

Table of Contents

1. [FrontPage - Python Wiki](#)
2. [Welcome to Python.org](#)
3. [BeginnersGuide - Python Wiki](#)
4. [BeginnersGuide - Python Wiki](#)
5. [FrontPage - Python Wiki](#)
6. [RecentChanges - Python Wiki](#)
7. [FindPage - Python Wiki](#)
8. [HelpContents - Python Wiki](#)
9. [BeginnersGuide - Python Wiki](#)
10. [Info for "BeginnersGuide" - Python Wiki](#)
11. [Attachments for "BeginnersGuide" - Python Wiki](#)
12. [Login - Python Wiki](#)
13. [BeginnersGuideChinese - Python Wiki](#)
14. [BeginnersGuide/Overview - Python Wiki](#)
15. [BeginnersGuide/Download - Python Wiki](#)
16. [IntegratedDevelopmentEnvironments - Python Wiki](#)
17. [HowToEditPythonCode - Python Wiki](#)
18. [BeginnersGuide/NonProgrammers - Python Wiki](#)
19. [BeginnersGuide/Programmers - Python Wiki](#)
20. [Languages - Python Wiki](#)
21. [Python on Windows FAQ](#)
22. [Beginner's Python Tutorial: Learn Python](#)
- 23.
24. [Learn Python 2 | Codecademy](#)
25. [Coding Bootcamps - Ultimate source of learning coding](#)
26. [Learn R, Python & Data Science Online](#)
27. [Dataquest: The fastest way to learn AI and data skills online](#)
28. [HackInScience — Python Exercises](#)
29. [High School Technology Services in Washington DC](#)
30. [19 Keywords Every Coder Must Know – Be on the Right Side of Change](#)
31. [3.12.2 Documentation](#)
32. [The Python Tutorial](#)
33. [The Python Standard Library](#)
34. [The Python Language Reference](#)
35. [PythonEditors - Python Wiki](#)
36. [IntegratedDevelopmentEnvironments - Python Wiki](#)
37. [IDLE](#)
38. [Powerful Python One-Liners - Python Wiki](#)
39. [BeginnersGuide/Help - Python Wiki](#)
40. [IntroductoryBooks - Python Wiki](#)
41. [BeginnersGuide/Examples - Python Wiki](#)
42. [PythonEvents - Python Wiki](#)
43. [PythonTraining - Python Wiki](#)
44. [BeginnersGuide/Download - Python Wiki](#)
45. [BeginnersGuide/Examples - Python Wiki](#)

46. [BeginnersGuide/Help - Python Wiki](#)
47. [BeginnersGuide/Mathematics - Python Wiki](#)
48. [BeginnersGuide/NonProgrammers - Python Wiki](#)
49. [BeginnersGuide/NonProgrammersChinese - Python Wiki](#)
50. [BeginnersGuide/Overview - Python Wiki](#)
51. [BeginnersGuide/OverviewChinese - Python Wiki](#)
52. [BeginnersGuide/Programmers - Python Wiki](#)
53. [attachment:Cpp2Python.pdf of BeginnersGuide/Programmers - Python Wiki](#)
54. [BeginnersGuide/Programmers/SimpleExamples - Python Wiki](#)
55. [Be on the Right Side of Change](#)
56. [CheckiO - coding games and programming challenges for beginner and advanced](#)
57. [Python Quiz - After Hours Programming](#)
- 58.
59. [Python Tests / Quizes](#)
60. [Python Syntax Checker PEP8](#)
61. [PyPI · The Python Package Index](#)
62. [Welcome to Python.org](#)
63. [Welcome to Python.org](#)
64. [Welcome to Python.org](#)
- 65.
66. [Google](#)
67. [\[Collection\] 11 Python Cheat Sheets Every Python Coder Must Own – Be on the Right Side of Change](#)
68. [NumPy - Python Wiki](#)
69. [\[Collection\] 10 Best NumPy Cheat Sheets Every Python Coder Must Own – Be on the Right Side of Change](#)
70. [\[PDF Collection\] 7 Beautiful Pandas Cheat Sheets — Post Them to Your Wall – Be on the Right Side of Change](#)
71. [Best 15+ Machine Learning Cheat Sheets to Pin to Your Toilet Wall – Be on the Right Side of Change](#)
72. [List of issues - Python tracker](#)
73. [Python Developer's Guide](#)
74. [Welcome to Python.org](#)

FrontPage - Python Wiki

Source: <https://wiki.python.org/moin/FrontPage>

Welcome to the Python Wiki, a user-editable compendium of knowledge based around the Python programming language. Some pages are protected against casual editing - see [WikiEditingGuidelines](#) → <https://wiki.python.org/moin/WikiEditingGuidelines> for more information about editing content.

[Python](#) → <http://www.python.org/> is a great object-oriented, interpreted, and interactive programming language. It is often compared → <https://wiki.python.org/moin/LanguageComparisons> (favorably of course 😊)

) to Lisp, Tcl, Perl, Ruby, C#, Visual Basic, Visual Fox Pro, Scheme or Java... and it's much more fun.

Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems → <https://wiki.python.org/moin/GUI%20Programming%20in%20Python>. New built-in modules are easily written in C or C++ (or other languages, depending on the chosen implementation → <https://wiki.python.org/moin/PythonImplementations>). Python is also usable as an extension language for applications written in other languages → <https://wiki.python.org/moin/AppsWithPythonScripting> that need easy-to-use scripting or automation interfaces.

Getting Started

- [Python Conferences](#) → <https://wiki.python.org/moin/PythonConferences> - information about the Python conference scene
- [Local User Groups](#) → <https://wiki.python.org/moin/LocalUserGroups> - find a Python group near you
- [Python Training](#) → <https://wiki.python.org/moin/PythonTraining> - Python training courses
- [Python Events](#) → <https://wiki.python.org/moin/PythonEvents> - event listing for conferences, training courses and more
- [Python Event Calendars](#) → <https://wiki.python.org/moin/PythonEventsCalendar> - calendars for Python conferences and user groups
- [Participating in the Community](#) → <https://wiki.python.org/moin/Community> - where people using and producing Python get together
- [Python Software Foundation](#) → <https://wiki.python.org/moin/PythonSoftwareFoundation> - show your support by joining the Foundation behind Python
- [Find a job where you can use Python](#) → <https://wiki.python.org/moin/PythonJobs> - Python job boards around the world

Python Software

Using this Wiki

This Wiki is a community place to gather and organize all things about Python. Feel free to exercise your editorial skills and expertise to make it a useful knowledge base and up-to-date reference on all Python-related topics.

There are some [guidelines](#) → <https://wiki.python.org/moin/WikiGuidelines> describing the policies and rules governing this Wiki and how you can most effectively contribute to it. A list of [site improvements](#) → <https://wiki.python.org/moin/SiteImprovements> describes various tasks where your help would be appreciated. To keep up with changes on this site, check [RecentChanges](#) → <https://wiki.python.org/moin/RecentChanges> frequently or follow it using RSS: [RSS feed](#) → https://wiki.python.org/moin/FrontPage?action=rss_rc.

Creating a Wiki account

In order to sign up for a wiki account, please go to the [Create new account → https://wiki.python.org/moin/FrontPage?action=newaccount](https://wiki.python.org/moin/FrontPage?action=newaccount) form, enter your account name (using the format FirstnameLastname to avoid issues - please don't use spaces in the name) and provide a password, plus email address (for password recovery).

Editing pages

Since spamming and vandalism on this wiki had reached a level that required constant intervention, unfamiliar users are no longer allowed to edit pages. However all you need to do is introduce yourself to the wiki admin group to become an editor.

If you want to edit a page and have just signed up, or find that you can no longer edit a page that you could edit before, please write to the [pydorg-www mailing list → mailto:pydorg-www@python.org](mailto:pydorg-www@python.org) describing what you would like to edit, and we'll add you to the [EditorsGroup → https://wiki.python.org/moin/EditorsGroup](https://wiki.python.org/moin/EditorsGroup). **Please include your account name (wiki name) in this message.**

Sorry for any inconvenience, but we want to keep this wiki a useful tool for the community, while at the same time preventing the wiki admins from burning out cleaning up junk.

Reporting problems

In case of emergency, please contact the [python.org maintainers → mailto:webmaster@python.org](mailto:webmaster@python.org), or if experiencing difficulties, contact the [pydorg-www mailing list → mailto:pydorg-www@python.org](mailto:pydorg-www@python.org) to say "help".

Wiki Attack in January 2013

The wiki was subject to an attack on January 5 2013. Since it was not clear whether user account data was stolen, all passwords were subsequently reset, so you will have to use the [password recovery function → https://wiki.python.org/moin/FrontPage?action=recoverpass](https://wiki.python.org/moin/FrontPage?action=recoverpass) to get a new password.

See the [wiki attack description page → https://wiki.python.org/moin/WikiAttack2013](https://wiki.python.org/moin/WikiAttack2013) for more details. If you find problems, please report them to the pydorg-www mailing list <pydorg-www@python.org>.

HTTPS access to the Wiki

We have enabled HTTPS access to the wiki to further enhance security and avoid having to send clear text passwords over the network in order to log in to the wikis.

If you have not been using HTTPS links to the wiki login page, please be advised that your password may have been sniffed on the network at e.g. a conference. It is best to [change it → https://wiki.python.org/moin/FrontPage?action=userprefs](https://wiki.python.org/moin/FrontPage?action=userprefs) and stop using HTTP links to the wiki login page.

Welcome to Python.org

Source: <http://www.python.org>

Notice: While JavaScript is not essential for this website, your interaction with the content will be limited. Please turn JavaScript on for the full experience.

Download

Python source code and installers are available for download for all versions!

Latest: [Python 3.12.2 → http://www.python.org/downloads/release/python-3122/](http://www.python.org/downloads/release/python-3122/)

Docs

Documentation for Python's standard library, along with tutorials and guides, are available online.

[docs.python.org → https://docs.python.org/](https://docs.python.org/)

Jobs

Looking for work or have a Python related position that you're trying to hire for? Our **relaunched community-run job board** is the place to go.

[jobs.python.org → http://jobs.python.org/](http://jobs.python.org/)

Latest News

More → <https://blog.python.org/>

- 2024-02-29 [White House recommends use of memory-safe languages like Python → https://pyfound.blogspot.com/2024/02/white-house-recommends-.html](https://pyfound.blogspot.com/2024/02/white-house-recommends-.html)
- 2024-02-15 [Python 3.13.0 alpha 4 is now available → https://pythoninsider.blogspot.com/2024/02/python-3130-alpha-4-is-now-available.html](https://pythoninsider.blogspot.com/2024/02/python-3130-alpha-4-is-now-available.html)
- 2024-02-08 [Software Bill-of-Materials documents are now available for CPython → https://pyfound.blogspot.com/2024/02/software-bill-of-materials-now-available-for-cpython.html](https://pyfound.blogspot.com/2024/02/software-bill-of-materials-now-available-for-cpython.html)
- 2024-02-07 [Introducing PSF Grants Program Office Hours → https://pyfound.blogspot.com/2024/02/introducing-psf-grants-office-hours.html](https://pyfound.blogspot.com/2024/02/introducing-psf-grants-office-hours.html)
- 2024-02-07 [Python 3.12.2 and 3.11.8 are now available. → https://pythoninsider.blogspot.com/2024/02/python-3122-and-3118-are-now-available.html](https://pythoninsider.blogspot.com/2024/02/python-3122-and-3118-are-now-available.html)

Upcoming Events

More → <http://www.python.org/events/calendars/>

- 2024-03-15 [Django Girls Eket Workshop → http://www.python.org/events/python-user-group/1710/](http://www.python.org/events/python-user-group/1710/)
- 2024-03-15 [PyCon SK 2024 → http://www.python.org/events/python-events/1509/](http://www.python.org/events/python-events/1509/)
- 2024-03-23 [Python Barcamp Karlsruhe → http://www.python.org/events/python-user-group/1660/](http://www.python.org/events/python-user-group/1660/)
- 2024-03-23 [PyLadies Amsterdam: Python Open Source Sprints → http://www.python.org/events/python-user-group/1704/](http://www.python.org/events/python-user-group/1704/)
- 2024-03-27 [PyLadies Amsterdam: Global AI Bootcamp → http://www.python.org/events/python-user-group/1705/](http://www.python.org/events/python-user-group/1705/)

Use Python for...

More → <http://www.python.org/about/apps>

- **Web Development:** [Django](http://www.djangoproject.com/) → <http://www.djangoproject.com/>, [Pyramid](http://www.pylonsproject.org/) → <http://www.pylonsproject.org/>, [Bottle](http://bottlepy.org/) → <http://bottlepy.org/>, [Tornado](http://tornadoweb.org/) → <http://tornadoweb.org/>, [Flask](http://flask.pocoo.org/) → <http://flask.pocoo.org/>, [web2py](http://web2py.com/) → <http://web2py.com/>
- **GUI Development:** [tkInter](http://wiki.python.org/moin/TkInter) → <http://wiki.python.org/moin/TkInter>, [PyGObject](https://wiki.gnome.org/Projects/PyGObject) → <https://wiki.gnome.org/Projects/PyGObject>, [PyQt](http://www.riverbankcomputing.co.uk/software/pyqt/intro) → <http://www.riverbankcomputing.co.uk/software/pyqt/intro>, [PySide](https://wiki.qt.io/PySide) → <https://wiki.qt.io/PySide>, [Kivy](https://kivy.org/) → <https://kivy.org/>, [wxPython](http://www.wxpython.org/) → <http://www.wxpython.org/>, [DearPyGui](https://dearpygui.readthedocs.io/en/latest/) → <https://dearpygui.readthedocs.io/en/latest/>
- **Scientific and Numeric:** [SciPy](http://www.scipy.org/) → <http://www.scipy.org/>, [Pandas](http://pandas.pydata.org/) → <http://pandas.pydata.org/>, [IPython](http://ipython.org/) → <http://ipython.org/>
- **Software Development:** [Buildbot](http://buildbot.net/) → <http://buildbot.net/>, [Trac](http://trac.edgewall.org/) → <http://trac.edgewall.org/>, [Roundup](http://roundup.sourceforge.net/) → <http://roundup.sourceforge.net/>
- **System Administration:** [Ansible](http://www.ansible.com/) → <http://www.ansible.com/>, [Salt](https://saltproject.io/) → <https://saltproject.io/>, [OpenStack](https://www.openstack.org/) → <https://www.openstack.org/>, [xonsh](https://xon.sh/) → <https://xon.sh/>

>>> **Python Enhancement Proposals (PEPs)** → <http://www.python.org/dev/peps/>: The future of Python is discussed here.

BeginnersGuide - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide>

Beginner's Guide to Python

New to programming? Python is free and easy to learn if you know where to start! This guide will help you to get started quickly.

[Chinese Translation/](#) → <https://wiki.python.org/moin/BeginnersGuideChinese>

New to Python?

Read [BeginnersGuide/Overview](#) → <https://wiki.python.org/moin/BeginnersGuide/Overview> for a short explanation of what Python is.

Getting Python

Next, install the Python 3 interpreter on your computer. This is the program that reads Python programs and carries out their instructions; you need it before you can do any Python programming. Mac and Linux distributions may include an outdated version of Python (Python 2), but you should install an updated one (Python 3). See [BeginnersGuide/Download](#) → <https://wiki.python.org/moin/BeginnersGuide/Download> for instructions to download the correct version of Python.

There are also Python interpreter and IDE bundles available, such as [Thonny](#) → <http://thonny.org/>. Other options can be found at [IntegratedDevelopmentEnvironments](#) → <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>.

At some stage, you'll want to edit and save your program code. Take a look at [HowToEditPythonCode](#) → <https://wiki.python.org/moin/HowToEditPythonCode> for some advice and recommendations.

Learning Python

Next, read a tutorial and try some simple experiments with your new Python interpreter.

- If you have never programmed before, see [BeginnersGuide/NonProgrammers](#) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers> for a list of suitable tutorials.
- If you have previous programming experience, consult [BeginnersGuide/Programmers](#) → <https://wiki.python.org/moin/BeginnersGuide/Programmers>, which lists more advanced tutorials.
- If English isn't your first language, you might be more comfortable with a tutorial that's been translated into your language. Consult python.org's list of Non-English resources → <https://wiki.python.org/moin/Languages>.

Most tutorials assume you know how to run a program on your computer. If you are using Windows and need help with this, see [How do I Run a Program Under Windows](#) → <http://www.python.org/doc/faq/windows/#how-do-i-run-a-python-program-under-windows>.

Here are some sites that focus on beginners and offer in-browser coding:

- [Beginners Python tutorial](#) → <https://python.land/python-tutorial> at Python Land (free)
- [Codéex](#) → <https://www.codedex.io/python> (non-free)
- [Coding Bootcamps](#) → <https://www.coding-bootcamps.com/> (non-free)
- [DataCamp](#) → <https://www.datacamp.com/> (non-free)
- [Dataquest](#) → <https://www.dataquest.io/> for Python for data science. (free)

- [HackInScience](https://www.hackinscience.org/) → <https://www.hackinscience.org/> free and open source platform.
- [High School Technology Services](https://www.mybsts.org/) → <https://www.mybsts.org/> for general Python (non-free)

Print a [cheat sheet](https://blog.finxter.com/python-cheat-sheet/) → <https://blog.finxter.com/python-cheat-sheet/> of the most important Python features and post it to your office wall until you know the basics well.

Once you have read a tutorial, you can browse through [Python's online documentation](http://docs.python.org/) → <http://docs.python.org/>. It includes a [tutorial](http://docs.python.org/tut/) → <http://docs.python.org/tut/> that might come in handy, a [Library Reference](http://docs.python.org/lib/) → <http://docs.python.org/lib/> that lists all of the modules that come standard with Python, and the [Language Reference](http://docs.python.org/ref/) → <http://docs.python.org/ref/> for a complete (if rather dry) explanation of Python's syntax.

When you are ready to write your first program, you will need a [text editor](https://wiki.python.org/moin/PythonEditors) → <https://wiki.python.org/moin/PythonEditors> or an [IDE](https://wiki.python.org/moin/IntegratedDevelopmentEnvironments) → <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>. If you don't want to use Thonny or something more advanced, then you can use [IDLE](https://docs.python.org/3/library/idle.html) → <https://docs.python.org/3/library/idle.html>, which is bundled with Python and supports [extensions](http://idlex.sourceforge.net/) → <http://idlex.sourceforge.net/>.

This Python wiki also contains a page about [Python One-Liners](https://wiki.python.org/moin/Powerful%20Python%20One-Liners) → <https://wiki.python.org/moin/Powerful%20Python%20One-Liners> -- an obscure but interesting subculture in Python.

Need Help?

Need help with any of this? Read [BeginnersGuide/Help](https://wiki.python.org/moin/BeginnersGuide/Help) → <https://wiki.python.org/moin/BeginnersGuide/Help> for mailing lists and newsgroups.

Most Python books will include an introduction to the language; see [IntroductoryBooks](https://wiki.python.org/moin/IntroductoryBooks) → <https://wiki.python.org/moin/IntroductoryBooks> for suggested titles.

Consult [BeginnersGuide/Examples](https://wiki.python.org/moin/BeginnersGuide/Examples) → <https://wiki.python.org/moin/BeginnersGuide/Examples> for small programs and little snippets of code that can help you learn.

Or, if you prefer to learn Python through listening to a lecture, you can attend a training course or even hire a trainer to come to your company. Consult the [PythonEvents](https://wiki.python.org/moin/PythonEvents) → <https://wiki.python.org/moin/PythonEvents> page to see if any training courses are scheduled in your area and the [PythonTraining](https://wiki.python.org/moin/PythonTraining) → <https://wiki.python.org/moin/PythonTraining> page for a list of trainers.

Teachers can join the [EDU-SIG](http://www.python.org/sigs/edu-sig/) → <http://www.python.org/sigs/edu-sig/>, a mailing list for discussion of Python's use in teaching at any level ranging from K-12 up to university.

Complete list of Beginner's Guide pages

1. [BeginnersGuide/Download](https://wiki.python.org/moin/BeginnersGuide/Download) → <https://wiki.python.org/moin/BeginnersGuide/Download>
2. [BeginnersGuide/Examples](https://wiki.python.org/moin/BeginnersGuide/Examples) → <https://wiki.python.org/moin/BeginnersGuide/Examples>
3. [BeginnersGuide/Help](https://wiki.python.org/moin/BeginnersGuide/Help) → <https://wiki.python.org/moin/BeginnersGuide/Help>
4. [BeginnersGuide/Mathematics](https://wiki.python.org/moin/BeginnersGuide/Mathematics) → <https://wiki.python.org/moin/BeginnersGuide/Mathematics>
5. [BeginnersGuide/NonProgrammers](https://wiki.python.org/moin/BeginnersGuide/NonProgrammers) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers>
6. [BeginnersGuide/NonProgrammersChinese](https://wiki.python.org/moin/BeginnersGuide/NonProgrammersChinese) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammersChinese>
7. [BeginnersGuide/Overview](https://wiki.python.org/moin/BeginnersGuide/Overview) → <https://wiki.python.org/moin/BeginnersGuide/Overview>
8. [BeginnersGuide/OverviewChinese](https://wiki.python.org/moin/BeginnersGuide/OverviewChinese) → <https://wiki.python.org/moin/BeginnersGuide/OverviewChinese>
9. [BeginnersGuide/Programmers](https://wiki.python.org/moin/BeginnersGuide/Programmers) → <https://wiki.python.org/moin/BeginnersGuide/Programmers>
10. [BeginnersGuide/Programmers \(Cpp2Python.pdf\)](https://wiki.python.org/moin/BeginnersGuide/Programmers(Cpp2Python.pdf)) → <https://wiki.python.org/moin/BeginnersGuide/Programmers?action=AttachFile&do=view&target=Cpp2Python.pdf>
11. [BeginnersGuide/Programmers/SimpleExamples](https://wiki.python.org/moin/BeginnersGuide/Programmers/SimpleExamples) → <https://wiki.python.org/moin/BeginnersGuide/Programmers/SimpleExamples>

Quiz and Exercises

- Finxter - [How good are your Python skills? Test and Training with a Daily Python Puzzle](https://finxter.com/) → <https://finxter.com/>
- CheckIO - [Online learning, testing and improving your python skills](http://www.checkio.org/) → <http://www.checkio.org/>
- After Hours Programming - [Python Quiz](http://www.afterhoursprogramming.com/tutorial/Python/Python-Quiz/) → <http://www.afterhoursprogramming.com/tutorial/Python/Python-Quiz/>
- PyGUI - [Collection of python quiz answers, Examples And GUI Tkinter Tutorials For Beginners](http://www.pythongui.com/) → <http://www.pythongui.com/>
- Pythonspot - [Python Quiz](https://pythonspot.com/python-tests-quizes/) → <https://pythonspot.com/python-tests-quizes/>
- Python Challenge - [A Python Quiz App on Android Platform](https://play.google.com/store/apps/details?id=sg.apps.garden.pythonchallenge) → <https://play.google.com/store/apps/details?id=sg.apps.garden.pythonchallenge>
- CS Circles - [online lessons and graded exercises](http://cscircles.cemc.uwaterloo.ca/) → <http://cscircles.cemc.uwaterloo.ca/>

Python Style Checker

- [Pythonchecker.com](http://pythonchecker.com/) → <http://pythonchecker.com/> - An educative online tool to rate your Python style (with dynamic score computation and hints)

Looking for a particular Python module or application?

- The first place to look is the [Python Package Index](http://pypi.python.org/pypi) → <http://pypi.python.org/pypi>.
- If you can't find anything relevant in the Package Index, try [searching python.org](http://www.python.org/search/) → <http://www.python.org/search/> - you can find anything mentioned on the Python site, in the [FAQs](http://www.python.org/doc/FAQs/) → <http://www.python.org/doc/FAQs/>, or in the newsgroup. More info: [where to search](http://www.python.org/search/) → <http://www.python.org/search/>.
- You may also try our external guest project, pydoc.net → <http://pydoc.net/>, for advanced package and module search.
- Next, try [Google](http://www.google.com/) → <http://www.google.com/> or another search engine of your choice. Searching for "python" and some relevant keywords will usually find something helpful.
- Finally, you can try posting a query to the comp.lang.python Usenet group.
- Python: [Collection of 11 Best Python Cheat Sheets](https://blog.finxter.com/collection-5-cheat-sheets-every-python-coder-must-own/) → <https://blog.finxter.com/collection-5-cheat-sheets-every-python-coder-must-own/>
- NumPy - <https://wiki.python.org/moin/NumPy>: [Collection of 10 Best NumPy Cheat Sheets](https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/) → <https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/>
- Pandas: [Collection of 7 Beautiful Pandas Cheat Sheets](https://blog.finxter.com/pandas-cheat-sheets/) → <https://blog.finxter.com/pandas-cheat-sheets/>
- Machine Learning: [Collection of 15 Machine Learning Cheat Sheets](https://blog.finxter.com/machine-learning-cheat-sheets/) → <https://blog.finxter.com/machine-learning-cheat-sheets/>

Want to contribute?

- Python is a product of the [Python Software Foundation](http://www.python.org/psf/) → <http://www.python.org/psf/>, a non-profit organization that holds the copyright. [Donations to the PSF](http://www.python.org/psf/donations/) → <http://www.python.org/psf/donations/> are tax-deductible in the USA, and you can donate via credit card or [PayPal](https://wiki.python.org/moin/PayPal).
- To report a bug in the Python core, use the [Python Bug Tracker](http://bugs.python.org/) → <http://bugs.python.org/>.
- To contribute a bug fix or other patch to the Python core, read the [Python Developer's Guide](http://www.python.org/dev/) → <http://www.python.org/dev/> for more information about Python's development process.

- To contribute to the official [Python documentation](http://www.python.org/doc/) → <http://www.python.org/doc/>, join the [Documentation SIG](http://www.python.org/sigs/doc-sig/) → <http://www.python.org/sigs/doc-sig/>, write to docs@python.org → <mailto:docs@python.org>, or use the [Issue Tracker](http://bugs.python.org/) → <http://bugs.python.org/> to contribute a documentation patch.
 - To announce your module or application to the Python community, use comp.lang.python.announce → comp.lang.python.announce. See [the guide to Python mailing lists](http://www.python.org/community/lists/#comp-lang:python-announce) → <http://www.python.org/community/lists/#comp-lang:python-announce> for more information.
 - To propose changes to the Python core, post your thoughts to comp.lang.python → comp.lang.python. If you have an implementation, follow the [Python Patch Guidelines](http://www.python.org/patches/) → <http://www.python.org/patches/>.
 - If you have a question are not sure where to report it, check out the [WhereDoIReportThis?](https://wiki.python.org/moin/WhereDoIReportThis?) → <https://wiki.python.org/moin/WhereDoIReportThis%3F> page.
-

[CategoryDocumentation](https://wiki.python.org/moin/CategoryDocumentation) → <https://wiki.python.org/moin/CategoryDocumentation>

BeginnersGuide - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide>

Beginner's Guide to Python

New to programming? Python is free and easy to learn if you know where to start! This guide will help you to get started quickly.

[Chinese Translation/](#) → <https://wiki.python.org/moin/BeginnersGuideChinese>

New to Python?

Read [BeginnersGuide/Overview](#) → <https://wiki.python.org/moin/BeginnersGuide/Overview> for a short explanation of what Python is.

