

IT Architecture and System Design Fundamentals with Python

An introductory software lifecycle and small-system project

FA-0713 · 4 days

Software Development

Introduction

Follow a small software project from requirements and system modelling to a working Python implementation and review. Learn how architecture and lifecycle decisions shape development while building foundational programming skills.

The four-day scope is introductory: use a modest data-handling application and selected iterations rather than promising a complete enterprise multi-tier system or high-performance computing expertise.

Learning outcomes

- Describe pre-development, development and post-development activities in a software lifecycle.
- Model a small system using use cases, data flows, boundaries and basic component interactions.
- Write foundational Python code with collections, control flow, functions, modules and exceptions.
- Read and write sample files with appropriate encoding and use selected external libraries.
- Explain classes and common object-oriented concepts in a small example.
- Use basic Linux navigation and authorised SSH access in a supplied lab and evaluate simple scope or performance trade-offs.

Prerequisites

- Basic computer, browser and file-management skills; no prior Python knowledge is assumed.
- A supported Python 3 environment or supplied notebook environment, with approved installation permissions if needed.
- Access to a supplied Linux lab and an authorised test account for remote exercises; unrestricted root access to a live server is not required.
- For remote delivery, a reliable connection and supported conferencing setup; ability to follow the course language.

Training outline

1. Day 1 — Software context and Python foundations

- Programming versus software engineering; algorithms, programs, language translation and the role of development tools.
- Review the software lifecycle, requirements and pre-development decisions using one small project.
- Set up a supported Python 3 release and editor or notebook, consult documentation and run a first script.
- Jupyter and hosted Colab workflows: cells, execution state, saving and export; Colab itself does not require local installation.
- Identifiers, keywords, indentation, comments, numbers, strings, formatting and core operators.

2. Day 2 — Control flow and reusable code

- Python Lists
- Python Tuples
- Python Dictionaries
- Python Sets
- Copying
- Decision Making
- The if Statement
- The if else Statement
- For Loop
- While Loop
- Break And Continue
- Defining Your Own Functions

- Parameters
- Function Documentation
- Passing Collections to a Function
- Variable Number of Arguments
- Scope

Introduce map, filter and lambda through small examples; distinguish these from ordinary loops and avoid unnecessary abstraction.

- What Are Modules
- Importing Modules
- Aliasing
- Importing Set Of Element From A Module
- Namespace
- What Are Packages
- dir function
- help function

3. Day 3 — Data, errors and objects

- String operations, regular expressions and shallow versus deeper copying of collections.
- Catch appropriate exceptions, handle specific failures and raise clear errors rather than suppressing all exceptions.
- Open sample files using appropriate modes and encoding; read and write text and introduce XML or PDF extraction with suitable libraries.
- Distinguish file acquisition from authorised remote queries; practise basic Linux file navigation and SSH access in the supplied lab.
- Classes, instances, attributes, methods, constructors, inheritance and polymorphism; explain class/static methods rather than an unsupported "static class" construct.

4. Day 4 — System modelling and an iterative project

- The SDLC Overview.
- Pre-development phases of SDLC.
- Specific phases of SDLC.
- Post-development phases of the SDLC.
- System Modeling Overview.
- Software Architecture.
- Use Cases.

- Data Structure and Flow.
- Interprocess Communication.
- System Scope and Scale.
- Methodologies, Paradigms, and Practices Overview.
- Process Methodologies.
- Development Paradigms.
- Development Practices.

Apply use cases, data structures, data flow and basic component boundaries to the small Python application; discuss interprocess communication and scaling at an introductory level.

Implement a selected iteration, inspect failure handling and simple performance measurements, and assess the result against its stated requirements.