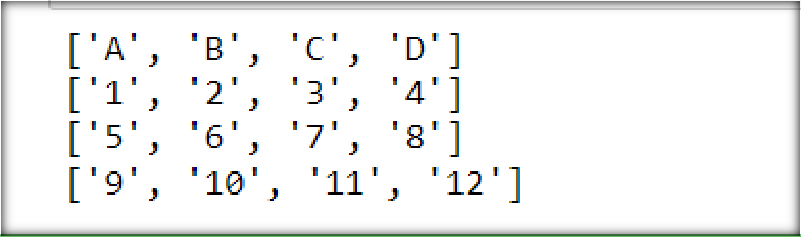
with open(file\_path, 'r') as file:

reader = csv.reader(file)

for row in reader:

print(row)

file\_path = 'myfile.csv' parse\_csv\_file(file\_path)



### # 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

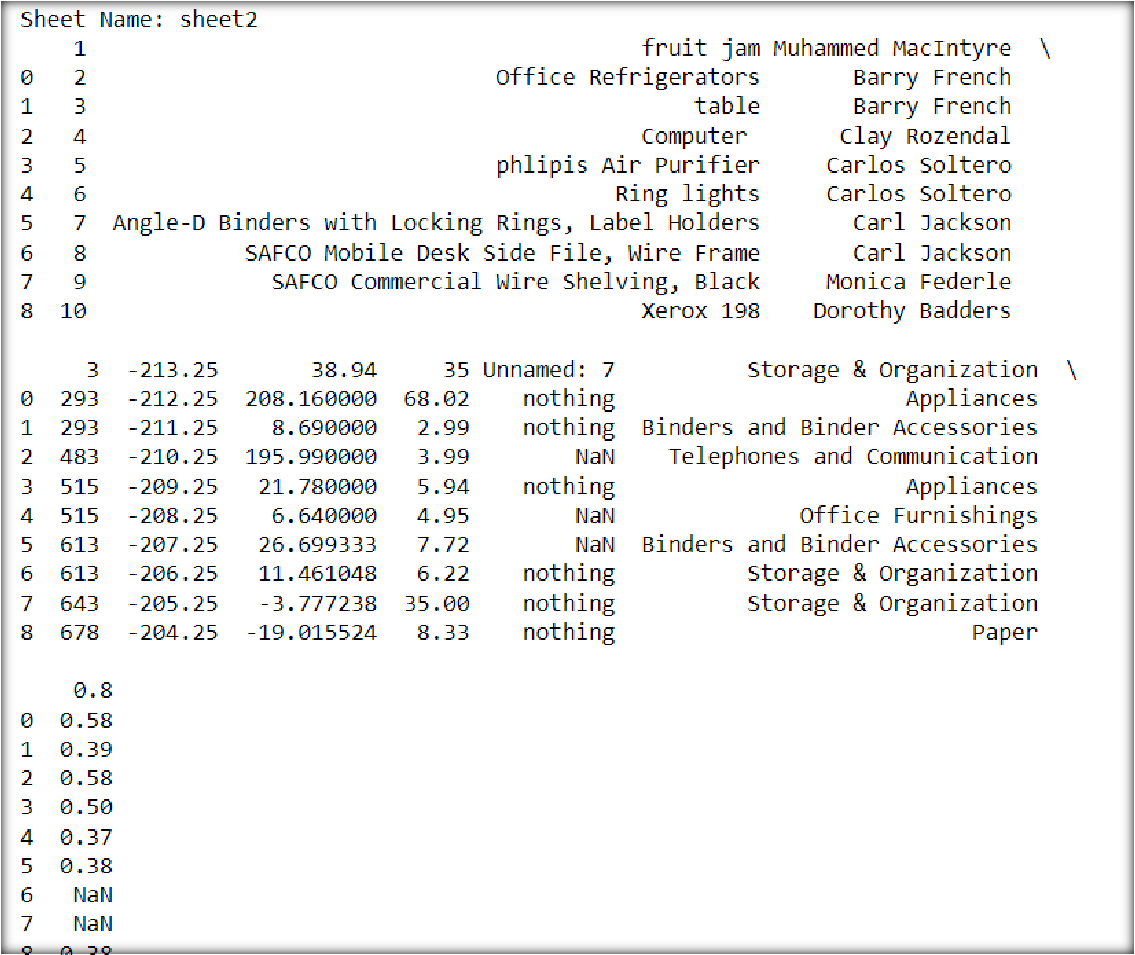
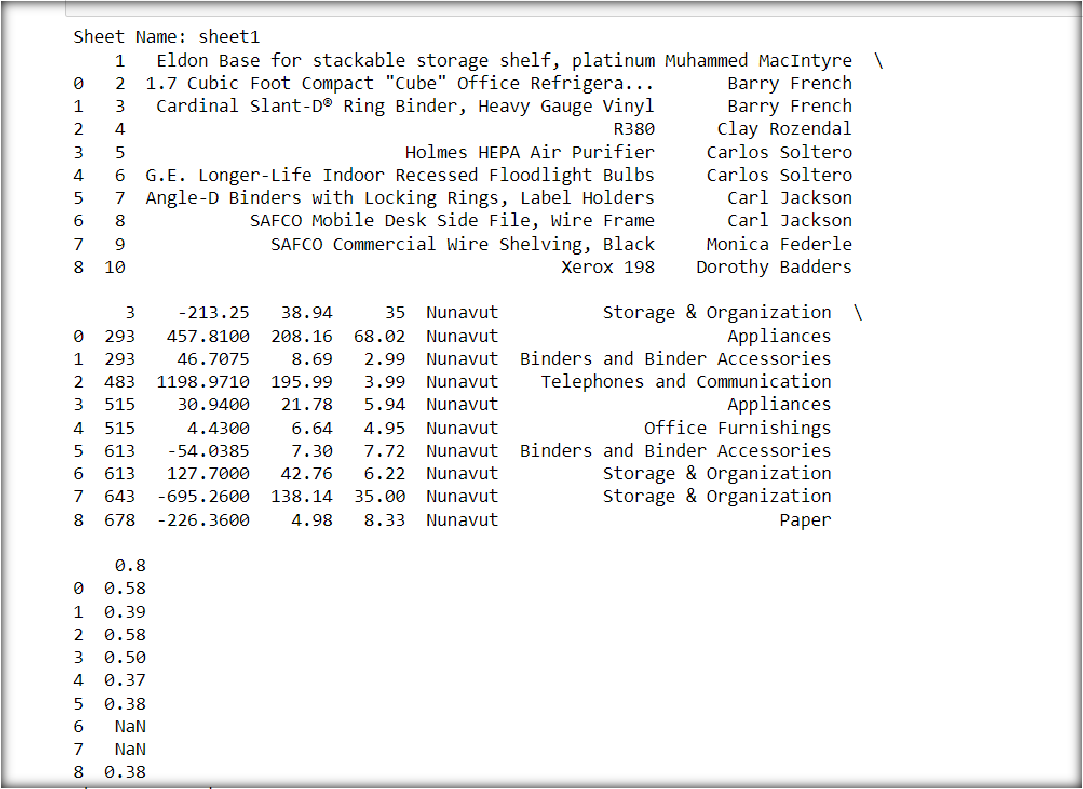
print(f"Sheet Name: {sheet\_name}")

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)



**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 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

### Code:

**# gray scaling**

from skimage.io import imread

from skimage.transform import resize from matplotlib import pyplot as plt import matplotlib.cm as cm

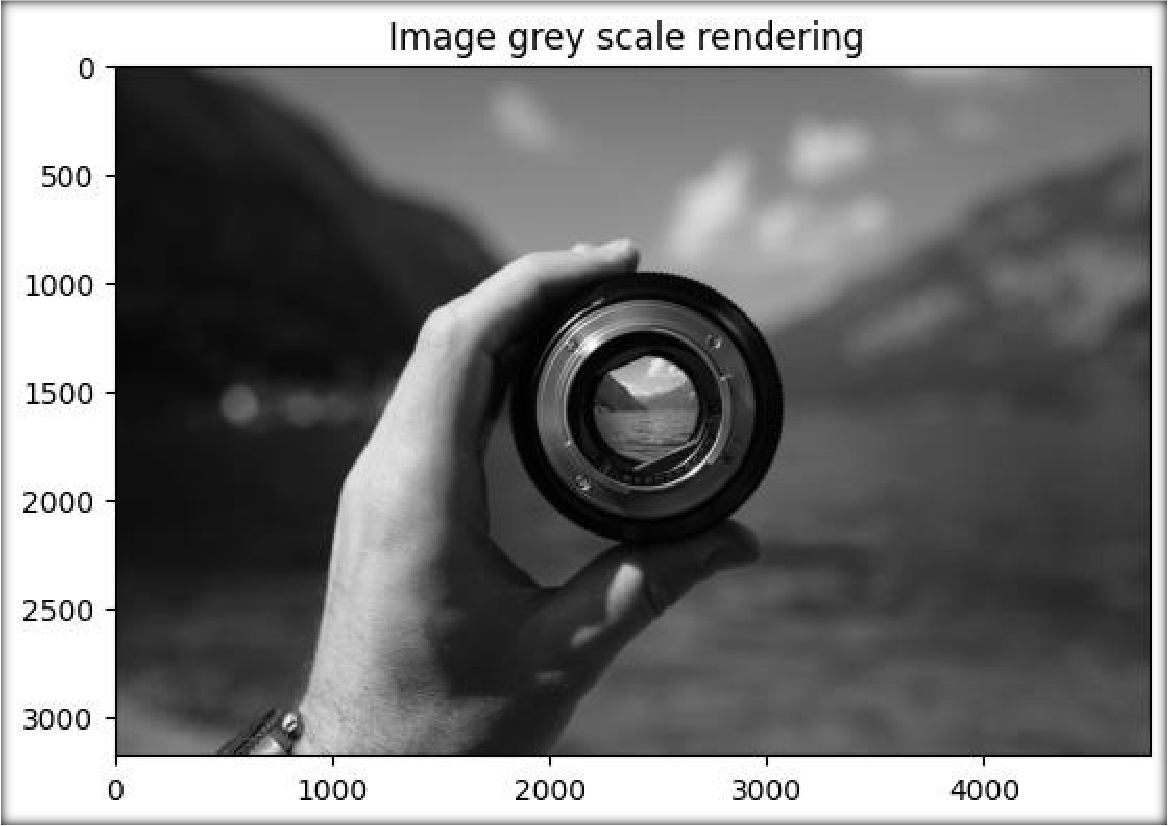
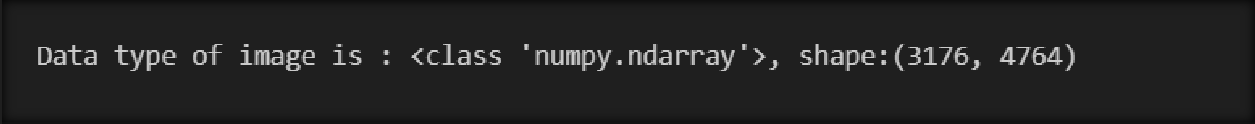
example\_file=("https://[www.seiu1000.org/sites/main/files/main-](http://www.seiu1000.org/sites/main/files/main-) images/camera\_lense\_0.jpeg")

