CUET UG 2026

COMPUTER SCIENCE / IP

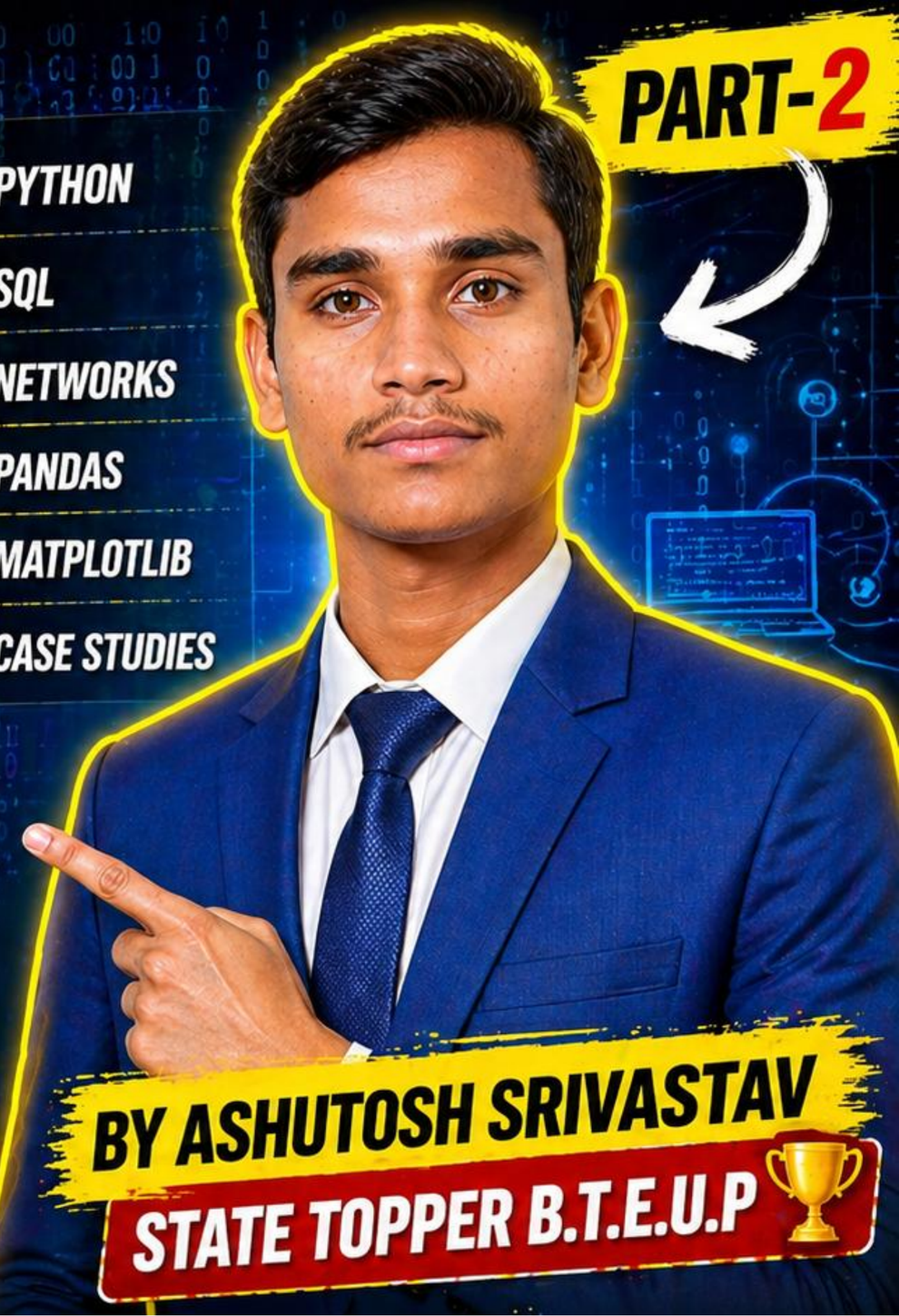
IN 1 VIDEO 😱

PART-2

- PYTHON
- SQL
- NETWORKS
- PANDAS
- MATPLOTLIB
- CASE STUDIES

COMPLETE SYLLABUS | SCORE 200+ | PYQs COVERED | STRATEGY + REVISION

250/250 MARKS 🎯



BY ASHUTOSH SRIVASTAV

STATE TOPPER B.T.E.U.P 🏆

SYLLABUS

Section A

1: Database Concepts

2: Structured Query Language – I

3: Structured Query Language – II

4: Computer Networks

Section B1: Computer Science

1: Exception and File Handling in Python

2: Stack

3: Queue

4: Searching

5: Sorting

6: Understanding Data

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SYLLABUS

7: Database Concepts

8: Structured Query Language

9: Computer Networks

10: Data Communication

11: Security Aspects

Section B2: Informatics Practices

1: Database Query using SQL

2: Data Handling using Pandas – I

3: Data Handling using Pandas – II

4: Plotting Data using Matplotlib

5: Introduction to Computer Networks

6: Societal Impacts

7: Project Based Learning

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Home
Shorts
Subscriptions
You
History

Sign in to like videos, comment, and subscribe.
Sign in

Explore
Shopping
Music
Movies
Show more

More from YouTube
YouTube Premium

KHUSHI FOUNDATION ACADEMY

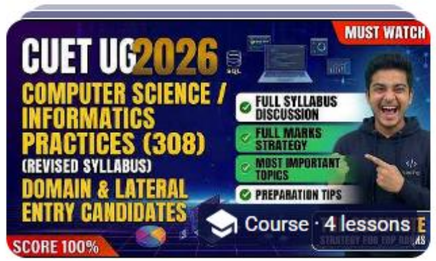
@KHUSHI.FOUNDATION.ACADEMY · 2.47K subscribers · 194 videos

यहां हम " गणित" का समाधान हिंदी और अंग्रेजी दोनों भाषाओं में आसान और स्पष्ट तरीके से प्रदान करने क ...more

t.me/khushifoundationacademy and 1 more link

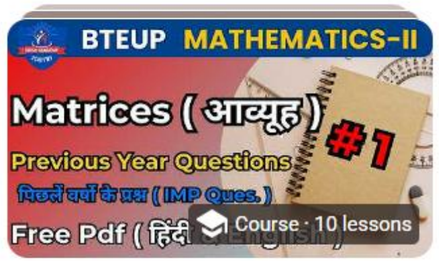
Subscribe Join

Home Videos Live Courses Playlists Posts



CUET UG 2026 Computer Science / Information Practices Full...

View full course

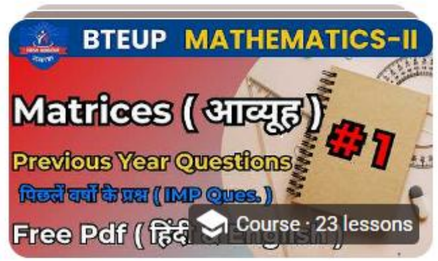


Matrix Questions for UP Polytechnic Second Semester Math for all...

View full course

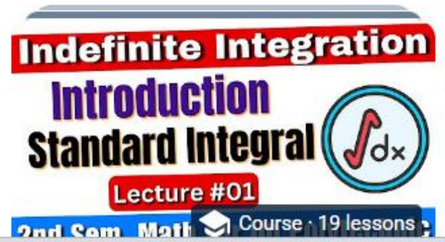
Integral Calculus 2nd Semester Math for UP Polytechnic 2025

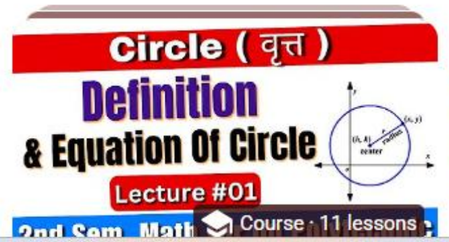
View full course



Applied Mathematics 3rd Important Questions #importantquestions...

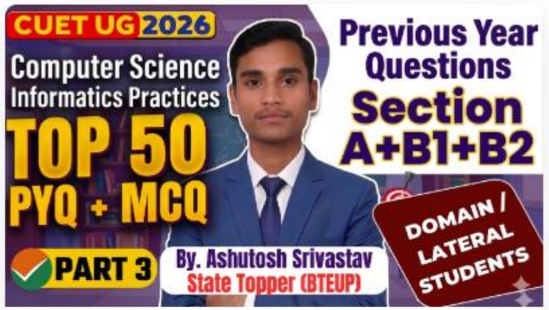
View full course



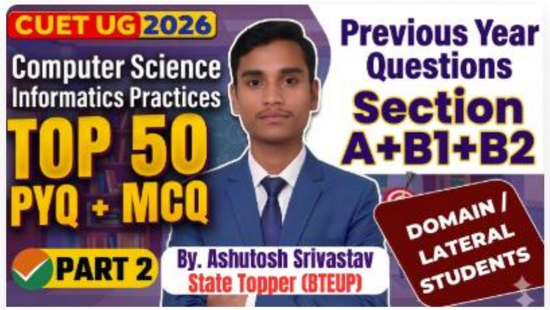


FREE NOTES

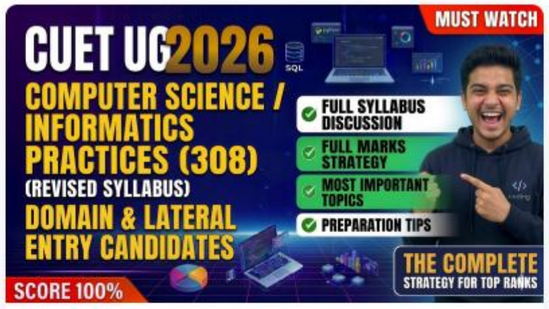
Download All Available Free Notes



Cuet Ug PYQ Part 3



CSE CUET PART 2



CUET UG CS/IP 2026 Syllabus Discussion

STACK

1. INTRODUCTION TO STACK

Definition:

A stack is a **linear data structure** that follows the principle:

LIFO (Last In First Out)

Real-Life Example:

Stack of plates

Last plate placed → first removed

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

STACK

Concept:

If elements are inserted like:

Push → 10, 20, 30

Stack looks like:

Top → 30

20

10

Now removal:

Pop → 30 (first removed)

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

STACK

2. BASIC OPERATIONS ON STACK

2.1 PUSH

Definition:

Insert element into stack

Before: 10, 20

Push 30

After: 10, 20, 30

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

STACK

2.2 POP

Definition:

Remove top element

Before: 10, 20, 30

Pop → 30 removed

After: 10, 20

STACK

2.3 PEEK (TOP)

Definition:

View top element without removing

Stack: 10, 20, 30

Peek → 30

2.4 isEmpty()

Returns:

True if stack has no elements

2.5 SIZE

Returns:

Number of elements

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

STACK

2.3 PEEK (TOP)

Definition:

View top element without removing

Stack: 10, 20, 30

Peek → 30

2.4 isEmpty()

Returns:

True if stack has no elements

2.5 SIZE

Returns:

Number of elements

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

STACK

3. CONDITIONS ON STACK

3.1 STACK OVERFLOW

Definition:

Occurs when you try to push into a **full stack**

Example:

Max size = 3

Stack = [10, 20, 30]

Push 40 → ERROR (Overflow)

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

STACK

3.2 STACK UNDERFLOW

Definition:

Occurs when you try to pop from an **empty stack**

Example:

Stack = empty

Pop → ERROR (Underflow)

STACK

4. IMPLEMENTATION OF STACK (IMPORTANT)

Using Python (Simple Version)

```
stack = []
# Push
stack.append(10)
stack.append(20)
stack.append(30)
print("Stack:", stack)
# Pop
stack.pop()
print("After pop:", stack)
# Peek
print("Top element:", stack[-1])
```

Output:

Stack: [10, 20, 30]

After pop: [10, 20]

Top element: 20

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

STACK

5. STACK IMPLEMENTATION (ARRAY LOGIC)

Basic Idea:

Use array

Maintain variable: TOP

Steps:

Initially:

TOP = -1

PUSH:

TOP = TOP + 1

stack[TOP] = value

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

STACK

5. STACK IMPLEMENTATION (ARRAY LOGIC)

Basic Idea:

Use array

Maintain variable: TOP

Steps:

Initially:

TOP = -1

PUSH:

TOP = TOP + 1

stack[TOP] = value

POP:

value = stack[TOP]

