

PROFESSIONAL CORE PYTHON PROGRAM | CAREER-FOCUSED CURRICULUM

Learning Roadmap

A visual path through the complete 7-module curriculum.

2 Months	7 Modules	Core + OOP + Projects
Program Duration	Complete Curriculum	Applied Learning

01 **Programming Fundamentals & Python Setup** — Foundation	02 **Control Flow & Functions** — Logic & Reusability
03 **Data Structures & Strings** — Working with Data	04 **Object-Oriented Programming** — Classes & Objects
05 **File Handling, Exceptions & Modules** — Persistence & Reliability	06 **Advanced Python & Standard Library** — Pythonic Programming

Program at a Glance

Key information about the Core Python program.

Program	Professional Core Python Program
Duration	2 Months (approx. 8 weeks), 7 modules
Learning Approach	Instructor-led sessions, hands-on labs, module assignments and portfolio projects
Eligibility	Graduates and final-year students from any stream; basic computer knowledge. No prior programming experience required
Technology Stack	Python 3, VS Code, Git and GitHub, pip, venv, Python Standard Library
Program Outcome	Confident Python programmer who can write clean, modular programs and build complete applications

Program Structure & Timeline

Module-wise duration, phase and primary tools, progressing from Python fundamentals to OOP, advanced concepts and projects.

No.	Module	Phase	Duration	Key Tools
01	**Programming Fundamentals & Python Setup**	Foundation	1 Week	Python 3, VS Code, pip, Git
02	**Control Flow & Functions**	Core Python	1 Week	Python 3, VS Code, Debugger
03	**Data Structures & Strings**	Core Python	1 Week	Strings, List, Tuple, Set, Dictionary
04	**Object-Oriented Programming**	Core Python	1 Week	Classes, Inheritance, Dunder Methods
05	**File Handling, Exceptions & Modules**	Core Python	1 Week	open(), json, csv, os, pathlib, venv, pip
06	**Advanced Python & Standard Library**	Advanced	1 Week	collections, itertools, datetime, re, logging, unittest
07	**Real-World Projects**	Projects	2 Weeks	Python 3, OOP, File Handling, JSON, CSV, Git
Total Program Duration			**8 Weeks**	Approx. 2 months

MODULE 01

Programming Fundamentals & Python Setup

01

Foundation-level introduction to programming, Python installation and the core building blocks: variables, data types, operators and input/output.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	**Beginner**	**Python 3, VS Code, pip, Git**	**Working Python environment and the ability to write basic programs**

Part 1: Getting Started

- Introduction to Programming
- Features and Uses of Python
- Python Installation
- Interpreter and REPL
- Running Scripts in VS Code
- Comments and Indentation
- Command Line Basics
- Version Control Basics (Git)

Part 2: Variables & Data Types

- Variables and Naming Rules
- int, float, complex
- str and bool
- None Type
- Type Checking and Casting
- input() and print()
- f-strings and Formatting

Part 3: Operators

- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Assignment Operators
- Bitwise Operators
- Membership and Identity

Software to Install

- Python 3 (latest stable)
- Visual Studio Code
- Git and GitHub
- pip (package manager)

Practice Programs

Simple Calculator **Temperature Converter** **Area and Perimeter Calculator**

WHAT THIS MODULE SUPPORTS

SETUP
Configure Python, VS Code and Git on your machine.

WRITE
Create and run your first Python scripts.

CALCULATE
Work with data types, operators and user input.

MODULE 02

Control Flow & Functions

02

Decision making, repetition and code reuse: conditional statements, loops, functions, scope, recursion and modules.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	Beginner	Python 3, VS Code, Debugger	Write logical, reusable programs using conditions, loops and functions

Part 1: Conditional Statements

- if, elif, else
- Nested Conditions
- Ternary Expression
- match-case Statement

Part 2: Loops

- for Loop
- while Loop
- range()
- break, continue, pass
- Nested Loops
- Pattern Printing

Part 3: Functions

- Defining and Calling Functions
- Parameters and Arguments
- Default and Keyword Arguments
- *args and **kwargs
- Return Values
- Scope (LEGB Rule)
- Recursion
- Lambda Functions

Part 4: Modules

- import and from-import
- Built-in Modules (math, random)
- Creating Your Own Module

Practice Programs

Number Guessing Game | **Prime and Fibonacci Series** | **Multiplication Table** | **Menu-driven ATM Program**

WHAT THIS MODULE SUPPORTS

DECIDE
Control program flow with conditions.

REPEAT
Automate repetition with loops.

REUSE
Organize code into functions and modules.

