Python Course Content

Chapter 1: Python Introduction & Installation

- What is Python & Languages?
- Why Python?
- Writing your first Python program
- Features and advantages of Python
- Different Python versions (2.x vs 3.x)
- Installing Python
- Application of Python
- Setting up IDEs (IDLE, VS Code, PyCharm, etc.)
- Running Python scripts (command line, interactive mode)

Practicals:

- Install Python and print “Hello, World!”
- List a few real-world applications of Python.
- Write a Python program that prints your name and age.

Chapter 2: Syntax, Variables and Data Types

- Python syntax rules & indentation
- Naming conventions
- Variables and dynamic typing
- Built-in data types: int, float, str, bool, complex
- Type checking with type()
- input() function and reading user input
- Type casting (int(), float(), str())

Practicals:

- Create variables for name, age, and height and print them
- Convert int to float and vice versa
- What are the rules for naming a variable in Python?
- Identify the data types: "25", 25.0, True, 'A'
- Write a program that swaps two numbers.

Chapter 3: Operators

- Arithmetic operators (+, -, *, /, %, **)
- Comparison operators (==, !=, >, <, >=, <=)
- Logical operators (and, or, not)
- Assignment operators (+=, -=, etc.)
- Bitwise operators (optional / advanced)

Practicals:

- Create a calculator to add, subtract, multiply, and divide two numbers
- Accept user input for age and check eligibility to vote.
- Write a program to get user input for two numbers and display their sum.
- Write a program to find the area of a rectangle from user input.

Chapter 4: Control Flow (Conditionals) & Loops (for and while)

- if, elif, else statements
- Nested conditionals
- for loops: iterating over sequences
- while loops and loop control
- break, continue, and else in loops
- Looping with range()
- Comprehensions (basic intro)

Practicals:

- Write a program to find if a number is positive, negative, or zero
- Print even numbers from 1 to 100
- Use while loop to implement a simple menu
- Write a program to check if a number is divisible by 3 and 5.
- Create a program that prints the multiplication table of a number.
- Use a loop to print Fibonacci sequence up to n terms.

Chapter 5: Data Structures (List, Tuples, Set, Dictionary)

- Lists: creation, indexing, slicing, methods
- Tuples: immutability and usage

- Sets: unique elements, set operations
- Dictionaries: key-value pairs, methods
- Nested data structures
- Mutable vs Immutable & Examples
- Common functions: len(), sorted(), etc.

Practical:

- Manage a contact book using a dictionary
- Find unique words in a text with sets
- Combine lists and dictionaries to store student data

Chapter 6: Functions and Modules

- Defining functions with def
- Function parameters and return values
- Default & keyword arguments
- *args and **kwargs
- Recursion Function
- Lambda Function
- Importing modules
- Creating and using custom modules
- Built-in modules (math, random, datetime, etc.)

Practicals:

- Write a function to calculate factorial
- Create a module with multiple math functions and import it
- Write a function that checks if a number is prime.
- Write a recursive function to calculate Fibonacci numbers.

Chapter 7: Classes and Objects

- Introduction to Object-Oriented Programming (OOP)
- Object-oriented programming basics
- Defining classes and creating objects
- __init__ constructor method
- Instance variables and methods
- Class variables and methods (@classmethod)

- @staticmethod
- __str__ and other dunder methods

Practicals:

- Create a Car class with attributes and methods
- Define a class Student with a method to display details
- Write a class to represent a bank account with deposit and withdraw methods.
- Create a class to store and display employee details.

Chapter 8: Inheritance & Abstraction

- Inheriting from a parent class
- Overriding methods
- Using super()
- Abstract classes and methods (abc module)
- When to use abstraction

Practicals:

- Create a base class Person and derive a class Student
- Write a program to show multilevel inheritance with three classes.
- Implement multiple inheritance using two base classes.
- Use super() to call the parent class method.

Chapter 9: Polymorphism & Encapsulation

- Concept of polymorphism
- Method overriding and method overloading (Python style)
- Duck typing
- Encapsulation and private/protected variables
- Getters and setters (property decorators)

Practicals:

- Write a program to demonstrate polymorphism with a speak() method in Dog and Cat classes.
- Create a class hierarchy with overridden methods
- Use property decorators to protect data

Chapter 10: Exception Handling & File Handling

- Exceptions and errors in Python
- Using try, except, else, finally
- Raising exceptions
- Creating custom exceptions
- Reading and writing text files
- File modes (r, w, a, etc.)

Practical:

- Read names from a file and print them
- Write user input to a text file

Chapter 11: Advanced Python - decorators & Multithreading

- Introduction to decorators
- Writing custom decorators
- Decorators with arguments
- Built-in decorators (@property, etc.)
- Introduction to multithreading
- threading module basics

Practical:

- Write a decorator to log function calls
- Create threads to run functions in parallel

Chapter 12: Tkinter/GUI Programming

- Introduction to GUI development
- Tkinter basics: windows, labels, buttons
- Layout management (pack, grid)
- Event handling and callbacks
- Adding input fields, menus, and dialogs
- Simple projects (calculator, text editor)

Practical:

- Build a basic calculator GUI
- Make a to-do list app with add/delete items

Chapter 13: Data Analysis (Numpy,panda,matlilap)

- Introduction to data analysis workflow
- NumPy arrays and operations
- pandas DataFrames and Series
- Importing/exporting CSV, Excel
- Data cleaning and preprocessing
- Data visualization with matplotlib and seaborn
- Basic statistics and aggregation

Practical:

- Plot a bar chart of product sales
- Clean missing data

Chapter 14: Databases in Python (MysQL, CURD Application)

- What is a database & SQL basics
- Connecting Python to MySQL (mysql-connector / PyMySQL)
- Performing CRUD operations (Create, Read, Update, Delete)
- Parameterized queries to prevent SQL injection
- Using SQLite (optional / bonus)
- Simple console-based CRUD application

Practical:

- Build a Python app to add, edit, delete records in a MySQL table

Chapter 15: GitHub Project & Interview Preparation & Projects