TOP = TOP - 1

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

STACK

6. APPLICATIONS OF STACK (VERY IMPORTANT)

6.1 PARENTHESIS MATCHING

Example:

((a + b))

Stack checks:

Opening → push

Closing → pop

STACK

6.2 INFIX TO POSTFIX

Infix:

A + B

Postfix:

AB+

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

STACK

6.3 EXPRESSION EVALUATION

Used in compilers

Used in calculators

6.4 FUNCTION CALL STACK

Tracks function calls

Stores return addresses

STACK

7. EXPRESSIONS (VERY IMPORTANT)

7.1 INFIX

A + B

Operator in middle

7.2 PREFIX

+ A B

Operator first

7.3 POSTFIX

A B +

Operator last

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

STACK

8. SHUNTING YARD ALGORITHM (ADVANCED)

Purpose:

Convert **Infix** → **Postfix**

Basic Idea:

Use stack for operators

Follow precedence rules

Example:

Infix:

A + B * C

Postfix:

ABC*+

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

STACK

8. SHUNTING YARD ALGORITHM (ADVANCED)

Purpose:

Convert **Infix** → **Postfix**

Basic Idea:

Use stack for operators

Follow precedence rules

Example:

Infix:

A + B * C

Postfix:

ABC*+

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

QUEUE

1. INTRODUCTION TO QUEUE

Definition:

A queue is a **linear data structure** in which elements are:

Inserted from **rear**

Removed from **front**

Principle:

FIFO (First In First Out)

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

QUEUE

Real-Life Example:

Queue at ticket counter:

Person A → Person B → Person C

A enters first → A exits first

Visualization:

Front → [10, 20, 30] ← Rear

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

QUEUE

2. QUEUE OPERATIONS

2.1 ENQUEUE (INSERT)

Definition:

Add element at **rear**

Before: [10, 20]

Enqueue 30

After: [10, 20, 30]

QUEUE

2.2 DEQUEUE (DELETE)

Definition:

Remove element from **front**

Before: [10, 20, 30]

Dequeue → 10 removed

After: [20, 30]

QUEUE

2.2 DEQUEUE (DELETE)

Definition:

Remove element from **front**

Before: [10, 20, 30]

Dequeue → 10 removed

After: [20, 30]

QUEUE

2.3 FRONT

Definition:

Get first element (no removal)

2.4 REAR

Definition:

Get last element

2.5 isEmpty()

Returns true if queue is empty

2.6 isFull()

Used in fixed-size queue

QUEUE

3. IMPLEMENTATION (LOGIC)

Python Example:

```
queue = []
# Enqueue
queue.append(10)
queue.append(20)
queue.append(30)
print("Queue:", queue)
# Dequeue
queue.pop(0)
print("After dequeue:", queue)
```

Output:

Queue: [10, 20, 30]

After dequeue: [20, 30]

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

QUEUE

4. TYPES OF QUEUE

4.1 SIMPLE QUEUE

Features:

Insert at rear

Delete from front

4.2 CIRCULAR QUEUE

Problem in Simple Queue:

Unused space after deletion

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

QUEUE

Solution:

Circular queue reuses space

Rear wraps to beginning

Advantage:

Efficient memory usage

QUEUE

4.3 PRIORITY QUEUE

Concept:

Each element has **priority**

Higher priority → processed first

Example:

Emergency patients in hospital

QUEUE

5. DEQUE (DOUBLE-ENDED QUEUE)

Definition:

Deque allows insertion and deletion from **both ends**

Operations:

5.1 INSERT FRONT

Add element at front

5.2 INSERT REAR

Add element at rear

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

QUEUE

5.3 DELETE FRONT

Remove from front

5.4 DELETE REAR

Remove from rear

Visualization:

[10, 20, 30]

Insert front → [5, 10, 20, 30]

Insert rear → [5, 10, 20, 30, 40]

QUEUE

6. APPLICATIONS OF QUEUE

6.1 TASK SCHEDULING

OS manages processes

6.2 PRINT QUEUE

Printer handles jobs in order

6.3 BFS (Breadth First Search)

Graph traversal

6.4 REQUEST HANDLING

Web servers process requests

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

QUEUE

7. APPLICATIONS OF DEQUE

7.1 MEMORY MANAGEMENT

Efficient allocation

7.2 UNDO/REDO SYSTEM

Software history tracking

7.3 TASK PRIORITIZATION

Different priority tasks

QUEUE

QUEUE VS STACK

Feature	Stack	Queue
Principle	LIFO	FIFO
Insert	Top	Rear
Delete	Top	Front

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SEARCHING

1. WHAT IS SEARCHING?

Definition:

Searching is the process of **finding a specific element in a collection of data**.

Example:

List = [10, 20, 30, 40]

Find = 30

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SEARCHING

2. LINEAR SEARCH (SEQUENTIAL SEARCH)

Core Idea:

Check **each element one by one** from beginning to end.

Steps:

Start from first element

Compare with target

If match → stop

Else move to next

If end reached → not found

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SEARCHING

Algorithm (Python Style):

```
def linear_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return i
    return -1
```

Dry Run Example:

```
arr = [10, 20, 30, 40]
target = 30
Step-by-step:
Compare 10 → not match
Compare 20 → not match
Compare 30 → match
Result:
Index = 2
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SEARCHING

Time Complexity:

Best case → O(1)

Worst case → O(n)

Key Feature:

Works on:

Sorted list

Unsorted list

Applications:

Small data

Unsorted data

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SEARCHING

3. BINARY SEARCH (VERY IMPORTANT)

Core Idea:

Divide the list into halves repeatedly

IMPORTANT CONDITION:

List must be **sorted**

Steps:

Find middle element

Compare with target

If equal → found

If target < middle → search left half

If target > middle → search right half

Repeat

SEARCHING

Algorithm (Iterative):

```
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif target < arr[mid]:
            high = mid - 1
        else:
            low = mid + 1
    return -1
```

Dry Run Example:

arr = [10, 20, 30, 40, 50]

target = 40

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SEARCHING

Step 1:

mid = 2 → value = 30

target > 30 → search right

Step 2:

mid = 3 → value = 40

match found

Result:

Index = 3

SEARCHING

4. RECURSIVE BINARY SEARCH

Function calls itself

```
def binary_search_recursive(arr, low, high, target):
    if low > high:
        return -1
    mid = (low + high) // 2
    if arr[mid] == target:
        return mid
    elif target < arr[mid]:
        return binary_search_recursive(arr, low, mid - 1, target)
    else:
        return binary_search_recursive(arr, mid + 1, high, target)
```

SEARCHING

5. DRY RUN (IMPORTANT CONCEPT)

Definition:

Dry run means:

Manually executing algorithm

Tracking values step-by-step

Why Important:

Understand logic

Find errors

Exam questions often ask dry run

SEARCHING

COMPARISON: LINEAR vs BINARY SEARCH

Feature	Linear Search	Binary Search
Method	Check one by one	Divide into halves
Requirement	Any list	Sorted list
Time Complexity	O(n)	O(log n)
Efficiency	Slow for large data	Fast for large data

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SEARCHING

7. TIME COMPLEXITY (VERY IMPORTANT)

Linear Search:

Best → O(1)

Worst → O(n)

Binary Search:

Best → O(1)

Worst → O(log n)

Insight:

Binary search is much faster for large data

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SEARCHING

8. APPLICATIONS

Linear Search:

Unsorted list

Small data

Binary Search:

Sorted data

Dictionaries

Phone directory

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SEARCHING

8. APPLICATIONS

Linear Search:

Unsorted list

Small data

Binary Search:

Sorted data

Dictionaries

Phone directory

SORTING

1. WHAT IS SORTING?

Definition:

Sorting is the process of **arranging elements in a specific order**:

Ascending (small → large)

Descending (large → small)

Why Sorting is Important:

Makes searching faster

Improves data organization

Used in real systems (databases, apps)

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

2. TYPES OF SORTING

2.1 COMPARISON-BASED SORTING

Idea:

Elements are compared with each other

Examples:

Bubble Sort

Selection Sort

Insertion Sort

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

2.2 NON-COMPARISON SORTING

Idea:

No direct comparison

Based on:

Counting

Distribution

Examples:

Radix Sort

Bucket Sort

SORTING

3. BUBBLE SORT

Core Idea:

Compare adjacent elements and swap if needed

Working:

Example:

[5, 3, 2]

Pass 1:

Compare 5 & 3 → swap → [3, 5, 2]

Compare 5 & 2 → swap → [3, 2, 5]

Pass 2:

Compare 3 & 2 → swap → [2, 3, 5]

Sorted

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

Code:

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
```

Complexity:

Worst → $O(n^2)$, Best → $O(n)$

Advantages:

Easy to understand

Disadvantages:

Very slow for large data

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

4. SELECTION SORT

Core Idea:

Find smallest element and place it at correct position

Working:

Example:

[5, 3, 2]

Step 1:

Smallest = 2 → swap with 5 → [2, 3, 5]

SORTING

Code:

```
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        min_index = i
        for j in range(i+1, n):
            if arr[j] < arr[min_index]:
                min_index = j
        arr[i], arr[min_index] = arr[min_index], arr[i]
```

Complexity: :- $O(n^2)$ always

Advantage:

No extra memory

Disadvantage:

Inefficient for large lists

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

5. INSERTION SORT

Core Idea:

Build sorted list one element at a time

Working:

Example:

[5, 3, 2]

Step 1:

Take 3 → insert before 5 → [3, 5, 2]

Step 2:

Take 2 → insert → [2, 3, 5]

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

Code:

```
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = key
```

Complexity: :- Worst → $O(n^2)$ Best → $O(n)$

Advantage:

Efficient for small/nearly sorted data

Disadvantage:

Slow for large data

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

6. MERGE SORT (VERY IMPORTANT)

Divide → Sort → Merge

Steps:

Divide array into halves

Sort each half

Merge sorted halves

Example:

[5, 3, 2, 1]

Split:

[5,3] [2,1]

Sort:

[3,5] [1,2]

Merge:

[1,2,3,5]

SORTING

Code:

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr)//2
        left = arr[:mid]
        right = arr[mid:]
        merge_sort(left)
        merge_sort(right)
        i = j = k = 0
        while i < len(left) and j < len(right):
            if left[i] < right[j]:
                arr[k] = left[i]
                i += 1
            else:
                arr[k] = right[j]
                j += 1
            k += 1

        while i < len(left):
            arr[k] = left[i]
            i += 1
            k += 1

        while j < len(right):
            arr[k] = right[j]
            j += 1
            k += 1
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

Complexity:

Always → O(n log n)

Advantages:

Very efficient

Stable

Disadvantages:

Extra memory required

SORTING

COMPARISON TABLE

Algorithm	Time Complexity	Best Use
Bubble	$O(n^2)$	Learning
Selection	$O(n^2)$	Small data
Insertion	$O(n^2)$	Nearly sorted
Merge	$O(n \log n)$	Large data

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

COMPARISON TABLE

Algorithm	Time Complexity	Best Use
Bubble	$O(n^2)$	Learning
Selection	$O(n^2)$	Small data
Insertion	$O(n^2)$	Nearly sorted
Merge	$O(n \log n)$	Large data

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

1. TIME & SPACE COMPLEXITY OF SORTING

What is Time Complexity?

It tells **how fast an algorithm runs** based on input size (n).

What is Space Complexity?

It tells **how much extra memory is used**

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

TABLE (UNDERSTAND LOGIC, NOT JUST MEMORIZE)

Algorithm	Best	Average	Worst	Space
Bubble Sort	O(n)	O(n²)	O(n²)	O(1)
Selection Sort	O(n²)	O(n²)	O(n²)	O(1)
Insertion Sort	O(n)	O(n²)	O(n²)	O(1)
Merge Sort	O(n log n)	O(n log n)	O(n log n)	O(n)

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

Deep Insight:

Why Bubble Best = O(n)?

If already sorted → only one pass

Why Selection always $O(n^2)$?

Always scans entire list

Why Insertion best = O(n)?

If sorted → no shifting needed

Why Merge = O(n log n)?