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))



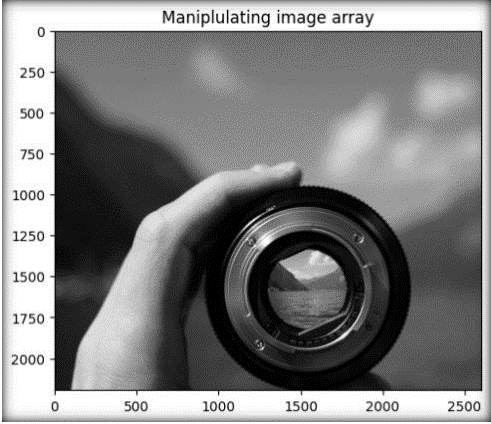
**# manipulating image array**

image2=image1[5:2200 ,900:3500] plt.imshow(image2,cmap=cm.gray)

plt.title("Maniplulating image array")

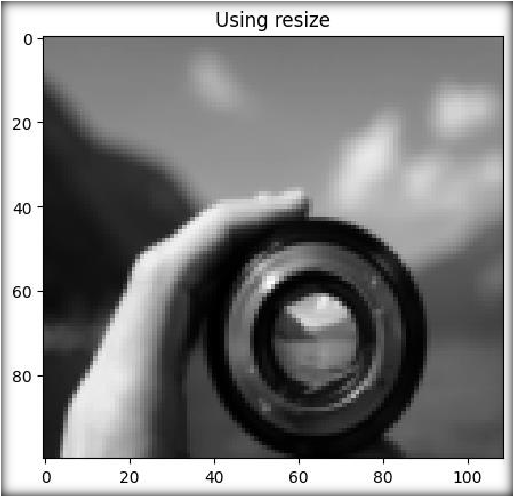
plt.show()

### OUTPUT:



**# using resize() function** image3=resize(image2,(100,109),mode='symmetric') plt.imshow(image3,cmap=cm.gray)

plt.title("Using resize") plt.show()



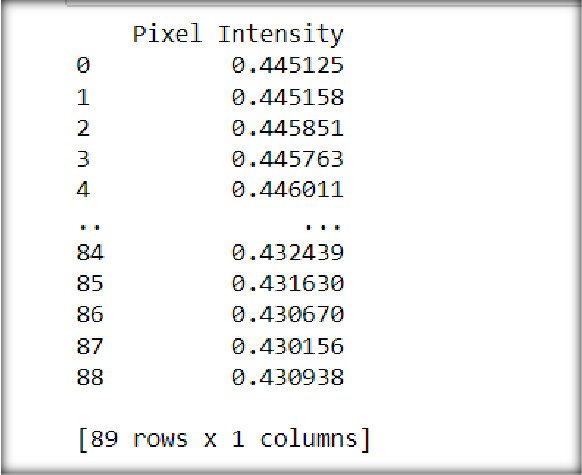
### # convert to dataset

image\_row = image3.flatten()

df = pd.DataFrame(image\_row, columns=["Pixel Intensity"])

csv\_filename = "image3\_dataset.csv" # Replace with the desired file name df.to\_csv(csv\_filename, index=False)

print(df.head(89))



**PRACTICAL 25**

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

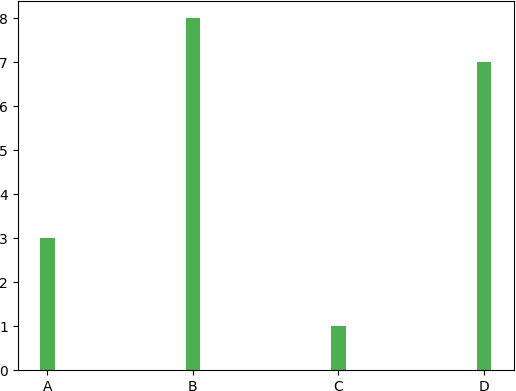
* bar(), barh() while changing their default color, width, and height .

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

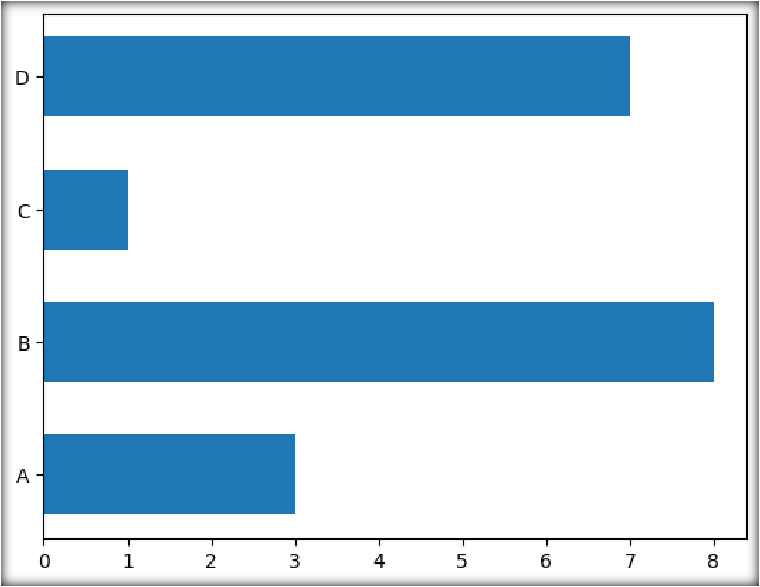
import numpy as np

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 and default colors, explode all wedges.

### Code:

import matplotlib.pyplot as plt import numpy as np

Marks = np.array([99,98,78,69])

subject= ['Maths','Science','Reasoning','Geo.'] myexplode = [0.1, 0.2, 0.1, 0.3]

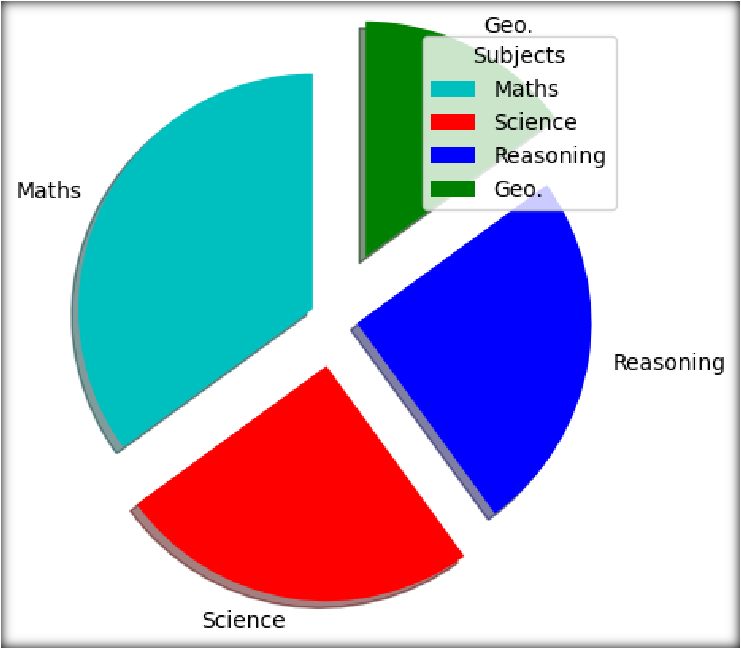
mycolors = ['c','r','b','g']

plt.pie(y, labels = subject, startangle = 90, explode = myexplode, shadow = True, colors = mycolors)

plt.legend(title = "Subjects",loc=1)

plt.show()

**OUTPUT:**



**PRACTICAL 26**

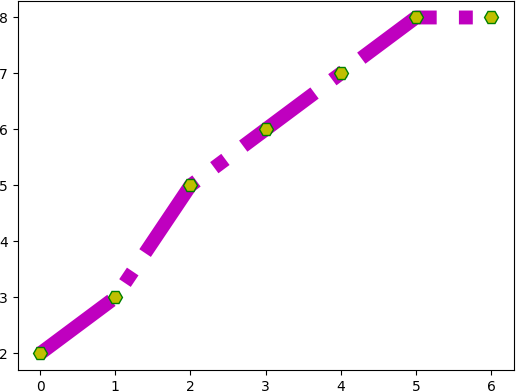
Q: Write a python program to draw a line with y points only(i.e. default x points) having following properties:

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

import matplotlib.pyplot as plt import 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])

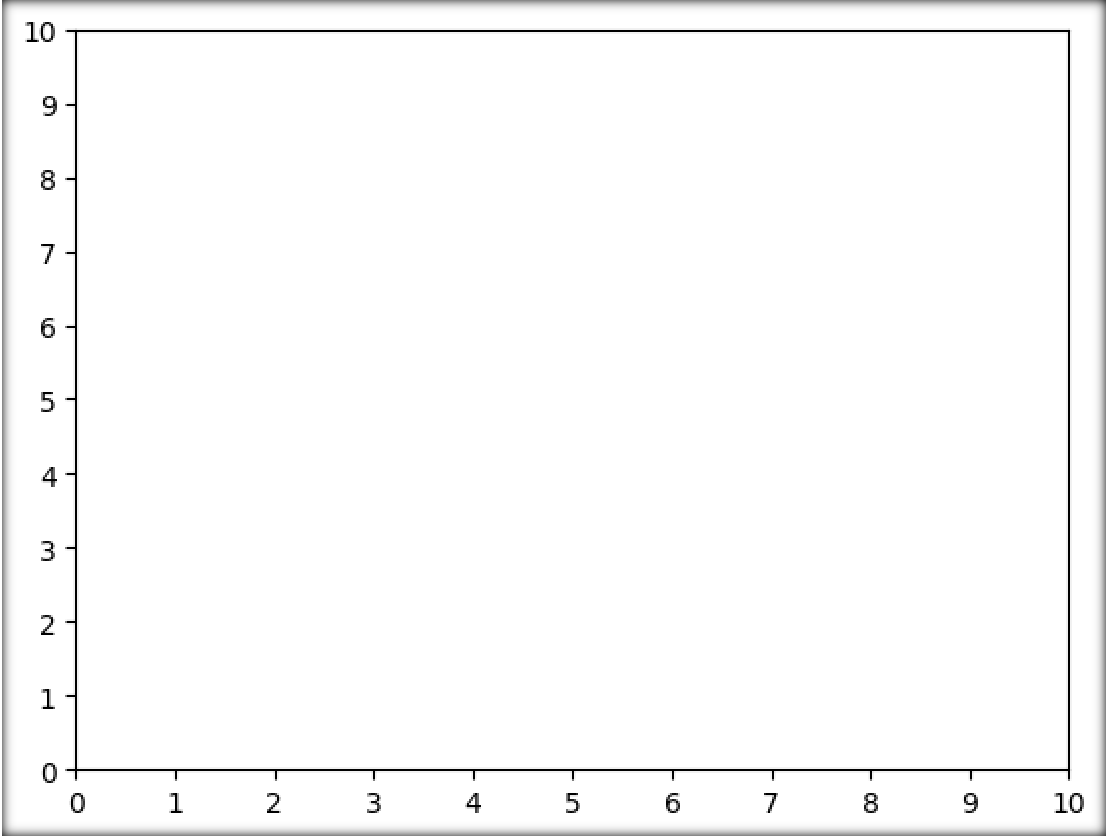
ax.set\_xticklabels([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

