GUJARAT TECHNOLOGICAL UNIVERSITY

BE 5 SEMESTER – OLD PAPER – S22 TO S25 – ANSWER BANK

Subject Name & Code: Python for Data Science - 3150713

Chapter 1: Overview of Python and Data Structures

Repeated Questions:

1. Python Features and Basics (Theory):

- **Q.** Why Python is popular / List advantages of Python. **(SUMMER 2022, 3 Marks; WINTER 2023, 3 Marks)**

Answer:

- **Easy to learn and read:** Simple English-like syntax; less code for same task.
- **Interpreted and interactive:** Runs line-by-line; useful for quick testing and Jupyter/REPL.
- **Large standard library and ecosystem:** Built-in modules + third-party libraries like NumPy, Pandas, Matplotlib.
- **Portable and versatile:** Works on Windows/Linux/macOS; supports scripting, web, data science, automation, AI.

- **Q.** List and explain different programming/coding styles supported by Python. **(SUMMER 2022, 4 Marks; SUMMER 2024, 3 Marks)**

Answer:

- **Procedural style:** Code written as sequence of functions/statements; example: def add(a,b): return a+b.
- **Object-Oriented style:** Uses classes and objects; example: class Student: ....
- **Functional style:** Uses functions as first-class objects, map(), filter(), lambda, comprehensions.
- **Interactive/scripted/modular style:** Can run in REPL/Jupyter cell-by-cell, as .py scripts, or as modules/packages.

- **Q.** What is the use of magic functions in Python/Jupyter? Explain with example. **(WINTER 2022, 3 Marks OR; WINTER 2023, 4 Marks)**

Answer:

- Magic functions are special IPython/Jupyter commands prefixed with % or %%.
- **Line magic:** acts on one line: %timeit, %matplotlib inline.
- **Cell magic:** acts on whole cell: %%time, %%writefile.
- **Use:** timing code, loading files, plotting setup, running shell commands.
- **Example:** %timeit sum(range(1000)) measures execution time.

---

2. Data Structures and Strings (Theory):

- **Q.** Explain/List different data structures used in Python and explain any three/four with suitable examples. **(WINTER 2022, 7 Marks; WINTER 2023, 7 Marks)**

Answer:

- **List:** Ordered, mutable, allows duplicates, indexed. Example: L = [10, 20, 30]; L[0] gives 10.
- **Tuple:** Ordered, immutable, allows duplicates. Example: T = (1, 2, 3); cannot change T[0].
- **Set:** Unordered, mutable, unique elements only, no indexing. Example: S = {1, 2, 2, 3} gives {1,2,3}.

- **Dictionary:** Key-value pairs, mutable, keys unique and hashable. Example: D = {"name": "Ravi", "age": 20}; access D["name"].
- **String:** Immutable sequence of characters; supports indexing/slicing. Example: s = "Python".
- **Range:** Immutable sequence of numbers used in loops. Example: range(1, 5).
- **Comparison:** List/Tuple are sequence types; Set is collection of unique items; Dictionary stores mapped data.

- **Q.** Explain String Slicing in Python with example. **(SUMMER 2022, 3 Marks; WINTER 2024, 3 Marks)**

Answer:

- String slicing extracts a substring using s[start:stop:step]; stop is excluded.
- **Example:** s = "Python"; s[1:4] → "yth".
- Negative indices and reverse: s[-3:] → "hon"; s[::-1] → "nohtyP".

---

3. Built-in Functions:

- **Q.** Explain range() function of Python with suitable example. **(WINTER 2023, 3 Marks; WINTER 2024, 3 Marks; SUMMER 2025, 3 Marks)**

Answer:

- range() returns an immutable sequence of numbers: range(start, stop, step).
- stop is excluded; default start=0, step=1.
- **Example:** list(range(5)) → [0,1,2,3,4]; list(range(1,10,2)) → [1,3,5,7,9].

- **Q.** Write a Python program/code to print current date and time. **(SUMMER 2022, 3 Marks OR; WINTER 2023, 3 Marks OR)**

Answer:

```
import datetime
now = datetime.datetime.now()
print("Current date and time:", now)
```

---

4. Programs:

- **Q.** Write a Python program to display the Fibonacci sequence / Fibonacci series. **(SUMMER 2022, 7 Marks OR; WINTER 2022, 7 Marks; WINTER 2023, 7 Marks; SUMMER 2025, 7 Marks)**

Answer:

- **Given:** Number of terms n.
- **To Find:** Fibonacci sequence up to n terms.

Code:

```
n = int(input("Enter number of terms: "))
a, b = 0, 1
for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
```

- **Explanation:** Start with a=0, b=1; print a; update a=b, b=a+b.
- **Output for n=7:** 0 1 1 2 3 5 8

- **Q.** Write a Python code/program to read data from a text file. **(WINTER 2022, 7 Marks OR; SUMMER 2025, 3 Marks OR)**

Answer:

- **Given:** Text file data.txt.
- **To Find:** Read and display file contents.

Code:

```
with open("data.txt", "r") as f:
    data = f.read()
print(data)
```

- **Explanation:** open() opens file in read mode; read() reads entire content; with closes file automatically.