Divides log n times

Merges n elements

SORTING

2. HASHING

Definition:

Hashing is a technique used to **store and retrieve data quickly using a key**

Core Idea:

Instead of searching → directly jump to location

Real-Life Example:

Dictionary:

Word → Meaning

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

In Programming:

```
my_map = {}
my_map["name"] = "John"
my_map["age"] = 30

print(my_map["name"])  # Output: John
```

Insight:

Key → "name"

Value → "John"

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

3. HASH FUNCTION (CORE OF HASHING)

Definition:

A function that converts a key into an index

Example:

Hash function:

index = key % table_size

Example:

key = 15

table size = 10

index = 15 % 10 = 5

→ Store at index 5

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

3. HASH FUNCTION (CORE OF HASHING)

Definition:

A function that converts a key into an index

Example:

Hash function:

index = key % table_size

Example:

key = 15

table size = 10

index = 15 % 10 = 5

→ Store at index 5

SORTING

4. HASH TABLE

Definition:

A data structure that stores data using hash function

Structure:

Index → Value

0 → -

1 → -

2 → Data

...

Important Point:

Each index stores **one element**

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

5. WHY HASHING IS FAST

Searching in normal list:

→ O(n)

Searching in hash table:

→ O(1) (average)

Reason:

Direct access using index

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

SORTING

6. COLLISION (VERY IMPORTANT)

Definition:

When two keys map to the same index

Example:

key1 = 15 → index 5

key2 = 25 → index 5

→ Collision occurs

Why Collision Happens:

Limited table size

Many keys

SORTING

7. COLLISION RESOLUTION METHODS

7.1 CHAINING

Idea:

Store multiple values in same index using list

Index 5 → [15, 25, 35]

Advantage:

Easy to implement

Disadvantage:

Extra memory

SORTING

7.2 OPEN ADDRESSING (LINEAR PROBING)

Idea:

Find next empty slot

Example:

Index 5 occupied → check 6 → store there

Problem:

Clustering

SORTING

7.3 REHASHING

Idea:

Use new hash function or resize table

Example:

Increase table size → reduce collisions

SORTING

8. PERFECT HASH FUNCTION

Definition:

A hash function with **no collisions**

Reality:

Rare in real systems

9. COMPLETE FLOW (IMPORTANT)

Key → Hash Function → Index → Store/Retrieve

SORTING

10. APPLICATIONS OF HASHING

1. Dictionaries / Maps
2. Databases
3. Password storage
4. Caching
5. Searching

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

1. WHAT IS DATA?

Definition:

Data is **raw facts and figures** collected for analysis or reference.

Key Understanding:

Data = raw (not processed)

Information = processed data

Examples of Data:

Numbers → 10, 20, 30

Text → “Hello”

Images → Photos

Sensor data → temperature

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

Sources of Data:

Surveys

Sensors

Databases

Observations

Real-Life Example:

Marks of students = Data

Average marks = Information

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

2. PURPOSE OF DATA (WHY DATA IS IMPORTANT)

2.1 Extracting Meaningful Insights

Idea:

Find patterns and trends

Example:

Sales increasing → business growth

2.2 Informed Decision Making

Idea:

Use data to make better decisions

Example:

Company decides price using data

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

2.3 Driving Conclusions

Idea:

Use data to:

Form conclusions

Test hypotheses

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

3. TYPES OF DATA

3.1 STRUCTURED DATA

Definition:

Data that is **organized in rows and columns**

Features:

Well-defined format

Stored in databases

Easy to search

Example:

Name | Age | Marks

Aman | 20 | 85

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

Advantages:

Easy querying

Fast processing

Disadvantages:

Less flexible

UNDERSTANDING DATA

3.2 UNSTRUCTURED DATA

Definition:

Data with **no fixed format**

Examples:

Videos

Images

Emails

Social media

Features:

Flexible

Hard to analyze

Problem:

Cannot use simple queries

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

COMPARISON

Feature	Structured	Unstructured
Format	Fixed	No fixed format
Storage	Databases	Files
Flexibility	Low	High
Searching	Easy	Difficult

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

4. STATISTICAL METHODS (VERY IMPORTANT)

These help us **understand data patterns**

4.1 MEAN (AVERAGE)

Formula:

$$\text{Mean} = \frac{\sum x}{n}$$

Meaning:

Average of all values

Example:

Data: 10, 20, 30

Mean = (10 + 20 + 30) / 3 = 20

Insight:

Represents **center of data**

UNDERSTANDING DATA

4.2 MEDIAN

Definition:

Middle value after sorting

Example:

Data: 10, 20, 30

Median = 20

Important:

Less affected by extreme values

UNDERSTANDING DATA

4.3 MODE

Definition:

Most frequent value

Example:

Data: 10, 20, 20, 30

Mode = 20

UNDERSTANDING DATA

4.4 RANGE

Formula:

$$\text{Range} = Max - Min$$

Example:

Data: 10, 20, 30

Range = 30 - 10 = 20

Insight:

Measures spread

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

4.5 VARIANCE

Idea:

Measures how far values are from mean

Meaning:

High variance → data spread out

Low variance → data close

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

4.6 STANDARD DEVIATION

Formula:

$$\sigma = \sqrt{\frac{\sum (x - \bar{x})^2}{n}}$$

Meaning:

Average deviation from mean

Insight:

Small → consistent data

Large → scattered data

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

4.7 INTERQUARTILE RANGE (IQR)

Definition:

Difference between Q3 and Q1

Meaning:

Measures spread of middle 50%

Important:

Less affected by outliers

Data → Analysis → Insight → Decision

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

4.7 INTERQUARTILE RANGE (IQR)

Definition:

Difference between Q3 and Q1

Meaning:

Measures spread of middle 50%

Important:

Less affected by outliers

Data → Analysis → Insight → Decision

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

1. BIG PICTURE (VERY IMPORTANT)

Think of data like this pipeline:

Collect → Organize → Clean → Transform → Analyze → Visualize → Report → Store

This is how real systems (Google, Amazon, AI systems) work.

2. DATA INTERPRETATION

Definition:

Understanding data by identifying **patterns, trends, and relationships**

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

2.1 GRAPHS

Types:

1. Bar Graph :- Compare categories

Example: Marks of students

2. Line Graph :- Show trends over time

Example: Temperature over days

3. Histogram :- Frequency distribution

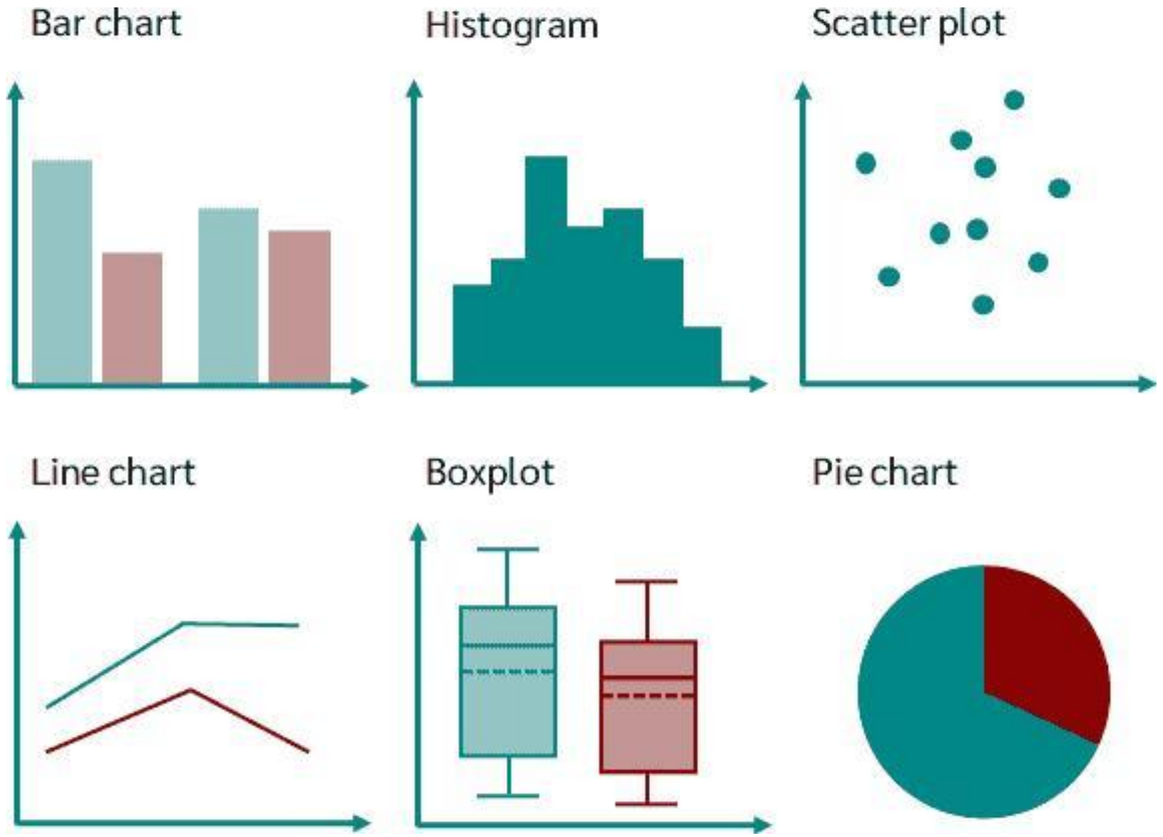
Example: Marks distribution

4. Scatter Plot :- Relationship between variables

Example: Height vs Weight

5. Network Graph :- Shows connections

Example: Social networks



Graphs help you see patterns instantly instead of reading raw data .

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

2.2 DIAGRAMS

1. Flowchart

Shows steps/process

2. Venn Diagram

Shows relationships between sets

UNDERSTANDING DATA

3. DATA VISUALIZATION (TOOLS)

3.1 MATPLOTLIB

Definition:

Python library for creating graphs

Features:

Line, bar, pie charts

Customization

Example:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3], [10,20,30])
plt.show()
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

3.2 SEABORN

Definition:

Advanced visualization library built on Matplotlib

Features:

More attractive graphs

Better statistical plots

Insight:

Matplotlib = basic

Seaborn = advanced

UNDERSTANDING DATA

4. COLLECTION & ORGANIZATION OF DATA

4.1 DATA COLLECTION

Gathering data from different sources

Sources:

Surveys

Sensors

Databases

Experiments

Important Point:

Data must be:

Accurate

Reliable

Consistent

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

4.2 DATA ORGANIZATION

Definition:

Arranging data into structured format

Why Needed:

Remove errors

Improve clarity

Example:

Raw data:

20, ?, 30, wrong value

Organized:

20, 25, 30

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

5. DATA PROCESSING STEPS (VERY IMPORTANT)

5.1 DATA CLEANING

Definition:

Removing:

Errors

Missing values

Outliers

Example:

Age: 20, 22, 200 → 200 is error

UNDERSTANDING DATA

5.2 DATA TRANSFORMATION

Definition:

Convert data into usable format

Techniques:

Normalization

Encoding

Example:

Male/Female → 1/0

UNDERSTANDING DATA

5.3 DATA ANALYSIS

Definition:

Applying techniques to find insights

Methods:

Statistics

Machine learning

Example:

Finding average marks

Predicting sales

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

5.4 DATA REPORTING

Definition:

Presenting results

Forms:

Graphs

Reports

Dashboards

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

6. DATA STORAGE (VERY IMPORTANT)

Definition:

Saving data for future use

TYPES OF STORAGE DEVICES

6.1 SSD (Solid State Drive)

Features:

Fast

No moving parts

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

UNDERSTANDING DATA

6.2 HDD (Hard Disk Drive)

Features:

Traditional

Slower than SSD

6.3 PEN DRIVE

Features:

Portable

Easy data transfer

UNDERSTANDING DATA

6.4 MEMORY CARD

Uses:

Mobile

Camera

6.5 CD/DVD

Uses:

Archiving data

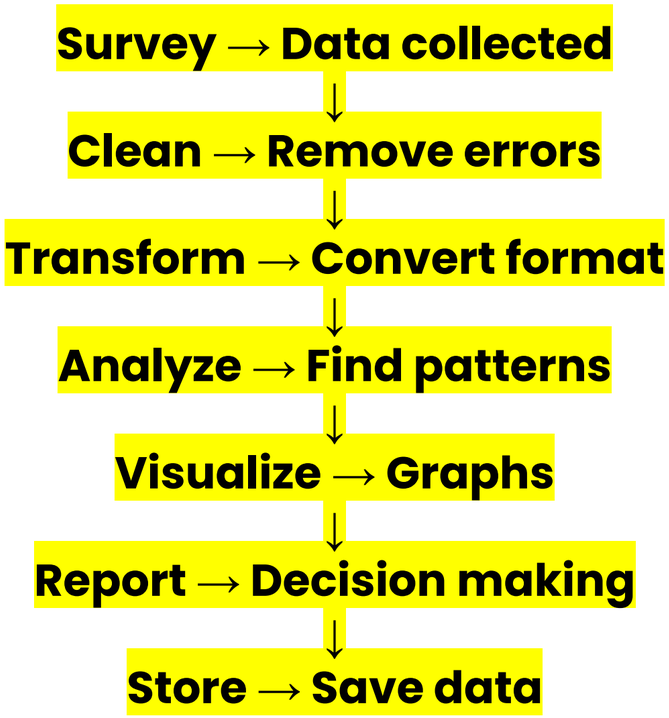
6.6 TAPE DRIVE

Uses:

Large backup systems

UNDERSTANDING DATA

COMPLETE DATA FLOW



By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Security Aspects

Network Security Concepts

These are basic concepts used to protect systems and data from unauthorized access.