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

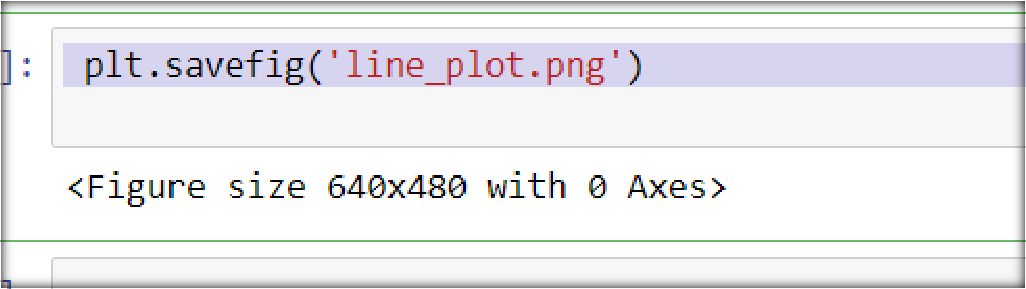
plt.show()



* 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 linewidth and 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")

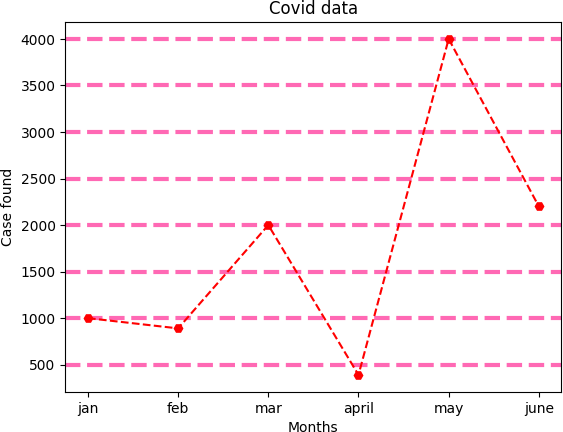
plt.xlabel("Months")

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 the graph(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)

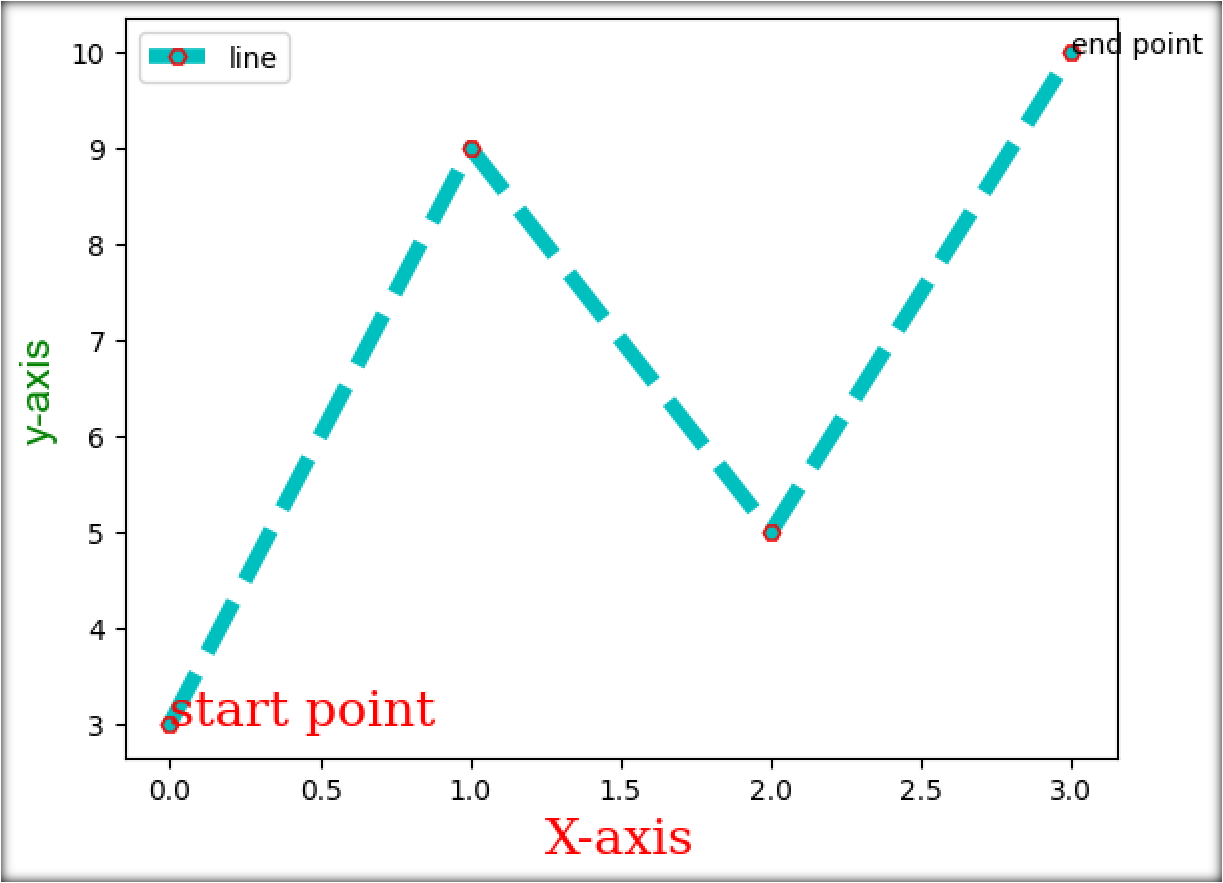
plt.text(0,3,"start point",font1)

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

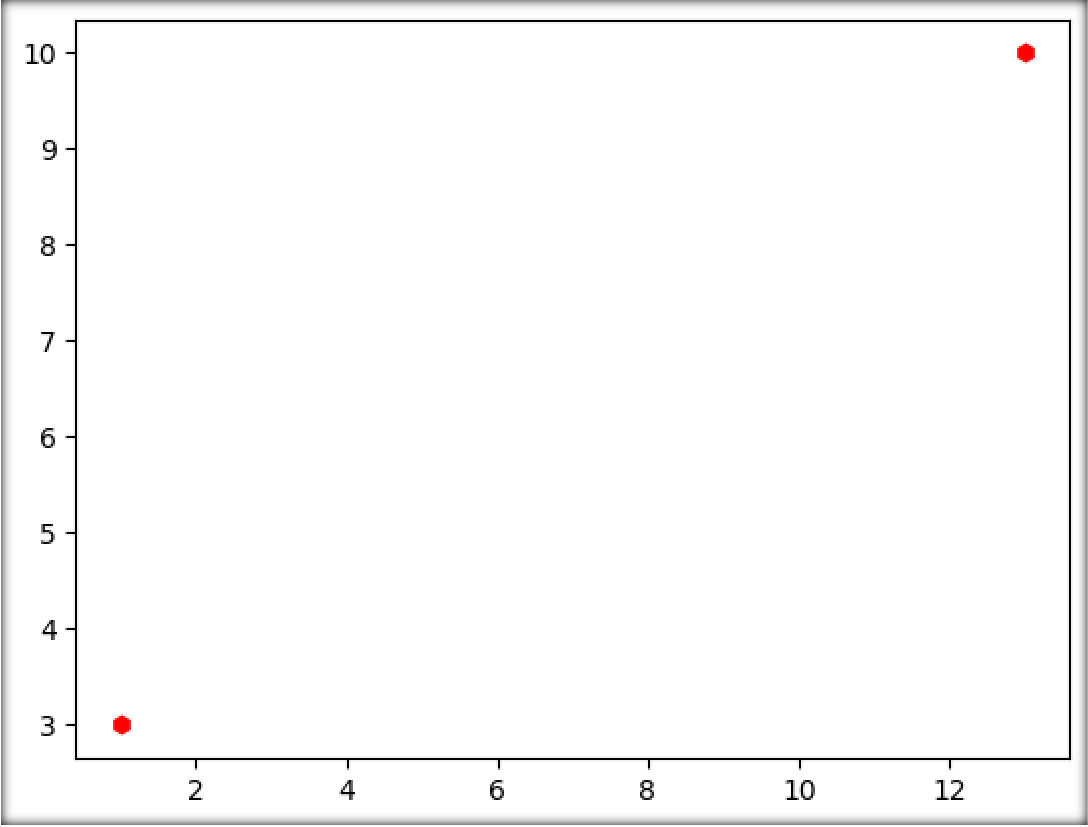
import numpy as np

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])

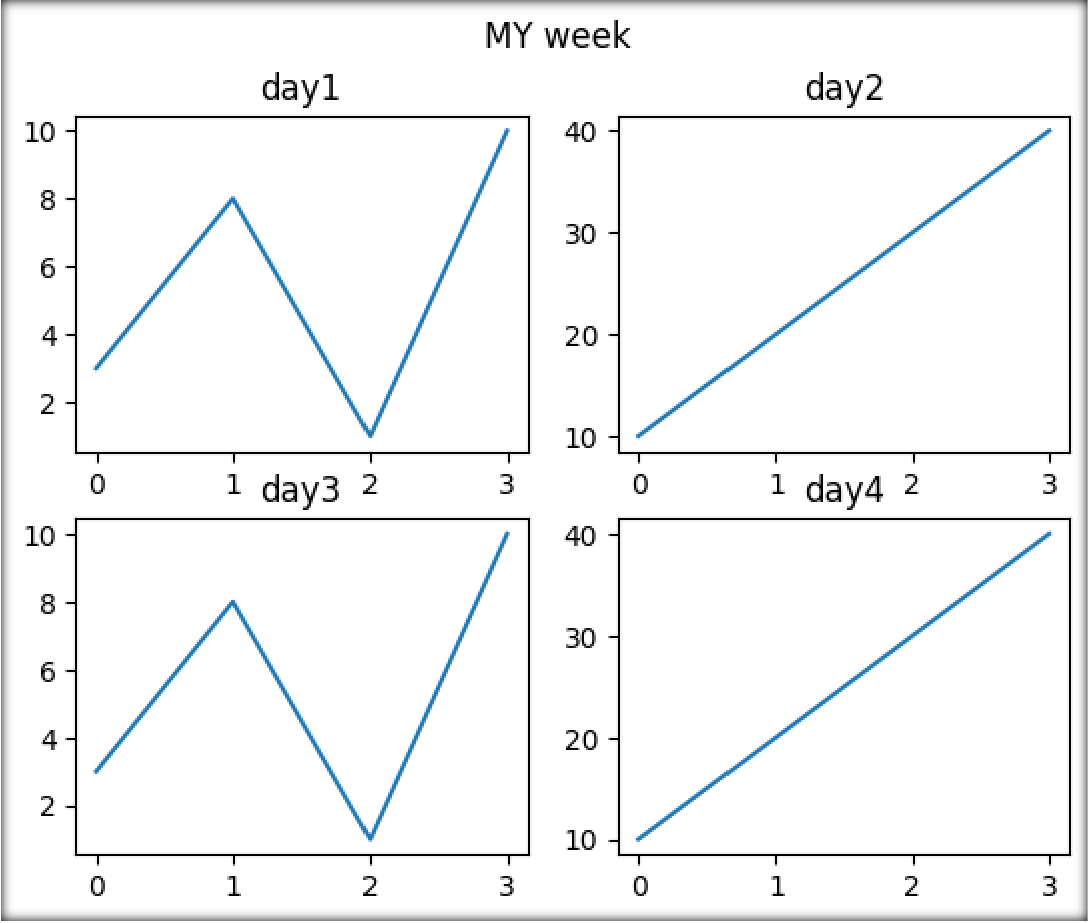
plt.subplot(2, 2, 4)

