DVS PYTHON Content

Table of Content:

Duration: **1.5 Months long instructor- led pure class room training**

Features:

- ✓ Main focus on hands-on training
- ✓ Guidance in Resume Preparation
- ✓ Contains real world business problems and examples.
- ✓ Rich material and handouts for reference
- ✓ Life Time Validity for re-attending classes

1. Introduction to Python

- ✓ What is Python
- ✓ Where are all python is using?
- ✓ History of Python?
- ✓ Why the name was python?
- ✓ Python versions
- ✓ Python supports Functional and OOPs as well
- ✓ Solving few requirements by using C, Java and python
- ✓ Machine language
- ✓ Translator
- ✓ Interpreter
- ✓ Compiler
- ✓ Python keywords (33)
- ✓ Features of Python
- ✓ Flavors of Python
- ✓ Python versions
- ✓ Programs

2. Coding introduction

- ✓ Installing Python on Windows OS
- ✓ Ways to write python program
- ✓ Python program execution steps
- ✓ Understanding first python program
- ✓ First python program executing by using python IDLE
- ✓ Python program internal flow: Short answer

- ✓ Python program internal flow: Long answer
- ✓ Python Virtual Machine
- ✓ JIT Compiler
- ✓ Programs

3. Naming conventions in python

- ✓ What is an identifier?
- ✓ Why should we follow naming conventions?
- ✓ Rules to define identifiers in Python:
- ✓ Summary points about rules
- ✓ Validating identifier names by taking examples
- ✓ Python program total identifiers in simple table
- ✓ Smart suggestions while writing identifiers
- ✓ Indentation and Comments
- ✓ Programs

4. Variables in python

- ✓ What is variable?
- ✓ Properties of variable
- ✓ Creating variable
- ✓ Invalid cases for variables
- ✓ Multiple variables in single line
- ✓ Single value for multiple variables
- ✓ Variable re-initialization
- ✓ Programs

5. Data types in python

- ✓ What is data type?
- ✓ type() function
- ✓ Different types of data types
- ✓ Built-in data types
- ✓ Numeric types
- ✓ int data type
- ✓ float data type

- ✓ complex data type
- ✓ bool data type (boolean data type)
- ✓ None data type
- ✓ Sequences in Python
- ✓ str data type
- ✓ bytes data type
- ✓ bytearray data type
- ✓ list data structure
- ✓ tuple data structure
- ✓ range data type
- ✓ Accessing range values
- ✓ set data structure
- ✓ dictionary data structure
- ✓ User defined data types
- ✓ Converting from one data type into another data type.
- ✓ Programs

6. Operators

- ✓ What is an operator?
- ✓ Different type of operators
- ✓ Arithmetic Operators
- ✓ Assignment operator
- ✓ Unary minus operator
- ✓ Relational operators
- ✓ Logical operators
- ✓ Membership operators
- ✓ Identity operators
- ✓ Programs

7. Input and Output

- ✓ Why should we learn about input and output?
- ✓ Input and output
- ✓ Convert from string type into other type
- ✓ eval() function
- ✓ Command line arguments
- ✓ IndexError

- ✓ len() function
- ✓ Programs

8. Flow control

- ✓ Why should we learn about flow control?
- ✓ Flow control
- ✓ Sequential flow
- ✓ Conditional flow
- ✓ Looping flow
- ✓ Sequential statements
- ✓ Conditional or Decision-making statements
- ✓ if statement
- ✓ if else statement
- ✓ if elif else statement
- ✓ Looping
- ✓ while loop
- ✓ for loop
- ✓ Infinite loops
- ✓ Nested loops
- ✓ Loops with else block (or) else suit
- ✓ Where loops with else block flow is helpful?
- ✓ break statement
- ✓ continue statement
- ✓ pass statement
- ✓ return statement
- ✓ Programs

9. String in python

- ✓ Gentle reminder
- ✓ What is a string?
  - Definition 1
  - Definition 2
- ✓ String is more popular
- ✓ Creating string
- ✓ When should we go for triple single and triple double quotes?
- ✓ Multi-line string objects

- ✓ Empty string
- ✓ Accessing string characters
- ✓ Indexing in string
- ✓ Python support two types of indexes
- ✓ Positive index
- ✓ Negative index
- ✓ Internal representation
- ✓ Slicing in string
- ✓ Internal representation
- ✓ Default values
- ✓ String several cases in slicing
- ✓ What is Immutable?
- ✓ Strings are immutable
- ✓ Mathematical operators on string objects
- ✓ Interesting about + and * operators
- ✓ Addition (+) operator with string
- ✓ Multiplication (*) operator with string
- ✓ Length of the string
- ✓ Membership operators
- ✓ Comparing strings
- ✓ Removing spaces from String
- ✓ Predefined methods to removes spaces
- ✓ Finding substrings
- ✓ index() method
- ✓ Counting substring in the given String
- ✓ Replacing a string with another string
- ✓ String objects are immutable, Is replace () method will modify the string objects?
- ✓ Splitting of Strings
- ✓ Joining of Strings
- ✓ Changing case of a String
- ✓ Formatting the Strings
- ✓ Character data type

10. Function

- ✓ General example why function required

- ✓ When should we go for function?
- ✓ What is Function?
- ✓ Advantage of function
- ✓ Types of function
- ✓ Pre-defined or built-in functions
- ✓ User Defined Functions:
- ✓ Function related terminology
- ✓ Function explanation
- ✓ Defining a function
- ✓ Calling a function
- ✓ Function without parameters
- ✓ Function with parameters
- ✓ return keyword in python
- ✓ return vs None
- ✓ Returning multiple values from function
- ✓ Function can call other function
- ✓ Understandings
- ✓ Functions are first class objects
- ✓ Assigning a function to variable
- ✓ Pass function as a parameter to another function.
- ✓ Define one function inside another function
- ✓ function can return another function
- ✓ Formal and actual arguments
- ✓ Types of arguments
- ✓ Positional arguments
- ✓ Keyword arguments
- ✓ Default arguments
- ✓ non-default argument follows default argument
- ✓ Variable length arguments
- ✓ keyword variable length argument (**variable)
- ✓ Function vs Module vs Library:
- ✓ Types of variables
- ✓ Local variables
- ✓ Global variables
- ✓ The global keyword
- ✓ When we can choose global keyword?
- ✓ Recursive function

- ✓ Anonymous functions or Lambdas
- ✓ Advantage
- ✓ A simple difference between normal and lambda functions
- ✓ Where lambda function fits exactly?
- ✓ filter() function
- ✓ map() function
- ✓ reduce() function
- ✓ Function Aliasing
- ✓ Function decorators
- ✓ Decorator explanation
- ✓ Steps to create decorator
- ✓ @ symbol
- ✓ Function generators
- ✓ next(generator) function
- ✓ Programs

11. Modules

- ✓ What is module?
- ✓ import module
- ✓ Renaming or aliasing a module
- ✓ from and import keywords
- ✓ import * (star symbol)
- ✓ member aliasing
- ✓ Reloading a Module:
- ✓ dir() function
- ✓ The Special variable __name__
- ✓ Programs

12. Package

- ✓ What is package?
- ✓ Advantage
- ✓ Programs

13. List Data structure

- ✓ Why should we learn about data structures?
- ✓ Python Data structures
- ✓ Sequence of elements
- ✓ Introduction about list data structure
- ✓ Creating list
- ✓ creating empty list
- ✓ Creating list with elements
- ✓ Creating list by using list() function
- ✓ Difference between list() function and list class
- ✓ list having mutable nature
- ✓ Accessing elements from list
- ✓ Accessing list by using index
- ✓ What is Indexing
- ✓ Accessing list elements by Slicing
- ✓ Accessing list by using loops
- ✓ Important functions and methods in list
- ✓ len() function
- ✓ count() method
- ✓ append method
- ✓ insert() method
- ✓ Difference between append and insert
- ✓ extend() method
- ✓ remove() method
- ✓ pop() method
- ✓ Difference between remove() and pop() methods
- ✓ Ordering elements of List
- ✓ reverse()
- ✓ sort() method
- ✓ Aliasing and Cloning of list objects
- ✓ Aliasing list visualization
- ✓ Cloning or Copying
- ✓ By using slice operator
- ✓ By using copy method
- ✓ Mathematical + and * operators
- ✓ Concatenation operator +

- ✓ Repetition operator *
- ✓ Comparison of lists
- ✓ Membership operators
- ✓ Nested Lists
- ✓ list comprehensions
- ✓ Programs

14. Tuple data structure

- ✓ What is tuple data structure?
- ✓ When should we go for tuple data structure?
- ✓ Single value tuple
- ✓ Different ways to create a tuple.
- ✓ empty tuple
- ✓ Single value tuple
- ✓ Tuple with group of values
- ✓ By using tuple() function
- ✓ Accessing elements of tuple:
- ✓ Accessing tuple elements by using Index
- ✓ Accessing tuple elements by using slice operator:
- ✓ Tuple vs immutability:
- ✓ Mathematical operators on tuple:
- ✓ Concatenation operator (+):
- ✓ Multiplication operator (*)
- ✓ Important functions and methods of Tuple:
- ✓ len() function
- ✓ count() method
- ✓ index() method
- ✓ sorted() function
- ✓ min() and max() functions:
- ✓ Tuple packing
- ✓ Tuple unpacking:
- ✓ Tuple comprehension
- ✓ Differences between List and Tuple:
- ✓ Programs

15. Set Data structure

- ✓ What is set data structure?
- ✓ What symbol is required to create set?
- ✓ Set elements separated by what?
- ✓ Creating set by using same type of elements
- ✓ Creating set by using different type of elements
- ✓ Creating set by range() type of elements
- ✓ Difference between set() function and set class
- ✓ Empty set
- ✓ Important function and methods of set:
- ✓ add(x) method
- ✓ update(x, y) method
- ✓ Difference between add() and update() methods in set?
- ✓ Which of the following are valid for set s?
- ✓ copy() method
- ✓ pop() method
- ✓ remove(x) method
- ✓ discard(x) method
- ✓ clear() method
- ✓ Mathematical operations on the Set
- ✓ union() method
- ✓ intersection() method
- ✓ difference() method
- ✓ symmetric_difference():
- ✓ Membership operators: (in and not in)
- ✓ Set Comprehension
- ✓ Remove duplicates in list elements
- ✓ Frozen set
- ✓ Creating frozen set
- ✓ Programs

16. Dictionary data structure

- ✓ What is dictionary data structure?
- ✓ What symbol is required to create set?
- ✓ Create dictionary
- ✓ Empty dictionary

- ✓ Access values by using keys from dictionary
- ✓ How to handle this KeyError?
- ✓ Update dictionary
- ✓ Removing or deleting elements from dictionary
- ✓ By using del keyword
- ✓ We can delete total dictionary object
- ✓ clear() method
- ✓ Important functions and methods of dictionary
- ✓ dict() function
- ✓ dict({key1:value1, key2:value2}) function
- ✓ dict([tuple1, tuple2])
- ✓ len() function
- ✓ clear() method
- ✓ get() method
- ✓ pop() method
- ✓ popitem() method
- ✓ keys() method
- ✓ values() method
- ✓ items() method
- ✓ copy() method
- ✓ Dictionary Comprehension
- ✓ Programs

17. OOPS Part 1 -class, object, variables, methods

- ✓ Before OOPS
- ✓ Limitations
- ✓ After OOPS
- ✓ Advantages
- ✓ Is Python follows Functional approach or Object-oriented approach
- ✓ OOPs (Object Oriented Programming Principles)
- ✓ Features of Object-Oriented Programming System
- ✓ Class

- Definition
- Definition

✓ How to define a class
✓ Brief discussion about class
✓ Self-Variable
✓ object
✓ Why should we create an object?
✓ What is an object?
- Definition 1
- Definition 2
- Definition 3
- Definition 4

✓ Syntax to create an object
✓ Calling instance data
✓ Can we create more than one object?
✓ Constructor
✓ What is the main purpose of constructor?
✓ When constructor will be executed?
✓ How many times constructor will be executed?
✓ Is constructor mandatory to define?
✓ Can I call constructor explicitly like a method?
✓ How many times constructor will be executed?
✓ Constructor can contain how many parameters?
✓ If constructor having no parameters, then how to define?
✓ If trying to print Self Variable
✓ If constructor having more parameters, then how to define?
✓ Difference between methods and constructor
✓ Types of Variables:
✓ 1. Instance variables:
✓ What is instance variable?
✓ Separate copy for every object
✓ Where should we declare instance variable?
✓ By using constructor
✓ The __dict__ attribute
✓ By using instance method
✓ By using object name
✓ Accessing instance variable

- ✓ By using Self Variable
- ✓ By using object name
- ✓ Every object has a separate copy of variable exists
- ✓ static variable (class level variable)
- ✓ What is static variable
- ✓ Where can we declare static variable?
- ✓ Only one copy of static variable to all objects
- ✓ How can we access static variable?
- ✓ Instance Variable vs Static Variable
- ✓ Declaring static variable
- ✓ Inside class and outside of the method
- ✓ Inside constructor
- ✓ Inside instance method
- ✓ Inside class method
- ✓ class name
- ✓ cls variable
- ✓ Inside static method
- ✓ @static method declarator
- ✓ Accessing static variable
- ✓ Inside constructor
- ✓ Inside instance method
- ✓ Inside class method
- ✓ Inside static method
- ✓ Local variable
- ✓ What is local variable
- ✓ Why we need to use local variable
- ✓ Where can we access local variable
- ✓ Types of methods
- ✓ Instance methods
- ✓ What is the use of Setter and Getter Methods
- ✓ Setter method
- ✓ Getter Method
- ✓ Class Methods
- ✓ What is class method
- ✓ When we can go to class methods
- ✓ @classmethod decorator
- ✓ How to access class methods

- ✓ static Methods
- ✓ How to declare static method
- ✓ How to access static methods
- ✓ Passing members of one class to another class
- ✓ Inner classes
- ✓ Garbage Collection
- ✓ What is the main objective of Garbage Collector
- ✓ How to enable and disable Garbage Collector in our program:
- ✓ Importance functions in gc module
- ✓ gc.isenabled()
- ✓ gc.disable()
- ✓ gc.enable()
- ✓ Programs

18. OOPS Part 2 - Inheritance

- ✓ What is inheritance
- ✓ Conclusion of the story
- ✓ Advantage
- ✓ How to implement inheritance
- ✓ Types of Inheritance:
- ✓ Single Inheritance
- ✓ Multi-level Inheritance
- ✓ Multiple inheritance
- ✓ Parent classes can contain a method with same name
- ✓ If parent classes contain a method with same name, then which method will access by child class
- ✓ Scenarios
- ✓ Constructors in Inheritance
- ✓ If child class and super class both having constructors, then?
- ✓ Calling super class constructor from child class constructors
- ✓ Method Resolution Order (MRO)
- ✓ There are three principles followed by MRO
- ✓ Why should we understand MRO?
- ✓ Is there any predefined method to check sequence of execution of classes?
- ✓ Demo program 1 for method resolution order
- ✓ Demo Program-2 for Method Resolution Order

- ✓ super() function
- ✓ Which scenario we can use super() function in child class to call super class members?
- ✓ Calling method of a specific super class
- ✓ Different cases for super() function
- ✓ Programs

19. OOPS Part 3 – Polymorphism

- ✓ What is Polymorphism
- ✓ Duck Typing Philosophy of Python
- ✓ Overloading
- ✓ What is overloading
- ✓ Operator Overloading:
- ✓ Is python supports operator overloading?
- ✓ How operator overloading works in python?
- ✓ Magic methods
- ✓ List of operators and corresponding magic methods.
- ✓ Method Overloading
- ✓ How we can handle overloaded method requirements in Python
- ✓ Constructor Overloading
- ✓ Constructor with Default Arguments
- ✓ Constructor with Variable Number of Arguments:
- ✓ Method overriding:
- ✓ Demo Program for Method overriding
- ✓ Constructor overriding
- ✓ Calling parent class constructor from child class constructor
- ✓ Programs

20. OOPS Part 4 – abstract class, interface

- ✓ In python which things we can make as an abstract?
- ✓ There are two types of methods in-terms of implementation
- ✓ Implemented method
- ✓ Un-implemented method
- ✓ So, how to declare abstract method
- ✓ abstract method
- ✓ abstract class

- ✓ If child class missed to provide abstract methods implementation, then
- ✓ abstract class can contain concrete methods also
- ✓ Can abstract class contain more sub classes?
- ✓ On which scenario an abstract class contains more than one child classes?
- ✓ Abstract class object creation
- ✓ Different cases and scenarios for abstract class object creation
- ✓ Interface
- ✓ What is an interface?
- ✓ When should we go for ...?
    - Interface
    - Abstract class
    - Normal class or concrete class
- ✓ Double underscore (Name mangling)
- ✓ __str__(self) method
- ✓ Programs

21. Exceptional handling

- ✓ Syntax Errors
- ✓ Programmer responsible
- ✓ Runtime Errors
- ✓ Scenarios where runtime errors will occur?
- ✓ Normal flow of the execution
- ✓ Abnormal flow of the execution
- ✓ What we need to do if program terminates abnormally
- ✓ What is an Exception?
- ✓ Is it really required to handle the exceptions?
- ✓ What is the meaning of exception handling?
- ✓ Default Exception Handing in Python:
- ✓ Exception hierarchy diagram
- ✓ Programmer focus
- ✓ Handling exceptions by using try except
    - try block
    - except block
- ✓ try-except flow
- ✓ Control flow in try and except

    - 7 cases and scenarios with examples
- ✓ Printing exception information
- ✓ try with multiple except blocks
- ✓ Single except block can handle multiple exceptions
- ✓ Default except block
- ✓ finally block
- ✓ Why separate block for clean-up activities, can't we write inside try and except blocks
- ✓ Coming to the try block
- ✓ Coming to the except block
- ✓ What is the specialty of final block?
- ✓ Control flow about finally block
    - few cases and scenarios with examples
- ✓ Any situation like, finally will not execute?
- ✓ Control flow in try-except-finally
    - 6 cases and scenarios with examples
- ✓ Nested try-except-finally blocks
    - 4 cases and scenarios with examples
- ✓ else block with try-except-finally
- ✓ At what time which block
- ✓ Various possible combinations of try-except-else-finally
- ✓ Types of Exceptions
- ✓ Predefined Exceptions
- ✓ User Defined Exceptions
- ✓ User defined exceptions or Custom Exceptions
- ✓ Steps to follow to Define and Raise Customized Exceptions
- ✓ Programs

22. File IO

- ✓ What is a file
- ✓ Types of Files
- ✓ Text files
- ✓ Binary Files
- ✓ file modes and their meanings
- ✓ Opening a File
- ✓ Various properties of File Object

- ✓ Writing data to text files
- ✓ Instead of overriding the content, how to append the content to the file
- ✓ writelines(argument)
- ✓ Reading Character Data from text files
- ✓ readline() method
- ✓ readlines() method
- ✓ with keyword
- ✓ Advantage
- ✓ The seek() and tell() methods
- ✓ How to check a specific file exists or not?
- ✓ sys.exit(0)
- ✓ Program to print number of lines, words and characters present in the given file?
- ✓ Working with Binary data
- ✓ Working with csv files
- ✓ Zipping and Unzipping Files
- ✓ To create Zip file
- ✓ To perform unzip operation
- ✓ Working with Directories
- ✓ To know contents of directory
- ✓ Pickling and Unpickling of Objects
- ✓ Programs

23. Database connectivity

- ✓ Data Storage Areas
- ✓ Temporary Storage Areas
- ✓ Permanent Storage Areas
- ✓ File Systems
- ✓ Databases
- ✓ Python Database Programming
- ✓ Standard Steps for Python database Programming
- ✓ import database specific module
- ✓ Establish Connection.
- ✓ Create Cursor object
- ✓ In-built methods to execute the sql queries
- ✓ commit or rollback
- ✓ Fetch the result from the Cursor

- ✓ close the resources
- ✓ Important methods while working with Database
    - connect()
    - cursor()
    - execute()
    - executescript()
    - executemany()
    - commit()
    - rollback()
    - fetchone()
    - fetchall()
    - fetchmany(n)
    - close()
- ✓ Working with Oracle Database
- ✓ Installing cx_Oracle:
- ✓ How to Test Installation
- ✓ Expected common errors while executing programs
- ✓ Programs

24. Regular expressions

- ✓ What is Regular expression?
- ✓ When should we choose?
- ✓ The main important application areas of Regular Expressions are
- ✓ compile()
- ✓ finditer()
- ✓ On Match object we can call start(), end() and group() methodsmethods
- ✓ Character classes
- ✓ Pre-defined Character classes:
- ✓ Quantifiers
- ✓ Important functions of re module
    - match()
    - fullmatch()
    - search()
    - findall()

    - finditer()
    - sub()
    - subn()
    - split()
    - compile()
- ✓ ^ symbol
- ✓ $ symbol
- ✓ Programs

25. Multithreading

- ✓ Multi-Tasking
- ✓ Process based Multi-Tasking
- ✓ Where multi-tasking fits?
- ✓ Thread based Multi-tasking
- ✓ Where multi-tasking fits
- ✓ Areas of multi-threading
- ✓ The ways of Creating Thread in Python
- ✓ Creating a Thread without using any class
- ✓ Creating a Thread by inheriting Thread class
- ✓ Creating a Thread without inheriting Thread class
- ✓ Without multi-threading
- ✓ With multithreading
- ✓ Setting and Getting Name of a Thread:
- ✓ How to get and set name of the thread
- ✓ Thread Identification Number (ident):
- ✓ active_count()
- ✓ enumerate() function
- ✓ isAlive()
- ✓ join() method
- ✓ join(seconds) method
- ✓ Daemon Threads
- ✓ Example: Garbage Collector
- ✓ setDaemon() method
- ✓ Default Nature
- ✓ Main Thread is always Non-Daemon
- ✓ Scenarios from previous program

- ✓ Synchronization
- ✓ How to overcome data inconsistency problems?
- ✓ How to implement synchronization
    - Lock
    - RLock
    - Semaphore
- ✓ Synchronization By using Lock concept:
- ✓ acquire() method
- ✓ release() method
- ✓ Lock()
- ✓ Problem with Simple Lock
- ✓ RLock()
- ✓ Demo Program for synchronization by using RLock:
- ✓ Difference between Lock and RLock
- ✓ Synchronization by using Semaphore
- ✓ How to create Semaphore object?
- ✓ BoundedSemaphore:
- ✓ Difference between Lock and Semaphore
- ✓ Inter Thread Communication
- ✓ Interthread communication by using Event Objects
- ✓ Methods of Event class
- ✓ set()
- ✓ clear()
- ✓ isSet()
- ✓ wait() | wait(seconds)
- ✓ Interthread communication by using Condition Object
- ✓ Methods of Condition
- ✓ Interthread communication by using Queue:
- ✓ Important Methods of Queue
- ✓ Types of Queues
    - FIFO Queue
    - LIFO Queue
    - Propriety Queue
- ✓ Programs

26. Logging module

- ✓ Why Logging is required?
- ✓ What is Logging
- ✓ The main advantages of logging
- ✓ Can we use print statement for debugging?
- ✓ logging module
- ✓ How to implement Logging
- ✓ How to configure log file in over writing mode:
- ✓ How to Format log messages:
- ✓ To display only level name:
- ✓ To display levelname and message
- ✓ Add timestamp in the log messages:
- ✓ change date and time format
- ✓ Writing Python program exceptions to the log file
- ✓ Problems with root logger
- ✓ Need of Our own customized logger:
- ✓ Advanced logging Module Features
- ✓ Logger
- ✓ Steps for Advanced Logging
- ✓ Demo Program for File Handler:
- ✓ Programs

27. Data Science Introduction

- ✓ What is Data science?
- ✓ Where python is using in Data science?
- ✓ Data science programs by using python

28. IDEs explanation

- ✓ IntelliJ
- ✓ PyCharm

29. Final Project-Python

- Business understanding-Banking System
- Requirement gathering
- Data injection to oracle Database

- Identifying the implementation (Functional and OOPS Approach)
- Implementing Business logic using user defined functions (Functional Approach)
- Implementing Business logic using class (OOPS Approach)
- Integrating Exception Handling and logging levels in business logic
- Creating modules and packages
- Creating DB tables and preparing DB queries.

30. FAQs & Real Time Interview Questions