Alternative line-by-line:

```
with open("data.txt", "r") as f:
    for line in f:
        print(line.strip())
```

Other Important Questions:

1. Python Fundamentals (Theory):

- **Q.** Differentiate: C and Python. **(WINTER 2022, 3 Marks)**

Answer:

Feature	C	Python
Execution	Compiled	Interpreted
Typing	Static, explicit types	Dynamic, inferred types
Syntax	More complex, braces/semicolons	Simple, indentation-based
Use	System/low-level programming	Data science, scripting, AI, web

---

- **Q.** Explain the input function of Python that demonstrates type casting. **(WINTER 2022, 4 Marks)**

Answer:

- input() always returns a string.
- To use numeric input, cast using int(), float(), etc.

Example:

```
age = int(input("Enter age: "))
marks = float(input("Enter marks: "))
print(age + 1, marks + 0.5)
```

Here int() and float() convert string input into numeric types; without casting, arithmetic may fail or concatenate.

- **Q.** How to format Date and Time in Python? Explain with example. **(WINTER 2022, 4 Marks)**

Answer:

- Use datetime.strftime(format) to format date/time.
- Common codes: %d day, %m month, %Y year, %H hour, %M minute, %S second.

Example:

```
import datetime
now = datetime.datetime.now()
print(now.strftime("%d-%m-%Y %H:%M:%S"))
```

Output example: 20-11-2026 14:35:10.

- **Q.** Write any four Python commands that demonstrate Python as calculator. **(WINTER 2023, 4 Marks)**

Answer:

- print(10 + 5) → 15
- print(10 * 5) → 50
- print(10 / 4) → 2.5
- print(10 // 3) → 3
- Also: print(2 ** 5) → 32; print(10 % 3) → 1.

---

2. Data Structures (Theory):

- **Q.** Discuss characteristics of Set and Dictionary data structures in Python. **(SUMMER 2024, 3 Marks)**

Answer:

- **Set:** Unordered, mutable, unique elements only; no indexing; supports union/intersection.
- **Dictionary:** Stores key-value pairs; keys unique and hashable; values can be any type; mutable.
- Example Set: {1,2,3}; Example Dictionary: {"a":1, "b":2}.

- **Q.** Write a short note on List. **(SUMMER 2025, 4 Marks)**

Answer:

- List is an ordered, mutable collection written as [item1, item2, ...].
- Allows duplicate values and heterogeneous data types.
- Supports indexing, slicing, iteration, and methods like append(), remove(), sort().
- Example: L = [10, "Python", 2.5]; L.append(99) adds 99.

- **Q.** Differentiate List and Tuple in Python. **(WINTER 2024, 4 Marks)**

Answer:

Feature	List	Tuple
Syntax	[1,2,3]	(1,2,3)
Mutability	Mutable	Immutable
Speed/Use	Slightly slower; dynamic data	Faster; fixed data
Methods	Many add/remove methods	Fewer methods; used as records/keys

---

3. Functions:

- **Q.** Write a program to perform the following operations about Function: 1. Function with multiple return values 2. Default arguments. **(WINTER 2024, 4 Marks)**

Answer:

```
def calculate(a, b=10):
    return a + b, a * b   # multiple return values as tuple

sum_val, mul_val = calculate(5)      # default b=10
print("Sum:", sum_val, "Product:", mul_val)

sum_val, mul_val = calculate(5, 3)
print("Sum:", sum_val, "Product:", mul_val)
```

- **Explanation:** b=10 is a default argument; return a+b, a*b returns two values as a tuple.

---

4. Programs: Strings and Dictionaries:

- **Q.** Write a single line code to get the value of "type" from the given dictionary in such a way that it does not produce any error or exception even if any key from the dictionary is misspelled. Dictionary: {"batters": {"batter": [{"batter": [{"batter": [{"batter": [{"type": "Regular"}]}]}]}]}} **(SUMMER 2023, 3 Marks)**

Answer:

Dictionary:

```
d = {"batters": {"batter": [{"batter": [{"batter": [{"type": "Regular"}]}]}]}}
```

Single-line safe code:

```
value = d.get("batters", {}).get("batter", [{}])[0].get("batter", [{}])[0].get("batter",
[{}])[0].get("type", None)
print(value)   # Regular
```

- **Explanation:** Each .get(key, default) avoids KeyError; [0] selects first list element; final .get("type", None) returns None if misspelled.

- **Q.** Write a Python program to take two strings and position value (starting from zero) as input from the user. Insert the second string in the first string after the given position. **(SUMMER 2024, 7 Marks)**

Answer:

- **Given:** s1, s2, position pos (0-based).
- **To Find:** Insert s2 in s1 after pos.

Code:

```
s1 = input("Enter first string: ")
s2 = input("Enter second string: ")
pos = int(input("Enter position (starting from 0): "))

result = s1[:pos+1] + s2 + s1[pos+1:]
print("Result:", result)
```

- **Explanation:** s1[:pos+1] takes characters up to and including position pos; then s2; then remaining part s1[pos+1:].
- **Example:** s1="Hello", s2="XYZ", pos=1 → HeXYZllo.