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 cars in 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:

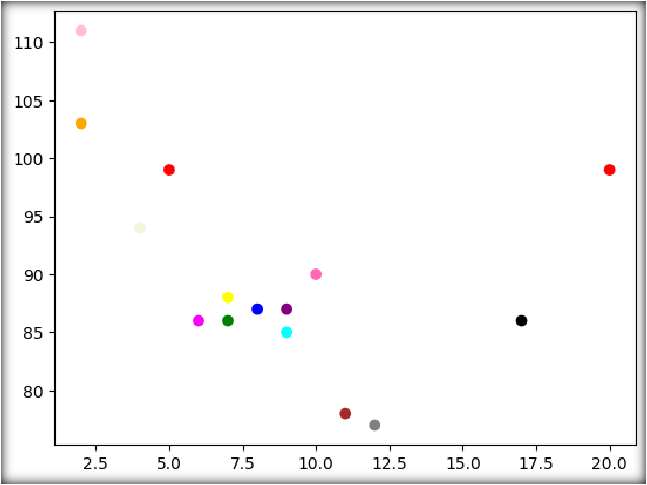
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6,20,10])

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])

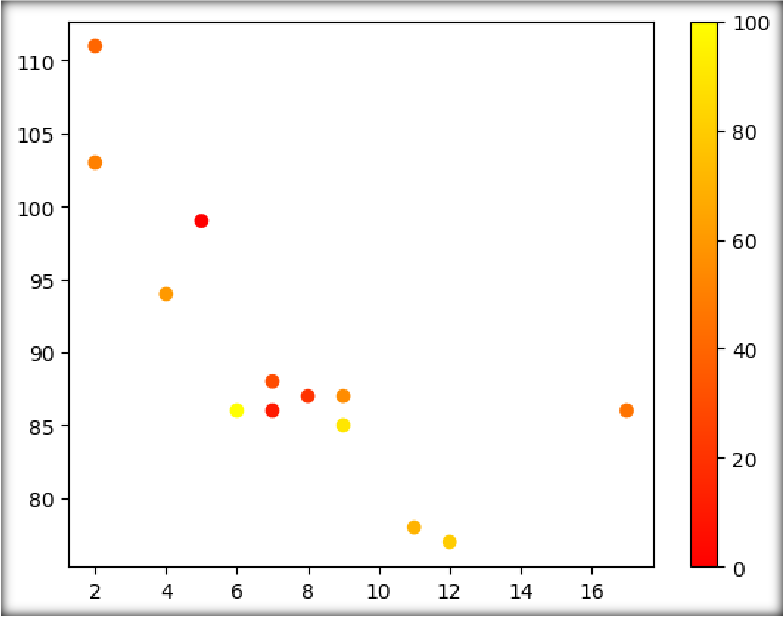
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

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

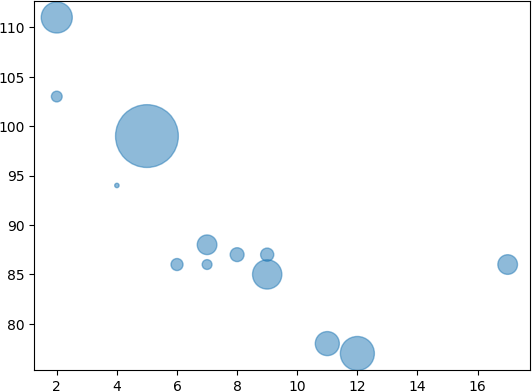
import numpy as np

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:

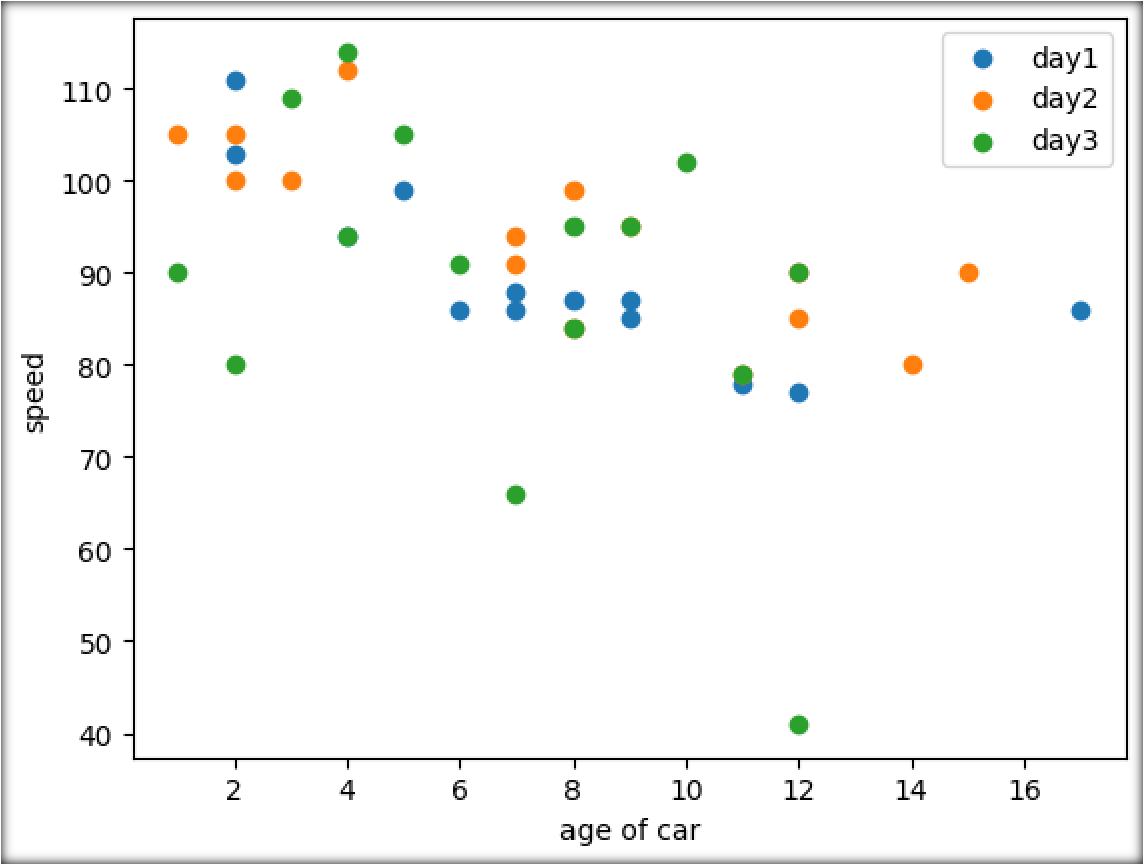
x = np.array([2,5,8,10,12,8,1,9,4,3,11,4,6,7,12])

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")

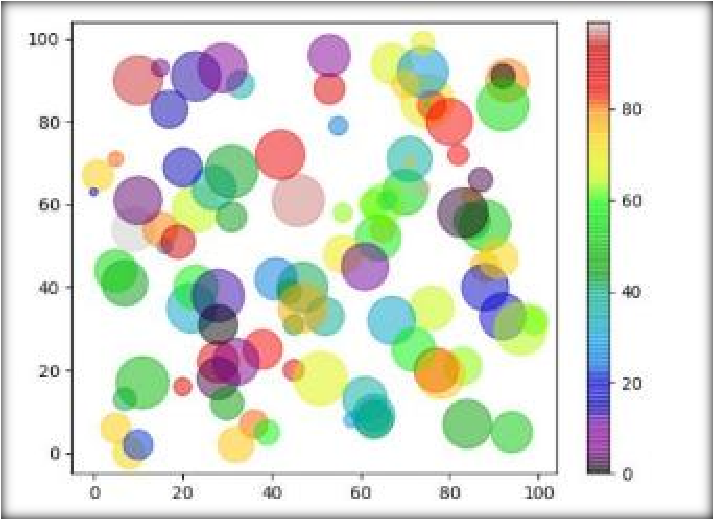
plt.legend(['day1','day2','day3']) plt.show()

**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))

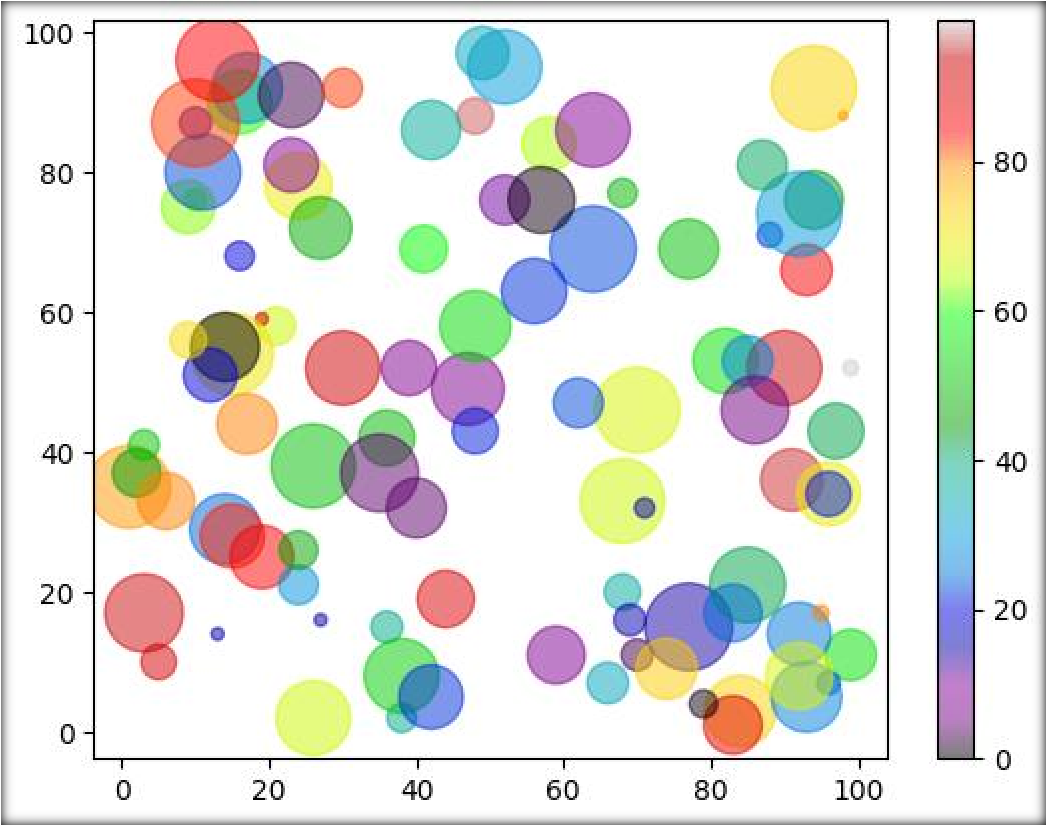
colors = np.random.randint(100, size=(100)) sizes = 10 \* np.random.randint(100, size=(100))

