Python programming

Students Will Learn:

- Python Language Syntax
- Flow Control Constructs
- Python Program Deployment
- Organizing Code Using Functions
- Programming for User Interaction

- Using Regular Expressions
- Exception Handling
- GUI Programming Concepts
- Object Oriented Framework

Course Description: This hands on Python programming course shows how to rapidly develop and maintain effective Python programs. The course includes thorough coverage of Python 3 syntax, built in data types and control constructs. The course takes a practical approach to creating and organizing Python programs using functions, packages, modules and classes as part of Python's object-oriented paradigm. Attendees will learn to identify and correct problems through unit testing and exception handling. Attendees will use regular expressions to rapidly process data captured from users and from the file system. Students will learn how to create GUI based Python programs to gracefully interact with the user. Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Prerequisites: Prior scripting experience or knowledge of fundamental programming concepts.

Python Programming Course Overview:

Introduction to Python 3

- Origin and Goals of Python
- Overview of Python Features
- Getting and Installing Python
- Accessing Python Documentation: Python Enhancement Proposals (PEP)
- Python's Strengths
- Using Python with Other Programming Languages

Using Python

- Executing Python Programs from the Command Line
- Creating and Executing Python Programs Using IDLE
- Python Command Line Options
- Environment Variables that Influence Python
- Creating Python GUI Applications
    - Standalone vs. Web-Enabled

Interfaces
- The Python Standard Library

Language Fundamentals

- Python's Lexical Analyzer
- Using Whitespace to Structure Programs
- Identifiers and Keywords
- Python's Execution Model
  - Naming Objects and Binding
- Python's Data Model
  - Immutable and Mutable Objects
  - Values
  - Types
- Creating and Using Variables

Flow Control Constructs

- if/elif/else Statements
- Creating Loops with while and for
- Understanding Iterators
- Returning Values with return Statements
- Loop Modification with break and continue
- Returning Generator Iterators with the yield Statement
- Retrieving Iterators with next()

Expressions

- Unary and Binary Arithmetic Operations
- Comparison and Boolean Operations
- Conditional Expressions
- Lambda Expressions
- Order of Operations and Operator Evaluation
- Expression Lists
- Assignment Operations

Exception Handling

- Types of Python Exceptions
- Handling Exceptions with try/except/finally
- Triggering Exceptions with raise
- Defining New Exception Types
- Implementing Exception Handling in Functions, Methods and Classes
- Working with the Regular Expression Error Exception

Unit Testing and Debugging

- Using docstrings to Document Code
- Instrumenting Code with print Statements
- Testing Programs with PyUnit
- Creating Tests with the TestCase Object
- Organizing Tests with the TestSuite Object
- Working with pdb (Python Debugger)

Using the String Object

- Using ASCII and Unicode Strings
- Manipulating Strings with String Methods
- Using the format() Function to Format Strings
- Using Escape Sequences
- Working with Raw Strings

Arrays, Collections and Dictionaries

- Sequenced Data Structures
  - Arrays
  - Collections
  - Dictionaries
- Creating and Accessing Lists

Organizing Code

- Defining Functions
- Calling Functions
- Creating Anonymous Functions
- Altering Function Functionality with Decorator Functions

- Manipulating Lists
- Creating and Accessing Tuples
- Understanding the Differences Between Lists and Tuples
- Using Dictionaries to Create Data Records
- Manipulating Dictionaries Using Dictionary Methods
- Creating Sets
- Performing Set Operations
    - Union
    - Intersect
    - Difference
- Differences Between Sets and Dictionaries
- Using Generators to Return Iterators

Working with Arguments

- Passing Arguments to Functions by Reference and by Value
- Defining Functions with Required Arguments
- Defining Functions with Default Arguments
- Defining Flexible Functions that Take Variable Length Arguments

Object Oriented Programming Concepts

- The Object Oriented Programming Paradigm
- Encapsulating Information
- Classes vs. Instances of Objects
- Built-in Class Attributes
- Implementing Class Inheritance
- Using Objects in Code
- Creating Classes with the class Statement
- Creating Objects as Class Instances
- Using Preexisting Classes as the Basis of a New Class
- Using Modules to Group Related Functions, Classes and Variables
- Locating and Importing Modules
- Using Packages to Group Modules Together

GUI Programming

- GUI Programming Concepts
- GUI Programming Frameworks
- Installing the Tkinter Framework
- Using Tkinter to Create GUIs
- Using Tkinter Widgets

Working with XML Data

- Discussing Python XML Processing Modules
- Importing the ElementTree API
- Parsing XML Documents
- Searching XML Documents for Specific Nodes
- Creating XML Documents
- Using XML to Exchange Data Between Python Programs

I/O Handling

- Sending Output to STDOUT Using the print() Method
- Reading Input with the input() Method

- Creating File Objects with the open() Method
- Controlling File Access Modes
- Working with File Object Attributes
- Closing File Objects with the close() Method
- Reading and Writing to File Objects with read() and write()
- Using File Processing Functions from the OS Module

Regular Expressions

- Regular Expression Syntax
- Using Regular Expressions in Python
- Altering Regular Expression Processing with Regular Expression Modifiers
- Using Regular Expression Operators
- Scanning Through Strings Using the search() and match() Methods
- Creating Reusable Patterns by Using the compile() Method