- **Q.** Write a Python program to input the string from user and count number of vowels in given string. **(WINTER 2023, 3 Marks)**

Answer:

```
s = input("Enter a string: ")
count = 0
for ch in s.lower():
    if ch in "aeiou":
        count += 1
print("Number of vowels:", count)
```

---

5. Programs: Numbers and Patterns (Numerical):

- **Q.** Write a Python program to find the factorial of a natural number n. **(WINTER 2022, 7 Marks OR)**

Answer:

- **Given:** Natural number n.
- **To Find:** n! = 1 × 2 × 3 × ... × n.

Code:

```
n = int(input("Enter a natural number: "))
fact = 1
for i in range(1, n + 1):
    fact = fact * i
print("Factorial =", fact)
```

- **Example:** n=5 → Factorial = 120.

- **Q.** Write a program to check whether the given number is prime or not. **(SUMMER 2022, 7 Marks)**

Answer:

- **Given:** Integer n.
- **To Find:** Prime or not.

Code:

```
n = int(input("Enter a number: "))
if n < 2:
    print("Not Prime")
else:
    is_prime = True
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            is_prime = False
            break
    print("Prime" if is_prime else "Not Prime")
```

- **Explanation:** Check divisibility from 2 to √n; if divisible, not prime.

- **Q.** Write a program to print the following patterns. (1) * / * * / * * * / * * * *; (2) $ $ $ / $ $ / $; (3) # # # # / # # # / # # # / # # # / # # # **(SUMMER 2022, 7 Marks)**

Answer:

(1)

```
for i in range(1, 5):
    print("* " * i)
```

(2)

```
print("$ $")
print("$")
print("$")
```

(3)

```
for i in range(5):
    print("# # #")
```

- **Explanation:** Pattern 1 uses row count to repeat *; pattern 2 prints fixed lines; pattern 3 repeats # # # five times.

- **Q.** Write a program which takes 2 digits X, Y as input and generates a 2-dimensional array of size X * Y. The element value in the i-th row and j-th column should be i*j. **(SUMMER 2022, 7 Marks)**

Answer:

- **Given:** X rows, Y columns.
- **To Find:** 2D array with arr[i][j] = i*j.

Code:

```
X = int(input("Enter rows: "))
Y = int(input("Enter columns: "))

arr = [[i * j for j in range(Y)] for i in range(X)]
for row in arr:
    print(row)
```

- **Example:** X=3, Y=4 →

```
[0,0,0,0]
[0,1,2,3]
[0,2,4,6]
```

---

6. Programs: Lists, Tuples and Dictionaries:

- **Q.** Write a program to interchange the List elements on two positions entered by a user. **(SUMMER 2022, 4 Marks OR)**

Answer:

```
L = [10, 20, 30, 40, 50]
p1 = int(input("Enter first position: "))
p2 = int(input("Enter second position: "))

L[p1], L[p2] = L[p2], L[p1]
print("List after interchange:", L)
```

- **Explanation:** Python tuple unpacking swaps values at given indices.

- **Q.** Write a Python program to demonstrate Tuple, List and Dictionary. **(WINTER 2024, 4 Marks OR)**

Answer:

```
my_list = [10, 20, 30]
my_tuple = ("Python", 3.14, 2026)
my_dict = {"name": "Ravi", "age": 20}

print("List:", my_list)
print("Tuple:", my_tuple)
print("Dictionary:", my_dict)
print("Dictionary value:", my_dict["name"])
```

- **Explanation:** List is mutable [], tuple is immutable (), dictionary stores key-value pairs {}.

Chapter 2: Data Science and Python

Repeated Questions:

1. Data Science Pipeline and Role of Python:

- **Q.** Explain different stages/phases of the Data Science Pipeline / Explain data science pipeline in detail / How to create data science pipeline. **(WINTER 2022, 4 Marks OR; WINTER 2023, 4 Marks; SUMMER 2024, 7 Marks; WINTER 2024, 4 Marks; SUMMER 2025, 7 Marks)**

Answer:

- **1. Problem definition:** Define business/research goal and success criteria.
- **2. Data collection:** Gather data from CSV, databases, web scraping, APIs, images, etc.
- **3. Data cleaning/preprocessing:** Handle missing values, outliers, duplicates, type conversion, scaling.
- **4. Exploratory Data Analysis (EDA):** Summarize data using statistics and plots; find patterns, correlations.
- **5. Feature engineering/selection:** Create/select useful variables for modeling.
- **6. Modeling:** Apply ML/statistical algorithms using Scikit-learn etc.
- **7. Evaluation:** Check accuracy, precision, recall, RMSE, cross-validation; tune model.
- **8. Deployment & communication:** Deploy model/API/dashboard and present insights; pipeline is iterative.