Getting Python

Next, install the Python 3 interpreter on your computer. This is the program that reads Python programs and carries out their instructions; you need it before you can do any Python programming. Mac and Linux distributions may include an outdated version of Python (Python 2), but you should install an updated one (Python 3). See [BeginnersGuide/Download](#) → <https://wiki.python.org/moin/BeginnersGuide/Download> for instructions to download the correct version of Python.

There are also Python interpreter and IDE bundles available, such as [Thonny](#) → <http://thonny.org/>. Other options can be found at [IntegratedDevelopmentEnvironments](#) → <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>.

At some stage, you'll want to edit and save your program code. Take a look at [HowToEditPythonCode](#) → <https://wiki.python.org/moin/HowToEditPythonCode> for some advice and recommendations.

Learning Python

Next, read a tutorial and try some simple experiments with your new Python interpreter.

- If you have never programmed before, see [BeginnersGuide/NonProgrammers](#) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers> for a list of suitable tutorials.
- If you have previous programming experience, consult [BeginnersGuide/Programmers](#) → <https://wiki.python.org/moin/BeginnersGuide/Programmers>, which lists more advanced tutorials.
- If English isn't your first language, you might be more comfortable with a tutorial that's been translated into your language. Consult python.org's list of Non-English resources → <https://wiki.python.org/moin/Languages>.

Most tutorials assume you know how to run a program on your computer. If you are using Windows and need help with this, see [How do I Run a Program Under Windows](#) → <http://www.python.org/doc/faq/windows/#how-do-i-run-a-python-program-under-windows>.

Here are some sites that focus on beginners and offer in-browser coding:

- [Beginners Python tutorial](#) → <https://python.land/python-tutorial> at Python Land (free)
- [Codéex](#) → <https://www.codedex.io/python> (non-free)
- [Coding Bootcamps](#) → <https://www.coding-bootcamps.com/> (non-free)
- [DataCamp](#) → <https://www.datacamp.com/> (non-free)
- [Dataquest](#) → <https://www.dataquest.io/> for Python for data science. (free)

- [HackInScience](https://www.hackinscience.org/) → <https://www.hackinscience.org/> free and open source platform.
- [High School Technology Services](https://www.mybsts.org/) → <https://www.mybsts.org/> for general Python (non-free)

Print a [cheat sheet](https://blog.finxter.com/python-cheat-sheet/) → <https://blog.finxter.com/python-cheat-sheet/> of the most important Python features and post it to your office wall until you know the basics well.

Once you have read a tutorial, you can browse through [Python's online documentation](http://docs.python.org/) → <http://docs.python.org/>. It includes a [tutorial](http://docs.python.org/tut/) → <http://docs.python.org/tut/> that might come in handy, a [Library Reference](http://docs.python.org/lib/) → <http://docs.python.org/lib/> that lists all of the modules that come standard with Python, and the [Language Reference](http://docs.python.org/ref/) → <http://docs.python.org/ref/> for a complete (if rather dry) explanation of Python's syntax.

When you are ready to write your first program, you will need a [text editor](https://wiki.python.org/moin/PythonEditors) → <https://wiki.python.org/moin/PythonEditors> or an [IDE](https://wiki.python.org/moin/IntegratedDevelopmentEnvironments) → <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>. If you don't want to use Thonny or something more advanced, then you can use [IDLE](https://docs.python.org/3/library/idle.html) → <https://docs.python.org/3/library/idle.html>, which is bundled with Python and supports [extensions](http://idlex.sourceforge.net/) → <http://idlex.sourceforge.net/>.

This Python wiki also contains a page about [Python One-Liners](https://wiki.python.org/moin/Powerful%20Python%20One-Liners) → <https://wiki.python.org/moin/Powerful%20Python%20One-Liners> -- an obscure but interesting subculture in Python.

Need Help?

Need help with any of this? Read [BeginnersGuide/Help](https://wiki.python.org/moin/BeginnersGuide/Help) → <https://wiki.python.org/moin/BeginnersGuide/Help> for mailing lists and newsgroups.

Most Python books will include an introduction to the language; see [IntroductoryBooks](https://wiki.python.org/moin/IntroductoryBooks) → <https://wiki.python.org/moin/IntroductoryBooks> for suggested titles.

Consult [BeginnersGuide/Examples](https://wiki.python.org/moin/BeginnersGuide/Examples) → <https://wiki.python.org/moin/BeginnersGuide/Examples> for small programs and little snippets of code that can help you learn.

Or, if you prefer to learn Python through listening to a lecture, you can attend a training course or even hire a trainer to come to your company. Consult the [PythonEvents](https://wiki.python.org/moin/PythonEvents) → <https://wiki.python.org/moin/PythonEvents> page to see if any training courses are scheduled in your area and the [PythonTraining](https://wiki.python.org/moin/PythonTraining) → <https://wiki.python.org/moin/PythonTraining> page for a list of trainers.

Teachers can join the [EDU-SIG](http://www.python.org/sigs/edu-sig/) → <http://www.python.org/sigs/edu-sig/>, a mailing list for discussion of Python's use in teaching at any level ranging from K-12 up to university.

Complete list of Beginner's Guide pages

1. [BeginnersGuide/Download](https://wiki.python.org/moin/BeginnersGuide/Download) → <https://wiki.python.org/moin/BeginnersGuide/Download>
2. [BeginnersGuide/Examples](https://wiki.python.org/moin/BeginnersGuide/Examples) → <https://wiki.python.org/moin/BeginnersGuide/Examples>
3. [BeginnersGuide/Help](https://wiki.python.org/moin/BeginnersGuide/Help) → <https://wiki.python.org/moin/BeginnersGuide/Help>
4. [BeginnersGuide/Mathematics](https://wiki.python.org/moin/BeginnersGuide/Mathematics) → <https://wiki.python.org/moin/BeginnersGuide/Mathematics>
5. [BeginnersGuide/NonProgrammers](https://wiki.python.org/moin/BeginnersGuide/NonProgrammers) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers>
6. [BeginnersGuide/NonProgrammersChinese](https://wiki.python.org/moin/BeginnersGuide/NonProgrammersChinese) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammersChinese>
7. [BeginnersGuide/Overview](https://wiki.python.org/moin/BeginnersGuide/Overview) → <https://wiki.python.org/moin/BeginnersGuide/Overview>
8. [BeginnersGuide/OverviewChinese](https://wiki.python.org/moin/BeginnersGuide/OverviewChinese) → <https://wiki.python.org/moin/BeginnersGuide/OverviewChinese>
9. [BeginnersGuide/Programmers](https://wiki.python.org/moin/BeginnersGuide/Programmers) → <https://wiki.python.org/moin/BeginnersGuide/Programmers>
10. [BeginnersGuide/Programmers \(Cpp2Python.pdf\)](https://wiki.python.org/moin/BeginnersGuide/Programmers(Cpp2Python.pdf)) → <https://wiki.python.org/moin/BeginnersGuide/Programmers?action=AttachFile&do=view&target=Cpp2Python.pdf>
11. [BeginnersGuide/Programmers/SimpleExamples](https://wiki.python.org/moin/BeginnersGuide/Programmers/SimpleExamples) → <https://wiki.python.org/moin/BeginnersGuide/Programmers/SimpleExamples>

Quiz and Exercises

- Finxter - [How good are your Python skills? Test and Training with a Daily Python Puzzle](https://finxter.com/) → <https://finxter.com/>
- CheckIO - [Online learning, testing and improving your python skills](http://www.checkio.org/) → <http://www.checkio.org/>
- After Hours Programming - [Python Quiz](http://www.afterhoursprogramming.com/tutorial/Python/Python-Quiz/) → <http://www.afterhoursprogramming.com/tutorial/Python/Python-Quiz/>
- PyGUI - [Collection of python quiz answers, Examples And GUI Tkinter Tutorials For Beginners](http://www.pythongui.com/) → <http://www.pythongui.com/>
- Pythonspot - [Python Quiz](https://pythonspot.com/python-tests-quizes/) → <https://pythonspot.com/python-tests-quizes/>
- Python Challenge - [A Python Quiz App on Android Platform](https://play.google.com/store/apps/details?id=sg.apps.garden.pythonchallenge) → <https://play.google.com/store/apps/details?id=sg.apps.garden.pythonchallenge>
- CS Circles - [online lessons and graded exercises](http://cscircles.cemc.uwaterloo.ca/) → <http://cscircles.cemc.uwaterloo.ca/>

Python Style Checker

- [Pythonchecker.com](http://pythonchecker.com/) → <http://pythonchecker.com/> - An educative online tool to rate your Python style (with dynamic score computation and hints)

Looking for a particular Python module or application?

- The first place to look is the [Python Package Index](http://pypi.python.org/pypi) → <http://pypi.python.org/pypi>.
- If you can't find anything relevant in the Package Index, try [searching python.org](http://www.python.org/search/) → <http://www.python.org/search/> - you can find anything mentioned on the Python site, in the [FAQs](http://www.python.org/doc/faq/) → <http://www.python.org/doc/faq/>, or in the newsgroup. More info: [where to search](http://www.python.org/search/) → <http://www.python.org/search/>.
- You may also try our external guest project, pydoc.net → <http://pydoc.net/>, for advanced package and module search.
- Next, try [Google](http://www.google.com/) → <http://www.google.com/> or another search engine of your choice. Searching for "python" and some relevant keywords will usually find something helpful.
- Finally, you can try posting a query to the comp.lang.python Usenet group.
- Python: [Collection of 11 Best Python Cheat Sheets](https://blog.finxter.com/collection-5-cheat-sheets-every-python-coder-must-own/) → <https://blog.finxter.com/collection-5-cheat-sheets-every-python-coder-must-own/>
- NumPy - <https://wiki.python.org/moin/NumPy>: [Collection of 10 Best NumPy Cheat Sheets](https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/) → <https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/>
- Pandas: [Collection of 7 Beautiful Pandas Cheat Sheets](https://blog.finxter.com/pandas-cheat-sheets/) → <https://blog.finxter.com/pandas-cheat-sheets/>
- Machine Learning: [Collection of 15 Machine Learning Cheat Sheets](https://blog.finxter.com/machine-learning-cheat-sheets/) → <https://blog.finxter.com/machine-learning-cheat-sheets/>

Want to contribute?

- Python is a product of the [Python Software Foundation](http://www.python.org/psf/) → <http://www.python.org/psf/>, a non-profit organization that holds the copyright. [Donations to the PSF](http://www.python.org/psf/donations/) → <http://www.python.org/psf/donations/> are tax-deductible in the USA, and you can donate via credit card or [PayPal](https://wiki.python.org/moin/PayPal).
- To report a bug in the Python core, use the [Python Bug Tracker](http://bugs.python.org/) → <http://bugs.python.org/>.
- To contribute a bug fix or other patch to the Python core, read the [Python Developer's Guide](http://www.python.org/dev/) → <http://www.python.org/dev/> for more information about Python's development process.

- To contribute to the official [Python documentation](http://www.python.org/doc/) → <http://www.python.org/doc/>, join the [Documentation SIG](http://www.python.org/sigs/doc-sig/) → <http://www.python.org/sigs/doc-sig/>, write to docs@python.org → <mailto:docs@python.org> , or use the [Issue Tracker](http://bugs.python.org/) → <http://bugs.python.org/> to contribute a documentation patch.
 - To announce your module or application to the Python community, use [comp.lang.python.announce](#) → [comp.lang.python.announce](#). See [the guide to Python mailing lists](#) → <http://www.python.org/community/lists/#comp-lang:python-announce> for more information.
 - To propose changes to the Python core, post your thoughts to [comp.lang.python](#) → [comp.lang.python](#). If you have an implementation, follow the [Python Patch Guidelines](#) → <http://www.python.org/patches/>.
 - If you have a question are not sure where to report it, check out the [WhereDoIReportThis?](#) → <https://wiki.python.org/moin/WhereDoIReportThis%3F> page.
-

[CategoryDocumentation](#) → <https://wiki.python.org/moin/CategoryDocumentation>

FrontPage - Python Wiki

Source: <https://wiki.python.org/moin/FrontPage>

Welcome to the Python Wiki, a user-editable compendium of knowledge based around the Python programming language. Some pages are protected against casual editing - see [WikiEditingGuidelines](#) → <https://wiki.python.org/moin/WikiEditingGuidelines> for more information about editing content.

[Python](#) → <http://www.python.org/> is a great object-oriented, interpreted, and interactive programming language. It is often compared → <https://wiki.python.org/moin/LanguageComparisons> (favorably of course 😊)

) to Lisp, Tcl, Perl, Ruby, C#, Visual Basic, Visual Fox Pro, Scheme or Java... and it's much more fun.

Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems → <https://wiki.python.org/moin/GUI%20Programming%20in%20Python>. New built-in modules are easily written in C or C++ (or other languages, depending on the chosen implementation → <https://wiki.python.org/moin/PythonImplementations>). Python is also usable as an extension language for applications written in other languages → <https://wiki.python.org/moin/AppsWithPythonScripting> that need easy-to-use scripting or automation interfaces.

Getting Started

- [Python Conferences](#) → <https://wiki.python.org/moin/PythonConferences> - information about the Python conference scene
- [Local User Groups](#) → <https://wiki.python.org/moin/LocalUserGroups> - find a Python group near you
- [Python Training](#) → <https://wiki.python.org/moin/PythonTraining> - Python training courses
- [Python Events](#) → <https://wiki.python.org/moin/PythonEvents> - event listing for conferences, training courses and more
- [Python Event Calendars](#) → <https://wiki.python.org/moin/PythonEventsCalendar> - calendars for Python conferences and user groups
- [Participating in the Community](#) → <https://wiki.python.org/moin/Community> - where people using and producing Python get together
- [Python Software Foundation](#) → <https://wiki.python.org/moin/PythonSoftwareFoundation> - show your support by joining the Foundation behind Python
- [Find a job where you can use Python](#) → <https://wiki.python.org/moin/PythonJobs> - Python job boards around the world

Python Software

Using this Wiki

This Wiki is a community place to gather and organize all things about Python. Feel free to exercise your editorial skills and expertise to make it a useful knowledge base and up-to-date reference on all Python-related topics.

There are some [guidelines](#) → <https://wiki.python.org/moin/WikiGuidelines> describing the policies and rules governing this Wiki and how you can most effectively contribute to it. A list of [site improvements](#) → <https://wiki.python.org/moin/SiteImprovements> describes various tasks where your help would be appreciated. To keep up with changes on this site, check [RecentChanges](#) → <https://wiki.python.org/moin/RecentChanges> frequently or follow it using RSS: [RSS feed](#) → https://wiki.python.org/moin/FrontPage?action=rss_rc.

Creating a Wiki account

In order to sign up for a wiki account, please go to the [Create new account → https://wiki.python.org/moin/FrontPage?action=newaccount](https://wiki.python.org/moin/FrontPage?action=newaccount) form, enter your account name (using the format FirstnameLastname to avoid issues - please don't use spaces in the name) and provide a password, plus email address (for password recovery).

Editing pages

Since spamming and vandalism on this wiki had reached a level that required constant intervention, unfamiliar users are no longer allowed to edit pages. However all you need to do is introduce yourself to the wiki admin group to become an editor.

If you want to edit a page and have just signed up, or find that you can no longer edit a page that you could edit before, please write to the [pydotorg-www mailing list → mailto:pydotorg-www@python.org](mailto:pydotorg-www@python.org) describing what you would like to edit, and we'll add you to the [EditorsGroup → https://wiki.python.org/moin/EditorsGroup](https://wiki.python.org/moin/EditorsGroup). **Please include your account name (wiki name) in this message.**

Sorry for any inconvenience, but we want to keep this wiki a useful tool for the community, while at the same time preventing the wiki admins from burning out cleaning up junk.

Reporting problems

In case of emergency, please contact the [python.org maintainers → mailto:python.org_maintainers@python.org](mailto:python.org_maintainers@python.org), or if experiencing difficulties, contact the [pydotorg-www mailing list → mailto:pydotorg-www@python.org](mailto:pydotorg-www@python.org) to say "help".

Wiki Attack in January 2013

The wiki was subject to an attack on January 5 2013. Since it was not clear whether user account data was stolen, all passwords were subsequently reset, so you will have to use the [password recovery function → https://wiki.python.org/moin/FrontPage?action=recoverpass](https://wiki.python.org/moin/FrontPage?action=recoverpass) to get a new password.

See the [wiki attack description page → https://wiki.python.org/moin/WikiAttack2013](https://wiki.python.org/moin/WikiAttack2013) for more details. If you find problems, please report them to the pydotorg-www mailing list <pydotorg-www@python.org>.

HTTPS access to the Wiki

We have enabled HTTPS access to the wiki to further enhance security and avoid having to send clear text passwords over the network in order to log in to the wikis.

If you have not been using HTTPS links to the wiki login page, please be advised that your password may have been sniffed on the network at e.g. a conference. It is best to [change it → https://wiki.python.org/moin/FrontPage?action=userprefs](https://wiki.python.org/moin/FrontPage?action=userprefs) and stop using HTTP links to the wiki login page.

RecentChanges - Python Wiki

Source: <https://wiki.python.org/moin/RecentChanges>

The menu bar can be changed in [UserPreferences](https://wiki.python.org/moin/UserPreferences) → <https://wiki.python.org/moin/UserPreferences>.



[RSS]



[DIFF]



[INFO]



[DIFF]



[INFO]



[DIFF]



[INFO]



[DIFF]



[INFO]



[DIFF]



[INFO]



[DIFF]

 [INFO]

 [DIFF]

 [INFO]

 [DIFF]

 [INFO]

2024-03-08

[LocalUserGroups](#) → <https://wiki.python.org/moin/LocalUserGroups> 08:05 [CalPaterson](#) → <https://wiki.python.org/moin/CalPaterson> #01 add archipylago (turku, finland)
#02 Add HelPY
erson [1-2]

2024-03-07

[EditorsGroup](#) → <https://wiki.python.org/moin/EditorsGroup> 17:44 [MatsWichmann](#) → <http://wiki.python.org/moin/MatsWichmann>

[Python Software Foundation/Fellow Nominations](#) → <https://wiki.python.org/moin/PythonSoftwareFoundation/FellowNominations> 15:21 [ThomasWouters](#) → <http://wiki.python.org/moin/ThomasWouters>

2024-03-05

[Handling Exceptions](#) → <https://wiki.python.org/moin/HandlingExceptions> 20:29 [MatsWichmann](#) → <http://wiki.python.org/moin/MatsWichmann> Fix some dead/unintended links; use print()

2024-03-03

[Beginners Guide/Programmers](#) → <https://wiki.python.org/moin/BeginnersGuide/Programmers> 16:02 [MatsWichmann](#) → <http://wiki.python.org/moin/MatsWichmann> [1]
[ranjitbhatta](#) → <https://wiki.python.org/moin/ranjitbhatta> [2] #01 Fix markup error in recent addition - does not deal with question if this is the right bucket for this entry, or whether it's titled suitably (I consider both answers to be no)

2024-02-27

[WebFrameworks](#) → <https://wiki.python.org/moin/WebFrameworks> 12:48 [JuhaKoskelainen](#) → <http://wiki.python.org/moin/JuhaKoskelainen> Update versions

2024-02-25

[GuiProgramming](#) → <https://wiki.python.org/moin/GuiProgramming> 07:39 [krakengore](#) → <https://wiki.python.org/moin/kraken> #02 Added a missing framework for GUI development
[1-2]

2024-02-21

[Python Consulting](#) → <https://wiki.python.org/moin/PythonConsulting> 06:11 [Riddhesh](#) → <https://wiki.python.org/moin/Riddhesh> [1-2]

marks older pages that have at least one backup version stored (click for an author diff)



[DIFF]

marks pages edited since you set your bookmark (click for a bookmark diff)



[UPDATED]

marks pages created since you set your bookmark, and were not edited after creation



[NEW]

marks page deletions



[DELETED]

marks page renames



[RENAMEDE]

 An editing conflict happened, please resolve it by merging both versions of the problematic paragraphs together.

[CONFLICT]

FindPage - Python Wiki

Source: <https://wiki.python.org/moin/FindPage>

You can use this page to search all entries in this [WikiWikiWeb](#) → <https://wiki.python.org/moin/WikiWikiWeb>. Searches are not case sensitive.

Good starting points to explore a wiki are:

- [RecentChanges](#) → <https://wiki.python.org/moin/RecentChanges>: see where people are currently working
- FindPage: search or browse the database in various ways
- [TitleIndex](#) → <https://wiki.python.org/moin/TitleIndex>: a list of all 3426 pages in the wiki
- [WordIndex](#) → <https://wiki.python.org/moin/WordIndex>: a list of all words that are part of page title (thus, a list of the concepts in a wiki)
- [WikiSandBox](#) → <https://wiki.python.org/moin/WikiSandBox>: feel free to change this page and experiment with editing

Here's a title search. Try something like *manager*:

•

Here's a full-text search.

•

You can also use regular expressions, such as

seriali[sz]e

Or go direct to a page, or create a new page by entering its name here:

•

HelpContents - Python Wiki

Source: <https://wiki.python.org/moin/HelpContents>

Help Contents

Welcome to [MoinMoin](#) → <https://wiki.python.org/moin/MoinMoin>. You will find here the help pages for the wiki system itself.

If you would like a quick overview of [MoinMoin](#) → <https://wiki.python.org/moin/MoinMoin>'s syntax, have a look at [HelpOnMoinWikiSyntax](#) → <https://wiki.python.org/moin/HelpOnMoinWikiSyntax>. If you are looking for something for a presentation, look at [WikiCourse](#) → <https://wiki.python.org/moin/WikiCourse>.

- [HelpForUsers](#) → <https://wiki.python.org/moin/HelpForUsers> - is help for users who are new to a [MoinMoin](#) → <https://wiki.python.org/moin/MoinMoin> wiki.
- [HelpOnAdministration](#) → <https://wiki.python.org/moin/HelpOnAdministration> - how to configure and maintain a [MoinMoin](#) → <https://wiki.python.org/moin/MoinMoin> wiki
- [HelpIndex](#) → <https://wiki.python.org/moin/HelpIndex> - is a list of all available help pages.



Fulltext search is available on [FindPage](#) → <https://wiki.python.org/moin/FindPage>.

If you find any errors on the help pages, please describe them on [HelpErrata](#) → <http://moinmo.in/HelpErrata>. Please do not edit or create help pages in other wikis than MoinMaster (see [HelpContents](#) → <http://master.moinmo.in/HelpContents>), because the pages from MoinMaster will overwrite any other changes on wiki engine upgrades. Please follow the established structure of help pages if you want to change pages in MoinMaster. Before doing any major or widespread changes please discuss that on the mailing list or MoinMoin wiki.

BeginnersGuide - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide>

Beginner's Guide to Python

New to programming? Python is free and easy to learn if you know where to start! This guide will help you to get started quickly.

[Chinese Translation/](#) → <https://wiki.python.org/moin/BeginnersGuideChinese>

New to Python?

Read [BeginnersGuide/Overview](#) → <https://wiki.python.org/moin/BeginnersGuide/Overview> for a short explanation of what Python is.

Getting Python

Next, install the Python 3 interpreter on your computer. This is the program that reads Python programs and carries out their instructions; you need it before you can do any Python programming. Mac and Linux distributions may include an outdated version of Python (Python 2), but you should install an updated one (Python 3). See [BeginnersGuide/Download](#) → <https://wiki.python.org/moin/BeginnersGuide/Download> for instructions to download the correct version of Python.

There are also Python interpreter and IDE bundles available, such as [Thonny](#) → <http://thonny.org/>. Other options can be found at [IntegratedDevelopmentEnvironments](#) → <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>.

At some stage, you'll want to edit and save your program code. Take a look at [HowToEditPythonCode](#) → <https://wiki.python.org/moin/HowToEditPythonCode> for some advice and recommendations.

Learning Python

Next, read a tutorial and try some simple experiments with your new Python interpreter.

- If you have never programmed before, see [BeginnersGuide/NonProgrammers](#) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers> for a list of suitable tutorials.
- If you have previous programming experience, consult [BeginnersGuide/Programmers](#) → <https://wiki.python.org/moin/BeginnersGuide/Programmers>, which lists more advanced tutorials.
- If English isn't your first language, you might be more comfortable with a tutorial that's been translated into your language. Consult python.org's list of Non-English resources → <https://wiki.python.org/moin/Languages>.

Most tutorials assume you know how to run a program on your computer. If you are using Windows and need help with this, see [How do I Run a Program Under Windows](#) → <http://www.python.org/doc/faq/windows/#how-do-i-run-a-python-program-under-windows>.

Here are some sites that focus on beginners and offer in-browser coding:

- [Beginners Python tutorial](#) → <https://python.land/python-tutorial> at Python Land (free)
- [Codéex](#) → <https://www.codedex.io/python> (non-free)
- [Coding Bootcamps](#) → <https://www.coding-bootcamps.com/> (non-free)
- [DataCamp](#) → <https://www.datacamp.com/> (non-free)
- [Dataquest](#) → <https://www.dataquest.io/> for Python for data science. (free)

- [HackInScience](https://www.hackinscience.org/) → <https://www.hackinscience.org/> free and open source platform.
- [High School Technology Services](https://www.mybsts.org/) → <https://www.mybsts.org/> for general Python (non-free)

Print a [cheat sheet](https://blog.finxter.com/python-cheat-sheet/) → <https://blog.finxter.com/python-cheat-sheet/> of the most important Python features and post it to your office wall until you know the basics well.

Once you have read a tutorial, you can browse through [Python's online documentation](http://docs.python.org/) → <http://docs.python.org/>. It includes a [tutorial](http://docs.python.org/tut/) → <http://docs.python.org/tut/> that might come in handy, a [Library Reference](http://docs.python.org/lib/) → <http://docs.python.org/lib/> that lists all of the modules that come standard with Python, and the [Language Reference](http://docs.python.org/ref/) → <http://docs.python.org/ref/> for a complete (if rather dry) explanation of Python's syntax.

When you are ready to write your first program, you will need a [text editor](https://wiki.python.org/moin/PythonEditors) → <https://wiki.python.org/moin/PythonEditors> or an [IDE](https://wiki.python.org/moin/IntegratedDevelopmentEnvironments) → <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>. If you don't want to use Thonny or something more advanced, then you can use [IDLE](https://docs.python.org/3/library/idle.html) → <https://docs.python.org/3/library/idle.html>, which is bundled with Python and supports [extensions](http://idlex.sourceforge.net/) → <http://idlex.sourceforge.net/>.

This Python wiki also contains a page about [Python One-Liners](https://wiki.python.org/moin/Powerful%20Python%20One-Liners) → <https://wiki.python.org/moin/Powerful%20Python%20One-Liners> -- an obscure but interesting subculture in Python.

Need Help?

Need help with any of this? Read [BeginnersGuide/Help](https://wiki.python.org/moin/BeginnersGuide/Help) → <https://wiki.python.org/moin/BeginnersGuide/Help> for mailing lists and newsgroups.

Most Python books will include an introduction to the language; see [IntroductoryBooks](https://wiki.python.org/moin/IntroductoryBooks) → <https://wiki.python.org/moin/IntroductoryBooks> for suggested titles.

Consult [BeginnersGuide/Examples](https://wiki.python.org/moin/BeginnersGuide/Examples) → <https://wiki.python.org/moin/BeginnersGuide/Examples> for small programs and little snippets of code that can help you learn.

Or, if you prefer to learn Python through listening to a lecture, you can attend a training course or even hire a trainer to come to your company. Consult the [PythonEvents](https://wiki.python.org/moin/PythonEvents) → <https://wiki.python.org/moin/PythonEvents> page to see if any training courses are scheduled in your area and the [PythonTraining](https://wiki.python.org/moin/PythonTraining) → <https://wiki.python.org/moin/PythonTraining> page for a list of trainers.

