Air Aware Smart Air Quality Prediction System

presented by -Likhitha Karri

1. Flask API

- Flask is a simple Python framework used to create APIs and web apps.
- It allows you to create URLs (routes) and write Python code that runs when someone visits that URL.

Example:

```
from flask import Flask
app = Flask(__name__)

@app.route("/hello")
def hello():
    return "Hello!"
```

2. How to import Flask

```
from flask import Flask, request, jsonify
```

- Flask → create app
- request → read data (payload)
- jsonify → send JSON response

3. Fast API

- Fast API is a modern, fast Python framework to build APIs with automatic documentation.
- It is faster than Flask and supports async features.

Example:

```python
from fastapi import FastAPI
app = FastAPI()

@app.get("/hello")
def hello():
    return {"msg": "Hello!"}
```

4. Payload

- Payload = data sent by the client to the server.
- Usually sent as JSON in the request body.

Example JSON Payload:

```json
{
  "name": "Likhitha",
  "age": 21
}
```

5. Postman

- Postman is a tool used to test APIs.
- You can send GET/POST requests, add payloads, headers, and check responses.

6. WebSocket

- WebSocket is a communication method where both client and server can send messages anytime (real-time).
- Used for chat apps, live updates, notifications.

7. Streaming

- Streaming means sending data in small parts instead of sending everything at once.
- Used for live logs, video/audio, real-time updates.

8. Flashing (Flask Flash Messages)

Flashing is used to show one-time messages to the user like:
✓ “Login successful”
✓ “Saved successfully”

Example:

```
flash("Data saved!")
```

1. git add .

- Adds all changed files in the current folder to the staging area.

👉 Means: “Git, track ALL my changes.”

2. git add main.py

- Adds only one file (main.py) to the staging area.

👉 Means: “Track only this file.”

3. git commit -m "message"

- Saves the staged changes with a message.

👉 Means: “Create a checkpoint with this message.”

Example:

- git commit -m "fixed bug"

4. git push

- Sends your committed changes to the remote repository (GitHub).

👉 Means: “Upload my work to GitHub.”

5. git branch

- Shows all local branches.

👉 Means: “Show branches saved on my computer.”

6. git branch --all

- Shows local + remote branches.

👉 Means:

- Local branches
- Remote branches (from GitHub)

7. git fetch --all

- Downloads information about all remote branches, but does NOT merge anything.

👉 Means:
"Check if GitHub has new updates, but don't apply them yet."

8. git pull

- **Use:** Download the latest changes from GitHub and merge them into your local branch

👉 "Get updates + merge."

9. git checkout -b "sh_new"

- **Use:** Create a new branch and switch to it.

👉 "Make a new branch named *sh_new*."

10. git checkout "sh_new"

- **Use:** Switch to an existing branch.

👉 "Move to sh_new branch."

11. python -m venv venv

- **Use:** Create a virtual environment named venv.

👉 "Separate Python environment for this project."

12. python -m venv shakthi

- **Use:** Create a virtual environment named shakthi.
  (Same as above but different name.)

13. requirements.txt

- **Use:** A file that lists all Python packages your project needs.

👉 Helps others install the same dependencies.

14. pip install -r requirements.txt

- **Use:** Install all packages listed in requirements.txt

👉 "Install everything needed for the project."

15. deactivate

- **Use:** Exit the virtual environment.

👉 Go back to the normal system Python.

AI / ML Concepts

AI Models

Programs trained to perform tasks like chat, prediction, or image generation.

Platforms/Companies

- Google → Gemini models
- OpenAI → GPT models
- Groq → Ultra-fast inference hardware + LLMs
- Microsoft → Copilot, Azure OpenAI services

LLM Families

Gemini

- Google’s family of Large Language Models.

GPT

- OpenAI’s Large Language Models (ChatGPT).

LLaMA

- Meta’s open-source LLM family.

Copilot

- Microsoft’s AI assistant built on LLMs**.**

LLM Basics

LLM (Large Language Model)

- An AI model trained on massive text data to understand and generate human language.

AI Platforms

Google AI Studio

- Google’s platform to build apps with Gemini models.

OpenAI Platform

- Dashboard to use GPT models, APIs, playgrounds.

Groq Platform

- Super-fast inference platform to run LLMs.

Rate/Token topics

Rate Limit

- 100 requests/mUse: Maximum number of API calls you can make per minute/hour.
- Example:in → after that API will block temporarily.

Token Limit

- Use: Maximum text the model can handle in one request (input + output).

Example:

- GPT-4o: 128k tokens

Chat Completion Component

- The part of an API that allows conversation-style messages.

Example:

```
{
  "model": "gpt-4",
  "messages": [
    {"role": "user", "content": "Hello"}
  ]
}
```

Prompt

A prompt is the *input or instruction* you give to an AI model.

Example:
"Explain machine learning in simple words."

ML (Machine Learning)

Machine Learning is a method where computers learn from data without being explicitly programmed.

Three main types of Machine Learning

1. Supervised Learning

The model learns from labeled data (input + correct output).

👉 **Example:**

- Email → “spam” or “not spam”
- Image → “cat” or “dog”

Types inside Supervised Learning:

✓ Regression

Predict continuous values
Examples:

- House price prediction
- Temperature prediction

✓ Classification

Predict categories/labels
Examples:

- Spam or not spam
- Disease or no disease

2. Unsupervised Learning

The model learns from unlabeled data (no correct answers given).

👉 **Example:**

- Grouping similar customers
- Finding patterns in data

Most common technique:

✓ **Clustering**
Example:

- Grouping customers by buying behavior

3. Reinforcement Learning

- The model learns by trial and error using rewards and punishments.

Examples:

- Training robots
- Game-playing AI (Chess, Go)
- Self-driving cars

Embedding

Definition:
Embedding is a way of converting text into numbers (vectors) so that a computer/AI model can understand the meaning.

Simple example:
"cat", "dog", "lion" → their embeddings will be close to each other because meanings are similar.

Use cases:

- Search
- Recommendations
- Chatbots
- Similarity matching

NLTK (Natural Language Toolkit)

Definition:
NLTK is a Python library used for text processing and NLP tasks.

It helps in:

- Tokenization (splitting text into words)
- Stemming
- Lemmatization
- Removing stopwords
- Basic NLP experiments

Example:

```
import nltk
from nltk.tokenize import word_tokenize


word_tokenize("I love AI")
```

Day - 8

Milestone 1 - Project Progress Review(25%)

- As part of Milestone 1, 25% of the project work has been successfully completed.
- The project mentor reviewed and verified the current progress of the project.
- The mentor provided valuable suggestions to improve the accuracy and performance of the project.
- Necessary areas for improvement were identified and clearly explained.
- A detailed discussion about the project design, implementation, and future enhancements was conducted with the mentor.

Day - 9

- The mentor reviewed the documents of all students.
- She checked the project work completed so far.
- The project presentation (PPT) was also reviewed.
- Necessary feedback was given based on the current progress and presentation quality.
- The review helped in identifying corrections and improvements in both the documentation and PPT.

