UNIT-3

Page No.
Date

~~df = pd.series~~

```
d1 = {'a': 100, 'b': 200, 'c': 300, 'd': 400, 'e': 500}
print ('Original', d1)
new_se = pd.series(d1)
print('New series', new_se)
```

Julya, Python, R

↑

Jupyter

Jupyter notebook is an open source, web based interactive environment which allows to create and share documents that contain live ~~post~~ code, mathematical eq^n, graphics, maps, plots, visualizations, and narrative text. It integrates with many programming languages like Python, PHP, R, C# etc

Advantages

1. All in one place - Since Jupyter is an open source web based interactive environment that combines code, text, images, videos, mathematical eq^n, plots, maps, GUI and widgets to a single document.
2. Easy to convert - Jupyter notebook allow users to convert the notebook into other formats such as HTML & PDF. It also ~~of~~ has online tools and which allows you to render a publically available notebook in the browser directly.
3. Easy to share - JN are shared in JSON format which make them easily shareable

4. Language Independent :- It is platform independent beos it is represented as JSON format which is a language independent text based file format, it can also be converted to any file format such as marked down, HTML, pdf etc
5. Interactive Code - It uses ipywidgets packages which provides many common user interfaces for exploring code and data interactively.

Disadvantages

1. It is very hard to test long asynchronous tasks.
2. Less security
3. It runs cell out of order. code correction.
4. No IDE integration, No Unting and no code style correction.

Components of Jypyter Notebook

1. Notebook web Application - It allows users to edit code in the browser with automatic syntax highlighting & indentation.
   - Run code on the browser
   - See results of computation with media representation such as HTML, Png, pdf, latex etc
   - Create and use Js widgets
2. Kernels - Kernels are the seperate processes started by the notebook web application i.e used to run a users code in the given language & return output to the notebook web app.

In JN Kernel is available in following lang. Python, Julia, Ruby, R, Scala, Go, NodeJS.

3 Notebook Documents - Contains a rep. of all contents which is visible in the notebook web application, including input and output computations, text, mathematical eqn, graphs, images.

Types of cells in JN

1. Code cell
2. Mark down cell → For documentation
   ① Bold & Italics (font styling)
   ↳ __, ** ↳ _, *
   ② Headers
   # H1 → H6

#Heading1 ##Heading2
<h1> Header 1 </h1>
<h6> Header 6 </h6>

③ Ordered list
↳ 1 A
1 A 1
2 A 2
2 B
1 B 1
2 B 2

④ Bullet list
-, *
- A
* A.1
- B
* B1

:A
■A
•B
■B.

Hyper Links

[Google] (www.google.com)

Table content

Markdown cell allows you to create a table using pipe symbol(|) and dash symbol(-)
| → column
- → rows

Ex:-
RollNo	Name	Age
01	Jay	18
02	Amit	19

RollNo	Name	Age
01	Jay	18
02	Amit	19

Page No.
Date

Images

To insert a image in a markdown cell you first need to insert the image in the same directory and for this go to Jupyter dashboard and select upload, and then specify the path of an image and click on open. after that go to your current notebook and type the following code ~~<img src=~~
<img src = "Image.png">

3. Raw NB Convert cell

It provides a place where you can write output directly. These cells are not evaluated by the notebook Kernel.

4. Heading Cell

The JN does not support heading cell and simply shows the pop-up.

Qus1 Diff b/w Ipython, Jupyter, Python

Qus2 List out any 5 companies that integrate with Ipython and for Jupyter

Qus3 what tools integrate with Ipython & with Jupyter.

Ans 1

Ipython	python	Jupyter
• Ipython is an interactive command-line terminal for python. • It is an enhanced version of python shell.	Python is a general-purpose programming language. i.e widely used for Web dev. AI, DS.	Jupyter is a web-based interactive computing environment that supports several prog. lang. and share doc containing code, eq, visualization text etc

Page No.
Date

Ans 2 Ipython – Trivago, Raff, Frappe, wonderlut AI, Explor
Jupyter – MS Azure, Google colaboratory, Uber, ~~Amazon~~
Amazon sage Maker, Kbr, Nscorp

Ans 3 Tools – Jupyter → Jupyter contrib nb extensions,
Magic functions, widgets, Nbdime, nbga,

Ipython – matplotlib, numpy, pandas, scipy, seabor

Understanding the Tools

Magic Commands → Current session details

%quickref] to tell how to obtain additional
help info. about –
object ?
?
1. Using ipython to perform usefull tasks
2. Obtaining info. about magic fun"
3. Learning about python prog. Language
4. Discovering facts about the packages, objects, methods that you use in python to interact with data.

