Python Programming

Why Learn Python ?

Python is a general-purpose language, which means it can be used to build just about anything, which will be made easy with the right tools/libraries.Professionally, Python is great for backend web development, data analysis, artificial intelligence, and scientific computing. Many developers have also used Python to build productivity tools, games, and desktop apps, so there are plenty of resources to help you learn how to do those as well.

Python is also one of the hottest skills to have. The latest Stack Overflow Developer Survey (2016) ranked Python as the 4th most popular and most wanted technology of the year. It is today the 2nd most popular programming language in the world based on the PYPL Popularity of Programming Language Index.(2017)

Why Learn Python? Here Are 8 Data-Driven Reasons :

1. You Can Use Python for Pretty Much Anything
2. Python Is Widely Used in Data Science
3. Python Pays Well
4. Demand for Python Developers Is High (And Growing)
5. Python Saves Time
6. Python Is Beginner Friendly
7. All the Big Names Use Python
8. Python Has an Amazing Ecosystem

Course Description :

This course is designed for any student interested in using computation to enhance their problem solving abilities. No prior experience in programming is necessary. Students will use their problem solving abilities to implement programs in Python

The learning objectives of this course are:

- To understand why Python is a useful scripting language for developers.
- To learn how to design and program Python applications.
- To learn how to use lists, tuples, and dictionaries in Python programs.
- To learn how to identify Python object types.
- To learn how to use indexing and slicing to access data in Python programs.
- To define the structure and components of a Python program.
- To learn how to write loops and decision statements in Python.
- To learn how to write functions and pass arguments in Python.
- To learn how to build and package Python modules for reusability.
- To learn how to read and write files in Python.
- To learn how to design object-oriented programs with Python classes.
- To learn how to use class inheritance in Python for reusability.
- To learn how to use exception handling in Python applications for error handling.
- To learn how to use use various specialized libraries and moudules in Python

Duration : 60 hrs

Impetus IT Services Pvt.Ltd.
B-16, First floor, Sant Tukaram Vyapar Sankul, Sector - 24, Nigdi, Pune, Maharashtra. India. Pin – 411044.
Mobile 9970600774, 9730012775|Board 91-20-27640406|Fax 91-20-27641703
Email : hrishikesh@impetusitservices.com | Website http://impetusits.in

Instructor led Complete Python Programming course

Course Content :

1. Introduction to Computers, Programs, and Python

1.1. What Is a Computer?
1.2. Programming Languages
1.3. Operating Systems
1.4. The History of Python
1.5. Getting Started with Python
1.6. Programming Style and Documentation
1.7. Programming Errors
1.8. Getting Started with Graphics Programming

2. Elementary Programming

2.1. Writing a Simple Program
2.2. Reading Input from the Console
2.3. Identifiers
2.4. Variables, Assignment Statements, and Expressions
2.5. Simultaneous Assignments
2.6. Named Constants
2.7. Numeric Data Types and Operators
2.8. Evaluating Expressions and Operator Precedence
2.9. Augmented Assignment Operators
2.10. Type Conversions and Rounding
2.11. Software Development Process
2.12. Case Study

3. Mathematical Functions, Strings, and objects

3.1. Introduction
3.2. Common Python Functions
3.3. Strings and Characters
3.4. Introduction to Objects and Methods
3.5. Formatting Numbers and Strings
3.6. Drawing Various Shapes
3.7. Drawing with Colors and Fonts
3.8. Case Study

4. Selections

4.1. Introduction
4.2. Boolean Types, Values, and Expressions
4.3. Generating Random Numbers
4.4. if Statements
4.5. Case Study; Guessing Birthdays
4.6. Two-Way if-else Statements
4.7. Nested if and Multi-Way if-el if-else Statements
4.8. Common Errors in Selection Statements
4.9. Logical Operators
4.10. Conditional Expressions
4.11. Operator Precedence and Associativity
4.12. Detecting the Location of an Object
4.13. Case Study

Impetus IT Services Pvt.Ltd.
B-16, First floor, Sant Tukaram Vyapar Sankul, Sector - 24, Nigdi, Pune, Maharashtra. India. Pin – 411044.
Mobile 9970600774, 9730012775|Board 91-20-27640406|Fax 91-20-27641703
Email : hrishikesh@impetusitservices.com | Website http://impetusits.in

5. Loops

5.1. The while Loop
5.2. The for Loop
5.3. Nested Loops
5.4. Minimizing Numerical Errors
5.5. Keywords break and continue
5.6. Case Study

6. Functions

6.1. Introduction
6.2. Defining a function
6.3. Calling a function
6.4. Functions with/without Return Values
6.5. Positional and Keyword Arguments
6.6. Passing Arguments by Reference Values
6.7. Modularizing Code
6.8. The Scope of Variables
6.9. Default Arguments
6.10. Returning Multiple Values
6.11. Function Abstraction and Stepwise Refinement
6.12. Case Study

7. Objects and Classes

7.1. Defining Classes for Objects
7.2. UML Class Diagrams
7.3. Immutable Objects vs. Mutable Objects
7.4. Hiding Data Fields
7.5. Class Abstraction and Encapsulation
7.6. Object-Oriented Thinking

8. More on Strings and Special Methods

8.1. The str Class
8.2. Case Study; Checking Palindromes
8.3. Case Study: Converting Hexadecimals to Decimals
8.4. Operator Overloading and Special Methods
8.5. Case Study: The Rational Class

9. GUI Programming Using Tkinter

9.1. Getting Started with Tkinter
9.2. Processing Events
9.3. The Widget Classes
9.4. Canvas
9.5. The Geometry Managers
9.6. Case Study: Loan Calculator
9.7. Displaying Images
9.8. Menus
9.9. Popup Menus
9.10. Mouse, Key Events, and Bindings
9.11. Animations
9.12. Scrollbars
9.13. Standard Dialog Boxes

10. Lists

10.1. List Basics
10.2. Copying Lists
10.3. Passing Lists to Functions
10.4. Returning a List from a Function
10.5. Searching Lists
10.6. Sorting Lists
10.7. Case Study

11. Multidimensional Lists

11.1. Processing Two-Dimensional Lists
11.2. Passing Two-Dimensional Lists to Functions
11.3. Case Study

12. Inheritance and Polymorphism

12.1. Superclasses and Subclasses
12.2. Overriding Methods
12.3. The object Class
12.4. Polymorphism and Dynamic Binding
12.5. The isinstance Function
12.6. Class Relationships
12.7. Designing a Class for Stacks
12.8. Case Study

13. Files and Exception Handling

13.1. Text Input and Output
13.2. File Dialogs
13.3. Retrieving Data from the Web
13.4. Exception Flandling
13.5. Raising Exceptions
13.6. Processing Exceptions Using Exception Objects
13.7. Defining Custom Exception Classes
13.8. Binary IO Using Pickling
13.9. Case Study

14. Tuples, Sets, and Dictionaries

14.1. Tuples
14.2. Sets
14.3. Comparing the Performance of Sets and Lists
14.4. Dictionaries
14.5. Case Study

15. Recursion

15.1. Problem Solving Using Recursion
15.2. Recursive Helper Functions
15.3. Recursion vs. Iteration
15.4. Tail Recursion
15.5. Case Study