Day - 10

AI - concepts

- **Introduction to AI :** Artificial Intelligence (AI) is a technology that enables machines to think, learn, and make decisions like humans. It is widely used in applications such as virtual assistants, healthcare, robotics, and self-driving cars.
- **Key fields :** machine learning, deep learning, NLP, reinforcement learning, knowledge based systems.
- **Introduction to ML :** Deep Learning (DL) is a subset of Artificial Intelligence that uses neural networks to mimic the way the human brain learns from large amounts of data. It is widely used in image recognition, speech processing, and natural language understanding.
- **Introduction to DL :** Deep Learning (DL) is a branch of Artificial Intelligence that uses neural networks with multiple layers to automatically learn from large amounts of data. It is widely used in image recognition, speech detection, chatbots, medical diagnosis, and self-driving cars.

Models :

- **CNN (Convolutional Neural Network):**
  CNN is a deep learning model mainly used for image and video recognition. It automatically detects features like edges, shapes, and objects from images.
- **RNN (Recurrent Neural Network):**
  RNN is designed for sequential data where past information helps predict the future. It is used in speech recognition, time-series data, and language translation.

- **LSTM (Long Short-Term Memory):**
  LSTM is a special type of RNN that remembers important information for a long time. It solves the problem of vanishing gradients in normal RNNs.

1. **Forget Gate:** Decides which information should be removed from the memory.
2. **Input Gate:** Decides which new information should be added to the memory.
3. **Output Gate:** Decides what information should be sent as the output.

- **RNN Transforms (Transformers):**
  Transformers are advanced deep learning models that process data using attention instead of memory. They are faster and more powerful than RNNs for tasks like language translation and chatbots.
- **Data using metrics :** Data metrics are used to measure, analyze, and evaluate the performance and quality of data or a model.

Day - 11

- **NLP (Natural Language Processing) :**
  NLP is a branch of AI that helps computers understand, interpret, and generate human language. It is used in chatbots, translation, sentiment analysis, and voice assistants.
- **NLTK Library :**
  NLTK is a Python library used for text processing and NLP tasks.
- **How to Install NLTK :**
  Install NLTK using: pip install nltk
- **NLP Techniques with Importing Statements :**
- **Tokenization & Importing:**
  Tokenization is splitting text into words or sentences – from nltk.tokenize import word_tokenize
- **Stop-Word Removal & Importing:**
  Stop-word removal removes common useless words – from nltk.corpus import stopwords
- **Stemming & Importing:**
  Stemming reduces words to their root form – from nltk.stem import PorterStemmer
- **Lemmatization & Importing:**
  Lemmatization converts words into meaningful base form – from nltk.stem import WordNetLemmatizer
- **Part of Speech (POS) Tagging & Importing:**
  POS tagging identifies grammar roles of words – from nltk import pos_tag
- **Named Entity Recognition (NER) & Importing:**
  NER identifies names of people, places, and organizations – from nltk import ne_chunk
- **SVM (Support Vector Machine) :**
  SVM is a supervised machine learning algorithm used for classification and regression. It works by finding the best hyperplane that separates different data classes
- **TF-IDF (Term Frequency – Inverse Document Frequency)** is a numerical method used to convert text into meaningful numbers.
  It measures how important a word is in a document compared to all other documents.
  TF SHOWS : how often a word appears in a document. **IDF :** reduces the weight of

common words like “is”, “the”, etc.
TF-IDF is widely used in search engines, text classification, and information retrieval.

Day - 12

- **SVM – support vector machine**
  Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression. It works by finding the optimal hyperplane that best separates different data classes.
- **Hard Margin SVM**
  Hard Margin SVM separates data with a strict boundary and allows no misclassification. It is used only when the data is perfectly linearly separable.
- **Soft Margin SVM**
  Soft Margin SVM allows some misclassification to handle overlapping and noisy data. It provides better generalization for real-world datasets.
- **Reinforcement Learning**
  Reinforcement Learning is a machine learning technique where an agent learns by interacting with an environment. The agent improves its actions using rewards and penalties.
- **Agent with Tools – 2 Types**

1. **Human-in-the-Loop Agent:**
   This agent works with human guidance during decision-making. Humans monitor, correct, and improve the agent’s actions.
2. **Fully Autonomous Agent (Without Human Interruption):**
   This agent works completely on its own without human involvement. It observes, decides, and acts automatically using tools and models.

- **AutoGen Framework**
  AutoGen is an open-source framework used to create multi-agent AI systems that can communicate and collaborate. It is mainly used to build autonomous AI workflows with minimal human input.

Day - 13

- **HTML Basics :**
  HTML Tags:
  Used to create elements like text, images, links, tables, and forms.
  HTML Headings:
  <h1> to <h6> are used for titles and subheadings.
  Meta Elements:
  Used in <head> for SEO, responsiveness, and page information.
- **CSS Sizing Concepts:**
  Sizing:
  Controls element size using px, %, em, rem.
  Intrinsic Size:

Element size based on its content.
Extrinsic Size:
Size forced using CSS width and height.
Min & Max Sizes: min-width, max-width, min-height, max-height control limits.

- **Overflow Handling:**
  visible:
  Shows extra content outside the box.
  hidden:
  Hides overflow content.
  scroll:
  Always shows scrollbar.
  auto: Shows scrollbar only when needed.
- **Box Model & Box Sizing:**
  Box Model:
  Content + Padding + Border + Margin.
  content-box:
  Size applies only to content.
  border-box: Size includes padding and border.
- **CSS Selectors:**
  Universal Selector (*) selects all elements.
- **Flexbox & Flex Attributes:**
  Flexbox:
  Used for responsive one-direction layouts.
  Flex Properties:
  display: flex, flex-direction, justify-content, align-items, gap.
  align-items: Controls vertical alignment.
- **Dynamic CSS:**
  CSS adjusts layout dynamically using media queries, transitions, and animations.
- **Python + HTML using Flask (Integration):**
  Flask connects Python backend with HTML frontend using routes and templates.
  It sends data from Python to HTML using **Jinja templates** for dynamic web pages.

```python
from flask import Flask
app = Flask(__name__)

@app.route("/")
def home():
    return "Hello, Flask!"

app.run(debug=True)
```

Day - 14

- **Flexbox Properties (CSS)**

  Flex-Wrap:
  Controls whether flex items stay in one line or move to the next line when space is less.
  Align-Self:
  Aligns a single flex item independently inside a flex container.
  Flex-Start:
  Aligns flex items to the starting position of the container.
  Stretch:
  Stretches flex items to fill available space inside the container.
  Auto:
  Automatically sets size or alignment based on available space.
  Order:
  Changes the display order of flex items without changing HTML structure.
- **OpenAI Installation & Import**

  Install OpenAI : pip install openai
  Import OpenAI : from openai import OpenAI
- **OpenAI Model Concepts**

  Model:
  A model is the AI system used to generate text, images, or answers.
  Message (Model Message):
  Input and output communication between user and AI.
  Temperature:
  Controls randomness of responses (low = accurate, high = creative).