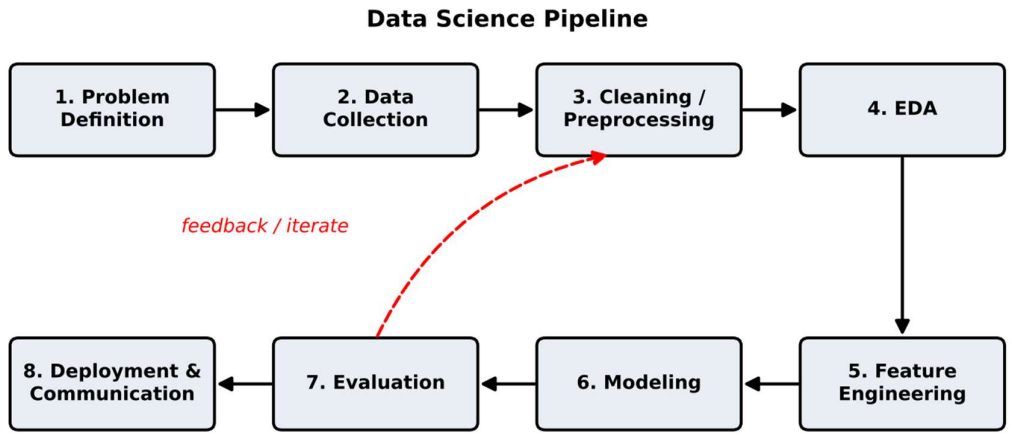


Fig. 1: Data Science Pipeline (stages 1–8 with feedback).

- **Q.** Explain the role/popularity/advantages of Python in Data Science. **(WINTER 2022, 3 Marks; SUMMER 2023, 3 Marks; SUMMER 2025, 4 Marks)**

Answer:

- **Simple syntax:** Easy to learn and write; faster prototyping.
- **Rich ecosystem:** NumPy, Pandas, Matplotlib, Scikit-learn, SciPy, Seaborn.
- **Integration & portability:** Works with databases, web APIs, C/C++, Java; cross-platform.
- **Community & reproducibility:** Large community, Jupyter notebooks, open-source libraries.

- **Q.** Discuss major libraries available with Python language and their application in Data Science / Explain use of Matplotlib and Scikit-learn in Data Science. **(WINTER 2023, 4 Marks; SUMMER 2024, 7 Marks OR)**

Answer:

- **NumPy:** Fast numerical arrays and vectorized operations.
- **Pandas:** DataFrames for reading, cleaning, grouping, merging tabular data.
- **Matplotlib:** Creates line, bar, pie, histogram, scatter plots for EDA and reporting.
- **Scikit-learn:** Provides regression, classification, clustering, preprocessing, model selection, evaluation.
- **Others:** SciPy, Seaborn, Statsmodels, NLTK, BeautifulSoup.

---

2. NumPy and Pandas:

- **Q.** Differentiate/Compare NumPy and Pandas. **(SUMMER 2022, 4 Marks; WINTER 2022, 7 Marks; WINTER 2023, 4 Marks OR; SUMMER 2025, 3 Marks)**

Answer:

Feature	NumPy	Pandas
Main object	ndarray	Series, DataFrame
Data type	Mostly homogeneous numeric arrays	Heterogeneous tabular data with labels
Indexing	Position-based; fast numeric ops	Label/position-based; aligned data
Use	Numerical computing, linear algebra	Data cleaning, analysis, CSV/Excel handling

- **Note:** NumPy is lower-level and faster for arrays; Pandas is built on NumPy and better for labeled tabular analysis.

- **Q.** Write a Python program to read data from a CSV file using pandas. [Diagram required – refer to the original question paper] **(SUMMER 2024, 4 Marks; WINTER 2024, 7 Marks; SUMMER 2025, 7 Marks)**

Answer:

- **Given:** CSV file data.csv.
- **To Find:** Read and display data using Pandas.

Code:

```
import pandas as pd

df = pd.read_csv("data.csv")
print(df.head())
print(df.shape)
print(df.columns)
```

- **Explanation:** read_csv() loads CSV into a DataFrame; head() shows first 5 rows; shape gives (rows, columns); columns lists column names.

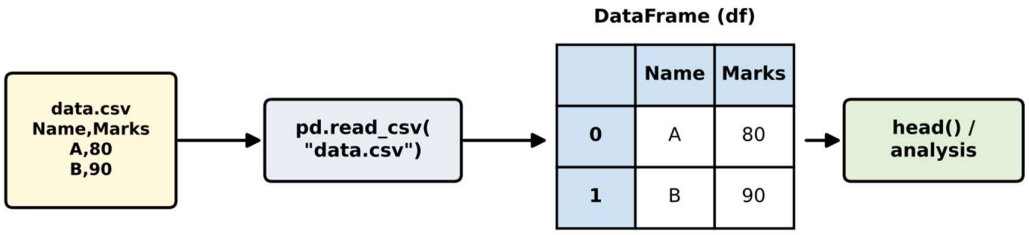


Fig. 2: Reading a CSV file into a pandas DataFrame.

- **Q.** Explain Groupby function in pandas with example. **(SUMMER 2022, 3 Marks OR; WINTER 2024, 3 Marks)**

Answer:

- groupby() splits data by a key, applies aggregation, and combines results.

