

Python Foundations for PCEP Preparation

Beginner programming and independent exam-oriented practice

FA-0659 · 5 days

Software Development

Learn to write, read and debug small Python programs, then review foundational PCEP topics using independently authored practice exercises.

Introduction

This five-day course introduces Python through small scripts and logic-building exercises. No prior coding experience is required, but learners should be comfortable with technical or numerical reasoning and ready to practise daily.

The learning path covers syntax, variables, operators, input/output, conditions, loops, collections, functions and exceptions. Practical examples are illustrative and designed to make language behaviour visible rather than promise immediate professional proficiency.

Independent PCEP-oriented review provides a way to consolidate foundational topics. The course does not award the PCEP credential, imply official affiliation or guarantee an exam pass or employment. Exam registration, eligibility, current version and testing requirements are separate and should be checked with the certification provider.

Learning outcomes

The course teaches participants to:

- Explain basic syntax, semantics and runtime behaviour.
- Write small programs using variables, types, operators and console input/output.
- Apply conditions and loops, including bounded validation/retry logic.
- Work with strings, lists, tuples and dictionaries.

- Define functions, pass arguments, reason about scope and explore simple recursion.
- Recognise syntax/runtime errors and handle suitable exceptions.
- Read and debug short code examples, identify knowledge gaps and review relevant PCEP foundational objectives using independent practice.

Prerequisites

- No prior coding experience; comfort with basic technical/numerical reasoning is helpful.
- A prepared, currently supported Python 3 environment on Windows, macOS or Linux.
- Basic computer/file skills and willingness to complete daily coding exercises.

Exam access or a voucher is not required for the programming exercises. Any certification registration and associated costs are separate from the course.

Training outline

1. Day 1 – Programming Foundations and Console Programs

Programming and Python

- Interpreting and compilation concepts, and how Python source is executed in the selected environment
- Lexical structure of Python (keywords, identifiers, whitespace sensitivity)
- The Python runtime environment and interactive mode
- Importance of indentation and code readability
- Role of comments and documentation in real-world projects

Variables, types and operators

- Variable assignment, dynamic typing and naming conventions
- Core built-in scalar types: integers, floats, strings and booleans
- Understanding type conversion and coercion
- Working with different numeral systems (binary, octal, hexadecimal)
- Arithmetic, logical, relational, and bitwise operators in practice
- Real-world use cases: calculators, simple data processing scripts

Input and output

- Capturing user input using `input()`
- Formatting output with `print()`, `sep=`, `end=`, and f-strings
- Typecasting inputs for computation
- Interactive mini-programs using illustrative quantities, such as unit conversion or a calculator

2. Day 2 — Control Flow and Logic

- Boolean logic and conditionals: `if`, `elif`, `else`
- Nested conditions and decision logic, not machine-learning decision-tree models
- Iteration using `while` and `for` loops
- Understanding the `range()` function in real tasks
- Loop control with `break/continue`, and `pass` as a no-operation placeholder
- Constructing validation and retry loops
- Read and debug short conditional and iterative programs.
- Consolidate learning through bounded validation and repeated-input exercises.

3. Day 3 — Built-In Collections

- Strings as sequences: slicing, indexing, immutability
- Lists: creation, access, modification, and built-in methods
- Tuples: fixed element references and uses in returns; contained objects may still be mutable
- Dictionaries: key-value pairs, nesting, and practical lookups
- List comprehensions and dictionary comprehension basics
- Illustrative shopping carts, fictional contact books and search filters

4. Day 4 — Functions and Scope

- Defining functions with `def` and returning values
- Parameters: positional, default, and keyword arguments
- Understanding variable scope (local, global)
- Writing reusable utility functions
- Introduction to recursion with factorials and Fibonacci sequences
- Practical modularization: splitting logic into callable functions

5. Day 5 — Exceptions and Independent Exam-Oriented Review

Errors and resilience

- The anatomy of Python errors: `SyntaxError`, `TypeError`, etc.
- Structured error handling with `try`, `except`, `else`, and `finally`
- Specific exception types: `ZeroDivisionError`, `IndexError`, `KeyError`
- Logging errors vs. printing them
- Real-world scenarios: handling user input errors, file handling `try`-blocks

PCEP topics and practice

- Review the published PCEP syllabus relevant to the intended exam version; this outline draws on PCEP-30-02 foundations
- Common language misunderstandings and code-reading pitfalls
- Independently authored exam-style practice, not actual or confidential exam questions
- Timed practice with explanation and correction of errors
- Knowledge-gap review and a checklist for checking the provider's current exam requirements