





**Python Language and Introduction to OpenPyXL**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
| 1.0 | 22-01-2024 | Ramakant |  |  |  |
| 2.0 | 15-02-2024 | Ramakant |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Document History**

# 

**Course Title: Python Language and Introduction to OpenPyXL**

# Course Summary:

This Python Language training program provides a comprehensive exploration of fundamental concepts and advanced features.

Starting with the basics, participants will delve into data types, control structures, and collections. The course progresses to cover functions, generators, iterators, regular expressions, object-oriented programming, and exception handling. Additionally, it delves into the creation and publishing of modules, alongside the nuances involved in it.

The training concludes with an introduction to OpenPyXL for working with Excel files.

1. **Pre-Requisite**

No prior experience with Python is required, but a familiarity with any programming language is beneficial.

1. **Audience**

Beginners in Python

1. **Hardware & Network Requirements**
   1. Local admin policy on candidate machine should allow installation of required modules using ‘pip’ installer.
   2. Access to
      1. <https://github.com/>
      2. <https://www.python.org/>
      3. <https://docs.python.org>
      4. [https://peps.python.org](https://peps.python.org/pep-0000/)
      5. <https://pypi.org/>
      6. <https://www.python.org/downloads/>
      7. <https://scipy.org>
      8. <https://docs.scipy.org>
2. **Software Requirements**
   1. OS: Windows 10/11 Pro or Enterprise
   2. IDE: Visual Studio Code (<https://code.visualstudio.com/>)
   3. Git
      1. [Git for windows](https://github.com/git-for-windows/git/releases/download/v2.43.0.windows.1/Git-2.43.0-64-bit.exe) (for Windows platform)
      2. git package for Linux/MacOs
   4. Installed software/modules:
      1. Python Python 3.11.3 (<https://www.python.org/ftp/python/3.11.3/python-3.11.3-amd64.exe>)
      2. Pip
         1. curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
         2. python get-pip.py # Install pip
         3. pip --version # Verify if pip installed
      3. Python3 Modules (On command line, where python is installed)
         1. py -m pip install --upgrade pip
         2. pip install virtualenv numpy pandas scipy urllib3 multipledispatch requests
      4. Should be able to run below program (SetupTest.py) from participant login

# SetupTest.py

import platform

import sys

import virtualenv

import numpy

import pandas

import scipy

import urllib3

import multipledispatch

import requests

print("Python version:", platform.python\_version())

print("virtualenv version:", virtualenv.\_\_version\_\_)

print("numpy version:", numpy.\_\_version\_\_)

print("pandas version:", pandas.\_\_version\_\_)

print("scipy version:", scipy.\_\_version\_\_)

print("urllib3 version:", urllib3.\_\_version\_\_)

print("multipledispatch version:", multipledispatch.\_\_version\_\_)

print("requests version:", requests.\_\_version\_\_)

* + 1. ***Program should run without error and list out the versions of the modules correctly.***

# Learning Outcomes:

By the end of this program, participants will:

* Have a strong understanding of Python fundamentals.
* Be proficient in using collections and control structures.
* Master functions, generators, iterators, and object-oriented programming.
* Handle exceptions and errors effectively in Python programs.
* Apply decorators to enhance the functionality of functions and classes.
* Create, publish, and use modules in Python.
* Gain an introduction to working with Excel files using OpenPyXL.

