PYTHON CLASS

MILESTONE 1

Matplotlib API

- **Definition:** Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. The API (Application Programming Interface) refers to the set of functions, methods, and classes that developers use to interact with and control the library.
- **Key components:**

  matplotlib.pyplot: A collection of command-style functions that make Matplotlib behave like MATLAB. It's often used for quick, interactive plot generation.

NUMPY

NumPy stands for Numerical Python and is used to operate efficient computations of arrays and matrices behind the scenes of machine learning models. The building block of Numpy is the array, which is a data structure very similar to the list, with the difference that it provides a huge amount of mathematical functions. In other words, the Numpy array is a multidimensional array object**.**

PANDAS

Pandas is built on top of NumPy and provides powerful, flexible, and easy-to-use data structures for data manipulation and analysis, particularly for tabular data.

HTTP methods: POST and GET

- **Definition:** HTTP (Hypertext Transfer Protocol) methods indicate the desired action to be performed on a resource. GET and POST are two of the most common.
- **GET method:**

  Purpose: Used to request and retrieve data from a server.

  Payload: Typically does not have a request body (payload).
- **POST method:**

  Purpose: Used to send data to a server to create or update a resource.

  Payload: Carries a request body (payload) containing the data to be sent, such as form data or a file.

pip install

- **Definition:** pip is the standard package manager for Python. The pip install command is used to install packages and libraries from the Python Package Index (PyPI) or other sources.

Payload

- **Definition:** In the context of APIs, the payload is the actual data sent in a request body or received in a response body.
- **Purpose:** It contains the core information needed for the API to perform an action. For example, when creating a new user, the payload holds the user's name, email, and password.

- **Usage with HTTP methods:** POST, PUT, and PATCH requests commonly use a payload, while GET and DELETE typically do not.

Git

Git is a distributed version control system for tracking changes in source code during software development.

- **git checkout:**

  Function: Primarily used to switch between different branches.

  .

- **git push:**

  Function: Uploads local branch commits to the corresponding remote repository. This makes your changes visible to other collaborators.

  Common use: git push origin <branch-name>.

- **git branch:**

  Function: Manages branches in your local repository.

  Commands:

  git branch: Lists all local branches.

  git branch <branch-name>: Creates a new branch.

- **git fetch:**

  Function: Downloads new data (commits, branches, tags) from a remote repository but does not integrate them into your local working directory.

  Purpose: Allows you to review changes made by others before merging them. It updates your remote-tracking branches (e.g., origin/main).

- **Git add:**
  Stages changes from your working directory to the staging area (also known as the index). This tells Git which specific changes you want to include in the next commit.
- **Git commit**:
  Records the staged changes as a new snapshot in your local repository's history. Each commit has a unique identifier and a commit message explaining the changes.

Relational database

- **Definition:** A database that stores and provides access to data points that are related to one another.
- **Structure:** Data is organized into tables, with each table consisting of rows (records) and columns (attributes).

Normalization

- **Definition:** The process of organizing data in a relational database to minimize redundancy and improve data integrity.
- **Normal forms:**

  First Normal Form (1NF): Each column contains only atomic (indivisible) values, and there are no repeating groups of columns.

  Second Normal Form (2NF): Is in 1NF, and all non-key attributes are fully dependent on the primary key.

CRUD

- **Definition:** An acronym for **C**reate, **R**ead, **U**pdate, and **D**elete, representing the four basic functions for interacting with persistent storage, such as a database.
- **Create:** The operation of adding new data or records. In SQL, this corresponds to the INSERT statement.
- **Read:** The operation of retrieving data. In SQL, this is the SELECT statement.
- **Update:** The operation of modifying existing data. In SQL, this is the UPDATE statement.
- **Delete:** The operation of removing data or records. In SQL, this is the DELETE statement.

API

API, or **Application Programming Interface**, is a set of rules and protocols that allows different software applications to communicate and exchange information. In simpler terms, it's a messenger that takes a request from one application, delivers it to another, and then returns the response. This process happens constantly in the background, powering many of the digital experiences.

FLASK API

Flask is a lightweight WSGI microframework, emphasizing simplicity and flexibility. It provides a core set of features and allows developers to choose and integrate extensions for additional functionalities like database interaction, authentication, and form handling.

FAST API

FastAPI is more opinionated than Flask, guiding developers towards best practices for API development through its use of type hints and built-in features. This can simplify development for complex APIs but might have a steeper initial learning curve for those unfamiliar with modern Python features.

FSTRING

Strings are ordered collections of characters. Each character in a string occupies a specific position, accessible via indexing.

```
Price=59

Txt=f"the price is{price}dollars"

Print(txt)
```

MILESTONE 2

ARTIFICIAL INTELIIGENCE

AI is a method or set of techniques that allow systems to learn patterns, make decisions, and act autonomously without direct human intervention, mimicking human intelligence.

Key subfields include:

Machine Learning (ML)

Deep Learning (DL)

Natural Language Processing (NLP)

Reinforcement Learning (RL)

Typical steps in an AI project:

Collect data

Prepare and engineer features

Train a model

Evaluate on a holdout dataset

Deploy to production and monitor

Common applications:

Demand forecasting

Dynamic pricing

Customer analytics

LSTM and NLP Concepts

LSTM (Long Short-Term Memory) is a special type of Recurrent Neural Network (RNN) designed to remember information for long periods, which works well with time-series data and in Natural Language Processing (NLP) tasks.

It has three gates: input, forget, and output gate.

NLP concepts mentioned include Sentiment analysis, Tokenization (breaking text into smaller units), Stemming (reducing words to their root form), Lemmatisation (reducing words to their base or dictionary form), and Part-of-speech (POS) tagging.

Named entity recognition (NER) is also listed, which involves identifying entities like persons, organizations, and dates.

SVM and TF-IDF

SVM (Support Vector Machine) is a supervised machine learning technique used for classification, regression, and outlier detection, which works by finding the optimal separation line between data points.

The data points closest to the separation line are called support vectors.

Hard margin implies perfect separation with no misclassification, while soft margin allows some misclassification.

TF-IDF (Term Frequency-Inverse Document Frequency) is a numerical statistic that reflects how important a word is to a document in a collection or corpus.

TF measures how often a word appears in a document, and IDF measures how rare the word is across all documents.

Reinforcement Learning Concepts

Human in the loop and without human interaction

The image outlines two main concepts for Reinforcement Learning (RL) trials:

Human in the loop: This involves human interaction during the learning process.

Without human interaction: This refers to fully autonomous learning where no human involvement is needed.

AI Agents Goals and Actions

Check stock levels automatically, predict when items will be out of stock, send alert to managers, auto generate purchase orders, and update demand forecasts daily

The general goals for AI agents listed are understanding, planning, taking actions, and a feedback loop. Specific agent actions provided as examples include:

Checking stock levels automatically

Predicting when items will be out of stock

Sending alerts to managers

Automatically generating purchase orders

Updating demand forecasts daily

Git Version Control Basics

A version control system

Git is described as a version control system. GitHub is noted as a cloud hosting service for Git repositories.

Types of Agents

Reactive agent and Goal driven agent

The two types of agents mentioned are:

Reactive agents

Goal driven agents

Multi driven agents

Agent with Tools

Functions which have some of the main logics

Tools for an agent are defined as functions that contain some of the main logics, which the agent can decide to select and use. In the Autogen framework, userproxy and assistant agents are mentioned.

OpenAI Agent Workflow

The left side describes the core logic of a user proxy agent and assistant agent interaction:

User Proxy Agent: This agent receives the initial task from the user, possibly writes a plan, and provides the context of the workflow to the assistant.

Assistant Agent: This agent loves the task and works memory for the assistant.

Key Steps:

The user prompt triggers the agent system.

The assistant agent performs actions and provides output back to the user or user proxy.

Code snippets show how to initialize the client (from openai import OpenAI; client = OpenAI()) and make a chat completion request using a model like gpt-4o-mini with a specified system and user message.

The example shows generating a two-line poem about "chai".

Google GenAI Function Calling

The right side focuses on implementing function calling (tools) with Google's GenAI models:

Setup: Shows how to import and initialize the client (from google import generativeai, genai; model = genai.GenerativeModel(...)).

User Interaction: Describes a user sending a message, the model generating a response, and the chat completion returning the next message.

Function Definition: Demonstrates how to define a function (get-temperature) with parameters (e.g., city) that the model can use to retrieve external information.

Model Integration: The model is initialized with the defined tools (model = genai.GenerativeModel(..., tools=[& functions])).

functions = {

"name": "get_temperature"

"description": "returns temperature of a city"

"parameters": {

"type": "object"

"properties": {"city": {“type": "string"}}

}}

Using function:

• model = genai.GenerativeModel("gemini-1.5-flash", tools=[functions])

• stream = Tru

Flow

Text → Tokens Input text is broken into tokens.

Tokens → Embeddings Tokens are converted into dense vectors.

Embeddings → Transformer Layers Transformer processes embeddings.

Attention Mechanism Model identifies which words are important.

Output → Text (decoded) Final generated/decoded text is returned.

Temperature

Controls creativity of output.

Range: 0.0 to 1.0 7.

Rate Limits & Quotas

Each API has usage limits depending on model and plan.