Firewall

A firewall acts as a **security barrier between internal and external networks**.

It monitors and controls incoming and outgoing traffic based on security rules.

- Blocks unauthorized access
- Allows safe communication

Example: A firewall blocks suspicious websites from accessing your system.

Security Aspects

Cookies

Cookies are **small files stored on a user's device** by websites to remember user data.

- Stores login details, preferences
- Can be used for tracking

Prevention:

- Clear cookies regularly
- Adjust browser settings

Hackers and Crackers

- **Hackers**: People who use their skills to find and fix security problems
- **Crackers**: People who break security for illegal purposes

Types:

- White Hat → ethical
- Black Hat → malicious
- Grey Hat → neutral

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Security Aspects

Antivirus and Its Working

Antivirus software is used to **detect, prevent, and remove malicious programs**.

How it works:

- Scans files and programs
- Detects suspicious patterns
- Removes or quarantines threats

Example: Windows Defender, Quick Heal

Security Aspects

Threats and Prevention

Viruses

Programs that attach to files and spread by infecting other files.

Effects:

- Data corruption
- System slowdown

Prevention:

- Install antivirus
- Avoid unknown downloads

Security Aspects

Worms

Self-replicating malware that spreads across networks.

Prevention:

- Keep system updated
- Use firewall

Trojan Horse

Malicious software disguised as legitimate software.

Prevention:

- Download from trusted sources
- Avoid unknown links

Security Aspects

Ransomware

Blocks access to data and demands payment.

Prevention:

- Backup data regularly
- Avoid suspicious files

Spam

Unwanted or irrelevant emails/messages.

Prevention:

- Use spam filters
- Avoid clicking unknown links

Security Aspects

Adware

Software that shows unwanted advertisements.

Prevention:

- Install ad blockers
- Avoid free unsafe software

HTTP vs HTTPS

- HTTP → not secure
- HTTPS → secure (uses encryption)

Always prefer HTTPS for:

- Online payments
- Login activities

Security Aspects

Network Security Threats

Snooping

Monitoring network traffic to capture data.

- Can steal sensitive information
- Also called sniffing

Eavesdropping

Unauthorized real-time interception of communication.

- Targets calls, messages
- Happens during transmission

Security Aspects

Denial of Service (DoS)

Attack that makes a system or network unavailable.

Prevention:

- Firewalls
- Load balancing
- Intrusion detection systems

Intrusion Problems

Unauthorized access or misuse of network resources.

Prevention:

- Strong passwords
- Regular monitoring

Security Aspects

Simple Awareness Example

```
# Simple password strength checker
password = input("Enter password: ")

if len(password) >= 8:
    print("Password is Strong")
else:
    print("Password is Weak")
```

Data Handling using Pandas – I

Introduction to Python Libraries

(a) Pandas

Used for **data manipulation and analysis**

Works with structured data (tables)

Provides two main data structures:

- **Series (1D)**
- **DataFrame (2D)**

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – I

(b) NumPy

Used for **numerical computing**

Supports arrays and matrices

Faster than Python lists

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

Output:

[2 4 6]

Data Handling using Pandas – I

(c) Matplotlib

Used for **data visualization**

```
import matplotlib.pyplot as plt


x = [1, 2, 3]
y = [10, 20, 30]
plt.plot(x, y)
plt.show()
```

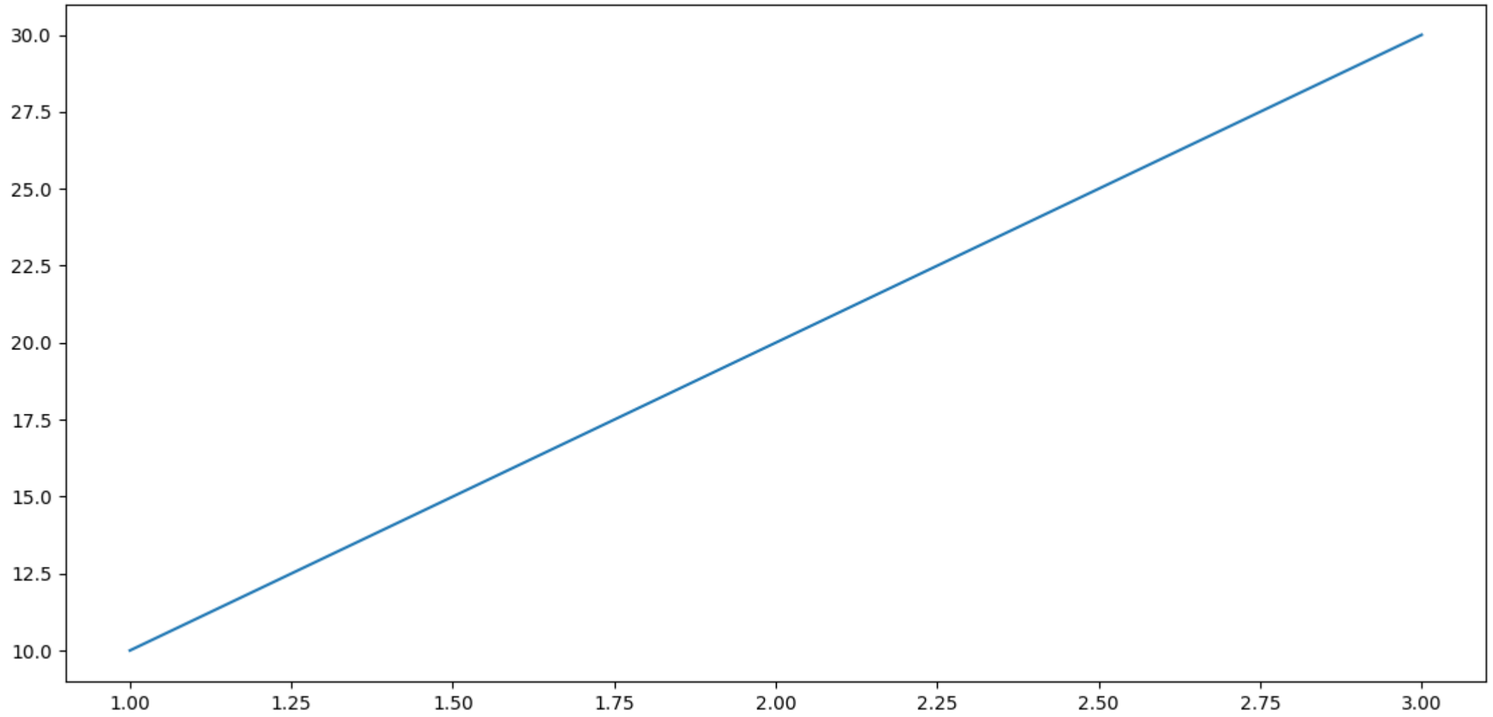
Data Handling using Pandas – I

(c) Matplotlib

Used for **data visualization**

```
import matplotlib.pyplot as plt

x = [1, 2, 3]
y = [10, 20, 30]
plt.plot(x, y)
plt.show()
```



Data Handling using Pandas – I

Data Structures in Pandas

2.1 Series (1D Data)

A **Series** is a one-dimensional labeled array.

Creation of Series

(1) From Array

```
import pandas as pd
import numpy as np
s = pd.Series(np.array([1, 2, 3]))
print(s)
```

Output:

```
0    1
1    2
2    3
dtype: int64
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – I

(2) From Dictionary

```
data = {'A': 10, 'B': 20, 'C': 30}
s = pd.Series(data)
print(s)
Output:
A    10
B    20
C    30
dtype: int64
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – I

(3) From Scalar Value

```
s = pd.Series(5, index=['A', 'B', 'C'])
print(s)
Output:
A    5
B    5
C    5
dtype: int64
```

Data Handling using Pandas – I

Mathematical Operations on Series

Operations are performed **element-wise**.

```
s1 = pd.Series([1, 2, 3])
s2 = pd.Series([4, 5, 6])
print(s1 + s2)
print(s1 - s2)
```

```
Output:
0    5
1    7
2    9
0   -3
1   -3
2   -3
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – I

Head and Tail Functions

head(n) → returns first n elements

tail(n) → returns last n elements

```
s = pd.Series([10, 20, 30, 40, 50])
print(s.head(2))
print(s.tail(2))
Output:
0    10
1    20


3    40
4    50
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – I

Selection, Indexing, and Slicing

(a) Label-based Selection

```
s = pd.Series([10, 20, 30], index=['A', 'B', 'C'])
print(s['A'])
Output:
10
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – I

(b) Position-based Selection

```
print(s.iloc[0])
```

```
Output:
```

```
10
```

(c) Slicing

```
print(s[1:3])
```

```
Output:
```

```
B    20
C    30
```

Data Handling using Pandas – I

DataFrame: A **DataFrame** is a two-dimensional labeled data structure with rows and columns.

Creation of DataFrame

(a) From Dictionary of Series

```
data = {
'Name': ['A', 'B', 'C'],
    'Marks': [80, 90, 85]
}
df = pd.DataFrame(data)
print(df)
```

Output:

```
  Name  Marks
0    A     80
1    B     90
2    C     85
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – I

(b) From List of Dictionaries

```
data = [
    {'Name': 'A', 'Marks': 80},
    {'Name': 'B', 'Marks': 90}
]

df = pd.DataFrame(data)
print(df)
```

Data Handling using Pandas – I

(c) From CSV File

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

Display Functions

```
print(df.head())
print(df.tail())
```

Iteration

```
for index, row in df.iterrows():
    print(row['Name'], row['Marks'])
```

Data Handling using Pandas – I

3. Operations on Rows and Columns

Add Column

```
df['Grade'] = ['A', 'A+', 'A']
```

Select Columns

```
print(df['Name'])
print(df[['Name', 'Marks’]])
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – I

3. Operations on Rows and Columns

Delete Column

```
df.drop('Grade', axis=1, inplace=True)
```

Rename Column

```
df.rename(columns={'Marks': 'Score'}, inplace=True)
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – I

4. Indexing in DataFrame

loc (Label-based)

Used to access data using row and column labels.

```
print(df.loc[0, 'Name’])
```

Data Handling using Pandas – I

iloc (Position-based)

Used to access data using integer positions.

```
print(df.iloc[0, 1])
```

Boolean Indexing

Used to filter data based on condition.

```
print(df[df['Marks'] > 80])
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – I

5. Styling and Formatting Data

Used to improve appearance of DataFrame.

```
df.style.set_properties(**{'color': 'blue'})
```

Data Handling using Pandas – I

6. Joining, Merging, and Concatenation

Concatenation

Combines DataFrames vertically or horizontally.

```
df1 = pd.DataFrame({'A': [1, 2]})
df2 = pd.DataFrame({'A': [3, 4]})
print(pd.concat([df1, df2]))
Output:
   A
0  1
1  2
0  3
1  4
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – I

Merging

Combines DataFrames based on a common column.

```
df1 = pd.DataFrame({'id': [1, 2], 'name': ['A', 'B']})
df2 = pd.DataFrame({'id': [1, 2], 'marks': [80, 90]})
print(pd.merge(df1, df2, on='id’))
```

```
Output:
   id name  marks
0   1    A     80
1   2    B     90
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – I

Joining

Combines DataFrames using index.