Day - 15

- **Logistic Regression :** Logistic Regression is a supervised machine learning classification algorithm used to predict binary outcomes like 0 or 1, Yes or No, True or False.
  It uses the Sigmoid function to convert output into a probability between 0 and 1.
- Key Formula :

$$y = \frac{1}{1 + e^{-z}}, \quad z = w_0 + w_1 x_1 + w_2 x_2$$

- **Final Prediction Rule**

1. If $y \geq 0.5 \rightarrow$ Class = 1
2. If $y < 0.5 \rightarrow$ Class = 0

- **Sklearn Model:**

  A sklearn model is a ready-made machine learning algorithm provided by the scikit-learn library in Python, used for training and prediction.
- **How to Import:**
  from sklearn.linear_model import LogisticRegression

- **Decision Tree:**
  A Decision Tree is a supervised machine learning algorithm used for classification and regression.
  It makes decisions using a tree structure:

3. Root node → First condition
4. Branches → Decision rules
5. Leaf node → Final output

example:

Is it sunny?

/ \

Yes No

/ \

Is temperature hot? play = no

/ \

Yes No

||

Play = No Play = Yes

- **Entropy formula:** Entropy=−plogp−qlogq
- **information gain:** IG=Entropy(parent)−Entropy(children)
- **Decision Tree (sklearn):**
  from sklearn.tree import DecisionTreeClassifier
  clf = DecisionTreeClassifier()
- **Random forest:**
- Random Forest is an ensemble algorithm that:
  Uses many decision trees
  Combines their results for better accuracy
- **code:**
  from sklearn.ensemble import RandomForestClassifier
  rf = RandomForestClassifier()
- Builds multiple decision trees, each seeing a random part of data.
  from sklearn.ensemble import
  RandomForestClassifier
- rf = RandomForestClassifier()

K-Nearest Neighbors(KNN)

KNN is a supervised machine learning algorithm used for classification and regression.
It works by finding the K nearest data points and taking a majority vote.

- **formula:**
  d=(x1-x2)2+(y1-y2)2
- **import KNN:**
  from sklearn.neighbors import KNeighborsClassifier
  knn = KNeighborsClassifier(n_neighbors=5)
- **Example:**
  A = 170, B = 168 → Good for sports
  C = 150, D = 152 → Not fit
  New values: 169, 151
- **Using KNN:**
  169 → Classified as Good.
  151 → Classified as Not fit.
- **K-Means Clustering:**
  K-Means is an unsupervised machine learning algorithm used to group data into K clusters based on similarity.
- **Steps of K-Means:**
  Choose K centroids
  Assign points to nearest centroid
  Recalculate centroids
  Repeat until stable
- **Distance Formula in K-Means:**
  ∑||xi−centroid||2
- **K-Means:**
  from sklearn.cluster import KMeans
  kmeans = KMeans(n_clusters=3)
- **Linear Regression:**
  Used to:
  Predict continuous values like:
  Sales
  Price
  Temperature
- **Formula:**
  y=mx+c
- **MSE (Mean Squared Error):**
  Correct Formula:
  MSE=1/n∑(y−y^)2
- **Linear Regression:**
  from sklearn.linear_model import LinearRegression
  lr = LinearRegression()
- **XGBoost:**
  XGBoost = Extreme Gradient Boosting
  A powerful boosting algorithm

Uses many small decision trees
Very fast & accurate

- **Used in:**
  Kaggle
  Industry ML projects
  Stock, fraud, prediction systems
- **Formula:**
  $F_{new}(x) = F_{old}(x) + \eta \cdot h(x)$
- Where:
  η= **learning rate**
  $h(x)$ = **small decision tree**
  $F(x)$= **final prediction**
  from xgboost import XGBClassifier
  model = XGBClassifier

Day - 16

SQL(Structured Query Language):

- It is a standard language used to store, retrieve, update, and manage data in databases.
- In Simple Words:
- 👉 SQL is used to talk to a database.

Where SQL is Used:

- Websites
- Mobile apps
- Banking systems
- E-commerce (Amazon, Flipkart)
- Student management systems

What You Can Do with SQL:

- Create a database
- Insert data
- Read data
- Update data
- Delete data

example:
SELECT * FROM table name;

- ## Commands:
- **1. SELECT :** Used to read data from a table
  SELECT * FROM students;
  Use: To view data.

- **2. INSERT :** Used to add new data into a table
  INSERT INTO students VALUES (1, 'Ravi', 20);
  Use: To add record.
- **3. UPDATE :** Used to change existing data
  UPDATE students SET age = 21 WHERE id = 1;
  Use: To modify record.
- **4. DELETE :** Used to remove data from a table
  DELETE FROM students WHERE id = 1;
  Use: To delete record.
- **5. CREATE :** Used to create a new table/database
  CREATE TABLE students (
  id INT,
  name VARCHAR(20),
  age INT
  );
  Use: To create table.
- **6. ALTER :** Used to change table structure
  ALTER TABLE students ADD address VARCHAR(50);
  Use: To add/modify column.
- **7. DROP :** Used to completely delete a table
  DROP TABLE students;
  Use: To remove table permanently.
- **8. RENAME :** Used to change table name
  RENAME TABLE students TO student_info;
  Use: To rename table.
- **9. TRUNCATE :** Used to delete all records but keep table
  TRUNCATE TABLE students;
  Use: To clear all data fast.

using DISTINCT keyword in select:

SELECT DISTINCT(names)

WHERE:

SELECT * From Customer WHERE country = ‘Mexica’;

Operators:

Operator	Meaning	commands
=	Equal	SELECT * FROM students WHERE age = 18;
!= or <>	Not equal	SELECT * FROM students WHERE city != 'Hyderabad';
>	Greater than	SELECT * FROM students WHERE marks > 80;
<	Less than	SELECT * FROM students WHERE age < 18;
>=	Greater than or equal	SELECT * FROM students WHERE age >= 18;
<=	Less than or equal	SELECT * FROM students WHERE age <= 20;

Operator	Meaning	Example Command	What It Does
AND	All conditions must be TRUE	sql SELECT * FROM students WHERE age >= 18 AND city = 'Hyderabad';	Selects students who are **18+ AND from Hyderabad**
OR	Any one condition must be TRUE	sql SELECT * FROM students WHERE city = 'Hyderabad' OR city = 'Vijayawada';	Selects students from **Hyderabad OR Vijayawada**
NOT	Reverses the condition	sql SELECT * FROM students WHERE NOT city = 'Hyderabad';	Selects students **NOT from Hyderabad**

ORDER BY:

👉 ORDER BY is used to sort the result in ascending or descending order.

