Python for Data Science and Data Analytics form basic to advance

---

Lecture 1: Introduction to Python

- What is Python? Key features and applications
- Installing Python and setting up the environment
- Python syntax basics:
    - Writing and running Python scripts
    - Comments and indentation rules
- Input and output in Python
- First program: "Hello, World!"

Lecture 2: Variables, Data Types, and Operators

- Variables and their scope
- Common data types: integers, floats, strings, booleans
- Type conversion (casting)
- Operators:
    - Arithmetic, comparison, logical, assignment, and bitwise operators
- Operator precedence

Lecture 3: Control Flow in Python

- Conditional statements:
    - `if`, `elif`, `else`
- Loops in Python:
    - `for` loops and `while` loops
    - Break, continue, and pass statements
- Writing simple programs with decision-making

Lecture 4: Functions in Python

- Defining and calling functions
- Function arguments: positional, keyword, and default arguments

- Return values
- Scope of variables: local and global scope
- Recursion basics

Lecture 5: Working with Strings

- String creation and manipulation
- String slicing and indexing
- String methods: `find()`, `replace()`, `split()`, `join()`, etc.
- Escape characters and formatting strings
- String comparison

Lecture 6: Data Structures: Lists and Tuples

- Creating and accessing lists
- List methods: `append()`, `remove()`, `pop()`, etc.
- Nested lists
- Tuples: definition and use cases
- List comprehension basics

Lecture 7: Dictionaries and Sets

- Introduction to dictionaries: key-value pairs
- Adding, updating, and deleting dictionary items
- Iterating over dictionaries
- Introduction to sets and their properties
- Set operations: union, intersection, difference

Lecture 8: File Handling in Python

- Reading from and writing to files
- Modes: `r`, `w`, `a`, `r+`
- Working with file pointers
- Exception handling in file operations
- Writing a program to process simple text data

Lecture 9: Error Handling and Debugging

- Types of errors: syntax errors, runtime errors, logical errors
- Using `try`, `except`, `else`, and `finally` blocks
- Raising exceptions manually
- Debugging techniques and tools (basic)
- Writing robust Python programs

Lecture 10: Advanced Concepts and Project

- Understanding and using classes and objects (OOP basics)
- Defining classes, attributes, and methods
- Using inheritance and polymorphism
- Final project: Develop a program using core Python concepts
    - Example: Create a text-based menu-driven system (e.g., student management or calculator).

This plan focuses on building a strong foundation in Python's core functionalities, ensuring learners can confidently write standalone Python programs without relying on external libraries.