

# ADDITIONAL RESOURCES

# The Python documentation

That should be your top bookmark for Python

<https://docs.python.org>

Don't forget to make sure you are looking at your version of Python.

3.6 or above please!

The screenshot shows the Python 3.6.4 documentation page. A yellow arrow points to the version dropdown menu in the top navigation bar, which is currently set to 3.6.4. Another yellow arrow points to the 'Tutorial' link in the 'Parts of the documentation' section, with a third yellow arrow pointing to the 'Language Reference' link below it. The page layout includes a left sidebar with links for downloading documents, other versions, and resources. The main content area features a welcome message and a list of documentation parts. The right sidebar contains links for installing, distributing, and extending Python modules, as well as FAQs.

Python » English ▼ 3.6.4 ▼ Documentation »

Quick search  Go | modules | in

## Python 3.6.4 documentation

Welcome! This is the documentation for Python 3.6.4.

**Parts of the documentation:**

- [What's new in Python 3.6?](#)  
*or all "What's new" documents since 2.0*
- [Tutorial](#)  
*start here*
- [Library Reference](#)  
*keep this under your pillow*
- [Language Reference](#)  
*describes syntax and language elements*
- [Python Setup and Usage](#)  
*how to use Python on different platforms*
- [Python HOWTOs](#)  
*in-depth documents on specific topics*

**Indices and tables:**

- [Installing Python Modules](#)  
*installing from the Python Package Index & other sources*
- [Distributing Python Modules](#)  
*publishing modules for installation by others*
- [Extending and Embedding](#)  
*tutorial for C/C++ programmers*
- [Python/C API](#)  
*reference for C/C++ programmers*
- [FAQs](#)  
*frequently asked questions (with answers!)*



## PEP – Python Enhancement Proposals

These are a fantastic resource to understand how certain things work in Python, and why they were implemented in a certain way..

Not all PEPs actually make it into Python. Some are rejected, deferred or even withdrawn.

Reading the PEPs that have not been accepted also provides a lot of insight!  
A lot of thought by many people go into these PEPs, whether they make it or not.

| PEP 274 -- Dict Comprehensions |                                    |
|--------------------------------|------------------------------------|
| PEP:                           | 274                                |
| Title:                         | Dict Comprehensions                |
| Author:                        | Barry Warsaw <barry at python.org> |
| Status:                        | Final                              |
| Type:                          | Standards Track                    |
| Created:                       | 25-Oct-2001                        |
| Python-Version:                | 2.7, 3.0 (originally 2.3)          |
| Post-History:                  | 29-Oct-2001                        |

Some PEPs are for language features  
some are informational only

Index page <https://www.python.org/dev/peps/>

search on that page

But sometimes a web search such as: [Python PEP Style Guide](#) is more practical

## PEP – Some Notable Ones

PEP 8 – Style Guide and Idiomatic Python

PEP 20 – Zen of Python      or just type `import this` in a Python console/Jupyter

PEP 484 – Type Hints

PEP 468 – Python 3.6 Release Schedule


PEP 537 – Python 3.7 Release Schedule

or whatever release your interested in at the time  
they provide links to other PEPs relevant to the release

And many many more, depending on what topic you're interested in



# Great resource for explanations of general computer science concepts



**WIKIPEDIA**  
The Free Encyclopedia

- Main page
- Contents
- Featured content
- Current events
- Random article
- Donate to Wikipedia
- Wikipedia store

---

Interaction

- Help
- About Wikipedia
- Community portal
- Recent changes
- Contact page

---

Tools

- What links here
- Related changes
- Upload file
- Special pages
- Permanent link
- Page information
- Wikidata item
- Cite this page

---

Print/export

- Create a book
- Download as PDF
- Printable version

---

In other projects

- Wikimedia Commons

## Hash table

From Wikipedia, the free encyclopedia

*Not to be confused with [Hash list](#) or [Hash tree](#).*

*"Rehash" redirects here. For the South Park episode, see [Rehash \(South Park\)](#). For the IRC command, see [List of Internet Relay Chat commands § REHASH](#).*

In **computing**, a **hash table** (**hash map**) is a **data structure** which implements an **associative array abstract data type**, a structure that can map **keys** to **values**. A hash table uses a **hash function** to compute an **index** into an array of *buckets* or *slots*, from which the desired value can be found.