Teachers can join the [EDU-SIG](http://www.python.org/sigs/edu-sig/) → <http://www.python.org/sigs/edu-sig/>, a mailing list for discussion of Python's use in teaching at any level ranging from K-12 up to university.

Complete list of Beginner's Guide pages

1. [BeginnersGuide/Download](https://wiki.python.org/moin/BeginnersGuide/Download) → <https://wiki.python.org/moin/BeginnersGuide/Download>
2. [BeginnersGuide/Examples](https://wiki.python.org/moin/BeginnersGuide/Examples) → <https://wiki.python.org/moin/BeginnersGuide/Examples>
3. [BeginnersGuide/Help](https://wiki.python.org/moin/BeginnersGuide/Help) → <https://wiki.python.org/moin/BeginnersGuide/Help>
4. [BeginnersGuide/Mathematics](https://wiki.python.org/moin/BeginnersGuide/Mathematics) → <https://wiki.python.org/moin/BeginnersGuide/Mathematics>
5. [BeginnersGuide/NonProgrammers](https://wiki.python.org/moin/BeginnersGuide/NonProgrammers) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers>
6. [BeginnersGuide/NonProgrammersChinese](https://wiki.python.org/moin/BeginnersGuide/NonProgrammersChinese) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammersChinese>
7. [BeginnersGuide/Overview](https://wiki.python.org/moin/BeginnersGuide/Overview) → <https://wiki.python.org/moin/BeginnersGuide/Overview>
8. [BeginnersGuide/OverviewChinese](https://wiki.python.org/moin/BeginnersGuide/OverviewChinese) → <https://wiki.python.org/moin/BeginnersGuide/OverviewChinese>
9. [BeginnersGuide/Programmers](https://wiki.python.org/moin/BeginnersGuide/Programmers) → <https://wiki.python.org/moin/BeginnersGuide/Programmers>
10. [BeginnersGuide/Programmers \(Cpp2Python.pdf\)](https://wiki.python.org/moin/BeginnersGuide/Programmers(Cpp2Python.pdf)) → <https://wiki.python.org/moin/BeginnersGuide/Programmers?action=AttachFile&do=view&target=Cpp2Python.pdf>
11. [BeginnersGuide/Programmers/SimpleExamples](https://wiki.python.org/moin/BeginnersGuide/Programmers/SimpleExamples) → <https://wiki.python.org/moin/BeginnersGuide/Programmers/SimpleExamples>

Quiz and Exercises

- Finxter - [How good are your Python skills? Test and Training with a Daily Python Puzzle](https://finxter.com/) → <https://finxter.com/>
- CheckIO - [Online learning, testing and improving your python skills](http://www.checkio.org/) → <http://www.checkio.org/>
- After Hours Programming - [Python Quiz](http://www.afterhoursprogramming.com/tutorial/Python/Python-Quiz/) → <http://www.afterhoursprogramming.com/tutorial/Python/Python-Quiz/>
- PyGUI - [Collection of python quiz answers, Examples And GUI Tkinter Tutorials For Beginners](http://www.pythongui.com/) → <http://www.pythongui.com/>
- Pythonspot - [Python Quiz](https://pythonspot.com/python-tests-quizes/) → <https://pythonspot.com/python-tests-quizes/>
- Python Challenge - [A Python Quiz App on Android Platform](https://play.google.com/store/apps/details?id=sg.apps.garden.pythonchallenge) → <https://play.google.com/store/apps/details?id=sg.apps.garden.pythonchallenge>
- CS Circles - [online lessons and graded exercises](http://cscircles.cemc.uwaterloo.ca/) → <http://cscircles.cemc.uwaterloo.ca/>

Python Style Checker

- [Pythonchecker.com](http://pythonchecker.com/) → <http://pythonchecker.com/> - An educative online tool to rate your Python style (with dynamic score computation and hints)

Looking for a particular Python module or application?

- The first place to look is the [Python Package Index](http://pypi.python.org/pypi) → <http://pypi.python.org/pypi>.
- If you can't find anything relevant in the Package Index, try [searching python.org](http://www.python.org/search/) → <http://www.python.org/search/> - you can find anything mentioned on the Python site, in the [FAQs](http://www.python.org/doc/FAQs/) → <http://www.python.org/doc/FAQs/>, or in the newsgroup. More info: [where to search](http://www.python.org/search/) → <http://www.python.org/search/>.
- You may also try our external guest project, pydoc.net → <http://pydoc.net/>, for advanced package and module search.
- Next, try [Google](http://www.google.com/) → <http://www.google.com/> or another search engine of your choice. Searching for "python" and some relevant keywords will usually find something helpful.
- Finally, you can try posting a query to the comp.lang.python Usenet group.
- Python: [Collection of 11 Best Python Cheat Sheets](https://blog.finxter.com/collection-5-cheat-sheets-every-python-coder-must-own/) → <https://blog.finxter.com/collection-5-cheat-sheets-every-python-coder-must-own/>
- [NumPy](https://wiki.python.org/moin/NumPy) → <https://wiki.python.org/moin/NumPy>: [Collection of 10 Best NumPy Cheat Sheets](https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/) → <https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/>
- Pandas: [Collection of 7 Beautiful Pandas Cheat Sheets](https://blog.finxter.com/pandas-cheat-sheets/) → <https://blog.finxter.com/pandas-cheat-sheets/>
- Machine Learning: [Collection of 15 Machine Learning Cheat Sheets](https://blog.finxter.com/machine-learning-cheat-sheets/) → <https://blog.finxter.com/machine-learning-cheat-sheets/>

Want to contribute?

- Python is a product of the [Python Software Foundation](http://www.python.org/psf/) → <http://www.python.org/psf/>, a non-profit organization that holds the copyright. [Donations to the PSF](http://www.python.org/psf/donations/) → <http://www.python.org/psf/donations/> are tax-deductible in the USA, and you can donate via credit card or [PayPal](https://wiki.python.org/moin/PayPal).
- To report a bug in the Python core, use the [Python Bug Tracker](http://bugs.python.org/) → <http://bugs.python.org/>.
- To contribute a bug fix or other patch to the Python core, read the [Python Developer's Guide](http://www.python.org/dev/) → <http://www.python.org/dev/> for more information about Python's development process.

- To contribute to the official [Python documentation](http://www.python.org/doc/) → <http://www.python.org/doc/>, join the [Documentation SIG](http://www.python.org/sigs/doc-sig/) → <http://www.python.org/sigs/doc-sig/>, write to docs@python.org → <mailto:docs@python.org>, or use the [Issue Tracker](http://bugs.python.org/) → <http://bugs.python.org/> to contribute a documentation patch.
 - To announce your module or application to the Python community, use comp.lang.python.announce → comp.lang.python.announce. See the [guide to Python mailing lists](http://www.python.org/community/lists/#comp-lang:python-announce) → <http://www.python.org/community/lists/#comp-lang:python-announce> for more information.
 - To propose changes to the Python core, post your thoughts to comp.lang.python → comp.lang.python. If you have an implementation, follow the [Python Patch Guidelines](http://www.python.org/patches/) → <http://www.python.org/patches/>.
 - If you have a question are not sure where to report it, check out the [WhereDoIReportThis?](https://wiki.python.org/moin/WhereDoIReportThis?) → <https://wiki.python.org/moin/WhereDoIReportThis%3F> page.
-

[CategoryDocumentation](https://wiki.python.org/moin/CategoryDocumentation) → <https://wiki.python.org/moin/CategoryDocumentation>

Info for "BeginnersGuide" - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide?action=info>

#	Date	Size	Editor	Comment	Action
153	2023-08-29 20:15:52	8510	to previous eriky → https://wiki.pyt... → https://wik...hon.org/moin/eriky i.python.org/moin/Beginn...ersGuide?acti...on=diff&rev1...=152&rev2=1...53	Fix order	view → http://wiki.pytho...n.org/moin/Be...ginnersGuide?...action=recall&r...ev=153
152	2023-08-29 15:10:52	8608	to previous eriky → https://wiki.pyt... → https://wik...hon.org/moin/eriky i.python.org/moin/Beginn...ersGuide?acti...on=diff&rev1...=151&rev2=1...52	A clear distinction between free and non-free courses, remove a Python 2 course	view → http://wiki.pytho...n.org/moin/Be...ginnersGuide?...action=recall&r...ev=152
151	2023-06-04 09:17:35	8468	to previous SonnyLi → https://wik... → https://wik...i.python.org/moin/Sonn...yLi i.python.org/moin/Beginn...ersGuide?acti...on=diff&rev1...=150&rev2=1...51	Update course link. Keep the list alphabetical.	view → http://wiki.pytho...n.org/moin/Be...ginnersGuide?...action=recall&r...ev=151
150	2022-11-04 04:33:02	8461	to previous SonnyLi → https://wik... → https://wik...i.python.org/moin/Sonn...yLi i.python.org/moin/Beginn...ersGuide?acti...on=diff&rev1...=149&rev2=1...50	The list isn't in alphabetical order for Python Land.	view → http://wiki.pytho...n.org/moin/Be...ginnersGuide?...action=recall&r...ev=150
149	2022-11-04 02:59:11	8443	to previous AtmanAn → mailto:twi... → https://wik...nsant@gmail.com i.python.org/moin/Beginn...ersGuide?acti...on=diff&rev1...=148&rev2=1...49	Add Chinese text	view → http://wiki.pytho...n.org/moin/Be...ginnersGuide?...action=recall&r...ev=149

148	2022-11-02 23:53:08	8427	to previous → https://wikil.python.org/moin/SonnyLi	SonnyLi → https://wikil.python.org/moin/SonnyLi	Add www.codedex.io in the alphabetized list.	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=148
147	2022-09-27 19:37:45	8385	to previous → https://wikil.python.org/moin/SrinivasRamesh	Srinivas Ramesh → https://wikil.python.org/moin/SrinivasRamesh	removed broken links	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=147
146	2022-03-29 19:22:45	8562	to previous → https://wikil.python.org/moin/eriky	eriky → https://wikil.python.org/moin/eriky	added tutorial with interactive code samples	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=146
145	2022-01-13 15:35:20	8438	to previous → https://wikil.python.org/moin/JulienPalard	JulienPalard → https://wikil.python.org/moin/JulienPalard	FIX domains redirecting to www (myhsts.org has no certificate for https://myhsts.org)	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=145
144	2020-07-04 14:00:41	8430	to previous → https://wikil.python.org/moin/ChrisM	ChrisM → https://wikil.python.org/moin/ChrisM	Added Subheading "Python-Related Cheat Sheets"	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=144

143	2020-06-14 16:12:10	7864	to previous	ChrisM → https://wiki.python.org/moin/Chris	Added internal reference to wiki page (Python One-Liners entry).	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=143
142	2020-05-29 20:08:04	7717	to previous	JulienPalard → http://wiki.python.org/moin/JulienPalard	Display in-browser coding sites as a list, and add my hackinscience pet project.	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=142
-	2020-03-20 17:35:21	0	-	MarcAndreLemburg	ATTDEL: Python- → https://wiki.python.org/moin/MarcAndreLemburg	Doc&Start16DEC2010.pdf
-	2020-03-20 14:35:08	0	-	111		ATTNEW: Python- Doc&Start16DEC2010.pdf
-	2020-03-20 14:35:07	0	-	111		ATTDEL: Python- Doc&Start16DEC2010.pdf
141	2019-10-20 14:41:01	7570	to previous	MarcAndreLemburg	Remove unnecessary ACL → https://wiki.python.org/moin/MarcAndreLemburg	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=141
140	2019-10-19 21:02:19	7609	to previous	FrancesHocutt → http://wiki.python.org/moin/FrancesHocutt	Update installation instructions to be Py3 only	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=140

139	2019-09-18	7902	to previous	ChrisM → https://wiki.python.org/moin/ChrisBeginnersGuide?action=recall&rev1=138&rev2=1	Grammarly run (fixed writing style issues), added style checker resource.	view → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev=139
138	2019-07-08	7739	to previous	MattZand → https://wiki.python.org/moin/MattZand		view → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev=138
137	2019-01-13	7629	to previous	ChrisM → https://wiki.python.org/moin/ChrisBeginnersGuide?action=recall&rev1=136&rev2=1	Used grammarly plugin to spellcheck 6 small bugs.	view → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev=137
136	2018-12-14	7628	to previous	ChrisM → https://wiki.python.org/moin/ChrisBeginnersGuide?action=recall&rev1=135&rev2=1	Full run of Grammarly spell checking tool (found 2 bugs ;)	view → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev=136
135	2018-12-14	7628	to previous	ChrisM → https://wiki.python.org/moin/ChrisBeginnersGuide?action=recall&rev1=134&rev2=1		view → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev=135

134	2018-06-18 01:57:10	7635	to previous	FrankM → https://wiki.python.org/moin/Frank	Fixed bug moin/BeginnersGuide?action=recall&rev1=133&rev2=1 34	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=134&rev2=1
133	2018-06-18 01:55:44	7634	to previous	FrankM → https://wiki.python.org/moin/Frank	Added cheat sheet for beginners (Py key-words). moin/BeginnersGuide?action=recall&rev1=132&rev2=1 33	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=133
132	2018-06-18 01:47:23	7465	to previous	FrankM → https://wiki.python.org/moin/Frank	linktext change moin/BeginnersGuide?action=recall&rev1=131&rev2=1 32	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=132
131	2018-05-30 13:48:28	7432	to previous	FrankM → https://wiki.python.org/moin/Frank	fixed bug moin/BeginnersGuide?action=recall&rev1=130&rev2=1 31	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=131
130	2018-05-30 13:47:07	7431	to previous	FrankM → https://wiki.python.org/moin/Frank	New learning resource Finxter added moin/BeginnersGuide?action=recall&rev1=129&rev2=1 30	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=130

129	2017-09-10	7349	to previous	dvs1 → https://wiki.pythond.org/moin/dvs1	improved the informativity of the last sentence in the 'Learning Python' section	view → https://wiki.pythond.org/moin/BeginnersGuide?action=recall&rev=129
128	2017-09-10	7444	to previous	dvs1 → https://wiki.pythond.org/moin/dvs1	reduced bundle recommendation to Thonny; PyCharm is bloatware for beginners (and it's freemium Java based software); added link to IntegratedDevelopmentEnvironments	view → https://wiki.pythond.org/moin/BeginnersGuide?action=recall&rev=128
127	2017-09-10	7594	to previous	martijntheuwissen → https://wiki.python.org/moin/martijntheuwisse	Minor change (codecademy link straight to python)	view → https://wiki.pythond.org/moin/BeginnersGuide?action=recall&rev=127
126	2017-09-06	7579	to previous	AivarAnnamaa → https://wiki.python.org/moin/AivarAnnamaa	Changed Thonny's Python version	view → https://wiki.pythond.org/moin/BeginnersGuide?action=recall&rev=126
125	2017-08-16	7579	to previous	vikp → https://wiki.pythond.org/moin/vikp	grammar changes	view → https://wiki.pythond.org/moin/BeginnersGuide?action=recall&rev=125

124	2017-05-15	7444	to previous	alinowe → https://wiki.python.org/moin/alinowe	fixed typo → https://wik... python.org/moin/Begin... ersGuide?acti... on=diff&rev1=123&rev2=1	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=124
123	2017-05-10	7443	to previous	durga_prathap → http://.../moin/DurgaPrathap	python.org/moin/Begin... ersGuide?acti... on=diff&rev1=122&rev2=1	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=123
122	2017-05-10	7441	to previous	durga_prathap → http://.../moin/DurgaPrathap	python.org/moin/Begin... ersGuide?acti... on=diff&rev1=121&rev2=1	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=122
121	2017-05-10	7428	to previous	durga_prathap → http://.../moin/DurgaPrathap	python.org/moin/Begin... ersGuide?acti... on=diff&rev1=120&rev2=1	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=121
120	2017-02-15	7316	to previous	KarlijnWillems → http://.../moin/KarlijnWillems	Minor adjustments in language → https://wik... python.org/moin/Begin... ersGuide?acti... on=diff&rev1=119&rev2=1	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=120

119	2016-11-30 17:34:09	7451	to previous	AivarAnnamaa → https:// python.org/in/AivarAnnamaa moin/BeginnersGuide?acti on=diff&rev1 =118&rev2=1 19	Update Thonny's address	view → http:// s://wiki.python.org/moin/BeginnersGuide? action=recall&rev=119
118	2016-10-01 05:53:26	7456	to previous	AivarAnnamaa → https:// python.org/in/AivarAnnamaa moin/BeginnersGuide?acti on=diff&rev1 =117&rev2=1 18	Added info about Thonny	view → http:// s://wiki.python.org/moin/BeginnersGuide? action=recall&rev=118
117	2016-09-12 17:39:36	7216	to previous	MartijnTH → https:// python.org/artijnTH moin/BeginnersGuide?acti on=diff&rev1 =116&rev2=1 17	Corrected Typo	view → http:// s://wiki.python.org/moin/BeginnersGuide? action=recall&rev=117
116	2016-01-25 18:41:05	7218	to previous	Matt Jones → https:// python.org/att%20Jones moin/BeginnersGuide?acti on=diff&rev1 =115&rev2=1 16		view → http:// s://wiki.python.org/moin/BeginnersGuide? action=recall&rev=116
115	2015-08-23 17:30:39	6953	to previous	pythonguru → https:// python.org/ythonguru moin/BeginnersGuide?acti on=diff&rev1 =114&rev2=1 15		view → http:// s://wiki.python.org/moin/BeginnersGuide? action=recall&rev=115

114	2015-06-03	6872	to previous	TimGolden → https:// → https://wik i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =113&rev2=1 14	view → http s://wiki.pytho n.org/moin/Be ginnerGuide? action=recall&r ev=114
113	2015-05-24	6766	to previous	pyharris → https://wik → https://wik i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =112&rev2=1 13	view → http s://wiki.pytho n.org/moin/Be ginnerGuide? action=recall&r ev=113
112	2014-12-17	6757	to previous	OMFGNuts → https:// → https://wik i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =111&rev2=1 12	view → http s://wiki.pytho n.org/moin/Be ginnerGuide? action=recall&r ev=112
111	2014-11-03	6757	to previous	Kok Cheng Tan → htt → https://wik i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =110&rev2=1 11	view → http s://wiki.pytho n.org/moin/Be ginnerGuide? action=recall&r ev=111
110	2014-11-03	6758	to previous	Kok Cheng Tan → htt → https://wik i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =109&rev2=1 10	view → http s://wiki.pytho n.org/moin/Be ginnerGuide? action=recall&r ev=110

109	2014-04-17 05:23:19	6615	to previous	DaleAthanasias → http://https://wik	change to internal link ps://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=108&rev2=109	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=109
108	2014-03-16 18:02:10	6643	to previous	moustafa.farhat → http://https://wik	ps://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=107&rev2=108	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=108
107	2013-02-27 18:13:45	6481	to previous	kirpit → https://wiki.pyth	pydoc.net list item for particular python package/module searching. hon.org/moin/kirpit	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=107
106	2013-01-26 16:14:39	6419	to previous	Bob Niece → https://w	added a quiz resource iki.python.org/moin/Bo	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=106
105	2012-05-07 21:38:30	6307	to previous	David Pritchard → http://https://wik	ps://wiki.python.org/moin/B	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=105

104	2011-11-22	6215	to previous	Doug → https://wiki.python.org/moin/Doug → https://python.org/moin/BeginnersGuide?action=recall&rev1=103&rev2=104	grammar and add a comment about pre-installed Python on mac and linux	view → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev=104
103	2011-08-21	5966	to previous	SkipMontanaro → https://python.org/moin/SkipMontanaro → https://python.org/moin/BeginnersGuide?action=recall&rev1=102&rev2=103		view → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev=103
102	2011-08-19	5917	to previous	221 → https://python.org/moin/BeginnersGuide?action=recall&rev1=101&rev2=102	Change to Default language	view → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev=102
101	2011-08-19	9100	to previous	221 → https://python.org/moin/BeginnersGuide?action=recall&rev1=100&rev2=101	Get resource code first then change to English version	view → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev=101
100	2011-08-18	5917	to previous	SkipMontanaro → https://python.org/moin/SkipMontanaro → https://python.org/moin/BeginnersGuide?action=recall&rev1=99&rev2=100	I'm fine with a Chinese translation, but it can't be the default. :-)	view → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev=100

99	2011-08-18	9100	to previous	119 → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=98&rev2=99	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=99
98	2011-08-18	9100	to previous	119 → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=97&rev2=98	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=98
97	2011-08-18	6048	to previous	119 → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=96&rev2=97	Translate into Chinese view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=97
96	2011-05-01	5917	to previous	109 → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=95&rev2=96	add link to checkio.org view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=96
95	2011-04-20	5818	to previous	MichaelFoord → http://wiki.python.org/moin/MichaelFoord → https://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=94&rev2=95	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=95
-	2010-12-19	0	-	AndrewKuchling → h https://wiki.python.org/moin/AndrewKuchling	ATTDEL: PythonInstallation15DEC2010
-	2010-12-19	0	-	76 20:52:58	ATTNEW: Python-Doc&Start16DEC2010.pdf
-	2010-12-19	0	-	76 20:15:10	ATTNEW: PythonInstallation15DEC2010

94	2010-12-11	5852	to previous	AndrewKuchling → h	Revert spam → https://wik... https://wiki.python.org/moin/BeginnersGuide?acti...&rev1=93&rev2=94	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=93&rev2=94
93	2010-12-11	6055	to previous	AMontsouris-553-1- → https://wik... https://wiki.python.org/moin/BeginnersGuide?acti...&rev1=92&rev2=93	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=92&rev2=93	
92	2010-11-04	5948	to previous	120 → https://wik... https://wiki.python.org/moin/BeginnersGuide?acti...&rev1=91&rev2=92	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=91&rev2=92	
91	2010-11-04	5852	to previous	AndrewKuchling → h Revert spam → https://wik... https://wiki.python.org/moin/BeginnersGuide?acti...&rev1=90&rev2=91	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=90&rev2=91	
90	2010-11-04	5948	to previous	120 → https://wik... https://wiki.python.org/moin/BeginnersGuide?acti...&rev1=89&rev2=90	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=89&rev2=90	
89	2010-10-24	5852	to previous	Kok Cheng Tan → ht... → https://wik... https://wiki.python.org/moin/Kok%20Cheng%20Ta...n	view → http://wiki.python.org/moin/Kok%20Cheng%20Tan	

88	2010-07-14	5725	to previous	Tshepan-Lekhonkhobe	Sourceforge is no longer used for bug reporting.	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=87&rev2=88
87	2010-06-21	5774	to previous	NickCoghlan	Note that the rest of the guide currently assumes you're going to use Python 2	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=86&rev2=87
86	2010-06-21	5674	to previous	NickCoghlan	Start the process of making this part of the wiki at least open to the idea of using Python 3 rather than Python 2	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=85&rev2=86
85	2010-01-14	5489	to previous	PaulBoddie	Revert spam.	view → https://wiki.python.org/moin/PaulBoddie?action=recall&rev1=84&rev2=85
84	2010-01-14	5566	to previous	222		view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=83&rev2=84
83	2009-12-14	5489	to previous	AndrewKuchling	Add content of /Courses/ page	view → https://wiki.python.org/moin/AndrewKuchling?action=recall&rev1=82&rev2=83

82	2009-01-25	5279	to previous	DavidGoodger → http → https://wikis://wiki.python.org/moin 13:34:36 i.python.org/ n/ DavidGoodger moin/BeginnersGuide?acti on=diff&rev1 =81&rev2=82	view → http s://wiki.pytho n.org/moin/BeginnersGuide? action=recall&r ev=82
81	2008-11-15	5255	to previous	localhost converted to 1.6 markup → https://wikis://wiki.python.org/moin 13:59:40 i.python.org/ n/ BeginnersGuide ersGuide?acti on=diff&rev1 =80&rev2=81	view → http s://wiki.pytho n.org/moin/BeginnersGuide? action=recall&r ev=81
80	2008-02-04	5209	to previous	DavidGoodger → http link & markup fixes → https://wikis://wiki.python.org/moin 15:46:36 i.python.org/ n/ DavidGoodger moin/BeginnersGuide?acti on=diff&rev1 =79&rev2=80	view → http s://wiki.pytho n.org/moin/BeginnersGuide? action=recall&r ev=80
79	2007-12-31	5242	to previous	MartinvonLoewis → Cheese Shop -> Package Index → https://wikis://wiki.python.org/moin 13:31:14 i.python.org/ n/ MartinvonLoewis moin/BeginnersGuide?acti on=diff&rev1 =78&rev2=79	view → http s://wiki.pytho n.org/moin/BeginnersGuide? action=recall&r ev=79
78	2007-03-19	5286	to previous	PaulBoddie → https://wikis://wiki.python.org/moin/PaulBoddie Added headings, linked to editing advice, reworded editor text. → https://wikis://wiki.python.org/moin/PaulBoddie 14:36:17 i.python.org/ n/ PaulBoddie moin/BeginnersGuide?acti on=diff&rev1 =77&rev2=78	view → http s://wiki.pytho n.org/moin/BeginnersGuide? action=recall&r ev=78
77	2005-12-17	5064	to previous	AndrewKuchling → h Make Kent Johnson's suggested changes → https://wikis://wiki.python.org/moin 02:49:38 i.python.org/ n/ AndrewKuchling moin/BeginnersGuide?acti on=diff&rev1 =76&rev2=77	view → http s://wiki.pytho n.org/moin/BeginnersGuide? action=recall&r ev=77

76	2005-12-08	4778	to previous	AndrewKuchling → h	Change help link → https://wik i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =75&rev2=76	view → http s://wiki.pytho n.org/moin/Be ginnersGuide? action=recall&r ev=76
75	2005-12-08	4774	to previous	AndrewKuchling → h	Remove video link → https://wik i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =74&rev2=75	view → http s://wiki.pytho n.org/moin/Be ginnersGuide? action=recall&r ev=75
74	2005-12-08	5172	to previous	AndrewKuchling → h	Update URLs; make some edits suggested → https://wik i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =73&rev2=74	view → http s://wiki.pytho n.org/moin/Be ginnersGuide? action=recall&r ev=74
73	2005-12-08	5255	to previous	AndrewKuchling → h	Move the table of contents down → https://wik i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =72&rev2=73	view → http s://wiki.pytho n.org/moin/Be ginnersGuide? action=recall&r ev=73
72	2005-11-16	5206	to previous	AndrewKuchling → h	Remove link → https://wik i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =71&rev2=72	view → http s://wiki.pytho n.org/moin/Be ginnersGuide? action=recall&r ev=72
71	2005-11-16	5291	to previous	AndrewKuchling → h	Remove list to reduce indentation → https://wik i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =70&rev2=71	view → http s://wiki.pytho n.org/moin/Be ginnersGuide? action=recall&r ev=71

70	2005-11-16	5356	to previous	AndrewKuchling → h	Add ACL; remove 'xxx' → https://wik https://wiki.python.org/m i.python.org/ oin/AndrewKuchling moin/Beginn ersGuide?acti on=diff&rev1 =69&rev2=70	view → http s://wiki.pytho n.org/moin/Be ginnersGuide? action=recall&r ev=70
69	2005-11-11	5345	to previous	12	→ https://wik https://wiki.python.org/ i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =68&rev2=69	view → http s://wiki.pytho n.org/moin/Be ginnersGuide? action=recall&r ev=69
68	2005-09-16	5341	to previous	p5487BF3E	[[PageList(BeginnersGuide())]]	view → http s://wiki.pytho n.org/moin/Be ginnersGuide? action=recall&r ev=68
67	2005-09-05	5308	to previous	ip9-73-58-81	→ https://wik https://wiki.python.org/ i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =66&rev2=67	view → http s://wiki.pytho n.org/moin/Be ginnersGuide? action=recall&r ev=67
66	2005-09-05	5334	to previous	ip9-73-58-81	→ https://wik https://wiki.python.org/ i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =65&rev2=66	view → http s://wiki.pytho n.org/moin/Be ginnersGuide? action=recall&r ev=66
65	2005-09-05	5308	to previous	braun	→ https://wik https://wiki.python.org/ i.python.org/ moin/Beginn ersGuide?acti on=diff&rev1 =64&rev2=65	view → http s://wiki.pytho n.org/moin/Be ginnersGuide? action=recall&r ev=65

64	2005-09-05	5309	to previous	braun 09:12:10	braun view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=63&rev2=64	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=64
63	2005-08-30	5311	to previous	209 20:25:38	209 view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=62&rev2=63	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev=63
62	2005-08-26	5308	to previous	12-214-236-163 17:54:37	12-214-236-163 The Python Package Index is now the Cheese Shop.	view → http://wiki.python.org/moin/BeginnersGuide?action=recall&rev1=61&rev2=62
61	2005-08-23	5264	to previous	19:31:13	SkipMontanaro → SkipMontanaro view → https://wiki.python.org/moin/SkipMontanaro	view → http://wiki.python.org/moin/SkipMontanaro?action=recall&rev=61
60	2005-08-22	5284	to previous	21:57:07	users-matrix-217-146-246-1 view → https://wiki.python.org/moin/UsersMatrix?action=recall&rev1=59&rev2=60	view → http://wiki.python.org/moin/UsersMatrix?action=recall&rev=60

Attachments for "BeginnersGuide" - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide?action=AttachFile>

Attached Files

No attachments stored for BeginnersGuide

You are not allowed to attach a file to this page.

