Python Programming Course Syllabus

Course Overview

Welcome to the **Python Programming Course**, a comprehensive journey from beginner to advanced Python development! This course covers **Python 3.12** (the latest release as of August 2025) and equips you with skills to build real-world applications in **web development**, **data science**, **automation**, and more. Through hands-on projects, case studies, and modern tools, you'll master Python and prepare for a career in tech.

- **Duration**: 12 weeks (3 months) (4–6 hours/week)
- **Course Fee**: Rs. 15,000 INR
- **Format**: Lectures, coding exercises, projects, and quizzes
- **Prerequisites**: Basic computer skills (no prior programming required)

Target Audience: Beginners to intermediate learners aiming to become Python developers

---

Course Objectives

By the end of this course, you'll:

- ✅ Master Python syntax, data structures, and object-oriented programming.
- ✅ Build modular, reusable code with functions and packages.
- ✅ Create data-driven applications using Pandas and Matplotlib.
- ✅ Develop web applications with Django and deploy them.
- ✅ Apply modern Python features like type hints and asynchronous programming.
- ✅ Solve real-world problems through hands-on projects.

📚 Course Structure

The course is divided into **15 modules**, each with clear learning outcomes, hands-on exercises, and case studies. Weekly quizzes and a capstone project ensure practical mastery.

📖 Detailed Syllabus

🌟 Module 1: Introduction to Programming and Python

Duration: 1 Week

Objective: Understand programming basics and Python's versatility.

- **What is Programming?**
  - 💻 Types of languages: High-level vs. low-level
  - 🔄 Translators: Compiler vs. Interpreter
  - 📜 Scripting vs. programming languages
  - 🧠 Paradigms: Procedural, Object-Oriented, Functional
- **Why Python?**
  - 🌍 Python's history, features, and real-world applications
  - 🆕 Python 3.12 updates (improved type hints, exception groups)
  - 🆚 Python 2.x vs. 3.x, 3.11 vs. 3.12
  - 💼 Industry use cases: Web, AI, automation
- **Setup and Tools**
  - 📦 Python distributions: CPython, Anaconda
  - 🛠️ Installation: Windows, macOS, Linux
  - 🌐 IDEs: VS Code, PyCharm, Jupyter Notebook
  - 📌 Virtual environments: `virtualenv` , `poetry`
- 📚 **Case Study**: Set up a Python environment with Poetry and VS Code

🛠️ Module 2: Python Language Fundamentals

Duration: 1 Week

Objective: Learn Python's core syntax and structure.

- **Python Basics**
  - 🔑 Keywords, identifiers, literals
  - 📊 Data types: `int` , `float` , `str` , `bool` , `bytes`
  - 🆚 Python vs. Java
  - 📝 Python syntax
- **Running Python**
  - 🖱️ Interactive mode (REPL)
  - 📜 Scripting mode
  - 🐞 Debugging with `pdb` and `breakpoint()`
- **Variables and I/O**
  - 🌍 Local, global, `nonlocal` variables
  - 🔤 Input/output: `input()` , `print()`
  - 🔄 Type conversion: `int()` , `float()` , `str()`
  - 🖥️ Command-line arguments with `argparse`
- 📚 **Case Study**: Build a command-line calculator

⚙️ Module 3: Operators and Control Structures

Duration: 1 Week

Objective: Master operators and control flow for decision-making.

- **Operators**
  - ➕ Arithmetic, comparison, assignment
  - 🔍 Logical, bitwise, membership (`in`), identity (`is`)
  - 🔢 Ternary operator, operator precedence
- **Control Structures**
  - 🔄 Conditional: `if` , `if-else` , `if-elif-else` , nested `if`
  - 🔁 Loops: `for` , `while` , nested loops
  - 🛑 Branching: `break` , `continue` , `pass` , `return`
- 📚 **Case Study**: Create a number guessing game

📦 Module 4: Data Structures (Collections)

Duration: 2 Weeks

Objective: Manipulate Python's built-in data structures.