Example:

```
import pandas as pd
df = pd.DataFrame({"Dept": ["A", "A", "B"], "Marks": [80, 90, 70]})
print(df.groupby("Dept")["Marks"].mean())
```

- **Output:** A 85.0, B 70.0.
- Used for department-wise average, sum, count, etc.

Other Important Questions:

1. NumPy:

- **Q.** What will be the output of the below Python code? import numpy as np; a = np.array([[1, 2.5, 3],[4, 5, 6.5]]); print(a.shape); print(a.ndim); print(a.itemsize); print(a.nbytes) **(SUMMER 2024, 4 Marks)**

Answer:

```
import numpy as np
a = np.array([[1, 2.5, 3], [4, 5, 6.5]])
print(a.shape)
print(a.ndim)
print(a.itemsize)
print(a.nbytes)
```

Output:

```
(2, 3)
2
8
48
```

- **Explanation:** Array has 2 rows × 3 columns; ndim=2; default dtype is float64, so itemsize=8 bytes; nbytes = 6 × 8 = 48.

- **Q.** Differentiate rand and randn function in NumPy. **(SUMMER 2022, 3 Marks)**

Answer:

Function	Distribution	Example
np.random.rand()	Uniform in [0, 1)	np.random.rand(2,2)
np.random.randn()	Standard normal: mean 0, std 1	np.random.randn(2,2)
Use	Random uniform values	Noise/normal-distributed data

---

2. Pandas DataFrames:

- **Q.** Explain different data structures available in pandas library. **(SUMMER 2024, 3 Marks)**

Answer:

- **Series:** One-dimensional labeled array; e.g., pd.Series([10,20]).
- **DataFrame:** Two-dimensional labeled table with rows and columns.
- **Index:** Immutable labels for rows/columns used for alignment and selection.

- **Q.** Explain DataFrame in Pandas with example. **(SUMMER 2022, 4 Marks)**

Answer:

- DataFrame is a 2D labeled table with rows and columns; each column can have a different type.

```
import pandas as pd
data = {"Name": ["A", "B"], "Marks": [80, 90]}
df = pd.DataFrame(data)
print(df)
```

Output:

```
  Name  Marks
0    A     80
1    B     90
```

- Supports selection, filtering, sorting, grouping, merging.

- **Q.** Consider the given dataframe. Write code to sort this dataframe by column A and by row 0 (first row). [Diagram required – refer to the original question paper] **(SUMMER 2024, 4 Marks OR)**

Answer:

```
# Sort by column A
df_sorted_col = df.sort_values(by="A", ascending=True)

# Sort columns by first row (row label 0)
df_sorted_row = df.sort_values(by=df.index[0], axis=1, ascending=True)
```

- **Explanation:** axis=0 sorts rows; axis=1 sorts columns using values of row 0.

- **Q.** Demonstrate how to remove a row and a column from a dataframe with appropriate example code. **(SUMMER 2024, 4 Marks)**

Answer:

```
import pandas as pd
df = pd.DataFrame({"A": [1, 2], "B": [3, 4]}, index=["r1", "r2"])

df.drop("r1", axis=0, inplace=True)   # remove row
df.drop("B", axis=1, inplace=True)    # remove column
print(df)
```

- **Explanation:** axis=0 removes row; axis=1 removes column; inplace=True modifies DataFrame.

- **Q.** Differentiate join and merge functions in pandas. **(SUMMER 2022, 4 Marks)**

Answer:

Feature	join()	merge()
Join key	Usually on index	On specified column(s)
Flexibility	Convenient for index-based joins	More flexible; supports on, left_on, right_on
Types	left/right/inner/outer	inner/left/right/outer
Use	Quick index joins	SQL-style column joins

- **Q.** What is the use of following operations on Pandas DataFrames? Explain with a small example of each: 1. shape 2. tail() 3. describe(). **(WINTER 2024, 7 Marks OR)**

Answer:

Operation	Use	Example
df.shape	Returns tuple (rows, columns)	(100, 5)
df.tail(n)	Shows last n rows; default 5	df.tail(3)
df.describe()	Summary statistics for numeric columns: count, mean, std, min, quartiles, max	df.describe()
Use in EDA	Quick size check, recent records, distribution overview	—

3. Data Sources and File Formats:

- **Q.** Write a Python code to read the local image, obtain statistics about that image, and display the image onscreen. **(WINTER 2023, 7 Marks)**

Answer:

- **Given:** Local image file local.jpg.
- **To Find:** Read image, print statistics, display image.

