GLOBAL PRESENCE

WE PROVIDE QUALITY IT SERVICES & SOLUTIONS

Python
Course Equipment and Material

Course Equipment: Required computer hardware and software.

COURSE SYLLABUS

COURSE NAME: Python

OVERVIEW	Python is a general-purpose, versatile and popular programming language. It's great as a first language because it is concise and easy to read, and it is also a good language to have in any programmer's stack as it can be used for everything from web development to software development and scientific applications. it offers several advanced features that can help to greatly improve the programming experience. The latest releases of Python 2.x and 3.x add interesting features that can be used passively without deeper understanding about how they work. The course teaches how these features work and provides details about meta-programming and other advanced techniques.
ELIGIBILITY	Anyone who needs to learn how to write programs in Python. Students should have some experience with at least one programming language, ie. C, C++, Java, Perl, Ruby, VB or anything equivalent. This course targets medium level Python programmers who would like to dive deeper into the language. Alternatively, participants can attend the course Python for Programmers to be able to take full advantage of this advanced course.
DURATION	60 hours

LESSON 1:	An Introduction to Python - A Brief History of Python. Python Versions. Installing Python. Environment Variables. Executing Python from the Command Line. IDLE. Editing Python Files. Python Documentation. Getting Help. Dynamic Types. Python Reserved Words. Naming Conventions.
LESSON 2:	Basic Python Syntax - Basic Syntax. Comments. String Values. String Methods. The format Method. String Operators. Numeric Data Types. Conversion Functions. Simple Input and Output. The % Method. The print Function.
LESSON 3:	Language Components - Indenting Requirements. The if Statement. Relational Operators. Logical Operators. Bit Wise Operators. The while Loop. break and continue. The for Loop.
LESSON 4:	Collections- Lists. Tuples. Sets. Dictionaries. Sorting Dictionaries. Copying Collections.
LESSON 5:	Functions- Defining Your Own Functions. Parameters. Function Documentation. Keyword and Optional. Parameters. Passing Collections to a Function. Variable Number of Arguments. Scope. Functions - "First Class Citizens". Passing Functions to a Function. Mapping Functions in a Dictionary. Lambda. Inner Functions. Closures.
LESSON 6:	Modules – Introduction to Modules. Standard Modules – sys. Math. Time. dir

	Function
LESSON 7:	Exceptions – Errors. Run Time Errors. The Exception Model. Exception Hierarchy. Handling Multiple Exceptions – raise, assert. Writing Your Own Exception Classes.
LESSON 8:	Input and Output - Data Streams. Creating Your Own Data Streams. Access Modes. Writing Data to a File Reading Data from a File. Additional File Methods. Using Pipes as Data Streams. Handling IO Exceptions. Working with Directories. Metadata. Pickle Module.
LESSON 9:	Simple Character Matches- Special Characters. Character Classes. Quantifiers. The Dot Character. Greedy Matches. Grouping. Matching at Beginning or End. Match Objects. Substituting. Splitting a String. Compiling Regular Expressions. Flags
LESSON 10:	Comprehensions - The principle comes from the functional language Haskell but integrates very well into Python. After list comprehension came generator expressions followed by dictionary and set comprehensions. The course introduces this style of programming with examples focusing on advantages and disadvantages for certain tasks.
LESSON 11:	Iterators and generators - Iterators and generator make lazy evaluation, that is generating an object just when it is needed, very convenient. The concept of yielding instead of returning plays a central role. The course shows how to use generators to simplify programming tasks. Furthermore, coroutines will be used to implement concurrent solutions. An overview over the itertools module shows how to elegantly solve iteration tasks.
LESSON 12:	Decorators - Decorator provide a very useful method to add functionality to existing functions and classes. The course uses examples for caching, proxying, and checking of arguments to demonstrate how decorators can improve code readability and can simplify solutions.
LESSON 13:	Context managers - The with statement helps to make code more robust by simplifying exception handling. The course shows how to use the with statement with the standard library and how to write your own objects that take advantage of with. The contextlib from the standard library helps to make this easier.
LESSON 14:	Descriptors - Descriptors determine how attribute of object are accessed. The course uses examples to show how descriptors work and how they can be used to customize attribute access.
LESSON 15:	Metaclasses - Metaclasses offer a powerful way to change how classes in Python behave. Whíle being an advanced feature that should be used sparingly, it can provide interesting help for complex problems. The course shows how to apply metaclasses and gives examples where they can be useful.
LESSON 16:	Conventions - Python offers a lot of functionality out of the box where other languages need to use design patterns. These patterns are general solutions for certain types of problems.
LESSON 15:	Python offers what is called the "pythonic" way for solving a problem. The course presents of a few of these solutions: wrapping instead of inheritance. dependency injections. Factories. Duck typing. Monkey patching. Callbacks.
LESSON 16:	PATTERNS - "It's easier to ask for forgiveness than permission (EFAP)". One pythonic principle is "It's easier to ask for forgiveness than permission (EFAP)". Opposed to the approach to look before you leap, this principle states that you should first try an action and if it fails react appropriately. Python' strong exception handling supports this principle and helps to develop robust and fault tolerant programs.
LESSON 17:	Singleton - Singletons are objects of which only one instance is supposed to exist. Python provides several ways to implement singletons. These possibilities are shown using examples. Null Objects - Null objects can be used instead of the type None to avoid tests for None. Implementation, usage as well as advantages and disadvantages are covered.
LESSON 18:	Proxy - Proxies stand for other objects. Setup and usage of proxies are covered.

	Observer - The observer pattern allows several objects to have access to the same data. The principles of this pattern are shown with a comprehensive example.
LESSON 19:	Constructor - Parameters of constructors are often assigned to instance variables. This pattern can replace a many lines of manual assignment with only one line of code.

TAKE AWAYS	Knowledge to understand Python programming from the basis to high end. Python developer, Python developer, Data engineer, Validation engineer, Software engineer – Python, etc.