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

plt.colorbar()

plt.show()

**OUTPUT:**



**PRACTICAL 30**

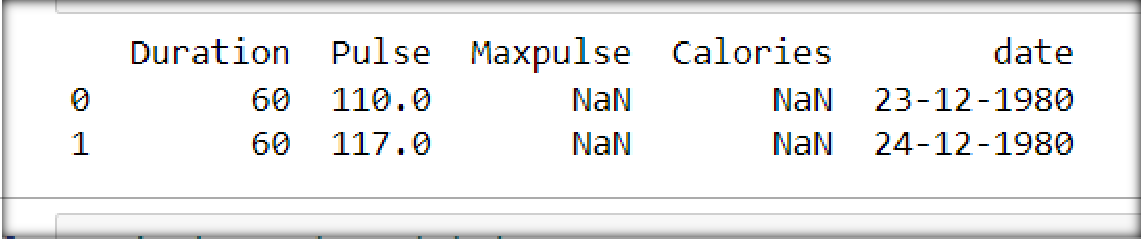
Q: Students can make assumptions for this question.

* 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')

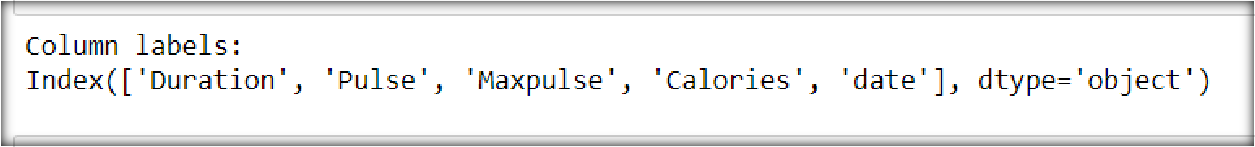
print(df.head(2))



* Display all the column labels of your dataset .

print("Column labels:")

print(df.columns)

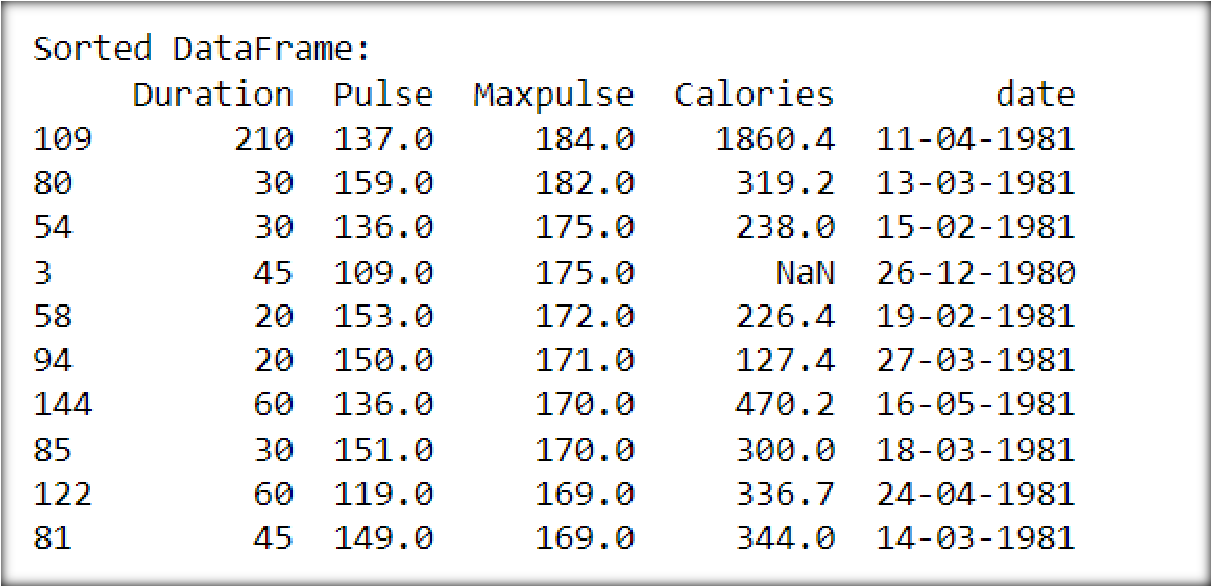


* Sort the DataFrame based on a particular numerical column in descending order .

sorted\_df = df.sort\_values(by='Maxpulse', ascending=False)

print("\nSorted DataFrame:")

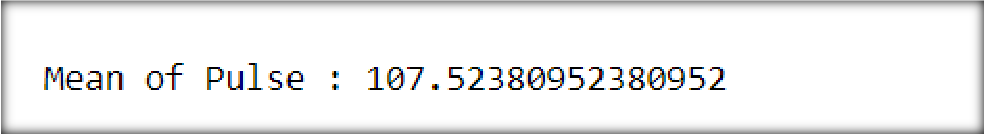
print(sorted\_df.head(10))



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

column\_mean = df['Pulse'].mean()

print("\n Mean of Pulse :", column\_mean)

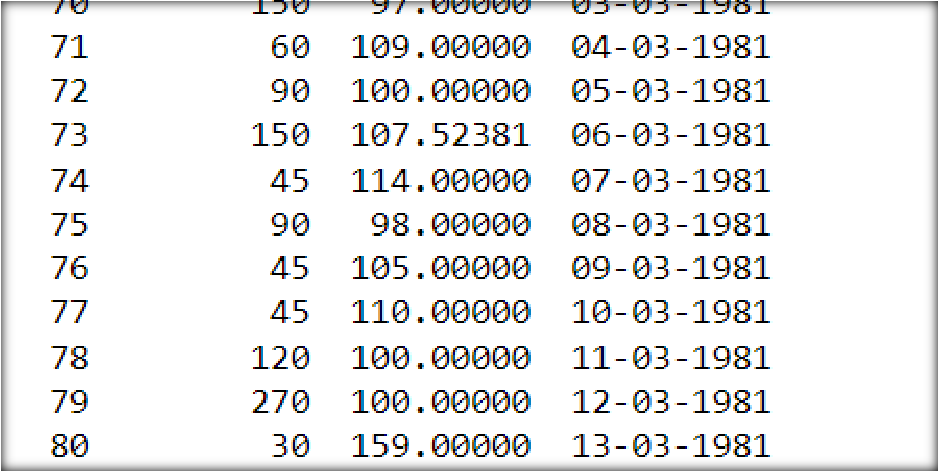
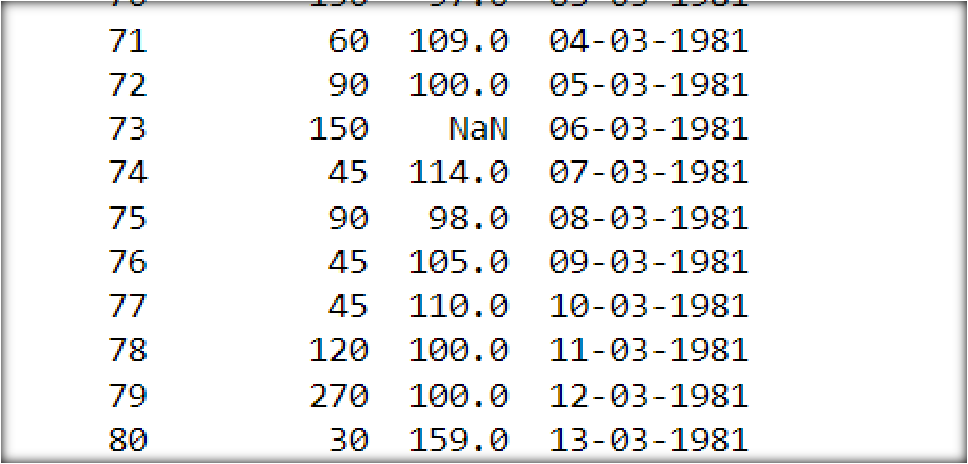


* Fill the missing values in column A with this mean value assuming no outliers are present in that column.

df['Pulse'].fillna(column\_mean, inplace=True)

print("\nDataFrame after filling missing values:")

print(df.to\_string())



* Remove the column having more than 4 null values.

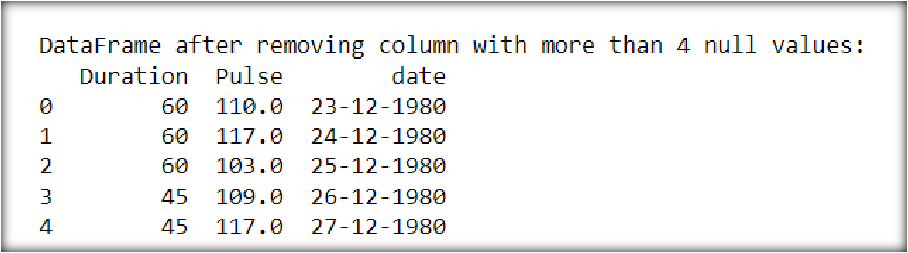
### Code:

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:**



**PRACTICAL 31**

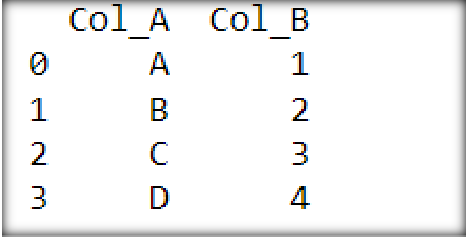
Q: Assume that you have a DataFrame as DataFrame([["A", 1], ["B", 2], ["C", 3], ["D", 4]],columns = ["Col\_A", "Col\_B"]).

import pandas as pd

import numpy as np

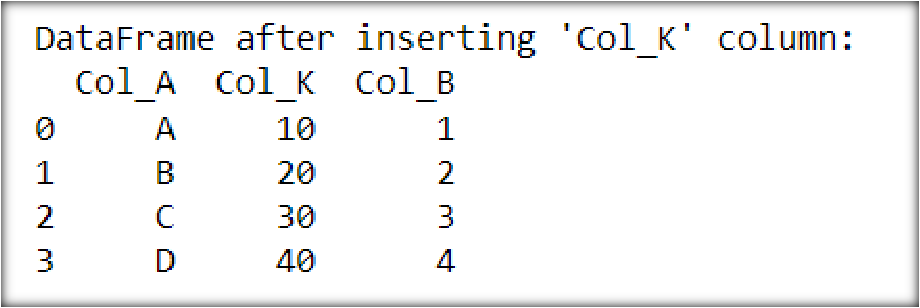
df = pd.DataFrame([["A", 1], ["B", 2], ["C", 3], ["D", 4]], columns=["Col\_A", "Col\_B"])

print(df)