Code:

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread("local.jpg")
print("Shape:", img.shape)
print("Min:", img.min())
print("Max:", img.max())
print("Mean:", img.mean())
print("Mean per channel:", img.mean(axis=(0, 1)))

plt.imshow(img)
plt.axis("off")
plt.show()
```

- **Explanation:** imread() loads image as NumPy array; shape gives height, width, channels; min/max/mean give pixel statistics; imshow() displays image.

- **Q.** Describe the three levels of flat-file dataset. Explain any two with the help of example. **(WINTER 2023, 7 Marks OR)**

Answer:

- **1. Plain text flat file:** Unstructured text; read using open()/read().
- **2. Delimited flat file:** Fields separated by comma/tab; e.g., CSV. Read using pd.read_csv("data.csv").
- **3. Fixed-width flat file:** Each field has fixed character width; parse using slicing or pd.read_fwf().
- **Example CSV:** Name,Marks / A,80 / B,90; Pandas creates columns automatically.
- **Example fixed-width:** A 80 / B 90; fields occupy fixed positions, so slicing extracts Name and Marks.
- **Use:** Flat files are simple, portable, human-readable, but lack relationships and complex schema.

Chapter 3: Visualizing the Data

Repeated Questions:

1. Importance of Visualization and Magic Functions:

- **Q.** Why data visualization is important in Data Science? **(SUMMER 2023, 3 Marks; SUMMER 2025, 3 Marks)**

Answer:

- Data visualization is graphical representation of data to reveal patterns, trends, outliers, and relationships.
- Helps in exploratory data analysis (EDA): quickly understand distribution, correlation, and anomalies.
- Aids communication: conveys insights to non-technical stakeholders effectively.
- Supports decision making: visual evidence guides model selection and business actions.

- **Q.** Explain %matplotlib magic function / use of %matplotlib inline command. **(WINTER 2022, 3 Marks; WINTER 2023, 3 Marks OR)**

Answer:

- %matplotlib is an IPython magic command to configure matplotlib backend in Jupyter.
- %matplotlib inline renders plots directly in the notebook output cell, below code.
- Without it, plots may open in separate windows or not display in notebook.
- Usage: %matplotlib inline at start of notebook; other backends: %matplotlib notebook for interactive plots.

---

2. Chart Types:

- **Q.** Explain/create pie chart using matplotlib library with suitable example. **(WINTER 2022, 7 Marks OR; SUMMER 2024, 3 Marks; WINTER 2024, 7 Marks OR; SUMMER 2025, 4 Marks)**

Answer:

- Pie chart shows proportions of a whole as slices of a circle; each slice angle proportional to value.

Code:

```
import matplotlib.pyplot as plt
labels = ['A', 'B', 'C', 'D']
sizes = [30, 20, 25, 25]
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.title('Pie Chart Example')
plt.axis('equal')  # equal aspect ratio ensures circle
plt.show()
```

- **Explanation:** plt.pie() takes sizes and labels; autopct displays percentage; startangle rotates start; axis('equal') makes pie circular. Use for categorical share comparison.

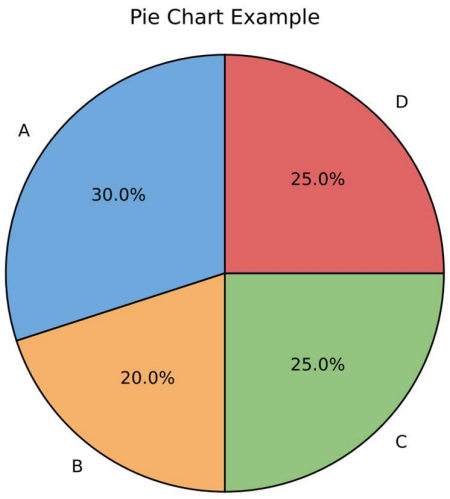


Fig. 3: Pie chart with four slices A, B, C, D.

- **Q.** Explain/create histogram with suitable example / create histogram with 4 bins from given data. [Diagram required – refer to the original question paper] **(SUMMER 2024, 7 Marks OR; SUMMER 2025, 3 Marks OR)**

Answer:

- Histogram shows frequency distribution of continuous/numeric data by grouping into bins.

Code (4 bins):

```
import matplotlib.pyplot as plt
data = [12, 15, 18, 22, 25, 28, 30, 32, 35, 38, 40, 42]
plt.hist(data, bins=4, color='skyblue', edgecolor='black')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram with 4 Bins')
plt.show()
```

- **Explanation:** bins=4 divides data range into 4 equal-width intervals; y-axis shows count per bin.

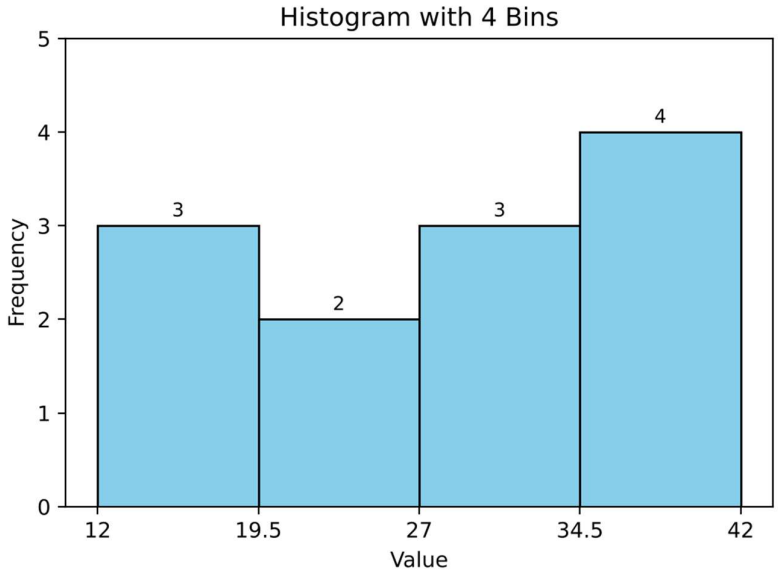


Fig. 4: Histogram with 4 bins.

- **Q.** Explain/draw bar chart with suitable example/utilizing at least three properties. **(WINTER 2023, 7 Marks OR; SUMMER 2025, 4 Marks OR)**

Answer:

- Bar chart represents categorical data with rectangular bars; height/length proportional to values.
- Properties used: color, width, edgecolor, label, title, xlabel, ylabel.

Code:

```
import matplotlib.pyplot as plt
categories = ['Math', 'Science', 'English']
marks = [85, 90, 78]
plt.bar(categories, marks, color='green', width=0.6, edgecolor='black')
plt.xlabel('Subjects')
plt.ylabel('Marks')
plt.title('Marks by Subject')
plt.show()
```

- **Explanation:** plt.bar() draws vertical bars; color sets fill; width controls bar thickness; edgecolor outlines bars.

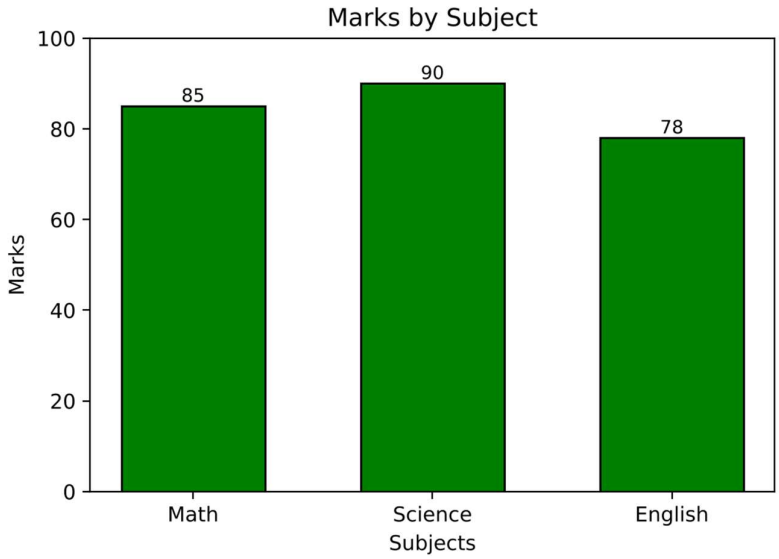


Fig. 5: Bar chart of marks by subject.

---

3. Customising Plots:

- **Q.** Explain importance of Labels, Annotations and Legends in Graphs/Matplotlib with example. **(SUMMER 2022, 3 Marks; WINTER 2022, 3 Marks; SUMMER 2023, 4 Marks OR; WINTER 2023, 4 Marks OR; WINTER 2024, 3 Marks; SUMMER 2025, 3 Marks)**

Answer:

- **Labels (xlabel, ylabel):** identify axes and units, making plot interpretable.
- **Title:** describes overall plot purpose.
- **Legend (plt.legend()):** identifies multiple data series (lines/bars) using labels.
- **Annotations (plt.annotate()):** highlight specific points with text/arrows.
- **Example:** plt.plot(x,y,label='Sales'); plt.xlabel('Year'); plt.ylabel('Revenue'); plt.legend(); plt.annotate('Peak', xy=(2020,100)).

- **Q.** Explain line styles / markers / line color / draw lines with different colors in charts. **(WINTER 2023, 3 Marks; WINTER 2023, 7 Marks; SUMMER 2024, 3 Marks OR)**

Answer:

- **Line styles:** linestyle or ls with values '-' (solid), '--' (dashed), '-.' (dash-dot), ':' (dotted).
- **Markers:** marker with 'o' (circle), 's' (square), '^' (triangle), '*' (star), etc.
- **Line color:** color or c with names ('r', 'g', 'b') or hex.

Example:

```
plt.plot(x, y1, color='red', linestyle='--', marker='o', label='Line 1')
plt.plot(x, y2, color='blue', linestyle='-', marker='s', label='Line 2')
plt.legend()
```

Different colors/styles distinguish multiple series.

---

4. NetworkX:

- **Q.** Write a brief note on NetworkX library / Draw graph using NetworkX / Draw undirected graph from adjacency matrix. [Diagram required – refer to the original question paper] **(SUMMER 2022, 4 Marks OR; SUMMER 2024, 4 Marks OR; WINTER 2024, 4 Marks OR)**

Answer:

- NetworkX is a Python library for creating, manipulating, and studying graphs/networks.
- Supports undirected/directed graphs, multigraphs, node/edge attributes, algorithms (shortest path, centrality).

Code to draw undirected graph from adjacency matrix:

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

adj = np.array([[0,1,1],[1,0,1],[1,1,0]])
G = nx.from_numpy_array(adj)
nx.draw(G, with_labels=True, node_color='lightblue', edge_color='gray')
plt.show()
```

- **Explanation:** from_numpy_array() creates graph from matrix; nx.draw() visualizes nodes and edges.