Limits apply to:

Requests per minute

Tokens per minute

Daily quota

Decision Making

The notes suggest that a decision-making agent might use tools like a calculator or unit converter. An example provided is a simple math problem: "what is \(2+2\)", which is likely used as a basic illustration within a larger context of complex AI decision processes.

Logistic Regression

Logistic regression is described as a classification algorithm, used for problems with binary outcomes (e.g., Yes/No, Spam/not spam)

. Equation: The core of the model uses the sigmoid (logistic) function to output a probability between 0 and 1: \(P=\sigma (z)=\frac{1}{1+e^{-z}}\) where \(z\) is a linear combination of input features and weights (\(z=w_{1}x_{1}+w_{2}x_{2}+\dots +b\)).

Decision Rule: A threshold (e.g., 0.5) is used to classify the outcome. If \(p>0.5\), it's classified as class 1; otherwise, it's class 0.Uses:Fraud detectionMedical diagnosisCustomer churn predictionEmail spam detection

Comparison with Decision Tree

Both Logistic Regression and Decision Trees are used for classification tasks. Logistic Regression is a linear model that assumes a linear relationship between input variables and the log-odds of the outcome, and it can struggle with complex, non-linear relationships.Decision Trees are non-linear classifiers that can handle complex relationships and outliers better than logistic regression. They work by creating a hierarchical tree structure of rules to partition the data.

Machine Learning

Mentions algorithms like Random Forest, K-nearest neighbors (KNN), and K-means clustering.

Outlines the steps for K-means clustering: choose features, assign points to centroids, recalculate centroids, and repeat.

Includes Python import statements from libraries like sklearn.

SQL

Defines SQL as a structured query language.

Lists key SQL commands: SELECT, UPDATE, DELETE, INSERT INTO, CREATE, ALTER, DROP.

Provides examples of SQL syntax using clauses like WHERE, BETWEEN, and DISTINCT

Create

Creates a new database

Alter

Modifies the structure of a database object

Drop

Permanently deletes a database object

Truncate

Deletes all rows from a table quickly

SELECT

Extracts data from a database

UPDATE

Updates data in a database

DELETE

Deletes data from a database

INSERT

Inserts new data into a database

ALTER: Mentioned in the context of "Modifis adb" (modifying a database).

SELECT: Notes include examples for selecting all columns (SELECT column names FROM tablename), distinct values (SELECT DISTINCT Column name FROM table name), and counting distinct values (SELECT COUNT(DISTINCT Country) FROM table Nam).

WHERE clause: Explanations for filtering data using the WHERE clause, including syntax for column names, values, and operators.

Operators: A list of operators such as =, !=, >, <, >=, <=, Between, Like, AND, OR, and NOT.

Examples: Specific examples are provided for using LIKE (name like y.s), BETWEEN (Salary between 10000 and 50000, WHERE Price BETWEEN 500 AND 1500).

Some examples

ASC DEsC

SELECT * FROM Students

ORDER BY Course ASC

SELECT * FROM students

ORDER BY marks ASC;

SELECT product_name, price

FROM products

ORDER BY price ASC;

SELECT * FROM Students

ORDER BY Course ASC, Marks DESC;

SELECT * FROM Employees

ORDER BY department ASC, name ASC;

Ascending order by default

asc

INSERT INTO

insert into employees (id,name) VALUES(1,naini)

INSERT INTO Table name(column name1, column name 2,)values('value 1',' value2');

Update;

UPDATE students

SET marks = 95

WHERE id = 10;

Update students SET name = 'shakthi', age = '23' WHERE id=06;

UPDATE employees

SET salary = 50000

WHERE id = 3;

UPDATE students

SET name = 'Sneha R', age = 20

WHERE id = 3;

Delete

DELETE FROM students

WHERE id = 2;

MIN()

MAX()

COUNT

AVG

SUM

Like Operator:

a%

%a

%or%

-r%

a_%

a__%

a%0

select fname from employee where fname like 'a_%';

SELECT * FROM words

WHERE name LIKE 'a__%';

```
SELECT * FROM words

WHERE name LIKE 'a%0';


SELECT * FROM Employees

WHERE Name LIKE 'a_%';


SELECT *

FROM employees

WHERE name LIKE "a_%" ;


SELECT column


SELECT *
```

a% - apple

%e = apple

 %pl% = apple

 _p% = apple

 a%e = apple

IN

NOT IN

```
SELECT *

FROM employees

WHERE department IN ('HR','Finance');


SELECT * FROM custormers where country IN ('Germany', "France','UK')


SELECT * FROM students

WHERE city IN ('Mumbai', 'Delhi');
```

Joins

Inner Join

Cross Join

left join

right join

Macthing values in both the tables

table 1:

std id, std name, address

table 2:

subject id, std id, subject name

inner join result will be the std id

UNION

UNION ALL

GROUP BY

we use them for aggregate functions

Having clause

SELECT Product, COUNT(*)

FROM Sales

GROUP BY Product

HAVING COUNT(*) >= 2;

select count(customer id), country from customers Group BY country Having count(costomer id)>5;

MILESTONE 3

String Method:

strip()

split()

join()

replace()

upper()

lower()

startswith()

endswith()

find()

isdigit()

isalpha()

list Methods:

append()

extend()

insert()

remove()

pop()

index()

count()

sort()

reverse()

Dictionary Method:

get()

keys()

values()

items()

update()

pop()

popitem()

clear()

Set Methods:

add()

remove() - removes an elemet

discard() - removes an element if present.

pop() - Removes and returns an elements

clear() - Removes all elemets.

union()

intersection()

difference()

File I/O Methods:

open()

read()

readline()

readlines()

write()

writelines()

close()

General Purpose:

len()

range()

print()

type()

id()

sorted()

enumerate()

zip()

string

upper()

lower()

capitalize()

strip()

istrip()

rstrip()

replace()

split()

join()

find()

count()

startswith() / endswith()

isalpha()

isnumeric()

isalnum()

isdigit()

swapcase()

format()

open()

Used to open a file.

```
file = open("example.txt", "w")   # Opens file in write mode
file.write("Hello!")
file.close()
```

read()

Reads the entire file content.

```
file = open("example.txt", "r")
content = file.read()
print(content)
file.close()
```

readline()

Reads one line at a time.

```
file = open("data.txt", "r")
```

```
line1 = file.readline()
line2 = file.readline()
print(line1)
print(line2)
file.close()
```

readlines()

Reads all lines and returns a list.

```
file = open("data.txt", "r")
lines = file.readlines()
print(lines)   # Example: ['line1\n', 'line2\n']
file.close()
```

write()

Writes a string to a file.

```
file = open("notes.txt", "w")
file.write("Python is fun!")
file.close()
```

writelines()

Writes multiple lines to a file (list of strings).

```
file = open("items.txt", "w")
file.writelines(["Apple\n", "Banana\n", "Orange\n"])
file.close()
```

close()

Closes a file.

```
file = open("sample.txt", "r")
# ... operations ...
file.close()
```

GENERAL-PURPOSE PYTHON FUNCTIONS

len()

Returns the length of a sequence.

```
text = "Hello"
print(len(text))  # Output: 5
```

range()

Generates a sequence of numbers.

```
for i in range(3):
   print(i)
# Output: 0, 1, 2
```

print()

Displays output.

```
print("Hello Python!")
```

4. type()

Returns the data type.

```
a = 10
print(type(a))   # <class 'int'>
```

id()

Returns the memory address.

```
a = 100
print(id(a)
```

sorted()

Returns a new sorted list.

```
numbers = [5, 2, 9, 1]
print(sorted(numbers))   # [1, 2, 5, 9]
```

enumerate()

Adds index to iterable.

```
fruits = ["apple", "banana", "orange"]
for index, fruit in enumerate(fruits):
   print(index, fruit)
```

zip()

Combines iterables element-wise.

```
names = ["Aru", "Balu", "Chinna"]
marks = [90, 85, 92]

for n, m in zip(names, marks):
   print(n, m)
```

conversion function:

int()

float()

str()

list()

dict()

set()

tuple()

```
a= 1
print(a.float)
```

mathematical functions:

abs() - returns the absolute value.

sum() - sums the items.

min() - return the min.

Max() - return the max.

pow() - raises a number to a power.

round() - rounds a number.

Functional programming tools:

Filter()

Map()

Reduce()

```
add_ten = lambda x:x+10
print(add_ten(5))
def (a,b)
a = lambda a,b : a*b
```

print(a(3,4))

input and output:

input()

forward()

a = input(“enter”)

format()

class and object related:

getattr()

setattr()

hasattr()

delattr()

isinstance()

issubclass()

Miscellaneous:

globals()

locals()

collable()

exec()

eval()

Exception Handling:

try

except

finally

Memory and object Management:

del()

gc.collect()

import gc

working with iterables:

next()

iter()

Decorators and Metaprogramming

staticmethod()

classmethod()

INTERVIEW QUESTIONS

1.What are Python's key feature?

Interpreted, high-level, dynamically typed, object-oriented, cross-platform, supports multiple paradigms, and has extensive libraries.

2.What are python's data types?

int, float, complex, list, str, tuple, set, dict, bool, nonetype

3.what isPEP 8 and why is it important?

python's style guide, ensre consistent, clean and readable code.

4.Explain mutable vs immutable object?

