

# PTN-102 Python programming

## COURSE DESCRIPTION

Python Fundamentals is a 4-day training course in the Python language and its many applications. The course covers the language itself, explains object-oriented as well as functional programming techniques, error handling, packaging, system and network programming, many of the Python extensions (libraries), as well as best practices. All concepts are explained through hands-on examples and exercises, so students learn by coding in Python.

**Prerequisite:** basic Linux/UNIX and programming skills.

## Delivery Method

Instructor-led training (ILT)

## Duration

Four days

## Course outline

### Chapter 1: Introduction to Python

- ^ What is Python?
- ^ What is Python 2 & 3?
- ^ Why Python?
- ^ Performance downsides
- ^ The community
- ^ Running Python interactively
- ^ Python scripts
- ^ Python help
- ^ Anatomy of a Python script
- ^ Modules
- ^ Functions and built-ins

### Chapter 2: Fundamental Variables

- ^ Python is Object Oriented
- ^ Python variables
- ^ Variable names
- ^ Type specific methods
- ^ Operators and type
- ^ Augmented assignments
- ^ Python types
- ^ Switching types
- ^ Python lists introduced
- ^ Python tuples introduced
- ^ Python dictionaries introduced

### Chapter 3: Flow Control

- ^ Python conditionals
- ^ Indentation
- ^ What is truth?
- ^ Boolean and logical operators
- ^ Chained comparisons
- ^ Sequence and collection tests
- ^ Object types
- ^ A note on Exception Handling
- ^ While loops

- △ Loop control statements
- △ For loops
- △ enumerate
- △ Counting ‘for’ loops
- △ Zipping through multiple lists
- △ Conditional expressions
- △ Unconditional flow control

#### **Chapter 4: String Handling**

- △ Python 3 strings
- △ The print function
- △ Cooking strings
- △ String concatenation
- △ “Quotes”
- △ String methods
- △ String tests
- △ String formatting
- △ Other string formatting aids
- △ Slicing a string
- △ String methods – split and join

#### **Chapter 5: Collections**

- △ Python types – reminder
- △ Useful tuple operations
- △ Python lists
- △ Tuple and list slicing
- △ Extended iterable unpacking
- △ Adding items to a list
- △ Removing items by position
- △ Removing list items by content
- △ Sorting
- △ List methods
- △ Sets
- △ Exploiting sets
- △ Set operators
- △ Python dictionaries
- △ Dictionary values
- △ Removing items from a dictionary
- △ Dictionary methods
- △ View objects

#### **Chapter 6: Regular Expressions**

- △ Python regular expressions
- △ Elementary extended RE meta-characters
- △ Regular expression objects
- △ Regular expression substitution
- △ Regular expression split
- △ Matching alternatives
- △ Anchors
- △ Class shortcuts
- △ Flags
- △ Repeat quantifiers
- △ Quantifiers

- ^ Parentheses groups
- ^ Back-references
- ^ Global matches

## **Chapter 7: Data Storage and File Handling**

- ^ New file objects
- ^ Reading files into Python
- ^ Reading tricks
- ^ Filter programs – fileinput module
- ^ Binary mode
- ^ Writing to files from Python
- ^ Standard streams
- ^ More tricks
- ^ Random access
- ^ Python pickle persistence
- ^ Pickle protocols
- ^ Build some shelves
- ^ Compression
- ^ Database interface overview
- ^ Example – SQLite from Python

## **Chapter 8: Functions**

- ^ Python functions
- ^ Function parameters
- ^ Variadic functions
- ^ Assigning default values to parameters
- ^ Named (keyword) parameters
- ^ Enforcing named parameters
- ^ Returning objects from a function
- ^ Variables in functions
- ^ Nested functions
- ^ Variables in nested functions
- ^ Function documentation
- ^ Lambda functions
- ^ Lambda as a sort key
- ^ Lambda in re.sub

## **Chapter 9: Advanced Collections**

- ^ Advanced list functions – filter
- ^ List comprehensions
- ^ Set and dictionary comprehensions
- ^ Lazy lists
- ^ Generators
- ^ Generator objects and next – coroutines
- ^ List comprehensions as generators
- ^ Copying collections – problem
- ^ Copying collections – slice solution?
- ^ Copying collections – deepcopy solution

## **Chapter 10: Modules and Packages**

- ^ What are modules and packages?
- ^ Multiple source files
- ^ How does Python find a module?

- △ Importing a module
- △ Importing names
- △ Directories as packages
- △ Writing a module
- △ Module documentation
- △ Testing a module
- △ Python debugger
- △ Python profiler
- △ Distributing libraries – distutils

## **Chapter 11: Classes and OOP**

- △ Classes and OOP
- △ Object-Oriented terminology
- △ Object-Oriented Programming
- △ Using objects
- △ Defining classes
- △ Defining methods
- △ Constructing an object
- △ Special methods
- △ Operator overload special methods
- △ Properties
- △ Properties and decorators
- △ Class methods
- △ Inheritance
- △ Inheritance terminology

## **Chapter 12: Error Handling and Exceptions**

- △ Writing to stderr
- △ Controlling warnings
- △ Exception handling
- △ Exception syntax
- △ Multiple exceptions
- △ Exception arguments
- △ The finally block
- △ Order of execution
- △ The Python exception hierarchy
- △ A common mistake
- △ The raise statement
- △ Raising our own Exceptions
- △ assert

## **Chapter 13: Multitasking**

- △ Family life
- △ Creating a process from Python
- △ Old interface examples
- △ Waiting for a child
- △ Using the subprocess module
- △ The subprocess.Popen class
- △ Running a basic process
- △ Capturing the output
- △ Passing data through a pipe
- △ Processes and threads
- △ Very basic threads in Python

- ^ Synchronization objects in threading
- ^ The trouble with threads
- ^ Using the multiprocessing module
- ^ Queue objects

## **Chapter 14: The Python Standard Library**

- ^ The Standard Library
- ^ Example – converting Python 2 scripts to Py3
- ^ Pretty Printer – a useful utility
- ^ Operating System interfaces – os and friends
- ^ System specific attributes – sys
- ^ Signal handling – signal
- ^ Converting a signal to an exception
- ^ Configuration files
- ^ The configparser module
- ^ The datetime module and friends
- ^ The platform module
- ^ External function interface – ctypes
- ^ The socket module
- ^ future
- ^ Other modules