PYTHON

Overview of Python

- Why Python?
- Comparison with R, MATLAB, and other languages
- Applications in Machine Learning, AI, and Big Data

2. Installing Python and Setting Up the Environment

- Downloading and installing Python (Windows, Mac, Linux)
- Setting up Anaconda for Data Science
- Using pip and conda for package management
- Understanding virtual environments (venv, conda env)

3. Jupyter Notebook, Anaconda, Google Colab, VS Code

- Introduction to Jupyter Notebook
- Creating and running notebooks
- Magic commands and markdown cells
- Google Colab: Cloud-based Python environment
- Importing datasets from Google Drive
- VS Code and Python extensions for development
- Terminal vs. IDE-based execution

4. Introduction to Python Syntax and Basic Constructs

- Writing and executing Python scripts
- Understanding indentation and comments
- Python's interactive shell (python, IPython)

5. Variables, Data Types, and Operators

- Declaring and using variables
- Numeric types: int, float, complex
- Text data: str
- Boolean values: True, False
- Type conversion (int(), float(), str())
- Operators:
- Arithmetic: +, -, *, /, %, **
- Comparison: ==, !=, <, >, <=, >=
- Logical: and, or, not
- Assignment: =, +=, -=, *=, /=
- Membership operators: in, not in

6. Conditional Statements (if, else, elif)

- Writing conditional statements
- Nested if statements
- Ternary operators

7. Loops (for, while, break, continue)

- Using for loops with sequences (lists, tuples, strings)
- Using while loops for repetitive execution
- Loop control statements: break, continue, pass

8. Lists, Tuples, and Dictionaries

- Creating lists and tuples
- Accessing elements using indexing and slicing
- Adding, updating, and removing elements
- Iterating over lists and tuples
- Understanding dictionary key-value pairs

- Adding, updating, and deleting dictionary entries

9. Creating and Manipulating Lists

- List methods: append(), extend(), insert(), pop(), remove()
- Sorting and reversing lists (sort(), reverse())
- List comprehension for data manipulation

10. Dictionary Operations and Key-Value Pairs

- Using keys(), values(), items()
- Dictionary comprehension for quick transformations

11. Sets and String Manipulation

- Creating and using sets
- Set operations: Union, Intersection, Difference
- Removing duplicates from lists using sets

12. String Operations

- String formatting (format(), f-strings)
- Splitting, joining, and replacing strings
- Regular expressions using the re module

13. Defining Functions and Lambda Functions

- Defining functions using def
- Function arguments (positional, keyword, default)
- Returning values from functions
- Using *args and **kwargs
- Anonymous functions using lambda

14. Working with Modules and Packages (math, os, sys)

- Importing and using built-in modules (math, os, sys)
- Creating and importing user-defined modules
- Installing external libraries using pip

15. Exception Handling (try-except blocks)

- Handling runtime errors using try-except
- Using finally for cleanup operations
- Raising custom exceptions with raise

16. Iterators, Generators, and Decorators

- Understanding iter() and next()
- Creating custom iterators
- Using yield to create generators
- Understanding Python decorators
- Implementing function decorators (@staticmethod, @classmethod)

17. Reading and Writing Text Files (open(), read(), write())

- Opening files in different modes (r, w, a, rb, wb)
- Reading and writing text files
- Using with statement for safe file handling

18. Working with CSV Files (csv module)

- Reading CSV files using csv.reader()
- Writing CSV files using csv.writer()
- Using Pandas for CSV file manipulation (read_csv(), to_csv())

19. JSON Data Handling (json module)

- Understanding JSON format
- Reading JSON data from files (json.load())
- Writing JSON data (json.dump())
- Converting Python objects to JSON (json.dumps())

DJANGO

1. Introduction to Django

- What is Django?
- Features of Django
- MVC vs MVT Architecture
- Installation and Setup
- Creating a Django Project
- Understanding manage.py and Django Project Structure
- Running the Django Development Server

2. Django Basics

- Creating a Django App
- Understanding settings.py, urls.py, views.py, models.py
- Working with Django Shell
- Understanding Django Requests and Responses
- URL Routing and Regular Expressions

