Python Programming with Testing, Networking & Git Fundamentals

Duration: 5 days (40 hours)
Prerequisites: Knowledge of any OOPs-based programming language

Day 1 – Python Basics & Data Structures

Objective: Understand Python setup, syntax, variables, and core data structures.

📘 Module 1: Introduction & Environment Setup

Topics & Subtopics

- Python Overview
  - Use cases (Web, Automation, Data)
  - Python execution model (Interpreter)
- Installation & Setup
  - Installing Python (Windows/Linux)
  - Setting environment variables
- IDE Setup
  - PyCharm installation & configuration
  - Creating first project & running script

📘 Module 2: Python Syntax & Variables

Topics & Subtopics

- Python Syntax Rules
  - Indentation & code blocks
- Variables
  - Naming conventions
  - Dynamic typing
- Data Types
  - int, float, string, NoneType
  - Type casting & type checking

📘 Module 3: Data Structures

Topics & Subtopics

- Lists
  - Indexing, slicing
  - Methods (append, remove, sort)
- Tuples
  - Immutability & use cases

- Dictionaries
    - Key-value operations
    - Nested dictionaries
- Sets
    - Unique elements
    - Union, intersection

📘 Module 4: Operators & Booleans

Topics & Subtopics

- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Boolean expressions & evaluation

Labs (Detailed)

- Install Python & PyCharm and configure project
- Create a script to store and print employee details using different data types
- Write a program to accept user input and perform arithmetic operations
- Create a list of marks and perform slicing, sorting, and updating values
- Build a dictionary-based contact manager (Add/Search/Delete contact)
- Remove duplicate values from a list using set operations
- Evaluate complex boolean expressions using user input

Day 2 – Control Flow, Functions & Scope

Objective: Build logical programs using conditions, loops, and reusable functions.

📘 Module 1: Conditional Statements

Topics & Subtopics

- if, elif, else
- Nested conditions
- Decision-making scenarios

📘 Module 2: Loop Structures

Topics & Subtopics

- for loop
    - Iterating lists, dicts
- while loop

- break, continue, pass

📘 Module 3: Functions

Topics & Subtopics

- Function definition & calling
- Parameters
    - Positional, keyword, default
    - Variable-length (*args, **kwargs)
- Return values
- Recursion basics

📘 Module 4: Scope & Lifetime

Topics & Subtopics

- Local vs Global variables
- Variable lifetime
- Scope resolution

🧪 Labs (Detailed)

- Create a grading system using nested if-else
- Build a number guessing game using while loop
- Write a program to print patterns using loops
- Create a calculator using functions (add, subtract, multiply, divide)
- Implement factorial using recursion
- Write a function using *args to calculate sum of multiple numbers
- Demonstrate global vs local variable behavior with example

Day 3 – OOP, Exception Handling & File Operations

Objective : Implement object-oriented programming and handle files/errors.

📘 Module 1: OOP Concepts

Topics & Subtopics

- Classes & Objects
- Attributes & Methods
- Constructor (**init**)
- Inheritance
- Polymorphism (method overriding)
- Encapsulation basics

📘 Module 2: Exception Handling

Topics & Subtopics

- Types of errors
- try, except, finally
- Multiple exceptions
- Custom exceptions

📘 Module 3: File Handling

Topics & Subtopics

- File modes (r, w, a)
- Reading & writing files
- Working with text data

📘 Module 4: Iterators & Built-in Functions

Topics & Subtopics

- Iterators & iteration protocol
- Built-in functions (map, filter, zip, len, sum)

📘 Module 5: Package Management

Topics & Subtopics

- Installing packages using pip
- Virtual environments (venv)
- Dependency management basics

Labs (Detailed)

- Create a class Employee with attributes and methods
- Implement inheritance (Manager extends Employee with additional attributes)
- Demonstrate polymorphism using method overriding
- Handle division by zero using try-except
- Create a custom exception for invalid salary input
- Write employee data into a file and read it back
- Use map/filter to process a list of numbers (square, even filter)
- Install and use an external package using pip

Day 4 – Testing & Networking with Python

Objective : Learn testing practices and implement networking using Python.

📘 Module 1: Software Testing Basics

Topics & Subtopics

- Importance of testing
- Types of testing (Unit, Integration)

📘 Module 2: Unit Testing with PyTest

Topics & Subtopics

- PyTest introduction
- Writing test functions
- Assertions
- Fixtures
- Parametrization

📘 Module 3: Networking Fundamentals

Topics & Subtopics

- Client-server architecture
- HTTP protocol basics
- APIs and JSON

📘 Module 4: Requests Library

Topics & Subtopics

- GET & POST requests
- Handling JSON responses
- Status codes & error handling

📘 Module 5: Socket Programming

Topics & Subtopics

- Socket basics
- TCP client-server model
- Data transmission

🧪 Labs (Detailed)

- Write PyTest test cases for calculator functions
- Create reusable test setup using fixtures
- Implement parameterized test cases
- Call a public API using requests and print response data
- Build a mini API client to fetch and display JSON data
- Handle API errors (invalid URL, timeout)
- Create a simple TCP server
- Build a client program to communicate with server

Day 5 – Git & Collaboration

Objective: Manage code versions and collaborate using Git.

📘 Module 1: Git Introduction

Topics & Subtopics

- Version control concepts
- Git workflow

📘 Module 2: Git Basics

Topics & Subtopics

- git init, add, commit
- git status, log

📘 Module 3: Branching & Merging

Topics & Subtopics

- git branch, checkout
- Merging branches
- Best practices

📘 Module 4: Remote Repositories

Topics & Subtopics

- GitHub/GitLab overview
- git clone, push, pull

📘 Module 5: Conflict Resolution

Topics & Subtopics

- Merge conflicts
- Resolving conflicts
- Team collaboration workflow

Labs (Detailed)

- Initialize Git repo and commit Python project
- Track changes and analyze commit history
- Create feature branch and merge into main
- Push project to GitHub repository
- Pull latest updates and resolve conflicts
- Simulate team workflow with multiple branches and conflict resolution