Complete Python Developer Roadmap - 30 Weeks

This comprehensive roadmap is designed to take you from beginner to professional Python developer in 30 weeks, assuming 10-15 hours of study per week. Each module builds on previous knowledge and includes recommended projects to reinforce your learning.

MODULE 1: PYTHON FUNDAMENTALS (Weeks 1-4)

Week 1: Python Basics and Setup

- **Day 1-2**: Python installation & environment setup
  - Installing Python & choosing an IDE (VSCode, PyCharm)
  - Setting up virtual environments (venv, pipenv, conda)
  - Package management with pip
- **Day 3-4**: Basic syntax and data types
  - Variables, comments, and basic operations
  - Numbers, strings, booleans
  - Type conversion and checking
- **Day 5-7**: Control flow
  - Conditional statements (if, elif, else)
  - Loops (for, while, nested loops)
  - Loop control (break, continue, pass)
  - **Weekend Project**: Simple calculator program

Week 2: Data Structures

- **Day 1-2**: Lists and tuples
  - Creating, accessing, and modifying
  - List comprehensions
  - Slicing operations
- **Day 3-4**: Dictionaries and sets
  - Dictionary creation and manipulation
  - Dictionary comprehensions
  - Set operations and methods
- **Day 5-7**: Collection operations and methods

- Advanced operations on all data structures
- Collections module (Counter, defaultdict, namedtuple)
- **Weekend Project**: Contact book application with CRUD operations

Week 3: Functions and Modules

- **Day 1-2**: Function basics
  - Defining and calling functions
  - Parameters and return values
  - Default and keyword arguments
- **Day 3-4**: Advanced function concepts
  - Variable scope and closures
  - *args and **kwargs
  - Lambda functions
- **Day 5-7**: Modules and packages
  - Creating and importing modules
  - Standard library modules
  - Creating packages
  - **Weekend Project**: Simple task management tool with multiple modules

Week 4: File Handling and Error Handling

- **Day 1-2**: File I/O operations
  - Reading and writing text files
  - Working with CSV and JSON
  - Context managers (with statement)
- **Day 3-5**: Error handling
  - Try-except blocks
  - Handling multiple exceptions
  - Finally and else clauses
  - Custom exceptions
- **Day 6-7**: Review and consolidation
  - Practice exercises covering weeks 1-4
  - **Weekend Project**: Log analyzer tool that processes log files and handles exceptions

MODULE 2: INTERMEDIATE PYTHON (Weeks 5-10)

Week 5: Object-Oriented Programming I

- **Day 1-2**: Classes and objects
  - Creating classes and instances
  - Attributes and methods
  - Constructors (**init**)
- **Day 3-4**: OOP principles
  - Encapsulation and data hiding
  - Inheritance basics
  - Method overriding
- **Day 5-7**: Special methods
  - Magic/dunder methods
  - String representation (**str**, **repr**)
  - Operator overloading
  - **Weekend Project**: Bank account management system with different account types

Week 6: Object-Oriented Programming II

- **Day 1-2**: Advanced inheritance
  - Multiple inheritance
  - Method resolution order
  - Abstract base classes
- **Day 3-4**: Advanced OOP concepts
  - Composition vs inheritance
  - Mixins and interfaces
  - Static and class methods
- **Day 5-7**: Design patterns in Python
  - Singleton, Factory, and Observer patterns
  - Property decorators
  - **Weekend Project**: E-commerce system with products, cart, and checkout features

Week 7: Functional Programming

- **Day 1-2**: Functional programming concepts

    - Pure functions
    - First-class functions
    - Higher-order functions
- **Day 3-4**: Built-in functions
    - map, filter, reduce
    - all, any, zip, enumerate
    - sorted with key functions
- **Day 5-7**: Iterators and generators
    - Creating iterators
    - Generator functions and expressions
    - yield and next()
    - **Weekend Project**: Data processing pipeline using functional programming