Login - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide?action=login>

Name

Password

OpenID



[myOpenID](#)



If you do not have an account, [you can create one now](#) → <https://wiki.python.org/moin/BeginnersGuide?action=newaccount>. [Forgot your password?](#) → <https://wiki.python.org/moin/BeginnersGuide?action=recoverpass>

If you do not have an account yet, you can still log in with your OpenID and create one during login.

BeginnersGuideChinese - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuideChinese>

- : <http://wiki.python.org/moin/BeginnersGuide>
Python

¶¶¶Python¶

[Python](#) → <https://wiki.python.org/moin/BeginnersGuide/OverviewChinese> Python

¶¶Python

Python
Python
Python
Python 2 Python 3 Python2orPython3 → <https://wiki.python.org/moin/Python2>
orPython3 2010 6 21 Python
[BeginnersGuide/Download](#) → <https://wiki.python.org/moin/BeginnersGuide/Download>
Python [HowToEditPythonCode](#) → <https://wiki.python.org/moin/HowToEditPythonCode>

¶¶Python

Python
• → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammersChinese>
• [BeginnersGuide/Programmers](#) → <https://wiki.python.org/moin/BeginnersGuide/Programmers>
• python.org's → <http://wiki.python.org/moin/Languages>.

Windows Windows → <http://www.python.org/doc/faq/windows/#how-do-i-run-a-python-program-under-windows>
Python → <http://docs.python.org/g/tut/> a Library Reference → <http://docs.python.org/lib/> Python
[[<http://docs.python.org/ref/>][the Language Reference]] Python

Python [PythonEditors](#) → <https://wiki.python.org/moin/PythonEditors>
Python

?

Python	Python	BeginnersGuide/Help → https://wiki.python.org/moin/BeginnersGuide/Help	IntroductoryBooks → https://wiki.python.org/moin/IntroductoryBooks
BeginnersGuide/Examples → https://wiki.python.org/moin/BeginnersGuide/Examples			
Python			
PythonEvents → https://wiki.python.org/moin/PythonEvents	PythonTraining → https://wiki.python.org/moin/PythonTraining	EDU-SIG → http://www.python.org/sigs/edu-sig/	K-12
Python			

8

1. [BeginnersGuide/Download](https://wiki.python.org/moin/BeginnersGuide/Download) → <https://wiki.python.org/moin/BeginnersGuide/Download>
 2. [BeginnersGuide/Examples](https://wiki.python.org/moin/BeginnersGuide/Examples) → <https://wiki.python.org/moin/BeginnersGuide/Examples>
 3. [BeginnersGuide/Help](https://wiki.python.org/moin/BeginnersGuide/Help) → <https://wiki.python.org/moin/BeginnersGuide/Help>
 4. [BeginnersGuide/Mathematics](https://wiki.python.org/moin/BeginnersGuide/Mathematics) → <https://wiki.python.org/moin/BeginnersGuide/Mathematics>
 5. [BeginnersGuide/NonProgrammers](https://wiki.python.org/moin/BeginnersGuide/NonProgrammers) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers>
 6. [BeginnersGuide/NonProgrammersChinese](https://wiki.python.org/moin/BeginnersGuide/NonProgrammersChinese) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammersChinese>
 7. [BeginnersGuide/Overview](https://wiki.python.org/moin/BeginnersGuide/Overview) → <https://wiki.python.org/moin/BeginnersGuide/Overview>
 8. [BeginnersGuide/OverviewChinese](https://wiki.python.org/moin/BeginnersGuide/OverviewChinese) → <https://wiki.python.org/moin/BeginnersGuide/OverviewChinese>
 9. [BeginnersGuide/Programmers](https://wiki.python.org/moin/BeginnersGuide/Programmers) → <https://wiki.python.org/moin/BeginnersGuide/Programmers>
 10. [BeginnersGuide/Programmers \(Cpp2Python.pdf\)](https://wiki.python.org/moin/BeginnersGuide/Programmers_(Cpp2Python.pdf)) → <https://wiki.python.org/moin/BeginnersGuide/Programmers?action=AttachFile&do=view&target=Cpp2Python.pdf>
 11. [BeginnersGuide/Programmers/SimpleExamples](https://wiki.python.org/moin/BeginnersGuide/Programmers/SimpleExamples) → <https://wiki.python.org/moin/BeginnersGuide/Programmers/SimpleExamples>

5

- Pyschools - A Collection of Python Quiz and Exercise Questions → <http://www.pyschools.com/>
 - CheckIO - Online learning, testing and improving your python skills → <http://www.checkio.org/>

Python

- [Python Package Index](http://pypi.python.org/pypi) → <http://pypi.python.org/pypi>.
• [searching.python.org](http://www.python.org/search/) → <http://www.python.org/search/> - Python
• <http://www.python.org/doc/FAQ/> → <http://www.python.org/doc/FAQ/>
 - [Google](http://www.google.com) → <http://www.google.com> “python”
 - comp.lang.python

Python.org?

- Python [Python Software Foundation](http://www.python.org/psf/) → <http://www.python.org/psf/> PSF → h
<http://www.python.org/psf/donations/> PayPal → <http://www.paypal.com/>
 - [Python bug](http://bugs.python.org/) → <http://bugs.python.org/> Python bug
 - python bug [Python](http://www.python.org/dev/) → <http://www.python.org/dev/> Python
 - Python [Python](http://www.python.org/doc/) → <http://www.python.org/doc/> Documentation SIG]
docs@python.org → [<http://bugs.python.org/> → <http://www.python.org/sigs/doc-sig/>
 - Python [comp.lang.python.](http://www.python.org/community/lists/#comp-lang:python-announce) → <http://www.python.org/community/lists/#comp-lang:python-announce> Python → <http://www.python.org/>
 - Python [comp.lang.python](http://www.python.org/patches/) → <http://www.python.org/patches/> Python
-

[CategoryDocumentation](https://wiki.python.org/moin/CategoryDocumentation) → <https://wiki.python.org/moin/CategoryDocumentation> r's Guide to Python

BeginnersGuide/Overview - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Overview>

Python is a clear and powerful object-oriented programming language, comparable to Perl, Ruby, Scheme, or Java.

Some of Python's notable features:

- Uses an elegant syntax, making the programs you write easier to read.
- Is an easy-to-use language that makes it simple to get your program working. This makes Python ideal for prototype development and other ad-hoc programming tasks, without compromising maintainability.
- Comes with a large standard library that supports many common programming tasks such as connecting to web servers, searching text with regular expressions, reading and modifying files.
- Python's interactive mode makes it easy to test short snippets of code. There's also a bundled development environment called IDLE.
- Is easily extended by adding new modules implemented in a compiled language such as C or C++.
- Can also be embedded into an application to provide a programmable interface.
- Runs anywhere, including Mac OS X → <https://www.python.org/downloads/mac-osx/>, Windows → <https://www.python.org/downloads/windows/>, Linux → <https://docs.python.org/3/using/unix.html>, and Unix → <https://docs.python.org/3/using/unix.html>, with unofficial builds also available for Android → <https://wiki.python.org/moin/Android> and iOS.
- Is free software in two senses. It doesn't cost anything to download or use Python, or to include it in your application. Python can also be freely modified and re-distributed because while the language is copyrighted it's available under [an open-source license](#) → <http://www.python.org/psf/license/>.

Some programming-language features of Python are:

- A variety of basic data types are available: numbers (floating point, complex, and unlimited-length long integers), strings (both ASCII and Unicode), lists, and dictionaries.
- Python supports object-oriented programming with classes and multiple inheritances.
- Code can be grouped into modules and packages.
- The language supports raising and catching exceptions, resulting in cleaner error handling.
- Data types are strongly and dynamically typed. Mixing incompatible types (e.g. attempting to add a string and a number) causes an exception to be raised, so errors are caught sooner.
- Python contains advanced programming features such as generators and list comprehensions.
- Python's automatic memory management frees you from having to manually allocate and free memory in your code.

See the [SimplePrograms](#) → <https://wiki.python.org/moin/SimplePrograms> collection of short programs, gradually increasing in length, which shows off Python's syntax and readability.

Writing Pythonic code is not hard---but you have to get used to the (PEP) code style rules. You can test, check, and improve your code style at online resources such as [Pythonchecker.com](#) → <http://pythonchecker.com/>.

Translations:

- [ChineseOverview](#) → <https://wiki.python.org/moin/Todo>

BeginnersGuide/Download - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Download>

Downloading Python

On many systems Python comes pre-installed, you can try running the `python` command to start the Python interpreter to check and see if it is already installed. On windows you can try the `py` command which is a launcher which is more likely to work. If it is installed you will see a response which will include the version number, for example:

```
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit  
(AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.
```

If you don't see this, you will need to install Python on your system.

If the version number is Python 2.x.y (where x and y are any number) you are using Python 2 which is no longer supported and is not a good choice for development. You can try running `python3` to see if there is also a Python 3.x.y version installed, if not you'll want to install the latest version of Python.

If you do not have Python installed or need a newer version you can go to:

<https://www.python.org/downloads/>

which will provide a button to download an installer for your particular system. The Python documentation also has a detailed guide on how to install and setup Python here:

<https://docs.python.org/3/using/index.html>

Below are some system specific notes to keep in mind.

Windows

On Windows the most stable build is available from the official download page

<https://www.python.org/downloads/>

You should download and run the installer from that page to get the latest version of Python for your system. You can refer to the Python documentation for more details on the installation process and getting started:

<https://docs.python.org/3/using/windows.html>

Mac

For macOS 10.9 (Jaguar) up until 12.3 (Catalina) the operating system includes Python 2, which is no longer supported and is not a good choice for development. You should go to do the downloads page: <https://www.python.org/downloads/> and download the installer.

For newer versions of macOS, Python is no longer included by default and you will have to download and install it. You can refer to the Python documentation for more details on the installation process and getting started:

<https://docs.python.org/3/using/mac.html>

Linux

On most Linux distributions Python comes pre-installed and/or available via the distribution's package managers. Below are some common examples, but refer to your specific distribution's documentation and package list to get the most up to date instructions.

If you'd like to download and build Python from source (or your distribution's package manager does not include a version of Python you need) you can download a source tarball from the general download page: <https://www.python.org/downloads/>

Red Hat, CentOS, or Fedora

```
dnf install python3 python3-devel
```

Debian or Ubuntu

```
apt-get install python3 python3-dev
```

Gentoo

```
emerge dev-lang/python
```

Arch Linux

```
pacman -S python3
```

IntegratedDevelopmentEnvironments - Python Wiki

Source: <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>



Please keep wiki links as wiki links, use external links only if there is no existing page for the IDE.

See also Wikipedia's [list of Python IDEs](http://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments#Python) → http://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments#Python and [ShowMeDo](https://wiki.python.org/moin>ShowMeDo) → <https://wiki.python.org/moin>ShowMeDo> videos for Wing 3 Professional, Python Development With SPE, Eclipse PyDev → <https://wiki.python.org/moin/PyDev> and IPython (see site page for updated information).

Name	Platform	Entry Updated	Notes
Thonny	Windows, Linux, Mac OS X, more	2023	For teaching/learning programming. Focused on program runtime visualization. Provides stepping both in statements and expressions, no-hassle variables view, separate mode for explaining references etc.
Komodo	Windows/Linux/Mac OS X	2023	Multi-language IDE with support for Python 2.x and Python 3. Available as https://www.activestate.com/products/komodo-ide/ now open source only; last updated 2022
CodeLobster	Windows/Linux/Mac OS X	2023	Multi-language IDE with free support for Python: code completion, navigation and highlighting etc.
LiClipse	Linux/Mac OS X/Windows	2023	Commercial Eclipse-based IDE which provides a standalone bundling PyDev → http://pydev.org/ , Workspace Mechanic, Eclipse Color Theme, StartExplorer and AnyEdit, along with lightweight support for other languages, and other usability enhancements (such as multi-caret-edition).
NetBeans	Linux, Mac, Solaris, Windows	2023	Python/Jython support in NetBeans → https://wiki.python.org/moin/NetBeans → Open source, allows Python and Jython Editing, code-completion, debugger, refactoring, templates, syntax analysis, etc. Note: the Python plugin as a community-supported project, and may trail behind. Currently it works for 8.1, does not appear to be available for 8.2
PyCharm	Linux/Mac OS X/Windows	2023	The <i>Community</i> edition is a free IDE with a smart Python editor providing quick code navigation, code completion, refactoring, unit testing and debugger. The commercial <i>Professional</i> edition fully supports Web development with Django, Flask, Mako and Web2Py → https://wiki.python.org/moin/Web2Py and allows to develop remotely. JetBrains offers free PyCharm Professional licenses for open-source projects under certain conditions https://www.jetbrains.com/buy/opensource/ . There is also free access for Student/Educational use.

Python for VS Code	Linux/Mac X/Windows	OS	2023	Free open-source → https://github.com/Microsoft/vscode-python extension for Visual Studio Code (now maintained by Microsoft). Supports syntax highlighting, debugging, code completion, code navigation, unit testing, refactoring, with support for Django, multi threaded, local and remote debugging.
KDevelop	Linux/Mac X/(Windows)	OS	2023	Free open-source IDE with a focus on static analysis-based code completion, navigation and highlighting. Also features a VI emulation mode.
PyDev	Eclipse		2023	Free, open-source plugin for Eclipse → http://www.eclipse.org/ -- Allows Python, Jython, and IronPython → https://wiki.python.org/moin/IronPython editing, code-completion, debugger, refactoring, quick navigation, templates, code analysis, unittest integration, Django integration, etc.
Wing	Windows, Mac OS X	Linux,	2023	Family of Python IDEs with advanced debugger, editor with vi, emacs, visual studio and other key bindings, auto-completion, auto-editing, import management, multi-selection, inline code warnings, snippets, goto-definition, find uses, refactoring, unit testing with code coverage, remote development, support for containers and clusters, array and dataframe viewer, bookmarking, project management with version control, Python environment creation with virtualenv, pipenv, conda, and Docker, Python package management with pip, pipenv, and conda, source browser, PEP 8 / Black / YAPF reformatting, and much more. Product levels, include free and paid versions with a fully functional trial and free licenses for educational use and unpaid open source developers. Documentation includes help for using Wing with Django, Flask, Docker, AWS, Vagrant, Matplotlib, Jupyter, Blender, Maya, any many other third party tools and packages. See product comparison → http://wingware.com/downloads and pricing → http://wingware.com/store/purchase for details.
PyScripter	Windows		2023	MIT licensed IDE written in Delphi with debugger, integrated unit testing, source browser, code navigation and syntax coloring/autocomplete editor.

Spyder → http://www.spyder-ide.org/	Windows/Linux/macOS	2023	A powerful, free/open-source scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. Features a unique combination of the advanced editing, analysis, debugging and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection and beautiful visualization capabilities of a scientific package. Furthermore, offers built-in integration with many popular scientific packages, including NumPy → https://wiki.python.org/moin/NumPy , SciPy → https://wiki.python.org/moin/SciPy , Pandas, IPython, QtConsole → https://wiki.python.org/moin/QtConsole , Matplotlib, SymPy → https://wiki.python.org/moin/SymPy , and more, and can be easily extended with plugins. It is conveniently integrated in the cross-platform Anaconda distribution → https://www.anaconda.com/ , and is the centerpiece of the Python(x,y) → https://python-xy.github.io/ and WinPython → https://winpython.github.io/ distributions.
IDLE	Windows/Linux/Mac OS X/All Tk Platforms	2023	Multi-window colorized source browser, autoindent, autocompletion, tool tips, code context panel, search in files, class and path browsers, debugger, executes code in clean separate subprocess with one key-stroke. 100% pure Python, part of Python 2.x and 3.x distributions (may be packaged separately in some situations).
IdleX → http://idlex.sourceforge.net/	Windows/Linux/Mac OS X/All Tk Platforms	2023	IdleX is a collection of over twenty extensions and plugins that provide additional functionality to IDLE, a Python IDE provided in the standard library. It transforms IDLE into a more useful tool for academic research and development as well as exploratory programming. Last updated 2022.
μ.dev → http://sakurastudio.yo-lasite.com/micro-dev.php	Windows Vista and XP	2023	An open-source IDE, created using Lazarus. It's only for Python. include syntax highlighting, project manager, and uses pdb for debugging. Development stopped in 2011.
Pyzo (formerly IEP) → http://www.pyzo.org/	Windows/Linux/Mac OS X	2023	Open-source Python IDE focused on interactivity and introspection, which makes it very suitable for scientific computing. Its practical design is aimed at simplicity and efficiency. Pyzo consists of two main components, the editor and the shell, and uses a set of pluggable tools to help the programmer in various ways: e.g. source structure, interactive help, workspace, file browser (with functionality for searching). Also includes a post-mortem debugger.

Python Toolkit (PTK)	Win-dows/Linux/Mac OS X	2023	An interactive environment for python built around a matlab style console window and editor. It was designed to provide a python based environment similar to Matlab for scientists and engineers however it can also be used as a general purpose interactive python environment especially for interactive GUI programming. Features include: Multiple independent python interpreters. Interactively program with different GUI toolkits (wxPython, TkInter → https://wiki.python.org/moin/Tkinter , pyGTK, pyQT4 and PySide → https://wiki.python.org/moin/PySide). Matlab style namespace/workspace browser. Object auto-completions, calltips and multi-line command editing in the console. Object inspection and python path management. Simple code editor and integrated debugger. Last released in 2014.
Python Tools for Visual Studio	Windows	2023	Open-source plugin for Visual Studio 2010, 2012 onwards (now maintained by Microsoft). Supports syntax highlighting, debugging and rich intellisense, unit testing, refactoring, object browser, MPI cluster debugging, Django intellisense and debugging, development REPL window and a debugging REPL window. Supports mixed-mode Python/C/C++ debugging.
Exedore	Mac OS X	2015	Commercial with feature-limited free trial. A Mac-native, single-window IDE inspired by Xcode. Features integrated debugger, tabs, code completion with tab triggers, syntax highlighting themes, search and replace with regex, integrated REPL sessions, goto definition, file browser, integrated documentation browser. As of June 2015, does not support input() meaning any console input using this function is not supported.

Name	Platform	Updated	Notes
Komodo	Windows/Mac/Linux	2012	Komodo Edit → http://www.activestate.com/komodo-edit (open source, as part of the Open Komodo → http://www.openkomodo.com/ project). Little brother to Komodo IDE.
BlackAdder	Windows/Linux	2004	Commercial; integrated debugger; interfaces with Qt Designer
eric	Python + PyQt	2018	Open Source, interfaces with Qt Designer, Qt Linguist, unittest; integrated debugger
SPE	Windows, MacOsX, more	2008	Open-source with wxPython → http://www.wxpython.org/ interface. Code completion, call tips, class explorer, source index, auto todo list, Blender → http://www.blender.org/ support, integrated PyChecker → http://pychecker.sourceforge.net/ (source code doctor) and Kiki → http://project5.freezone.org/kiki (regex console). Download instructions → http://pythonide.blogspot.com/2007/02/how-to-download-latest-spe-from_26.html

Pida	→ http://pid.a.co.uk/	Linux, FreeBSD, ..., 2007 (2008 dev)	(Windows in progress)	Open-source with GTK interface, written in Python. Supports different languages, python trough rope	→ http://rope.sourceforge.net/ and pyflakes as well as rpdb2. Support different Editors (Vim, Medit, Emacs) Current Repos → http://pida.co.uk/trac/wiki/DeveloperRepos
SharpDevelop	.net CLR	26/7/2009		FOSS IDE uses IronPython	→ https://wiki.python.org/moin/IronPython to support making python module solutions.
NINJA-IDE	→ h ttp://ninja-ide.org/	Python + PyQt	→ https://wiki.python.org/moin/PyQt	2011 + (Linux/Windows/Mac OS X)	NINJA-IDE (from: "Ninja Is Not Just Another IDE"), is a cross-platform integrated development environment specially design to build Python Applications.
Aptana Studio 3	Linux, Windows and Mac OS X	10/01/2012		Aptana Studio3 is a professional, open source development tool for the open web	
Pcode	→ https://github.com/fortharries/Pcode	Windows, Linux and Mac OSX	2014	Python 3x IDE with emphasis on power, usability and simplicity.	

HowToEditPythonCode - Python Wiki

Source: <https://wiki.python.org/moin/HowToEditPythonCode>

Contents

1. [History](#)
2. [Programming philosophy](#)
3. [Name and neologisms](#)
4. [Usage](#)
5. [Syntax and semantics](#)
 1. [Indentation](#)
 2. [Statements and control flow](#)
 3. [Expressions](#)
 4. [Methods](#)
 5. [Typing](#)
 6. [Mathematics](#)
6. [Implementations](#)
 1. [CPython](#)
 2. [Alternative implementations](#)
 3. [Interpretational semantics](#)
7. [Development](#)
8. [Standard library](#)
9. [Influence on other languages](#)

History

Python was conceived in the late 1980s and its implementation was started in December 1989 by Guido van Rossum at CWI in Amsterdam. Python is a successor to the ABC programming language (itself inspired by SETL) capable of exception handling and interfacing with the Amoeba operating system. Van Rossum is Python's principal author, and was for many years known by the humorous title *Benevolent Dictator for Life* (BDFL). Python evolution is now guided by an elected Steering Committee.

Programming philosophy

Python is a multi-paradigm programming language. Rather than forcing programmers to adopt a particular style of programming, it permits several styles: object-oriented programming and structured programming are fully supported, and there are a number of language features which support functional programming and aspect-oriented programming (including metaprogramming and "by magic" methods). Many other paradigms are supported using extensions, such as pyDBC and Contracts for Python which allow Design by Contract.

Rather than requiring all desired functionality to be built into the language's core, Python was designed to be highly extensible. New built-in modules can be easily written in C, C++ or Cython. Python can also be used as an extension language for existing modules and applications that need a programmable interface. This design, a small core language with a large standard library with an easily extensible interpreter, was intended by Van Rossum from the very start because of his frustrations with ABC (which espoused the opposite mindset).

Name and neologisms

An important goal of the Python developers is making Python fun to use. This is reflected in the origin of the name (based on the television series Monty Python's Flying Circus), in the common practice of using Monty Python references in example code, and in an occasionally playful approach to tutorials and reference materials. [24][25] For example, the metasyntactic variables often used in Python literature are spam and eggs, instead of the traditional foo and bar.

Usage

Python is often used as a scripting language for web applications, e.g. via mod_wsgi for the Apache web server. With Web Server Gateway Interface, a standard API has been developed to facilitate these applications. Web application frameworks like Django, Pylons, [TurboGears](https://wiki.python.org/moin/TurboGears) → <https://wiki.python.org/moin/TurboGears>, web2py, Flask, and Zope support developers in the design and maintenance of complex applications. Libraries like [NumPy](https://wiki.python.org/moin/NumPy) → <https://wiki.python.org/moin/NumPy>, [SciPy](https://wiki.python.org/moin/SciPy) → <https://wiki.python.org/moin/SciPy>, and Matplotlib allow Python to be used effectively in scientific computing.

Syntax and semantics

Python was intended to be a highly readable language. It is designed to have an uncluttered visual layout, frequently using English keywords where other languages use punctuation. Python requires less boilerplate than traditional manifestly typed structured languages such as C or Pascal, and has a smaller number of syntactic exceptions and special cases than either of these. For a detailed description of the differences between 2.x and 3.x versions, see History of Python.

Indentation

Python uses whitespace indentation, rather than curly braces or keywords, to delimit blocks (a feature also known as the *off-side rule*). An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block.

Statements and control flow

Python's statements include (among others):

- The ***if*** statement, which conditionally executes a block of code, along with ***else*** and ***elif*** (a contraction of else-if).
- The ***for*** statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The ***while*** statement, which executes a block of code as long as its condition is true.
- The ***try*** statement, which allows exceptions raised in its attached code block to be caught and handled by ***except*** clauses; it also ensures that clean-up code in a ***finally*** block will always be run regardless of how the block exits.
- The ***class*** statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The ***def*** statement, which defines a function or method.

- The ***with*** statement (from Python 2.5), which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run, and releasing the lock afterward).
- The ***pass*** statement, which serves as a NOP and can be used in place of a code block.
- The ***assert*** statement, used during debugging to check for conditions that ought to apply.
- The ***yield*** statement, which returns a value from a generator function. (From Python 2.5, ***yield*** is also an operator. This form is used to implement coroutines -- see below.)

Expressions

- In Python 3, the result of the division operator `/` with integer operands is always a floating-point value; the operator `//` may be used to perform integer division (the result truncated to an integer).
 - In Python 2, the `/` operator on integers performs integer division. Floating-point division on integers can be achieved by converting one of the integers to a float (e.g. `float(x) / y`), or, since 2.2, the behavior of Python 3 can be enabled using `from __future__ import division`.
- In Python, `==` compares by value, in contrast to Java, where it compares by reference. (Value comparisons in Java use the `equals()` method.) Python's `is` operator may be used to compare object identities (comparison by reference). Comparisons may be chained, for example `a <= b <= c`.
- Python uses the words ***and***, ***or***, ***not*** for its boolean operators rather than the symbolic `&&`, `||`, `!` used in C.
- Python has a type of expression known as a list comprehension. Python 2.4 extended list comprehensions into a more general expression known as a generator expression.
- Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be a single expression.
- Conditional expressions in Python are written as `x if c else y` (different in order of operands from the `?:` operator common to many other languages).
- Python makes a distinction between lists and tuples. Lists, written as `[1, 2, 3]`, are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples, written as `(1, 2, 3)`, are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The parentheses around the tuple are optional in some contexts. Tuples can appear on the left side of an equal sign; hence a statement like `x, y = y, x` can be used to swap two variables.
- Multiple string-formatting operations are supported. The original `%` operator provided behavior analogous to printf format strings in C, e.g. `foo=%s bar=%d" % ("blah", 2)` evaluates to `foo=blah bar=2`. String templates were introduced in Python 2.4. Python 3.0 brought in the `string format()` method, with syntax like `foo={0} bar={1}" .format("blah", 2)`. Formatted string literals ("F-strings") were introduced in 3.6, with syntax like `f'foo={"blah"} bar={2}'"`
- Python has various kinds of string literals:
 - Strings are delimited by single or double quotation marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quotation marks and double quotation marks function identically. Both kinds of strings use the backslash (`\`) as an escape character and there is no implicit string interpolation such as `"$foo"`.
 - Triple-quoted strings, which begin and end with a series of three single or double quotation marks, may span multiple lines and function like here-documents in shells, Perl and Ruby.