- **Overview**
  - 🗂️ Importance of data structures
  - 📋 Types: Sequence (`str` , `list` , `tuple` , `range`), Non-sequence (`set` , `dict` , `frozenset`)
- **Strings**
  - ✂️ Indexing, slicing, f-strings
  - 🔧 String methods, immutability

- **Lists**
  - 📋 Creation, comprehension, mutability
  - ✂️ Indexing, slicing, nested lists
  - 🔄 Shallow vs. deep copy, `zip()`
- **Tuples**
  - 🔒 Immutability, methods
  - 🆚 List vs. tuple
- **Sets**
  - 🔗 Operations: Union, intersection, difference
  - 🌐 Frozen sets
- **Dictionaries**
  - 🔍 Creation, comprehension, methods
  - 🔄 Accessing, updating, sorting
- 📚 **Case Study**: Build a contact management system

🛠️ Module 5: Functions

Duration: 1 Week

Objective: Write modular, reusable code with functions.

- **Function Basics**
  - 📌 Defining and calling functions
  - 🔄 Types: No args/no return, with args/with return
  - 🔁 Recursion, lambda functions
  - 🗺️ Functional tools: `map()` , `filter()` , `reduce()`

- **Advanced Functions**
  - 🔢 Default, keyword, `*args` , `**kwargs`
  - 🎨 Decorators, generators, iterators
  - 📝 Type hints with `typing` module
- 📚 **Case Study**: Create a decorator to log function execution time

---

📦 Module 6: Modules and Packages

Duration: 1 Week

Objective: Organize code with modules and packages.

- **Modules**
  - 📚 Pre-defined vs. user-defined
  - 📥 Importing: `import` , `from ... import`
  - 🔄 Module aliasing
- **Packages**
  - 🗂️ Creating and importing packages
  - 📦 Package vs. folder
  - 🛠️ Package management: `pip` , `poetry`
- 📚 **Case Study**: Build a modular project with a custom package

---

🏗️ Module 7: Object-Oriented Programming (OOP)

Duration: 2 Weeks

Objective: Build structured code using OOP principles.

- **OOP Basics**
  - 🧩 Classes, objects, `self` , `cls`
  - 🔒 Encapsulation, polymorphism, inheritance
  - 🔧 Instance, class, static methods
- **Advanced OOP**
  - 🔄 Method/constructor overriding
  - ➕ Operator overloading
  - 🌳 Inheritance types: Single, multilevel, multiple
  - 🔍 Abstract base classes (`abc`), method resolution order (MRO)
- 📚 **Case Study**: Design a library management system

🛡️ Module 8: Exception Handling

Duration: 1 Week

Objective: Handle errors gracefully. •

Exception Basics

- 🛑 Syntax vs. runtime errors
- 🚨 Common exceptions: `ValueError` , `IndexError`
- 🔧 `try` , `except` , `else` , `finally`

- **Advanced Exceptions**
  - 🔄 Handling multiple exceptions
  - ✂️ Custom exceptions with `raise`
  - 🆕 Exception groups (Python 3.12)
- 📚 **Case Study**: Build a file parser with error handling

🔍 Module 9: Regular Expressions

Duration: 1 Week

Objective: Extract data using pattern matching.

- **Regular Expressions** `re` module: `match()` , `search()` ,
  - 🔎 `findall()`
  - 📧 Patterns: Email, phone, URL
  - 🔣 Special characters, character classes
- 📚 **Case Study**: Extract emails and URLs from text

📂 Module 10: File and Directory Handling

Duration: 1 Week

Objective: Manage files and directories.

- **File Operations**
  - 📖 Read, write, append modes
  - 🛠️ `pathlib` vs. `os`
  - 📊 CSV, JSON, XML parsing
  - 📦 Serialization with `pickle`
- **Directory Operations**
  - 🗂️ Create, rename, delete directories
  - 🔄 `os` , `shutil` modules
- 📚 **Case Study**: Build a file organizer script

🌌 Module 11: Advanced Python Features

Duration: 1 Week

Objective: Explore advanced Python tools.