Ipython provides a standard feature called as cls (clear screen) which is not present in Py consol

Changing window Appearance

2. Option:
a Fonts:
3 Layout:
4 Colors: % colors

? → Interactive help.

?? → Full details

Page No. _____
Date

Getting Python Help

Modules → Help mode → Full details
↳ Interactive Help mode → Particular detail of obj.

We use help mode when you want to explore the lang. when we specifically need help with an object

There are certain commands to obtain a listing of some other topics

1. Modules – Current session list
2. Keywords – List of all keywords
3. Topics –

Using Ipython object help

If you have a list called mylist then just by typing mylist? we can discover obj. type, content, length.

Ques :- what do you understand by Raw NB convert

%pdoc
%pdef
%psource
%pfile
%pinfo
%pinfo2

plt.show().

Page No.
Date

Performing multimedia & Graphic Integration

→ Loading from online sites

`%.load` http://_____.py

→ obtaining online graphics and multimedia

* Ipython.display

from Ipython.display import Image

Embed = Image(' ')

Embed

or

Softlinked = Image(url='_____')

↳ If the Image is ~~removed~~ changed from the website then it will change automatically.

To change Image format

from Ipython.display import set_matplotlib_formats

set_matplotlib_formats('pdf', 'svg')

~~To convert a csv~~

Page No. _____
Date

To create a csv file from Dataframe

1. Load & read csv
2. Dataframe to csv

```
import pandas as pd
df = pd.DataFrame([[1,2,3,4],[5,6,7,8],[9,10,11,12]], columns =
                  ['A1', 'B1', 'C1', 'D1'])
df.head()
df.to_csv('export.csv')
```

	A1	B1	C1	D1
0	1	2	3	4
1	5	6	2	8
2	9	10	4	12

or

```
df.to_csv(export.csv, index = False)
```

→ remove index 2

Pandas → Concat

```
dict 1 = {'Name' : ['A', 'B', 'C'],
          'Age' : [12, 18, 20],
          'Address' : ['XYZ1', 'ABC2', 'DEF3']
          'Qualification' : ['10th', '11th', '12th']
         }
dict 2 = {'Name'
         }
df = pd.DataFrame(dict 1, index = 0,1,2,3)
df1 = pd.Dataframe(dict 2, index = 4,5,6,7)
frames = [df, df1]
res1 = pd.concat(frames)
print(res1)
```

Page No.
Date

Use of Matplotlib

```
From matplotlib import pyplot as plt
plt.plot([1, 2, 3], [3, 5, 7])
plt.show()

plt.title('Today Lab')
plt.ylabel('y-axis')
plt.xlabel('x-axis')
```

Ques. 2 Draw a graph containing comparison of 2 lines in a single plot and also perform the following.

① change the style of the plot.
ggplot()

② Add the labels on the graph (using legend function)
legend()

```
from matplotlib. import pyplot as plt
import numpy as np
from matplotlib import style
data = np.random.randn(50)
plt.style.use('ggplot')
plt.plot(data, linestyle = ':', linewidth = 2)
```

Page No.
Date

Magic Commands :- are special commands that can help you with running & analysing data in your notebook.

They add a special functionality i.e. not straightforward to achieve with python code or Jupyter notebook interface.

1. Line Magic - It is rep. using %. prefix and operates on a particular line of input.
It is used in the form of exp. and their return value can be assigned to a variable.

