Python Developer Learning Path

Core Python (Fundamentals)

- Syntax and Semantics
- Data Types - int, float, str, list, tuple, dict, set
- Variables and Constants
- Control Structures: if, else, elif; for and while loops
- Functions: defining, arguments, *args, **kwargs, lambda
- Error Handling: try, except, finally, raise
- Modules and Packages
- File Handling: reading/writing files
- List Comprehensions
- Pythonic Idioms: zip(), enumerate(), unpacking, etc.

Intermediate Python

- Object-Oriented Programming: classes, inheritance, polymorphism, encapsulation
- Decorators and Generators
- Context Managers (with)
- Virtual Environments
- pip and PyPI usage
- Basic Unit Testing: unittest, pytest
- Regular Expressions (re module)

Commonly Used Libraries

General Purpose:

- os, sys, math, random, datetime, logging

Web Development:

- Flask or Django, requests, jinja2

Data Science & ML:

Python Developer Learning Path

- numpy, pandas, matplotlib, seaborn, scikit-learn, tensorflow or pytorch

Scripting/Automation:

- argparse, subprocess, shutil, pathlib, selenium, pyautogui, schedule

Web and APIs

- HTTP Basics
- Building REST APIs with Flask or FastAPI
- Consuming APIs using requests
- JSON and XML Parsing

Testing and Debugging

- Testing: unittest, pytest, doctest
- Debugging: pdb
- Code linters: flake8, pylint

Advanced Topics (Optional but Useful)

- Multithreading and Multiprocessing
- Async programming (asyncio)
- Design Patterns in Python
- Memory Management and Garbage Collection
- C extensions & performance tuning

Practical Skills

- Version Control: Git & GitHub
- Project Structure and Packaging
- Writing Documentation
- Using IDEs (VSCode, PyCharm)
- CI/CD Basics (e.g., GitHub Actions)

Python Developer Learning Path

Build Projects

Start Small:

- Calculators, To-do Lists

Intermediate:

- REST APIs, Web Scrapers

Advanced:

- Automation Bots, Dashboards, Full-stack Apps