5.what are python's built-in data structures?

6.whare python's memort management features?

Private heap, reference counting, garbage collector

7.Explain indentation in python

complied into bytecode

8. explain namespaces.

mapping of variable names to objects

List vs Tuple differences.

How are sets used?

what are dictionaries?

how to merge dictionaries?

dict1.update(dict2).

shallow vs deep copy?

implement stack & queue

function vs method?

what is *args and kwargs?

what are decorators?

what are closures?

function with access to variables fromenclosing scope.

what is lambda function

multiple vs multilevel inheritance?

monkey patching?

dynamically changing class/module behavious at runtime

syntax vs runtime error

invalid code error and runtime occurs during execution

pip

delete file?

os.remove

atomatic alignment of arrays of differnt shapes is calledboardcasting?

Fuctional consultant/Business Analyst

who will desing a function specification document

Developer/Programmer.

Technical document.

Testing/QA(Quality Assurance)

Test Script

based on function specification.

Adminisation

Network

Database

Application

Security

Product/Project Management

Sales & Marketing

Human Resource

Documenenentention

Support

A * Search Algorithm

Start Node: A
Goal Node: F

Nodes: A, B, C, D, E, F
Edges with weights:
(A, B, 1), (A, C, 4),
(B, D, 3), (B, E, 5),
(C, F, 2),
(D, F, 1), (D, E, 1),
(E, F, 2)

Sort:

arrange in some order

te order many be in two type:

asc or dec

Bubble Sort

Selection Sort

Insertion sort

Merge Sort

Quick Sort

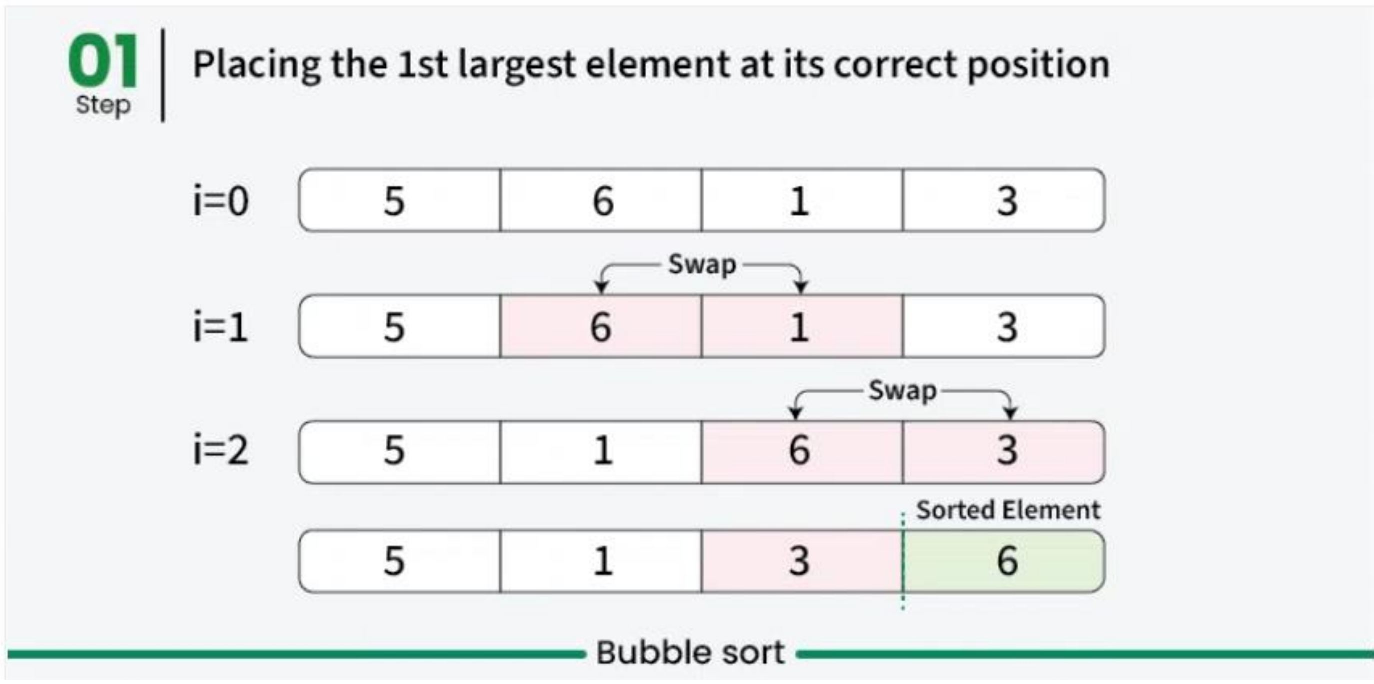
5613

5163

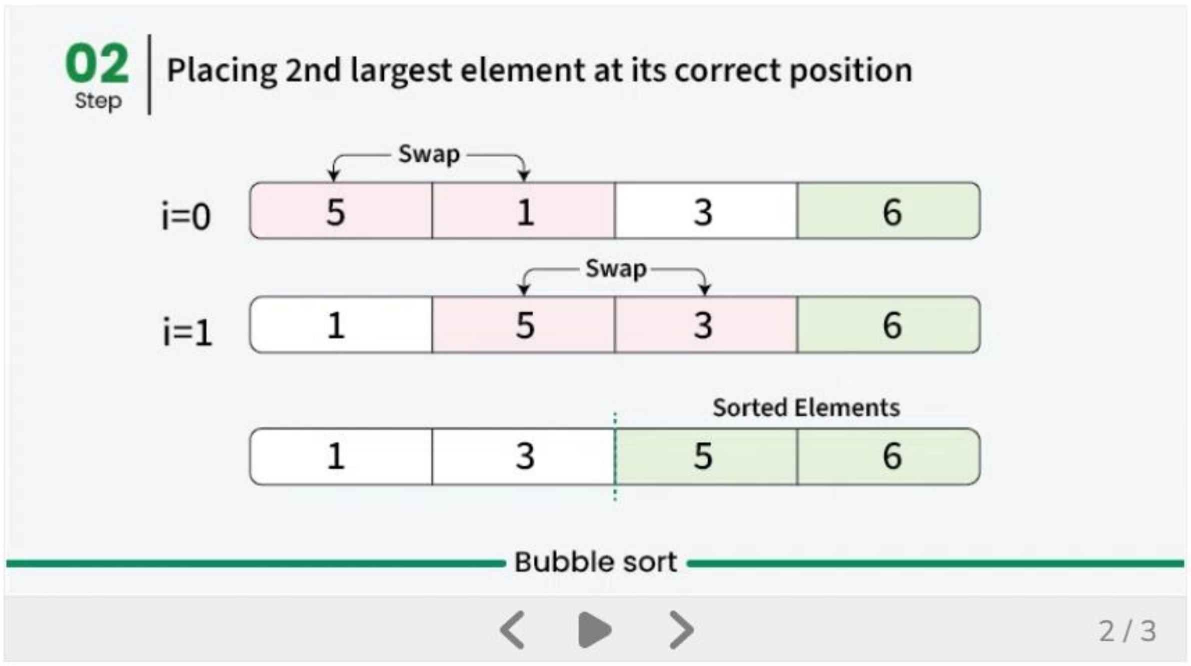
5136

1536

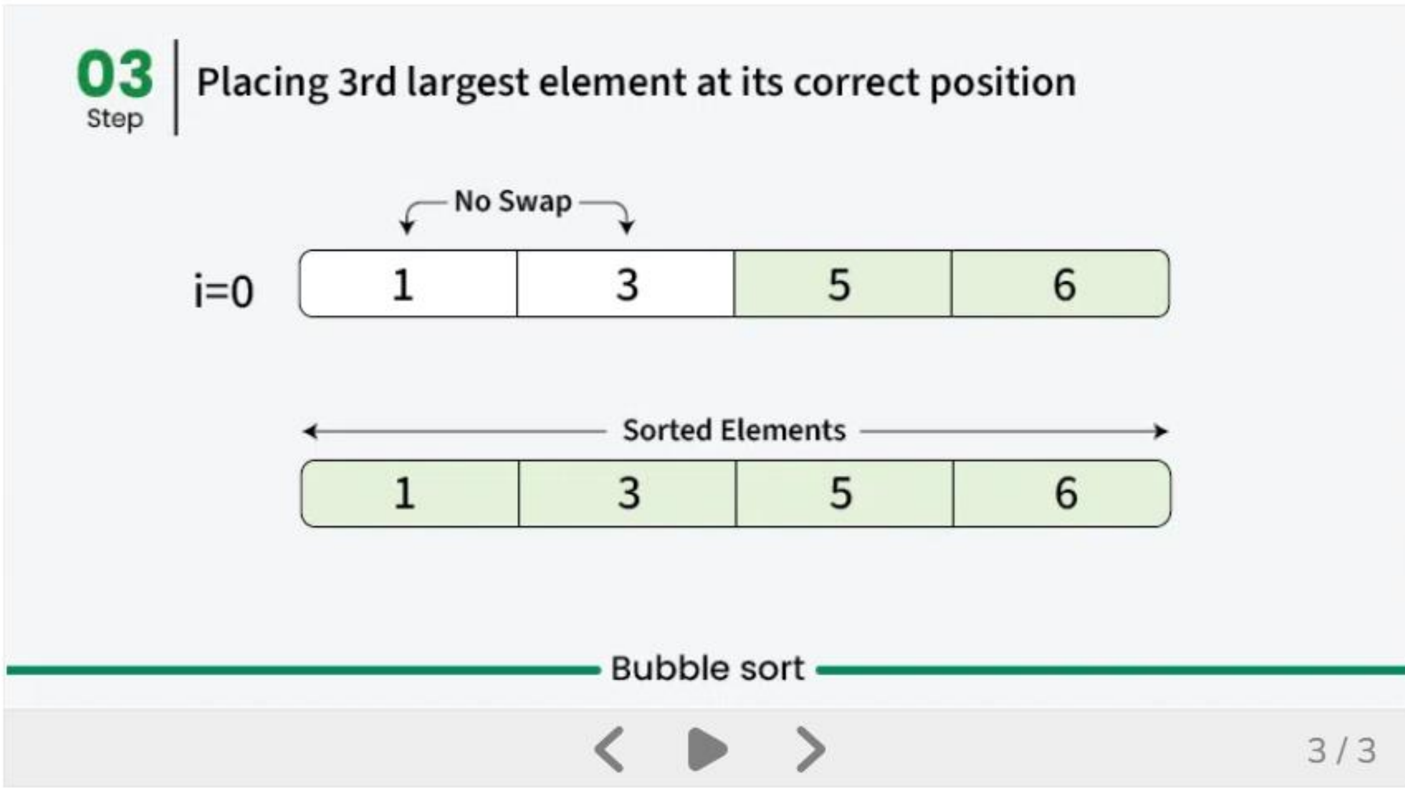
1356

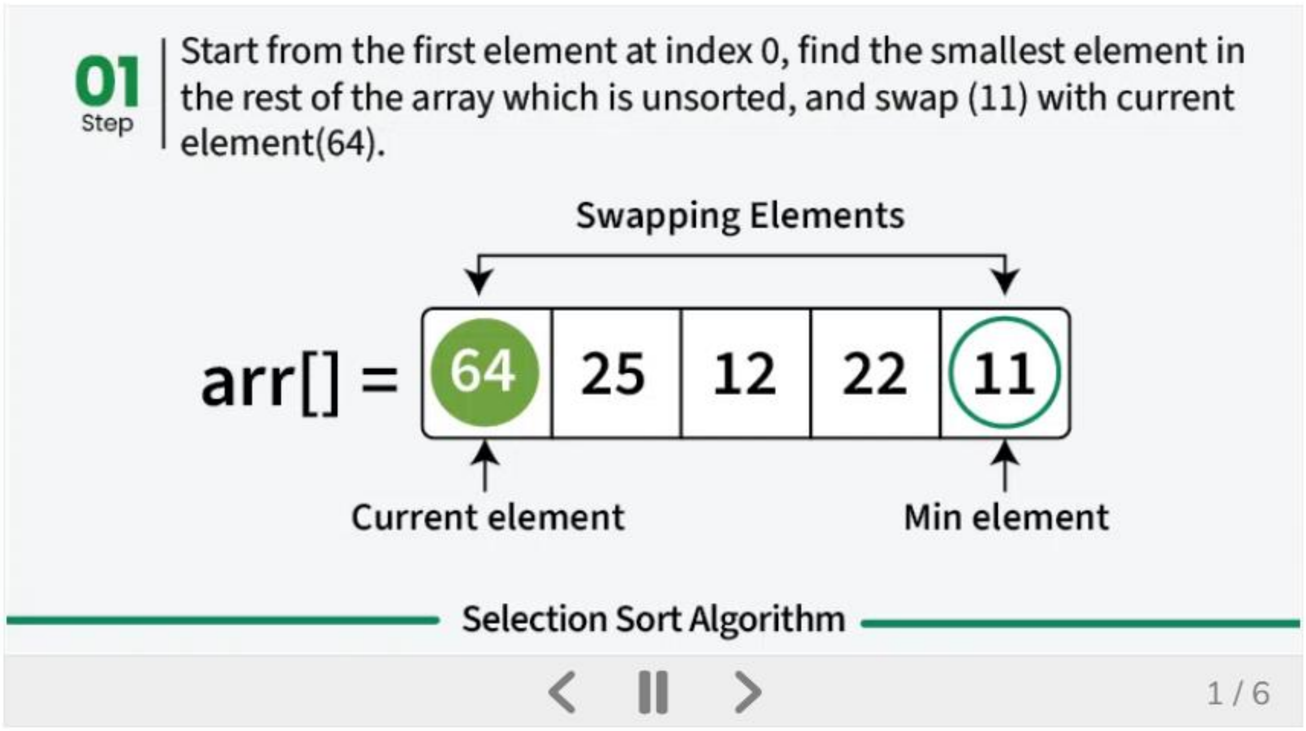


01 Step
Placing the 1st largest element at its correct position
i=0
5 6 1 3
Swap
i=1
5 6 1 3
Swap
i=2
5 1 6 3
Sorted Element
5 1 3 6
Bubble sort



02 Step
Placing 2nd largest element at its correct position
Swap
i=0
5 1 3 6
Swap
i=1
1 5 3 6
Sorted Elements
1 3 5 6
Bubble sort
2 / 3




01
Step
Start from the first element at index 0, find the smallest element in the rest of the array which is unsorted, and swap (11) with current element(64).
Swapping Elements
arr[] =
64
25
12
22
11
Current element
Min element
Selection Sort Algorithm
1 / 6