# Course Content (day wise):

|  |  |
| --- | --- |
| Course Schedule | |
| Day 1 | **Foundations of Python**  * Python as a programming language   + Low level vs. High level Languages   + Compiled vs. Interpreted Languages   + Programming paradigms   + Best practices / Idioms - Zen of Python   + Enhancements and Documentations * Programming environment   + Installations and configurations   + Using iPython (REPL) and its nuances   + Running a basic Python program   + Running Executable Python scripts * Basic Data Types   + Introduction to Bool and None   + Numbers: Integer, Floats, Decimals, Fractions, Complex * Control Structures   + Conditional Statements: if-elif-else, match-cases   + Loops: for-else, while-else |
| Day 2 | **Collections and Comprehensions in Pythons**  * Collections   + Object Identity     - Id, type, value   + Mutability     - Distinction in Python     - Handling, implications and parameters of choice   + Mutables:     - Lists       * Member methods and Usage       * Performance behaviours     - Sets       * Hashing       * Member methods and Usage       * Immutability of Elements       * Applications     - Dictionaries       * Comparison with sets       * Member methods and Usage       * Applications and implications of key immutability       * Collection as key in a dictionary   + Immutables:     - String type       * Member methods and Usage       * String slicing     - Bytes and ByteArrays       * Mutability characteristics       * Conversion from and to strings       * Techniques to perform conversions     - Tuples       * Comparison against lists and sets.       * Member methods and Usage       * Tuples as keys     - Named Tuples       * Extension of Tuples       * Member methods and Usage       * Compare against dictionary       * Named Tuples as keys     - Frozen Sets       * Taking sets to the next level       * Member methods and Usage       * Frozen sets as keys * Comprehensions for:   + List   + Set   + Dictionary   + Generator class   + Tuple |
| Day 3 | **Advanced Concepts and Functions**  * Virtual Environments   + Creating and Managing   + Activation/Deactivation   + Module Isolation * Functions   + Syntax and Docstrings   + Parameter Handling and Scopes     - Positional arguments     - Named arguments     - Packing and Unpacking of elements     - Variable arguments     - Keyworded variable arguments   + Special Arguments     - Enforcing argument structure via syntax * Advanced Functions   + Method Overloading     - Mechanism in other languages     - Support in Python   + Special Constructs:     - Generators     - Iterators     - Lambda Functions   + Utility Methods:     - Map     - Filter     - Reduce * Modules   + Creating Modules and Packages   + Import Syntax and Variations   + Interface Publishing |
| Day 4 | **Regular Expressions**  * Introduction to Regular Expressions:   + Definition and purpose of regular expressions.   + Applications in text processing and pattern matching. * Working with re Module:   + Importing the re module.   + Functions   + Objects   + Exceptions * Basic Patterns:   + Literal characters and metacharacters.   + Character classes and ranges.   + Special characters like dot (.) and caret (^). * Quantifiers and Repetition:   + Asterisk (\*), plus (+), and question mark (?) operators.   + Using curly braces ({}) for specifying repetition. * Anchors and Boundaries:   + Caret (^) and dollar sign ($) as anchors.   + Word boundaries (\b) and non-word boundaries (\B). * Character Classes:   + Predefined character classes (\d, \w, \s).   + Negation and custom character classes. * Grouping and Capturing:   + Parentheses for grouping.   + Capturing groups and non-capturing groups.   + Backreferences. * Assertions:   + Positive lookahead (?=) and negative lookahead (?!).   + Positive lookbehind (?<=) and negative lookbehind (?<!). * Modifiers and Flags:   + Case-insensitive matching.   + Multiline matching.   + Dot-all mode. * Advanced Patterns:   + Greedy vs. non-greedy quantifiers.   + Using alternation (|) for OR conditions.   + Escape sequences. |
| Day 5 | **Duck Typing, Object-Oriented Programming and Exception Handling**  * Duck Typing   + Ideology and its interpretation   + Impact on Python as a language.   + Implications on the design for Python programmers * Classes & OOP   + Object Oriented Paradigm     - Major Pillars of Object-oriented paradigm     - Minor Pillars     - Translating concepts to implementations   + Basic constructs     - Constructor and Initializer     - Variables, and Methods     - Instance members vs. Class members     - Static methods     - Inheritance (Single)     - Design considerations when using Inheritance. * Exception Handling   + Syntax   + Built-in exceptions and handling * Excel Interfacing   + Introduction to OpenPyXL   + Demonstration and basic exercises   Closing discussion |

# Course Structure:

|  |  |
| --- | --- |
| **Activity** | **Indicative Number of Hours** |
| Pre-Read Hours | N/A |
| Teaching Hours | 15 hrs. |
| Hands on Sessions Hours | 10 hrs. |
| Assignments & Tutorial Hours | 10 hrs. |
| Mock Project Hours | N/A |

# Course Structure:

|  |  |  |
| --- | --- | --- |
| **Method of Assessment** | **Yes/No** | **Weightage** |
|  |  |  |
| Pre-Assessment | Yes | 0% |
| Mid-Assessment | No | - |
| Post-Assessment | Yes | 100% |
| Project Work | No | - |

# Course Resources:

* 1. **Code Samples: Sample code snippets and solutions for better understanding.**
  2. **Assignments: Practical assignments to reinforce learning and build real-world skills.**

# Recommended Reading Links:

1. **Python Documentation:** [Python Official Documentation](https://docs.python.org/3/)
2. **Real Python Tutorials:** [Real Python](https://realpython.com/)
3. **Corey Schafer's Python Tutorials:** [Corey Schafer - Python Tutorials](https://youtube.com/playlist?list=PL-osiE80TeTt2d9bfVyTiXJA-UTHn6WwU&si=FdETWYx8WD-9teaQ)
4. **Effective Python - Brett Slatkin:** [Effective Python](https://effectivepython.com/)
5. **OpenPyXL Documentation:** [OpenPyXL Documentation](https://openpyxl.readthedocs.io/en/stable/)

# Course Owner (s):

|  |  |  |
| --- | --- | --- |
| **Employee Name** | **Employee Mail ID** | **Business Unit** |
|  |  |  |
|  |  |  |
|  |  |  |