**PY101-*Python Essentials***
Duration: 5 Days; Instructor-led

**WHAT YOU WILL LEARN**

Python is a popular, general-purpose, multi-paradigm, open-source, scripting language. It is designed to emphasize code readability – has a clean syntax with high level data types. It is suited for interactive work and quick prototyping, while being powerful enough to write large applications in.

Python has a large number of available and well-written modules for everything from abstract syntax trees to ZIP file manipulation. Its ecosystem features an extensive set of tools and fancy.

This course introduces the student to the Python language. On completion of this class the student should feel comfortable with writing Python programs. The course provides insight to the features of Python that make it an excellent choice for projects of virtually any size

**AUDIENCE**
This workshop is for those who have some experience in using at least one scripting language but who do not know Python. It is assumed that you can edit a text file using your favorite editor, and be able to execute your script file on the command line of a shell.

**PREREQUISITES**
This course requires participants to meet the following prerequisites:
- Introduction to operating system.
- Basic programming experience

**METHODOLOGY**
This program will be conducted through Instructor-led (classroom)

**COURSE OBJECTIVES**
After completing this course, students will be able to:
- use Python interactively
- execute a Python script at the shell prompt
- use Python types, expressions, and None
- use string literals and string type
- use Python statements (if...elif..else, for, pass, continue, . . . )
- understand the difference between expressions and statements
- understand assignment semantics
- write and call a simple function
- utilize high-level data types such as lists and dictionaries
- understand the difference between mutable and immutable types
- write a simple class and access methods and attributes
- import and utilize a module
- read from and write to a text file

**COURSE OUTLINE**

**Module 1 - Why Python?**
- Why do people use Python?
- Is Python a scripting language?
- What can I do with Python?
- What are Python's technical strengths?

**Module 2 - How Python Runs Programs**
- Introducing the Python interpreter
- Program execution
- Execution model variations

**Module 3 -Numbers**
- Python program structure
- Why use built-in types?
- Numbers
- Python expression Operators
- Numbers in action
- The dynamic typing interlude

**Module 4 - Strings**
- String literals
- Strings in action
- String formatting
- String methods
- General type categories

**Module 5 - Lists & Dictionaries**
- Lists
- Lists in action
- Dictionaries
- Dictionaries in action

**Module 6 - Tuples, Files & Everything Else**
- Tuples
- Files
- Type categories revisited
- Object generality
- References versus copies
- Comparisons, equality, and truth
- Python's type hierarchies
- Other types in Python
- Built-in type gotchas

**Module 7 - Assignment, Expressions, & Print**
- Assignment statements
- Expression statements
- Print statements

**Module 8 - If Tests**
- If statements
- Python syntax rules
- Truth tests

**Module 9 - While and For Loops**
- While loops
- Break, continue, pass, and the loop else
- For loops
- Loop variations

**Module 10 - Function Basics**
- Why use functions?
- Coding functions
- Definition & calls
- Intersecting sequences

**Module 11 - Scopes & Arguments**
- Scope rules
- The global statement
- Scopes and nested functions
- Passing arguments
- Special argument matching modes

**Module 12 - The Big Picture**
- Why use modules?
- Python program architecture
- How imports work

**Module 13 - Coding Basics**
- Module creation
- Module usage
- Module namespaces
- Reloading modules

**Module 14 - Packages**
- Package import basics
- Package import example
- Why use package imports?

**Module 15 - OOP: The Big Picture**
- Why use classes?
- OOP from 30,000 feet

**Module 16 - Class Coding Basics**
- Classes generate multiple instance objects
- Classes are customized by inheritance
- Classes can intercept python operators

**Module 17 - Class Coding Details**
- The class statement
- Methods
- Inheritance
- Operator overloading
- Namespaces

**Module 18 - Designing with Classes**
- Python and OOP
- Classes as records
- OOP and inheritance: "is-a" relationships
- OOP and composition: "has-a" relationships
- OOP and delegation
- Multiple inheritance
- Classes are objects: Generic object factories
- Methods are objects: Bound or unbound
- Classes versus modules

**Module 19 - Exception Basics**
- Why use exceptions?
- Exception handling: The short story
- The try/except/else statement
- The try/finally statement
- The raise statement
- The assert statement

**Module 20 - Exception Objects**
- String-based exceptions
- Class-based exceptions
- General raise statement forms

**Module 21 - Designing with Exceptions**
- Nesting exception handlers
- Exception idioms
- Exception design tips
- Exception gotchas
- Core language summary

**Module 22 - Common Tasks in Python**
- Exploring on your own
- Conversions, numbers and comparisons
- Manipulating strings
- Data structure manipulations
- Manipulating files and directories
- Internet-related modules
- Executing programs
- Debugging, testing, timing, profiling

**Module 23 - Advanced Topics**
- Web frameworks
- GUI frameworks
- Content management frameworks