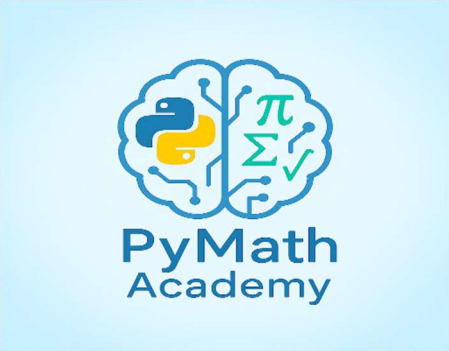
Python Programming Roadmap



Kumar's Classes

Level 1: Core Python (Beginner)

This foundational level is for absolute beginners. Covers the core fundamentals. No prior coding knowledge needed.

Estimated Time to Complete: 25-30 Hours

- **1. Introduction to Python** (2 Hours)
    - History, features, and modern applications of Python.
    - Installing Python and setting up an IDE (e.g., Anaconda, Colab, Jupyter Notebook etc).
    - Running your first scripts (using the interactive shell and .py files).
    - Understanding **PEP 8** style guidelines for writing clean code.
- **2. Basic Syntax & Data Types** (5 Hours)
    - **Variables**, keywords, and identifiers.
    - Using comments and understanding Python's indentation rules.
    - Numeric types: int, float, complex.
    - **Strings**: Creation, indexing, slicing, and common methods.

    - The **Boolean** type (True/False) and truth values.
    - **Type Casting**: Converting between data types (e.g., int() to str()).
- **3. Operators** (2 Hours)
    - **Arithmetic** (+, -, *, /), **assignment** (=, +=), and **comparison** (==, >) operators.
    - **Logical** (and, or), **identity** (is), and **membership** (in) operators.
- **4. Core Data Structures** (6 Hours)
    - **Lists**: Creating, indexing, slicing, and using list methods.
    - **Tuples**: Understanding immutability and tuple methods.
    - **Sets**: Working with unique items and performing set operations (union, intersection).
    - **Dictionaries**: Storing and accessing data using key-value pairs.
- **5. Control Flow** (5 Hours)
    - Conditional logic with if, elif, and else statements.
    - Looping with for (over sequences) and while (based on a condition).
    - Controlling loops with break, continue, and pass.
- **6. Functions** (6 Hours)
    - Defining and calling your own reusable functions.
    - Using parameters: positional, keyword, and default arguments.
    - Handling a variable number of arguments with *args and **kwargs.
    - Understanding return values and creating simple **lambda** functions.
    - **Variable Scope**: Local vs. Global variables.
- **7. File I/O** (2 Hours)
    - Reading from and writing to text (.txt) and CSV (.csv) files.
    - Using the with statement for safe and automatic file handling.

Level 2: Intermediate Python

This level focuses on writing efficiently, modular, and professional-grade code. You'll move beyond simple scripts to build robust and scalable programs using Object-Oriented principles.

Estimated Time to Complete: 25-30 Hours

- **1. Object-Oriented Programming (OOP)** (15 Hours)
    - Understanding **Classes**, **Objects**, attributes, and methods.
    - The __init__ constructor for initializing objects.
    - Mastering the four pillars of OOP:
        - **Inheritance**
        - **Encapsulation**
        - **Polymorphism**
        - **Abstraction**
    - Using class methods and static methods.
- **2. Modules & Packages** (5 Hours)
    - Importing modules from the **Python Standard Library**.
    - Creating your own custom modules and packages.
    - Installing and managing third-party packages with **pip**.
- **3. Exception Handling** (4 Hours)
    - Gracefully handling errors with try, except, else, and finally.
    - Catching multiple specific exceptions.
    - Raising your own custom exceptions.
- **4. Advanced Data Structures & Comprehensions** (5 Hours)
    - Writing concise **List, Dictionary, and Set Comprehensions**.
    - Using the collections module: Counter, defaultdict, deque.
- **5. Virtual Environments** (2 Hours)

- Understanding why code isolation is crucial.
- Creating and managing virtual environments with venv.

Level 3: Advanced Python & Specialization

This level covers high-performance programming concepts and provides a launchpad into specialized fields like web development and data science.

Estimated Time to Complete: 25-30 Hours

- **1. Advanced Programming Concepts** (10 Hours)
  - **Iterators & Generators**: Understanding the iterator protocol (__iter__, __next__) and creating memory-efficient data streams with the yield keyword.
  - **Generator Expressions**: A high-performance, memory-efficient alternative to list comprehensions.
- **2. Specialization: Web Development** (30 Hours)
  - **Django**: A high-level framework for rapid development. Learn Models, Views, Templates, and its powerful ORM.
  - **Flask**: A lightweight micro-framework for smaller applications and APIs. Learn routing, templates, and request handling.
  - **REST APIs**: Build APIs with Django REST Framework or Flask-RESTful.
- **3. Specialization: Data Science & Machine Learning** (40 Hours)
  - **NumPy**: The fundamental package for scientific computing, focusing on ndarrays and vectorized operations.
  - **Pandas**: The ultimate tool for data manipulation and analysis using **DataFrames** and **Series**.

- **Matplotlib & Seaborn**: Creating a wide range of static, animated, and interactive visualizations.
- **Scikit-learn**: Applying machine learning models for regression, classification, and clustering.
- **TensorFlow / PyTorch**: An introduction to building and training deep learning models.