Core Python Programming

Table of **Contents:**

Program **Overview:**

Our Core Python Programming is a comprehensive course designed to provide you with a solid foundation in Python programming, object-oriented programming (OOP) concepts, and essential data structures. Whether you're new or seasoned, it equips you with essential skills to excel in Python development. Explore core Python principles, OOP's power, and master data structures to unlock your potential as a Python programmer.

Program **Features:**

- Self-Paced videos
- Live Classes & Doubt Clearing Sessions
- Industry Based Case Studies
- Assignments & Quizzes
- Project Assessment

Mode of **Delivery**

- Self-Paced Videos - **19 Hours**
- Live Lectures - **6 Hours**
- Short Learning Material - **1 Hour**
- Capstone Project/Case Study - **2 Hours**
- Assessments & Assignments - **2 Hours**

Instructor **Details**

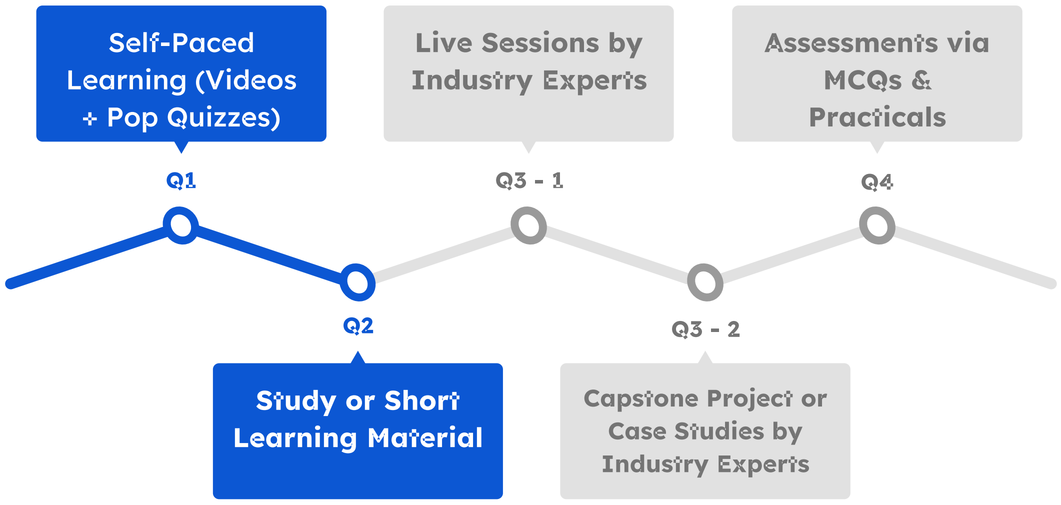
Rakhee Das
(Faculty at NMIMS)

With 15+ years in Engineering Education and EdTech, Dr. Das excels in curriculum design, teacher training, project leadership, and research (Scopus, Springer, IEEE). Passionate about technical writing and machine learning, Dr. Das holds a PhD and specializes in Python and Deep Learning.

Rocky Jagtiani
(Corporate Trainer)

With 18+ years of experience, Mr. Jagtani has trained over 18,000 professionals in core programming, Python, Data Science, ML, and AI. A skilled trainer and content developer in Databases and Data Science, Mr. Jagtani is passionate about ML/NLP and IT recruitment.

Course **Structure**


Self-Paced Learning (Videos + Pop Quizzes)
Q1
Live Sessions by Industry Experts
Q3 - 1
Assessments via MCQs & Practicals
Q4
Q2
Study or Short Learning Material
Q3 - 2
Capstone Project or Case Studies by Industry Experts