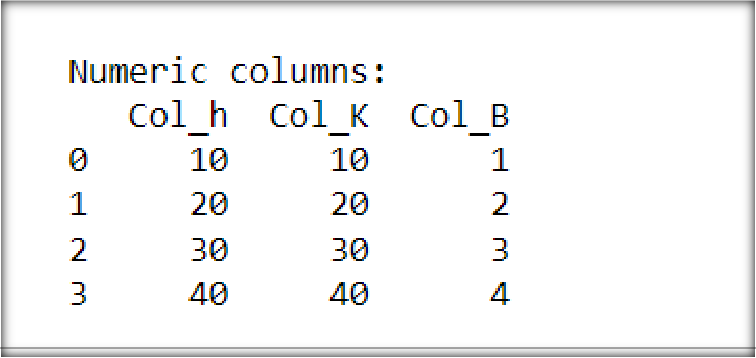
* Insert a column at a specific location in a DataFrame. df.insert(loc=1, column='Col\_K', value=[10,20,30,40]) print("DataFrame after inserting 'Col\_K' column:")

print(df)



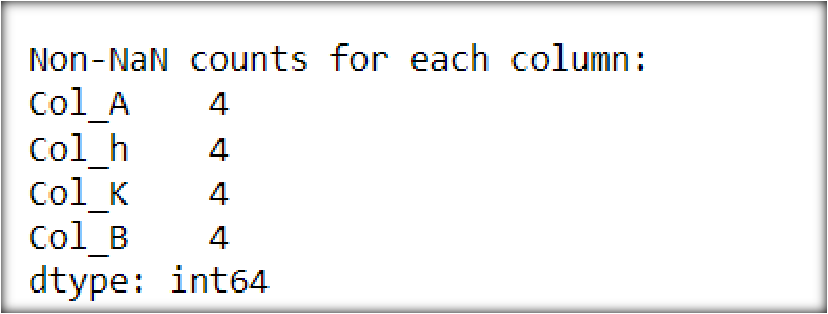
* Select columns based on the column’s Data Type .. numeric\_columns = df.select\_dtypes(include='number')

print("\nNumeric columns:") print(numeric\_columns)



* 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)



* Split DataFrame into equal parts .

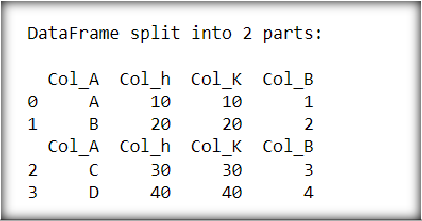
num\_parts = 2

df\_parts = np.array\_split(df, num\_parts)

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

for part in df\_parts:

print(part)



* Reverse DataFrame row-wise or column-wise

### Code:

**#rowwise**

df\_reverse\_row = df[::-1]

print("\nDataFrame after reversing row-wise:")

print(df\_reverse\_row)

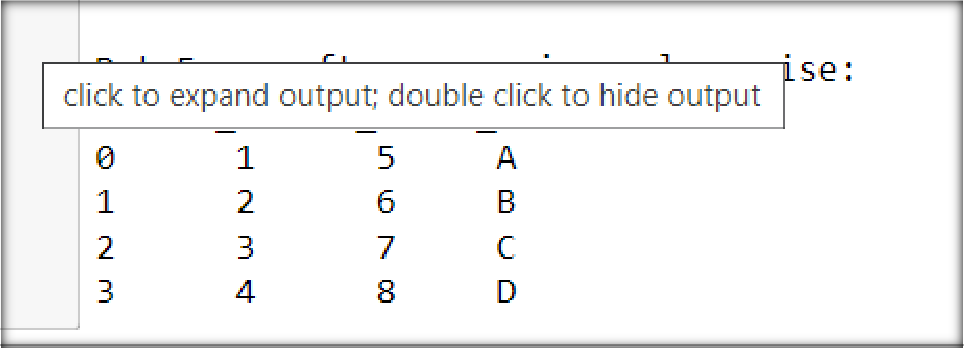
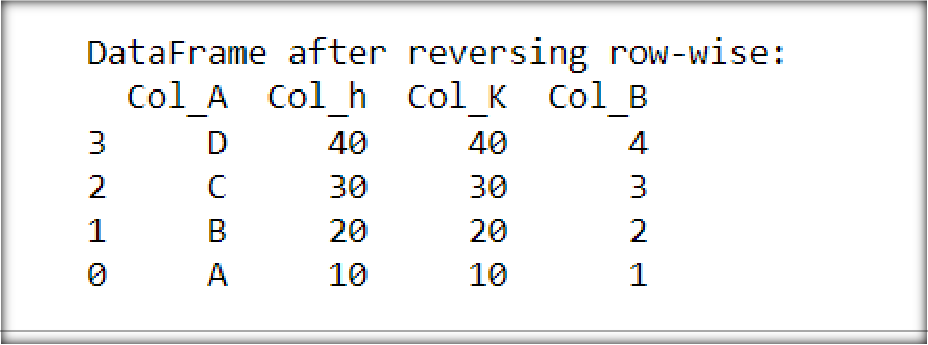
### #columnwise

df\_reverse\_column = df.iloc[:, ::-1]

print("\nDataFrame after reversing column-wise:")

print(df\_reverse\_column)

**OUTPUT:**



**PRACTICAL 32**

Q: 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.

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('company\_sales\_data.csv') profitList = df ['total\_profit'].tolist()

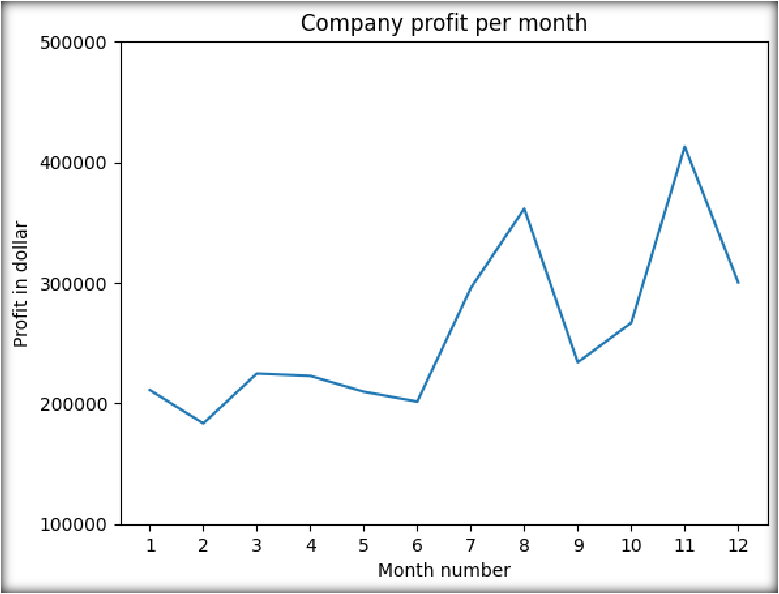
monthList = df ['month\_number'].tolist()

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 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.

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 ')

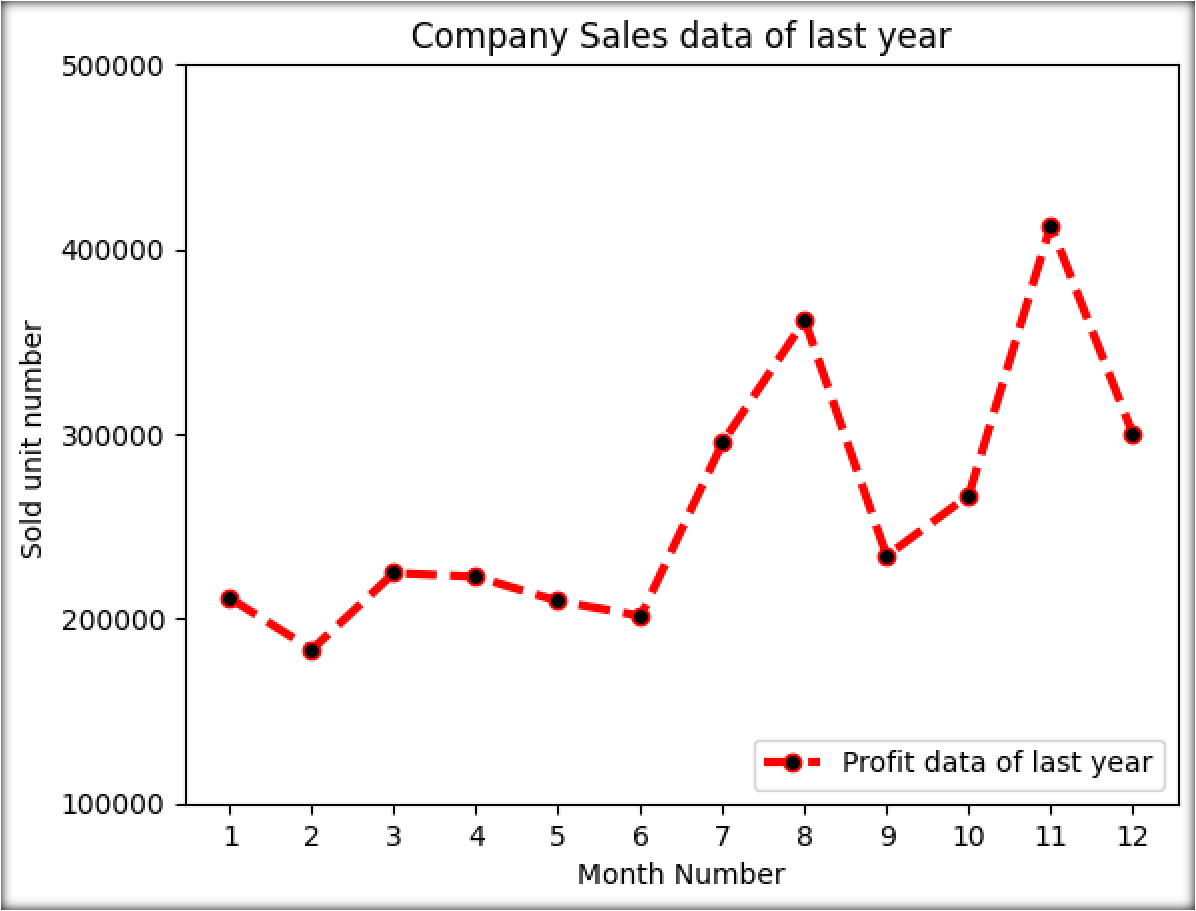
plt.legend(loc='lower right')

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)