MODULE 03

Data Structures & Strings

03

Python's built-in data structures: strings, lists, tuples, sets and dictionaries, along with comprehensions and useful built-in functions.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	Beginner to Intermediate	Strings, List, Tuple, Set, Dictionary	Store, organize and process data using Python's built-in structures

Part 1: Strings

- Indexing and Slicing
- String Methods
- String Formatting
- Escape Characters
- Immutability of Strings

Part 2: Lists & Tuples

- Creating and Indexing Lists
- List Methods
- Sorting and Reversing
- Nested Lists
- Shallow vs Deep Copy
- Tuples and Unpacking

Part 3: Sets & Dictionaries

- Set Operations
- Frozen Sets
- Dictionary Methods
- Nested Dictionaries
- Iterating over Dictionaries
- Counting with Dictionaries

Part 4: Comprehensions & Built-ins

- List Comprehension
- Dictionary and Set Comprehension
- enumerate and zip
- map and filter
- sorted, min, max, sum
- any and all

Practice Programs

Word Counter **Student Marks Record** **Contact Book** **Palindrome and Anagram Checker**

WHAT THIS MODULE SUPPORTS

STORE
Choose the right structure for your data.

PROCESS
Search, sort and transform collections.

SIMPLIFY
Write concise code with comprehensions.

MODULE 04

Object-Oriented Programming

04

Object-oriented design in Python: classes, objects, encapsulation, inheritance, polymorphism, abstraction and special methods.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	Intermediate	Classes, Inheritance, Dunder Methods	Design programs using classes, objects and the four OOP pillars

Part 1: Classes & Objects

- Class and Object
- Instance, Class, Static Methods
- __init__ and self
- __str__ and __repr__
- Instance and Class Attributes

Part 2: Four Pillars of OOP

- Encapsulation
- super()
- Single and Multilevel Inheritance
- Polymorphism and Overriding
- Multiple Inheritance and MRO
- Abstraction (abc module)

Part 3: Advanced OOP

- Properties (@property)
- Composition vs Inheritance
- Magic (Dunder) Methods
- Dataclasses
- Operator Overloading

Practice Programs

Bank Account System **Student Management Classes** **Library Classes** **Shape Hierarchy**

WHAT THIS MODULE SUPPORTS

MODEL
Represent real-world entities as classes.

EXTEND
Reuse code through inheritance.

PROTECT
Control access with encapsulation.

MODULE 05

File Handling, Exceptions & Modules

05

Persisting data with files, handling errors gracefully and organizing code with packages and virtual environments.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	**Intermediate**	**open(), json, csv, os, pathlib, venv, pip**	**Read and write files, handle errors and manage packages**

Part 1: File Handling

- Reading and Writing Text Files
- Working with CSV Files
- Binary Files Basics
- Append Mode
- Working with JSON Files
- with Statement
- os and pathlib Modules

Part 2: Exception Handling

- try, except, else, finally
- Custom Exceptions
- Multiple except Blocks
- Common Built-in Errors
- raise Statement
- Debugging Techniques

Part 3: Packages & Environments

- Packages and __init__.py
- requirements.txt
- pip and PyPI
- if __name__ == "__main__"
- Virtual Environments (venv)

Practice Programs

Expense Tracker (CSV) **Notes App (JSON)** **Safe Input Validator** **Log File Reader**

WHAT THIS MODULE SUPPORTS

PERSIST
Save and load data from files.

PROTECT
Handle errors without crashing programs.

ORGANIZE
Structure projects with packages and venv.

MODULE 06

Advanced Python & Standard Library

06

Advanced language features and the most useful standard library modules, with an introduction to code quality and testing.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
1 Week	**Intermediate**	**collections, itertools, datetime, re, logging, unittest**	**Write clean, efficient, well-tested Pythonic code**

Part 1: Advanced Language Features

- Closures
- Decorators
- Iterators
- Generators (yield)
- Context Managers
- Higher-order Functions

Part 2: Standard Library

- collections (Counter, deque)
- itertools
- datetime and time
- Regular Expressions (re)
- random and math
- logging

Part 3: Code Quality

- Type Hints
- Docstrings and PEP 8
- Unit Testing (unittest)
- Introduction to Threading

Practice Programs

Function Timer Decorator **Regex Email Validator** **Unit-tested Utility Module**

WHAT THIS MODULE SUPPORTS

OPTIMIZE
Use generators and iterators efficiently.

LEVERAGE
Use the standard library instead of reinventing.

VERIFY
Test and document your code.

MODULE 07

Real-World Projects

07

Applied portfolio work designed to reinforce Python fundamentals, OOP, file handling and problem-solving skills through complete working programs.

DURATION	LEVEL	KEY TOOLS	MODULE OUTCOME
2 Weeks	Applied Practice	Python 3, OOP, File Handling, JSON, CSV, Git	Complete a portfolio of working Python projects on GitHub

01 Calculator App	02 Number Guessing Game
03 To-Do List Manager	04 Contact Book
05 Expense Tracker	06 Library Management System
07 Student Management System	08 ATM Banking System

Portfolio Practice

Build console-based projects using the concepts from the previous modules, progressing from small scripts to complete menu-driven applications with classes, file storage and error handling. Each project emphasizes planning, clean code, working features and a well-documented GitHub repository.

This curriculum is suitable for a 2-month professional training program, progressing from Python fundamentals to OOP, file handling and real-world applications.