hunarho

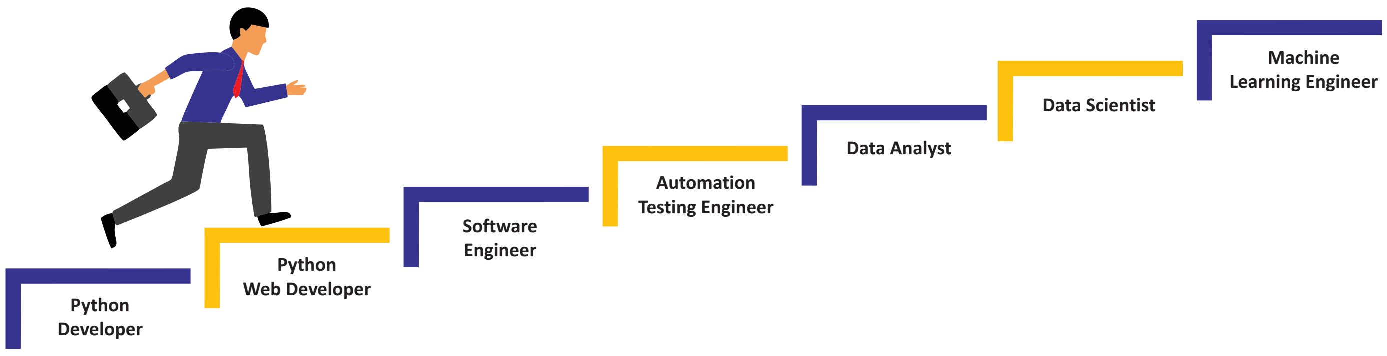
Program **Ideal For**

- Students from IT/CS Background
- Students aspiring to build a career in data analytics and data science
- Students aspiring to build a career in Python development
- Anyone wanting to build work with web development or automation

Learning **Outcomes**

- Write well-structured Python programs using essential language constructs.
- Utilize functions, modules, and packages for code organization and reusability.
- Interact with files for data persistence.
- Implement robust error-handling mechanisms.
- Design and develop object-oriented programs using core OOP principles.
- Apply various data structures to solve problems efficiently.

Your **Pathway To Success!**



Course **Details**

Week 1 :

1. **Overview of Python:** print() statement, how to give comments, what's the use """ or ''', declaring variables of different types, two ways to format String -> {} and .format(), different types of escape sequences, order of MATH operations -> PEMDAS, use input() to get input, use cmd line arguments i.e. argv to get input, define and call functions.
2. **Data-types & Operators in Python:** Python Numbers : (long)int, float, complex no. , Python String, Python Lists, Python Tuples, Python Dictionary, Types of Operators - Arithmetic Operators, Comparison (Relational) Operators, Assignment Operators, Logical Operators, Bitwise Operators, Membership Operators, Identity Operators; Decision making, Loops, Iterator & Generator.
3. **Project:** Web Scrapping Tool and Simplified Hotel Bill
4. **Build-in functions of Python Data-types:** Number build-in functions, String build-in functions, List build-in functions, Tuple build-in functions, zip() and use of zip(), Properties of Dictionary Keys, dictionary build-in functions, Difference between shallow and deep copy, Date & time functions.
5. **Functions, Modules & Packages:** Syntax of user-defined function, Pass by Reference vs. Pass by Value, Function Arguments - a> Required arguments, b> Keyword arguments, c> Default arguments, d> Variable-length arguments; use pointer notation(*): to accept many arguments, Anonymous Functions: use lambda keyword, not, def keyword, Scope of Variables - a> Global variables, b> Local variables; Concept of a module, from...import <name> Statement, from...import * Statement, Executing Modules as Scripts, Locating Modules, dir() Function, globals() and locals(), Packages, don't confuse between importing from a package w.r.t importing from a module, install packages
6. **Reading/Writing from files:** Opening and Closing Files, Reading from files, Writing to files, Check current position of file pointer, Copy from one file to another, Renaming file, Deleting file/s, Functions to operate with directories - a> mkdir() Method, b> chdir() Method, c> getcwd() Method, d> rmdir() Method.
7. **Project:** Simple Invoicing Program
8. **Assertions & Exceptions: Assertions:** when and why to use? About Exceptions, except Clause with no Exceptions, except Clause with multiple Exceptions, try-finally Clause, Argument of an Exception, Raising an Exception, User-Defined Exceptions, Coding Exercises.
9. **Live Session:** Doubt clearing session the learnings from the self paced videos.
10. **Core Python Programming Assessment**

hunarho

Week 2 :