2. Cell Magic - It is rep. using %%. prefix and works on a complete cell or multiple lines of input. They receive the whole block as a string

#

Built-In Magic commands

1. %. autocall :- make fun callable without having to type parentheses
2. %. automagic
3. %. automagic
4. %. cd
5. %. dhist
6. %. edit
7. %. env
8. %. gui
9. %. lsmagic
10. %. notebook
11. %. who
12. %. pwd
13. %. recall
14. %. matplotlib -- list
15. %. time
16. %. quickref
17. %. load

Magic functions

1 % autocall :- Make fun" callable without having to type parentheses.

Ex
```
def add(x,y):
    return x+y
% autocall
add 3,5   → Output ----→ add(3,5)
                   ----→ add(3,5)
```

8

2. % automagic :- Make magic fun" callable without having to type the Initial %.

Ex:-
```
% automagic
cd, dhist, ls etc.
```

3 % cd :- change the current working directory
% cd ..

4 % dhist :- Print your history of visited directories

5 % env :- Give a list of environment variables

6 % run :- Run the named file inside IPython as a progra
% edit :- Open editor to execute code with history of current session.

Ex:-
```
run hello.py
edit hello.py
```

7 % lsmagic :- List currently available magic functions.

8 % matplotlib --list :- Available matplotlib backends ('tk','gtk', 'svg', 'pdf'] etc

9 % who :- Print all interactive variables, with some minimal formatting.

10 % pwd :- Return the current working directory path

To ... → %matplotlib inline
%matplotlib

Page No.
Date

%load

%load :- load code into current frontend file.py
%quickref :- Show a quick reference sheet
%recall :- Repeat a command, or get a command to input line for editing Eg:- recall 45 → line no.
%notebook :- This fun' converts current IPython history into an Ipython notebook file with ipynb extension
Eg:- %notebook filename

%time :- Time execution of a Python statement or expression.

[Data cleaning]

1 Empty cells → dropna()

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

① `df.dropna(inplace = True)` — To remove from original data

② `df.dropna()`

③ `print(df.to_string())`

④ fillna
0 or mean, median, mode.

⑤ `df = pd.read_csv('data.csv')`
`df.fillna(130, inplace = True)`

⑤ `x = df['column'].median()`
~~df.fillna~~ `df['column'].fillna(x, inplace = True)`

Page No.
Date

Data of Wrong Format

To date (Wrong format of date)

```
df['Data'] = pd.to_datetime(df['Date'])
print(df.to_string())
```

```
df = pd.read_csv('data.csv')
for x in df.index:
    if df.loc[x,'Duration'] > 120
        df.loc[x,'Duration'] = 120
```

* (df.duplicate())

Working with real Data

files 1. colors.txt , 3. Values.xls
2. titanic.csv , 4. XMLData.xml

→ We have to access data in a no. of forms and locations

Eg:- Memory streams rep. a form of data storage that your computer support natively, flat file exist on your hard drive, relational databases usually appear on networks. (although smaller relational data bases like that of MS access on a H.D), and web based data from the Internet

Uploading, Streaming & Sampling Data

DS call the columns in a database features/variables.
The rows are called as cases. Each row rep. a collection of variables that you can analyse.

Uploading

1. Small amount

colors.txt

color	val
Red	1
Blue	2

```
with open("colors.txt",'r') as open_file:
    print("Colors.txt Content:\n", + open_file.read())
```

In this the entire data set is loaded from the Memory into free memory.

Streaming

Some data sets are usually large such that you wouldn't be able to fit them entirely in memory at one time. There are also cases when some data sets load slowly bcoz they reside on a remote site.
Streaming answers both needs by making it possible to work with the data a little at a time.

Note:- You download individual pieces, making it possible to work with just part of the data and to work with it as you receive it, rather than the entire data set to download.

```
with open ("colors.txt",'r') as open_file:
    for observation in open_file:
        print("Colors.txt content:\n" + observation)
```

- Python streams each records from the source. This means that you must perform a read for each record you want

Sampling

Data streaming obtains all the records from a data source but some times you don't need all the records, and in such cases you can save time & resources by sampling the data. This means retrieving records a set no. of records apart such as every 5th record or by making random samples.