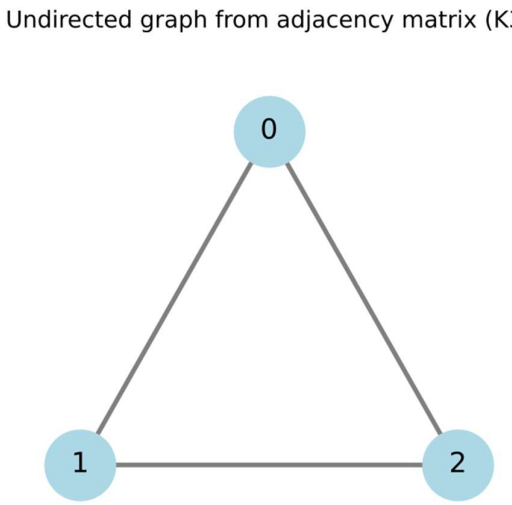


Fig. 6: Undirected graph (K3) drawn from the adjacency matrix.

Other Important Questions:

1. Plot Types:

- **Q.** Depict steps to create a scatter plot with example. **(SUMMER 2023, 3 Marks)**

Answer:

- Scatter plot shows relationship between two numeric variables as points.
- Steps: import matplotlib → prepare x,y data → plt.scatter(x,y) → add labels/title → plt.show().

Example:

```
import matplotlib.pyplot as plt
x = [1,2,3,4,5]
y = [2,4,5,4,5]
plt.scatter(x, y, color='red')
plt.xlabel('X'); plt.ylabel('Y')
plt.title('Scatter Plot')
plt.show()
```

- **Q.** Write a Python program to create boxplot from given marks data. What important observation about data can be derived from the boxplot? [Diagram required – refer to the original question paper] **(SUMMER 2024, 7 Marks)**

Answer:

- Boxplot displays distribution via quartiles, median, and outliers.

Code:

```
import matplotlib.pyplot as plt
marks = [45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95]
plt.boxplot(marks)
plt.ylabel('Marks')
plt.title('Boxplot of Marks')
plt.show()
```

- **Explanation:** Box shows interquartile range (Q1–Q3); line inside box = median; whiskers extend to min/max within 1.5×IQR; points beyond are outliers.
- **Observations:** Median marks, spread (IQR), skewness (if median not centered), presence of outliers.

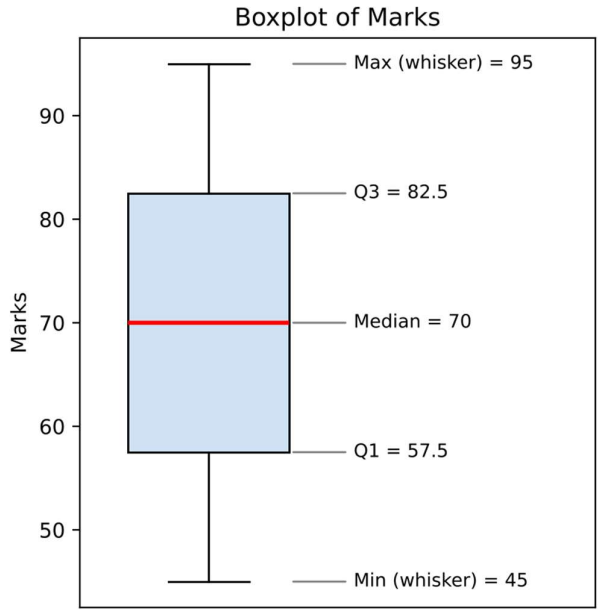


Fig. 7: Boxplot of marks.

- **Q.** List and explain different graphs in Matplotlib. **(SUMMER 2022, 7 Marks OR)**

Answer:

- **Line plot:** plt.plot() – trends over continuous intervals.
- **Bar chart:** plt.bar() – compare categorical values.
- **Histogram:** plt.hist() – frequency distribution of numeric data.
- **Scatter plot:** plt.scatter() – relationship between two variables.
- **Pie chart:** plt.pie() – proportion of categories.
- **Boxplot:** plt.boxplot() – distribution summary and outliers.
- **Heatmap:** plt.imshow() – matrix data as colors.

  Each graph suits different data types and analysis goals.

---

2. Plot Customisation:

- **Q.** Explain with example use of axes() and grid() method in plots. **(WINTER 2023, 4 Marks)**

Answer:

- plt.axes() sets or gets current axes; can add inset axes: plt.axes([left, bottom, width, height]) for custom position.
- plt.grid() adds grid lines to plot for easier reading; grid(True) enables, grid(False) disables.

Example:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3], [4,5,6])
plt.grid(True)
ax = plt.axes([0.6, 0.6, 0.2, 0.2])  # inset axes
ax.plot([1,2,3], [6,5,4])
plt.show()
```

- **Explanation:** grid(True) shows background grid; axes() with list adds a smaller inset plot at specified relative coordinates.