Methods

Methods on objects are functions attached to the object's class; the syntax `instance.method(argument)` is, for normal methods and functions, syntactic sugar for `Class.method(instance, argument)`. Python methods have an explicit `self` parameter to access instance data, in contrast to the implicit `self` in some other object-oriented programming languages (for example, Java, C++ or Ruby).

Typing

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Mathematics

Python defines the modulus operator so that the result of `a % b` is in the half-open interval $[0,b)$, where b is a positive integer. When b is negative, the result lies in the interval $(b,0]$. However, this consequently affects how integer division is defined. To maintain the validity of the equation $b * (a // b) + a \% b == a$, integer division is defined to round towards minus infinity. Therefore $7 // 3$ is 2, but $(-7) // 3$ is -3. This is different from many programming languages, where the result of integer division rounds towards zero, and Python's modulus operator is consequently defined in a way that can return negative numbers.

Implementations

C^{Py}thon

The mainstream Python implementation, known as C^{Py}thon, is written in C meeting the C89 standard. C^{Py}thon compiles Python programs into intermediate bytecode,[65] which are then executed by the virtual machine. It is distributed with a large standard library written in a mixture of C and Python. C^{Py}thon ships in versions for many platforms, including Microsoft Windows and most modern Unix-like systems. C^{Py}thon was intended from almost its very conception to be cross-platform; its use and development on esoteric platforms such as Amoeba, alongside more conventional ones like Unix and Mac OS, has greatly helped in this regard. Unofficial builds are also available for [Android](https://wiki.python.org/moin/Android) → <https://wiki.python.org/moin/Android> and iOS.

Alternative implementations

Jython compiles the Python program into Java byte code, which can then be executed by every Java Virtual Machine implementation. This also enables the use of Java class library functions from the Python program. IronPython → <https://wiki.python.org/moin/IronPython> follows a similar approach in order to run Python programs on the .NET Common Language Runtime. PyPy → <https://wiki.python.org/moin/PyPy> is a fast self-hosting implementation of Python, written in Python, that can output several types of bytecode, object code and intermediate languages. There also exist compilers to high-level object languages, with either unrestricted Python, a restricted subset of Python, or a language similar to Python as the source language. PyPy → <https://wiki.python.org/moin/PyPy> is of this type, compiling RPython to several languages; other examples include Pyjamas compiling to [JavaScript](https://wiki.python.org/moin/JavaScript) → <https://wiki.python.org/moin/JavaScript>; Shed Skin compiling to C++; and Cython and Pyrex compiling to C.

Interpretational semantics

Most Python implementations (including CPython) can function as a command-line interpreter, for which the user enters statements sequentially and receives the results immediately. In short, Python acts as a shell. While the semantics of the other modes of execution (bytecode compilation, or compilation to native code) preserve the sequential semantics, they offer a speed boost at the cost of interactivity, so they are usually only used outside of a command-line interaction (e.g., when importing a module).

Other shells add capabilities beyond those in the basic interpreter, including IDLE and IPython. While generally following the visual style of the Python shell, they implement features like auto-completion, retention of session state, and syntax highlighting.

Development

Python's development is conducted largely through the Python Enhancement Proposal (PEP) process, described in [PEP 1 → https://www.python.org/dev/peps/pep-0001/](https://www.python.org/dev/peps/pep-0001/). PEPs are standardized design documents providing general information related to Python, including proposals, descriptions, design rationales, and explanations for language features. Outstanding PEPs are reviewed and commented upon on the [python-dev mailing list](#), which is the primary forum for discussion about the language's development and approved by the Steering Council (see [PEP 13 → https://www.python.org/dev/peps/pep-013](https://www.python.org/dev/peps/pep-013) for the governance model); specific issues are discussed in the bug tracker maintained at [bugs.python.org → https://bugs.python.org/](https://bugs.python.org). Development of the reference implementation takes place on the [GitHub cpython → https://github.com/python/cpython](https://github.com/python/cpython) repository.

CPython's public releases come in three types, distinguished by which part of the version number is incremented:

- backward-incompatible versions, where code is expected to break and must be manually ported. The first part of the version number is incremented. These releases happen infrequently—for example, version 3.0 was released 8 years after 2.0.
- major or 'feature' releases, which are largely compatible but introduce new features. The second part of the version number is incremented. These releases are scheduled to occur roughly every 18 months, and each major version is supported by bugfixes for several years after its release.
- bugfix releases, which introduce no new features but fix bugs. The third and final part of the version number is incremented. These releases are made whenever a sufficient number of bugs have been fixed upstream since the last release, or roughly every 3 months. Security vulnerabilities are also patched in bugfix releases.

A number of alpha, beta, and release-candidates are also released as previews and for testing before the final release is made. Although there is a rough schedule for each release, this is often pushed back if the code is not ready. The development team monitor the state of the code by running the large unit test suite during development, and using the [BuildBot → https://wiki.python.org/moin/BuildBot](https://wiki.python.org/moin/BuildBot) continuous integration system.

Standard library

Python has a large standard library, commonly cited as one of Python's greatest strengths,[81] providing pre-written tools suited to many tasks. This is deliberate and has been described as a "batteries included" Python philosophy. The modules of the standard library can be augmented with custom modules written in either C or Python. Boost C++ Libraries includes a library, Boost.Python, to enable interoperability between C++ and Python. Because of the wide variety of tools provided by the standard library, combined with the ability to use a lower-level language such as C and C++, which is already capable of interfacing between other libraries, Python can be a powerful glue language between languages and tools.

The standard library is particularly well-tailored to writing Internet-facing applications, with a large number of standard formats and protocols (such as MIME and HTTP) already supported. Modules for creating graphical user interfaces, connecting to relational databases, arithmetic with arbitrary precision decimals, manipulating regular expressions, and doing unit testing are also included.

Some parts of the standard library are covered by specifications (for example, the WSGI implementation `wsgiref` follows PEP 333), but the majority of the modules are not. They are specified by their code, internal documentation, and test suite (if supplied). However, because most of the standard library is cross-platform Python code, there are only a few modules that must be altered or completely rewritten by alternative implementations.

For software testing, the standard library provides the `unittest` and `doctest` modules.

Influence on other languages

Python's design and philosophy have influenced several programming languages, including:

- Pyrex and its derivative Cython are code translators that are targeted at writing fast C extensions for the CPython interpreter. The language is mostly Python with syntax extensions for C and C++ features. Both languages produce compilable C code as output.
 - Boo uses indentation, a similar syntax, and a similar object model. However, Boo uses static typing and is closely integrated with the .NET framework.[84]
 - Cobra uses indentation and similar syntax. Cobra's "Acknowledgements" document lists Python first among the languages that influenced it. However, Cobra directly supports design-by-contract, unit tests and optional static typing.
 - \ borrowed iterators, generators, and list comprehensions from Python.
 - Go is described as incorporating the "development speed of working in a dynamic language like Python".
 - Groovy was motivated by the desire to bring the Python design philosophy to Java.
 - OCaml has an optional syntax, called `twt` (The Whitespace Thing), inspired by Python and Haskell.
-

BeginnersGuide/NonProgrammers - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers>

Python for Non-Programmers

If you've never programmed before, the tutorials on this page are recommended for you; they don't assume that you have previous experience. If you have programming experience, also check out the [BeginnersGuide/Programmers](#) → <https://wiki.python.org/moin/BeginnersGuide/Programmers> page.

Books

Each of these books can be purchased online but is also available as free textual, website, or video content.

- **Automate the Boring Stuff with Python - Practical Programming for Total Beginners** by *Al Sweigart* is "written for office workers, students, administrators, and anyone who uses a computer to learn how to code small, practical programs to automate tasks on their computer." ||website → <https://automatetheboringstuff.com/> ||print version → <http://www.amazon.com/gp/product/1593275994/> ||
- **How To Think Like a Computer Scientist** is a classic open-source book by *Allen Downey* with contributions from *Jeffrey Elkner* and *Chris Meyers*. It was updated to Python 3 by *Peter Wentworth*. ||website → <http://openbookproject.net/thinkcs/python/english3e/> ||print version → <http://openbookproject.net/thinkcs/python/english3e/> ||
- **Making Games with Python & Pygame** by *Al Sweigart* introduces the Pygame framework for novices and intermediate programmers to make graphical games. ||website → <http://inventwithpython.com/pygame> ||print version → <http://www.amazon.com/Making-Games-Python-Pygame-Sweigart/dp/1469901730?ie=UTF8&tag=playwithpyt-20&linkCode=as2&camp=1789&creative=9325&creativeASIN=0982106017> ||
- **Python One-Liners** by *Christian Mayer* teaches you how to read and write "one-liners": concise statements of useful functionality packed into a single line of code. ||website with free one-liner explainer videos → <http://pythononeliners.com/> ||print version → <https://www.amazon.com/gp/product/B07ZY7XMX8> ||
- **Think Python** by *Allen B. Downey* teaches you how to think like a computer scientist. ||website → <http://greenteapress.com/thinkpython/html/index.html> ||print version → <https://www.amazon.com/Think-Python-Like-Computer-Scientist/dp/1491939362/> ||

You can find many free Python books online. For example, check out [this article with 101 free Python books](#) → <https://blog.finxter.com/free-python-books/>.

Interactive Courses

These sites give you instant feedback on programming problems that you can solve in your browser.

- [A beginner-friendly and free Python tutorial](#) → <https://python.land/python-tutorial> with interactive code examples, explaining the Python language in an easy-to-understand way.
- [A beginner-friendly Python course](#) → <https://programiz.pro/learn/master-python> that teaches to learn to code through bite-size lessons, quizzes and 100+ challenges.
- [CheckiO](#) → <http://www.checkio.org/> is a gamified website containing programming tasks that can be solved in Python 3.
- [Codéex](#) → <https://www.codedex.io/> is a learn to code platform for K-12 and college students.

- [Codecademy](https://www.codecademy.com/search?query=pythonPython) (→ <https://www.codecademy.com/search?query=pythonPython>)
- [Code the blocks](https://codetheblocks.com/) → <https://codetheblocks.com/> combines Python programming with a 3D environment where you "place blocks" and construct structures. It also comes with Python tutorials that teach you how to create progressively elaborate 3D structures.
- [Codevisionz Python](https://codevisionz.com/learn:python:programming/) → <https://codevisionz.com/learn:python:programming/> 10+ hrs of Python learning material - Learn common programming concepts through code examples, quizzes, and challenges
- [Computer Science Circles](http://cscircles.cemc.uwaterloo.ca/) → <http://cscircles.cemc.uwaterloo.ca/> has 30 lessons, 100 exercises, and a message system where you can ask for help. Teachers can use it with their students. It is also available in Dutch, French, German, and Lithuanian.
- [DataCamp Python Tutorial](https://www.datacamp.com/courses/intro-to-python-for-data-science) → <https://www.datacamp.com/courses/intro-to-python-for-data-science> Unlike most other Python tutorials, this 4 hour tutorial by [DataCamp](https://www.datacamp.com/) → <https://www.datacamp.com/> focuses on Python specifically for Data Science. It has 57 interactive exercises and 11 videos.
- [Finxter](https://finxter.com/) → <https://finxter.com/> - How good are your Python skills? Test and Training with >300 hand-picked Python puzzles.
- [HackInScience](https://hackinscience.org/) → <https://hackinscience.org/> - 50+ Python exercises on a free, adless, simple, and open-source platform.
- [How to Think Like a Computer Scientist: Interactive Edition](https://runestone.academy/ns/books/published/t_hinkcsipy/index.html) → https://runestone.academy/ns/books/published/t_hinkcsipy/index.html is an interactive reimagination of Elkner, Downey and Meyer's book with visualizations and audio explanations.
- [LearnPython](https://www.learnpython.org/) → <https://www.learnpython.org/> is an interactive Python tutorial that is suitable for absolute beginners.
- [Learn Python](https://learn-python.adamemory.dev/) → <https://learn-python.adamemory.dev/> - A no install Python course with interactive exercises powered by Pyodide.

Resources for Younger Learners

(This section was previously called "K-12 Oriented", K-12 being a USA-centric term which refers to the primary and secondary educational stages; through level 3 on the UNESCO ISCED education levels list.)

- [Guido van Robot](http://gvr.sourceforge.net/) → <http://gvr.sourceforge.net/> A teaching tool in which students write simple programs using a Python-like language to control a simulated robot. Field-tested at Yorktown High School, the project includes a lesson plan.
- [Python for Kids](http://jasonrbriggs.com/python-for-kids/index.html) → <http://jasonrbriggs.com/python-for-kids/index.html> by Jason R Briggs. Book with sample code and puzzles.
- [PythonTurtle](http://pythonturtle.org/) → <http://pythonturtle.org/> A learning environment for Python suitable for beginners and children, inspired by Logo. Geared mainly towards children, but known to be successful with adults as well.
- [Webucator's self-paced Python 3 course](https://www.webucator.com/self-paced-training/index.cfm#?courseId=P_YT111) → https://www.webucator.com/self-paced-training/index.cfm#?courseId=P_YT111 free for homeschoilers and other students (use HOMESCHOOL as the coupon code when checking out). This course is appropriate for students 13 and up. **From our experience, these students can learn at least as quickly as adults new to programming.**

Tutorials and Websites

- [A Byte of Python](https://python.swaroopch.com/) → <https://python.swaroopch.com/>, by Swaroop C.H., is also an introductory text for people with no previous programming experience.
- [Afternerd](https://www.afternerd.com/) → <https://www.afternerd.com/>, by Karim Elghamrawy, is a Python tutorials blog that is geared towards Python beginners.
- [Ask Python](https://askpython.com/) → <https://askpython.com/> Absolute Beginners Python Tutorial.

- [Hands-on Python Tutorial](http://anh.cs.luc.edu/handsonPythonTutorial/) → <http://anh.cs.luc.edu/handsonPythonTutorial/> Beginners' Python, graphics, and simple client/server introduction, with videos.
- [Learning to Program](http://www.alan-g.me.uk/l2p2) → <http://www.alan-g.me.uk/l2p2> An introduction to programming for those who have never programmed before, by Alan Gauld. It introduces several programming languages but has a strong emphasis on Python. (Python 2 and 3)
- [ItsMyCode](https://itsmycode.com/) → <https://itsmycode.com/> A Python Blog and tutorials built for developers who love coding
- [After Hours Programming Python 3 Tutorial](https://www.afterhoursprogramming.com/tutorial/Python/Overview/) → <https://www.afterhoursprogramming.com/tutorial/Python/Overview/>
- [Letsfindcourse - Python](http://letsfindcourse.com/python) → <http://letsfindcourse.com/python>: Best Python tutorials and courses recommended by experts.
- [The Wikibooks Non-Programmer's Tutorial for Python by Josh Cogliati](http://en.wikibooks.org/wiki/Non-Programmer's_Tutorial_for_Python_3.0) → http://en.wikibooks.org/wiki/Non-Programmer's_Tutorial_for_Python_3.0
- [Online Python Courses](https://www.coursesonline.co.uk/courses/python/) → <https://www.coursesonline.co.uk/courses/python/> Compare online Python courses from learning providers from across the UK
- [Learn Python](https://overiq.com/python/3.4/intro-to-python/) → <https://overiq.com/python/3.4/intro-to-python/> An Introductory yet in-depth tutorial for Python beginners.
- The [Python tips](http://pythontips.com/) → <http://pythontips.com/> blog includes Python tips and tutorials for beginners and professional programmers.
- [Python Tutorial in Python's documentation set](http://docs.python.org/py3k/tutorial/) → <http://docs.python.org/py3k/tutorial/>. It's not written with non-programmers in mind, but it will give you an idea of the language's flavor and style.
- [The Python-Course.eu's extensive tutorial for complete beginners](http://www.python-course.eu/python3_course.php) → http://www.python-course.eu/python3_course.php, with lots of illustrations.
- [Pythonspot Tutorials](https://www.pythonspot.com/) → <https://www.pythonspot.com/> Python tutorials.
- [The Python Guru](http://thepythonguru.com/) → <http://thepythonguru.com/> A beginner-friendly guide for aspiring programmers.
- [CodersLegacy](https://coderslegacy.com/) → <https://coderslegacy.com/> A website + blog geared towards both new and experienced programmers. Mainly focused on teaching Python.
- [Discover Python & Patterns with game programming](https://www.patternsgameprog.com/series/discover-python-and-patterns/) → <https://www.patternsgameprog.com/series/discover-python-and-patterns/> Discover Python by programming video games.
- [QuizCure: A Python Learning Platform](https://www.quizcure.com/topic/python/) → <https://www.quizcure.com/topic/python/> Contains a list of Commonly asked Python Questions and Answers with Examples.

Tutorial Aggregators / lists

- [Gitconnected Python](https://gitconnected.com/learn/python) → <https://gitconnected.com/learn/python> tutorials submitted and ranked by Python developers with the best rising to the top
- [Coursesity - Python](https://coursesity.com/best-tutorials-learn/python) → <https://coursesity.com/best-tutorials-learn/python> - Curated list of the best python courses and tutorials for beginners.
- [Classpert - Python](https://classpert.com/python-programming) → <https://classpert.com/python-programming> - A large collection of free and paid Python online courses, from a wide range of providers.
- [Hackr.io - Python](https://hackr.io/tutorials/learn-python) → <https://hackr.io/tutorials/learn-python>: Programming community-recommended best Python tutorials and courses

Tutorials for Scientific Audiences

These websites are written in support of science courses but are general enough that anyone can learn from them.

- [Beginning Python for Bioinformatics](http://www.onlamp.com/pub/a/python/2002/10/17/biopython.html) → <http://www.onlamp.com/pub/a/python/2002/10/17/biopython.html> by Patrick O'Brien. An introduction to Python aimed at biologists that introduces the [PyCrust](https://wiki.python.org/moin/PyCrust) → <https://wiki.python.org/moin/PyCrust> shell and Python's basic data types.

- [Python for Number Theory](http://illustratedtheoryofnumbers.com/prog.html) → <http://illustratedtheoryofnumbers.com/prog.html> is a series of Python notebooks (for Jupyter) for applications to number theory and cryptography. They assume no prior programming experience and are suitable for someone learning elementary number theory at the same time. They conclude with an introduction to primality testing and cryptography (Diffie-Hellman, RSA).

Apps

- [Programiz App to Learn Python](https://www.programiz.com/learn-python) → <https://www.programiz.com/learn-python> - A beginner-friendly app on Android and iOS to learn Python step by step with an in-built interpreter and quizzes.

Videos

- [Python Programming Tutorials for Beginners](https://youtu.be/uCzFUKWtzgA?list=PLboXykqtm8dy_DNg1NZiS08Dnyj35PWxw) → https://youtu.be/uCzFUKWtzgA?list=PLboXykqtm8dy_DNg1NZiS08Dnyj35PWxw: Installation, IDE, variables, functions, strings, lists, OOP
- The [Young Programmers Podcast](http://youngprogrammers.blogspot.com/search/label/python) → <http://youngprogrammers.blogspot.com/search/label/python> contains video lessons on Python, Pygame, Jython, Scratch, Alice, Java, and Scala (somewhat outdated content!)

Email Academies

- [Finxter Email Computer Science Academy](https://blog.finxter.com/email-academy/) → <https://blog.finxter.com/email-academy/>: 20+ free Python and computer science courses delivered in email video lessons. **Content:** cheat sheets, Python basics, data structures, [NumPy](https://wiki.python.org/moin/NumPy) → <https://wiki.python.org/moin/NumPy>, data science, career advancement, coding productivity, and machine learning.
 - [Thonny, Python IDE for beginners](http://thonny.org/) → <http://thonny.org/>
-

[CategoryPythonInEducation](https://wiki.python.org/moin/CategoryPythonInEducation) → <https://wiki.python.org/moin/CategoryPythonInEducation> [CategoryPythonInEducation](https://wiki.python.org/moin/CategoryPythonInEducation) → <https://wiki.python.org/moin/CategoryPythonInEducation>

BeginnersGuide/Programmers - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Programmers>

Please Note

This is a Wiki page. Users with edit rights can edit it. You are, therefore, free to (in fact, encouraged to) add details of material that other Python users will find useful. It is **not** an advertising page and is here to serve the whole Python community. Users who continually edit pages to give their own materials (particularly commercial materials) prominence, or spam the listing with multiple entries which point to resources with only slightly altered material, may subsequently find their editing rights disabled. *You have been warned.* On a cheerier note - there is a constant stream of new and updated information on Python as the language is exploding in popularity. Only enthusiastic volunteers can keep this page current, so if something helps you, feel free to link it here.

The tutorials on this page aim at people with previous experience with other programming languages (C, Perl, Lisp, Visual Basic, etc.). Also of potential interest are such related Beginners Guides as [BeginnersGuide/Overview](https://wiki.python.org/moin/BeginnersGuide/Overview) → <https://wiki.python.org/moin/BeginnersGuide/Overview> and [BeginnersGuide/NonProgrammers](https://wiki.python.org/moin/BeginnersGuide/NonProgrammers) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers>, and the tips in [MovingToPythonFromOtherLanguages](https://wiki.python.org/moin/MovingToPythonFromOtherLanguages) → <https://wiki.python.org/moin/MovingToPythonFromOtherLanguages>.

Books, Websites, Tutorials (non-interactive)

Resources

- [A beginner-friendly Python course](https://programiz.pro/learn/master-python) → <https://programiz.pro/learn/master-python> with interactive, bite-size lessons, and over 100 challenges.
- [A beginner-friendly Python tutorial](https://python.land/python-tutorial) → <https://python.land/python-tutorial> that starts with the absolute basics but also covers more advanced stuff like Python software deployment.
- [A Byte of Python](https://python.swaroopch.com/) → <https://python.swaroopch.com/>, by Swaroop C.H. An introductory text for beginners and experienced programmers looking to learn Python.
- [After Hours Programming's Python Introduction](http://www.afterhoursprogramming.com/tutorial/Python/introduction/) → <http://www.afterhoursprogramming.com/tutorial/Python/introduction/> A beginners introduction into Python.
- [Awesome Python](https://awesome-python.com/) → <https://awesome-python.com/> A curated list of awesome Python frameworks, libraries, software and resources.
- [CheckiO interactive learning resource](https://www.checkio.org/) → <https://www.checkio.org/> Creative way to improve Python skills with interesting tasks, it also supports Python 3|2.
- [Classpert - Python](https://classpert.com/python-programming) → <https://classpert.com/python-programming> - A collection of free and paid Python online courses from a wide range of providers.
- [Codedex](https://www.codedex.io.com/) → <https://www.codedex.io.com/> - A learn to code platform for K-12 and college students.
- [CodersLegacy](https://coderslegacy.com/) → <https://coderslegacy.com/> A website + blog geared towards both new and experienced programmers. Mainly focused on teaching Python.
- [Dive Into Python 3](https://diveintopython3.problemsolving.io/) → <https://diveintopython3.problemsolving.io/> by Mark Pilgrim.
- [Elements of Python Style](https://github.com/ambianti/elements-of-python-style) → <https://github.com/ambianti/elements-of-python-style> This document goes beyond PEP8 to cover the core of what the author thinks of as great Python style.
- [Finxter](https://finxter.com/) → <https://finxter.com/> - Solve Python puzzles and test your Python skill level (beginner to grandmaster level).
- [Full Stack Python](https://www.fullstackpython.com/) → <https://www.fullstackpython.com/> Once you know the basics, learn how to build, deploy and operate Python Applications.

- [ItsMyCode](https://itsmycode.com/) → <https://itsmycode.com/> A Python Programming Blog which teaches Python basics and helps to solve various issues which developers face in day to day Programming
- [Python 3 Patterns, Recipes, and Idioms](https://python-3-patterns-idioms-test.readthedocs.io/en/latest/) → <https://python-3-patterns-idioms-test.readthedocs.io/en/latest/> by Bruce Eckel and Friends.
- [Learn Python Step by Step](https://www.techbeamers.com/python-tutorial-step-by-step/) → <https://www.techbeamers.com/python-tutorial-step-by-step/> - Start learning python from the basics to pro-level and attain proficiency.
- [Learn Python OverIQ](https://overiq.com/python/3.4/intro-to-python) → <https://overiq.com/python/3.4/intro-to-python> - An entry-level course to get you started with Python Programming.
- [Learn Python - Tutorial for Beginners](https://www.programiz.com/python-programming) → <https://www.programiz.com/python-programming> A comprehensive Python guide to get started, Python tutorials, and examples for beginners.
- [Free python tips and tutorials](http://freepythontips.wordpress.com/) → <http://freepythontips.wordpress.com/> Python tips and tutorials for beginners and professional programmers.
- [Intro to Python](http://stromberg.dnsalias.org/~dstromberg/Intro-to-Python/) → <http://stromberg.dnsalias.org/~dstromberg/Intro-to-Python/> - A Brief Presentation about Python mainly aimed at experienced programmers. Might be nice as a first pass over the language.
- [Learn Python in 10 minutes](https://www.stavros.io/tutorials/python/) → <https://www.stavros.io/tutorials/python/>
- [Python Course](http://www.python-course.eu/) → <http://www.python-course.eu/> - This online Python course is aiming at beginners and with advanced topics at experienced programmers as well.
- [Python Koans](https://github.com/gregmalcolm/python_koans) → https://github.com/gregmalcolm/python_koans Learn Python through TDD
- [Python Programming for Beginners](http://www.linuxjournal.com/lj-issues/issue73/3946.html) → <http://www.linuxjournal.com/lj-issues/issue73/3946.html> A short introduction to writing command-line applications in Python by Jacek Artymiaik.
- [PythonSpeed.com](https://pythonspeed.com/performance/) → <https://pythonspeed.com/performance/> Great resource with insightful ways to speed up your Python code
- [Python Essential Reference](https://www.dabeaz.com/per.html) → <https://www.dabeaz.com/per.html> (book) If you want a highly compressed K&R-style 'just the facts' overview, David Beazley's "Python Essential Reference" covers practically all of the language in about a hundred pages. A version that covers Python 3.7 is in progress.
- [Resources for Learning Python](http://codecondo.com/10-ways-to-learn-python/) → <http://codecondo.com/10-ways-to-learn-python/> 10 of the most popular / recommended platforms in the World when it comes to learning Python, either as a complete beginner or someone who knows their way around.
- [Python Tutorial](http://docs.python.org/tut/) → <http://docs.python.org/tut/> This tutorial is part of Python's documentation set and is updated with each new release.
- [Wikiversity:Python](http://en.wikiversity.org/wiki/Topic:Python) → <http://en.wikiversity.org/wiki/Topic:Python> The Wiki(anything) information about Python.
- [Python Programming Tutorials](https://pythonspot.com/) → <https://pythonspot.com/> Python programming tutorials.
- [Python Tutorials](http://thepythonguru.com/getting-started-with-python/) → <http://thepythonguru.com/getting-started-with-python/> Python in plain English.
- [Learn Python - Programming Made Easy](http://www.trytoprogram.com/python-programming/) → <http://www.trytoprogram.com/python-programming/> Simplified tutorials for beginners (Learn with relevant examples).
- [Pandas Cookbook](https://tutswiki.com/pandas-cookbook/) → <https://tutswiki.com/pandas-cookbook/> A newbie friendly introduction to pandas with real-life examples.
- [Ultimate Python study guide](https://github.com/huangsam/ultimate-python) → <https://github.com/huangsam/ultimate-python> Ultimate Python study guide for newcomers and professionals alike.
- [Learn coding with Python notebooks](https://www.nbshare.io/) → <https://www.nbshare.io/> A place where users can learn lot about Python coding with Python notebooks.
- [Learn Python Programming](https://www.scaler.com/topics/python/) → <https://www.scaler.com/topics/python/> Easy to understand Python tutorial explained with examples for beginners and professionals alike.
- [Computer Science Circles](http://cscircles.cemc.uwaterloo.ca/) → <http://cscircles.cemc.uwaterloo.ca/>
- [HackInScience: free and open-source Python training website](https://hackinscience.org/) → <https://hackinscience.org/>
- [Learn Python](https://learn-python.adamemory.dev/) → <https://learn-python.adamemory.dev/> - A no install Python course with interactive exercises powered by Pyodide.

- [Python Editor](https://python-editor.adamemery.dev/) → <https://python-editor.adamemery.dev/> - A web app for writing and running basic Python scripts
- [Python visualizer tool](http://people.csail.mit.edu/pgbovine/python/tutor.html) → <http://people.csail.mit.edu/pgbovine/python/tutor.html>
- [Thonny](http://thonny.org/). Python IDE for beginners. Has intuitive features for program runtime visualization → <http://thonny.org/>
- [PyFlo](https://pyflo.net/) → <https://pyflo.net/> - A free, interactive guide to becoming a Python Programmer

Python Video Tutorials

- <https://www.youtube.com/watch?v=rfscVS0vtbw> - Python Beginners Course by [FreeCodeCamp](#) → <https://wiki.python.org/moin/FreeCodeCamp> (4 hours)
- <https://www.youtube.com/watch?v=HGOBQPFzWKO> - Python Intermediate Course by [FreeCodeCamp](#) → <https://wiki.python.org/moin/FreeCodeCamp> (6 hours)
- <https://www.webucator.com/django-training/course/writing-your-first-django-app/> - A free Django course with videos based on the Django Software Foundations official tutorial.
- [Crawl the Web With Python](#) → <https://code.tutsplus.com/courses/crawl-the-web-with-python> - learn to build a web crawler and scraper (paid/commercial).
- [Django Basics](#) → <https://overiq.com/django/1.10/intro-to-django/> - An introductory course to learn basics of Django framework in great detail.
- [MIT's Introduction to Computer Science and Programming in Python](#) → <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-videos/>
- [Khan Academy computer science](#) → <https://www.khanacademy.org/computing/computer-science> playlist teaches Python.
- [Python Exception Handling for beginners](#) → <https://youtu.be/HJSlyzm4j6Y> - Exception handling with Python.
- [Python Lists and Object Tutorial for Beginners](#) → https://www.youtube.com/watch?v=dFLD3JsvJo&list=PLboXykqt_m8dy_DNg1NZiS08Dnyj35PWXw - Sorting Objects with Python.
- [Python OOP Tutorial for Beginners](#) → <https://youtu.be/RZF17FfRllo> - Getting started with OOP programming with Python.
- [Python Screencasts](#) → <https://www.youtube.com/playlist?list=PLlgoYPTU6ljCEggReCMF0m0760QTot9Qz> (36 videos)

Python video tutorial (commercial/paid)

- [Getting Started With Django](#) → <https://code.tutsplus.com/courses/getting-started-with-django> - learn the Django back-end framework from scratch (paid/commercial)
- [Build a News Aggregator With Django](#) → <https://code.tutsplus.com/courses/build-a-news-aggregator-with-django> - learn advanced Django skills with a hands-on project (paid/commercial)
- [Data Handling With Python](#) → <https://code.tutsplus.com/courses/data-handling-with-python> - learn the basics of handling data in the Python language (paid/commercial).

Free Python Courses

- [Free Python 3 email course](#) → <https://blog.finxters.com/subscribe> (almost daily Python lesson + cheat sheets, email required)

Other Python Resource Aggregators

- [Learn Python - Best Python Tutorials and Courses](https://hackr.io/tutorials/learn-python) → <https://hackr.io/tutorials/learn-python> Python tutorials & courses recommended by the programming community.
 - [Learn Python - Best Python Courses](https://gitconnected.com/learn/python) → <https://gitconnected.com/learn/python> Python tutorials submitted and ranked by Python developers with the best rising to the top
 - [Paid Python 3 course](https://www.codecademy.com/learn/learn-python-3) → <https://www.codecademy.com/learn/learn-python-3> (almost daily Python lesson + cheat sheets)
-

[CategoryPythonWebsite](https://wiki.python.org/moin/CategoryPythonWebsite) → <https://wiki.python.org/moin/CategoryPythonWebsite> [CategoryCategory](https://wiki.python.org/moin/CategoryCategory) → <https://wiki.python.org/moin/CategoryCategory>

Languages - Python Wiki

Source: <https://wiki.python.org/moin/Languages>

Attempt to have languages and links listed in the native tongue of the user.

Ideally, all the pages should be like the Polish or Turkish pages - all native language, only the necessary English.

There are some ground rules, some are laid down by the site admins, some are my suggestions:

1. Pages must be named in ASCII and English ([PolishLanguage](https://wiki.python.org/moin/PolishLanguage) → <https://wiki.python.org/moin/PolishLanguage>)
2. Pages must have an explanation in English at the top (Links to Python information in <language X>)
3. (my suggestion) We probably want to limit invites to edit the pages to people we know well, or Pythonistas with a track record. Hopefully, this is inclusive enough without opening the site up to a spam flood and vandalismfest.

Where these pages really need help:

1. check links, remove broken ones.
2. add new links that are quality Python information and active.
3. some care for languages that have next to nothing, but do have people in the Python community - even a link to the Wikipedia page for Python, in that language, is a start (Some are pretty complete and of high quality - the Russian language Wikipedia page for Python, for instance, packs a lot in).
 - [AfrikaansLanguage](https://wiki.python.org/moin/AfrikaansLanguage) → <https://wiki.python.org/moin/AfrikaansLanguage> Afrikaans
 - [AlbanianLanguage](https://wiki.python.org/moin/AlbanianLanguage) → <https://wiki.python.org/moin/AlbanianLanguage> Shqip
 - [AmharicLanguage](https://wiki.python.org/moin/AmharicLanguage) → <https://wiki.python.org/moin/AmharicLanguage> አማርኛ
 - [ArabicLanguage](https://wiki.python.org/moin/ArabicLanguage) → <https://wiki.python.org/moin/ArabicLanguage> العربية
 - [ArmenianLanguage](https://wiki.python.org/moin/ArmenianLanguage) → <https://wiki.python.org/moin/ArmenianLanguage> Հայերեն
 - [AssameseLanguage](https://wiki.python.org/moin/AssameseLanguage) → <https://wiki.python.org/moin/AssameseLanguage>
 - [AzerbaijaniLanguage](https://wiki.python.org/moin/AzerbaijaniLanguage) → <https://wiki.python.org/moin/AzerbaijaniLanguage> Azərbaycan dili
 - [BelarusianLanguage](https://wiki.python.org/moin/BelarusianLanguage) → <https://wiki.python.org/moin/BelarusianLanguage> Беларуская мова
 - [BengaliLanguage](https://wiki.python.org/moin/BengaliLanguage) → <https://wiki.python.org/moin/BengaliLanguage>
 - [BodoLanguage](https://wiki.python.org/moin/BodoLanguage) → <https://wiki.python.org/moin/BodoLanguage> বড়ো
 - [BosnianLanguage](https://wiki.python.org/moin/BosnianLanguage) → <https://wiki.python.org/moin/BosnianLanguage> bosanski
 - [BulgarianLanguage](https://wiki.python.org/moin/BulgarianLanguage) → <https://wiki.python.org/moin/BulgarianLanguage> български език
 - [BurmeseLanguage](https://wiki.python.org/moin/BurmeseLanguage) → <https://wiki.python.org/moin/BurmeseLanguage>
 - [CatalanLanguage](https://wiki.python.org/moin/CatalanLanguage) → <https://wiki.python.org/moin/CatalanLanguage> català
 - [ChineseLanguage](https://wiki.python.org/moin/ChineseLanguage) → <https://wiki.python.org/moin/ChineseLanguage>
 - [CroatianLanguage](https://wiki.python.org/moin/CroatianLanguage) → <https://wiki.python.org/moin/CroatianLanguage> hrvatski
 - [CzechLanguage](https://wiki.python.org/moin/CzechLanguage) → <https://wiki.python.org/moin/CzechLanguage> českina
 - [DanishLanguage](https://wiki.python.org/moin/DanishLanguage) → <https://wiki.python.org/moin/DanishLanguage> dansk
 - [DogriLanguage](https://wiki.python.org/moin/DogriLanguage) → <https://wiki.python.org/moin/DogriLanguage> ଡୋଗରୀ Devanagari script
 - [DutchLanguage](https://wiki.python.org/moin/DutchLanguage) → <https://wiki.python.org/moin/DutchLanguage> Nederlands
 - [EsperantoLanguage](https://wiki.python.org/moin/EsperantoLanguage) → <https://wiki.python.org/moin/EsperantoLanguage> Esperanto
 - [EstonianLanguage](https://wiki.python.org/moin/EstonianLanguage) → <https://wiki.python.org/moin/EstonianLanguage> eesti keel
 - [FinnishLanguage](https://wiki.python.org/moin/FinnishLanguage) → <https://wiki.python.org/moin/FinnishLanguage> suomi
 - [FrenchLanguage](https://wiki.python.org/moin/FrenchLanguage) → <https://wiki.python.org/moin/FrenchLanguage> français
 - [GeorgianLanguage](https://wiki.python.org/moin/GeorgianLanguage) → <https://wiki.python.org/moin/GeorgianLanguage> ქართული ენა
 - [GermanLanguage](https://wiki.python.org/moin/GermanLanguage) → <https://wiki.python.org/moin/GermanLanguage> Deutsch

- [GreekLanguage](https://wiki.python.org/moin/GreekLanguage) → **Νέα Ελληνικά**
- [GujaratiLanguage](https://wiki.python.org/moin/GujaratiLanguage) → **ગુજરાતી**
- [HausaLanguage](https://wiki.python.org/moin/HausaLanguage) → **هائزا**
- [HebrewLanguage](https://wiki.python.org/moin/HebrewLanguage) → **עברית**
- [HindiLanguage](https://wiki.python.org/moin/HindiLanguage) → **हिन्दी**
- [HungarianLanguage](https://wiki.python.org/moin/HungarianLanguage) → **magyar nyelv**
- [IndonesianLanguage](https://wiki.python.org/moin/IndonesianLanguage) → **Bahasa Indonesia**
- [IcelandicLanguage](https://wiki.python.org/moin/IcelandicLanguage) → **íslenska**
- [IgboLanguage](https://wiki.python.org/moin/IgboLanguage) → **Asusu Igbo**
- [ItalianLanguage](https://wiki.python.org/moin/ItalianLanguage) → **italiano**
- [JapaneseLanguage](https://wiki.python.org/moin/JapaneseLanguage) → **日本語**
- [KannadaLanguage](https://wiki.python.org/moin/KannadaLanguage) → **ಕನ್ನಡ**
- [KashmiriLanguage](https://wiki.python.org/moin/KashmiriLanguage) → **کاشمیری (Koshur)**
- [KazakhLanguage](https://wiki.python.org/moin/KazakhLanguage) → **Қазақ тілі**
- [KhmerLanguage](https://wiki.python.org/moin/KhmerLanguage) → **ខ្មែរ**
- [KonkaniLanguage](https://wiki.python.org/moin/KonkaniLanguage) → **ಕಂಕಣಿ** Devangari script
- [KoreanLanguage](https://wiki.python.org/moin/KoreanLanguage) → **/**
- [LaoLanguage](https://wiki.python.org/moin/LaoLanguage) → **ພາສັນລະນະ**
- [LatvianLanguage](https://wiki.python.org/moin/LatvianLanguage) → **latviešu valoda**
- [LithuanianLanguage](https://wiki.python.org/moin/LithuanianLanguage) → **lietuvių kalba**
- [MalayLanguage](https://wiki.python.org/moin/MalayLanguage) → **Bahasa Melayu**
- [MalayalamLanguage](https://wiki.python.org/moin/MalayalamLanguage) → **മലയാളം**
- [MarathiLanguage](https://wiki.python.org/moin/MarathiLanguage) → **मराठी**
- [MongolianLanguage](https://wiki.python.org/moin/MongolianLanguage) → **Монгол хэл**
- [NepaliLanguage](https://wiki.python.org/moin/NepaliLanguage) → **नेपाली**
- [NorwegianLanguage](https://wiki.python.org/moin/NorwegianLanguage) → **norsk**
- [OriyaLanguage](https://wiki.python.org/moin/OriyaLanguage) → **ଓଡ଼ିଆ**
- [OromoLanguage](https://wiki.python.org/moin/OromoLanguage) → **Afaan Oromoo**
- [PersianLanguage](https://wiki.python.org/moin/PersianLanguage) → **فارسی**
- [PolishLanguage](https://wiki.python.org/moin/PolishLanguage) → **język polski**
- [PortugueseLanguage](https://wiki.python.org/moin/PortugueseLanguage) → **português**
- [PunjabiLanguage](https://wiki.python.org/moin/PunjabiLanguage) → **ਪੰਜਾਬੀ**
- [WesternPunjabiLanguage](https://wiki.python.org/moin/WesternPunjabiLanguage) → **پنجابی**
- [RomanianLanguage](https://wiki.python.org/moin/RomanianLanguage) → **limba română**
- [RussianLanguage](https://wiki.python.org/moin/RussianLanguage) → **русский язык**
- [SanskritLanguage](https://wiki.python.org/moin/SanskritLanguage) → **संस्कृतम्**
- [SlovakLanguage](https://wiki.python.org/moin/SlovakLanguage) → **slovenský jazyk**
- [SloveneLanguage](https://wiki.python.org/moin/SloveneLanguage) → **slovenščina**
- [SerbianLanguage](https://wiki.python.org/moin/SerbianLanguage) → **Српски**
- [SinhalaLanguage](https://wiki.python.org/moin/SinhalaLanguage) → **සිංහල**
- [SpanishLanguage](https://wiki.python.org/moin/SpanishLanguage) → **español**
- [SwahiliLanguage](https://wiki.python.org/moin/SwahiliLanguage) → **Kiswahili**
- [SwedishLanguage](https://wiki.python.org/moin/SwedishLanguage) → **svenska**
- [TagalogLanguage](https://wiki.python.org/moin/TagalogLanguage) → **Wikang Tagalog**
- [TamilLanguage](https://wiki.python.org/moin/TamilLanguage) → **தமிழ்**
- [TeluguLanguage](https://wiki.python.org/moin/TeluguLanguage) → **తెలుగు**
- [ThaiLanguage](https://wiki.python.org/moin/ThaiLanguage) → **ไทย**
- [TigrinyaLanguage](https://wiki.python.org/moin/TigrinyaLanguage) → **ትግርኛ**

- [TurkishLanguage](https://wiki.python.org/moin/TurkishLanguage) → https://wiki.python.org/moin/TurkishLanguage **Türkçe**
- [UkrainianLanguage](https://wiki.python.org/moin/UkrainianLanguage) → https://wiki.python.org/moin/UkrainianLanguage **українська мова**
- [UrduLanguage](https://wiki.python.org/moin/UrduLanguage) → https://wiki.python.org/moin/UrduLanguage **اردو**
- [UzbekLanguage](https://wiki.python.org/moin/UzbekLanguage) → https://wiki.python.org/moin/UzbekLanguage **O'zbek tilı**
- [Vietnameselanguage](https://wiki.python.org/moin/Vietnameselanguage) → https://wiki.python.org/moin/Vietnameselanguage **tiếng Việt**
- [XhosaLanguage](https://wiki.python.org/moin/XhosaLanguage) → https://wiki.python.org/moin/XhosaLanguage **isiXhosa**
- [ZuluLanguage](https://wiki.python.org/moin/ZuluLanguage) → https://wiki.python.org/moin/ZuluLanguage **isiZulu**

[CategoryLanguage](https://wiki.python.org/moin/CategoryLanguage) → https://wiki.python.org/moin/CategoryLanguage

[CategoryUnicode](https://wiki.python.org/moin/CategoryUnicode) → https://wiki.python.org/moin/CategoryUnicode [CategoryLanguage](https://wiki.python.org/moin/CategoryLanguage) → https://wiki.python.org/moin/CategoryLanguage

Python on Windows FAQ

Source: <http://www.python.org/doc/faq/windows/#how-do-i-run-a-python-program-under-windows>

Contents

- [Python on Windows FAQ](#)
 - [How do I run a Python program under Windows?](#)
 - [How do I make Python scripts executable?](#)
 - [Why does Python sometimes take so long to start?](#)
 - [How do I make an executable from a Python script?](#)
 - [Is a *.pyd file the same as a DLL?](#)
 - [How can I embed Python into a Windows application?](#)
 - [How do I keep editors from inserting tabs into my Python source?](#)
 - [How do I check for a keypress without blocking?](#)
 - [How do I solve the missing api-ms-win-crt-runtime-l1-1-0.dll error?](#)

How do I run a Python program under Windows?¶

This is not necessarily a straightforward question. If you are already familiar with running programs from the Windows command line then everything will seem obvious; otherwise, you might need a little more guidance.

Unless you use some sort of integrated development environment, you will end up *typing* Windows commands into what is referred to as a “Command prompt window”. Usually you can create such a window from your search bar by searching for cmd. You should be able to recognize when you have started such a window because you will see a Windows “command prompt”, which usually looks like this:

The letter may be different, and there might be other things after it, so you might just as easily see something like:

```
D:\YourName\Projects\Python>
```

depending on how your computer has been set up and what else you have recently done with it. Once you have started such a window, you are well on the way to running Python programs.

You need to realize that your Python scripts have to be processed by another program called the Python *interpreter*. The interpreter reads your script, compiles it into bytecodes, and then executes the bytecodes to run your program. So, how do you arrange for the interpreter to handle your Python?

First, you need to make sure that your command window recognises the word “py” as an instruction to start the interpreter. If you have opened a command window, you should try entering the command py and hitting return:

You should then see something like:

```
Python 3.6.4 (v3.6.4:d48ebeb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

You have started the interpreter in “interactive mode”. That means you can enter Python statements or expressions interactively and have them executed or evaluated while you wait. This is one of Python’s strongest features. Check it by entering a few expressions of your choice and seeing the results:

```
>>> print("Hello")
Hello
>>> "Hello" * 3
'HelloHelloHello'
```

Many people use the interactive mode as a convenient yet highly programmable calculator. When you want to end your interactive Python session, call the [exit\(\)](http://www.python.org/doc/faq/library/constants.html#exit) function or hold the **Ctrl** key down while you enter a **Z**, then hit the **Enter** key to get back to your Windows command prompt.

You may also find that you have a Start-menu entry such that results in you seeing the >>> prompt in a new window. If so, the window will disappear after you call the [exit\(\)](http://www.python.org/doc/faq/library/constants.html#exit) function or enter the **[Ctrl]-[Z]** character; Windows is running a single “python” command in the window, and closes it when you terminate the interpreter.

Now that we know the py command is recognized, you can give your Python script to it. You’ll have to give either an absolute or a relative path to the Python script. Let’s say your Python script is located in your desktop and is named `hello.py`, and your command prompt is nicely opened in your home directory so you’re seeing something similar to:

So now you’ll ask the py command to give your script to Python by typing `py` followed by your script path:

```
C:\Users\YourName> py Desktop\hello.py
hello
```

How do I make Python scripts executable?

On Windows, the standard Python installer already associates the .py extension with a file type (Python.File) and gives that file type an open command that runs the interpreter (`D:\Program Files\Python\python.exe "%1" %*`). This is enough to make scripts executable from the command prompt as ‘`foo.py`’. If you’d rather be able to execute the script by simple typing ‘`foo`’ with no extension you need to add `.py` to the PATHEXT environment variable.

Why does Python sometimes take so long to start?

Usually Python starts very quickly on Windows, but occasionally there are bug reports that Python suddenly begins to take a long time to start up. This is made even more puzzling because Python will work fine on other Windows systems which appear to be configured identically.

The problem may be caused by a misconfiguration of virus checking software on the problem machine. Some virus scanners have been known to introduce startup overhead of two orders of magnitude when the scanner is configured to monitor all reads from the filesystem. Try checking the configuration of virus scanning software on your systems to ensure that they are indeed configured identically. McAfee, when configured to scan all file system read activity, is a particular offender.

How do I make an executable from a Python script?

See [How can I create a stand-alone binary from a Python script? → http://www.python.org/doc/faq/windows/programming.html#faq-create-standalone-binary](http://www.python.org/doc/faq/windows/programming.html#faq-create-standalone-binary) for a list of tools that can be used to make executables.

Is a *.pyd file the same as a DLL?

Yes, .pyd files are DLL's, but there are a few differences. If you have a DLL named `foo.pyd`, then it must have a function `PyInit_foo()`. You can then write Python "import foo", and Python will search for `foo.pyd` (as well as `foo.py`, `foo.pyc`) and if it finds it, will attempt to call `PyInit_foo()` to initialize it. You do not link your .exe with `foo.lib`, as that would cause Windows to require the DLL to be present.

Note that the search path for `foo.pyd` is PYTHONPATH, not the same as the path that Windows uses to search for `foo.dll`. Also, `foo.pyd` need not be present to run your program, whereas if you linked your program with a DLL, the DLL is required. Of course, `foo.pyd` is required if you want to say `import foo`. In a DLL, linkage is declared in the source code with `__declspec(dllexport)`. In a .pyd, linkage is defined in a list of available functions.

How can I embed Python into a Windows application?

Embedding the Python interpreter in a Windows app can be summarized as follows:

1. Do **not** build Python into your .exe file directly. On Windows, Python must be a DLL to handle importing modules that are themselves DLL's. (This is the first key undocumented fact.) Instead, link to `pythonNN.dll`; it is typically installed in `C:\Windows\System`. *NN* is the Python version, a number such as "33" for Python 3.3.

You can link to Python in two different ways. Load-time linking means linking against `pythonNN.lib`, while run-time linking means linking against `pythonNN.dll`. (General note: `pythonNN.lib` is the so-called "import lib" corresponding to `pythonNN.dll`. It merely defines symbols for the linker.)

Run-time linking greatly simplifies link options; everything happens at run time. Your code must load `pythonNN.dll` using the Windows `LoadLibraryEx()` routine. The code must also use access routines and data in `pythonNN.dll` (that is, Python's C API's) using pointers obtained by the Windows `GetProcAddress()` routine. Macros can make using these pointers transparent to any C code that calls routines in Python's C API.

2. If you use SWIG, it is easy to create a Python "extension module" that will make the app's data and methods available to Python. SWIG will handle just about all the grungy details for you. The result is C code that you link *into* your .exe file (!) You do **not** have to create a DLL file, and this also simplifies linking.
3. SWIG will create an init function (a C function) whose name depends on the name of the extension module. For example, if the name of the module is `leo`, the init function will be called `initleo()`. If you use SWIG shadow classes, as you should, the init function will be called `initleoc()`. This initializes a mostly hidden helper class used by the shadow class.

The reason you can link the C code in step 2 into your .exe file is that calling the initialization function is equivalent to importing the module into Python! (This is the second key undocumented fact.)

4. In short, you can use the following code to initialize the Python interpreter with your extension module.