```
df1 = pd.DataFrame({'A': [1, 2]}, index=[0, 1])
df2 = pd.DataFrame({'B': [3, 4]}, index=[0, 1])
print(df1.join(df2))
```

```
Output:
   A  B
0  1  3
1  2  4
```

Data Handling using Pandas – I

7. Importing and Exporting Data

Import CSV

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

Export CSV

```
df.to_csv('output.csv', index=False)
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

Descriptive Statistics

These are used to **summarize and analyze numerical data**.

Maximum, Minimum, Sum, Count

max() → largest value

min() → smallest value

sum() → total

count() → number of non-null values

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

```
import pandas as pd
data = pd.Series([10, 20, 30, 40, 50])
print(data.max())
print(data.min())
print(data.sum())
print(data.count())
```

Output:

```
50
10
150
5
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

Mean (Average)

Mean is the **sum of all values divided by number of values**

```
print(data.mean())
```

```
Output:
```

```
30.0
```

Median

Median is the **middle value after sorting**

```
print(data.median())
```

```
Output:
```

```
30.0
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

Mode

Mode is the **most frequent value**

```
data = pd.Series([1, 2, 2, 3, 4])
print(data.mode())
```

Output:

```
0    2
dtype: int64
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

Variance

Variance measures **spread of data from mean**

```
print(data.var())
```

Standard Deviation

Standard deviation shows **how much data deviates from mean**

```
print(data.std())
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

Quartiles

Quartiles divide data into **4 equal parts**

```
data = pd.Series([10, 20, 30, 40, 50])
print(data.quantile(0.25))   # Q1
print(data.quantile(0.50))   # Q2 (median)
print(data.quantile(0.75))   # Q3
```

Output:

```
20.0
30.0
40.0
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

DataFrame Operations

Aggregation

Aggregation means **applying functions like sum, mean on data**

```
df = pd.DataFrame({
    'Marks': [80, 90, 85]
})
print(df['Marks'].sum())
print(df['Marks'].mean())
```

Multiple Aggregation

```
print(df.groupby('Dept')['Marks'].agg(['sum', 'mean']))
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

Group By

Grouping is used to **divide data into groups and apply functions**

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

Output:

```
Dept
A    85.0
B    70.0
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

Sorting

Sorting arranges data **ascending or descending**

```
df = pd.DataFrame({
    'Name': ['A', 'B', 'C'],
    'Marks': [80, 90, 70]
})
print(df.sort_values(by='Marks'))
```

Output:

```
  Name  Marks
2    C     70
0    A     80
1    B     90
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

Deleting and Renaming Index

Reset Index

```
df.reset_index(drop=True, inplace=True)
```

Rename Index

```
df.rename_axis("new_index_name", inplace=True)
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

Pivoting: Pivot reshapes data into a new format

```
df = pd.DataFrame({
    'Name': ['A', 'A', 'B'],
    'Subject': ['Math', 'Sci', 'Math'],
    'Marks': [80, 85, 90]
})
pivot_table = df.pivot_table(values='Marks', index='Name', columns='Subject')
print(pivot_table)
```

Output:

```
Subject  Math   Sci
Name
A        80.0  85.0
B        90.0   NaN
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

Handling Missing Values

Dropping Missing Values

Removes rows/columns with null values

```
df = pd.DataFrame({
    'A': [1, None, 3],
    'B': [4, 5, None]
})
print(df.dropna())
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

Filling Missing Values

Replaces null values

```
print(df.fillna(0))
```

Output:

```
     A    B
0  1.0  4.0
1  0.0  5.0
2  3.0  0.0
```

Data Handling using Pandas – II

Importing and Exporting Data (MySQL)

Importing from MySQL

```
import mysql.connector
import pandas as pd
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="password",
    database="testdb"
)
query = "SELECT * FROM table_name"
df = pd.read_sql(query, conn)
print(df)
conn.close()
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Data Handling using Pandas – II

Exporting to MySQL

```
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="password",
    database="testdb"
)

df.to_sql(name='table_name', con=conn, if_exists='replace', index=False)

conn.close()
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Plotting Data using Matplotlib

Introduction

Matplotlib is a Python library used to create **graphs and charts for data visualization**.

It helps in understanding data through visual representation like lines, bars, circles, etc.

To use Matplotlib:

```
import matplotlib.pyplot as plt
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

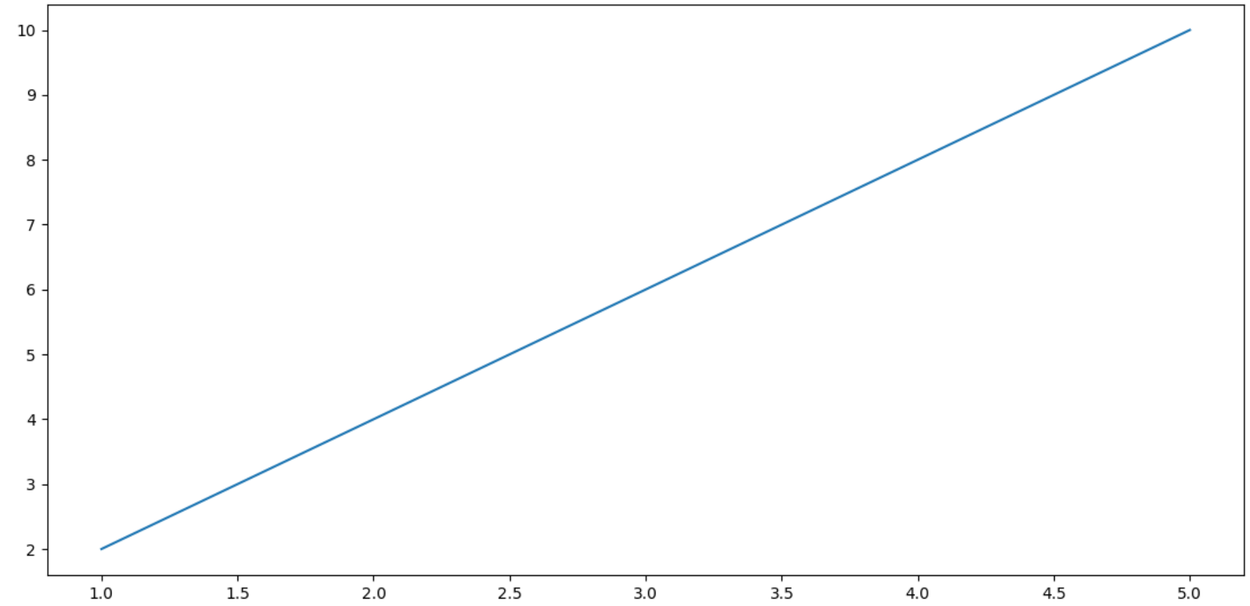
Plotting Data using Matplotlib

1. Line Plot

Used to show **trend or relationship between two variables**

Example Code:

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
plt.plot(x, y)
plt.show()
```



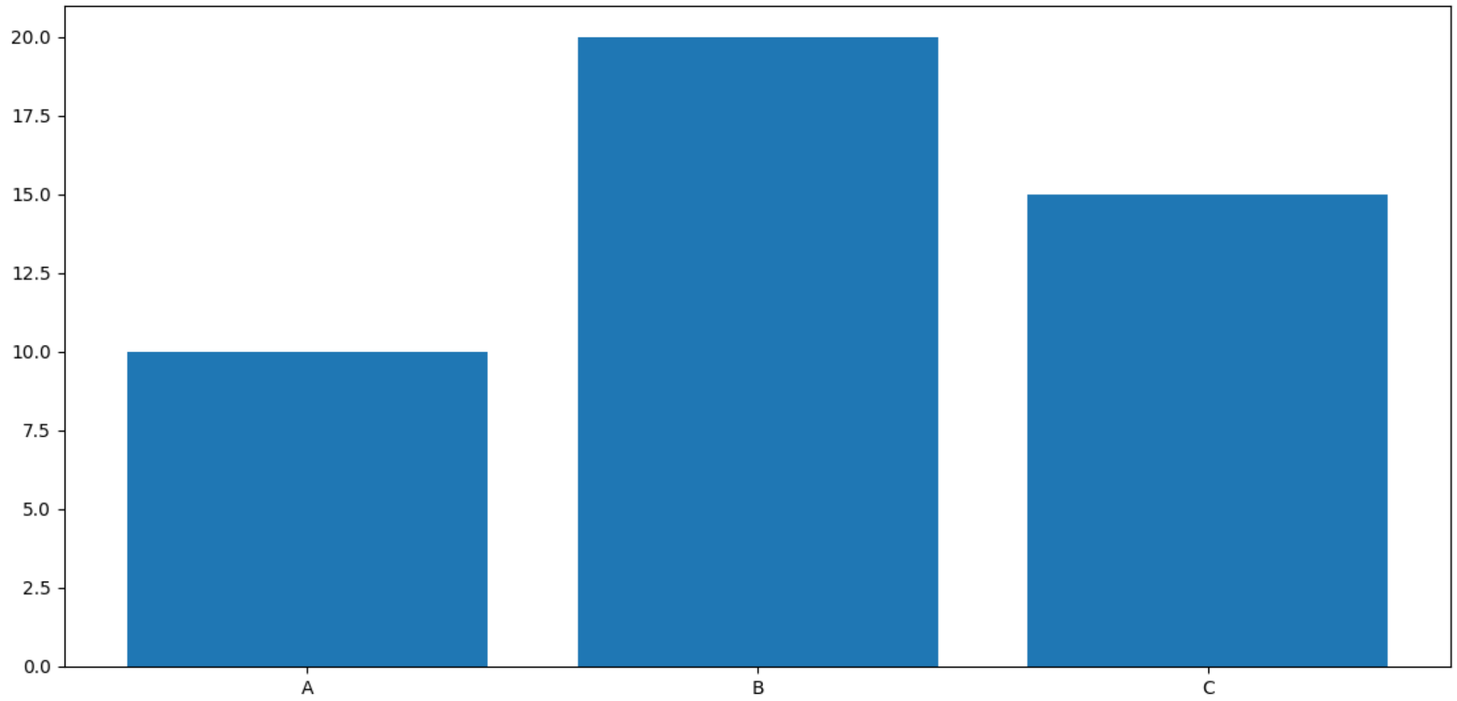
Output: A straight line increasing from left to right

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Plotting Data using Matplotlib

2. Bar Graph

Used to compare **categorical data**



Example Code

```
import matplotlib.pyplot as plt
categories = ['A', 'B', 'C']
values = [10, 20, 15]
plt.bar(categories, values)
plt.show()
```

Output: Vertical bars showing comparison

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Plotting Data using Matplotlib

3. Histogram

Used to show **frequency distribution of data**

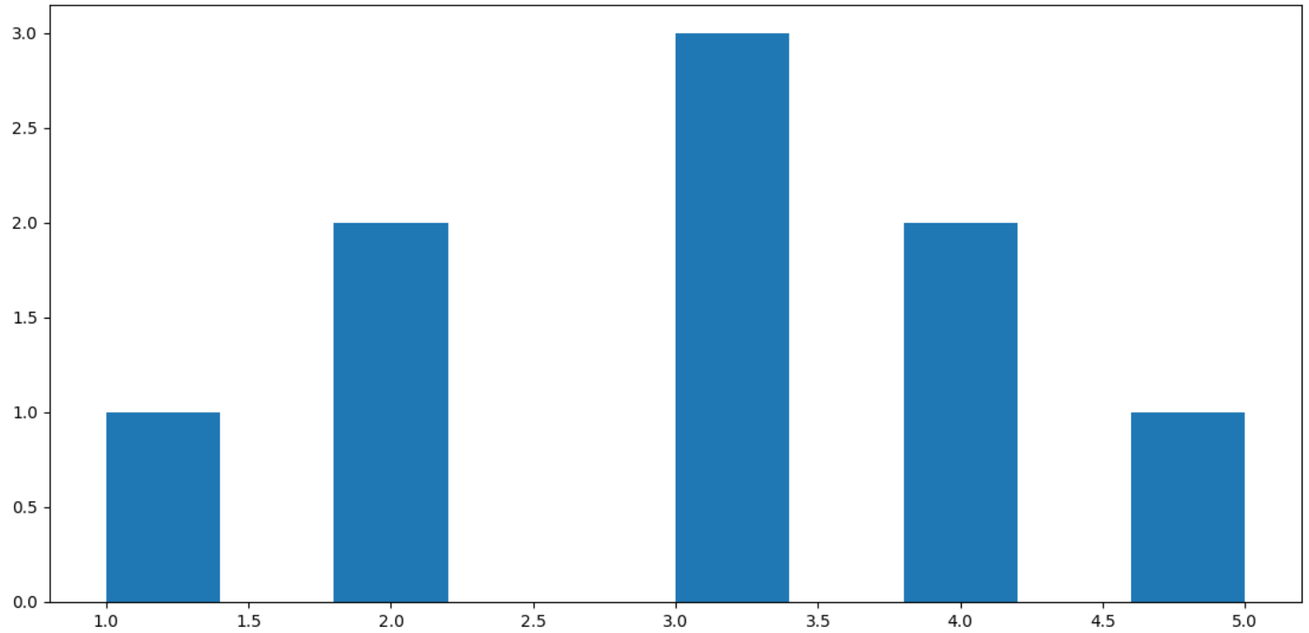
Example Code