SELECT column_name FROM table_name ORDER BY column_name;

1. Ascending Order (Default – ASC)

👉 Sorts from small → big (A → Z, 1 → 9)

SELECT * FROM students ORDER BY name ASC;

2. Descending Order (DESC)

👉 Sorts from big → small (Z → A, 9 → 1)

SELECT * FROM students ORDER BY marks DESC;

Operator	Use	Example
IN	Match multiple values	WHERE city IN ('Hyd','Vizag')
BETWEEN	Range check	WHERE age BETWEEN 18 AND 25
LIKE	Pattern match	WHERE name LIKE 'R%'
IS NULL	Check empty value	WHERE name IS NULL

Operator	Use
UNION	Combine results without duplicates
UNION ALL	Combine results with duplicates
INTERSECT	Common records
EXCEPT	Records not in second table

Function	Use	Example Command	What It Returns
MIN()	Finds **smallest value**	sql SELECT MIN(marks) FROM students;	Lowest marks
MAX()	Finds **largest value**	sql SELECT MAX(marks) FROM students;	Highest marks
COUNT()	Counts total records	sql SELECT COUNT(*) FROM students;	Total number of students
AVG()	Finds average value	sql SELECT AVG(marks) FROM students;	Average marks

Pattern	SQL Example	Matching Names
a%	WHERE name LIKE 'a%'	apple, ant, aero
%t	WHERE name LIKE '%t'	ant, bat
%a%	WHERE name LIKE '%a%'	apple, ant, bat, ball, cat, aero
_a%	WHERE name LIKE '_a%'	bat, ball, cat
a_	WHERE name LIKE 'a_'	ant
a__	WHERE name LIKE 'a__'	ant
a%o	WHERE name LIKE 'a%o'	aero

SQL HAVING – Table Explanation

Concept	WHERE	HAVING
Used on	Individual rows	Grouped data
Used with	SELECT, UPDATE, DELETE	GROUP BY
Aggregate functions	❌ Not allowed	✅ Allowed (COUNT, AVG, SUM)
When it works	Before grouping	After grouping

join Type	Meaning	SQL Command Example	Output Result
INNER JOIN	Shows **only matching records** from both tables	sql SELECT students.name, marks.marks FROM students INNER JOIN marks ON students.id = marks.id;	Ravi–90, Sita–85
LEFT JOIN	Shows **all records from left table + matching from right table**	sql SELECT students.name, marks.marks FROM students LEFT JOIN marks ON students.id = marks.id;	Ravi–90, Sita–85, **Raju–NULL**
CROSS JOIN	Shows **all possible combinations** (Cartesian product)	sql SELECT students.name, marks.marks FROM students CROSS JOIN marks;	All names ×

Day - 17

Milestone 2 - Project report(50%)

- As instructed by the guide, 50% of the project is completed.
- Today, we presented the backend code, PPT, and documentation.
- Backend development is 50% completed.
- Database connection and basic APIs are implemented.
- The guide verified our progress.
- She instructed us to continue with the remaining backend and frontend work.

Day - 18

Milestone 2 – Project Review Session

- Today, the guide reviewed everyone's backend code, output, PPT, and documentation.
- She checked the project implementation and results carefully.
- She suggested improvements and necessary changes to enhance the project quality.
- She also explained which tools and methods can be used to get better results.
- We gave a 50% project progress explanation as part of Milestone 2.
- We presented our backend AI chatbot code, along with the completed document and PPT.
- Overall, today's session was focused on review, feedback, and project improvement guidance.

Day - 19

- Some students presented their project, PPT, code, and documentation the previous day.
- Today, the remaining students presented their work.
- The mam evaluated their presentations and awarded marks.

Day - 20

- We worked on the project during the class.
- The mam explained the required changes in the project.
- We implemented those changes during the entire class time.

Day - 21

Built-in methods in strings:

1. strip() — Removes spaces from both sides

```
text = "   Hello World   "
print(text.strip())
```

Output:

Hello World

2. join() — Joins elements with a string

```
words = ["I", "love", "Python"]
result = " ".join(words)
print(result)
```

Output:

I love Python

3. replace() — Replaces part of a string

```
text = "I like Java"
print(text.replace("Java", "Python"))
```

Output:

I like Python

4. startswith() — Checks if string starts with given word

```
text = "Python is easy"
print(text.startswith("Python"))
```

Output:

True

5. endswith() — Checks if string ends with given word

```
text = "Learning Python"
print(text.endswith("Python"))
```

Output:

True

6. find() — Finds the position of a word

```
text = "I am learning Python"
print(text.find("Python"))
```

Output:

14

(If not found, it returns -1)

7. isdigit() — Checks if string contains only numbers

```
num = "12345"
print(num.isdigit())
```

Output:

True

8. isalpha() — Checks if string contains only letters

```
name = "Python"
print(name.isalpha())
```

Output:

True

Day - 22

Dictionary methods:

1. get() — Get value using key

👉 Use:

It returns the value of a key.
If the key is not found, it returns None (no error).

print(student.get("name"))

print(student.get("marks")) # Key not present

Output:

Rahul

None

2. keys() — Returns all keys

print(student.keys())

Output:

dict_keys(['name', 'age', 'course'])

3. values() — Returns all values

print(student.values())

Output:

dict_values(['Rahul', 20, 'Python'])

4. items() — Returns all key-value pairs

print(student.items())

Output:

dict_items([('name', 'Rahul'), ('age', 20), ('course', 'Python')])

5. update() — Update or Add new key-value pair

👉 Add new data:

student.update({"marks": 85})

print(student)

👉 Update existing data:

student.update({"age": 21})

print(student)

Output:

{'name': 'Rahul', 'age': 21, 'course': 'Python', 'marks': 85}

6. pop() — Removes a specific key

student.pop("age")

print(student)

Output:

{'name': 'Rahul', 'course': 'Python', 'marks': 85}

7. popitem() — Removes the last inserted item

student.popitem()

print(student)

Output:

{'name': 'Rahul', 'course': 'Python'}

8. clear() — Removes everything from dictionary

student.clear()

print(student)

Output:

{}

Set methods:

1. add()

add() – Adds an element to the set .

A = {a, b, c}

add d

Result = {a, b, c, d}

2. remove()

remove() – Removes an element (error if not found) .

A = {a, b, c}

remove b

Result = {a, c}

3. discard()

discard() – Removes an element if present .

A = {a, b, c}

discard d

Result = {a, b, c} (no error)

4. pop()

pop() – Removes and returns a random element .

A = {a, b, c}

pop one element

Result = {b, c} (any one element is removed randomly)

5. clear()

clear() – Removes all the elements.

A = {a, b, c}

clear all

Result = { }

6. union()

union() – Returns a new set with all elements from both sets .

A = {a, b, c}

B = {c, d, e}

Union = {a, b, c, d, e}

7. intersection()

intersection() – Returns a set with common elements .

A = {a, b, c}

B = {c, d, e}

Intersection = {c}

8. difference() (A - B)

difference() – Returns a set with elements present in first set only.

