Python Training: a practical introduction to programming

Standard Duration:

3 Days

About the course

Python is the go-to language for everything from web development and automation to data science and machine learning. This 3-day intensive workshop is designed for those with little to no prior coding experience, providing a structured, step-by-step entry into the world of programming.

We move beyond dry theory by grounding the learning in practical application. You will begin by exploring the Python ecosystem and setting up a professional development environment—including IDEs and virtual environments. From there, you will master the core logic of programming: variables, flow control, and data structures.

The course culminates in an introduction to Object-Oriented Programming (OOP) and professional practices like error handling and testing. Throughout the three days, you will apply your skills by building a cohesive "Number Guess Game" project, ensuring you leave with the confidence to write and debug your own functional Python code.

Instructor-led online and in-house face-to-face options are available - as part of a wider customised training programme, or as a standalone workshop, on-site at your offices or at one of many flexible meeting spaces in the UK and around the World.

Who should attend?

This **3-day intensive hands-on training course** is designed for individuals who are new to programming or new to the Python language and want to gain a solid foundation in Python programming fundamentals. It is ideal for:

- Individuals with no prior programming experience.
- Developers transitioning from other programming languages who want to learn Python.
- Data Analysts, Data Scientists, or Researchers who need to start using Python for data manipulation and analysis.
- Business Analysts and other professionals looking to leverage Python for automation or data tasks.

Python Training: a practical introduction to programming

- Anyone who needs a foundation in Python language and syntax in order to work with cloud-based Python APIs and web application frameworks.

Pre-requisite Skills

Participants should have basic computer literacy and be familiar with navigating a file system. Using a command-line interface (CLI) is helpful but not strictly required, as basic usage will be covered.

- Logical thinking and problem-solving skills are beneficial.
- **No prior programming experience is required.**

Standard Course Syllabus

Welcome and Introductions

- Welcome to the course.
- Introduction to the instructor and their background.
- Delegate Introductions.
- Plan of the course: Structure, goals, and approach.
- Opportunity for initial questions.

Introduction to Python and Coding Setup

- What is Python? Origins and design philosophy.
- Where is Python used? Overview of common application areas.
- Overview of common Python Libraries (Pandas, Numpy, SciPy, SkLearn, Flask) - illustrating Python's ecosystem.
- How Python Programs work: Interpreter, bytecode, execution.
- Running a Python program
- Exploring different coding environments: REPL (Read-Eval-Print Loop), IDEs (e.g., PyCharm, VS Code, Spyder), Jupyter Notebooks.
- Overview of Virtual Environments (venv) - what they are and why they are needed.
- Quick look at setting up a Virtual Environment (using pip or conda).
- Writing and running your first Python program: Python Hello World.

Lab 1: Setting up a Python Project using a virtual environment

Python Training: a practical introduction to programming

- *Practical exercise to install Python (if needed) and set up a basic project structure with a virtual environment.*

Variables and Values

- Variables, Symbol Names and Values: Understanding how data is stored and referenced.
- Variable declarations and assignment.
- Rules and conventions for Variable names.
- Variable values and how they are referenced.
- Touching on Type hints in Python (Introduction to basic type hinting syntax - **Python 3.5+**).
- Numerical Data types: Understanding numbers in Python.
- Integers.
- Floating point numbers.
- Arithmetic operations (+, -, , /, //, %, *).

Lab 2: Working with numbers

- *Practical exercise to create and manipulate numerical variables using arithmetic operations.*

Strings

- What are Strings? Representing text data.
- String operations.
- Converting numbers to strings.
- Simple String formatting.
- Formatted string literals aka f (F) strings (Python 3.6+) - the modern approach.
- Touching on Template Strings (aka t-strings) (released with Python 3.14)

Lab 3: Working with Strings

- *Practical exercise to create, manipulate, and format strings.*

Flow of Control

- Understanding how to control the order of execution in your programs.

Python Training: a practical introduction to programming

- Conditional statements: Making decisions.
- If statement.
- if-else statements.
- if-elif statements for multiple conditions.
- Boolean Logic: Using comparison operators (==, !=, <, >, <=, >=) and logical operators (and, or, not).
- Counted Loops – for loop: Iterating over sequences.
- Conditional Loops – while loops: Repeating code based on a condition.
- break and continue statements to alter loop execution.
- Overviews of:
    - Structural Pattern Matching (Python 3.10) - a brief conceptual introduction.
    - Coding formatting in Python using tooling (e.g., Black, autopep8) - overview of automated formatting.
    - Assignment expression aka walrus operator (:=) (Python 3.8) - a brief explanation.

