

PTN-105 Python programming

Python is a general-purpose and popular programming language. It's concise and easy to read.

Why Python?

It's easy to learn – the time needed to learn Python is shorter than for many other languages;

It's easy to use for writing new software – it's often possible to write code faster;

It's easy to understand – it's also often easier to understand someone else's code faster;

it's easy to obtain, install and deploy.

Course Objectives: By the end of this course, learners will be able to write, debug, and structure robust Python applications using core language features, standard libraries, and best practices, from basic scripting to advanced control flow and object-oriented design. They will gain practical experience with testing, logging, concurrency, data processing, databases, GUIs, and web frameworks, enabling them to build maintainable, real-world Python applications.

Prerequisites: basic Linux/UNIX and programming skills.

Delivery Method: Instructor-led training (ILT)

Course Length: 5 days

Course Outline

Module 1. Introduction

- **Why python?**
- **Executing Python Code**
- **Implications of PVM**
- **Execution model variations**
 - PyPy, Jython, IronPython, frozen binaries
- **Running python scripts**
 - Python CLI features
- **Very basic control structures**
 - if/elif/else
 - while/else
 - try/except/finally
- **Function syntax**
 - vargs, key-value arguments
 - function defaults
 - type enforcements for functions
 - Scopes, the LEGB rule
- **Debugging python scripts**
 - pdb, ipdb
- **Modules/packages**
 - Python search path for modules

- Creating simple modules
- Creating packages

Module 2. Basic types

- **Basic types: numeric**
 - int, float, complex
 - python3 changes on number arithmetic
- **Basic types: strings**
 - String vs, Unicode (python2)
 - ByteString vs. String (python3)
 - Raw strings
 - Printing formatted output
 - Dealing with user input
- **String type**
 - String manipulation functions
 - StringIO
 - The unicodedata module

Lab 1: Simple scripts

Module 3. Sequence types

- **Sequence types**
 - Mutables vs. Inmutables
 - Working with list, array, bytearray
 - Slice operator
 - Iterable objects, iterator
- **List/sequence functions**
 - Manipulation elemens (append,extend, pop)
 - Sorting
 - The range/xrange function
- **The dictionary type**
 - Restrictions
 - Getting keys/values/items
 - Dictionary iterators
 - defaultdict, Counter
- **Using set/frozenset type**
 - Set type features
 - Set operations
- **Other sequence types/functions**
 - Basic aggregations: min, max, sum
 - The filter function
 - The map function
 - The reduce function

- The collections module
- The itertools module
- Differences: python2 vs. python3

Module 4. I/O operations

- **Basic file operations**
 - open, io.open, codecs.open
 - iterating file objects
 - fileinput module
- **Path operations**
 - os module functions
 - os.path
 - os.listdir
 - os.walk
 - glob module functions
 - glob.glob
 - glob.iglob
- **Communicate with external processes**
 - limitations of os.call, os.popen*
 - subprocess module
 - call
 - Popen
- **Parsing command line arguments**
 - getopt module
 - argparse module

Module 5. Additional control structures

- **context manager and with/as**
- **comprehensions**
 - list comprehensions
 - dictionary comprehensions
 - generator comprehensions
 - embedded comprehensions
- **creating generator functions**

Lab 2: Scripts with sequence types

Module 6. Regular expressions in Python

- **Regular expression elements**
 - character ranges
 - multipliers
 - anchors
- **re module functions**
 - match vs search

- findall vs finditer
- split
- sub
- compile
- **Use cases for flags**
 - single vs multi-line match
 - unicode character ranges
 - compile vs flags
- **Additional features in regular expressions**
 - Back-referencing
 - back-referencing in match
 - back-referencing in sub
 - Capture groups
 - referencing groups
 - named groups
 - embedded groups
 - Greediness

Lab 3: Regular expressions

Module 7. Python development tools

- **Document your code**
 - Creating docstrings
 - Using Python doc framework Sphinx
- **Testing frameworks**
 - The importance of testing in Python
 - Doctest
 - Unittest
 - unittest + mock
 - nose
- **Logging in python**
 - module logging
 - logging configuration format
- **Parallel processing**
 - threading module
 - multiprocessing module

Module 8. Advanced topics

- **Decorator**
 - simple decorators
 - decorators with arguments
 - python built-in function/class decorators
- **Performance tips**

Module 9. Object oriented programming in Python

- **OOP basics**
 - Compare OO features of Python and Java/C++
 - python2 vs python3 classes
 - new-style vs old-style classes
 - `__slots__`
- **Member functions**
 - Constructors
 - Destructors
 - Writing member functions
- **Member attributes**
 - General implementation of instance attributes
 - Class static attributes
 - Getter/setter methods
 - Access control solutions
 - the property class
 - the descriptor model
- **Operators**
 - operator related functions
 - `__str__` vs `__repr__`
 - `__call__`
- **Class/function decorators**
 - `@staticmethod`, `@classmethod`
 - `@abstractmethod`, `@abstractproperty`
 - `@total_ordering`
- **Python class template**
 - inheritance
 - abstract template classes
 - metaclass

Lab 4: Python OO

Module 10. Python Database API

- **Connecting to DB API compliant Relational Database Management Systems**
 - Creating and populating tables
 - Retrieving data records
 - Executing parametrized queries
 - Exporting and importing table data with Postgresql

Module 11. Creating graphical user interfaces using Python/Tkinter

- **Tkinter widgets and their standard attributes**
 - Dimensions
 - Colors
 - Fonts

- Anchors
- Relief styles
- Bitmaps
- Cursors
- **Organizing widgets in the parent widget area**
 - Using pack
 - Using grid
 - Using place

Module 12. Intorduction to the Python Django WEB application framework

- **Architecture**
- **Creating a Hello world project**
- **Managing settings**
 - URL patterns

Lab 5: Using frameworks