A = {a, b, c}

B = {c, d, e}

A - B = {a, b}

File I/O methods:

open() – Opens a file

read() – Reads entire file content

readline() – Reads one line at a time

readlines() – Reads all lines as a list

write() – Writes data into file

writelines() – Writes multiple lines into file

close() – Closes the file

General purpose:

len() – Returns the length of an object

range() – Generates a sequence of numbers

print() – Displays output on the screen

type() – Returns the datatype of a value

id() – Returns the unique memory address of an object

sorted() – Returns a sorted list

enumerate() – Returns index and value pairs

zip() – Combines multiple iterables into pairs

Day - 23

conversion Functions:

- **int()** → Converts a value into an integer
  ✓ Example: int("10") → 10
- **float()** → Converts a value into a decimal number
  ✓ Example: float("5.5") → 5.5
- **str()** → Converts a value into a string (text)
  ✓ Example: str(100) → "100"
- **list()** → Converts a collection into a list
  ✓ Example: list((1,2,3)) → [1,2,3]
- **dict()** → Converts paired data into a dictionary (key-value)
  ✓ Example: dict([(1,"a"), (2,"b")]) → {1:"a", 2:"b"}
- **set()** → Converts data into a set (unique values)
  ✓ Example: set([1,1,2,3]) → {1,2,3}
- **tuple()** → Converts a collection into a tuple (fixed list)
  ✓ Example: tuple([1,2,3]) → (1,2,3)

Mathematical functions:

- **abs()** → Gives the positive value of a number
  ✓ Example: abs(-10) → 10
- **sum()** → Adds all values in a list/tuple
  ✓ Example: sum([1,2,3]) → 6
- **min()** → Finds the smallest value
  ✓ Example: min(5, 2, 9) → 2
- **max()** → Finds the largest value
  ✓ Example: max(5, 2, 9) → 9
- **pow()** → Gives the power of a number
  ✓ Example: pow(2, 3) → 8
- **round()** → Rounds a number to the nearest value
  ✓ Example: round(4.6) → 5

Functional Programing Tools:

- **filter()** → Selects values based on a condition
  ✓ Example: filter even numbers
  list(filter(lambda x: x%2==0, [1,2,3,4])) → [2,4]
- **map()** → Applies a function to every value
  ✓ Example:
  list(map(lambda x: x*2, [1,2,3])) → [2,4,6]
- **reduce()** → Combines all values into one result
  ✓ Example:
  reduce(lambda x,y: x+y, [1,2,3,4]) → 10
  ⚠ reduce() is inside functools module.

lambda:

- a = lambda a,b : a*b print(a(3,4))
- **input()** → Takes user input from the keyboard
  ✔ Example: name = input("Enter name: ")
- **format()** → Inserts values inside a string neatly
  ✔ Example:
  "My age is {}".format(20) → My age is 20

class and object related:

- **getattr()** → Gets the value of an object's attribute
  ✔ Example: getattr(obj, "name")
- **setattr()** → Sets or updates an attribute value
  ✔ Example: setattr(obj, "age", 20)
- **hasattr()** → Checks whether an attribute exists or not
  ✔ Example: hasattr(obj, "name") → True/False
- **delattr()** → Deletes an attribute from an object
  ✔ Example: delattr(obj, "age")
- **isinstance()** → Checks whether an object is of a specific class
  ✔ Example: isinstance(10, int) → True
- **issubclass()** → Checks whether a class is a child of another class
  ✔ Example: issubclass(B, A) → True
- **isinstance(object, class/type)**
  → Checks whether an object is an instance of a class, data type, or its subclass.
- ✔ Example:
  isinstance(10, int) → True
- And for comparison:
- **issubclass(child, parent)**
  → Checks whether one class is a subclass of another class.
- ✔ Example:
  issubclass(bool, int) → True

Miscellaneous:

- **globals()** → Returns all global variables as a dictionary
  ✔ Example: globals()
- **locals()** → Returns all local variables as a dictionary
  ✔ Example: locals()
- **callable()** ✅ (correct spelling, not *collable*)
  → Checks if an object is callable (can be used like a function)
  ✔ Example: callable(print) → True
- **eval()** → Evaluates and runs a single expression
  ✔ Example: eval("10+5") → 15
- **exec()** → Executes a block of Python code
  ✔ Example: exec("a=5\nb=6\nprint(a+b)") → 11

Exceptional handling:

- **try** → Write the code that may cause an error
  ✔ Example: try: x = 10/0
- **except** → Handles the error safely without stopping the program
  ✔ Example: except: print("Error occurred")
- **finally** → Always runs, whether error occurs or not
  ✔ Example: finally: print("Done")

Memory and object Management:

- **del()** → Deletes a variable or object from memory
  ✔ Example: del x
- **gc.collect()** ✅ (correct spelling, not *gc.collet*)
  → Forces garbage collection to free unused memory
  ✔ Example:
- import gc
- gc.collect()

importgc:

- **import gc** → Imports the Garbage Collection module in Python
  → It helps in automatically freeing unused memory.
- ✔ Example:
- import gc
- gc.collect()

gc:

- **gc** → Stands for Garbage Collection in Python
  → It automatically frees unused memory to improve performance.
- ✔ Example:
- import gc
- gc.collect()

collect():

- **gc.collect()** → Forces garbage collection to free unused memory immediately.
- ✔ Example:
- import gc
- gc.collect()

Working with iterables:

- **iter()** → Converts a collection into an iterator object
  ✔ Example:
  it = iter([10, 20, 30])

- **next()** → Retrieves the next value from an iterator
  ✓ Example:
  next(it) → 10

decorators and Metaprograming:

- **@staticmethod** ✅ (correct spelling, not *stacticmethod*)
  → A method that does not use object (self) or class (cls)
  ✓ Example:
- class A:
- @staticmethod
- def show():
- print("Hello")
- **@classmethod**
  → A method that works with the class using cls, not the object
  ✓ Example:
- class A:
- count = 0
- @classmethod
- def total(cls):
- return cls.count

context managers:

- **with** → Used to manage resources automatically (like files)
  → It opens and closes the resource automatically.
  ✓ Example:
- with open("data.txt") as f:
- print(f.read())
- **as** → Gives a nickname (alias) to the resource/module
  ✓ Example:
- with open("data.txt") as file:
- pass
- **import** → Used to bring a module into the program
  ✓ Example:
- import math

Day - 24

Python data types:

- **1. Numeric Types**
- Used for numbers.
- • **int** → whole numbers
- Example:
- age = 21
- • **float** → decimal numbers

- pi = 3.14
- • **complex** → numbers with real + imaginary part
- z = 2 + 3j
- **2. String (str)**
- Text data inside quotes.
- name = "Likitha"
- **3. Boolean (bool)**
- Only two values:
- True
- False
- Used in conditions.
- **4. List**
- Ordered, changeable, allows duplicates.
- my_list = [10, 20, 30]
- **5. Tuple**
- Ordered, **not changeable (immutable)**, allows duplicates.
- my_tuple = (10, 20, 30)
- **6. Set**
- Unordered, no duplicates.
- my_set = {1, 2, 3}
- **7. Dictionary (dict)**
- Stores data in **key-value pairs**.
- student = {"name": "Likitha", "age": 21}

8. NoneType

Represents “nothing” or “no value”.

x = None

PEP 8 (Python Enhancement Proposal 8)

PEP 8 is the official Python style guide that provides rules for writing clean, consistent, and readable Python code.
It includes guidelines for indentation, naming conventions, spacing, line length, comments, and how to structure code.