Week 8: Regular Expressions and Text Processing

- **Day 1-3**: Regular expressions basics
    - Pattern matching
    - Character classes and quantifiers
    - Groups and capturing
- **Day 4-5**: Advanced regular expressions
    - Lookahead and lookbehind
    - re module functions (search, match, findall, sub)
    - Regex flags
- **Day 6-7**: Text processing
    - String manipulation techniques
    - Text parsing strategies
    - **Weekend Project**: Data extraction tool from unstructured text files

Week 9: Advanced Data Handling

- **Day 1-2**: Working with dates and times
    - datetime module
    - Time zones with pytz
    - Time deltas and arithmetic

- **Day 3-4**: Advanced file formats
  - Working with Excel files (openpyxl)
  - XML processing
  - PDF extraction and creation
- **Day 5-7**: Binary data handling
  - Bytes and bytearrays
  - Struct module
  - Working with binary files
  - **Weekend Project**: Event scheduler with recurring events and timezone support

Week 10: Debugging, Testing, and Documentation

- **Day 1-2**: Debugging techniques
  - Using pdb debugger
  - Logging with the logging module
  - Traceback analysis
- **Day 3-5**: Testing principles
  - Unit testing with unittest
  - pytest framework
  - Test-driven development
- **Day 6-7**: Documentation
  - Docstrings and comments
  - Type hints with mypy
  - Documentation generation with Sphinx
  - **Weekend Project**: Well-tested library with complete documentation

MODULE 3: ADVANCED PYTHON CONCEPTS (Weeks 11-16)

Week 11: Concurrency with Threading and Multiprocessing

- **Day 1-2**: Threading basics
  - Creating and managing threads
  - Thread synchronization
  - Thread communication
- **Day 3-4**: Multiprocessing

    - Process-based parallelism
    - Inter-process communication
    - Process pools
- **Day 5-7**: Advanced concurrency patterns
    - Locks, semaphores, and barriers
    - Thread and process safety
    - **Weekend Project**: Web scraper using multiple threads/processes

Week 12: Asynchronous Programming

- **Day 1-2**: Async concepts
    - Event loops
    - Callbacks and futures
- **Day 3-5**: asyncio framework
    - Coroutines with async/await
    - Tasks and futures
    - Asynchronous context managers
- **Day 6-7**: Advanced asyncio
    - Synchronization primitives
    - Integration with threading
    - **Weekend Project**: Asynchronous API client

Week 13: Networking and Sockets

- **Day 1-2**: Network programming basics
    - TCP/IP concepts
    - Socket programming
- **Day 3-4**: Client-server architecture
    - Building TCP clients and servers
    - UDP programming
- **Day 5-7**: Higher-level networking
    - HTTP client libraries (requests)
    - WebSockets
    - **Weekend Project**: Chat application with client-server architecture

Week 14: Web Development with Flask

- **Day 1-2**: Flask basics
  - Routes and views
  - Templates with Jinja2
  - Request and response objects
- **Day 3-4**: Flask extensions
  - Forms with Flask-WTF
  - Database with Flask-SQLAlchemy
  - Authentication with Flask-Login
- **Day 5-7**: RESTful API development
  - API design principles
  - Serialization and marshalling
  - API documentation
  - **Weekend Project**: RESTful API for a resource management system

Week 15: Database Interaction

- **Day 1-2**: SQL databases
  - SQLite and SQLAlchemy core
  - Database schema design
  - CRUD operations
- **Day 3-4**: ORM concepts
  - SQLAlchemy ORM
  - Relationship mapping
  - Migrations with Alembic
- **Day 5-7**: NoSQL databases
  - MongoDB with PyMongo
  - Redis with redis-py
  - **Weekend Project**: Full-stack application with database backend

Week 16: Data Analysis and Visualization

- **Day 1-2**: NumPy basics
  - Arrays and operations

    - Broadcasting
    - Universal functions
- **Day 3-4**: Pandas for data analysis
    - Series and DataFrames
    - Data cleaning and transformation
    - Data aggregation
- **Day 5-7**: Data visualization
    - Matplotlib basics
    - Seaborn for statistical visualization
    - Interactive plots with Plotly
    - **Weekend Project**: Data analysis dashboard with visualization

MODULE 4: MODERN PYTHON DEVELOPMENT (Weeks 17-23)

Week 17: Advanced Web Development with Django

- **Day 1-2**: Django fundamentals
    - Project structure
    - Views and URL patterns
    - Django ORM
- **Day 3-4**: Django apps and templates
    - Template language
    - Static files
    - Class-based views
- **Day 5-7**: Django forms and admin
    - Forms and ModelForms
    - Custom admin interfaces
    - **Weekend Project**: Blog system with multiple user types

Week 18: API Development and GraphQL

- **Day 1-2**: Django REST Framework
    - Serializers
    - ViewSets and routers
    - Authentication and permissions

- **Day 3-5**: GraphQL with Python
  - Schema definition
  - Queries and mutations
  - Strawberry or Graphene implementation
- **Day 6-7**: API best practices
  - Security considerations
  - Rate limiting and caching
  - Versioning
  - **Weekend Project**: GraphQL API for a product catalog

Week 19: DevOps and Deployment

- **Day 1-2**: Version control with Git
  - Advanced Git workflows
  - Collaborative development
  - CI/CD concepts
- **Day 3-4**: Containerization
  - Docker basics
  - Dockerizing Python applications
  - Docker Compose
- **Day 5-7**: Cloud deployment
  - Deploying to AWS/Azure/GCP
  - Serverless Python with AWS Lambda
  - **Weekend Project**: Containerized application with CI/CD pipeline

Week 20: Code Quality and Performance

- **Day 1-2**: Code linting and formatting
  - PEP 8 and style guides
  - Tools: flake8, black, isort
- **Day 3-4**: Static type checking
  - Type annotations
  - mypy and pyright
  - Generic types

- **Day 5-7**: Performance optimization
  - Profiling with cProfile
  - Memory optimization
  - Algorithmic improvements
  - **Weekend Project**: Optimizing and refactoring an existing application

Week 21: Security Best Practices

- **Day 1-2**: Security fundamentals
  - Common vulnerabilities
  - Authentication and authorization
  - Input validation
- **Day 3-4**: Cryptography with Python
  - Hashing and encryption
  - Digital signatures
  - Secure storage of credentials
- **Day 5-7**: Security testing
  - Penetration testing tools
  - Vulnerability scanning
  - Security code review
  - **Weekend Project**: Security audit and hardening of a web application

Week 22: Advanced Testing and Quality Assurance

- **Day 1-2**: Advanced testing techniques
  - Integration testing
  - System testing
  - Property-based testing with Hypothesis
- **Day 3-4**: Mocking and test doubles
  - unittest.mock
  - Dependency injection
  - Test fixtures
- **Day 5-7**: Test automation
  - Continuous integration

- Test coverage analysis
- Mutation testing
- **Weekend Project**: Complete test suite for a critical system

Week 23: Machine Learning Basics

- **Day 1-2**: ML fundamentals
  - Supervised vs. unsupervised learning
  - Classification and regression
  - Model evaluation
- **Day 3-5**: Scikit-learn
  - Data preprocessing
  - Model selection and evaluation
  - Feature engineering
- **Day 6-7**: Simple neural networks
  - Introduction to TensorFlow/Keras
  - Building a basic neural network
  - **Weekend Project**: Classification system for a real-world dataset

MODULE 5: SPECIALIZED PYTHON (Weeks 24-28)

Week 24: Advanced Machine Learning

- **Day 1-3**: Deep learning
  - Convolutional neural networks
  - Recurrent neural networks
  - Transfer learning
- **Day 4-5**: Natural language processing
  - Text preprocessing
  - Word embeddings
  - Sentiment analysis
- **Day 6-7**: Reinforcement learning
  - Basic RL concepts
  - Q-learning
  - **Weekend Project**: Image classifier or NLP application

Week 25: Data Engineering

- **Day 1-2**: ETL processes
  - Data extraction techniques
  - Transformation operations
  - Loading strategies
- **Day 3-4**: Apache Airflow
  - DAG construction
  - Operators and sensors
  - Scheduling workflows
- **Day 5-7**: Big data processing
  - PySpark basics
  - Distributed computing concepts
  - **Weekend Project**: ETL pipeline with Airflow