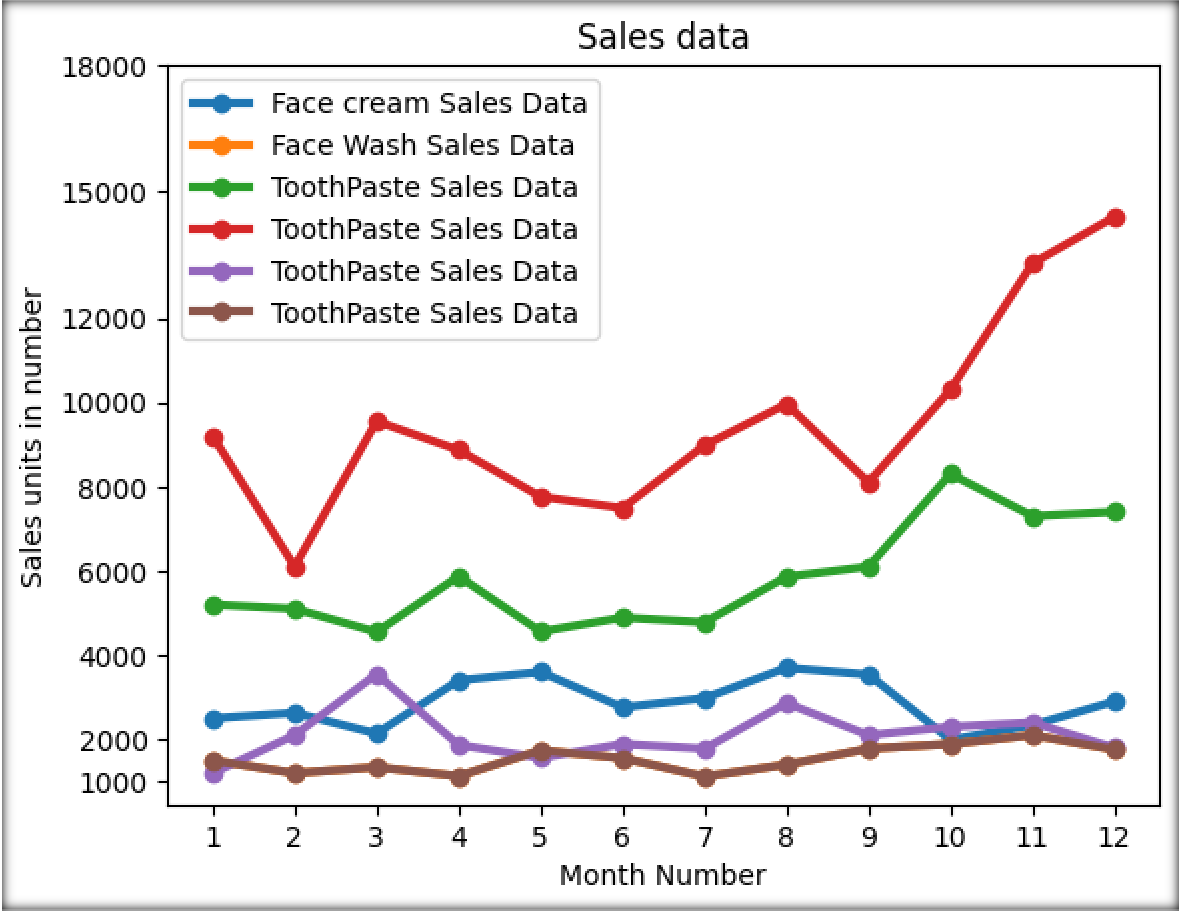
plt.plot(monthList, shampooSalesData, label = 'ToothPaste Sales Data', marker='o', linewidth=3)

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()



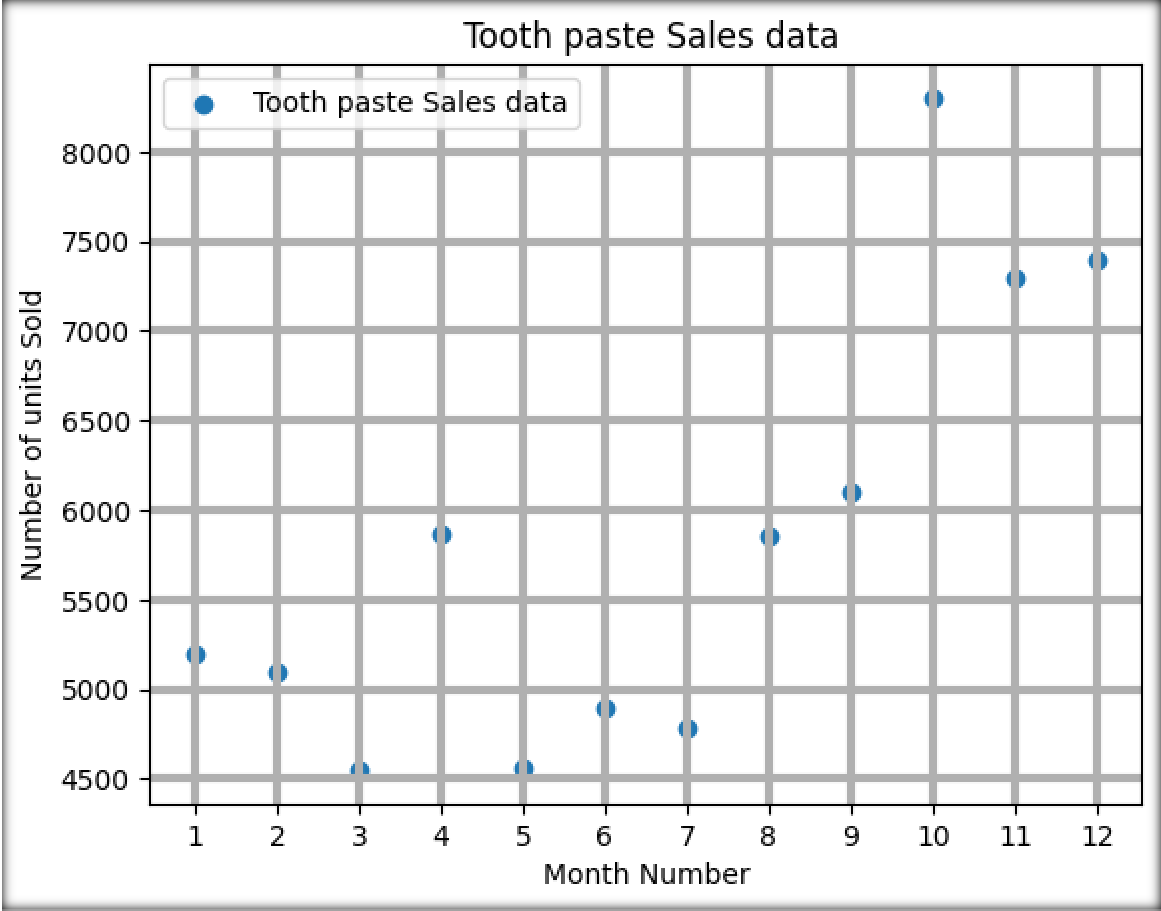
* Read toothpaste sales data of each month and show it using a scatter plot

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 the bar chart.

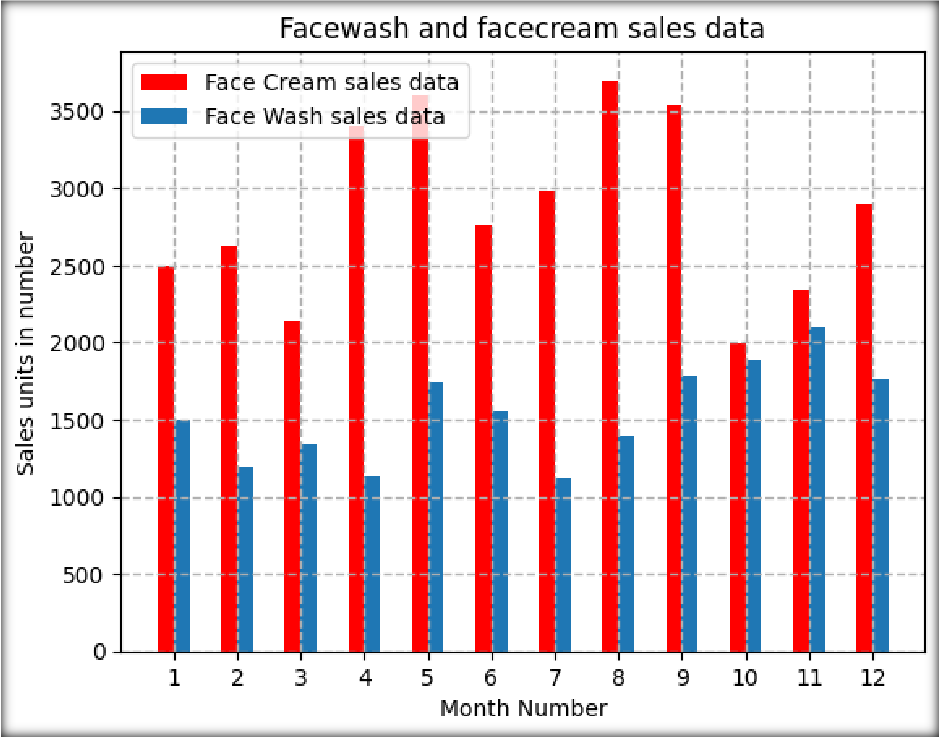
monthList = df ['month\_number'].tolist() faceCremSalesData = df ['facecream'].tolist() faceWashSalesData = df ['facewash'].tolist()

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

plt.bar([a+0.25 for a in monthList], faceWashSalesData, width= -0.25, label = 'Face Wash 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 bar chart. 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')