Why it's important:

- ✓ Makes code easy to read and understand
- ✓ Helps maintain consistency in large projects
- ✓ Reduces mistakes caused by bad formatting
- ✓ Makes teamwork smoother because everyone follows the same style
- ✓ Looks professional and follows industry standards

Python Memory Management:

Private Heap

Python stores all objects (lists, strings, numbers, etc.) in a **private heap**.
This heap is controlled only by Python, not directly by the user.

Memory Manager

Python has an internal memory manager that:

- Allocates memory for new objects
- Reuses freed memory
- Organizes the heap efficiently

You don't need to manage memory manually.

Reference Counting

Every object keeps a **count of how many variables refer to it**.
When the count becomes **0**, Python immediately frees that memory.

Garbage Collector

Some objects form cycles (A → B → A).
Reference counting cannot delete them, so Python uses a **garbage collector** to detect and remove these unused cycles.

Dynamic Memory Allocation

Python allocates memory **automatically at runtime**, depending on the type and size of the data.
No need to declare memory sizes manually.

Object Caching / Interning

Python **reuses small integers and some strings** instead of creating new ones to save memory and improve performance.

List and tuple:

- List is mutable (changeable) and uses [ ],
- while Tuple is immutable (not changeable) and uses ( ).

Dictionary:

A **dictionary** stores data in **key–value pairs** and is **changeable**.

Example:

{"name": "Likitha", "age": 21}

How to Merge Dictionaries

- ✅ Using update()
- d1.update(d2)
- ✅ Using |
- d3 = d1 | d2
- ✅ Using **
- d3 = {**d1, **d2}

Nested List

A nested list is a list inside another list.

Example:

- nested = [1, 2, [3, 4], 5]
- Here, [3, 4] is a list inside a list.

Flattened List

A flattened list means converting a nested list into a single-level list (no inner lists).

Example:

- nested = [1, 2, [3, 4], 5]
- flattened = [1, 2, 3, 4, 5]

Difference Between Shallow Copy and Deep Copy:

Shallow Copy

- Copies **only the outer structure**.
- Inner lists or objects are **shared** (not duplicated).
- Changes in inner objects affect both copies.

Example:

- import copy
- a = [1, [2, 3]]
- b = copy.copy(a) # shallow copy

Deep Copy

- Copies **both outer and all inner objects**.
- Completely **independent** copy.
- Changes in inner objects *do not* affect the original.

Example:

- import copy
- a = [1, [2, 3]]
- b = copy.deepcopy(a) # deep copy

How Slicing Works:

Slicing creates a **new list** from a part of an existing list.

Syntax:

list[start:end:step]

Example:

- a = [10, 20, 30, 40, 50]
- print(a[1:4]) # [20, 30, 40]
- print(a[:]) # copies whole list
- print(a[::-1]) # reverse list

How to Reverse a List:

Method 1: Using slicing

lst[::-1]

Method 2: Using reverse()

lst.reverse()

Method 3: Using reversed()

list(reversed(lst))

Frozenset:

A **frozenset** is an **immutable (unchangeable)** version of a set.

Example:

fs = frozenset([1, 2, 3])

- You **cannot add or remove** elements.
- Useful for dictionary keys or fixed sets.

3. Difference Between is and ==

Operator	Meaning	Checks
==	Equality	Values are same or not
is	Identity	Whether two variables refer to the same object in memory

Example:

- a = [1, 2]
- b = [1, 2]
- a == b # True (same values)
- a is b # False (different objects)

Day - 25

Features of Java:

- **Simple**
- Easy to learn and write because of clean syntax.
- **Object-Oriented**
- Everything is based on objects and classes.
- **Platform Independent**
- *Write Once, Run Anywhere (WORA)* — Java runs on any system using JVM.
- **Secure**
- Provides strong security features like bytecode verification, exception handling, and sandboxing.
- **Robust**
- Handles errors using exceptions and has strong memory management (Garbage Collection).
- **Multithreaded**
- Allows multiple tasks to run at the same time (threads).
- **Portable**
- Java programs can run on any device without modification.
- **High Performance**
- Uses JIT (Just-In-Time) compiler to run code faster.
- **Distributed**
- Supports networking and distributed systems easily (RMI, Socket programming).
- 🔟 **Dynamic**
- Loads classes dynamically at runtime, making programs flexible.

JVM (Java Virtual Machine):

- JVM loads, verifies, and executes Java bytecode
- Makes Java platform independent (Write Once, Run Anywhere)
- Handles memory management and garbage collection
- Provides security and runtime environment

Process in Java:

A **process** is an independent program running in its **own memory space**.

Key Points:

- Has its **own memory** and resources
- Heavyweight (takes more time to create)
- Communication between processes is slow
- Example: Running two separate Java programs

Thread in Java:

A **thread** is a **smallest unit of execution** inside a process.
Multiple threads share the **same memory** of the process.

Key Points:

- Lightweight (easy and fast to create)
- Share same memory & resources
- Used for multitasking (e.g., background tasks)
- Example: A Java program with multiple threads running at once

Collection framework:

The Collection Framework in Java is a set of classes and interfaces used to store and manage groups of objects in an efficient way.

It provides ready-made data structures like List, Set, Map, Queue, etc.

Primitive Data Types in Java:

Java has 8 primitive data types:

1. **byte**
   - Size: 1 byte
   - Range: -128 to 127
2. **short**
   - Size: 2 bytes
3. **int**
   - Size: 4 bytes
   - Most commonly used integer type
4. **long**
   - Size: 8 bytes

5. **float**
    - Size: 4 bytes
    - Decimal values (less precision)
6. **double**
    - Size: 8 bytes
    - Decimal values (more precision)
7. **char**
    - Size: 2 bytes
    - Stores a single character (e.g., 'A')
8. **boolean**
    - Stores: **true** or **false**

JRE (Java Runtime Environment):

JRE is the environment required to run Java programs.
It contains:

- **JVM** (Java Virtual Machine)
- **Libraries**
- **Class files**
- **Runtime components**

JRE **does NOT** include tools for development (compiler).

Access Modifiers :

Access modifiers control **visibility** of classes, methods, and variables.

Java has **4 access modifiers**:

public

- Accessible **from anywhere** in the program.

public int x;

private

- Accessible only within the same class.

private int x;

protected

- Accessible within:
    - Same class
    - Same package
    - Subclasses (even in different packages)

protected int x;

default (no keyword)

- Accessible only within the same package.

int x; // default access

Method Overloading:

Method Overloading means creating multiple methods with the same name but different parameters in the same class.