11 25 12 22

11 12 25 22 64

11 12 22 25 64

64 25 12 22 11

64 12 25 22 11

64 12 25 11 22

12 64 25 11 22

12 25 64 11 22

12 25 11 64 22

12 25 11 22 64

12 11 25 22 64

12 11 22 25 64

11 12 22 25 64

64 25 12 22 11

24 64 12 22 11

24 12 64 22 11

24 12 22 64 11

24 12 22 11 64

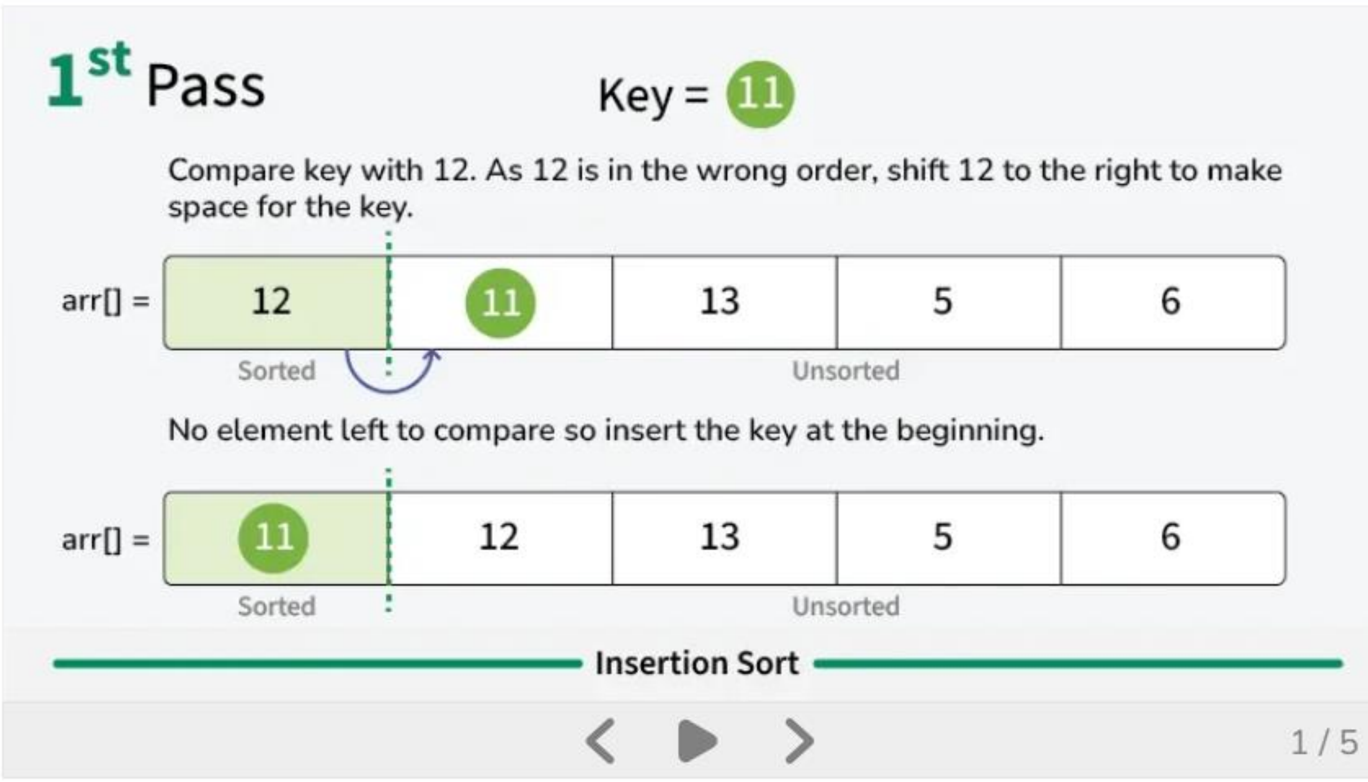
12 24 22 11 64

12 22 24 11 64

12 22 11 24 64

12 11 22 24 64

11 12 22 24 64



1st Pass

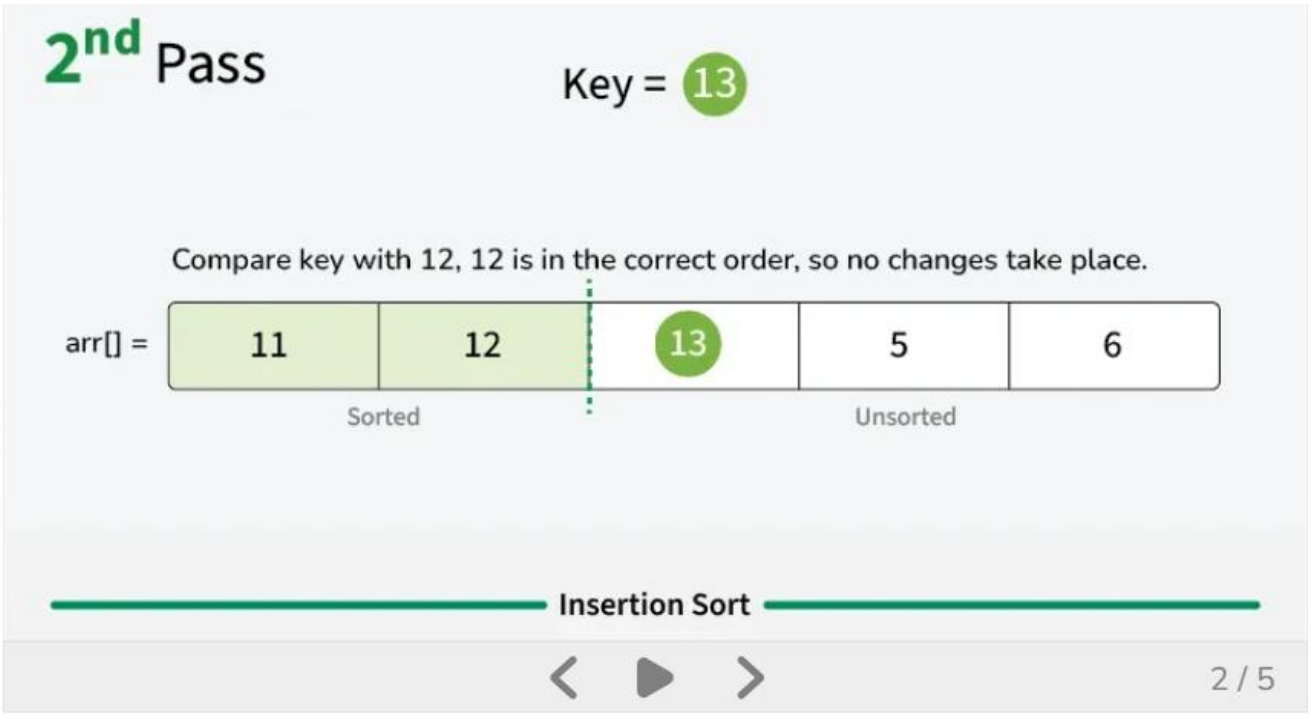
Key = 11

Compare key with 12. As 12 is in the wrong order, shift 12 to the right to make space for the key.

arr[] = | 12 | 11 | 13 | 5 | 6 |
Sorted
Unsorted

No element left to compare so insert the key at the beginning.

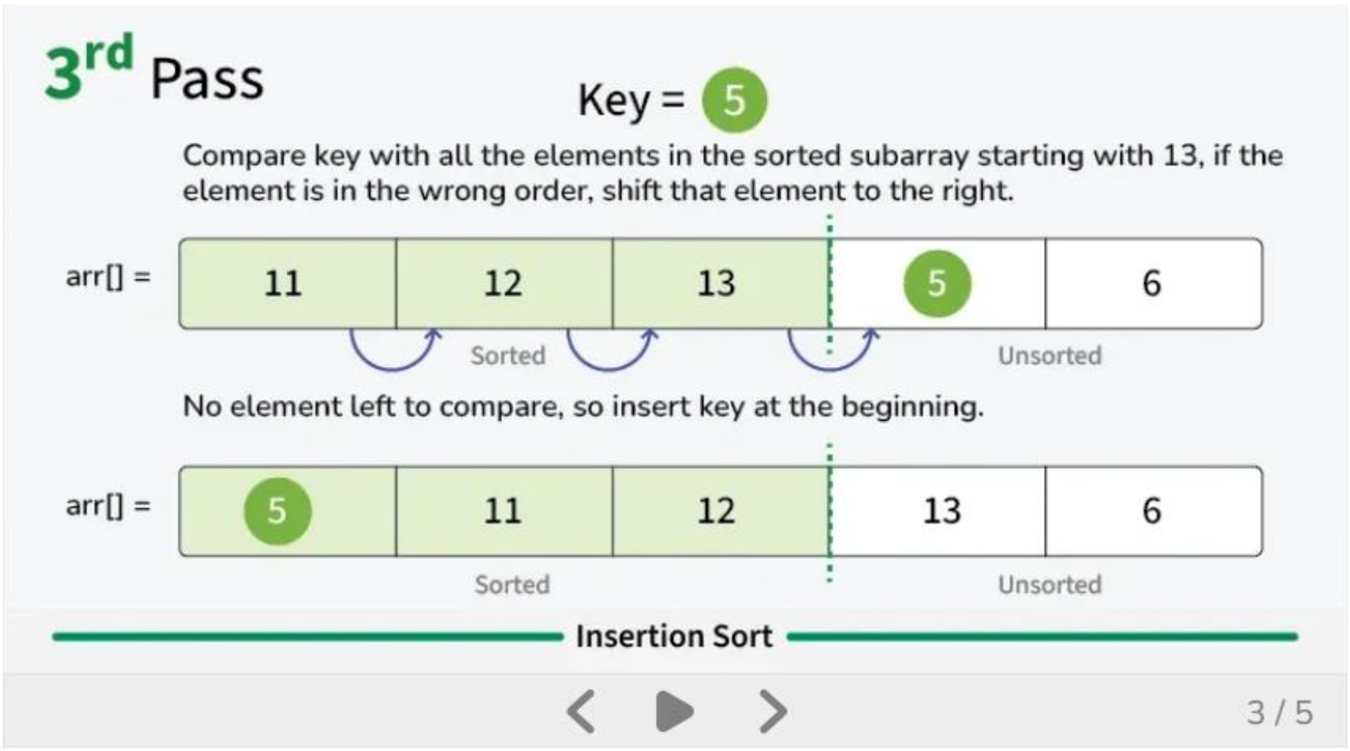
arr[] = | 11 | 12 | 13 | 5 | 6 |
Sorted
Unsorted
Insertion Sort
1 / 5



2nd Pass

Key = 13

Compare key with 12, 12 is in the correct order, so no changes take place.



3rd Pass

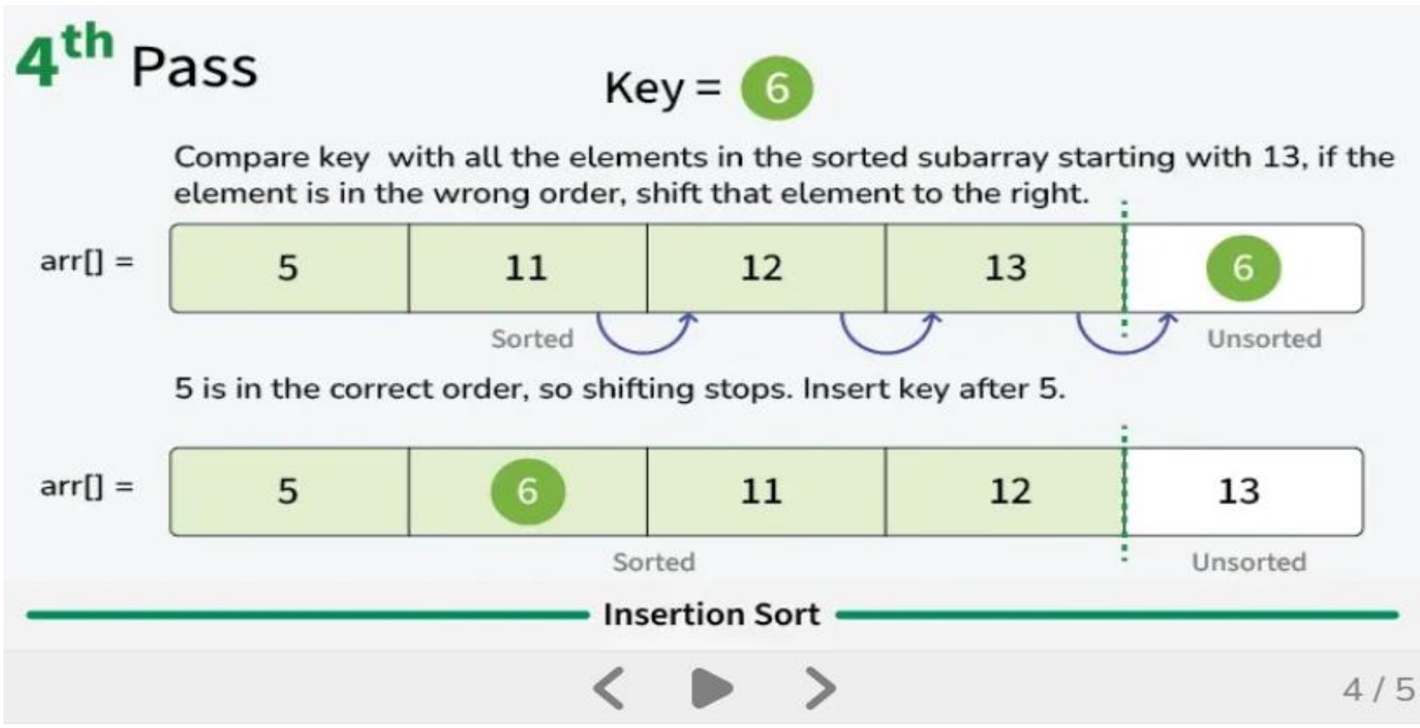
Key = 5

Compare key with all the elements in the sorted subarray starting with 13, if the element is in the wrong order, shift that element to the right.