1. **Introduction to Object Oriented Programming:** OOPs concepts - Attributes, Behavior, Class, Object. Some simple class-based program examples.
2. **Python Objects and Classes:** What is exactly instantiation?, Defining a Class in Python, docstring (__doc__), local namespace concept. Constructors in Python (__init__()), Some sample programs to explain parameterised constructors. Deleting Attributes and Objects (del statement) Programming Exercises.
3. **Python Inheritance:** What is Inheritance and Why is it needed?, Python Inheritance Syntax, Example of Inheritance in Python, Method Overriding in Python. Programming Exercises. Python Multiple Inheritance, Python Multilevel Inheritance, Method Resolution Order in Python, Programming Exercises.
4. **Python Operator Overloading:** Benefit of Overloading, List of Python operators that can be overloaded, Python Special Functions (begin with double underscore __). Examples - Overloading the + Operator, Overloading Comparison Operators. Programming Exercises.
5. **Python Iterators:** What is the purpose of Iterators, __iter__() and __next__(), Explain practically line by line code for "Iterating through an Iterator", Working of for loop for Iterators, Building Custom Iterators, Python Infinite Iterators. Programming Exercises.
6. **Python Generators:** How are Python Generators different from iterators and normal functions?, Create Generators in Python, Differences between Generator function and Normal function, Python Generators with a Loop. Python Generator Expression. Use of Python Generators - Easy to Implement, Memory Efficient and Represent Infinite Stream. Pipelining Generators. Programming Exercises.
7. **Python Closures:** Nonlocal variable in a nested function, Defining a Closure Function. When do we have closures? When to use closures? Introduction to Python Decorators. Simple Programming Exercises.
8. **Python Decorators:** Prerequisites for learning decorators, callable methods, __call__(), and Example programs. Decorating Functions with Parameters, Chaining Decorators in Python. Simple Programming Exercises.
9. **Python @property decorator:** Class Without Getters and Setters, Using Getters and Setters, explain property Class, explain @property Decorator with example codes. Simple Programming Exercises.
10. **Live Session:** Doubt clearing session the learnings from the self-paced videos.
11. **Object-Oriented Programming Final Assessment**

Week 3 :

1. **Need for Data Structures:** Basic Program Elements, Control Statements, Strings and Their Operations, Built-In Python Collections and Their Operations, Catching Exceptions, Files and Their Operations, Creating New Classes.
2. **An Overview of Collections:** Collection Types (will cover four general categories of collections — linear, hierarchical, graph, and unordered), Operations on Collections, Iterators and Higher-Order Functions, Implementations of Collections
3. **Searching, Sorting, and Complexity Analysis:** Measuring the Efficiency of Algorithms, Complexity Analysis, Search Algorithms, Basic Sort Algorithms, Analysis Summary
4. **Arrays and Linked Structures:** The Array Data Structure, Operations on Arrays, Two-Dimensional Arrays (Grids), Linked Structures, Operations on Singly Linked Structures.
5. **Live Session:** Doubt clearing session the learnings from the self paced videos.

Week 4 :

1. **Recalling important OOP concepts - Interfaces, Implementations, and Polymorphism:** Developing an Interface, Constructors and Implementing Classes, Developing an Array-Based Implementation, Developing a Link-Based Implementation
2. **Recalling another OOP concept - Inheritance and Abstract Classes:** Using Inheritance to Customize an Existing Class, Using Abstract Classes to Eliminate Redundant Code, An Abstract Class for All Collections.
3. **Stacks & Its Applications:** Overview of Stacks, Using a Stack, Three Applications of Stacks - Evaluating Arithmetic Expressions, Backtracking algorithm, Memory Management, implementations of Stack - using Arrays as ArrayStack and using linked Structure as LinkedStack.
4. **Application of Linked List:** Recalling Lists, Two Applications of Lists - Heap-Storage Management, Organization of Files on a Disk, List Implementations - array-based and linked structure-based
5. **Live Session (1 hour):** Doubt clearing session the learnings from the self-paced videos.

Week 5 :

1. **Trees and its Applications:** An Overview of Trees, Why Use a Tree?, The Shape of Binary Trees, Binary Tree Traversals, Three Common Applications of Binary Trees - Heaps, Binary Search Trees, Expression Trees. Developing a Binary Search Tree implementation.
2. **Graph & its Applications:** Why Use Graphs?, Graph Terminology, Representations of Graphs, Graph Traversals, Applications of Graph, Developing a Graph Collection
3. **Live Session:** Doubt clearing session the learnings from the self paced videos.
4. **Python Data Structures Assessment**

Week 6 :

Capstone Project: Building a Simple Address Book

This project aims to create a basic address book application using core Python programming, data structures, and object-oriented programming (OOP) principles.

x

Scan this QR code with your smartphone and discover more information about the program

FOR ENQUIRIES:

CALL +91 73040 02635

www.hunarho.com

support@hunarho.com

hunarho