Python Beginner

Python Beginner Training Course

This course is not for novices in coding, we assume you know the non OO fundamentals of any coding language. A critical course in Python

Prerequisites and Further Training

You should not be a complete beginner for this course. If you cannot pass this test, you must do Intro To Programming first.

Also have a look at our Python Bootcamps

Intended Audience

> Whether you are new to programming or a professional developer, this course is designed to bring you up to speed on the Python language in ways that more limited approaches cannot.

After this course you should be able to

> Start programming in Python and tackle real world solutions in Python as taught on our Advanced Python course

Further Training

> Consider doing our Advanced Python course

Course Material

> We give you an original copy of the book: Learning Python (by Mark Lutz) as we use this mainly,
> but we also give additional examples where it falls short.

Course Content

The contents could be adjusted slightly to best suit the group’s skill level and / or requirements.

DAY 1:

Getting Started

- Why Do People Use Python?
- Is Python a “Scripting Language”?
- What’s the Downside?
- Who Uses Python Today?
- What Can I Do with Python?
- How Is Python Developed and Supported?
- What Are Python’s Technical Strengths?
- How Does Python Stack Up to Language X?

How Python Runs Programs

- Introducing the Python Interpreter
- Program Execution
- Execution Model Variations

How You Run Programs

- The Interactive Prompt
- System Command Lines and Files
- Unix-Style Executable Scripts: #!
- Clicking File Icons
- Module Imports and Reloads
- Using exec to Run Module Files
- The IDLE User Interface
- Other IDEs
- Other Launch Options
- Which Option Should I Use?

Types and Operations

- Why Use Built-in Types?
- Python’s Core Data Types
- Numbers
- Strings
- Lists
- Dictionaries

- Tuples
- Files
- Other Core Types

Numeric Types

- Numeric Type Basics
- Numbers in Action
- Other Numeric Types
- Numeric Extensions

The Dynamic Typing Interlude

- The Case of the Missing Declaration Statements
- Shared References
- Dynamic Typing Is Everywhere

String Fundamentals

- String Basics
- String Literals
- Strings in Action
- String Methods
- String Formatting Expressions
- String Formatting Method Calls
- General Type Categories

DAY 2:

Lists and Dictionaries

- Lists
- Lists in Action
- Dictionaries
- Dictionaries in Action

Tuples, Files, and Everything Else

- Tuples
- Files
- Core Types Review and Summary

- Built-in Type Gotchas

Statements and Syntax

- The Python Conceptual Hierarchy Revisited
- Python’s Statements
- A Tale of Two ifs
- A Quick Example: Interactive Loops

Assignments, Expressions, and Prints

- Assignment Statements
- Expression Statements
- Print Operations

if Tests and Syntax Rules

- if Statements
- Python Syntax Revisited
- Truth Values and Boolean Tests
- The if/else Ternary Expression

Exceptions

- Why Use Exceptions?
- Exceptions: The Short Story

- The try/except/else Statement
- The try/finally Statement
- Unified try/except/finally
- The raise Statement
- The assert Statement
- Context Managers

DAY 3:

While and for Loops

- Loops
- continue, pass, and the Loop else
- Loops

- Loop Coding Techniques

Iterations and Comprehensions

- Iterations: A First Look
- List Comprehensions: A First Detailed Look
- Other Iteration Contexts
- New Iterables in Python 3
- Other Iteration Topics

The Documentation Interlude

- Python Documentation Sources
- Common Coding Gotchas

DAY 4:

Functions

- Why Use Functions?
- Coding Functions
- A First Example: Definitions and Calls
- A Second Example: Intersecting Sequences

Scopes

- Python Scope Basics
- The global Statement
- Scopes and Nested Functions
- The nonlocal Statement in 3
- Why nonlocal? State Retention Options

Modules

- Why Use Modules?
- Python Program Architecture
- How Imports Work
- Byte Code Files: __pycache__ in Python 3.2+
- The Module Search Path

- Module Creation

- Module Usage
- Module Namespaces
- Reloading Modules

DAY 5:

Classes and OOP

- Why Use Classes?

- Classes Generate Multiple Instance Objects
- Classes Are Customized by Inheritance
- Classes Can Intercept Python Operators
- The World’s Simplest Python Class

- The class Statement
- Methods
- Inheritance
- Namespaces: The Conclusion
- Documentation Strings Revisited
- Classes Versus Modules

- OOP and Inheritance: “Is-a” Relationships
- OOP and Composition: “Has-a” Relationships
- OOP and Delegation: “Wrapper” Proxy Objects
- Class Attributes
- Methods Are Objects: Bound or Unbound
- Classes Are Objects: Generic Object Factories
- Multiple Inheritance: “Mix-in” Classes
- Other Design-Related Topics
- 7 Steps to a Full Class-Based Application

Duration and pricing

Price Group B

Certificate

1. Upon completion of this course we will issue you with attendance certificate to certify your attendance and /

or completion of the prescribed minimum examples.
2. You may sit for our competency assessment test and on passing you will obtain our competency certificate.
3. Our competency assessment can be booked and taken by someone who has not attended the course at a cost of R2950.

Bookings

You can download the course registration form on our home page or by clicking here

Brochure

You may download a pdf copy of this page by clicking on the pdf icon at the top of the page.

Questions

Please email us

Schedule

On the calendar below. If your browser doesn't display the calendar below, please click on this link or try using Google Chrome, alternatively please enquire via our Contact Us page.

We are a member of the Python Software Foundation