Core Python Programming

**Recommended duration: 3 days**

**Course Description:**

This training program provides a basic introduction to Python programming language. The focus of this training program will be more on "pythonic" approach towards problem-solving.

**Target Audience (who should attend):**

\* Engineers who wish to learn python program to develop applications or to automate their applications/framework.

\* Engineers who wish to prototype new applications.

**Pre-requisites:**

\* Participants should be confortable with the following technologies:

- Basic programming background with good understanding of programming language ingredients that include variables and datatypes, flow control statements, and function/procedural programming paradigms.

- Knowledge of any scripting language would be beneficial.

- Knowledge of OOP and modular programming concepts in any programming language are recommended.

**Course Objectives:**

\* Understand the basic programming paradigms using Python 2 and Python 3

\* Understand builtin datatypes, variables, functions and flow control statements.

\* Learn how to use string, tuple, list, bytearray, set, dictionary types effectively.

\* Learn "pythonic" idioms and anti-idioms.

\* Learn functions, modules and file I/O operations.

\* Learn the basic concepts of OOP in Python.

\* Multi-threading and Multi-processing in Python.

**What will NOT be covered:**

\* Process and network automation related library modules/concepts.

\* Web programming or web automation.

\* Any third-party library/modules.

**Training mode:**

\* The Training program will be mostly demonstration oriented.

\* Most concepts will be taught by demonstrating code and participants are expected to learn by practicing 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/Linux/Mac OSX with Python 3.3+ installed.

- A good programmer's editor (Notepad++ or PSPad on windows, vim/emacs on Linux) installed.

\* LCD Projector with support to connect to trainer's laptop (running on Mac OSX).

\* Whiteboard, markers, scribble pad + pen for participants.

**Course Agenda**

**Day 1**

**Introduction to Python**

\* Introduction to python programming

\* An overview of scripting and the pragmatic development approach.

\* An overview on python interpreter

\* An overview on Python development tools

- Aptana Studio/PyDev, NetBeans, bpython, vim-python and IDLE

\* Python implementations (features and differences):

- CPython, Jython, IronPython, PyPy, Stackless Python

\* Python 2 vs Python 3 (feature differences)

**Getting started with Python**

\* 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, Duck typing

**Working with Strings**

\* An overview of strings in python

\* String operators

\* Built-in string manipulation functions

\* Built-in string methods

\* Special string features in python

\* Built-in modules for string handling

\* Unicode strings and bytearray

**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

**Hands On**

\* 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

**Day 2**

**Functions**

\* Creating user-defined functions

\* Passing functions

\* Formal arguments

\* Variable-length arguments

\* A walk-through on various built-in functions

**Modular development**

\* Creating modules

\* Variable scope

\* Understanding namespaces

\* Importing modules and module attributes

\* Module hierarchy

**File and Directory handling**

\* File I/O operations

\* Built-in file and directory handling libraries

\* fileinput

\* stat

\* filecmp and dircmp

\* glob, zipfile and tarfile

\* pickle and shelve modules

\* Serialization using json

**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

\* Using random, shutil, pprint, hashlib, md5, optparse

and logging modules

**Hands On**

\* Practice exercises on dictionaries, functions and modules

\* Practice exercises on file operations

\* Practice exercises on logging, os related features

**Day 3**

**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

\* 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

**Implementing multi-tasking in Python programs**

\* Introduction to threads and processes

\* Creating and managing threads and processes

\* threading vs multiprocessing module

\* Concurrency management using Lock and RLock

\* Producer Consumer algorithm using Queue

\* Implementing thread pool and process pool

**Testing, Debugging and Deployment**

\* Using the unittest module for writing

testcases

\* Using doctest

\* Regressing testing using test module

\* Documentation generation using pydoc

\* Using pdb debugger framework and inspect

\* Tracing python statement execution using

trace framework

\* Timing measurement using timeit module

\* Building and installing python modules

using distutils and pypi

**Hands On**

\* Create custom python library module

\* Practice exercises on OOP related features in python

\* Practice exercises on multi-threading and multi-processing