```
import matplotlib.pyplot as plt

data = [1, 2, 2, 3, 3, 3, 4, 4, 5]

plt.hist(data)

plt.show()
```

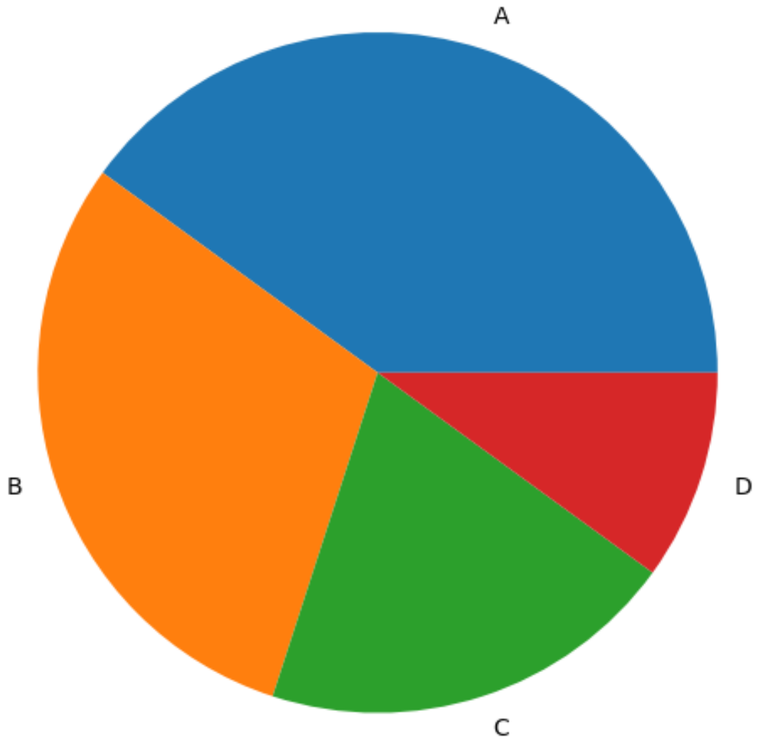


Output: Bars showing how often values occur

Plotting Data using Matplotlib

4. Pie Chart

Used to show **percentage/proportion**



Example Code

```
import matplotlib.pyplot as plt
values = [40, 30, 20, 10]
labels = ['A', 'B', 'C', 'D']
plt.pie(values, labels=labels)
plt.show()
```

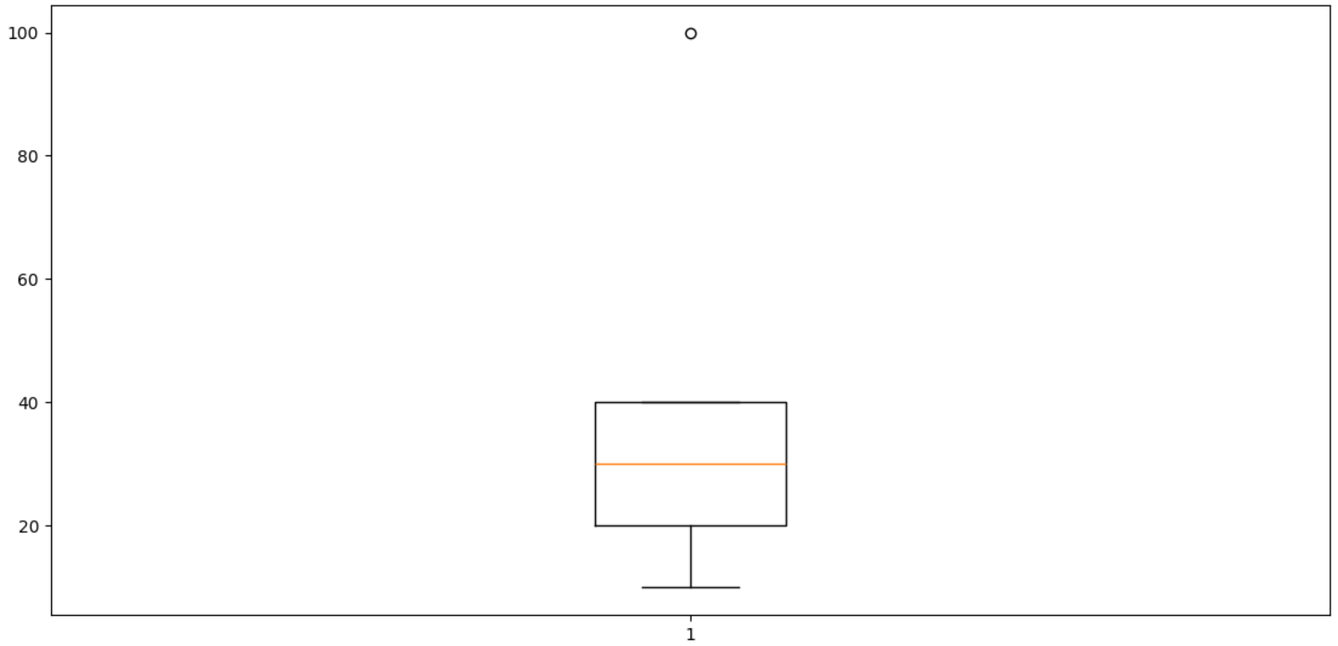
Output: Circular chart divided into parts

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Plotting Data using Matplotlib

5. Box Plot

Used to show **distribution and outliers**



Example Code

```
import matplotlib.pyplot as plt
data = [10, 20, 30, 40, 100]
plt.boxplot(data)
plt.show()
```

Output: Box showing median and one outlier (100)

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

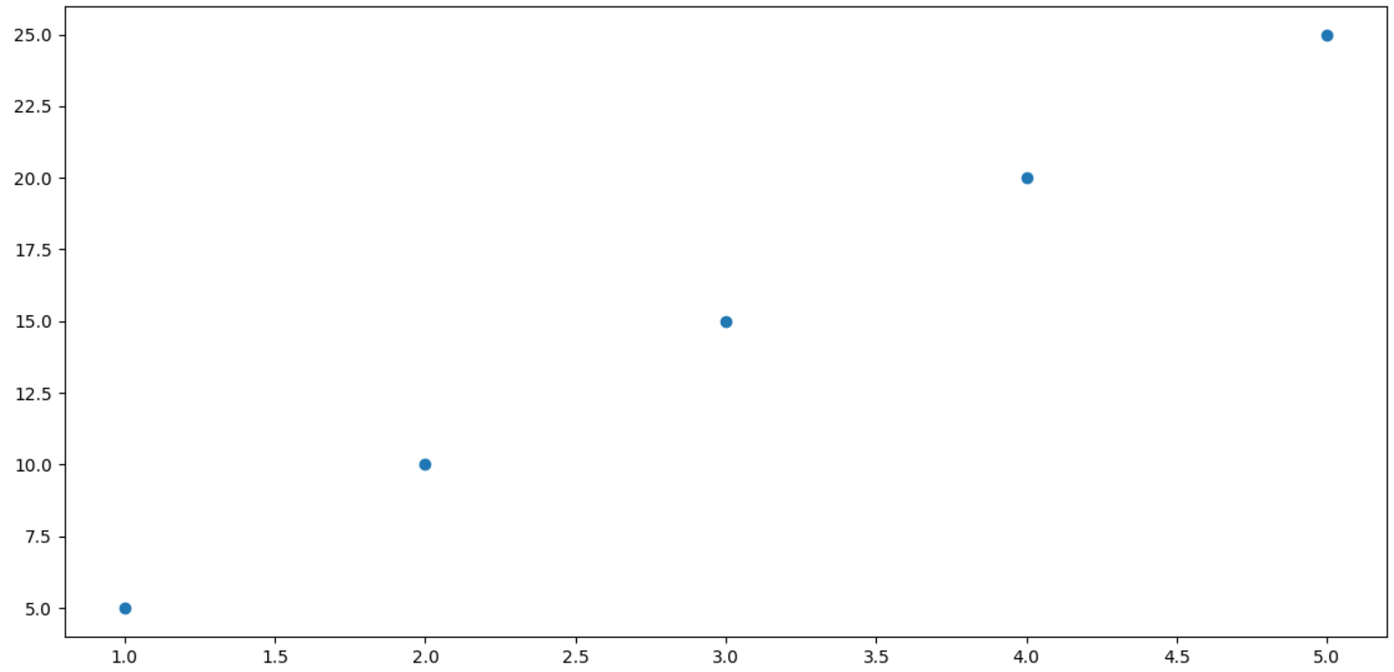
Plotting Data using Matplotlib

6. Scatter Plot

Used to show **relationship between two continuous variables**

Example Code

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [5, 10, 15, 20, 25]
plt.scatter(x, y)
plt.show()
```



Output: Points plotted on graph

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Plotting Data using Matplotlib

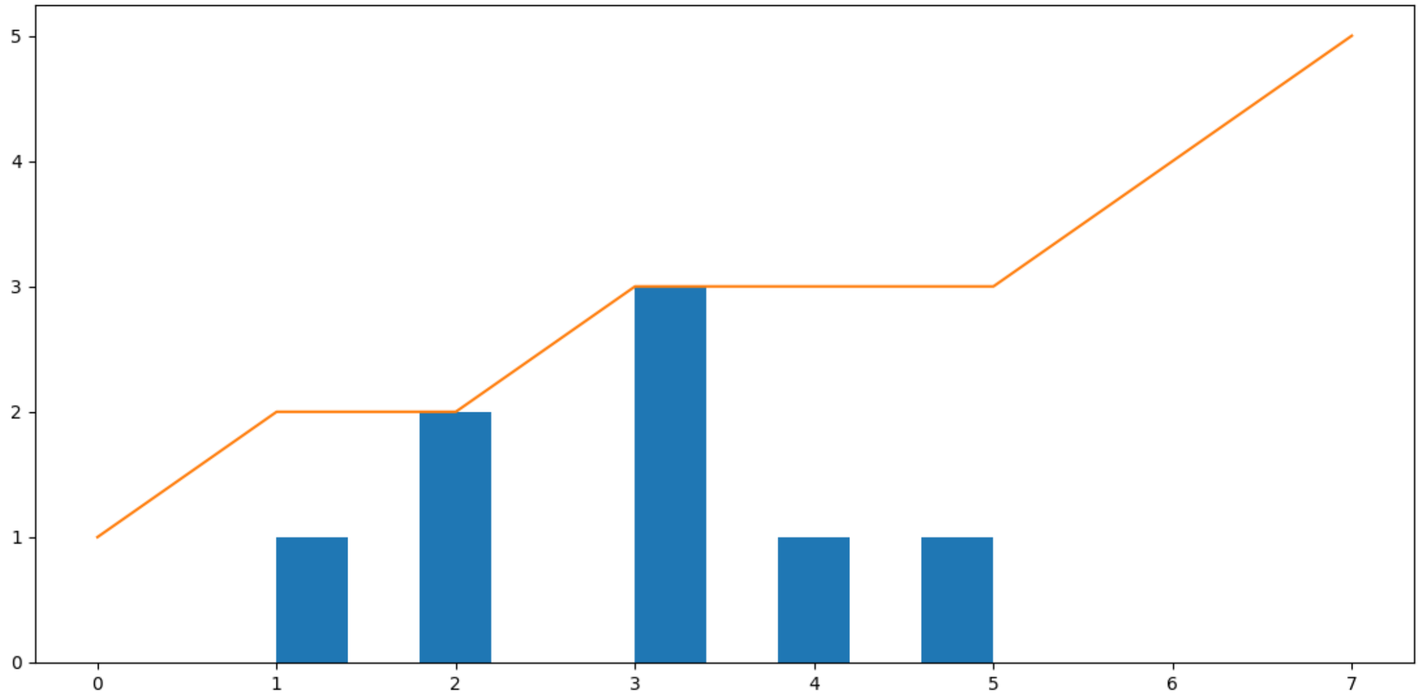
7. Frequency Polygon

It is a line representation of frequency distribution

Example Code

```
import matplotlib.pyplot as plt
data = [1, 2, 2, 3, 3, 3, 4, 5]
plt.hist(data)
plt.plot(sorted(data))
plt.show()
```

