Fast-Paced Python Learning Path (6 Weeks)

Yaswanth

May 2025

Overview

This 6-week learning path will take you from the fundamentals of Python to more advanced concepts. Each week includes key topics, coding challenges, and mini-projects designed to reinforce the material.

Week 1: Python Fundamentals

Topics:

- Syntax, Variables, Data Types
- Input/Output
- Conditionals (if/else)
- Loops (for, while)
- Functions
- Basic Error Handling

Challenges:

- **FizzBuzz**: Print numbers from 1 to 100, but replace multiples of 3 with "Fizz" and multiples of 5 with "Buzz".
- **Prime Number Checker**: Write a function that checks if a given number is prime.
- **Basic Calculator**: Create a CLI calculator that can add, subtract, multiply, and divide.
- **Guess the Number**: Create a guessing game that generates a random number, and the player has to guess it.

Mini Project:

- **CLI To-Do List Manager**: Create a command-line to-do list where users can add, remove, and view tasks.

Week 2: Python Functions & Advanced Data Structures

Topics:

- Default Arguments, Variable-Length Arguments (`*args`, `**kwargs`)
- Function Annotations
- Recursion (Introduction)
- List Comprehensions
- Tuples, Sets, and Dictionaries
- String Manipulation

Challenges:

- **Word Frequency Counter**: Create a function that takes a string and counts how many times each word appears.
- **List/Dict Manipulations**: Given a list of integers, return a dictionary of odd and even numbers.
- **Palindrome Checker**: Write a function to check if a word is a palindrome.

Mini Project:

- **Text-Based Contact Book**: Create a contact book that stores names, phone numbers, and email addresses in memory, and allows CRUD operations.

Week 3: Intermediate OOP & Pythonic Techniques

Topics:

- Classes and Objects
- `__init__`, `__str__`, `__repr__`
- Inheritance and Polymorphism
- Method Resolution Order (MRO)
- Abstract Classes and Method Overriding
- Design Patterns (Singleton, Factory, etc.)

Challenges:

- **Bank Account Class**: Create a class that models a bank account with deposit, withdraw, and balance methods.
- **Inventory Management**: Design a class to manage inventory, allowing for stock increase and decrease.
- **Shape Area Calculator**: Implement a base class for a Shape, then subclass for Circle, Rectangle, and Triangle.

Mini Project:

- **Simple Student Grading System (OOP)**: Create a class to represent students and their grades. Implement a method to calculate the average grade and the highest grade.

Week 4: Python Libraries, Regex, and File Handling

Topics:

- Regular Expressions (`re` module)
- Data Serialization (`pickle`, `json`, `csv`)
- Working with Files
- Multithreading vs Multiprocessing (Intro to concurrency)
- Introduction to `pandas` and `numpy` (for quick data handling)

Challenges:

- **File Organizer**: Write a script that organizes files in a directory by file type.
- **Date/Time Formatter**: Create a function that formats dates in various styles using `datetime`.
- **Generator-based Fibonacci**: Write a generator function that yields the Fibonacci sequence.

Mini Project:

- **File System Organizer CLI Tool**: Build a CLI tool that organizes files by extensions, creating folders for each file type.

Week 5: Web Scraping, Automation & API Handling

Topics:

- Web Scraping with `BeautifulSoup`
- API Handling (`requests`, `json`)
- Automation with `selenium`
- Sending Emails Programmatically (`smtplib`)
- Web Scraping Best Practices (Rate-Limiting, Proxies, etc.)

Challenges:

- **Currency Converter**: Build an API-driven currency converter.
- **Weather App**: Build an app that retrieves weather data for a given city.
- **Scrape Book Titles**: Write a scraper that extracts book titles from an online book store.

Mini Project:

- **News Aggregator CLI**: Build a CLI tool that pulls news from multiple sources and outputs it to the terminal.

Week 6: Capstone Project, Cloud, and Advanced Topics

Topics:

- Unit Testing with `unittest` or `pytest`
- Code Style (PEP8) and Linters
- Cloud Deployment (AWS, Heroku)
- Optimizing Code (Profiling, Time Complexity)
- Advanced Python Data Structures (Heaps, Tries)

Challenges:

- **Logging Decorator**: Create a decorator that logs function calls and results.
- **Async Function Simulation**: Implement a simple async function with `asyncio` to simulate I/O-bound tasks.
- **Unit Tests**: Write unit tests for any of the previous projects.

Capstone Project Options:

- **Expense Tracker with File or SQLite**: Build a personal finance tracker that stores expenses and income in a file or database.
- **Web API Data Dashboard (Console-based)**: Build a CLI tool that retrieves and displays API data (e.g., cryptocurrency prices).
- **Quiz Game with Scoring & Persistence**: Build a CLI-based quiz game that stores high scores and previous results.

Resources for Further Learning

- **Python Documentation**: `https://docs.python.org/3/`
- **Real Python**: `https://realpython.com/`
- **LeetCode (for algorithm challenges)**: `https://leetcode.com/`