Lab 4: Number Guess Game

- *Practical exercise to build the core logic of a number guess game using variables, user input, conditional statements, and loops.*

Sequential Data Containers

- Mutable and Immutable types: Understanding the difference and its implications.
- Tuples: Ordered, immutable sequences. Creating and accessing tuple elements.
- Lists: Ordered, mutable sequences. Creating, accessing, and modifying list elements.
- Common list methods (e.g., .append(), .insert(), .remove(), .pop(), .sort()).

Lab 5: Use a list to record guesses in Number Guess Game

- *Practical exercise to incorporate a list into the Number Guess Game to keep track of the user's guesses.*

Python Training: a practical introduction to programming

Advanced Data Containers

- Sets: Unordered collections of unique elements. Set operations (union, intersection, difference).
- Dictionaries: Unordered collections of key-value pairs. Creating, accessing, adding, and modifying dictionary elements.
- Common dictionary methods.

Lab 6: Added user high scores to number guess game using a dictionary

- *Practical exercise to use a dictionary to store and manage user high scores in the Number Guess Game.*

Functions

- Abstracting logic: Why use functions? Code reusability and organisation.
- Defining functions (def keyword).
- Calling functions.
- Parameter passing.
- Default parameter values.
- Mandatory and optional parameters.
- Named parameters.
- Returning values from functions (return statement).

Lab 7: Organising the number Guess game using functions

- *Practical exercise to refactor the Number Guess Game code by organising different parts into functions.*

Further Functions

- Local & Global Variables: Understanding variable scope.
- Modifying Global Variables within a function.
- Anonymous / Lambda Functions: Defining small, single-expression functions.
- Functions and Containers.
- filter(): Filtering elements.
- map(): Applying a function to each element.
- Nesting filter and map.

Python Training: a practical introduction to programming

Lab 8: Using filter and map with lists

- *Practical exercise to use filter and map with list data.*

Classes in Python

- Introduction to Object-Oriented Programming (OOP) concepts.
- Class Terminology: Objects, classes, attributes, methods, instances.
- Defining user defined classes.
- Instantiating Objects from Classes.
- Printing Objects.
- Being careful with Assignment.
- Class Comments.
- Defining behaviour: Adding methods to classes.
- Deleting an object.

Lab 9: Creating a player class for the number guess game

- *Practical exercise to create a Player class to represent players in the Number Guess Game and manage their attributes.*

Error Handling in Python

- Understanding Errors & Exceptions: What happens when things go wrong.
- Common built-in Exception types in Python.
- Exception Handling: Gracefully dealing with errors.
- try-except blocks.
- Using a default Exception Handler.
- The else Clause with try-except.
- The finally block for cleanup.
- Overview of:
    - Exception Groups and except* (Python 3.11) - a brief conceptual introduction.
    - add_note Method for Exceptions (Python 3.11) - a brief explanation.
- Raising an Exception / Error.
- Defining New Exceptions / Errors (creating custom exception classes).

Lab 10: Added error handling to the number guess game

Python Training: a practical introduction to programming

- *Practical exercise to add robust error handling to the Number Guess Game (e.g., handling invalid input).*

Python Modules & Packages

- Python Modules: Organising code into separate files.
- Importing Modules.
- Using from with import.
- The **main** Module: Understanding script execution.
- Importing non-local modules.
- Packages: Organising modules into directories.
- Sub Packages.
- The Module Search Path (sys.path).
- *Hands-on: Create and import custom modules.*

Lab 11: package the number guess game as a module

- *Practical exercise to refactor the Number Guess Game by packaging it as a module and making it importable.*

A brief intro to Testing using pytest and unittest

- Why Test? Understanding the importance of automated testing.
- What is pytest and unittest? Overview of two popular testing frameworks.
- Writing and Running a Test / using pytest runner.
- Basic Test Fixtures (in pytest).
- Parameterized Tests (in pytest - overview).
- Ignoring Tests.
- Testing for exceptions.

Lab 12: Added tests to the number guess game

- *Practical exercise to write basic unit tests for functions or components of the Number Guess Game using pytest or unittest.*

File IO

- Obtaining a file reference (open() function).

Python Training: a practical introduction to programming

- File access modes (read, write, append, binary).
- Reading from a file (.read(), .readline(), .readlines()).
- Writing to a file (.write(), .writelines()).
- Using with open(...) as ... for guaranteed file closing.
- Simple exception handling for File IO operations (e.g., FileNotFoundError).
- Re-naming and deleting files.

Lab 13: log program start up and results in a file

- *Practical exercise to add file I/O to the Number Guess Game, such as logging game starts or final results to a text file.*