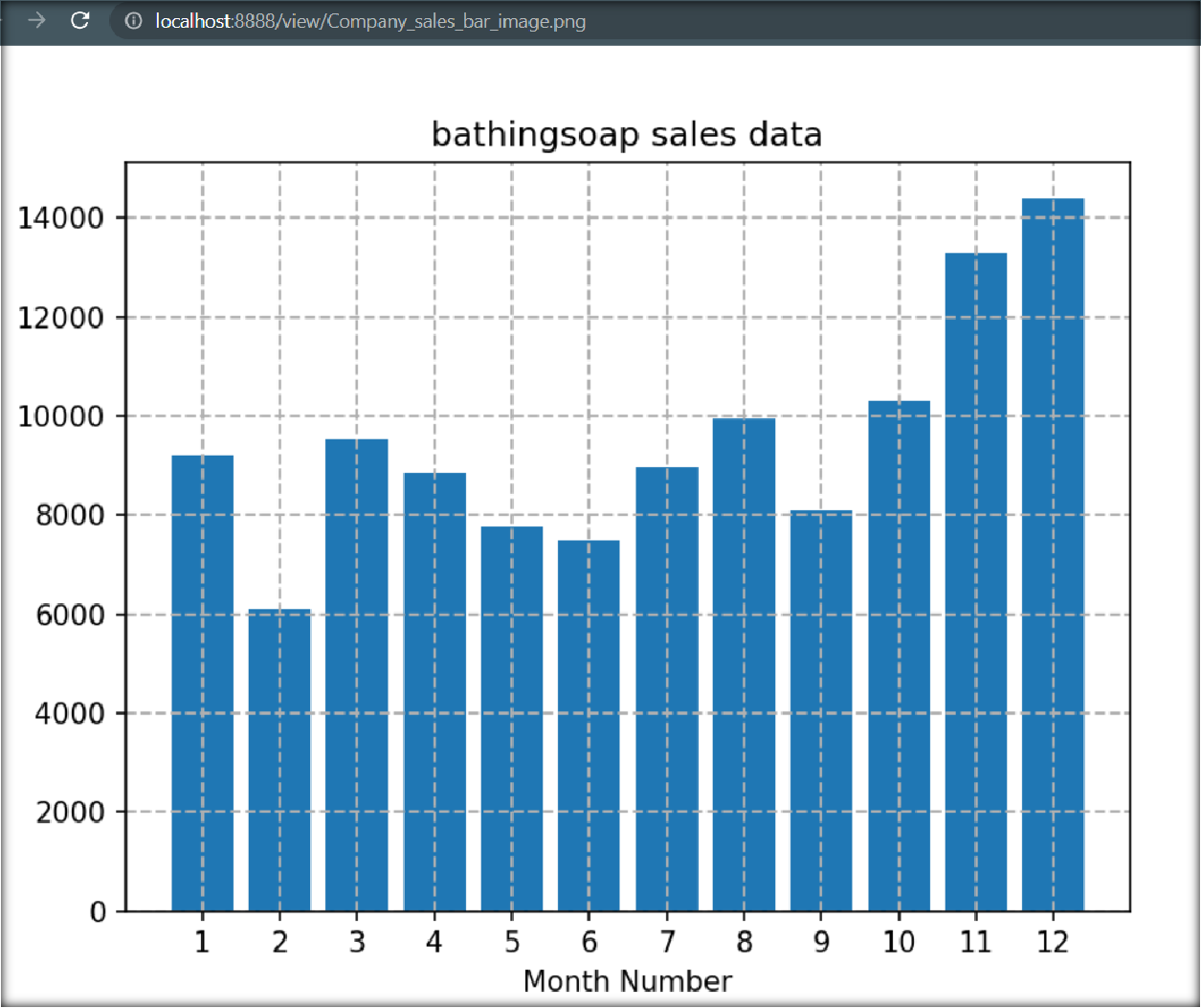
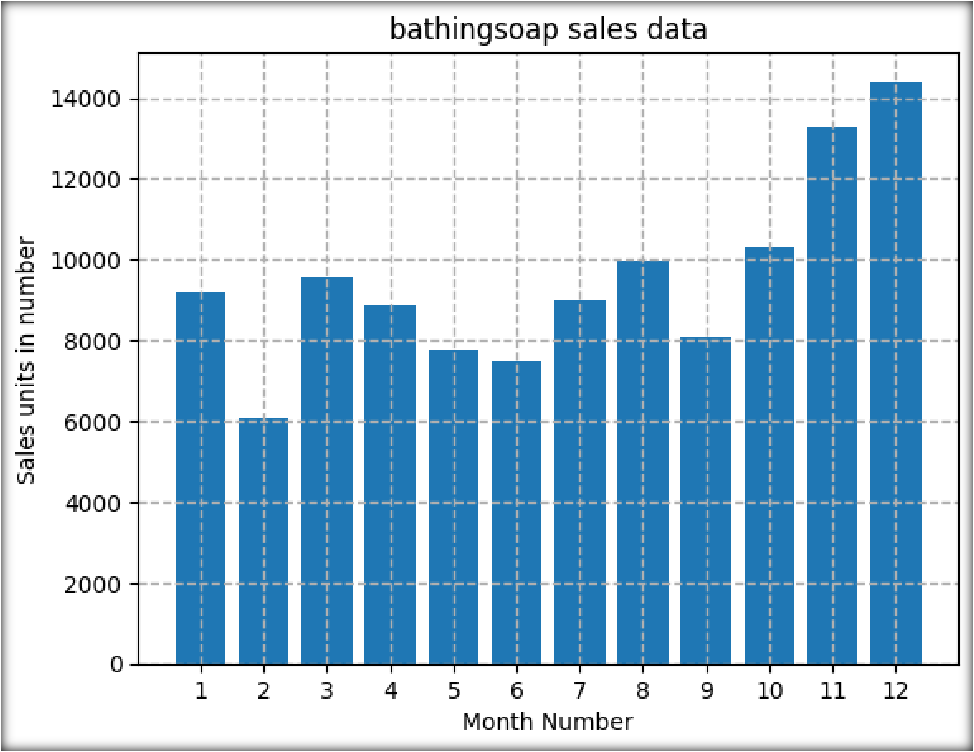
plt.xticks(monthList)

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()



Read the total profit of each month and show it using the histogram to see 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')

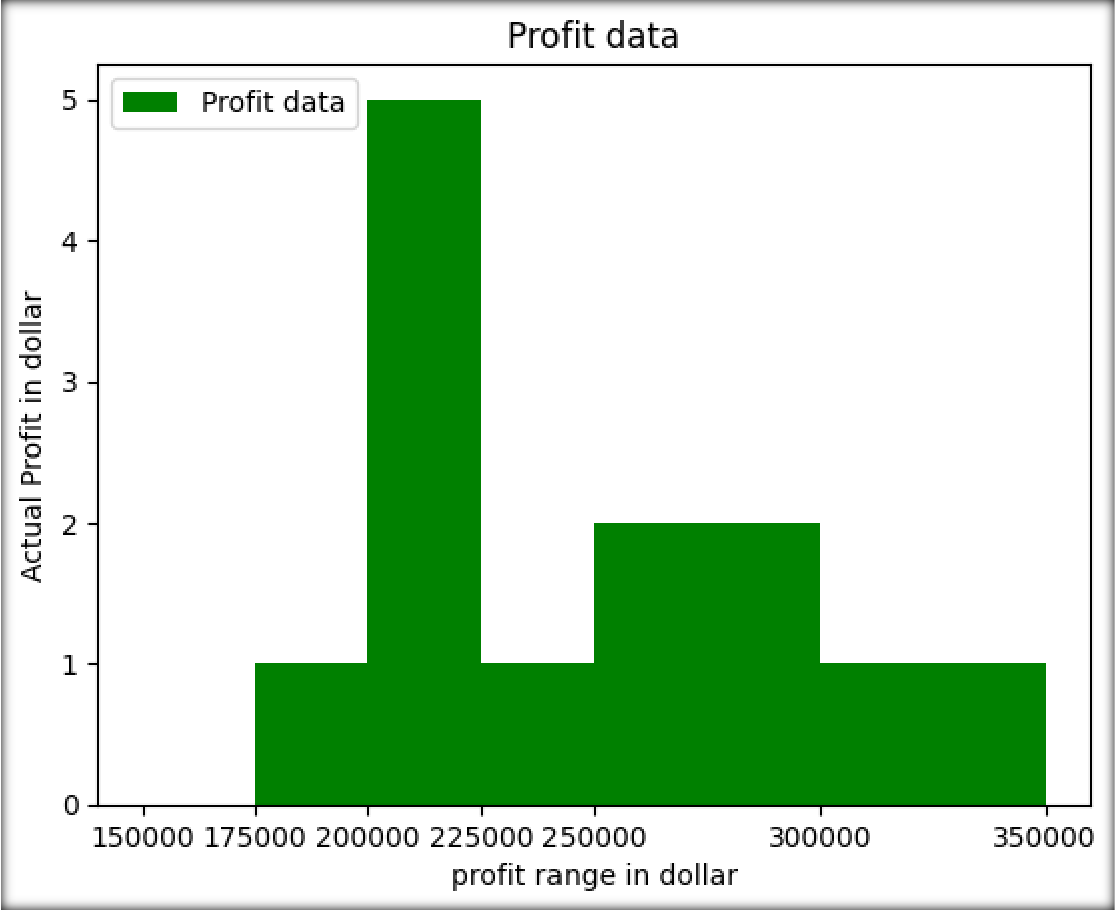
plt.ylabel('Actual Profit in dollar')

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 using a 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]

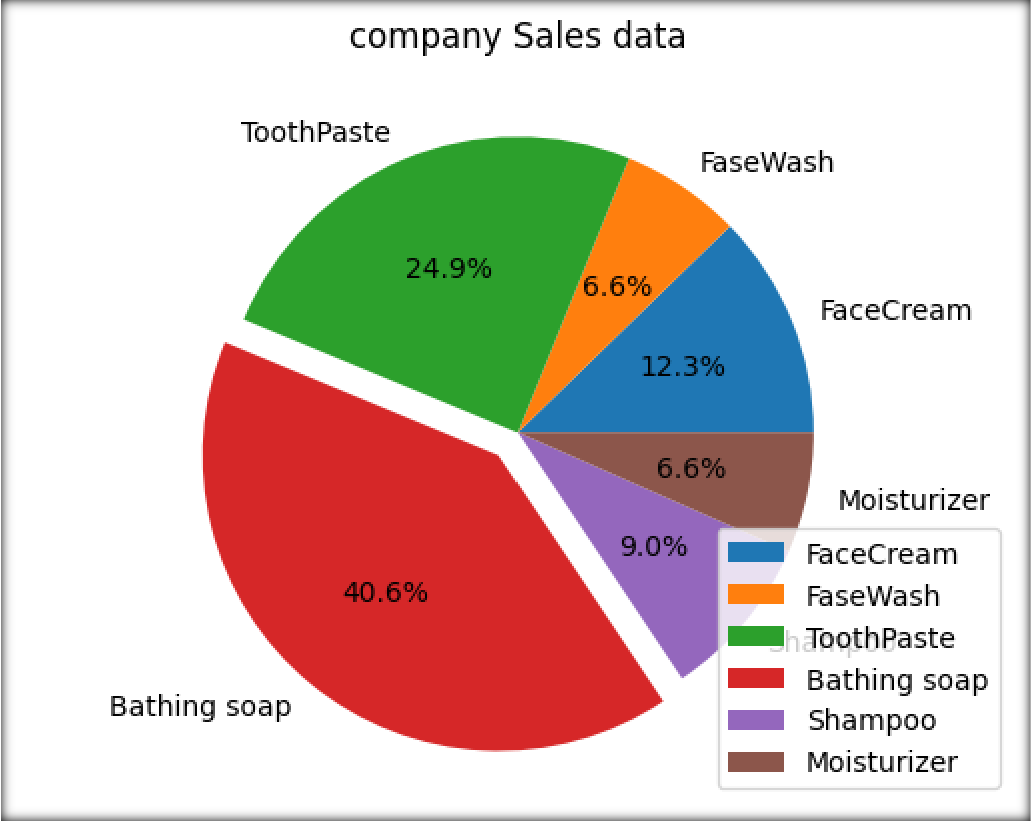
plt.axis("equal")

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 the Subplot.

### 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:

