Python

Python Training Overview

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python has been one of the premier, flexible, and powerful open-source language that is easy to learn, easy to use, and has powerful libraries for data manipulation and analysis

What are the Python Course Pre-requisites

There are no hard pre-requisites. Basic understanding of Computer Programming terminologies is sufficient. Also, basic concepts related to Programming and Database is beeficial but not mandatory.

Objectives of the Course

- To understand the concepts and constructs of Python
- To create own Python programs, know the machine learning algorithms in Python and work on a real-time project running on Python

Who should do the course

- Big Data Professionals
- IT Developers
- Those who are showing interest to build their career in Python

Python Training Course Duration

- 35 Days, Daily 1 Hours

Python Course Content

Core Python

Introduction to Languages

- What is Language?
- Types of languages
- Introduction to Translators
- Compiler
- Interpreter
- What is Scripting Language?
- Types of Script
- Programming Languages v/s Scripting Languages
- Difference between Scripting and Programming languages
- What is programming paradigm?
- Procedural programming paradigm
- Object Oriented Programming paradigm

Introduction to Python

- What is Python?

Python

- WHY PYTHON?
- History
- Features – Dynamic, Interpreted, Object oriented, Embeddable, Extensible, Large standard libraries, Free and Open source
- Why Python is General Language?
- Limitations of Python
- What is PSF?
- Python implementations
- Python applications
- Python versions
- PYTHON IN REALTIME INDUSTRY
- Difference between Python 2.x and 3.x
- Difference between Python 3.7 and 3.8
- Software Development Architectures

Python Software's

- Python Distributions
- Download &Python Installation Process in Windows, Unix, Linux and Mac
- Online Python IDLE
- Python Real-time IDEs like Spyder, Jupyter Note Book, PyCharm, Rodeo, Visual Studio Code, ATOM, PyDevetc

Python Language Fundamentals

- Python Implementation Alternatives/Flavors
- Keywords
- Identifiers
- Constants / Literals
- Data types
- Python VS JAVA
- Python Syntax

Different Modes of Python

- Interactive Mode
- Scripting Mode
- Programming Elements
- Structure of Python program
- First Python Application
- Comments in Python
- Python file extensions
- Setting Path in Windows
- Edit and Run python program without IDE
- Edit and Run python program using IDEs
- INSIDE PYTHON
- Programmers View of Interpreter
- Inside INTERPRETER
- What is Byte Code in PYTHON?
- Python Debugger

Python Variables

Python

- bytes Data Type
- byte array
- String Formatting in Python
- Math, Random, Secrets Modules
- Introduction
- Initialization of variables
- Local variables
- Global variables
- ‘global’ keyword
- Input and Output operations
- Data conversion functions – int(), float(), complex(), str(), chr(), ord()

Operators

- Arithmetic Operators
- Comparison Operators
- Python Assignment Operators
- Logical Operators
- Bitwise Operators
- Shift operators
- Membership Operators
- Identity Operators
- Ternary Operator
- Operator precedence
- Difference between “is” vs “==”

Input & Output Operators

- Print
- Input
- Command-line arguments

Control Statements

- Conditional control statements
- If
- If-else
- If-elif-else
- Nested-if
- Loop control statements
- for
- while
- Nested loops
- Branching statements
- Break
- Continue
- Pass
- Return
- Case studies

Data Structures or Collections

Python

- Introduction
- Importance of Data structures
- Applications of Data structures
- Types of Collections
- Sequence
- Strings, List, Tuple, range
- Non sequence
- Set, Frozen set, Dictionary
- **Strings**
- What is string
- Representation of Strings
- Processing elements using indexing
- Processing elements using Iterators
- Manipulation of String using Indexing and Slicing
- String operators
- Methods of String object
- String Formatting
- String functions
- String Immutability
- Case studies

List Collection

- What is List
- Need of List collection
- Different ways of creating List
- List comprehension
- List indices
- Processing elements of List through Indexing and Slicing
- List object methods
- List is Mutable
- Mutable and Immutable elements of List
- Nested Lists
- List_of_lists
- Hardcopy, shallowCopy and DeepCopy
- zip() in Python
- How to unzip?
- Python Arrays:
- Case studies

Tuple Collection

- What is tuple?
- Different ways of creating Tuple
- Method of Tuple object
- Tuple is Immutable
- Mutable and Immutable elements of Tuple
- Process tuple through Indexing and Slicing
- List v/s Tuple
- Case studies

Set Collection

Python

- What is set?
- Different ways of creating set
- Difference between list and set
- Iteration Over Sets
- Accessing elements of set
- Python Set Methods
- Python Set Operations
- Union of sets
- functions and methods of set
- Python Frozen set
- Difference between set and frozenset ?
- Case study

Dictionary Collection

- What is dictionary?
- Difference between list, set and dictionary
- How to create a dictionary?
- PYTHON HASHING?
- Accessing values of dictionary
- Python Dictionary Methods
- Copying dictionary
- Updating Dictionary
- Reading keys from Dictionary
- Reading values from Dictionary
- Reading items from Dictionary
- Delete Keys from the dictionary
- Sorting the Dictionary
- Python Dictionary Functions and methods
- Dictionary comprehension

Functions

- What is Function?
- Advantages of functions
- Syntax and Writing function
- Calling or Invoking function
- Classification of Functions
- No arguments and No return values
- With arguments and No return values
- With arguments and With return values
- No arguments and With return values
- Recursion
- Python argument type functions :
- Default argument functions
- Required(Positional) arguments function
- Keyword arguments function
- Variable arguments functions
- ‘pass’ keyword in functions
- Lambda functions/Anonymous functions
- map()
- filter()
- reduce()

Python

- Nested functions
- Non local variables, global variables
- Closures
- Decorators
- Generators
- Iterators
- Monkey patching

Advanced Python

Python Modules

- Importance of modular programming
- What is module
- Types of Modules – Pre defined, User defined.
- User defined modules creation
- Functions based modules
- Class based modules
- Connecting modules
- Import module
- From … import
- Module alias / Renaming module
- Built In properties of module

Packages

- Organizing python project into packages
- Types of packages – pre defined, user defined.
- Package v/s Folder
- py file
- Importing package
- **PIP**
- Introduction to PIP
- Installing PIP
- Installing Python packages
- Un installing Python packages

OOPs

- Procedural v/s Object oriented programming
- Principles of OOP – Encapsulation , Abstraction (Data Hiding)
- Classes and Objects
- How to define class in python
- Types of variables – instance variables, class variables.
- Types of methods – instance methods, class method, static method
-
- Object initialization
- 'self' reference variable
- 'cls' reference variable
- Access modifiers – private(__) , protected(_), public
- AT property class
- Property() object

Python

- Creating object properties using setaltr, getaltr functions
- Encapsulation(Data Binding)
- What is polymorphism?
- Overriding

1. i) Method overriding
2. ii) Constructor overriding

- Overloading

1. i) Method Overloading
2. ii) Constructor Overloading

iii) Operator Overloading

- Class re-usability
- Composition
- Aggregation
- Inheritance – single , multi level, multiple, hierarchical and hybrid inheritance and Diamond inheritance
- Constructors in inheritance
- Object class
- super()
- Runtime polymorphism
- Method overriding
- Method resolution order(MRO)
- Method overriding in Multiple inheritance and Hybrid Inheritance
- Duck typing
- Concrete Methods in Abstract Base Classes
- Difference between Abstraction & Encapsulation
- Inner classes
- Introduction
- Writing inner class
- Accessing class level members of inner class
- Accessing object level members of inner class
- Local inner classes
- Complex inner classes
- Case studies

Exception Handling & Types of Errors

- What is Exception?
- Why exception handling?
- Syntax error v/s Runtime error
- Exception codes – AttributeError, ValueError, IndexError, TypeError…
- Handling exception – try except block
- Try with multi except
- Handling multiple exceptions with single except block
- Finally block
- Try-except-finally
- Try with finally
- Case study of finally block
- Raise keyword
- Custom exceptions / User defined exceptions

Python

- Need to Custom exceptions
- Case studies

Regular expressions

- Understanding regular expressions
- String v/s Regular expression string
- “re” module functions
- Match()
- Search()
- Split()
- Findall()
- Compile()
- Sub()
- Subn()
- Expressions using operators and symbols
- Simple character matches
- Special characters
- Character classes
- Mobile number extraction
- Mail extraction
- Different Mail ID patterns
- Data extraction
- Password extraction
- URL extraction
- Vehicle number extraction
- Case study

File &Directory handling

- Introduction to files
- Opening file
- File modes
- Reading data from file
- Writing data into file
- Appending data into file
- Line count in File
- CSV module
- Creating CSV file
- Reading from CSV file
- Writing into CSV file
- Object serialization – pickle module
- XML parsing
- JSON parsing

Python Logging

- Logging Levels
- implement Logging
- Configure Log File in over writing Mode
- Timestamp in the Log Messages
- Python Program Exceptions to the Log File
- Requirement of Our Own Customized Logger

Python

- Features of Customized Logger

Date & Time module

- How to use Date & Date Time class
- How to use Time Delta object
- Formatting Date and Time
- Calendar module
- Text calendar
- HTML calendar

OS module

- Shell script commands
- Various OS operations in Python
- Python file system shell methods
- Creating files and directories
- Removing files and directories
- Shutdown and Restart system
- Renaming files and directories
- Executing system commands

Multi-threading & Multi Processing

- Introduction
- Multi tasking v/s Multi threading
- Threading module
- Creating thread – inheriting Thread class , Using callable object
- Life cycle of thread
- Single threaded application
- Multi threaded application
- Can we call run() directly?
- Need to start() method
- Sleep()
- Join()
- Synchronization – Lock class – acquire(), release() functions
- Case studies

Garbage collection

- Introduction
- Importance of Manual garbage collection
- Self reference objects garbage collection
- ‘gc’ module
- Collect() method
- Threshold function
- Case studies

Python Data Base Communications(PDBC)

- Introduction to DBMS applications
- File system v/s DBMS
- Communicating with MySQL

Python

- Python – MySQL connector
- connector module
- connect() method
- Oracle Database
- Install cx_Oracle
- Cursor Object methods
- execute() method
- executeMany() method
- fetchone()
- fetchmany()
- fetchall()
- Static queries v/s Dynamic queries
- Transaction management
- Case studies

Python – Network Programming

- What is Sockets?
- What is Socket Programming?
- The socket Module
- Server Socket Methods
- Connecting to a server
- A simple server-client program
- Server
- Client

Tkinter & Turtle

- Introduction to GUI programming
- Tkinter module
- Tk class
- Components / Widgets
- Label , Entry , Button , Combo, Radio
- Types of Layouts
- Handling events
- Widgets properties
- Case studies

Data analytics modules

- Numpy
- Introduction
- Scipy
- Introduction
- Arrays
- Datatypes
- Matrices
- N dimension arrays
- Indexing and Slicing
- Pandas
- Introduction
- Data Frames
- Merge , Join, Concat

Python

- MatPlotLib introduction
- Drawing plots
- Introduction to Machine learning
- Types of Machine Learning?
- Introduction to Data science

DJANGO

- Introduction to PYTHON Django
- What is Web framework?
- Why Frameworks?
- Define MVT Design Pattern
- Difference between MVC and MVT

<u>PANDAS</u>

Pandas – Introduction

Pandas – Environment Setup

Pandas – Introduction to Data Structures

- Dimension & Description
- Series
- DataFrame
- Data Type of Columns
- Panel

Pandas — Series

- Series
- Create an Empty Series
- Create a Series f
- rom ndarray
- rom dict
- rom Scalar
- Accessing Data from Series with Position
- Retrieve Data Using Label (Index)

Pandas – DataFrame

- DataFrame
- Create DataFrame
- Create an Empty DataFrame
- Create a DataFrame from Lists
- Create a DataFrame from Dict of ndarrays / Lists
- Create a DataFrame from List of Dicts
- Create a DataFrame from Dict of Series
- Column Selection
- Column Addition
- Column Deletion
- Row Selection, Addition, and Deletion

Python

Pandas – Panel

- Panel()
- Create Panel
- Selecting the Data from Panel

Pandas – Basic Functionality

- DataFrame Basic Functionality

Pandas – Descriptive Statistics

- Functions & Description
- Summarizing Data

Pandas – Function Application

- Table-wise Function Application
- Row or Column Wise Function Application
- Element Wise Function Application

Pandas – Reindexing

- Reindex to Align with Other Objects
- Filling while ReIndexing
- Limits on Filling while Reindexing
- Renaming

Pandas – Iteration

- Iterating a DataFrame
- iteritems()
- iterrows()
- itertuples()