3. Django Models & Database Handling

- Understanding Django Models
- Creating and Managing Models
- Migrations (makemigrations and migrate)
- Working with SQLite, PostgreSQL, MySQL
- CRUD Operations with Django ORM
- Querying the Database (filter(), get(), exclude(), Q objects)
- Foreign Key and Many-to-Many Relationships
- Using Django Admin Panel

Real-Time Project 1: Blog Application

- User authentication (login/logout/signup)
- Create, update, delete blog posts
- Category and tags for posts
- Comment system
- Admin panel customization

4. Django Views & Templates

- Function-Based Views (FBVs)
- Class-Based Views (CBVs)
- Template Rendering
- Template Inheritance
- Static Files (CSS, JS, Images)
- Template Filters and Tags
- Form Handling in Django
- Django Form Validation

Real-Time Project 2: Portfolio Website

- Dynamic portfolio sections
- Contact form with email functionality
- Admin-controlled content update

5. Django Forms & ModelForms

- Django Forms Basics

- ModelForms for CRUD Operations
- Form Validation
- Handling File Uploads (Images, PDFs)
- Customizing Form Fields and Widgets

Real-Time Project 3: Online Job Portal
- Employer and Job Seeker Registration
- Job Listings & Applications
- Resume Upload
- Email Notifications

6. Django Authentication & Authorization
- Django User Model
- User Registration & Login System
- Password Reset and Change
- User Roles and Permissions
- Custom User Model
- Social Authentication (Google, Facebook)

Real-Time Project 4: E-commerce Website (Part 1)
- User Authentication (Signup/Login)
- Profile Management

7. Django REST Framework (DRF) - API Development
- What is Django REST Framework?
- Serializers
- API Views
- Function-Based vs. Class-Based API Views
- Authentication & Permissions (JWT, Token Authentication)
- Pagination, Filtering, and Throttling
- Testing APIs with Postman

Real-Time Project 5: Task Management System (REST API)
- User Authentication
- CRUD APIs for Tasks
- Assigning Tasks to Users
- Status Updates

8. Django Middleware & Signals
- Custom Middleware in Django
- Built-in Middleware Functions
- Django Signals - Pre-save & Post-save

9. Django Caching & Performance Optimization
- Using Memcached & Redis
- Database Indexing & Query Optimization
- Asynchronous Tasks with Celery

10. Django Deployment & DevOps
- Preparing Django Project for Deployment

- Configuring Gunicorn & Nginx
- Deploying on AWS, Heroku, DigitalOcean
- CI/CD Pipelines with GitHub Actions
- Dockerizing a Django Project

Real-Time Project 6: E-commerce Website (Part 2)

- Shopping Cart System
- Order & Payment Integration
- Admin Dashboard
- Deployment on AWS/Heroku

NUMPY

Overview of NumPy
1.1 What is NumPy
1.2 Importance of NumPy in scientific computing
1.3 Key features: speed, efficiency, and functionality
1.4 Comparison between Python lists and NumPy arrays
1.5 Applications in data science, machine learning, and numerical analysis
Installing and Importing NumPy
2.1 Installation using pip and conda
2.2 Importing NumPy with alias (import numpy as np)
2.3 Verifying the NumPy version (np.__version__)
Creating NumPy Arrays
3.1 Array Creation from Existing Data
3.1.1 Creating arrays from Python lists or tuples using np.array()
3.1.2 Specifying the data type using the dtype parameter
3.2 Using Built-in Functions to Create Arrays
3.2.1 np.zeros() – Arrays filled with zeros
3.2.2 np.ones() – Arrays filled with ones
3.2.3 np.empty() – Uninitialized arrays
3.2.4 np.full() – Arrays filled with a specific value
3.2.5 np.arange() – Arrays with a range of values
3.2.6 np.linspace() – Arrays with evenly spaced values
3.2.7 np.logspace() – Arrays with logarithmically spaced values
Creating Random Arrays
4.1 Using np.random for random number generation
4.2 Generating random integers, floats, and arrays (np.random.rand(), np.random.randint())
4.3 Setting seeds for reproducibility (np.random.seed())
Array Properties
5.1 Checking the shape, size, and dimensions of an array
5.2 Array data types and dtype
Arithmetic Operations
6.1 Element-wise addition, subtraction, multiplication, and division
6.2 Broadcasting in NumPy arrays
Mathematical Functions
7.1 Sum, mean, median, min, max, and standard deviation
7.2 Element-wise functions like np.exp(), np.sqrt(), and np.log()
Comparisons
8.1 Element-wise comparisons (>, <, ==)

