Python Programming for Developers — Basics (48 Hours)

Course description

Python is a versatile, in-demand language used across automation, web development, data work, and more. This 48-hour, instructor-led course takes learners from first scripts through intermediate fundamentals.

Learners will practice: - Core data types and collections - Control flow (conditionals and loops) - Functions and modules - An introduction to classes - File I/O (text + JSON) - Exception handling

The course culminates in a practical command-line capstone project and a certification-style review.

Target audience

- Aspiring programmers and junior developers
- QA/automation and IT professionals who need Python for scripting
- Learners preparing for entry-level Python certification

Prerequisites

- Basic computer literacy
- Prior programming experience is helpful but not required

Duration and schedule

- **Total instructional time:** 48 hours
- **Delivery format:** Instructor-led (virtual or in-person)
- **Schedule: 12 class meetings × 4 hours each**
- **Recommended session rhythm:** recap → concept → demo → guided practice/lab → debrief (with a midpoint break)

Learning outcomes

By the end of the course, learners will be able to: - Set up a Python 3 environment and run scripts - Use core data types/operators and core collections (list/tuple/set/dict) - Implement control flow with conditionals and loops - Write organized code with functions and modules; introduce basic classes - Read/write files (text + JSON) and handle runtime/file errors with exceptions - Build and present a small command-line capstone project

---

Course modules

1. **Setting Up Python and Creating a Simple Application**
2. **Processing Simple Data Types (Strings, Numbers)**
3. **Working with Data Structures (Lists, Tuples, Sets, Dicts)**
4. **Writing Conditional Statements and Loops**
5. **Defining and Using Functions, Classes, and Modules**
6. **File and Directory Operations (Read/Write Files)**
7. **Exception Handling (Built-in and Custom)**
8. **Capstone Project & Certification Exam Prep**

Modules → Checkpoints: CP1: Modules 1–2; CP2: Module 3 (Lists); CP3: Module 3 (Tuples/Sets/Dicts); CP4: Module 4; CP5: Module 5; Capstone: Modules 6–8

12-day syllabus schedule (12 sessions × 4 hours)

> **Note:** Each session includes hands-on labs and short knowledge checks aligned to the runbook checkpoints.

Day	4-Hour Session Focus	Key deliverables
1	Orientation, tooling, running scripts; variables and expressions; console I/O	First scripts; environment verified
2	Strings + methods; formatting; numeric conversion; input validation patterns	**Checkpoint 1**
3	Comparisons and Boolean logic; string parsing (split/join); debugging habits	Practice set
4	Lists: create/update; iteration; list methods; nested lists	**Checkpoint 2**
5	Tuples + unpacking; sets; dictionaries; dict iteration patterns	Data-structure drills
6	Choosing structures; mini-project lab; review and reinforcement	**Checkpoint 3**
7	Conditionals in depth; while loops; for loops; range; loop patterns	Guided lab
8	CLI menu loops; input validation and state; mini-project lab	**Checkpoint 4**
9	Functions (params/return); scope; docstrings; modules/imports	Modular refactor lab
10	Intro OOP: classes, attributes, methods; collections of objects	**Checkpoint 5**
11	File I/O (text); JSON persistence; directories/paths; exceptions for I/O	Persistence lab

Day	4-Hour Session Focus	Key deliverables
12	Capstone build/polish; demo; final assessment + certification-style review	Capstone demo + review

Assessments and completion criteria

Checkpoints

- **Five checkpoints** distributed across the course to validate mastery of core skills:
- Fundamentals + strings
- Lists
- Data structures (tuple/set/dict)
- Control flow (if/loops/menu)
- Functions/modules + intro OOP

Capstone

CLI Personal Organizer (or equivalent) demonstrating: - A menu-driven loop (add/edit/delete/view) - Functions organized into modules - At least one class (e.g., Task/Note/Contact) - JSON persistence to a data folder - Exception handling for input and file/JSON errors

Final assessment

- Capstone presentation and rubric scoring
- Short certification-style review/quiz

Certification alignment (PCEP / PCAP)

PCEP (Certified Entry-Level Python Programmer)

PCEP focuses on fundamentals, control flow, collections, and functions/exceptions.

Course alignment: - **Fundamentals:** Days 1–2 - **Control Flow:** Days 3, 7–8 - **Collections:** Days 2–6 - **Functions & Exceptions:** Days 9–12

PCAP (Certified Associate in Python Programming)

PCAP expands into modules/packages, strings, exceptions, OOP, and I/O.

Course alignment (intro coverage): - **Strings:** Days 2–3 - **Modules:** Day 9 - **OOP (intro):** Day 10 - **I/O:** Day 11 - **Exceptions:** Days 11–12

> **Positioning:** This Basics course builds strong PCEP readiness and introduces PCAP topics; full PCAP readiness is typically achieved by pairing with the follow-on Advanced course.

Tools, platforms, and environment

- Python 3.x environment (local or remote lab)
- Code editor/IDE (e.g., VS Code) recommended
- File system access for I/O and JSON persistence labs

Scope boundaries (kept for Advanced)

To keep Basics focused, the following are explicitly out-of-scope and reserved for more advanced training: - Web frameworks/APIs (e.g., Flask/FastAPI) - Databases/SQL - GUI frameworks - Testing frameworks (pytest/coverage) - Packaging/deployment - Data science/ML toolchains

---

Suggested pacing notes (instructor)

- Reinforce habits early: naming, formatting, incremental testing, and debugging.
- Keep demos short and lab time generous.
- Use checkpoints to identify remediation needs before proceeding.
- Encourage learners to refactor into functions/modules by Day 9 to set up capstone success.