Output: Histogram + line showing trend



By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Plotting Data using Matplotlib

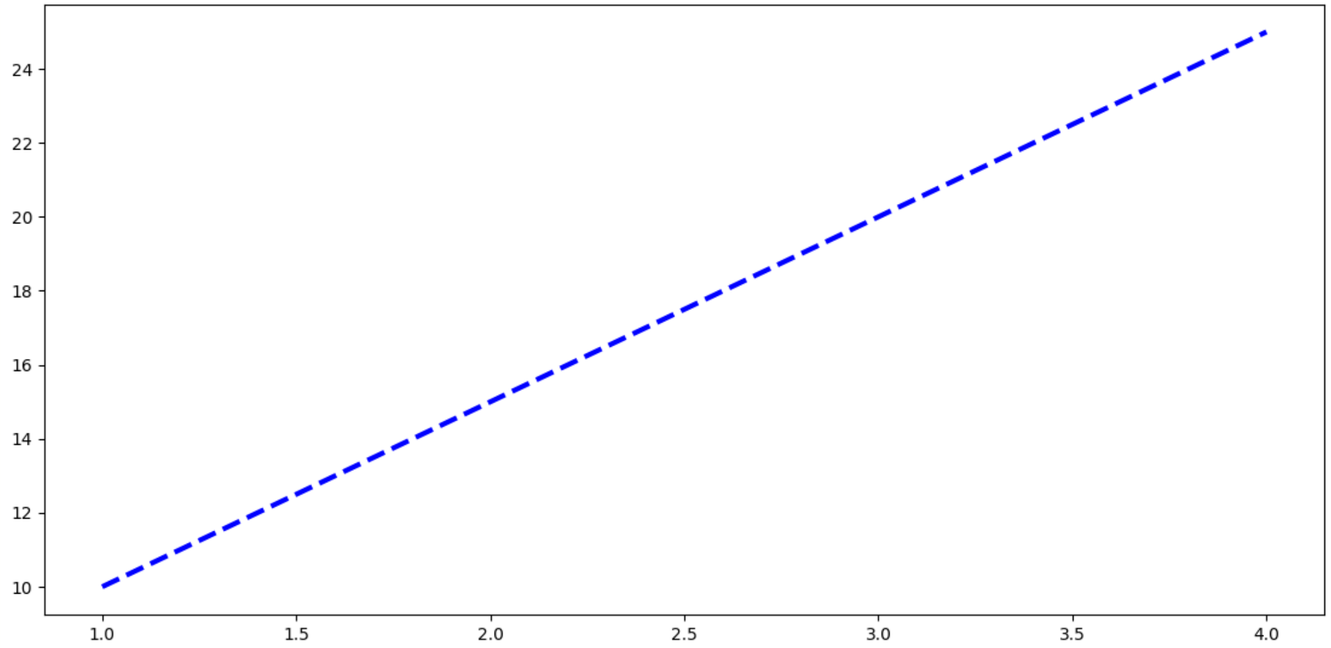
Customizing Plots: Customizing makes graphs **clear, attractive and informative**

8. Color, Style, and Width

color → line color

linestyle → dashed, dotted, solid

linewidth → thickness



Exampe Code 1

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [10, 15, 20, 25]
plt.plot(x, y, color='blue', linestyle='dashed', linewidth=3)
plt.show()
```

Output: Blue dashed thick line

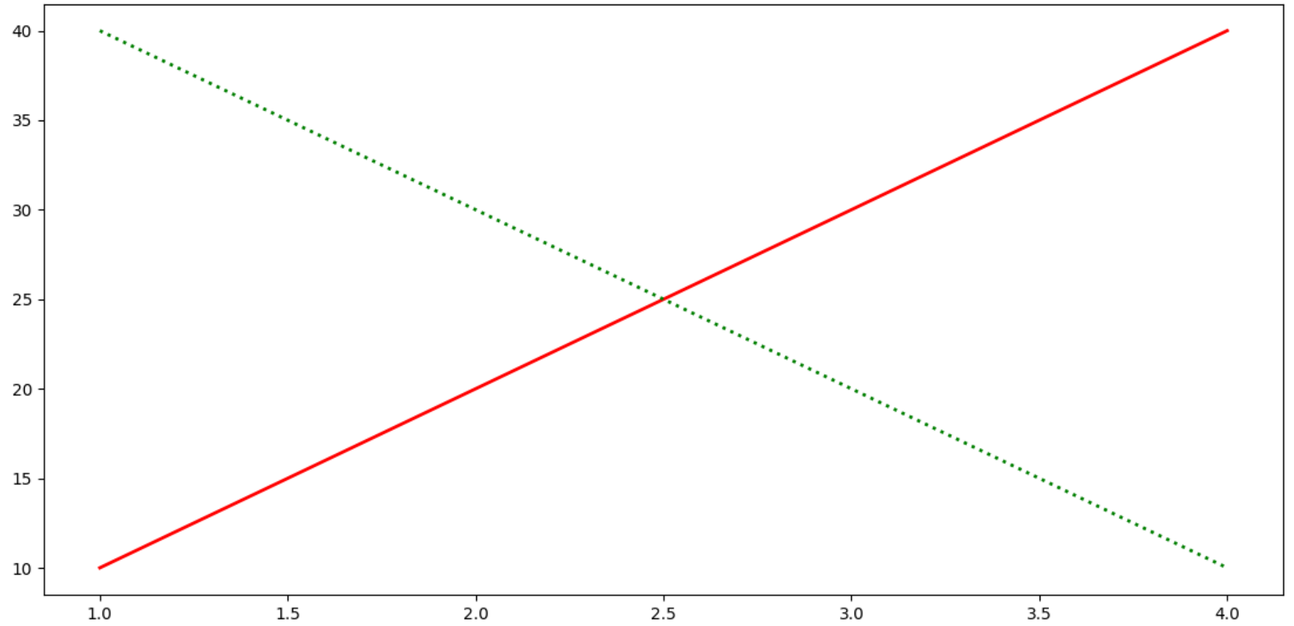
By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Plotting Data using Matplotlib

Example 2 (Multiple Styles)

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
plt.plot(x, [10, 20, 30, 40], color='red', linestyle='solid', linewidth=2)
plt.plot(x, [40, 30, 20, 10], color='green', linestyle='dotted', linewidth=2)
plt.show()
```

Output: Two lines with different styles



Plotting Data using Matplotlib

9. Labels, Title, and Legend

xlabel() → X-axis name

ylabel() → Y-axis name

title() → graph heading

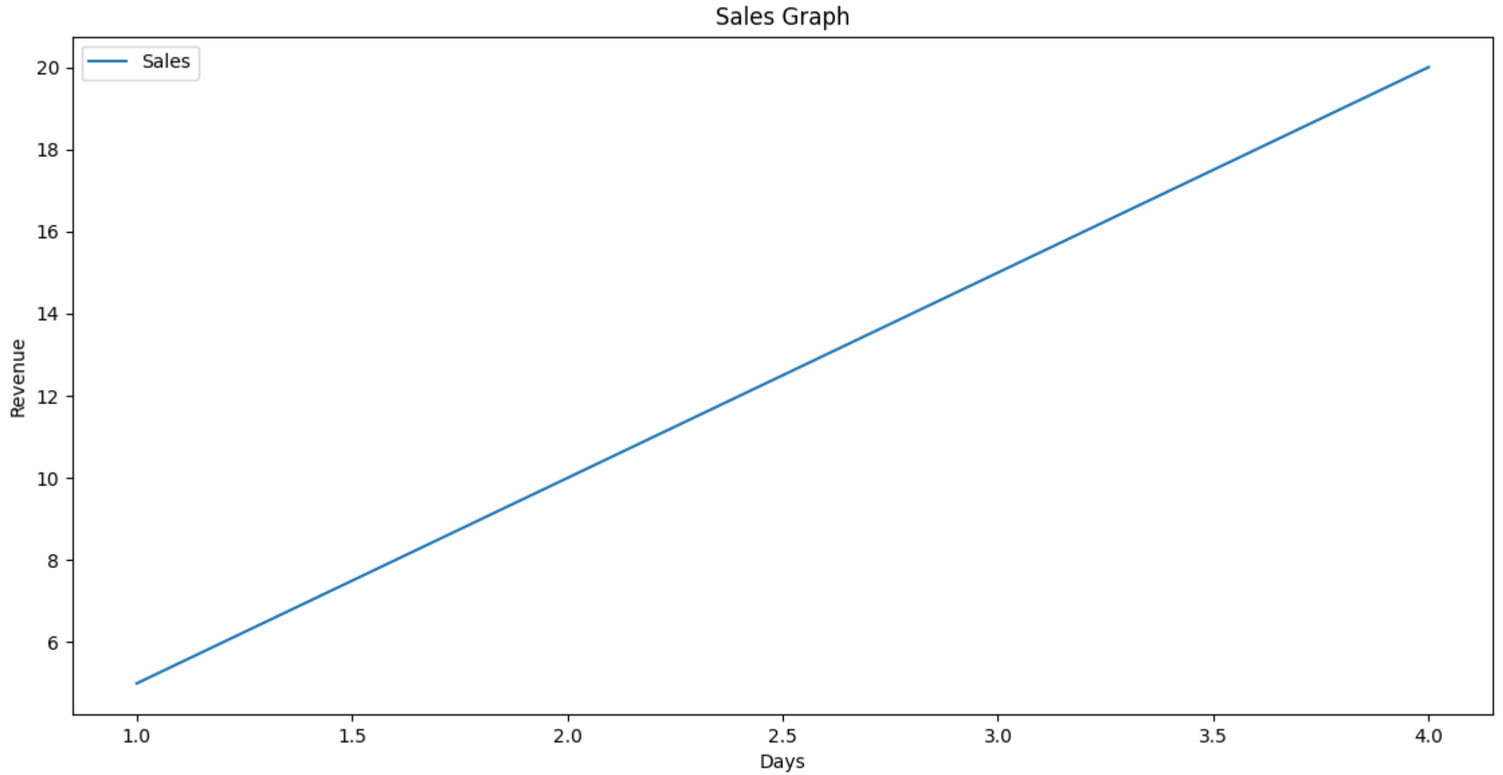
legend() → identifies lines

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Plotting Data using Matplotlib