```
#include <Python.h>
...
Py_Initialize(); // Initialize Python.
initmyAppc(); // Initialize (import) the helper class.
PyRun_SimpleString("import myApp"); // Import the shadow class.
```

5. There are two problems with Python's C API which will become apparent if you use a compiler other than MSVC, the compiler used to build pythonNN.dll.

Problem 1: The so-called "Very High Level" functions that take `FILE *` arguments will not work in a multi-compiler environment because each compiler's notion of a `struct FILE` will be different. From an implementation standpoint these are very low level functions.

Problem 2: SWIG generates the following code when generating wrappers to void functions:

```
Py_INCREF(Py_None);
_resultobj = Py_None;
return _resultobj;
```

Alas, `Py_None` is a macro that expands to a reference to a complex data structure called `_Py_NoneStruct` inside pythonNN.dll. Again, this code will fail in a mult-compiler environment. Replace such code by:

```
return Py_BuildValue("");
```

It may be possible to use SWIG's `%typemap` command to make the change automatically, though I have not been able to get this to work (I'm a complete SWIG newbie).

6. Using a Python shell script to put up a Python interpreter window from inside your Windows app is not a good idea; the resulting window will be independent of your app's windowing system. Rather, you (or the `wxPythonWindow` class) should create a "native" interpreter window. It is easy to connect that window to the Python interpreter. You can redirect Python's i/o to `_any_` object that supports read and write, so all you need is a Python object (defined in your extension module) that contains `read()` and `write()` methods.

How do I keep editors from inserting tabs into my Python source?¶

The FAQ does not recommend using tabs, and the Python style guide, [PEP 8 → https://peps.python.org/pep-0008/](https://peps.python.org/pep-0008/), recommends 4 spaces for distributed Python code; this is also the Emacs python-mode default.

Under any editor, mixing tabs and spaces is a bad idea. MSVC is no different in this respect, and is easily configured to use spaces: Take `,` and for file type "Default" set "Tab size" and "Indent size" to 4, and select the "Insert spaces" radio button.

Python raises [IndentationError → http://www.python.org/doc/faq/library/exceptions.html#IndentationError](http://www.python.org/doc/faq/library/exceptions.html#IndentationError) or [TabError → http://www.python.org/doc/faq/library/exceptions.html#TabError](http://www.python.org/doc/faq/library/exceptions.html#TabError) if mixed tabs and spaces are causing problems in leading whitespace. You may also run the `tabnanny` → <http://www.python.org/doc/faq/library/tabnanny.html#module-tabnanny> module to check a directory tree in batch mode.

How do I check for a keypress without blocking?

Use the [msvcrt](http://www.python.org/doc/faq/library/msvcrt.html#module-msvcrt) module. This is a standard Windows-specific extension module. It defines a function `kbhit()` which checks whether a keyboard hit is present, and `getch()` which gets one character without echoing it.

How do I solve the missing api-ms-win-crt-runtime-l1-1-0.dll error?

This can occur on Python 3.5 and later when using Windows 8.1 or earlier without all updates having been installed. First ensure your operating system is supported and is up to date, and if that does not resolve the issue, visit the [Microsoft support page](https://support.microsoft.com/en-us/help/3118401/) for guidance on manually installing the C Runtime update.

Beginner's Python Tutorial: Learn Python

Source: <https://python.land/python-tutorial>

Learn Python with our free beginner's Python tutorial. It contains carefully crafted, logically ordered Python articles full of information, advice, and Python practice! Hence, it helps both complete beginners and those with prior programming experience get up to speed with Python.

If you want to earn a certificate, track your progress, and have access to premium support, please consider our [Python Fundamentals course](#) → <https://python.land/product/python-course>.

Table of Contents

- [1 How to learn Python?](#)
- [2 Why learn Python?](#)
- [3 Why this Python tutorial?](#)
- [4 About the instructor](#)
- [5 Python history](#)
- [6 Run Your First Python Program](#)
- [7 Navigating the free Python tutorial](#)
- [8 How can you help me?](#)
- [9 Let's go and learn Python!](#)

How to learn Python?

If you're in a hurry to learn Python, I'll give you some shortcuts to get you started quickly. You have two options:

1. Our premium Python course, [Python Fundamentals](#) → <https://python.land/product/python-course>, is the fastest and easiest way to learn. The course is designed to learn Python quickly but properly, without distractions, using lots of quizzes, exercises, and a certificate of completion that you can add to your resume.
2. Use the free Python tutorial that we kick off right on this page. Simply keep reading. You can always decide to join the course at a later moment.

Since you're here, you probably know why, but let's go over the advantages of Python quickly!

Python is one of the world's most used and most popular programming languages. It's powerful, versatile, and easy to learn. Python is widely used in various applications, some notable ones:

- Web development
- Data Science
- Data analysis
- Machine learning
- Artificial Intelligence (AI)
- Scripting and tooling

Many people say that **Python comes with batteries included**. It's a fun way to state that it includes a comprehensive base library. In addition, because so many people use Python, hundreds of thousands of high-quality libraries and frameworks exist to get things done quickly and without hassle. You can do a lot with a little bit of Python code!

Learning Python is a no-brainer, and I promise you will be up and running quickly with this Python tutorial. Regardless of your future in IT, it will be a helpful tool to have in your toolbox!

Why *this* Python tutorial?

And here's why you should read this Python tutorial instead of all the others:

- This free Python tutorial is **easy-to-read, in plain English**.
- It's written by an **experienced writer and tutor** who puts great care into the learning material and the order in which it is presented.
- This course contains **interactive example code** you can edit and run. It's great fun and helps you to learn concepts much faster.
- This course is **practical**. While focussing on *getting stuff done in the real world*, I also explain how and why things work instead of teaching you tricks.
- It provides **carefully vetted links** on most pages to deepen your knowledge.
- Did I mention it's **completely free**, with no strings attached? We offer premium Python courses → <https://pythont.land/shop> for those looking for a premium experience, extra practice, and a certificate of completion.

About the instructor

So what makes me eligible to teach you Python? Let me introduce myself!



I'm Erik, and I've worked as a software engineer for over 25 years. I hold an MSc in computer science and used many programming languages in my career, but Python is my absolute favorite! Although I love programming and building complex systems, I also love writing and teaching. Hence, I combined these two by writing tutorials and courses on this site.

Eventually, I got fed up with the limited copy-and-paste code examples and wanted example code that is editable and runnable in-page. It resulted in a side project (crumb.sh → <https://crumb.sh/>) that offers a generic way to do this. The courses have many useful code crumbs sprinkled throughout them to improve the learning experience!

Stay up-to-date

If you're on Twitter, you can [follow me \(@erikyan\) → https://twitter.com/erikyan](#) to get updates on new content.

If you prefer e-mail, try my weekly Python newsletter. You can [subscribe here → https://python.land/newsletter](#) or at the bottom of any page. I use it to:

- Inform you about current Python trends
- New and existing articles you might like
- And I'll include occasional course discount codes too.

Python history

Let's start by defining more precisely what Python is. Python is a computer programming language. Or, in other words, **a vocabulary and set of grammatical rules for instructing a computer to perform tasks**. Its original creator, Guido van Rossum, named it after the BBC television show 'Monty Python's Flying Circus.' Hence, you'll find that Python books, code examples, and documentation sometimes contain references to this television show.

In 1987, Guido worked on a large distributed operating system at the CWI, a national research institute for mathematics and computer science in the Netherlands. Within that project, he had some freedom to work on side projects. With the knowledge and experience he had built up in the years before, working on a computer language called ABC, he started writing the Python programming language.

Python is easy to learn, and it's designed around a set of clearly defined principles ([the Zen of Python → http://python.land/the-zen-of-python](#)) that encourage Python core developers to make a language that is unambiguous and easy to use.

Extensible

In a [2003 interview with Bill Venners → https://www.artima.com/articles/the-making-of-python](#), Guido mentioned what was probably the biggest innovation in the new language:

I think my most innovative contribution to Python's success was making it easy to extend. That also came out of my frustration with ABC. ABC was a very monolithic design. There was a language design team, and they were God. They designed every language detail and there was no way to add to it. You could write your own programs, but you couldn't easily add low-level stuff.

Guido van Rossum

He decided you should be able to extend the language in two ways: by writing pure Python modules or by writing a module entirely in C. The former is obviously easier, but the latter allows for fast, high-performance modules like NumPy. It was a success because his CWI colleagues, the users, and Guido himself immediately started writing their own extension modules. The extension modules let you do all sorts of things. Just a small selection of modules that exist today:

- graphics libraries,
- data processing and [data science → https://python.land/data-science](#) libraries like [NumPy → https://python.land/data-science/numpy](#) and Pandas,
- AI and machine learning libraries,
- libraries to work with all sorts of file formats (like [JSON → https://python.land/data-processing/working-with-json](#), [YAML → https://python.land/data-processing/python-yaml](#)),
- all kinds of libraries to use and connect external services,
- build websites and website backends (e.g., Django, FastAPI)
- ... and so on

Since its inception, Guido has been actively involved in Python's development until this day. After a short retirement, he returned to work. Microsoft employs him, and his main focus is improving Python's speed.

A timeline

The following figure shows a global timeline of Python's historical and most defining releases:

Python 2 vs. Python 3

As you can see from the Python history timeline, Python 2 and 3 have been developed and maintained side by side for an extended period. The primary reason is that Python 3 code is not entirely backward compatible with Python 2 code. This incompatibility caused a prolonged adoption rate. Many people were happy with version 2 and had no reason to upgrade.

On top of that, Python 3 was initially slower than Python 2. As Python 3 kept improving and receiving new features, it eventually took off. With the latest efforts led by Guido, Python 3 is now faster than ever. In addition, Python 3 adds many useful features to the language, making it easier and more fun. Unless you need to maintain a legacy code base, avoid Python 2.

Run Your First Python Program

Let's dive in directly! We'll run the Python code from your browser to get started quickly. Later on, you'll learn all about installing Python on your computer.

You can use the Python Playground to test and experiment with the examples in this Python tutorial. It allows you to enter Python code and run it by pressing the big green play button. The code runs in our backend on a real computer. Try it!

Hello World

There's a tradition in which programming tutorials, books, and courses start with a so-called *Hello World* program. A Hello World program simply prints the words "Hello world" to the screen. The Python playground above does precisely this, using the `print()` function.

The `print()` function takes anything you put between the parentheses and prints it to the screen. But we must feed it the correct type of data for it to work. For example, we need to put text in Python between quotes. In the world of computer programming, we call this a [string → `https://python.land/introduction-to-python/strings`](https://python.land/introduction-to-python/strings).

Quotes around strings are essential because they precisely mark the start and end of a string. This way, a string is easy to recognize for Python. Here are a few more examples of valid strings:

```
'Hello world'  
'My name is Eric'  
'This one is a bit longer.'
```

You can print these strings or a string of your choosing in the Python playground above, and I encourage you to do so! Like most things in life, programming requires practice to master it.

Navigating the free Python tutorial

I did everything I could to make browsing the free Python tutorial as easy as can be! Primarily, you can use the top menu. In addition, navigational links at the top and end of each page guide you to the next topic or return to the previous one. I also link to related pages to deepen your knowledge.

I carefully ordered the course topics so that you can start from the beginning and work your way up. However, feel free to browse around!

How can you help me?

You, yes, that's you, can help me improve this course. There are several things you can do to help.

1. Get in touch if you...

- find any mistake,
- think something can be improved,
- or something is unclear to you.

2. Donate or buy the course

I've been working on this site for about three years, spending most of my spare time here with all my heart and soul. It must have been 1000s of hours by now. I hope it shows, and I genuinely hope you have a lot of fun learning Python here.

Buying my [Python course for beginners](https://python.land/product/python-course) → <https://python.land/product/python-course> is the best way to support my work. If that's not an option and you still want to show your appreciation, you can [buy me a coffee](https://www.buymeacoffee.com/pythonland) → <https://www.buymeacoffee.com/pythonland>. All the support I have gotten so far encourages me to keep writing and updating the content!

4. Follow me

You can [@erikyan](https://twitter.com/erikyan) → <https://twitter.com/erikyan> for updates on new content, links, quizzes, code snippets, etc. This site also has a dedicated mailing list; [subscribe to my newsletter](https://python.land/newsletter) → <https://python.land/newsletter>. It's low volume and mainly contains new articles I wrote, interesting links to read, and the occasional discounts for my premium courses!

Let's go and learn Python!

Ready to learn Python? You can use the navigational buttons at the bottom and top of each page to follow this tutorial in order, or you can browse around by using the site menu on the right (or top of you have a small screen). If you have any questions, don't hesitate to contact me through the [contact form](https://python.land/contact) → <https://python.land/contact>.

Learn Python properly through small, easy-to-digest lessons, progress tracking, quizzes to test your knowledge, and practice sessions. Each course will earn you a downloadable course certificate.

Source: <https://www.codedex.io/python>

Learn Python 2 | Codecademy

Source: <https://www.codecademy.com/learn/learn-python>

Learn the basics of the world's fastest growing and most popular programming language used by software engineers, analysts, data scientists, and machine learning engineers alike.

1,978,588 learners enrolled

-
-
- Certificate of completion
Included with paid plans
-

About this course

Python is a general-purpose, versatile and popular programming language. It's great as a first language because it is concise and easy to read, and it is also a good language to have in any programmer's stack as it can be used for everything from web development to software development and scientific applications.

Syllabus

20 lessons • 9 projects • 9 quizzes

-
-
-
-
-
-
-

The platform

Hands-on learning

Projects in this course

- [Project](#)

Tip Calculator

In this project, we're going to practice syntax in Python so you can hone your skills and feel confident taking them to the real world. Why? You've done a great job so far on your quest to learn Python. Let's build something to solidify your newfound knowledge.

→ <https://www.codecademy.com/courses/learn-python/projects/tip-calculator-project>

- [Project](#)

Python Mad Libs

In this project, we're going to practice inputs and print in Python so you can hone your skills and feel confident taking them to the real world. Why? Being able to take inputs and print results is a key part of programming.

– <https://www.codecademy.com/courses/learn-python/projects/madlibs-1>

- Project

Area Calculator

In this project, we're going to practice functions and conditionals in Python so you can hone your skills and feel confident taking them to the real world. Why? Ever wanted to automate your math homework? Time to build something in Python that does just that.

– <https://www.codecademy.com/courses/learn-python/projects/area-calculator>

Learn Python 2 course ratings and reviews

6,209 ratings

1. 5 stars
2. 4 stars
3. 3 stars
4. 2 stars
5. 1 star

- The progress I have made since starting to use codecademy is immense! I can study for short periods or long periods at my own convenience - mostly late in the evenings.

Chris

Codecademy Learner @ USA

- I felt like I learned months in a week. I love how Codecademy uses learning by practice and gives great challenges to help the learner to understand a new concept and subject.

Rodrigo

Codecademy Learner @ UK

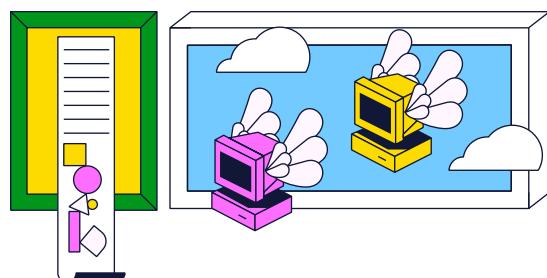
- Brilliant learning experience. Very interactive. Literally a game changer if you're learning on your own.

John-Andrew

Codecademy Learner @ USA

Our learners work at

Join over 50 million learners and start Learn Python 2 today!



Looking for something else?

Related resources

- [Article](#)

What is Python?

What is Python, and what can it do?

- <https://www.codecademy.com/article/what-is-python>
- Article

Programming in Python on a Chromebook

This article will teach you how to run Python code on Chromebooks so you can do off-platform Python projects on your Chromebook.

- <https://www.codecademy.com/article/programming-python-on-chromebook>
- Article

Installing Python 3 and Python Packages

Learn how to install Python packages and download Python 3 with Anaconda and Miniconda on Mac and Windows.

- <https://www.codecademy.com/article/install-python3>

Related courses and paths

- Free course

Python for Programmers

An introduction to the basic syntax and fundamentals of Python for experienced programmers.

- <https://www.codecademy.com/learn/python-for-programmers>
- Free course

Learn Intermediate Java: Input and Output

This course shows how programmers can code a Java program that considers, interprets, and responds to input and output.

- <https://www.codecademy.com/learn/learn-intermediate-java-input-and-output>
- Free course

Learn the Command Line: Redirecting Input and Output

Learn to redirect input and output to and from files and programs.

- <https://www.codecademy.com/learn/learn-the-command-line-redirecting-input-and-output>

Browse more topics

- Python → <https://www.codecademy.com/catalog/language/python> 4,574,713 learners enrolled
- Code Foundations → <https://www.codecademy.com/catalog/subject/code-foundations> 13,283,672 learners enrolled
- Data Science → <https://www.codecademy.com/catalog/subject/data-science> 5,838,183 learners enrolled
- Computer Science → <https://www.codecademy.com/catalog/subject/computer-science> 7,511,133 learners enrolled
- For Business → <https://www.codecademy.com/catalog/subject/for-business> 9,794,824 learners enrolled
- Web Development → <https://www.codecademy.com/catalog/subject/web-development> 6,925,003 learners enrolled
- Cloud Computing → <https://www.codecademy.com/catalog/subject/cloud-computing> 4,140,416 learners enrolled

- [Data Analytics](https://www.codecademy.com/catalog/subject/data-analytics) → https://www.codecademy.com/catalog/subject/data-analytics 3,945,235 learners enrolled
- [IT](https://www.codecademy.com/catalog/subject/information-technology) → https://www.codecademy.com/catalog/subject/information-technology 3,901,496 learners enrolled

[View full catalog](https://www.codecademy.com/catalog) → https://www.codecademy.com/catalog

Two people in conversation while learning to code with Codecademy on their laptops

Unlock additional features with a paid plan

- **Practice Projects**

Guided projects that help you solidify the skills and concepts you're learning.

- **Assessments**

Auto-graded quizzes and immediate feedback help you reinforce your skills as you learn.

- **Certificate of Completion**

Earn a document to prove you've completed a course or path that you can share with your network.

Coding Bootcamps - Ultimate source of learning coding

Source: <https://www.coding-bootcamps.com>

EDUCATION SOLUTION

Hands-on Courses Available for Anyone.

- More than 100 Courses
- 1k students
- 10+ years in business



image

image



image

Blockchain

09 Courses

→ <https://www.coding-bootcamps.com/course-category/blockchain/?tutor-course-filter-category=16>

Crash Bootcamps

10 Courses

→ <https://www.coding-bootcamps.com/course-category/crash-self-learning-bootcamps/?tutor-course-filter-category=24>

Featured

10 Courses

→ <https://www.coding-bootcamps.com/course-category/featured/?tutor-course-filter-category=11>

Hybrid Bootcamps

10 Courses

→ <https://www.coding-bootcamps.com/course-category/hybrid-bootcamps/?tutor-course-filter-category=14>

Learning Paths

10 Courses

→ <https://www.coding-bootcamps.com/course-category/learning-paths/?tutor-course-filter-category=13>

Private Tutoring

10 Courses

→ <https://www.coding-bootcamps.com/course-category/1-to1-tutoring/?tutor-course-filter-category=6>

Self-Learning Bootcamps

04 Courses

→ <https://www.coding-bootcamps.com/course-category/self-learning-bootcamps/?tutor-course-filter-category=37>

Software Engineering

09 Courses

→ <https://www.coding-bootcamps.com/course-category/software-engineering/?tutor-course-filter-category=23>

System Engineering

10 Courses

→ <https://www.coding-bootcamps.com/course-category/system-engineering/?tutor-course-filter-category=20>

Web Development

10 Courses

→ <https://www.coding-bootcamps.com/course-category/web-development/?tutor-course-filter-category=17>

Webinars

04 Courses

→ <https://www.coding-bootcamps.com/course-category/webinars/?tutor-course-filter-category=29>

Course List

Coding and Technology Classes



Self-Learning Bootcamps

Complete bundle for learning software engineering → <https://www.coding-bootcamps.com/courses/complete-bundle-for-learning-software-engineering/>

Free



Self-Learning Bootcamps

Complete training package for system admins → <https://www.coding-bootcamps.com/courses/complete-training-package-for-system-admins/>

Free



Featured

Introduction to Linux programming by hands-on examples → <https://www.coding-bootcamps.com/courses/introduction-to-linux-programming-by-hands-on-examples/>

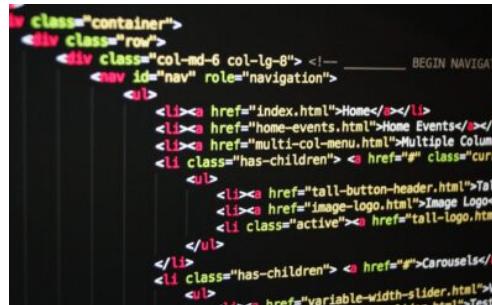
Free



Featured

Essential practical guide for Linux Bash scripting → <https://www.coding-bootcamps.com/courses/essential-practical-guide-for-linux-bash-scripting/>

Free



```
<# class="container">
<# div class="row">
<# div class="col-md-6 col-lg-8"> <!-- BEGIN NAVIGATION -->
<# nav id="nav" role="navigation">
<# ul>
<# li><a href="index.html">Home</a></li>
<# li><a href="home-events.html">Home Events</a></li>
<# li><a href="multi-col-menu.html">Multiple Columns</a>
<# li class="has-children"> <a href="#" class="current">
<# ul>
<# li><a href="tall-button-header.html">Tall Buttons</a>
<# li><a href="image-logo.html">Image Logo</a>
<# li class="active"> <a href="tall-logo.html">Tall Logo</a>
<# ul>
<# li><a href="#">Carousels</a>
<# li class="has-children"> <a href="#">Variable Width Sliders</a>
<# ul>
<# li><a href="variable-width-slider.html">Variable Width Sliders</a>
<# li><a href="test.html">Test</a>
```

Featured

Learn Python programming by hands-on examples → <https://www.coding-bootcamps.com/courses/learn-python-programming-by-hands-on-examples/>

Free



Featured

Intro to Cloud Technology, DevOps, Docker and Kubernetes → <https://www.coding-bootcamps.com/courses/intro-to-cloud-technology-devops-docker-and-kubernetes/>

Free



Featured

Essential practical guide to cybersecurity → <https://www.coding-bootcamps.com/courses/essential-practical-guide-to-cybersecurity/>

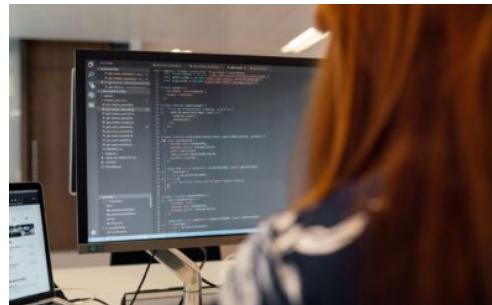
Free



Featured

Private 1-to-1 tutoring for mastering Python & Python OOP → <https://www.coding-bootcamps.com/courses/private-one-to-one-tutoring-for-mastering-python-and-python-object-oriented-programming/>

Free



Featured

Learn C programming by hands-on examples → <https://www.coding-bootcamps.com/courses/learn-c-programming-by-hands-on-examples/>

Free



Featured

Learn Python OOP by Examples → <https://www.coding-bootcamps.com/courses/learn-python-object-oriented-programming-by-examples/>

Free
Pricing Plan

Choose The Best Package For your Learning

FREE



https://themewant.com/products/wordpress/edurock/wp-content/uploads/2023/07/price__1.png

\$ 0/ month

Perfect for startup

- Intro to cybersecurity
- Cloud technology
- Bootstrap web design
- UX web design
- SEO
- Intro to IT roadmap

Get Started → <https://learn.coding-bootcamps.com/p/free-online-courses>

No credit card required

100

BASIC



https://themewant.com/products/wordpress/edurock/wp-content/uploads/2023/07/price__2.png

\$ 390/ month

Perfect for developers

- 10 self-paced classes
- Popular coding topics
- Hands-on projects
- Expert instructors
- Official certification
- Hands-on course outlines

Get Started → <https://www.coding-bootcamps.com/course-category/self-learning-bootcamps/>

Credit card required

PRO



https://themewant.com/products/wordpress/edurock/wp-content/uploads/2023/07/price__3.png

\$ 990/ month

Perfect for engineers

- 50 self-paced courses
- Popular coding topics
- Hands-on projects
- Expert instructors
- Official certification
- Hands-on course outlines

Get Started → <https://learn.coding-bootcamps.com/p/coding-bootcamps-membership>

Credit card required

News Blogs

Latest Tutorials & Blog

Contents covering web development, software engineering, blockchain development and system admin.



[Blockchain](https://www.coding-bootcamps.com/category/blockchain/) → <https://www.coding-bootcamps.com/category/blockchain/>

Integration of Ethereum with AI, ML and IoT → <https://www.coding-bootcamps.com/integration-of-ethereum-with-ai-ml-and-iot/>

Read More → <https://www.coding-bootcamps.com/integration-of-ethereum-with-ai-ml-and-iot/>



[Blockchain](https://www.coding-bootcamps.com/category/blockchain/) → <https://www.coding-bootcamps.com/category/blockchain/>

Ethereum and Know-Your-Customer Requirements → <https://www.coding-bootcamps.com/ethereum-and-know-your-customer-requirements/>

Read More → <https://www.coding-bootcamps.com/ethereum-and-know-your-customer-requirements/>



[Blockchain](https://www.coding-bootcamps.com/category/blockchain/) → <https://www.coding-bootcamps.com/category/blockchain/>

Review of 100 DApps that Run on Ethereum Network → <https://www.coding-bootcamps.com/review-of-100-dapps-that-run-on-ethereum-network/>

Read More → <https://www.coding-bootcamps.com/review-of-100-dapps-that-run-on-ethereum-network/>



Blockchain → <https://www.coding-bootcamps.com/category/blockchain/>

Review of Ethereum Tools and Frameworks → <https://www.coding-bootcamps.com/review-of-ethereum-tools-and-frameworks/>

Read More → <https://www.coding-bootcamps.com/review-of-ethereum-tools-and-frameworks/>



Blockchain → <https://www.coding-bootcamps.com/category/blockchain/>

Review of Ethereum Tools and Infrastructure → <https://www.coding-bootcamps.com/review-of-ethereum-tools-and-infrastructure/>

Read More → <https://www.coding-bootcamps.com/review-of-ethereum-tools-and-infrastructure/>



Blockchain → <https://www.coding-bootcamps.com/category/blockchain/>

Review of Mainstream Adoption for Ethereum Blockchain Development → <https://www.coding-bootcamps.com/mainstream-adoption-for-ethereum-blockchain-development/>

Read More → <https://www.coding-bootcamps.com/mainstream-adoption-for-ethereum-blockchain-development/>



Blockchain → <https://www.coding-bootcamps.com/category/blockchain/>

Review of Government Regulations for Ethereum Blockchain Development → <https://www.coding-bootcamps.com/government-regulations-for-ethereum-blockchain-development/>

Read More → <https://www.coding-bootcamps.com/government-regulations-for-ethereum-blockchain-development/>



Blockchain → <https://www.coding-bootcamps.com/category/blockchain/>

Review of Ethereum Governance for Ethereum Blockchain Development → <https://www.coding-bootcamps.com/ethereum-governance-for-blockchain-development/>

Read More → <https://www.coding-bootcamps.com/ethereum-governance-for-blockchain-development/>



Blockchain → <https://www.coding-bootcamps.com/category/blockchain/>

UI/UX and Design Thinking for Ethereum Blockchain Development → <https://www.coding-bootcamps.com/review-of-ui-and-ux-for-ethereum-blockchain-development/>

Read More → <https://www.coding-bootcamps.com/review-of-ui-and-ux-for-ethereum-blockchain-development/>



Blockchain → <https://www.coding-bootcamps.com/category/blockchain/>

Review of Challenges for Ethereum Blockchain Development → <https://www.coding-bootcamps.com/review-of-challenges-for-ethereum-blockchain-development/>

Read More → <https://www.coding-bootcamps.com/review-of-challenges-for-ethereum-blockchain-development/>



Blockchain → <https://www.coding-bootcamps.com/category/blockchain/>

Review of Ethereum Transaction Pool Options → <https://www.coding-bootcamps.com/review-of-ethereum-transaction-pool-options/>

Read More → <https://www.coding-bootcamps.com/review-of-ethereum-transaction-pool-options/>



Blockchain → <https://www.coding-bootcamps.com/category/blockchain/>

Review of Ethereum Networking Options → <https://www.coding-bootcamps.com/review-of-ethereum-networking-options/>

Read More → <https://www.coding-bootcamps.com/review-of-ethereum-networking-options/>

Learn R, Python & Data Science Online

Source: <https://www.datacamp.com>

Python R Sql ChatGPT Power BI Tableau Excel Docker DataBricks Snowflake Azure Git

Grow your data skills with DataCamp for Mobile

Make progress on the go with our mobile courses and daily 5-minute coding challenges.
Loved by learners at thousands of companies

Skill up at scale. Data and AI training designed for your business.

Join 2,500+ companies and 80% of the Fortune 1000 who use DataCamp to upskill their teams.

[Learn More → https://www.datacamp.com/business](https://www.datacamp.com/business)

What is DataCamp?

Learn the data and AI skills you need online at your own pace—from non-coding essentials to data science, AI, and machine learning.

[Start Learning for Free → https://www.datacamp.com/users/sign_up](https://www.datacamp.com/users/sign_up)

Hands-on learning experience

No installation required — run code from your browser



Screenshot of Campus exercise

Learn from the best instructors

[Become an Instructor → https://www.datacamp.com/create](https://www.datacamp.com/create)



Screenshot of Campus exercise

Interactive exercises short videos

```
script.py
1 # Previous customizations
2 plt.xscale('log')
3 plt.xlabel('GDP per Capita [in USD]')
4 plt.ylabel('Life Expectancy [in years]')
5 plt.title('World Development in 2007')
6 plt.xticks([1000,10000,100000],
7 ['1k','10k','100k'])
```

Run Code Submit Answer

Screenshot of campus exercise with photo of Richie Cotton

Practice and apply your skills

Complete the code to return the output