Key Points

- Same method name
- Different parameter list (type, number, or order)
- Return type may be same or different
- Happens within the same class
- Decided at compile time → *Compile-time polymorphism*

Deadlock:

A deadlock occurs when two or more threads wait for each other forever, and none of them can continue.

In simple words:
Two threads block each other by holding resources the other needs.
So the program gets stuck forever.

Example :

- Thread A has Resource 1 and waits for Resource 2
- Thread B has Resource 2 and waits for Resource 1

Both keep waiting → Deadlock

Conditions for Deadlock:

Deadlock happens when all these 4 conditions occur:

Mutual Exclusion
Hold and Wait
No Preemption
Circular Wait

Day - 26

Ma’am asked us to continue working on the project in class. She also told us to inform her if we have any errors or doubts.

Day - 27

Software Development Life Cycle (SDLC):

1. **Requirement Analysis** – Understand what the user needs
2. **Design** – Plan system architecture and design
3. **Development** – Write and build the code
4. **Testing** – Find and fix bugs
5. **Deployment** – Release the software to users
6. **Maintenance** – Update, fix issues, and improve performance

👉 SDLC helps in developing high-quality, reliable software in a systematic way.

Functional Consultant / Business Consultant:

- A Functional (Business) Consultant understands business requirements and converts them into functional solutions for the technical team.
- They act as a bridge between client (business) and developers.
- They do not code, but explain *what the system should do*.

Functional Specification Document (FSD):

- FSD is a document prepared by the Functional Consultant.
- It clearly explains business requirements, workflows, rules, and expected system behavior.
- Developers use this document to build the software correctly.

Developer / Programmer:

- A Developer / Programmer writes code to build software based on requirements.
- They convert functional specifications into working applications.
- They handle coding, debugging, and implementation.

Technical Specification Document (TSD):

- TSD is prepared by the Developer / Technical team.
- It explains how the system will be built technically.
- It acts as a technical guide for development.

Prototype:

- A Prototype is a sample or demo version of the software.
- It shows how the system will look and work before full development.
- Used to get client feedback and reduce mistakes.

Unit Testing:

- Unit Testing is done by Developers.
- It tests individual modules or functions of the code.
- Purpose: to ensure each unit works correctly.

Main Servers / Environments:

There are three main servers:

1. **Development Server**
   - Used by developers for coding and unit testing.
2. **Quality (QA) Server**
   - Used by Testing / QA team.
   - Here system testing, integration testing, and regression testing are performed.
3. **Production Server**
   - Final live server used by end users.
   - Only tested and approved code is deployed.

Testing / QA (Quality Assurance):

- QA team ensures the software meets quality standards.
- They find bugs, verify requirements, and ensure error-free delivery.
- QA focuses on process quality and product quality.

Test Script:

- A Test Script is a step-by-step set of instructions used by the QA/Test team to test an application.
- It checks whether the software works as expected.

Administration:

- Administration manages and controls the software system and servers.
- It ensures the system runs smoothly, securely, and continuously.

Administration Responsibilities:

- User access and role management
- Server and system maintenance
- Database backup and recovery
- Security monitoring
- Software updates and configuration
- Performance monitoring

Network:

- Handles communication and connectivity between systems.
- Manages LAN, WAN, internet, firewalls, and security.
- Ensures data transfer is fast and secure.

Database:

- Stores and manages application data.
- Ensures data integrity, backup, and recovery.
- Examples: MySQL, Oracle, SQL Server.

Application:

- The software program used by end users.
- Processes user requests and interacts with the database.
- Examples: web apps, mobile apps.

Security:

- Security protects the application, data, and network from unauthorized access and attacks.

Security Responsibilities:

- User authentication and authorization
- Data encryption
- Network security (firewalls, VPN)
- Secure access control
- Regular security updates and patches
- Monitoring threats and vulnerabilities

Product / Project Management:

- Product / Project Management plans, monitors, and controls the software development process.
- Ensures the project is delivered on time, within budget, and with quality.

Sales & Marketing :

- Sales & Marketing promote the software/product and generate customers and revenue

Human Resource (HR):

- Human Resource (HR) manages the people in an organization.
- Ensures the right talent is hired, developed, and retained.

Documentation Team:

- The Documentation Team creates and maintains written records related to the software project.
- Helps users, developers, and stakeholders understand the system clearly.

Support

- Support Team helps users after the software is deployed.

- They resolve issues and ensure smooth usage of the application.

Waterfall Model:

- The Waterfall Model is a linear and sequential software development model.
- Each phase must be completed before moving to the next phase.

Phases of Waterfall Model:

1. Requirement Analysis
2. System Design
3. Development (Coding)
4. Testing
5. Deployment
6. Maintenance

Agile Model:

- The Agile Model is an iterative and incremental software development approach.
- Work is divided into small cycles called sprints.

Key Features of Agile Model:

- Frequent customer feedback
- Continuous development and testing
- Flexible to changes
- Fast delivery of working software
- Team collaboration

Agile Process:

Plan → Design → Develop → Test → Review → Repeat

Day - 28

Ma'am asked us to continue working on the project in class. She also told us to inform her if we have any errors or doubts.

Day - 29

Sorting algorithms:

A sorting algorithm arranges data in a specific order (ascending or descending).

Example:
Unsorted → [5, 2, 9, 1]
Sorted → [1, 2, 5, 9]

Bubble Sort:

👉 Concept:

- Compare **adjacent elements**
- Swap if they are in the wrong order
- Largest element "**bubbles**" to the end

How it works:

- Repeat passes until list is sorted

Python Code:

```
arr = [5, 1, 4, 2, 8]


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


print(arr)
```

⏱️ Time Complexity:

- Best: O(n)
- Worst: O(n^2)

Selection Sort:

👉 Concept:

- Find the smallest element
- Place it at the correct position

How it works:

- Select minimum → swap → repeat

Python Code:

```
arr = [64, 25, 12, 22, 11]
```

```
for i in range(len(arr)):
    min_idx = i
    for j in range(i+1, len(arr)):
        if arr[j] < arr[min_idx]:
            min_idx = j
    arr[i], arr[min_idx] = arr[min_idx], arr[i]

print(arr)
```

⏱️ **Time Complexity:**

- Best/Worst: O(n²)

Insertion Sort:

👉 **Concept:**

- Works like sorting cards in hand
- Insert each element into its proper position

How it works:

- Take one element at a time
- Shift larger elements to the right

Python Code:

```
arr = [12, 11, 13, 5, 6]

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
```

```
print(arr)
```

⏱️ Time Complexity:

- Best: O(n)
- Worst: O(n²)

Merge Sort:

👉 Concept:

- Divide and Conquer
- Split list → sort → merge

How it works:

- Recursively divide array
- Merge sorted halves

Python Code:

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr)//2
        L = arr[:mid]
        R = arr[mid:]

        merge_sort(L)
        merge_sort(R)