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

* enumerate fun" is use to retrieve a row no. when j%n == 0 the row will be kept and the application outputs~~read~~ the row to Otherwise it will discard the value.

read_table → reads tabular data and creates a data frame

Parsing → breaking down of elements

Parsing → Dividing the data into blocks

Interpreting the code or data

Page No.
Date

Random Sampling

Values b/w → 0 and 1

```
from random import random
sample_size = 0.25
with open("colors.txt", 'rb') as open_file:
    for j, observation in enumerate(open_file):
        if random() <= sample_size:
            print('Reading line:' + str(j) +
                  ' content:' + observation)
```

Accessing Data in structured flat file form

→ A flat file presents the easiest kind of file to work with the data appears as a simple list of entries that you can read one at a time in memory.

→ Classes and methods in the Pandas library parses the flat file data to make it easier to manipulate.

→ A text file generally treats or data as strings so that one needs to convert numeric data in other forms.

→ A CSV files provide more formatting and more info. but it requires more effort to read this file.

Parsing

1. Reading from a text file

```
import pandas as pd
color_table = pd.io.parsers.read_table("colors.txt")
print(color_table)
```

Within Pandas there is a set of parsers, code used to read individual bits of data and determine

the purpose of each bit acc. to the format of the entire file.

No Parsers interprets the 1st row as field names.

2. Reading CSV delimiter file
   1. Header, fields, records, strings, integers, real no.

```
titanic = pd.io.parsers.read_table('titanic.csv
X = titanic[['age']]
print(X)
```

titanic = pd.io.parsers.read_table('titanic.csv', sep=',')
↳

Reading an excel and other MS applications - highly formatted content. One can specify every aspect of info. these files contains.

```
xls = pd.ExcelFile("value.xls")
fig_values = xls.parse("Sheet 1", index_col=name,
                       na_values=['na'])
print('fig values')
```

Sending Data in an Unstructured file format

- Unstructured data files consist of series of bits
- The file does not separate the bits from each other in any way.
- Unstructured file format rely on the file user to know how to interpret the data.

Eg:- Each pixel of a picture file could consist of 3

fields of 32 bits each knowing this depends upon the data scientist. However the beginning of the file may provide clues about interpreting the file.

Install → pip install ~~skimage~~ scikit-image

The imshow() function of matplotlib performs the rendering and uses a grey scale color map.
The show() fun' is used to just display the image for you.

The image is actually an array of pixels that can be manipulated in diff ways for example cropping of image (the purpose of cropping the image is to make it of a specific size both images must be of the same size for analysing purpose.
cropping is one way to ensure that the images are the correct size for analysis.
→ Another way to change the image size is to resize it.
→ Once you have the images of right size you need to flatten them. A dataset now is always a single dimension, not 2-dim.
→ The type of this array is numpy.ndarray you can add this array to a data set and then use that dataset for analysis purpose

Managing Data from Relational Database

The goal of db manager is to make data easy to manipulate the focus of most data storage is to make data easy to retrieve.

SQL shapes data in a particular way so as to ease ~~the~~ eliminate the shaping procedure. Creating a connection to a db is a complex procedure.

Once you have access to an engine you can use that engine to perform task specific to a DBMS. The output of a read method is always a dataframe object that contains the requested data.

1. read_sql_table() - Reads data from SQL table to a ~~read_sqlt~~ dataframe object.
2. read_sql_query() - Reads data from a database using a SQL query to a dataframe object
3. read_sql() - Read data either from a table/query
4. DataFrame.to_sql() - write the content of a DataFrame object to the specified table in the database.

Sqlalchemy

↳ Sqlite, MySQL, ~~PostSQL~~ PostgreSQL, SQL server, ODBC

Page No.
Date

Interacting with data from NoSQL db
MongoDB → PyMongo, MongoClient
This class is use to create the required engine.

The MongoDB engine relies on the find() function to locate the data

Accessing Data From the Web
Other service is a kind of web application that provides a means to ask questions and receive answers. Web services usually host a no. of input types.
There is a service called as micro-service (query system) that has a specific focus and provide only one specific query input and output.
Working with Web services and micro services means working with .xml.

```
<MyDataset>
    <Record>
        <Number> 1 </Number>
        <String> Welcome </String>
        <Boolean> True </Boolean>
    </Record>
        |
        |
        |
</MyDataset>
```

Visualisation

Page No.
Date

Performing data analysis is basically about the communication because unless you communicate your ideas, the art of obtaining, shaping and analysing data has a little value beyond individual personal needs.