PANDAS

Real-World Applications
28. Case Studies
28.1 Analysing real-world datasets: Titanic dataset, financial data, sales data
29. Integration with Other Libraries
29.1 Using Pandas with Scikit-learn for machine learning preprocessing
29.2 Exporting cleaned data to formats like Excel and SQL

MATPLOT LIB AND SEABORN

Module 1: Introduction to Data Visualization
1.1 Overview of Data Visualization
- Importance of visualization in data analysis
- Types of data visualization (static vs dynamic)
- Key principles of effective data visualization
- Tools for data visualization in Python

1.2 Introduction to Matplotlib
- Overview of Matplotlib and its history
- Installing and setting up Matplotlib
- Anatomy of a Matplotlib plot (Figure, Axes, Axis)
- Basic syntax and structure of a Matplotlib plot

Module 2: Basic Plotting with Matplotlib
2.1 Basic Plotting Concepts
- Line plots
- Scatter plots
- Bar plots
- Histogram plots
- Pie charts

2.2 Matplotlib Components
- Figure and Axes
- Plot, Subplot, and Grid
- Axis, ticks, and labels
- Titles, labels, and legends

2.3 Customizing Plots
- Customizing colors, markers, and line styles
- Adjusting axis limits and scales (linear, logarithmic)
- Adding gridlines
- Annotating plots (text and arrows)

2.4 Advanced Plotting
- Creating subplots and multi-plot grids
- 3D plotting (3D scatter, surface plots)
- Saving and exporting plots (PNG, PDF, SVG)
- Practical Exercise:
- Create multiple plots (line, scatter, bar, histogram) with customization and annotation.

Module 3: Introduction to Seaborn
3.1 Overview of Seaborn
- Difference between Matplotlib and Seaborn
- Installing Seaborn
- Introduction to Seaborn’s high-level interface

- Seaborn's built-in themes and color palettes

3.2 Seaborn Basics

- Creating simple plots with Seaborn (line, scatter, bar, etc.)
- Understanding Seaborn's syntax and default styling
- Visualizing categorical vs continuous data with Seaborn

Module 4: Advanced Data Visualizations with Seaborn

4.1 Seaborn's Advanced Plot Types

- Pair plots (pairwise relationships)
- Heatmaps (correlation matrix, clustered heatmap)
- Box plots, violin plots, and distribution plots
- Swarm plots and strip plots
- Facet grids (subplots based on categories)

4.2 Customization in Seaborn

- Customizing color palettes
- Customizing plot styles and themes
- Adjusting axes and legends in Seaborn plots
- Working with multiple categorical variables

4.3 Integration of Seaborn with Pandas

- Visualizing data directly from Pandas DataFrames
- Handling missing data in visualizations
- Aggregating data for Seaborn plots

Module 5: Comparing Matplotlib and Seaborn

5.1 When to Use Matplotlib vs Seaborn

- Strengths of Matplotlib
- Strengths of Seaborn
- Performance and flexibility considerations
- Combining Matplotlib and Seaborn for customized visualizations

5.2 Case Study: Visualization with Real Data

- Using both Matplotlib and Seaborn to visualize a real-world dataset (e.g., sales data, weather data, or a dataset from Kaggle)
- Visualizing trends, distributions, and relationships in the data
- Customizing the visualization for publication-quality plots

Module 6: Interactive Visualization (Optional)

6.1 Introduction to Interactive Visualization

- Importance of interactive visualizations
- Using Plotly for interactive plotting
- Integrating Matplotlib/Seaborn with Plotly

6.2 Basic Interactive Plots

- Creating interactive line, scatter, and bar plots
- Adding interactivity (zooming, hovering, etc.)
- Saving interactive plots as HTML