arr[] = 11 | 12 | 13 | 5 | 6

No element left to compare, so insert key at the beginning.

arr[] = 5 | 11 | 12 | 13 | 6



4th Pass

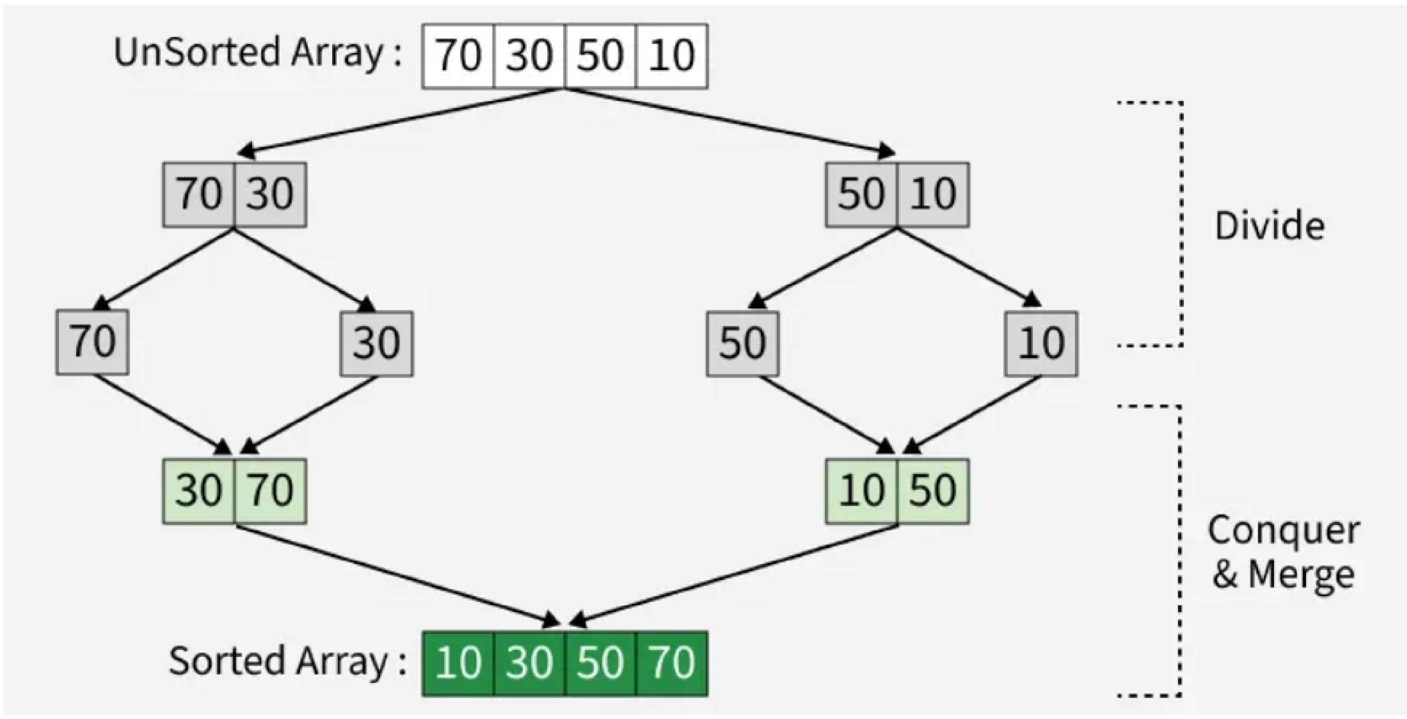
Key = 6

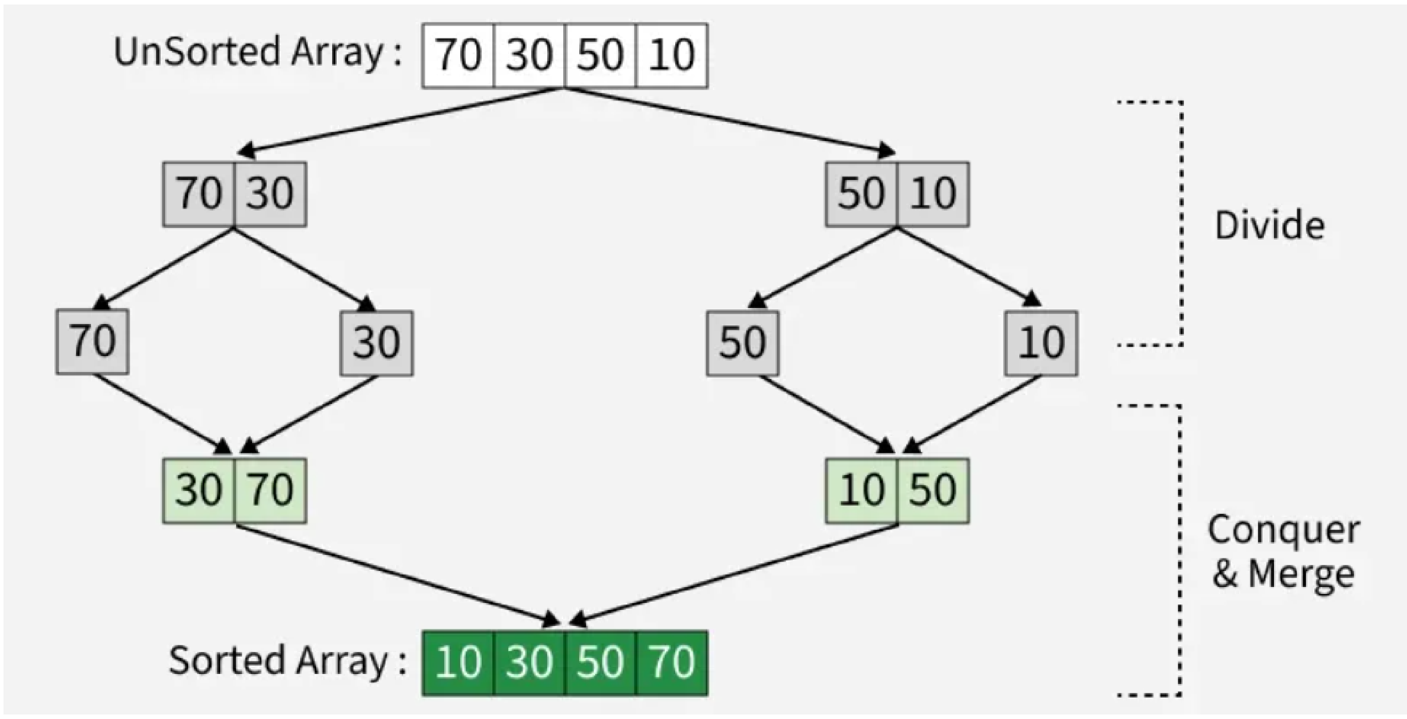
Compare key with all the elements in the sorted subarray starting with 13, if the element is in the wrong order, shift that element to the right.

