Python Programming

Duration: *5 days*

Overview:

This course is a comprehensive training program designed to introduce learners to the world of Python programming. Covering a breadth of topics from the very basics of running Python and writing simple scripts to more advanced concepts like handling exceptions, working with dates and times, and running Python scripts from the command line, this course is structured to provide a strong foundation in Python.

Pre-requisite:

There is no pre-requisite for this course

Course Syllabus:

Python Introduction and Basics

- The Python Interpreter
- Working with Command Line/IDLE
- Python Data Types
- Built in Operators, Functions and Methods
- The 'type()' and 'dir()' functions
- Blocks and Indentation
- Scope of Variables

Conditional Statements and Iterators

- Simple 'if' and Simple 'if … else' statements
- Multilevel 'if … elif … else' statements
- Nested 'if … else' statements
- The 'for' Iterator
- The 'while' loops
- Loop Control Statements – break and continue
- The 'pass' statement

Collections

- Lists – Definition, Operations and Comprehension
- Tuples – Definition and Operations
- Dictionary – Definition, Operations and Comprehension
- Sets – Definitions and Operations
- Membership and Identity Operators
- Enumerations
- Generators, Decorators and Closures

Functions

- Defining the Function
- Function without Arguments
- Function with Fixed number of arguments
- Function with keyword arguments
- Function with Default arguments
- Function with variable length arguments
- Returning single/multiple values from the function
- Lambda Expressions
- Map, Filter and Reduce Functions
- Globals and Locals

File I/O

- Creating the File
- Writing the contents to the file
- Using 'writelines()' method
- Explicit reading with read(), readline() and readlines()
- Reading Binary Files
- Use of 'with' statement

Modules and Packages

- What are modules?
- Import user defined modules
- Priority in reading the modules from the directories
- Creating '.pyc' file using '-m' option
- Advantage of '.pyc' file
- Controlling imports with __all__
- Pre-installed Modules
- Installing new Modules
- Python Repository
- Pip and Easy_install
- Creating Packages
- Accessing the Packages

Regular Expressions

- The 're' module
- Patterns
- Anchors
- Range of Characters
- Quantifiers

- Metacharacters
- Character Class Escape Range Sequences
- Alternatives
- Choices
- Groupings
- Greedy Matches
- Methods – search, match, findall, sub, split and compile

Standard Python Modules

- The 'sys' Module – Command Line Arguments
- OS Module – Filesystem Functions, os.path and os.walk
- 'time' and 'datetime' modules
- 'math' module
- 'subprocess' module

Classes and Objects

- Defining the Class
- The '__init__' and '__str__' methods
- Creating the Object
- The 'self' parameter
- Private and Public Attributes
- Concept of Encapsulation
- Concept of Polymorphism
- Concept of Inheritance
- Types of Inheritance – Single, Multiple, Multilevel and Hierarchial
- Magic Methods
- Method and Operator Overloading

Context Managers

- The with keyword
- Context manager as an object
- __enter__ and __exit__ magic functions
- Defining context manager classes
- Instantiating and using context managers
- Common mistakes to avoid

Closures

- Local functions
- Returning functions from functions
- Closure and nested scopes
- Function factories

- Non-Local Keyword

Exception Handling

- Standard Exception Hierarchy
- Handling the Exceptions
- Generic Exceptions
- Handling Multiple Exceptions
- User Defined Exceptions
- Raise and Assertions

Multithreaded Programming

- Why use threads?
- Threads are different
- Variables are shared
- Python threads modules
- The threading module
- The queue module
- The python thread manager
- Debugging threaded programs
- About GIL (Global Interpreter Lock)

Relational Database Interaction

- Use of Database in Real Time
- Modules to access the databases in Python
- Installation of DB Module, preferably for MySQL
- Connecting to the database
- CRUD Operations
- Transaction Management
- Connection and Cursor Objects
- Use of Query String Parameters

ORM in python using SQL Alchemy

- Overview of SQL Alchemy
- Constructing class to relate tables in databases
- Creating tables in databases
- Queriny tables
- Associations – One to many, many to many
- Automating table creation using alembic

Exercises

- Exercises to try the above concepts
- CSV Modules
- CSV Format
- Creating the CSV Files
- Writing the contents into the CSV Files
- Reading the CSV File
- Reading the Contents of CSV File using DictReader
- Writing the Contents of CSV File using DictWriter
- Other Delimiters instead of ‘,’

JSON Modules

- JSON Format
- Field data and Formats
- Converting Python Objects to JSON Format
- Converting JSON Format to Python Objects

Exercises

- Exercises to try the above concepts

XML Processing

- XML Format
- XML Tags and Attributes
- Recursive Tags
- Searching within XML
- Modifying and writing the modified contents to another XML
- Removing the elements from XML based on certain condition
- Creating the XML Tree
- Adding the Elements to the XML Tree
- Preserving XML Structure

Logging

- The ‘logging’ Module
- Creating a Logger
- Log Handler
- Log Formatter
- Logging Levels – DEBUG, INFO, WARN, ERROR and CRITICAL
- Filtering the Logging contents
- Multiprocessing Logs and Synchronization

Unit Testing

- PyUnit
- unittest module
- test fixture
- TestCase class
- The setUP() and tearDown() methods
- TestSuite class
- test runner and the run() method
- TestResult
- Different Types of Assertions

Overview of Pandas and Numpy

- Overview of 'NumPy' Module
- Array Manipulation with 'NumPy' Module
- Integer Array Indexing and Boolean Array Indexing
- Reindexing
- Matrix Manipulation

Programming with Pandas

- Introduction to Pandas
- Pandas – Series, DataFrame
- Accessing Elements from Series and DataFrame
- Reindexing
- Reading and Writing the CSV Files
- Reading the data from the Spreadsheet
- Generating the Spreadsheet