

# 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 (>, <, ==)

8.2 Logical operations (np.any(), np.all())

## **Introduction to Universal Functions (ufuncs)**

9.1 What are ufuncs

9.2 Importance of ufuncs in NumPy for efficient computations

## **Commonly Used Universal Functions**

10.1 Trigonometric functions: np.sin(), np.cos(), np.tan()

10.2 Rounding functions: np.round(), np.floor(), np.ceil()

10.3 Aggregate functions: np.sum(), np.mean(), np.prod()

## **Customizing Universal Functions**

11.1 Using out parameter for in-place operations

11.2 Applying universal functions on multidimensional arrays

## **Indexing and Slicing Arrays**

12.1 Selecting and Retrieving Data

12.2 Slicing

12.3 Advanced Indexing

## **Iterating Over NumPy Arrays**

13.1 Iterating Over 1D Arrays

13.2 Iterating Over 2D and Higher-dimensional Arrays

13.3 Iterating with Enumerations

## **Array Shape Manipulation**

14.1 Reshaping Arrays

14.2 Transposing and Swapping Axes

14.3 Expanding and Squeezing Dimensions

## **Combining and Splitting Arrays**

15.1 Stacking Arrays

15.2 Splitting Arrays

## **Copies and Views**

16.1 Understanding Memory Sharing in NumPy

16.2 Creating Copies

16.3 Practical Examples

## **Advanced Indexing Techniques**

17.1 Indexing with Boolean Arrays

17.2 Indexing with Arrays of Indices

## **Broadcasting in NumPy**

18.1 Introduction to Broadcasting

18.2 Practical Examples

## **Linear Algebra with NumPy**

19.1 Matrix Operations

19.2 Advanced Linear Algebra

## **Random Numbers and Simulation**

20.1 Generating Random Numbers

20.2 Seeding Random Generators

## **Saving and Loading Data**

21.1 Saving Arrays

21.2 Loading Arrays

## **Performance Optimization with NumPy**

22.1 Vectorization

22.2 Memory Efficiency

# PANDAS

## Introduction to Pandas

1. What is Pandas?
  - 1.1 Overview of the Pandas library
  - 1.2 Importance of Pandas in data analysis and machine learning
  - 1.3 Key components of Pandas: Series and DataFrame
2. Setting Up the Environment
  - 2.1 Installing Pandas using pip or conda
  - 2.2 Installing Anaconda and Jupyter Notebook
  - 2.3 Importing Pandas and understanding the pd alias
3. First Steps with Pandas
  - 3.1 Creating Series and DataFrames manually
  - 3.2 Loading data from various formats: CSV, Excel, JSON, HTML, SQL, Pickle
  - 3.3 Exploring datasets with head(), tail(), info(), and describe()

## Pandas Basics

4. Data Structures
  - 4.1 Series: One-dimensional labeled arrays
  - 4.2 DataFrame: Two-dimensional labeled data structures
  - 4.3 Index objects: Labels for rows and columns
5. Basic DataFrame Operations
  - 5.1 Accessing rows and columns using loc and iloc
  - 5.2 Modifying data: Adding and removing rows or columns
  - 5.3 Transposing data using .T
6. Data Selection and Slicing
  - 6.1 Selecting rows and columns by labels or positions
  - 6.2 Conditional selection using Boolean indexing
7. Data Types in Pandas
  - 7.1 Identifying data types with dtypes
  - 7.2 Converting data types using astype()
  - 7.3 Handling categorical data

## Data Cleaning and Preprocessing

8. Handling Missing Values
  - 8.1 Identifying missing data with isnull() and notnull()
  - 8.2 Dropping missing values using dropna()
  - 8.3 Filling missing values with fillna() (mean, median, or specific values)
9. Renaming Columns and Rows
  - 9.1 Using rename() for labels
  - 9.2 Renaming directly
10. Removing Duplicates
  - 10.1 Identifying duplicates with duplicated()
  - 10.2 Dropping duplicates using drop\_duplicates()
11. Replacing Data
  - 11.1 Using replace() to substitute values

## Data Transformation

12. Sorting Data
  - 12.1 Sorting by index using sort\_index()
  - 12.2 Sorting by values using sort\_values()

- 13. Merging, Joining, and Concatenation
  - 13.1 Combining datasets using merge() for SQL-like joins
  - 13.2 Using concat() for stacking datasets vertically or horizontally
  - 13.3 Using join() for index-based joining
- 14. GroupBy Operations
  - 14.1 Aggregation functions (sum, mean, count, etc.)
  - 14.2 Grouping data based on multiple columns
  - 14.3 Custom aggregation functions
- 15. Pivot Tables
  - 15.1 Creating pivot tables for multi-dimensional aggregation
  - 15.2 Difference between pivot and groupby
- 16. Melt and Stack
  - 16.1 Using melt() to convert wide to long format
  - 16.2 Transforming hierarchical indices using stack() and unstack()

### **Advanced Techniques**

- 17. Apply and Map Functions
  - 17.1 Element-wise operations with apply(), map(), and applymap()
- 18. Crosstab
  - 18.1 Creating summary tables of categorical data using pd.crosstab()
- 19. Cut and Binning
  - 19.1 Creating intervals for numerical data using cut()
  - 19.2 Equal-width vs. custom-width bins
- 20. Datetime Operations
  - 20.1 Converting columns to datetime using to\_datetime()
  - 20.2 Extracting date components (year, month, day, hour, etc.)
  - 20.3 Performing date arithmetic and filtering
- 21. Handling Large Datasets
  - 21.1 Optimizing memory usage
  - 21.2 Processing large datasets in chunks using iterators

### **Data Analysis and Visualization**

- 22. Exploratory Data Analysis (EDA)
  - 22.1 Performing summary statistics
  - 22.2 Identifying trends and patterns in data
- 23. Data Visualization
  - 23.1 Plotting using Pandas: Line, bar, histogram, scatter, and box plots
  - 23.2 Integrating Pandas with Matplotlib and Seaborn
- 24. Boolean Indexing
  - 24.1 Advanced conditional filtering
  - 24.2 Combining multiple conditions using logical operators

### **Performance Optimization**

- 25. Vectorized Operations
  - 25.1 Using NumPy under the hood for faster computations
- 26. Working with Sparse Data
  - 26.1 Reducing memory usage with sparse matrices
- 27. Profiling and Debugging
  - 27.1 Profiling performance with %timeit
  - 27.2 Debugging slow code



## **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

# **MATPLOTLIB 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