Python Programming Syllabus Design

A Structured Academic Curriculum for Python Programming

Author: D Charan Jeet

Curriculum Designer & Technical Author

Target Audience: B.Tech / Computer Science Students
Prerequisites: Basic Computational Logic
Format: Modular Progressive Learning

January 29, 2026

Contents

Chapter 1

Level I: BASICS

Establishing the Foundation of Python Syntax

Module 1: Python Basics *Foundation | 10 Hours*

Learning Objectives

- Understand the features and applications of the Python language.
- Configure the development environment (IDEs and Interpreters).
- Master variables, data types, and basic I/O operations.

Conceptual Topics

- **Introduction to Python**
  - Features (Interpreted, Dynamic, High-level) & Applications.
  - Installing Python & IDEs (VS Code, PyCharm, Jupyter).
  - The Python Interpreter & Execution modes (Script vs Interactive).
- **Variables & Data Types**
  - Primitive Types: `int`, `float`, `str` (string), `bool` (boolean).
  - Variable naming conventions and dynamic typing.
  - Type Conversion: Implicit vs Explicit (`int()`, `float()`, `str()`).
- **Operators & I/O**
  - Input & Output: `input()` function, `print()` function.
  - Operators: Arithmetic, Relational, Logical, Assignment.

Practical Labs & Basic Programs

- "Hello World" script.
- Basic Calculator (Arithmetic operations).
- Simple Interest Calculator.
- Area & Perimeter calculation programs.

Expected Outcome: Students will be able to write and execute basic Python scripts, handle user input, and perform fundamental arithmetic operations.

Chapter 2

Level II: CORE CONCEPTS

Logic Building, Data Structures, and Modularity

Module 2: Control Flow & Data Structures *Core Concept 1 | 14 Hours*

Learning Objectives

· Construct logic using conditional statements and loops.

· Manage collections of data using Python's built-in data structures.

· Solve algorithmic problems efficiently.

Conceptual Topics

- **Control Flow**
  - **Conditional Statements:** `if`, `if-else`, `elif` ladder.
  - **Loops:** `for` loop (range, iterables), `while` loop.
  - **Control Statements:** `break`, `continue`, `pass`.
  - Nested Loops and pattern printing.
- **Python Data Structures**
  - **List:** Creation, Indexing, Slicing, Methods (append, pop, sort).
  - **Tuple:** Immutability, packing/unpacking.
  - **Set:** Unique elements, set operations (union, intersection).
  - **Dictionary:** Key-Value pairs, accessing methods (keys, values, items).

Real-world Applications

◦ Inventory management using Dictionaries.

◦ Filtering unique items using Sets.

◦ Logic-based games (Number guessing).

Expected Outcome: Proficiency in logic building and choosing the appropriate data structure for specific problems.

Module 3: Functions & Strings

Core Concept 2 | 12 Hours

Learning Objectives

· Write modular, reusable, and structured code.

· Perform advanced text manipulation and formatting.

· Understand the scope of variables and recursion.

Conceptual Topics

- **Functions in Python**
  - Defining (`def`) & Calling Functions.
  - Parameters: Positional, Keyword, Default, Variable-length (*args, **kwargs).
  - Return Values and `None`.
  - **Advanced Functions:** Lambda functions, Basic Recursion.
- **String Handling**
  - String Methods: `upper`, `lower`, `split`, `replace`, `find`.
  - Slicing & Indexing strings.
  - String Formatting: f-strings, `.format()`.
- **Modules & Packages**
  - Importing Modules (`import`, `from ... import`).
  - Standard Library overview.

Expected Outcome: Ability to decompose problems into functions and efficiently process textual data.

Chapter 3

Level III: ADVANCED TOPICS

Object-Oriented Design and System Interaction

Module 4: Advanced Python *Advanced | 12 Hours*

Learning Objectives

· Handle external files and manage runtime errors gracefully.

· Implement Object-Oriented Programming (OOP) principles.

· Utilize essential Python libraries for real-world tasks.

Conceptual Topics

- **File Handling & Exceptions**
  - Opening modes: Read (`r`), Write (`w`), Append (`a`).
  - Context Managers: `with open(...)  as file`.
  - **Exception Handling:** `try`, `except`, `finally`, `raise`.
- **Object-Oriented Programming (OOP)**
  - Classes & Objects: Blueprints vs Instances.
  - Attributes (Variables) & Methods (Functions).
  - The Constructor: `__init__` method.
  - Concepts of Self.
- **Basic Libraries**
  - `math`: Mathematical functions.
  - `random`: Generating random numbers.
  - `datetime`: Handling dates and times.

Real-world Applications

◦ Log file analyzer (File I/O).

◦ Banking System simulation (OOP).

◦ Digital Clock or Timer (datetime).

Expected Outcome: Mastery of creating robust, object-oriented applications that interact with the file system.

End of Syllabus – Designed by D Charan Jeet