Ideally, the hash function will assign each key to a unique bucket, but most hash table designs employ an imperfect hash function, which might cause hash *collisions* where the hash function generates the same index for more than one key. Such collisions must be accommodated in some way.

In a well-dimensioned hash table, the average cost (number of **instructions**) for each lookup is independent of the number of elements stored in the table. Many hash table designs also allow arbitrary insertions and deletions of key-value pairs, at (**amortized**<sup>[2]</sup>) constant average cost per operation.<sup>[3][4]</sup>

In many situations, hash tables turn out to be more efficient than **search trees** or any other **table** lookup structure. For this reason, they are widely used in many kinds of computer **software**, particularly for associative arrays, **database indexing**, **caches**, and **sets**.

**Hash table**

**Type** Unordered **associative array**

**Invented** 1953

**Time complexity in big O notation**

| Algorithm     | Average      | Worst case |
|---------------|--------------|------------|
| <b>Space</b>  | $O(n)^{[1]}$ | $O(n)$     |
| <b>Search</b> | $O(1)$       | $O(n)$     |
| <b>Insert</b> | $O(1)$       | $O(n)$     |
| <b>Delete</b> | $O(1)$       | $O(n)$     |

**Contents** [hide]

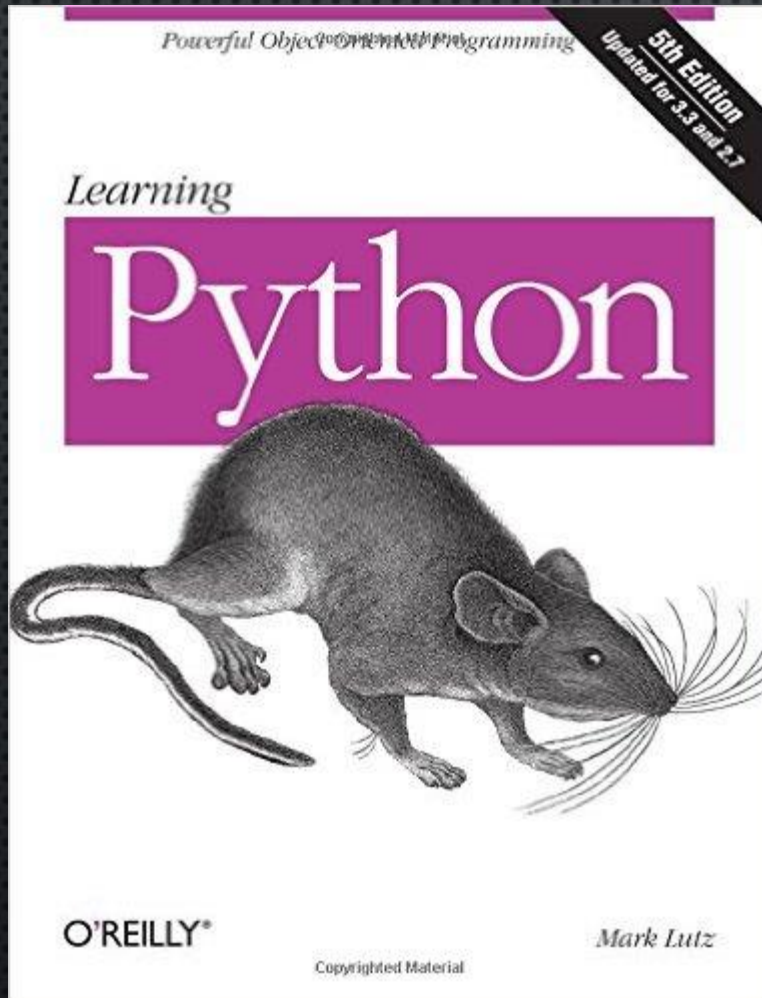
- Hashing
  - Choosing a hash function
  - Perfect hash function
- Key statistics
- Collision resolution
  - Separate chaining
    - Separate chaining with linked lists
    - Separate chaining with list head cells
    - Separate chaining with other structures
  - Open addressing
    - Coalesced hashing
    - Cuckoo hashing
    - Hopscotch hashing
  - Robin Hood hashing

A small phone book as a hash table

## Books

These are my favorite Python specific go to books

not in any particular order of importance!