- **Logging**
  - 📝 Logging levels, custom loggers
- **Date and Time** • ⏰ `datetime` , `timedelta` , time zones
- **OS Module**
  - 🖥️ File system operations, shell commands
- **Multithreading & Async** `threading` vs.
  - 🔄 `multiprocessing`
  - 🆕 Async programming with `asyncio` (task groups in Python 3.12)
- **Garbage Collection** `gc` module,
  - 🗑️ manual collection
- 📚 **Case Study**: Build a multithreaded web scraper with logging

---

🌐 Module 12: Database and Network Programming

Duration: 1 Week

Objective: Connect Python to databases and networks.

- **Database Programming**
  - 🗄️ MySQL/PostgreSQL with `mysql-connector` , `psycopg2`
  - 🔄 CRUD operations, transactions
- **Network Programming**
  - 🔌 Sockets: `socket` module
  - 📡 Client-server applications
- 📚 **Case Study**: Build a database-backed inventory system

🖼️ Module 13: GUI Programming and Data Visualization

Duration: 1 Week

Objective: Create GUIs and visualize data.

- **GUI Programming**
  - 🖱️ `tkinter` : Widgets, layouts, event handling
  - 🐢 `turtle` for simple graphics
- **Data Visualization** `matplotlib` : Bar, scatter,
  - 📊 pie charts `seaborn` for enhanced
  - 🎨 visuals
- 📚 **Case Study**: Build a GUI-based data dashboard

📈 Module 14: Data Science with Python

Duration: 2 Weeks

Objective: Analyze and visualize data.

- **NumPy**
  - 📊 Arrays, indexing, slicing
  - ➕ Linear algebra, statistical functions
- **Pandas**
  - 📋 Series, DataFrame
  - 🔄 Merging, grouping, cleaning
- **SciPy**
  - 🔬 Scientific computing
- **Machine Learning Intro**
  - 🧠 ML types: Supervised, unsupervised `scikit-learn` basics
  - 🤖
- 📚 **Case Study**: Analyze a dataset and visualize results

🌐 Module 15: Web Development with Django

Duration: 1 Week

Objective: Build web applications.

- **Django Basics**
  - 🏛️ MVT pattern
  - 🛠️ Models, views, templates, URLs
- **Advanced Django**
  - 🌐 Django REST Framework for APIs
  - 🔒 Authentication, deployment with Docker
- 📚 **Case Study**: Build a blog application

🛠️ Projects and Assessments

- **Mini Projects** (Weekly)
  - 💻 Command-line calculator
  - 📋 Contact management system
  - 📂 File organizer
  - 🌐 Multithreaded web scraper
- **Capstone Project** (Final 2 Weeks)
  - 🚀 Build a web app or data pipeline (e.g., e-commerce site, data dashboard)
  - Integrates Django, Pandas, and databases
- **Assessments**
  - 📝 Weekly quizzes
  - 💻 Coding challenges (LeetCode, HackerRank)
  - 👥 Peer-reviewed project submissions

🛠️ Tools and Technologies

- **Python Version**: 3.12
- **IDEs**: VS Code, PyCharm, Jupyter Notebook
- **Package Managers**: `pip` , `poetry`
- **Libraries/Frameworks**: `numpy` , `pandas` , `matplotlib` , `seaborn` , `scikit-learn` , `django` , `tkinter` , `asyncio`
- **Databases**: MySQL, PostgreSQL
- **Version Control**: Git, GitHub
- **Deployment**: Docker, AWS, Heroku

🎉 Learning Outcomes

By the end, you'll be able to:

- ✍️ Write clean, efficient Python code.
- 🛠️ Build modular applications.
- 📊 Analyze and visualize data.
- 🌐 Develop and deploy web apps.
- 🚀 Apply modern Python features like `asyncio` and type hints.

📝 Additional Notes

- 🆕 Features Python 3.12 (type hints, exception groups, task groups).
- 📦 Modern tools: `poetry` , `pathlib` .
- 🌍 Real-world projects aligned with industry needs.
- 💻 Contribute to open-source on GitHub for hands-on experience.