Basics

Official Python Tutorial

The most important starting point.

The Hitchhiker's Guide to Python!

"Best practices guidebook written for Humans."

Python 3 Quick Reference

I usually have this page open when I'm writing Python code. You may also scan it from time to time to see if you miss some important Python property.

Ned Batchelder: Facts and Myths about Python names and values

What actually Python names and values are?

Ned Batchelder: Loop like a native: while, for, iterators, generators

Python provides powerful primitives for iterating over your data in ways that let you express yourself clearly and directly. But even programmers familiar with the tools don't use them as fully as they could. This talk covers Python's iteration tools, from basic loops to generators and how to add iteration to your own classes. Come learn how looping was meant to be!

You may also read the blog entry or just the slides.

Idioms and Anti-Idioms in Python

Also, check Common Gotchas, The 10 Most Common Mistakes That Python Developers Make.

Transforming Code into Beautiful, Idiomatic Python

PEP 8 -- Style Guide for Python Code

Python coding style. Best practice for writing readable and maintainable code. These are not hard rules but rules that we should learn, user, and even break sometimes to achieve better readability.

Raymond Hettinger: Beyond PEP 8 -- Best practices for beautiful intelligible code - PyCon 2015

Learn to avoid some of the hazards of the PEP 8 style guide and learn what really matters for creating beautiful intelligible code.

Mark Smith - More Than You Ever Wanted To Know About Python Functions, EuroPython 2018

What exactly are functions? Let's talk about functions, methods, callables and closures - what they are, what you can give them, what they can give you, what you can do with them ... and what's inside.

You probably think you already know everything about functions, but you probably don't!

Variables and scope

Introduction to variables and scope rules. Also, check the StackOverFlow question Short Description of Scoping Rules

Brett Slatkin - How to Be More Effective with Functions - PyCon 2015

Functions improve readability, encourage reuse, and facilitate refactoring. Python has many unique features that make functions significantly more powerful. This talk will show you the best ways to use functions in Python: when *args is helpful and when it'll crash your programs; how to use generators for arguments and return values; the value of keyword vs. keyword-only arguments; and more!

73 Examples to Help You Master Python's f-strings

The most important bits about Python's f-strings.

Richard Haven - Python Decorators and Diversions

Python decorators are an easy way to add meta and mixin behavior to code. They can enhance functions, methods, and even whole classes. And once you can get your head around function variables, they are pretty straight-forward. With one exception.

Learn how the mechanism works so you can add timing, caching, security, metrics, and more to any part of your code

Doug Hellmann - Regular Expressions Are Nothing to Fear

Introduction to regular expressions in Python. examples

Regular Expressions (Regex) Tutorial: How to Match Any Pattern of Text

A simple tutorial for Regular Expressions.

Brandon Rhodes: All Your Ducks In A Row: Data Structures in the Std Lib and Beyond - PyCon 2014

Why are Python programmers crazy about lists and dictionaries, when other languages tout bitmaps, linked lists, and B+ trees? Are we missing out? Come learn how data structures are implemented on bare metal, how to select the right data structure, how the list and dictionary

cover a wide swath of use cases, and when to dip into the Standard Library or a third-party package for an alternative.

The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)

Basic introduction to character encoding, Ascii, Unicode, UTF8, ...etc.

Filter a Set for Matching String Permutations

Raymond Hettinger answer to this question uses different techniques to solve a simple problem with huge performance variations based on problem size.

Are tuples more efficient than lists in Python?

The Magic of Python Context Managers

Recipes for using and creating awesome Python context managers, that will make your code more readable, reliable, and less error-prone.

Intermediate

Ned Batchelder: Getting Started Testing

If you've never written tests before, you probably know you *should*, but view the whole process as a bureaucratic paperwork nightmare to check off on your ready-to-ship checklist. This is the wrong way to approach testing. Tests are a solution to a problem that is important to you: does my code work? I'll show how Python tests are written, and why.

Python's super() considered super!

Blog entry on how to use `super()` effectively. Note that he is using Python 3 syntax (where you do not need to pass class and instance to super), in python 2.7 you need to call `super(class, self)`.

Ned Batchelder: Pragmatic Unicode, or, How do I stop the pain?

Python has great Unicode support, but it's still your responsibility to handle it properly. A quick overview of what Unicode is, but only enough to get your program working properly.

Context Managers the Easy Way

A brief introduction to context managers.

Also, check Generic Exception Handling in Python the "Right Way"

Python dataclasses will save you HOURS, also featuring attrs - YouTube

Learn about dataclasses and how to use them, as well as the related attrs library that dataclasses were based on.

Closures

- Python Closures Explained
- Python Closures
- Python Closures and Decorators

The Clean Architecture in Python

Even design-conscious programmers find large applications difficult to maintain. Come learn about how the recently propounded “Clean Architecture” applies in Python, and how this high-level design pattern fits particularly well with the features of the Python language and answers questions that experienced programmers have been asking.

Raymond Hettinger, Keynote on Concurrency, PyBay 2017

An introduction to concurrency techniques in Python. This should clear confusion about the difference between multiple threading, multiple processes, and asyncio approaches.

Ned Batchelder : Shell = Maybe

How to use `subprocess` module to call external programs from Python properly and avoid *Shell Injection*.

Floating Point Arithmetic: Issues and Limitations

It is very important for any programmer to understand the trade-offs and limitations of floating point numbers representation in CPUs. This is a must-read if you do not exactly know why `3 * 0.1 == 0.3` is `False`.

Raymond Hettinger Modern Python Dictionaries A confluence of a dozen great ideas PyCon 201

Raymond Hettinger Pythons dictionaries are stunningly good. Over the years, many great ideas have combined together to produce the modern implementation in Python 3.6.

Difference between getattr vs getattribute Both method return the value of an object attribute. So, what is the difference and which one to override?

This item was added to the "Basics" section but moved to the "Intermediate" section as it describes the design principals of "dictionary" type which may not suitable for beginners level.

What is the difference between venv, pyvenv, pyenv, virtualenv, virtualenvwrapper, pipenv, etc?

A good answer to that question on StackOverflow. The comments have a lot of useful information too.

What Does It Take To Be An Expert At Python?

A very nice introduction to four Python concepts: Metaclasses, Decorators, Generators, and Context Managers.

Advanced

Raymond Hettinger: Super considered super! - PyCon 2015

Python's super() is well-designed and powerful, but it can be tricky to use if you don't know all the moves.

This talk offers clear, practical advice with real-world use cases on how to use super() effectively and not get tripped-up by common mistakes.

Descriptor HowTo Guide

Defines descriptors, summarizes the protocol and shows how descriptors are called. Examines a custom descriptor and several built-in Python descriptors including functions, properties, static methods, and class methods. Shows how each works by giving a pure Python equivalent and a sample application.

Learning about descriptors not only provides access to a larger toolset, but it also creates a deeper understanding of how Python works and an appreciation for the elegance of its design.

Python Descriptors Demystified

Description of Python descriptors. You should also read the discussion on Ned Batchelder: Explaining descriptors and Descriptor HowTo Guide

What is a metaclass in Python

A very informative answer on StackOverflow

Understanding Python metaclasses

A Primer on Python Metaclass Programming

Grok the GIL: How to write fast and thread-safe Python

Exploring Python's global interpreter lock and learn how it affects multithreaded programs.

What Every Computer Scientist Should Know About Floating Point Arithm