- Matplotlib in comp to matlab is relatively easy bcoz they use same sort of state machine to perform task and they have similar method of defining graphical elements.

Starting with Graph

A Graph or chart is simply a visual rep of numeric data.

1. Defining the plot.
2. Defining multiple lines to plot
3. Saving your work

Setting the Axis Ticks, Grids

→ The use of Axes (), Ticks () and Grids () make it possible to illustrate graphically the relative size of data elements so that the viewer gains an appreciation of comparative measure

1. Setting the axis
2. Formatting the axis
3. Adding Grid.

Page No. _____
Date

Matplotlib

Matplotlib is a low level graph plotting library that serves as a visualization utility.

1. Matplotlib Version
2. Matplotlib Installation
3. Use of Pyplot sub module
4. Plotting
   - Use of plot function
   - plotting without line
   - Multiple points
   - Default x points
5. Matplotlib Markers
   - Use of marker size ('MS')
   - Marker color ('MEC')
   - Marker face color ('MFC')
   - Format strings ('FMT') (Marker/Line/Color)
6. Matplotlib Line
   - Types of line reference (Default, dotted, dashed, dashdot)
   - Line style ('LS') . Line color ('c')
   - Line width ('LW') . Multiple lines
7. Matplotlib Labes and titles
   - xlabel, ylabel, title, font properties for title and label (using fontdict parameter)
   - Positioning the title (loc)
8. Add grid lines
   2. using grid function
   a. x-axis, y-axis grid 3. color of grid, 4. line properties (color, line style, line-width)

Subplots()
To draw multiple plots in 1 figure.

Add title to each plot
Supertitle

Practice Questions - 1

Q1 to 2 (1)
```
dict = {
        'Name' : ['Raman', 'Sheetal', 'Vikas', 'Shiv'],
        'Age' : [18, 17, 15, 20],
        'Sex' : ['Male', 'Female', 'Male', 'Female']
       }
import pandas as pd
import numpy as np
```

(2)
```
df = pd.DataFrame(dict)
df.columns
```
Output `['Name', 'Age', 'Sex']`

(3) Sort the df based on numerical column

~~`df['Age'].sort()`~~

```
df.sort_values('Age')
```
ascending = False

(4) Mean
```
men = df['Age'].mean()
```

(5)
```
df['Name'].plot(men)
```

⑥ nl = df.columns[df.isnull().sum() > 4]
df.dropna(nl)

⑦

	col-A	col-B
0	A	1
1	B	2
2	C	3
3	D	4

→ data.csv

df.insert(1, 'col-c', Data)
idx = 1
val = [2, 4, 6, 8]
df.insert(loc = idx, column = 'col-c', value = val)

output

	col-A	col-c	col-B
0	A	2	1
1	B	4	2
2	C	6	3
3	D	8	4

⑧ df.select_dtypes(include = ['Bool']).columns

⑨ df.count() → No. of non-Nan values in each columns
df.sum(df.count()) → sum of total count values which are not NAN

⑩ df.df_split = np.array_split(df, 2)
for df in df_split:
print(df)

⑪ Reversing Row wise
df.loc[::-1]

	col-B	col-C	col-A
3	4	8	D
2	3	6	C
1	2	4	B
0	1	2	A

Reversing column wise
df[df.columns[::-1]]

	col-B	col-C	col-A
0	1	2	A
1	2	4	B
2	3	6	C
3	4	8	D

Page No.
Date

Q1

	Python	Matlab
Purpose	General Purpose programming language	Numerical computing and technical computing environment
Syntax	uses Identation and is more readable.	uses traditional programming syntax
Data types	Supports dynamic typing and a wide range of data types.	Provides built-in support for numerical and matrix operations.
Libraries	Large ecosystem of libraries and packages are available	Rich set of toolboxes for various scientific applications.
Performance	Interpreted language, which can be slower for certain tasks	Compiled language generally faster for numerical computations.
Community support	Large community support	Smaller community support.
Applications	Web development, data analysis, ML etc.	Mathematical, engineering and scientific research

Q2

	Matplotlib	MATLAB
Purpose.	Python library for data visualization	• Integrated plotting and visualization in MATLAB
3-D visualization.	Supports 3D plotting and visualization	• Offers built-in 3D plotting capabilities.
Integration	Can be used with other Python libraries and tools	• Designed to work seamlessly within MATLAB.