Week 26: Desktop GUI and Application Development

- **Day 1-3**: GUI development
  - Tkinter fundamentals
  - PyQt or Kivy
  - Layout management
- **Day 4-5**: Advanced GUI concepts
  - Event handling
  - Custom widgets
  - Threading in GUI applications
- **Day 6-7**: Application deployment
  - Packaging with PyInstaller
  - Creating installers
  - **Weekend Project**: Desktop application with data visualization

Week 27: Automation and Scripting

- **Day 1-2**: System automation
  - OS interaction with os and sys modules
  - Subprocess management

    - File system operations
- **Day 3-4**: Web automation
    - Selenium for browser automation
    - Headless browsers
    - Web scraping ethics
- **Day 5-7**: Task automation
    - Scheduled tasks
    - Monitoring and alerts
    - **Weekend Project**: Automated workflow for a business process

Week 28: Specialized Libraries and Frameworks

- **Day 1-2**: Scientific computing
    - SciPy for scientific algorithms
    - SymPy for symbolic mathematics
- **Day 3-4**: Game development
    - PyGame basics
    - Game loops and physics
- **Day 5-7**: Other domains
    - Financial analysis with Python
    - Bioinformatics libraries
    - Image processing with OpenCV
    - **Weekend Project**: Domain-specific application in an area of interest

MODULE 6: INTERVIEW PREPARATION AND PROFESSIONAL DEVELOPMENT (Weeks 29-30)

Week 29: Python Interview Preparation

- **Day 1-2**: Data structures and algorithms
    - Common algorithms in Python
    - Big O notation and optimization
    - Problem-solving strategies
- **Day 3-4**: System design
    - Designing scalable applications

    - Microservices architecture
    - Design patterns implementation
- **Day 5-7**: Coding exercises
    - LeetCode and HackerRank problems
    - Mock interviews
    - **Weekend Project**: Portfolio of solved algorithm problems

Week 30: Professional Python Development

- **Day 1-2**: Building a portfolio
    - GitHub profile optimization
    - Open source contributions
    - Personal projects showcase
- **Day 3-4**: Modern Python ecosystem
    - Current trends and tools
    - Python 3.11+ features
    - Staying updated in the community
- **Day 5-7**: Advanced project
    - Full-stack Python application
    - DevOps integration
    - **Final Project**: Complete professional-grade application demonstrating multiple skills

RECOMMENDED LEARNING RESOURCES

Books

- "Python Crash Course" by Eric Matthes
- "Fluent Python" by Luciano Ramalho
- "Effective Python" by Brett Slatkin
- "Python Cookbook" by David Beazley and Brian K. Jones
- "Architecture Patterns with Python" by Harry Percival and Bob Gregory
- "Clean Code in Python" by Mariano Anaya
- "High Performance Python" by Micha Gorelick and Ian Ozsvald

Online Platforms

- Real Python (realpython.com)
- Python.org Official Documentation
- DataCamp
- Coursera Python Specializations
- Pluralsight Python Path
- freeCodeCamp Python Certification

YouTube Channels

- Corey Schafer
- Tech With Tim
- sentdex
- ArjanCodes
- mCoding

Blogs and Websites

- Python Weekly Newsletter
- Full Stack Python
- Python Software Foundation News
- PyCoder's Weekly
- Talk Python To Me Podcast
- Test & Code Podcast

GitHub Repositories

- Awesome Python (vinta/awesome-python)
- Python-patterns
- The Algorithms - Python
- Python Cheatsheet

MASTERY MILESTONES

Throughout this roadmap, you should aim to reach these milestones:

1. **Week 4**: Able to write basic Python scripts and understand core language features
2. **Week 10**: Competent in OOP and functional programming paradigms

3. **Week 16**: Capable of building complete applications with databases and web components
4. **Week 23**: Proficient in modern Python development practices and frameworks
5. **Week 28**: Specialized in multiple Python domains and able to tackle complex problems
6. **Week 30**: Professional-level Python developer ready for technical interviews and career advancement

Remember that consistent practice is the key to mastery. Each weekend project builds on the week's learning and helps solidify your understanding. Don't hesitate to revisit earlier topics if needed, as Python development is fundamentally iterative and requires a strong foundation.