Pandas – Sorting

- By Label
- Sorting Algorithm

Pandas – Working with Text Data

Pandas – Options and Customization

- get_option(param)
- set_option(param,value)
- reset_option(param)
- describe_option(param)
- option_context()

Pandas – Indexing and Selecting Data

Python

- .loc()
- .iloc()
- .ix()
- Use of Notations

Pandas – Statistical Functions

- Percent_change
- Covariance
- Correlation
- Data Ranking

Pandas – Window Functions

- .rolling() Function
- .expanding() Function
- .ewm() Function

Pandas – Aggregations

- Applying Aggregations on DataFrame

Pandas – Missing Data

- Cleaning / Filling Missing Data
- Replace NaN with a Scalar Value
- Fill NA Forward and Backward
- Drop Missing Values
- Replace Missing (or) Generic Values

Pandas – GroupBy

- Split Data into Groups
- View Groups
- Iterating through Groups
- Select a Group
- Aggregations
- Transformations
- Filtration

Pandas – Merging/Joining

- Merge Using 'how' Argument

Pandas – Concatenation

- Concatenating Objects
- Time Series

Pandas – Date Functionality

Pandas – Timedelta

Python

Pandas – Categorical Data

- Object Creation

Pandas – Visualization

- Bar Plot
- Histograms
- Box Plots
- Area Plot
- Scatter Plot
- Pie Chart

Pandas – IO Tools

- csv

Pandas – Sparse Data

Pandas – Caveats & Gotchas

Pandas – Comparison with SQL

<u>NUMPY</u>

NUMPY − INTRODUCTION

NUMPY − ENVIRONMENT

NUMPY − NDARRAY OBJECT

NUMPY − DATA TYPES

- Data Type Objects (dtype)

NUMPY − ARRAY ATTRIBUTES

- shape
- ndim
- itemsize
- flags

NUMPY − ARRAY CREATION ROUTINES

- empty
- zeros
- ones

NUMPY − ARRAY FROM EXISTING DATA

- asarray
- frombuffer

- fromiter

NUMPY − ARRAY FROM NUMERICAL RANGES

- arange
- linspace
- logspace

NUMPY − INDEXING & SLICING

NUMPY − ADVANCED INDEXING

- Integer Indexing
- Boolean Array Indexing

NUMPY − BROADCASTING

NUMPY − ITERATING OVER ARRAY

- Iteration
- Order
- Modifying Array Values
- External Loop
- Broadcasting Iteration

NUMPY – ARRAY MANIPULATION

- reshape
- ndarray.flat
- ndarray.flatten
- ravel
- transpose
- ndarray.T
- swapaxes
- rollaxis
- broadcast
- broadcast_to
- expand_dims
- squeeze
- concatenate
- stack
- hstack and numpy.vstack
- split
- hsplit and numpy.vsplit
- resize
- append
- insert
- delete
- unique

NUMPY – BINARY OPERATORS

- bitwise_and

Python

- bitwise_or
- invert()
- left_shift
- right_shift

NUMPY − STRING FUNCTIONS

NUMPY − MATHEMATICAL FUNCTIONS

- Trigonometric Functions
- Functions for Rounding

NUMPY − ARITHMETIC OPERATIONS

- reciprocal()
- power()
- mod()

NUMPY − STATISTICAL FUNCTIONS

- amin() and numpy.amax()
- ptp()
- percentile()
- median()
- mean()
- average()
- Standard Deviation
- Variance

NUMPY − SORT, SEARCH & COUNTING FUNCTIONS

- sort()
- argsort()
- lexsort()
- argmax() and numpy.argmin()
- nonzero()
- where()
- extract()

NUMPY − BYTE SWAPPING

- ndarray.byteswap()

NUMPY − COPIES & VIEWS

- No Copy
- View or Shallow Copy
- Deep Copy

NUMPY − MATRIX LIBRARY

- empty()

Python

- matlib.zeros()
- matlib.ones()
- matlib.eye()
- matlib.identity()
- matlib.rand()

NUMPY − LINEAR ALGEBRA

- dot()
- vdot()
- inner()
- matmul()
- Determinant
- linalg.solve()

NUMPY − MATPLOTLIB

- Sine Wave Plot
- subplot()
- bar()

NUMPY – HISTOGRAM USING MATPLOTLIB

- histogram()
- plt()

NUMPY − I/O WITH NUMPY

- save()
- savetxt()