Uses Python Syntax	Uses MATLAB Syntax
Open-source and free to use.	Commercial software with license fees.
Data analysis, scientific plotting, visualization.	Mathematics, engineering, and scientific research.
flatten()	ravel()
Returns a copy of the array as a flattened 1D array.	• Returns a view (if possible) or a copy of the array.
Allocates → Returns a new array.	• Returns a view whenever possible, avoids memory allocation.
May be slower due to memory allocation.	• Can be faster since it avoids memory allocation.
Does not modify the original array.	• Does not modify the original array.
Returns a 1D array regardless of the original shape.	• Preserves the shape of the original array.

Note 2

* The "flatten()" fun" always returns a copy of the array as a flattened 1D array, regardless of the original shape.
  - It guarantees a new memory allocation and returns a contiguous array.

* The 'ravel()' returns a view of the original array whenever possible, avoiding memory allocation.
  - It preserves the order of elements as stored in memory and maintains the shape of the original array.

Page No.
Date

	axis	axes
Fun^n.	Controls the limits and scaling of the plot's data.	• Represents the plotting area on a single subplot.
usage	Applied to an existing plot or subplot	• Used to create and manipulate plots & subplots.
control	Controls the axis limits, labels and tick marks.	• Provides methods to add and customize plot elements
Plots Affected	Affects the entire plot or subplot.	• Operates on a specific subplot or plotting area.
Returns	Returns the current axis limits and scaling.	• Returns a reference to the created 'axes' object
Multiple subplots	Affects all subplots if applied to the entire figure	• Associated with a specific subplot within the figure
Ex:-	plt.axis([xmin, xmax, ymin, ymax])	fig, ax = plt.subplots()

Note :- **Axes** is the region where the plots and graphs are drawn.
Each axes has its own title, x-axis and y-axis

Axis in matplotlib refers to the scale of the graph. A two-D graph has an x-axis and y-axis, whereas a 3-D graph as x, y and z-axis.

Defining the Line Appearance
1. Working with line style
2. Using colors
3. Adding markers.

#

Page No.
Date

Using label, Annotation and Legends

Label - It's purpose is to make it easy for the viewer to know the name or kind of data that is to be illustrated.

Annotation :- It augments the info. the viewer can immediately see about the data with notes, sources or any other useful info. In contrast to labels the purpose of annotation is to help extends the viewers knowledge of the data, rather than simply identify it.

Legends

Presents a listing of the data groups within the graph and often provide clues to make identification of the data group easier.

= "%d.%.1f%%", y,

Page No.
Date

Practice Question 2

Q1 Read total profit of all months and show in usable line plot.

```
plt.plot(total_profit)
```

Q2 ~~Legend~~ Styling

Q1 ?

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

df = pd.read("____")
df
```

Q1 ?

```
plt.plot(df['total_profit']), plt.title('Total Profit')
plt.legend('total_profit')
plt.show()
```

Q2 ?

```
plt.plot('df['total_profit']', c='red', ls='--', marker='x')
plt.title('Total Profit')
```