Example Code 1

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [5, 10, 15, 20]
plt.plot(x, y, label='Sales')
plt.xlabel('Days')
plt.ylabel('Revenue')
plt.title('Sales Graph')
plt.legend()
plt.show()
```



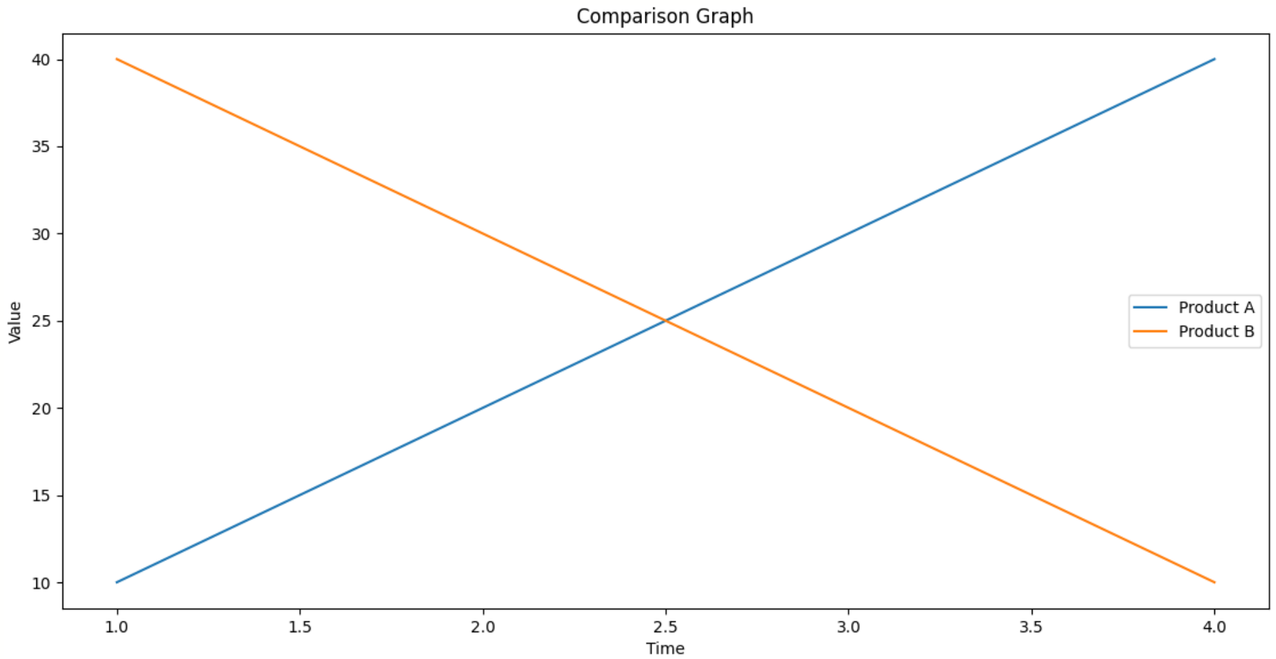
Output: Graph with labels and legend

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Plotting Data using Matplotlib

Example Code 2

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
plt.plot(x, [10, 20, 30, 40], label='Product A')
plt.plot(x, [40, 30, 20, 10], label='Product B')
plt.xlabel('Time')
plt.ylabel('Value')
plt.title('Comparison Graph')
plt.legend()
plt.show()
```



Output: Two lines with legend showing both labels

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

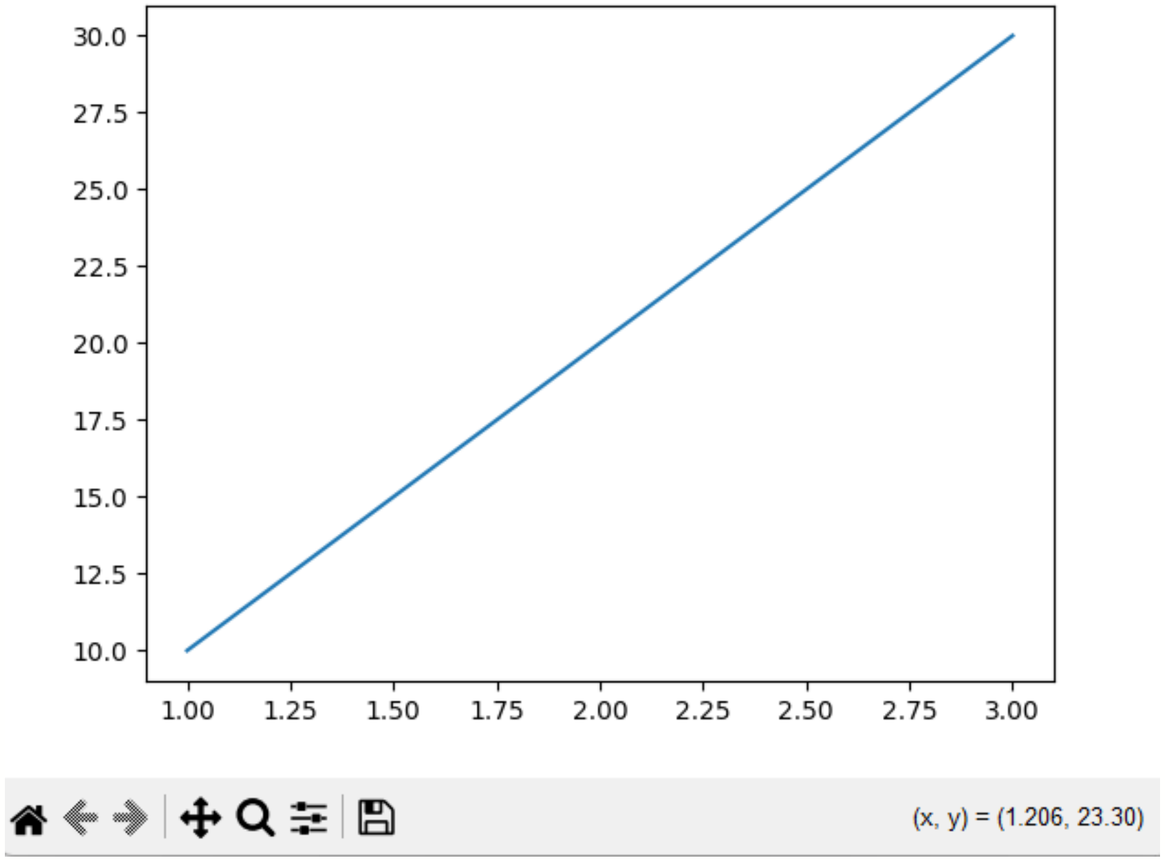
Plotting Data using Matplotlib

10. Saving Plot

Used to save graph as image

Example Code

```
import matplotlib.pyplot as plt
x = [1, 2, 3]
y = [10, 20, 30]
plt.plot(x, y)
plt.savefig('my_graph.png')
plt.show()
```



Output: Graph saved as image file

Societal Impacts

Digital Footprint

Digital footprint means the **record or trail of data** that a person leaves while using the internet.

It includes:

- Social media activity
- Browsing history
- Online transactions

Important points:

- Once data is online, it is difficult to remove
- It can affect reputation and privacy
- Companies and websites may track user behavior

Example:

Posting photos, comments, or liking posts on social media creates your digital footprint.

Societal Impacts

Net Surfing Etiquettes

These are rules to follow while using the internet responsibly.

- Do not open unknown or suspicious links
- Avoid downloading from untrusted websites
- Do not share personal information
- Use secure websites (https)

Example:

If you receive an unknown email link, avoid clicking it.

Societal Impacts

Social Media Communication Etiquettes

Guidelines for respectful communication online:

- Respect others' privacy
- Avoid spreading fake news
- Be polite and respectful
- Think before posting
- Use privacy settings properly

Example:

Before sharing a post, verify if the information is correct.

Societal Impacts

Data Protection

Data protection means **keeping personal and sensitive data safe from misuse**.

Methods:

- Encryption (protecting data using codes)
- Strong passwords
- Regular software updates
- Security audits

```
password = input("Enter password: ")
if len(password) >= 8:
    print("Strong password")
else:
    print("Weak password")
```

Societal Impacts

Intellectual Property Rights (IPR)

IPR protects creations of the human mind such as:

- Inventions
- Designs
- Artistic works
- Software

Types of IPR Violations

- Plagiarism → copying someone's work
- Copyright infringement → using content without permission

Example:

Copying code from internet and submitting as your own is plagiarism.

Societal Impacts

Licensing and Copyright

- Licensing → permission to use software/content
- Copyright → legal right of creator

Types of licenses:

- Proprietary license
- Creative Commons

Societal Impacts

Free and Open Source Software (FOSS)

FOSS allows users to:

- Use
- Modify
- Distribute software freely

Advantages:

- Free of cost
- Encourages collaboration
- Transparency

Example:

Linux operating system

Societal Impacts

Cybercrime

Cybercrime means illegal activities done using computers or internet.

Examples:

- Hacking
- Phishing
- Cyberbullying

Hacking

Unauthorized access to computer systems.

Societal Impacts

Phishing

Fake messages/emails used to steal personal data.

Example:

```
# Simple simulation of phishing awareness
email = "bank-update-link.com"
if "bank" in email:
    print("Be careful: could be phishing")
```

Cyberbullying

Harassing or threatening someone online.

Societal Impacts

Cyber Laws

Cyber laws are rules to prevent cybercrime.

Indian IT Act

- Regulates digital activities
- Provides legal recognition to electronic records
- Defines punishment for cybercrimes

Preventing Cybercrime

- Use strong passwords
- Enable two-factor authentication
- Keep software updated
- Avoid suspicious links

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Societal Impacts

E-Waste

- E-waste includes discarded electronic devices like:
- Computers
- Mobile phones
- Batteries

Hazards of E-Waste

- Contains toxic materials
- Causes environmental pollution
- Affects human health

Management of E-Waste

- Recycling
- Reusing devices
- Proper disposal

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Societal Impacts

Health Concerns Related to Technology

Effects on Eyesight

- Long screen time causes eye strain
- Dry eyes and blurred vision

Physiological Issues

- Lack of physical activity
- Poor posture
- Obesity

Societal Impacts

Ergonomic Aspects

- Proper sitting posture
- Screen at eye level
- Take regular breaks

Project Based Learning

Project Based Learning

Project Based Learning is a method in which students **learn by actively working on real-world projects**. It helps in developing practical knowledge, problem-solving skills, and teamwork ability.

Approaches for Solving Projects

Different approaches are used to solve a project effectively:

1. Problem-Based Approach

Focuses on identifying a problem and finding its solution.

Example: Creating an app to track daily expenses.

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Project Based Learning

2. Inquiry-Based Approach

Involves asking questions, researching, and discovering answers.

Example: Studying how social media affects students.

3. Design-Based Approach

Focuses on designing and creating a product or model.

Example: Building a website or mobile application.

4. Collaborative Approach

Involves working in groups and sharing ideas.

Example: Team project to develop a school management system.

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Project Based Learning

Steps in Project-Based Learning

A project is completed through the following steps:

1. Identifying the Problem

Choose a real-life problem or topic.

2. Planning

Decide objectives, tools, and timeline.

3. Research

Collect information from books, internet, etc.

4. Designing the Solution

Create a plan or model to solve the problem.

5. Implementation

Develop the actual project (coding, model, etc.)

6. Testing

Check for errors and improve the project.

7. Presentation

Present the project results clearly.

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Project Based Learning

Teamwork

Teamwork means working together to achieve a common goal.

Importance:

Improves efficiency

Encourages idea sharing

Reduces workload

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Project Based Learning

Components of Teamwork

1. Communication

Clear exchange of ideas among team members.

2. Collaboration

Working together and supporting each other.

3. Responsibility

Each member should complete assigned tasks.

4. Coordination

Proper management of tasks and time.

5. Problem-Solving

Handling issues together effectively.

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

Project Based Learning

Simple Example (Mini Project Idea)

Project: Student Marks Analyzer

Example Code

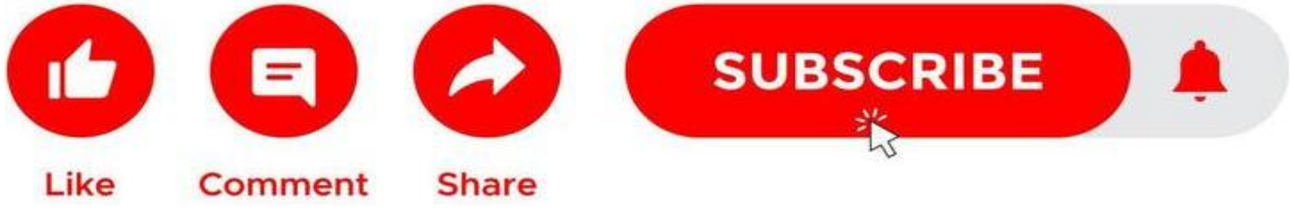
```
# Simple project: Calculate average marks
marks = [80, 75, 90, 85]
total = sum(marks)
average = total / len(marks)
print("Total Marks:", total)
print("Average Marks:", average)
```

Output:

```
Total Marks: 330
Average Marks: 82.5
```

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)

KHUSHI FOUNDATION ACADEMY



राष्ट्रीय परीक्षा एजेंसी
National Testing Agency

By :- Ashutosh Srivastav (State Topper B.T.E.U.P)