        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i += 1
```

```
        else:
            arr[k] = R[j]
            j += 1
        k += 1

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

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

arr = [38, 27, 43, 3, 9]
merge_sort(arr)
print(arr)
```

⏱️ **Time Complexity:**

- Best/Worst: O(n log n)

Quick Sort:

👉 **Concept:**

- Choose a **pivot**
- Elements smaller → left
- Elements larger → right

How it works:

- Partition around pivot

- Recursively sort

Python Code:

```
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[0]
    left = [x for x in arr[1:] if x < pivot]
    right = [x for x in arr[1:] if x >= pivot]
    return quick_sort(left) + [pivot] + quick_sort(right)


arr = [10, 7, 8, 9, 1, 5]
print(quick_sort(arr))
```

⏱️ **Time Complexity:**

- Best/Average: O(n log n)
- Worst: O(n²)

Heap Sort:

👉 **Concept:**

- Uses Heap data structure
- Build max heap → extract max

How it works:

- Convert array into heap
- Remove largest repeatedly

Python Code:

```
import heapq


arr = [12, 11, 13, 5, 6, 7]
heapq.heapify(arr)
```

```
sorted_arr = []


while arr:
   sorted_arr.append(heapq.heappop(arr))


print(sorted_arr)
```

⏱️ **Time Complexity:**

- Best/Worst: O(n log n)

Day - 30

Milestone - 3

- On Day 30, ma’am checked whether the work was 75% completed or not
- Some teams completed their presentations
- By the time those presentations finished, class time was over
- So remaining teams could not present
- Ma’am said the remaining presentations will be continued later

Day - 31

- On the 31st day, the remaining teams gave their presentations
- Ma’am evaluated their work
- Marks were awarded to those teams
- The session was completed.

Day - 32

- Some teams faced errors in their projects
- Ma’am checked the errors
- She gave suggestions to fix them
- Teams were asked to make the required corrections

Day - 33

- Ma’am assigned some tasks
- She asked us to write the required code
- The written codes were checked
- Ma’am gave feedback and corrections where needed.

Day - 34

Q Table:

- A Q-table is a table that stores Q-values.
- Q-value = Quality of an action taken in a particular state
- It tells the agent:
  👉 *"If I am in this state and take this action, how good will it be in the future?"*

Where is it used?

Q-table is mainly used in Q-learning, which is a model-free reinforcement learning algorithm.

🔹 Q-Value Formula (Basic):

$Q(s,a)=Q(s,a)+\alpha[r+\gamma \max Q(s',a')-Q(s,a)]$

Where:

- α (alpha) → learning rate
- γ (gamma) → discount factor
- r → reward
- s' → next state

Agentic Framework:

An Agentic Framework is a system design where AI agents can think, decide, act, and learn autonomously to achieve a goal — instead of just responding once like a normal chatbot.

🔹 Core Components of an Agentic Framework:

Agent (Brain)

- Usually powered by an LLM
- Makes decisions
- Chooses next action

Goal

- What the agent wants to achieve
  👉 Example: *"Generate leads from Instagram"*

Memory

- Short-term memory → current task context
- Long-term memory → past experiences, results

Tools

Agent can use:

- APIs
- Databases
- Web search
- Code execution
- Files (CSV, PDF, Excel)

Environment

- Where actions happen (web, app, system, user input)

Feedback / Reward

- Checks if action worked
- Improves next decision

Agentic memory:

Agentic Memory is the memory system of an AI agent that helps it remember, recall, and learn from past interactions so it can make better decisions over time.

Without memory → AI is forgetful
With memory → AI becomes smart, consistent, and personalized.

Day - 35

Backpropagation Algorithm:

Backpropagation is the core algorithm used to train Artificial Neural Networks.
It helps the network learn from mistakes by updating weights to reduce error.

How Backpropagation Works:

Forward Propagation

- Input data is passed through the network
- Each neuron calculates output using weights + activation function
- Final output is produced.

👉 Example:
Input → Hidden Layer → Output

Loss (Error) Calculation

- Compare actual output with expected output
- Calculate error using a loss function.

📌 **Common loss function:**

Error=(Expected−Actual)2

Backward Propagation

- Error is sent backwards from output layer to input layer
- Using chain rule (from calculus), we find:
  - how much each weight contributed to the error.

Weight Update (Learning)

- Weights are updated using Gradient Descent.

📌 **Formula:**

New Weight=Old Weight−η×∂Error/∂Weight

Where:

- η\etaη = learning rate

Repeat the Process

- Steps 1–4 are repeated for many epochs
- Error keeps reducing
- Model becomes more accurate.

Sigmoid:

Activation functions decide whether a neuron should fire or not.
They add non-linearity, so neural networks can learn complex patterns.

Sigmoid Function

Formula:

σ(x)=1/1+e−x

Range: 0 to 1
Use: Binary classification (Yes/No)

Pros

- Smooth curve
- Output can be treated as probability

Cons

- Vanishing gradient (learning becomes slow)
- Not zero-centered

Example: Spam / Not Spam

Tanh (Hyperbolic Tangent):

Formula:

tanh(x)=ex−e−x/ex+e−x

Range: -1 to +1
Use: Hidden layers

Pros

- Zero-centered (better than sigmoid)

Cons

- Still suffers from vanishing gradient

Example: Sentiment (negative to positive)

ReLU (Rectified Linear Unit):

Formula:

f(x)=max(0,x)

Range: 0 to ∞
Use: Most popular for hidden layers

Pros

- Very fast
- Solves vanishing gradient (mostly)

Cons

- “Dead neurons” for negative inputs

Example: Image recognition, deep networks

Day-36

1. BFS (Breadth First Search):

Definition:

Breadth First Search (BFS) is a graph traversal algorithm that visits nodes **level by level**, starting from a given source node.

Formula / Key Concept

- Uses Queue (FIFO)
- Time Complexity:
- O(V+E)
- where
  V = number of vertices,
  E = number of edges

2. Path Finding Algorithm:

Definition:

A Path Finding Algorithm is used to determine the **optimal path** between a start node and a destination node in a graph or grid.

Formula / Key Concept

- Path Cost:

$$\text{Total Cost} = \sum \text{Edge Weights}$$

- For BFS (unweighted graph):

$$\text{Shortest Path} = \text{Minimum number of edges}$$

3. GMM (Gaussian Mixture Model):

Definition:

Gaussian Mixture Model (GMM) is a probabilistic clustering algorithm that represents data as a mixture of multiple Gaussian distributions.

Day - 37

- A new mentor joined the class.
- She instructed us to start working on our project.
- She clearly explained the project guidelines.
- She emphasized that the project must be completed on time.
- She instructed us to submit the project within the given deadline.

Day - 38

- Mam instructed us to make the project 100% ready for Milestone 4.
- Mam asked us to prepare a group PPT for project presentation.
- Project source code must be uploaded:
  - In individual GitHub repository
  - In team GitHub repository

- She asked us to share our feedback/thoughts about the internship classes conducted so far.

Day - 39 & 40

- Mam reviewed project work in detail.
- She checked our project PPT, documentation, and source code.
- The group project repository was verified.
- Our individual GitHub repositories were also checked.
- She ensured that all required files were properly submitted and updated.