Learning Python  
Mark Lutz

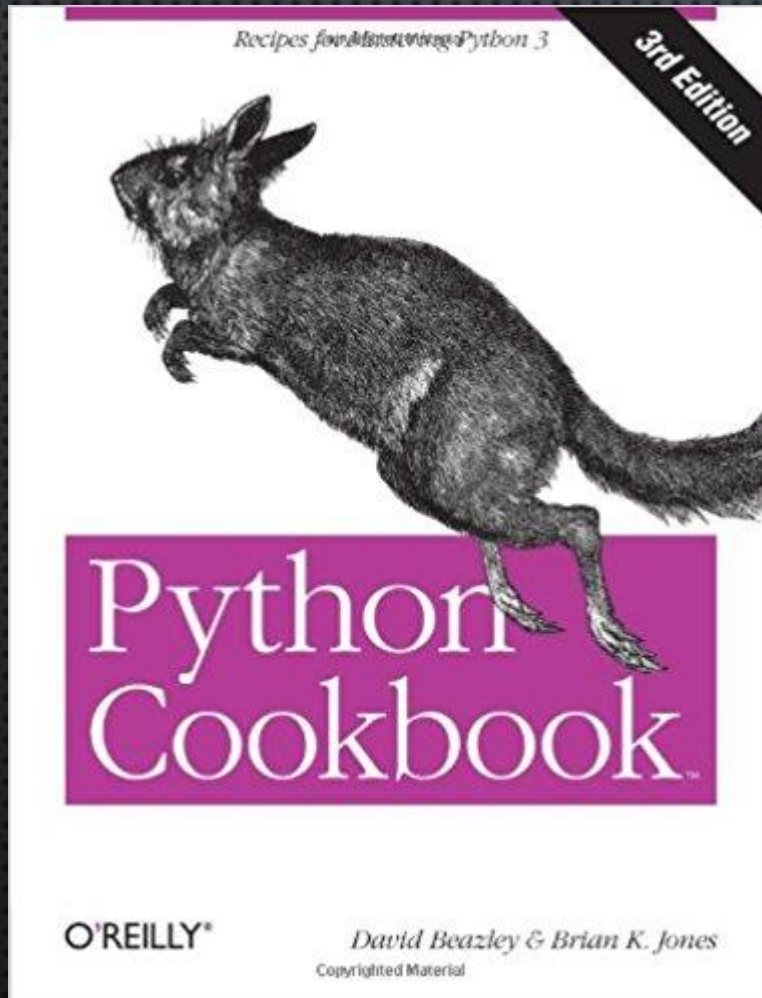


## Books



## Fluent Python Luciano Ramalho

## Books

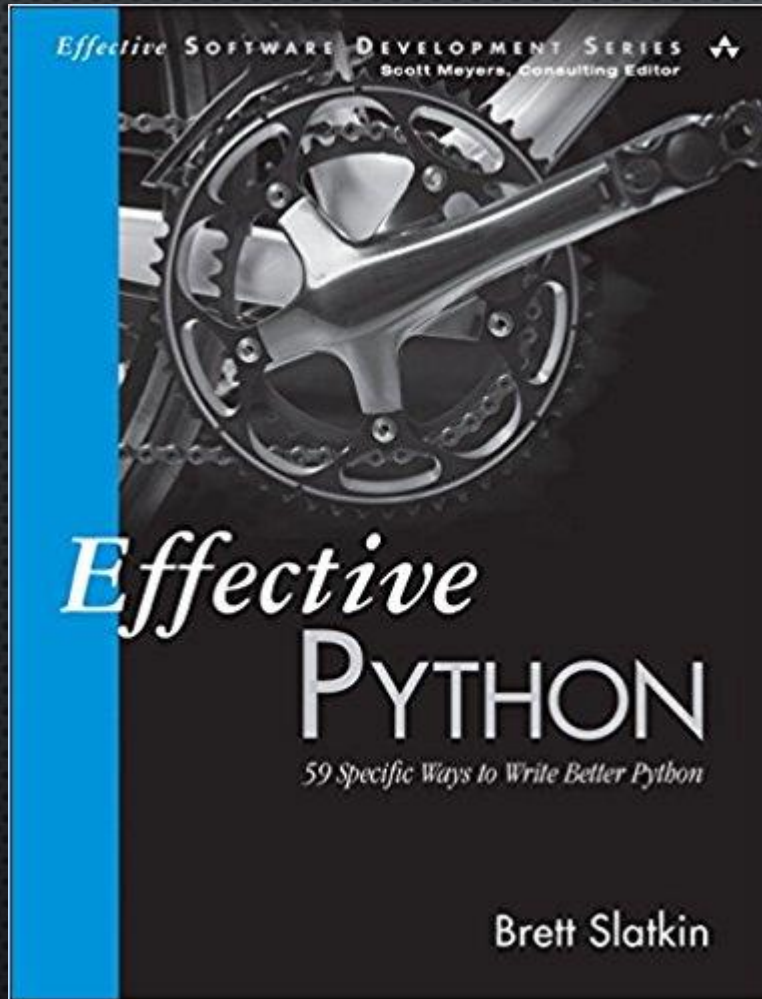


## Python Cookbook

David Beazley & Brian K. Jones

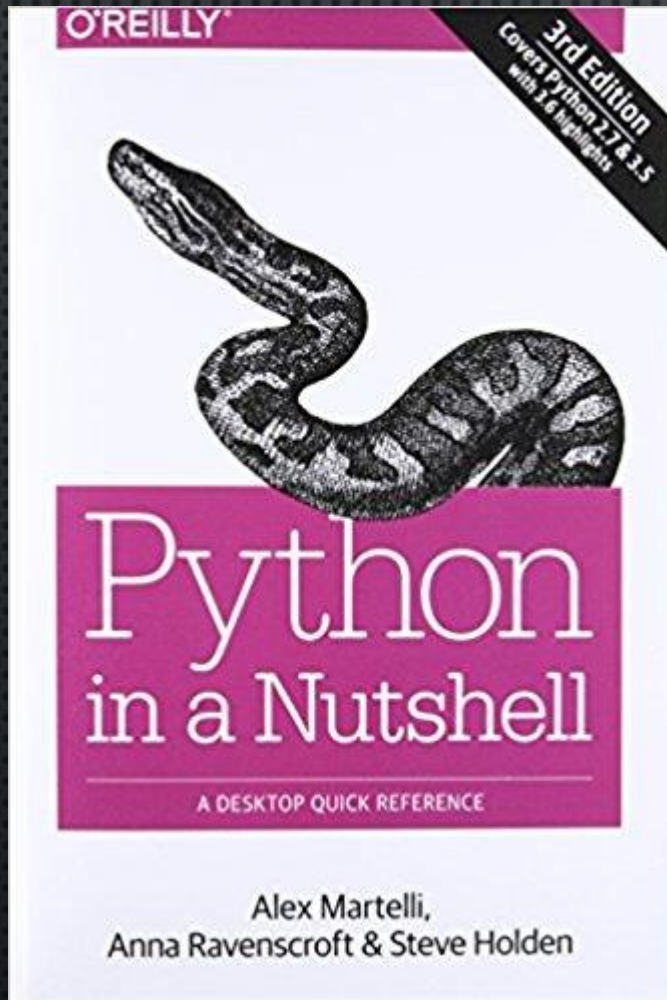


## Books



## Effective Python: 59 Specific Ways to Write Better Python Brett Slatkin

## Books



## Python in a Nutshell

Alex Martelli, Anna Ravenscroft & Steve Holden



## Other Online Resources I Regularly Use

Raymond Hettinger's Twitter Feed

[@raymondh](#)

just awesome!

example:

[#python](#) tip: `zip()` with star-arguments is great for transposing 2-D data:  
`m = [(1, 2, 3), (4, 5, 6)]`  
`list(zip(*m))`  
`[(1, 4), (2, 5), (3, 6)]`

YouTube

Lots of great videos on Python.  
Look out for [PyCon](#) videos – these are fantastic!  
Anything by [GvR](#), [Raymond Hettinger](#), [Alex Martelli](#)...  
And many more, including any library you're interested in

Planet Python Blog

<http://planetpython.org/>

Google Searches!

Stack Overflow

<https://stackoverflow.com/>