Programming in Python

Recommended duration: 3 days, 8 hours a day [24 hours]

Pre-requisites:

\* Participants should be comfortable with the following technologies:

Basic computing concepts

Knowledge of Windows or Linux

Training mode:

\* The Training program will be demonstration oriented.

\* Most of the concepts will be taught by demonstrating Code and the participants can practice the same.

Lab requirements:

\* As the training is highly lab oriented, each participant

Attending the training program must be provided with a

Computer with the following software installed:

- Windows or Linux with Python 3.6.2 installed

- Pycharm Community Edition

--------------------------------------------------------------

Proposed Course Agenda

--------------------------------------------------------------

[Day 1]

Note : Python Recap could be covered upon the Participants request

Python Basics a Recap

\* Introduction to python programming

\* An overview of scripting and the pragmatic development

Approach.

\* An overview on python interpreter

\* An overview on Python development tools

\* Numbers and expressions

\* Variables and statements

\* Conditional statements and loop

\* Handling user input

\* An overview of built-in functions and modules

\* Python syntax, style and coding conventions

\* Basic introspection using type() and dir() function

\* Types, Classes and dynamic typing

Working with Strings

\* An overview of strings in python

\* String operators

Lists, Tuples and Sets

\* Common sequence operations

\* Manipulation of Lists

\* Manipulation of Tuples

\* Manipulation of Sets

Working with dictionaries

\* Introduction to dictionaries

\* Creating, assigning, updating dictionaries

\* Dictionary operators, functions and built-in methods

Functions

\* Creating user-defined functions

\* Passing functions

\* Formal arguments

\* Variable-length arguments

Hands On

\* Learn and understand the tools provided with ActivePython.

\* Practice basic programming concepts using python.

\* Small exercises on understanding conditional constructs

and loops.

\* Practice various string operators, functions

and built-in methods

\* Practice exercises on Lists and Tuples

\* Practice exercises on dictionaries, functions and modules

Advanced Python constructs

\* List comprehension

\* List manipulation using map, filter and reduce functions

\* Tuplesets, zip function and parallel iteration

\* Generators, Iterators and Iterable objects

\* Range and XRange objects

\* Itertools and Collections framework

Modular development

\* Creating modules

\* Variable scope

\* Understanding namespaces

\* Importing modules and module attributes

\* Module hierarchy

Standard Python modules

\* Using the sys module

\* sys.argv, sys.path, sys.version

\* An overview on \_\_builtin\_\_ and \_\_future\_\_ modules

\* Using the os module

\* Filesystem/directory functions

\* Basic process management functions

\* Recursive directory iteration using os.walk

\* Using the os.path module

\* Determining basename, dirname, path manipulation

\* File type/size/timestamp and other stat determination

\* Using the time and datetime modules

Hands On

\* Parsing command line arguments using sys.argv

\* Command-line switches parsing using optparse module

\* Listing directory contents using os.listdir

\* File test operations using os.path module

\* Finding files using os.walk

\* File operations using shutil module

[Day 2]

Classes and Objects

\* Introduction to OOP using python

\* Classes and class attributes

\* Instances and instance attributes

\* Binding and method invocation

\* Composition, Subclassing and Derivation

\* Inheritance

\* Built-in functions for classes, instances and other objects

\* Privacy and Delegation

\* An overview of built-in python classes and modules

Errors and exception handling

\* Introduction to exceptions

\* Detecting and handling exceptions

\* Exceptions as Strings and Classes

\* Raising exceptions

\* Creating exceptions

\* Standard exceptions

Regular expressions

\* Introduction to regexps

\* Special symbols and characters for RE

\* Metacharacters and Metasymbols

\* Character classes

\* Quantifiers and Positions

\* Searching/Matching a string/pattern

\* Search and replace operations

\* Splitting and joining strings

\* Practical examples

File and Directory handling

\* File I/O operations

\* Built-in file and directory handling libraries

\* fileinput

\* stat

Hands On

\* Basic design patterns

- Singleton, Factory

\* Practice error handling and exceptions

\* Regular Expression examples

\* File and directory handling examples

\* Practice basic file operations and text

processing

\* Compressing and Uncompressing files

[Day 3]

Multithreaded programming

\* Introduction to threads and processes

\* Creating and managing threads

\* An overview of thread and threading modules

\* Concurrency management using Lock and RLock

\* Producer Consumer algorithm using Queue

Network automation modules

\* FTP automation using ftplib

\* Automating HTTP using requests

Hands On

\* Creating a simple multi-threading fileserver

that can serve files to client using a

thread-pool.

\* A simple web crawler program that can crawl a

website and report broken links on the

same by email.

Simple examples on using ftplib and telnetlib

Process management and process automation

\* Using os.system, os.popen, os.fork, os.exec

functions

\* Using the commands module

\* Using the subprocess module

\* Managing processes using various functions

in os module

\* Automating processes by capturing stdout,

stdin and stderr of the child process

\* Logging using syslog module

Logging

\* Python Logging module

\* Logging Level

\* Steam Logging

\* File Handling Logging

\* Log RollOver

Debugging

\* Using pdb debugger & IDE

Hands On

\* Automate simple interactive programs like

the UNIX bc command

\* Creating custom module packages using

distutils/pypi