ProTech protechtraining.com *"Charting the Course ... ... to Your Success!"*

Python Programming

Course Summary

Description

This course provides a complete introduction to the Python scripting language. Delegates will learn to program using all the features of Python 3, with an emphasis on writing maintainable and testable scripts. Both the functional and object-oriented aspects of Python are covered in depth, including best practices for using them in combination.

The course is normally taught over 3 days, but this can be reduced if delegates are fluent in another scripting language, such as Perl or Ruby. The delivery can be on either Linux or Windows as required. By default the 'PyCharm' IDE is used for examples and exercises, but other editors can be substituted on request.

Topics

- Introduction to Python
- The Basics of Python Scripting
- Working with Structured Data
- Regular Expressions in Python
- Object Orientation in Depth
- Functional Programming in Python
- Writing Maintainable Scripts
- Performing Common Tasks

Prerequisites

Students should ideally have several years programming experience, gained either from strongly typed languages like C, C++, C# and Java or other scripting languages like Perl, Ruby and JavaScript.

Duration

Three days

Python Programming

Course Outline

I. Introduction to Python
- A. The evolution of Python from Shell Scripting and Perl
- B. Installing Python and running / debugging simple scripts
- C. Differences between Python 3 and earlier versions
- D. Using the JVM and CLR based Python interpreters
- E. Comparing Python to Ruby, JavaScript and PowerShell

II. The Basics of Python Scripting
- A. Declaring and modifying variables
- B. Reading and writing from the console
- C. Working with the built in data types
- D. Making choices and performing iteration
- E. Declaring functions and passing parameters
- F. Identity, equality and references

III. Working with Structured Data
- A. Storing individual items in sets, lists and tuples
- B. Storing pairs of items in dictionaries (aka hashes)
- C. Understanding how class declarations work in Python
- D. Creating classes and adding attributes and operations
- E. Using your own types with the built-in collections

IV. Regular Expressions in Python
- A. What is a regular expression?
- B. How the Regular Expression Engine operates
- C. Functions in Python that accept regular expressions
- D. Creating character classes and specifying multiplicities
- E. Meta-characters for specifying positions in the input
- F. Understanding greedy vs. non-greedy matching
- G. Using parenthesis for grouping and sub matches
- H. Making use of look-behind and look-ahead
- I. Adding comments to your regular expressions

V. Object Orientation in Depth
- A. Creating class hierarchies in Python
- B. Overriding methods from base classes
- C. Adding properties to your classes
- D. The intrinsic class and object attributes
- E. Creating your own iterator objects and generators
- F. Using decorators to address Aspects in your design
- G. Guidelines for overloading operators sensibly
- H. Support for meta-programming in Python

VI. Functional Programming in Python
- A. Using functions as inputs to other functions
- B. Building and returning functions from functions
- C. Creating and working with lambda functions
- D. Using lamdbas with filter, map and reduce
- E. Simplifying your code using list comprehensions

VII. Writing Maintainable Scripts
- A. Raising and catching exceptions correctly
- B. Organizing your code into modules and packages
- C. Understanding symbol tables and accessing them in code
- D. Picking a testing framework and applying TDD in Python

VIII. Performing Common Tasks
- A. Reading and writing text, data and objects from files
- B. Basic network programming in Python using sockets
- C. Dynamically running code with exec and eval
- D. Working with SOAPy and RESTful web services
- E. Options for accessing relational databases
- F. Writing multi-threaded Python scripts