arr[] = 5 | 11 | 12 | 13 | 6

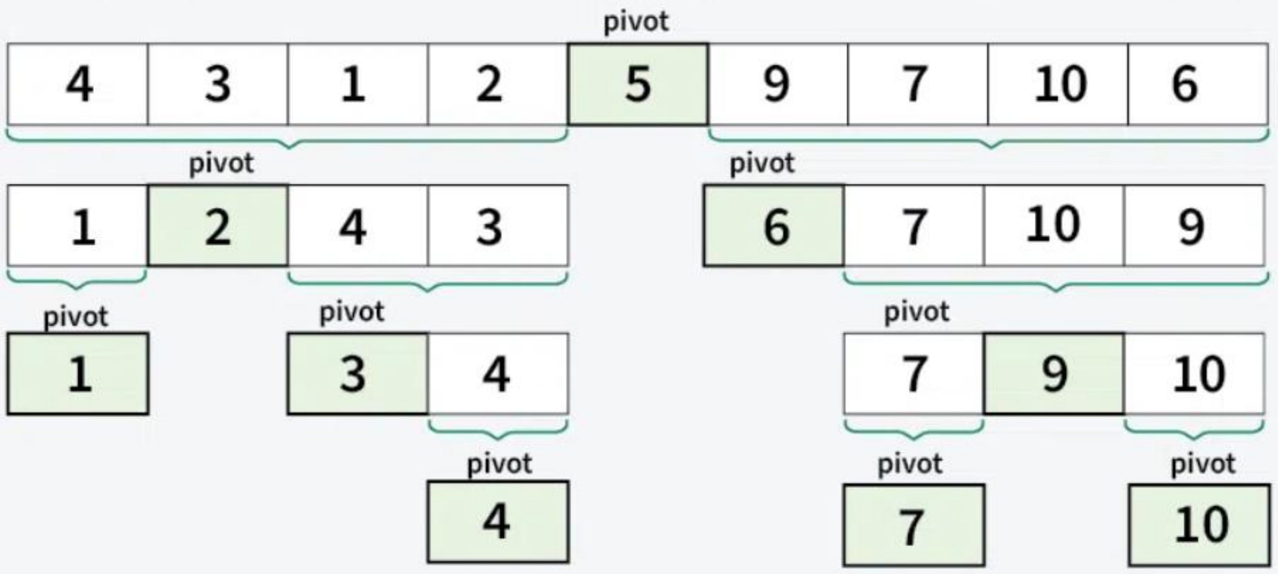
5 is in the correct order, so shifting stops. Insert key after 5.

arr[] = 5 | 6 | 11 | 12 | 13


UnSorted Array : 70 30 50 10
70 30
50 10
70
30
50
10
30 70
10 50
Sorted Array : 10 30 50 70
Divide
Conquer & Merge


UnSorted Array : 70 30 50 10
70 30
50 10
70
30
50
10
30 70
10 50
Sorted Array : 10 30 50 70
Divide
Conquer & Merge

Here, we have represented the recursive call after each partitioning step of the array.



Milestone 4

Reinforcement LEarning:

Q-Learning - value - based learning for decision-making.

Deep Q-Networks(DQN) - Comibines Q-Learning with deep neural networks.

Policy Gradient Methods - Directly Optimize policies.

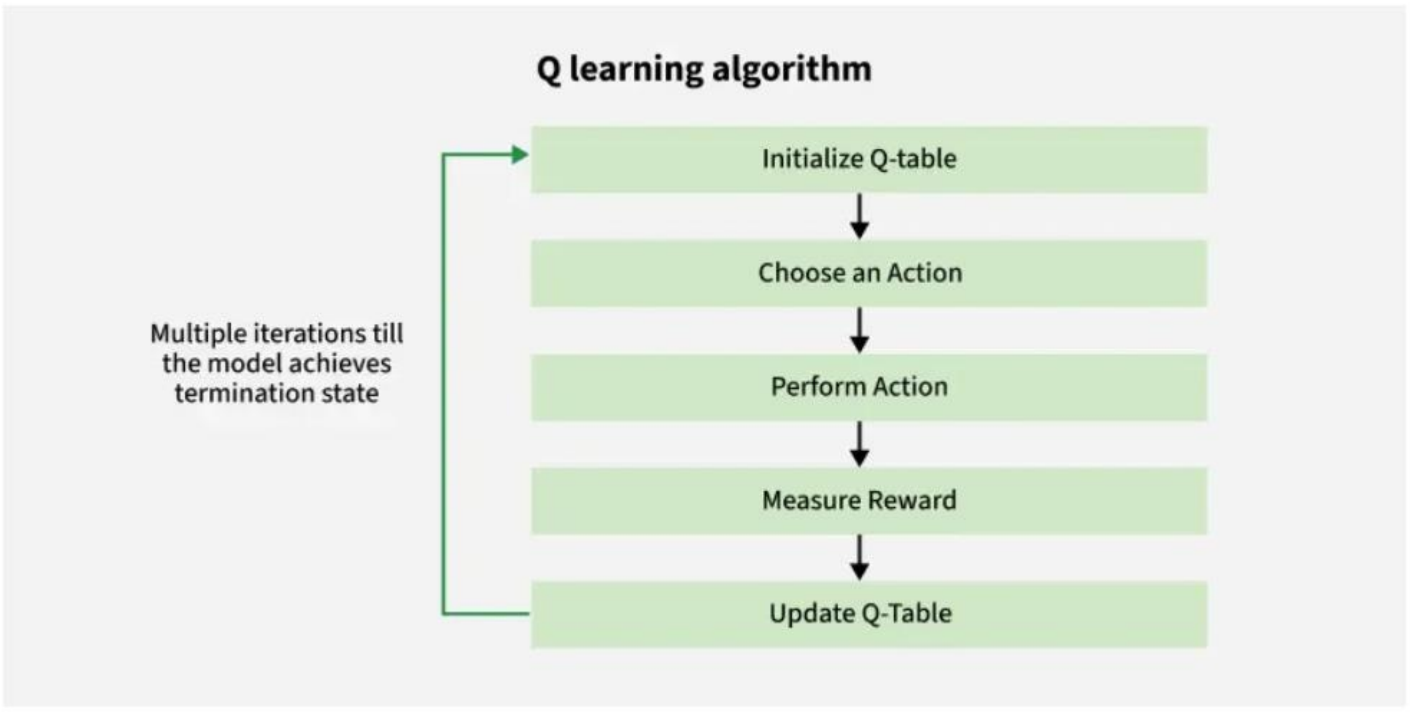
Input(APPLE) -> Response (Mongo) -> Feedback (Wrong answer the correct one will be APPLE) -> Learns(Note) -> model (rewrite it as the APPLE) -> takes this as the Input.

1. Q- Values or Action Values.
   Reward and Episodes
   Temporal Difference or TD- Update
   $Q(S,A) \leftarrow Q(S,A)+\alpha(R+\gamma Q(S',A')-Q(S,A))$

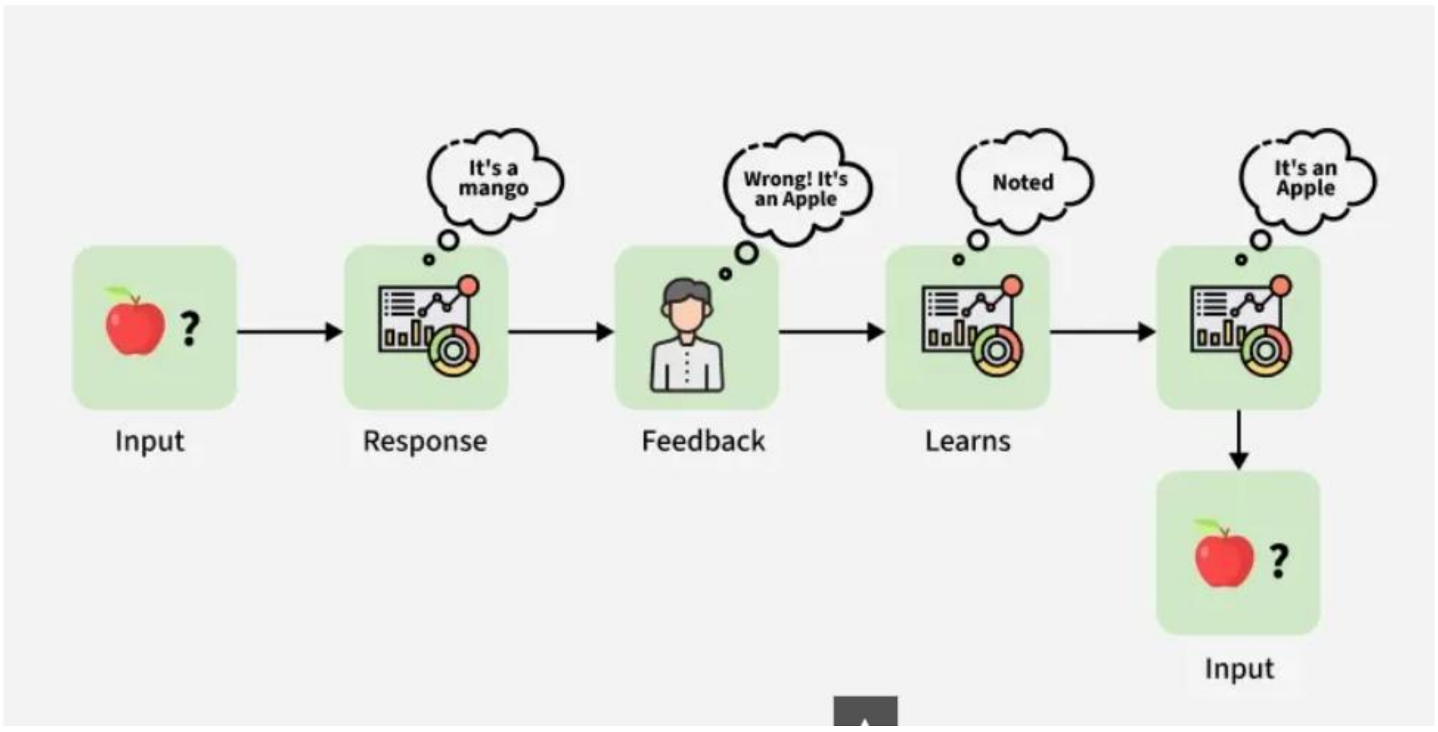
4. Greedy Policy

Exploitation

Exploration



Q learning algorithm



Methods for Determining Q-values

1. Temporal difference(TD)
   calculate the comparing the current state and action values with the previous onces.

Bellman's Equation:

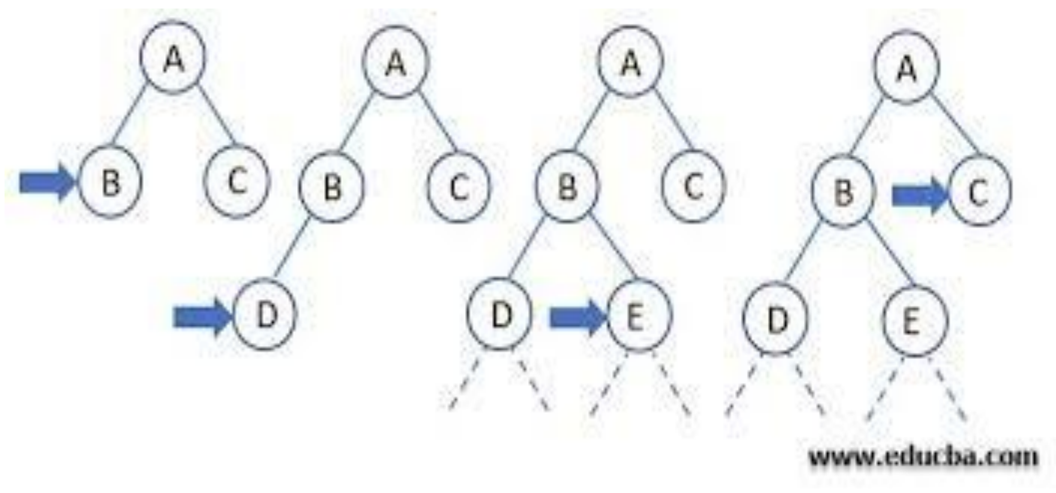
$Q(s,a)=R(s,a)+\gamma max a Q(s',a)$

DEPTH LIMITED SEARCH: Depth Limited Search (DLS) is an AI search algorithm, a variation of Depth First Search (DFS), that solves DFS's problem with infinite spaces by imposing a maximum depth limit, stopping exploration when the set depth is reached, treating nodes at that level as leaf nodes to prevent infinite loops and manage search time, making it useful for vast search spaces where the solution depth is known or expected to be shallow.

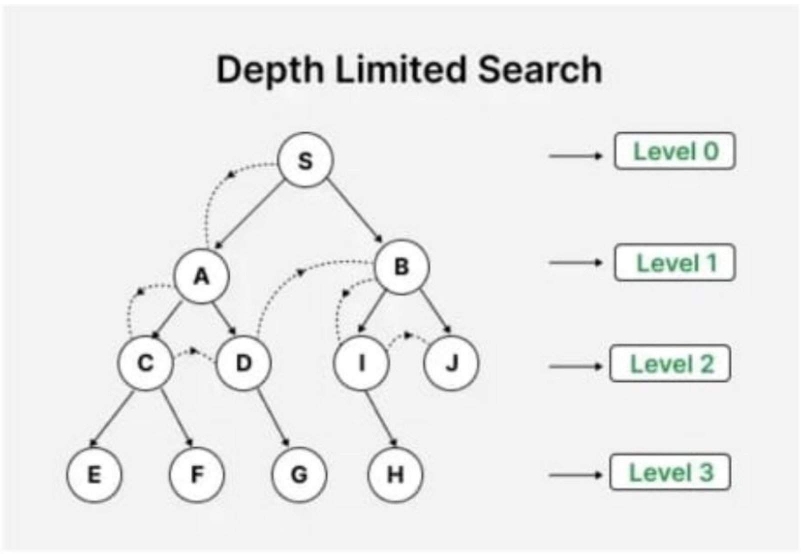
How it works:

- **Starts like DFS**: Begins at the root and explores as deeply as possible down one branch.
- **Applies a limit**: If it hits the specified depth (L), it stops exploring that path, even if the node isn't a goal.
- **Backtracks**: If the goal isn't found within the limit, it backtracks and explores other branches, respecting the depth constraint.

- **Treats deep nodes as leaves**: Nodes at depth 'L' are treated as if they have no children, effectively cutting off the search.



Depth Limited Search is a modified version of DFS that imposes a limit on the depth of the search. This means that the algorithm will only explore nodes up to a certain depth effectively preventing it from going down excessively deep paths that may also lead to not reaching the goal and is computationally expensive. By setting a maximum depth limit, DLS aims to improve efficiency and ensure more manageable search times.



Example of Depth Limited Search where Set Limit = Level 2

How Depth Limited Search Works

1. **Initialization**: Begin at the root node with a specified depth limit.
2. **Exploration**: Traverse the tree or graph, exploring each node's children.
3. **Depth Check**: If the current depth exceeds the set limit, stop exploring that path

ARTIFICIAL INTELLIGENCE

DFS WORKING:

Depth First Search (DFS) is a graph traversal method that starts from a source vertex and explores each path completely before backtracking and exploring other paths. To avoid revisiting nodes in graphs with cycles, a visited array is used to track visited vertices.

Note: There can be multiple DFS traversals of a graph according to the order in which we pick adjacent vertices. Here we pick vertices as per the insertion order.

Example:

Input: *adj[][] = [[1, 2], [0, 2], [0, 1, 3, 4], [2], [2]]*

Output: *[0, 1, 2, 3, 4]*
Explanation: *The source vertex is 0. We visit it first, then we visit its adjacent.*
Start at 0: Mark as visited. Print 0.
Move to 1: Mark as visited. Print 1.
Move to 2: Mark as visited. Print 2.
Move to 3: Mark as visited. Print 3.(backtrack to 2)
Move to 4: Mark as visited. Print 4(backtrack to 2, then backtrack to 1, then to 0).

Note that there can be more than one DFS Traversals of a Graph. For example, after 1, we may pick adjacent 2 instead of 0 and get a different DFS.

Input: *adj[][] = [[2, 3], [2], [0, 1], [0], [5], [4]]*

Output: *[0, 2, 1, 3, 4, 5]*
Explanation: *DFS Steps:*

Start at 0: Mark as visited. Print 0.
Move to 2: Mark as visited. Print 2.
Move to 1: Mark as visited. Print 1 (backtrack to 2, then backtrack to 0).
Move to 3: Mark as visited. Print 3 (backtrack to 0).

Start with 4.

Start at 4: Mark as visited. Print 4.
Move to 5: Mark as visited. Print 5. (backtrack to 4)

PCA:

*PCA (Principal Component Analysis) model is a statistical technique for **dimensionality reduction**, transforming large, complex datasets into a smaller set of new, uncorrelated variables (principal components) that capture most of the original information, primarily by identifying directions of maximum data variance, making data easier to visualize, analyze, and use in machine learning by reducing redundancy and preventing overfitting. It works by finding orthogonal axes (eigenvectors) with the highest variance (eigenvalues) and projecting the data onto these new axes, effectively simplifying high-dimensional data while preserving essential patterns.*

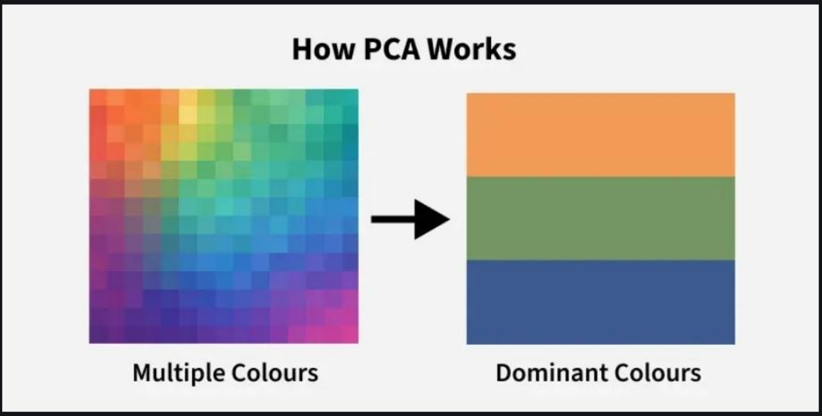
How a PCA Model Works (Key Steps)

1. ***Standardize Data**: Scale features to have a mean of 0 and variance of 1, ensuring all features contribute equally.*
2. ***Compute Covariance Matrix**: Calculate how variables change together.*
3. ***Calculate Eigenvalues & Eigenvectors**: Find the directions (eigenvectors) of greatest variance and their importance (eigenvalues) from the covariance matrix.*
4. ***Select Principal Components**: Sort components by eigenvalues (descending) and pick the top ones that explain a desired percentage of total variance (e.g., 95%).*
5. ***Transform Data**: Project the original data onto the selected principal components to get the reduced dataset.*

(PCA)

Last Updated : 13 Nov, 2025

PCA (Principal Component Analysis) is a dimensionality reduction technique and helps us to reduce the number of features in a dataset while keeping the most important information. It changes complex datasets by transforming correlated features into a smaller set of uncorrelated components.



Principal Component Analysis (PCA)

It helps us to remove redundancy, improve computational efficiency and make data easier to visualize and analyze.

How Principal Component Analysis Works

Open In App

PCA uses linear algebra to transform