Core Python Programming

Python is a general-purpose programming language, which means that it can be used for a wide range of applications. It has libraries and modules for various domains, including web development, data science, machine learning, artificial intelligence, game development, scientific computing, and more. This versatility makes Python a popular choice among developers, as it allows them to work on a variety of projects.

This course is focused on:

1. Syntax and basic programming concepts
2. Variables and data types
3. Control structures
4. Functions and modules
5. Libraries and modules
6. File handling
7. Exception handling
8. Object-oriented programming (OOP)
9. Best practices and coding standards

Course Objectives:

1. Introduction to Programming: For beginners, the course will introduce the basics of programming, including data types, variables, control structures, and functions.
2. Object-Oriented Programming: For intermediate level courses, the focus is on object-oriented programming concepts such as inheritance, polymorphism, and encapsulation.
3. Data Structures and Algorithms: Python is well suited for data manipulation, and students will learn about data structures such as lists, tuples, and dictionaries, as well as algorithms like searching and sorting.
4. Web Development: Python can be used to build web applications, and courses may include topics such as Django, Flask, and web scraping.
5. Data Science: Python is widely used in data science, and courses may cover topics such as NumPy, Pandas, and data visualization.
6. Machine Learning: Python is a popular choice for machine learning, and courses may cover topics such as scikit-learn, TensorFlow, and Keras.
7. Artificial Intelligence: Python is widely used in artificial intelligence, and courses may cover topics such as natural language processing, computer vision, and deep learning.

Target Audience:

1. Beginners: Python is an excellent programming language for beginners due to its simplicity and readability. A Python course for beginners may target high school students, college students, or adults with no programming experience.
2. Intermediate level: Intermediate level courses may target students who have some experience with programming but want to improve their skills in Python specifically. They may also target professionals in fields such as data analysis or web development who want to expand their skill set.
3. Advanced: Advanced Python courses may target professionals who work in fields such as data science, machine learning, or artificial intelligence. These professionals may already have experience with Python but want to take their skills to the next level.
4. Professionals looking for a career change: Python is a popular programming language used in various industries, and some professionals may want to switch careers or add Python skills to their resume to increase their job prospects. Python courses targeted towards professionals looking for a career change may cover a broad range of topics and focus on real-world applications.

Course Pre-Requisites:

A Python course for beginners typically has no prerequisites, as it is designed for students with no programming experience.

Students taking an intermediate level Python course should have a basic understanding of programming concepts such as data types, variables, and control structures. They should also be familiar with object-oriented programming concepts such as classes and methods.

Course Duration:

- 60 Hours
- 24 Classes

Details Course Outlines

Hour 1-2: Introduction to Programming

- Why programming is important and its applications
- Different programming languages and their uses
- Installing Python and setting up the environment
- Introduction to basic programming concepts

Hour 3-7: Variables, Data Types, and Operators

- Variables and how they work
- Common data types (e.g. integers, floats, strings, Booleans)
- Operators: arithmetic, comparison, logical
- Understanding precedence and order of operations
- Using built-in Python functions

Hour 8-10: Conditional Statements

- If/else statements
- Using comparison operators in if/else statements
- Nested if/else statements
- Case studies and examples

Hour 11-13: Loops

- Introduction to loops
- For loops
- While loops
- Nested loops
- Using loops to solve problems

Hour 14-16: Functions

- What are functions and how to use them
- Defining functions
- Using parameters and arguments
- Returning values from functions
- Using built-in functions

Hour 17-20: Lists

- Introduction to lists
- Creating and accessing lists
- Slicing and indexing
- Modifying lists (adding, removing, and replacing elements)
- Common list methods

Hour 21-23: Tuples and Sets

- Introduction to tuples and sets
- Creating and accessing tuples and sets
- Modifying tuples and sets
- Common tuple and set methods

Hour 24-27: Dictionaries

- Introduction to dictionaries
- Creating and accessing dictionaries
- Modifying dictionaries
- Common dictionary methods

Hour 28-30: String Manipulation

- Introduction to strings
- String methods
- Concatenation and formatting strings
- String slicing and indexing
- Regular expressions

Hour 31-33: File Input/Output

- Opening, reading, and writing to files
- Different file modes
- Using with statements
- Handling exceptions when working with files
- Case studies and examples

Hour 34-36: Object-Oriented Programming Basics

- Introduction to object-oriented programming
- Creating classes and objects
- Encapsulation and information hiding
- Using constructors and destructors
- Case studies and examples

Hour 37-40: Inheritance and Polymorphism

- Inheritance and why it's useful
- Overriding methods
- Using super()
- Polymorphism and its benefits
- Case studies and examples

Hour 41-43: Exception Handling

- Introduction to exception handling
- Catching exceptions
- Multiple except blocks
- Handling multiple exceptions
- Raising exceptions

Hour 44-46: Modules and Packages

- Importing modules
- Creating and importing your own modules
- Understanding packages
- Installing and using external packages
- Case studies and examples

Hour 47-50: Python Libraries

- Introduction to popular Python libraries
- Using NumPy for scientific computing
- Using Pandas for data analysis
- Using Matplotlib for data visualization
- Case studies and examples

Hour 51-54: Web Development

- Introduction to web development
- HTML, CSS, and JavaScript basics
- Introduction to Flask or Django web framework
- RESTful APIs and HTTP requests
- Parsing JSON and XML data

Hour 55-57: Debugging and Testing

- Introduction to debugging
- Using print statements and debuggers
- Handling errors and exceptions
- Introduction to testing
- Writing and running test cases

Hour 58-60: Final Project

- Project Name: Student Gradebook
- Project Description:
  – Create a student gradebook application using Python that allows a teacher to record and track grades for their students.