```
plt.plot(df['total_profit'], 'o:r', c='red', mfc='g', lw=3)
plt.xlabel('Month Number')
plt.ylabel('Sold Units number')
plt.legend(loc=4)
```

Q3 ?

Page No.
Date

Q4
```
toothpastesales = df['toothpaste']
mouth_no = df['mouth_number']
plt.scatter(toothpastesales, mouth_no)
plt.xlabel("Toothpaste sales")
plt.ylabel("mouth-No.")
plt.title("Toothpaste sales Data")
plt.show()
```

Q5
```
plt.bar(df['mouth-no'], df['face-cream'])
plt.show()
plt.xlabel('mouth')
plt.ylabel('face cream')
```

Q6
~~plt.plot~~ ~~plt.bar~~
```
plt.pie(df['Bathing-soap'], autopct='%1.1f%%')
plt.savefig('Bathing-soap.png')
plt.show()
```

Q7
```
plt.hist(df['total-profit'], bins=10, edgecolor='red')
plt.show()
```

Q8
```
Total_sales = df['total_units'].sum()
plt.pie(Total_sales, autopct="%1.1f%%")
plt.show()
```

Q9
```
plt.subplot(2,1,1)
plt.plot(df['Bathingsoap'],
         df['months'])
```

Page No.
Date

Q1 What is faker library? What it is use for. Generate a dataset using faker library to create a dataset of 100 fake names and email addresses

Q2 How to plot multiple bar charts using matplotlib in python.

Q3 Explain box plot in statistics

Ans 1 Faker library generates fake (but reasonable) data for you which can be used for various testing and building applications. It is commonly used in software development testing and data analysis to generate realistic-looking but fictional data

Usage = It is use to generate fake data in a wide range of categories, including names, addresses, phone numbers, email addresses, job titles, credit card numbers, date and more. You can customize the data generation to match specific requirements or generate random data by default

```
From faker import Faker
fake = Faker()
dataset = []
for _ in range(100):
    name = fake.name()
    email = fake.email()
    dataset.append([name, email])
for name, email in dataset:
    print("Name:", name)
    print("Email:", email)
```

A box plot, also known as a box-and-whisker plot, is a statistical visualization tool that provides a concise summary of dataset's distribution. It displays the median, quartiles, and potential outliers of the data in a graphical format. The plot consists of a rectangular box and two lines, which resembles a box with whiskers.

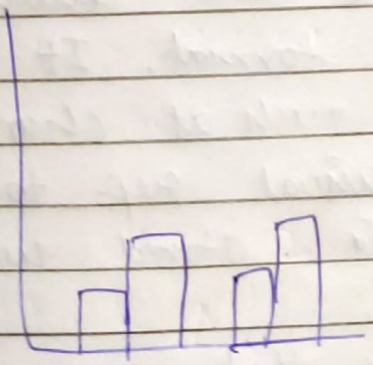
```
import numpy as np

x1 = [2.5, 3.5, 4.5, 5.5, 6.5, 7.5]
y1 = [3, 6, 9, 12, 15, 18]

x2 = [5, 10, 15, 20, 25, 30]
y2 = [2.6, 9.5, 14, 12, 8.3, 18]

r = np.arange(1)
width = 0.4

plt.bar(r, x1, width = width)
plt.bar(r + width, y1, width = width)
```



Page No.
Date

AMS Azure

Q1 write a short note on diff. customization options available with any plot.

Q2 Explain Data Visualization

Q3 Write an Example of data comparison where we can use the Scatter plot.

Q4 What is open data. Name any 2 websites from which we can download open data.

Ans 1 > Plotting libraries such as Matplotlib and seaborn offers various customization options to enhance the visual rep. of data.

1. Color and Style
2. Axes labels and Titles
3. Legends and Annotations
4. Plot Size and layout.
5. Axis Ticks and Tick Labels
6. Grid lines
7. Plot Annotations

Ans 2 > DV in DS refers to the process of rep. and displaying the structure and content of a database in a visual format. It involves using visual elements such as charts, graphs, diagrams and other visual rep. to gain insights into the data stored within a database.

Matplotlib :- A plotting library that provides a MATLAB-like interface for creating a wide range of static, animated and interactive visualizations.

Page No.
Date

* Features
→ Easy to use. → Versatile plots → Customization

* Limitations
→ Limited 3D plotting
→ Lack of interactivity

Ans 3
```
import matplotlib.pyplot as plt
heights = [63, 68, 70, 61, 72, 66, 64, 69, 67, 71]
weights = [150, 180, 200, 135, 210, 160, 145, 175, 185, 190]

plt.scatter(heights, weights)
plt.xlabel('Height')
plt.ylabel('Weight')
plt.title('Height VS Weight')
plt.show()
```

Ans 4 Open data refers to data that is freely available for anyone to access, use and share provided by govt, org" with intention of promoting transparency.

Eg- data.gov, Kaggle, opendata portal by European Union.