```
def square(x):
    y= ?
    return(y)
print(square(2))
```

4 $x \times x$ 3
x+1 1 x*4 2 x/3 4

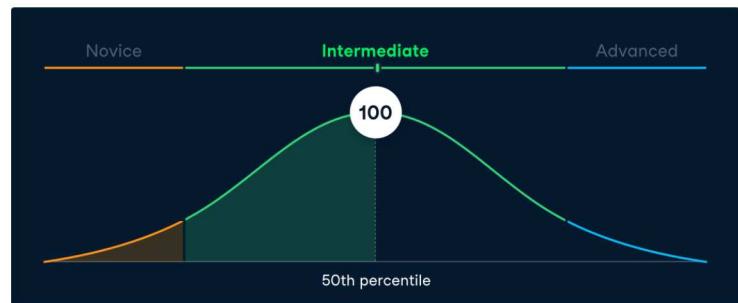
Screenshot of campus exercise and dragging answer



DataCamp Signal™

Discover your data skill level for free

Effective learning starts with assessment. Learning a new skill is hard work—Signal makes it easier.

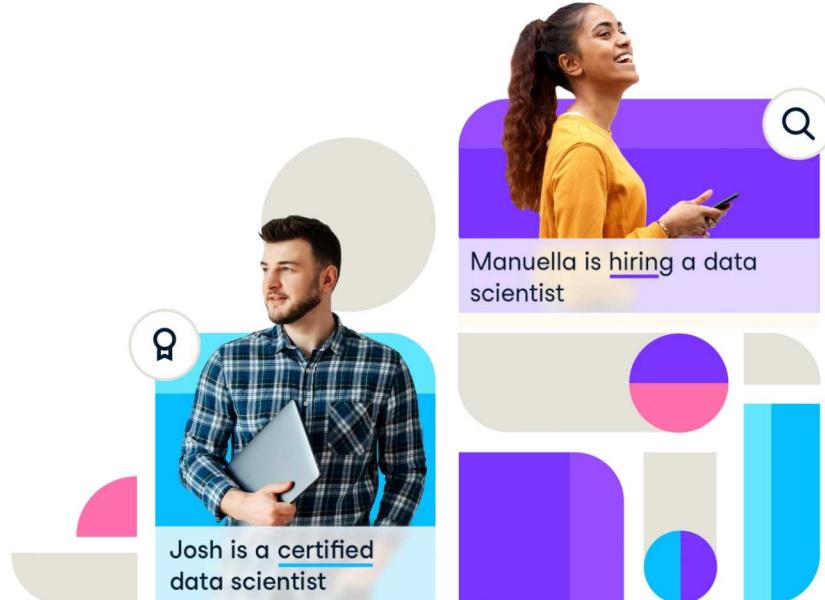


You received a score of 100. You performed better than 50% of your peers.

Screenshot of an assessment with a score of 100
certification

Land your dream job in data science

From a certification in data science to personalized resume reviews and interview prep—we've got you covered.
[Learn More → https://www.datacamp.com/certification](https://www.datacamp.com/certification)



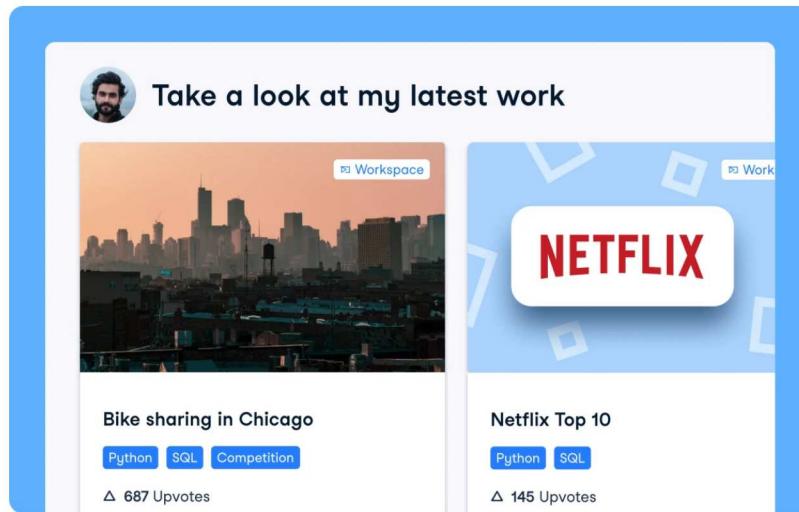
Screenshot of data scientist and hiring manager

Workspace

Build your data portfolio

Start from 60+ datasets and templates to create insightful analyses with DataCamp's AI-enabled data notebook.

[Learn More → https://www.datacamp.com/workspace](https://www.datacamp.com/workspace)



Workspaces in a user's portfolio

Don't just take our word for it.

Are you an educator?

DataCamp for Classrooms is always free for you and your students.

[Learn More → https://www.datacamp.com/universities](https://www.datacamp.com/universities)

- 350,000+ students served
- 3,700+ classrooms using DataCamp in 2020
- 180+ countries represented
- 6 months of free access for every classroom

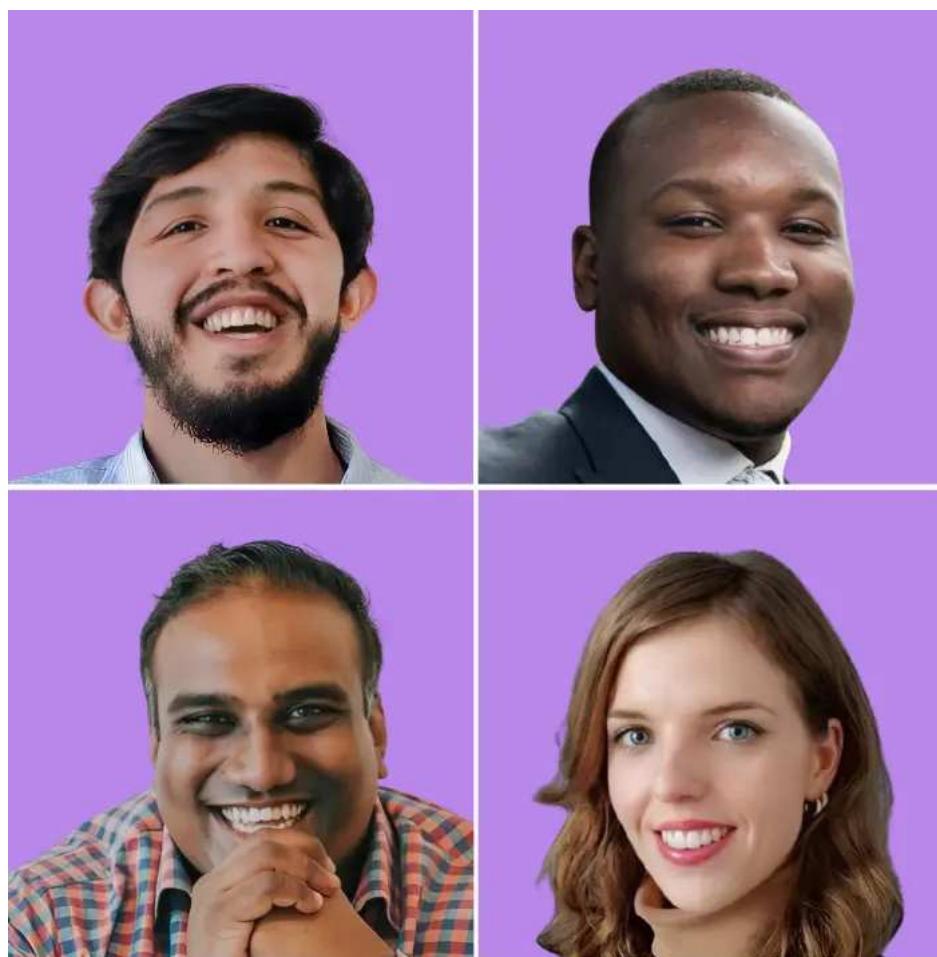
Join more than 13 million learners worldwide

Dataquest: The fastest way to learn AI and data skills online

By Dataquest makes your hardwork count.

Source: <https://www.dataquest.io>

Learn 10x faster and achieve your goals with Dataquest. Our guided paths, progress tracking, and AI-assisted learning will help you master new skills quickly and effectively. Find out today why Dataquest graduates say it is the best way to learn AI and data skills online.



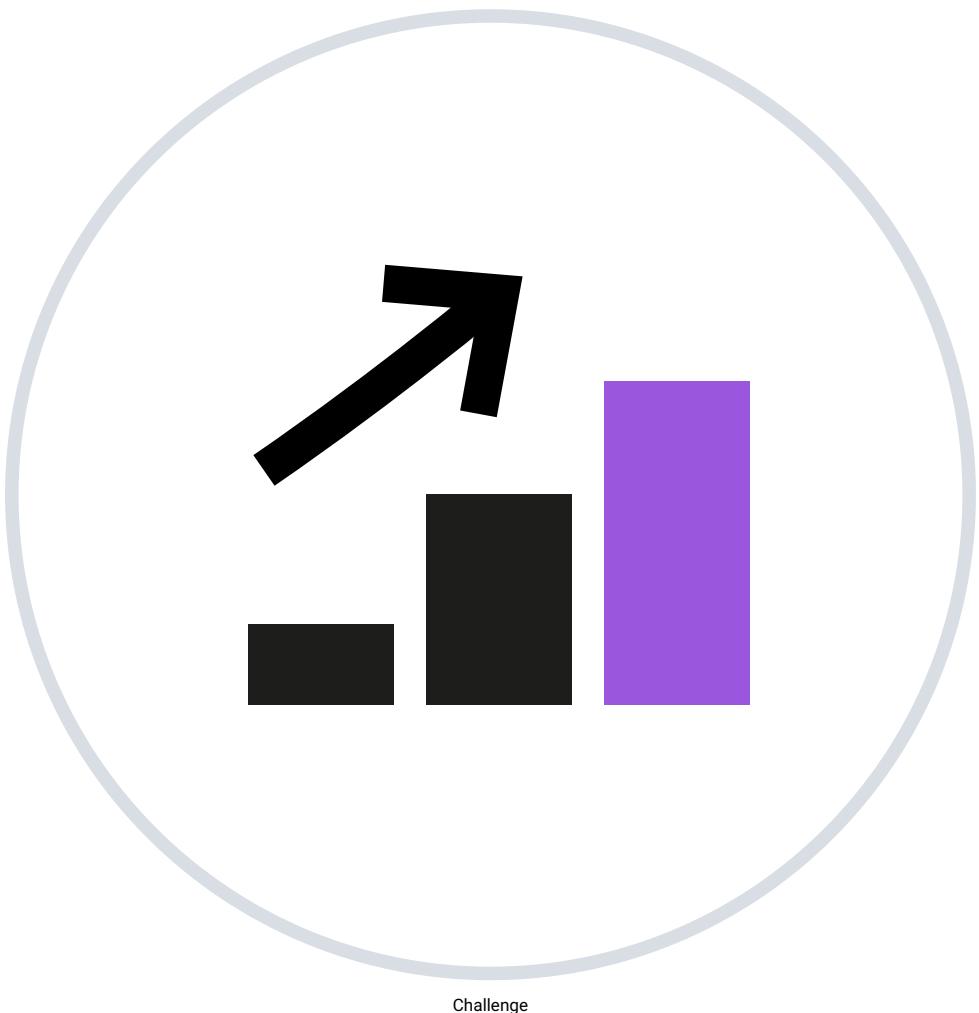
Learners

**Become an AI and data expert faster
with Dataquest.**



Learn efficiently

Learn exactly what you need to achieve your goal — and nothing extra.



Challenge

Challenge yourself with exercises

Work with real data from day one with hands-on exercises.



Build

Build your project portfolio

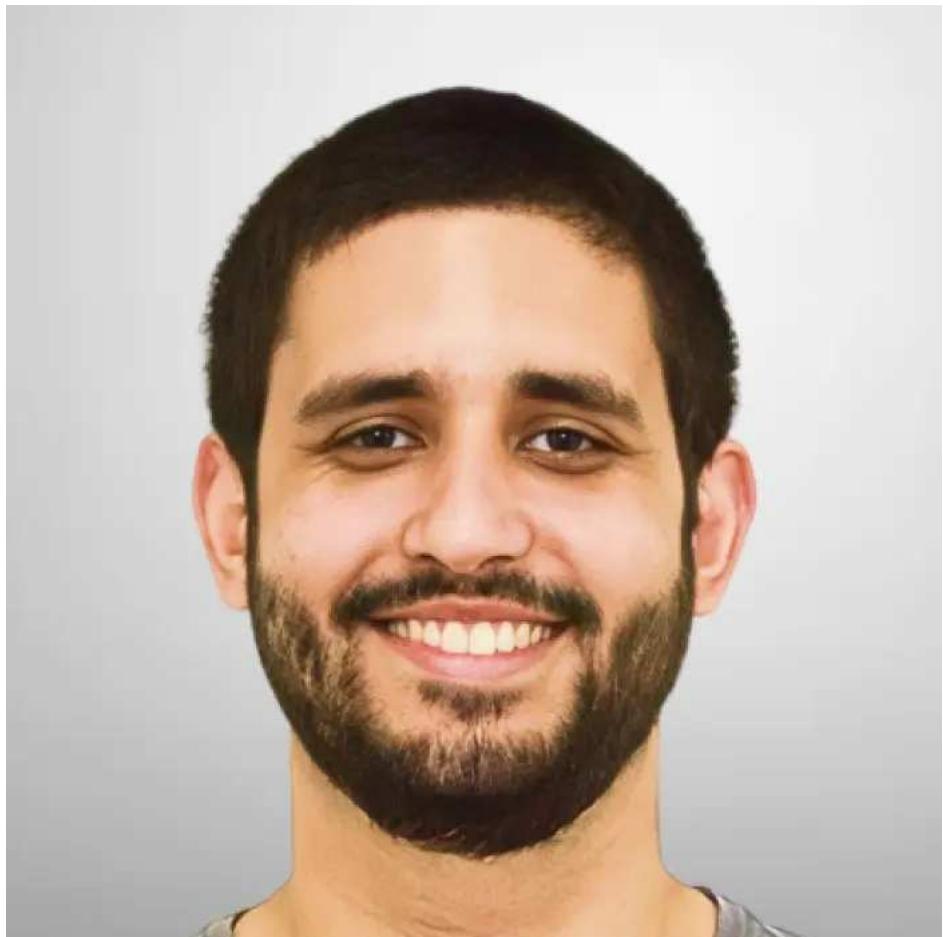
Gain confidence and show off your data skills with projects.
Nice work! [Sign up → https://app.dataquest.io/signup](https://app.dataquest.io/signup) to keep learning.

The screenshot shows a Jupyter Notebook interface. On the left, there are two tabs: "script.py" and "admissions.csv". The "script.py" tab is active, displaying the following Python code:

```
1 logistic_model = LogisticRegression()
2 logistic_model.fit(admissions[["gpa"]], admissions["admit"])
3 pred_probs =
4 logistic_model.predict_proba(admissions[["gpa"]])
5 plt.scatter(admissions["gpa"], pred_probs[:,1])
6
7
8
9
10
11
```

On the right, there is a "Run code" button and a "Next lesson" button. Below the code editor, there is a "Nice Work!" message and a "Script" link.

Script



Otávio

Otávio Silveira

"The learning paths on Dataquest are incredible. You don't have to guess what you should learn next."



Francisco

Francisco Sosa

"Dataquest finds a balance between being too easy and being so hard that you get discouraged."



Viktoria

Viktoria Jorayeva

"I can't believe how easily and clearly complex material is presented on Dataquest."



Sunishchal

Sunishchal Dev

"The lessons are easy to absorb. Dataquest helped me get my dream job!"



Stacey

Stacey Ustian

"Dataquest teaches you how you do data analysis in the real world."



Rahila

Rahila Hashim

"Dataquest is my favorite learning platform."

Build a team of AI and data experts.

Give your employees and students the AI and data skills they need to excel. Learn how to use AI, Python, R, SQL, Excel, PowerBI, Tableau, and other tools in the real world.

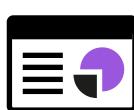
Learn with the Dataquest method.

Follow a proven path to achieve your goal.



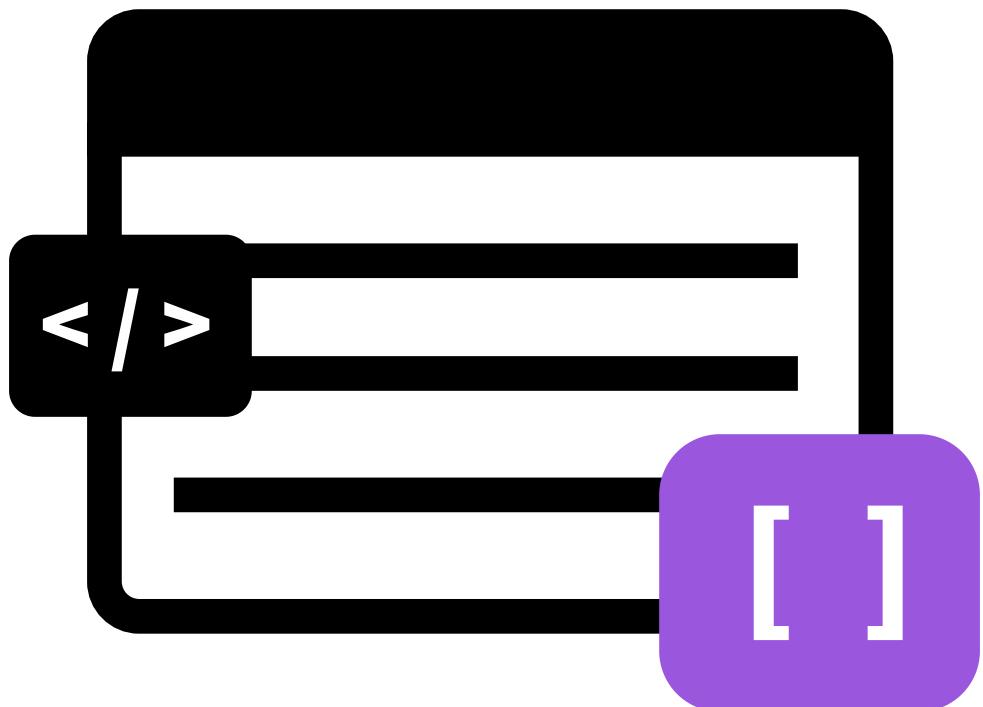
Icon

Learn faster with intuitive text explanations and diagrams.



Icon

Complete interactive exercises with real datasets.



Icon

Build real-world projects that get you job-ready.



Icon

Browse our most popular paths and courses.

Start a new career

Career paths get you job-ready in 3 to 9 months, and grant you a certificate.

27 Courses

Data Analyst in Python

New
23 Courses

Data Analyst in R

15 Courses

Business Analyst with Power BI

14 Courses

Business Analyst in Tableau

Learn a skill

Skill paths help you fully master a data topic.

5 Courses

SQL Fundamentals

7 Courses

Machine Learning in Python

12 Courses

Probability and Statistics

5 Courses

Data Analysis in Excel

1 Course

APIs and Web Scraping with Python

8 Courses

Generative AI Fundamentals

New

Take a course

Learn a new concept and grow your technical skills.

Python

Command Line

Git and Github

Introduction to Pandas and NumPy for Data Analysis

Hypothesis Testing in R

Optimizing Machine Learning Models in Python

Build your portfolio with projects

**Grow your career with
Dataquest.**

97%
of learners recommend
Dataquest for career advancement
4.9
Dataquest rating on
G2Crowd and SwitchUp
\$30k
Average salary boost
for learners who complete a path



Aaron

Aaron Melton

Business Analyst at Aditi Consulting

“Dataquest starts at the most basic level, so a beginner can understand the concepts. I tried learning to code before, using Codecademy and Coursera. I struggled because I had no background in coding, and I was spending a lot of time Googling. Dataquest helped me actually learn.”



Jessi

Jessica Ko

Machine Learning Engineer at Twitter

“I liked the interactive environment on Dataquest. The material was clear and well organized. I spent more time practicing than watching videos and it made me want to keep learning.”



Victoria

Victoria E. Guzik

Associate Data Scientist at Callisto Media

“I really love learning on Dataquest. I looked into a couple of other options and I found that they were much too handhold-y and fill in the blank relative to Dataquest’s method. The projects on Dataquest were key to getting my job. I doubled my income!”

Join 1M+ data learners on Dataquest.

1

Sign up for a free account

2

Choose a course or path

3

Learn with hands-on exercises

Start learning with a free account today.

HackInScience – Python Exercises

Source: <https://www.hackinscience.org>

Hackinscience is an interactive Python exercise platform:

Write code, get instant feedback from our correction bot

with detailed information about what went wrong so you'll never get stuck.

[Sign in](https://www.hackinscience.org/accounts/login/) → <https://www.hackinscience.org/accounts/login/> [Sign up](https://www.hackinscience.org/accounts/signup/) → <https://www.hackinscience.org/accounts/signup/>

↳

High School Technology Services in Washington DC

By <http://www.dcwebmakers.com/>

Source: <https://www.myhsts.org>

High School Technology Services

Coding and Technology Training for Students and Adults

Guaranteed To Run Classes

Here is the list of our confirmed virtual and in-person training classes.

Our Classes

We offer 30+ coding and technology [classes → https://www.myhsts.org/courses.html](https://www.myhsts.org/courses.html) for teenagers and adults.

Best training is finding that perfect balance between learning concepts and doing hands-on projects.

[See list of our classes → https://www.myhsts.org/courses.html](https://www.myhsts.org/courses.html)

"The Adwords workshop was great!"

The Adwords workshop was great! The presentation was thorough and well put together. The instructor (Youssef) is very knowledgeable about the subject and patient. I got a lot of value out of it. Because it was a small class I got a lot of 1on1 time with Youssef which made it even better. Great value! I would definitely recommended this workshop to anyone needing to get to know more about Adwords.

Jubee Vilceus from DC

Google AdWords and Google Analytics Workshop

"I learned a lot during this one-day training!"

I learned a lot during this one-day training. The Instructor has the knowledge and skill to get the job done.

Fahad Hashmi

Object-oriented programming training workshop

"The SQL workshop covered essential information!"

The SQL workshop covered essential information in building a foundation to what data structures are and the creating process. Along with coding it gave comfort by providing alternative ways to build a database with or without command line. Supportive group and overall a good learning experience.

Katherine Cruz from DC

SQL programming and database management training workshop
"The introduction to JavaScript- 30 hour course!"

I learned a lot from this course. The projects were very practical and hands-on. I highly recommend this course to those interested in learning web design.

Gabi Payano from DC

JavaScript programming and web design training course
"The C class was good. I learned a lot!"

The C class was good. I learned a lot and this definitely got me started on my journey. Plus I left the class with resources to continue my study. Juan is a great teacher and seemed like he really cared about my future and where skills in computer programming might take me. It was good to feel that.

Samuel Moore from MD

C programming course

Subscribe to our newsletter!

Contact Us

Attend our community events!

We organize community, free workshops, and networking events online

Submit the below form if you have specific questions regarding our classes or services

[View our classes!](https://www.myhsts.org/courses.html) → <https://www.myhsts.org/courses.html>

19 Keywords Every Coder Must Know – Be on the Right Side of Change

By Chris

Source: <https://blog.finxter.com/python-cheat-sheet/>

Hey friend!

If you're a Python beginner, you're probably overwhelmed with all the language features, libraries, and syntax elements.

Learning a new programming language can be painful because computers are unforgiving—they keep complaining until you can't take it anymore.

To help you overcome the "*valley of desperation*", I created a series of Python cheat sheets — you have found the first one on this page! 

Python Keywords Cheat Sheet

This cheat sheet is for beginners in the Python programming language. It explains everything you need to know about **Python keywords**.

Download and pin it to your wall until you feel confident using all these keywords! 

Python Cheat Sheet: Keywords

"A puzzle a day to learn, code, and play" → Visit finxter.com

Keyword	Description	Code example
<code>False, True</code>	Data values from the data type Boolean	<code>False == (1 > 2), True == (2 > 1)</code>
<code>and, or, not</code>	Logical operators: $(x \text{ and } y) \rightarrow$ both x and y must be True $(x \text{ or } y) \rightarrow$ either x or y must be True $(\text{not } x) \rightarrow$ x must be false	<code>x, y = True, False (x and y) == True # True (x or y) == False # True (not x) == True # True</code>
<code>break</code>	Ends loop prematurely	<code>while(True): break # no infinite loop print("hello world")</code>
<code>continue</code>	Finishes current loop iteration	<code>while(True): continue print("43") # dead code</code>
<code>class</code>	Defines a new class → a real-world concept (object oriented programming)	
<code>def</code>	Defines a new function or class method. For latter, first parameter ("self") points to the class object. When calling class method, first parameter is implicit.	<code>class Beer: def __init__(self): self.content = 1.0 def drink(self): self.content = 0.0 becks = Beer() # constructor - create class becks.drink() # beer empty: b.content == 0</code>
<code>if, elif, else</code>	Conditional program execution: program starts with "if" branch, tries the "elif" branches, and finishes with "else" branch (until one branch evaluates to True).	<code>x = int(input("your value: ")) if x > 3: print("Big") elif x == 3: print("Medium") else: print("Small")</code>
<code>for, while</code>	# For loop declaration <code>for i in [0,1,2]: print(i)</code>	# While loop - same semantics <code>j = 0 while j < 3: print(j) j = j + 1</code>
<code>in</code>	Checks whether element is in sequence	<code>42 in [2, 39, 42] # True</code>
<code>is</code>	Checks whether both elements point to the same object	<code>y = x = 3 x is y # True [3] is [3] # False</code>
<code>None</code>	Empty value constant	<code>def f(): x = 2 f() is None # True</code>
<code>lambda</code>	Function with no name (anonymous function)	<code>(lambda x: x + 3)(3) # returns 6</code>
<code>return</code>	Terminates execution of the function and passes the flow of execution to the caller. An optional value after the return keyword specifies the function result.	<code>def incrementor(x): return x + 1 incrementor(4) # returns 5</code>



Python Cheat Sheet Keywords

Download only this PDF → <https://blog.finxter.com/wp-content/uploads/2019/02/CheatSheet-Python-1.-Keywords1.pdf>

Click on the image or the button. You'll join my [free email academy](https://blog.finxter.com/subscribe/) → <https://blog.finxter.com/subscribe/> and I'll send you 5 additional cheat sheets about data structures, functions, tricks, interview tips, and object orientation.

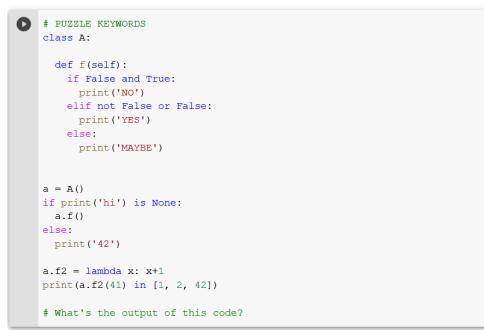
Did I already mention that I love cheat sheets? I'll also send you a regular Python training email course for continuous improvement in Python (it's free).

Alternatively, you can also check out [this direct PDF link](https://blog.finxter.com/wp-content/uploads/2019/02/CheatSheet-Python-1-Keywords1.pdf) → <https://blog.finxter.com/wp-content/uploads/2019/02/CheatSheet-Python-1-Keywords1.pdf> to download the PDF right away—and [subscribe here](https://blog.finxter.com/subscribe/) → <https://blog.finxter.com/subscribe/> for the remaining educational content!

Over time, this page has turned into a full-fledged Python tutorial with many additional resources, puzzles, tips, and videos. Go ahead—have some fun & try to learn a thing or two & become a better coder in the process!

Interactive Python Puzzle

I've written a short puzzle that incorporates all keywords discussed in the cheat sheet. Can you solve it?



```
# PUZZLE KEYWORDS
class A:

    def f(self):
        if False and True:
            print('NO')
        elif not False or False:
            print('YES')
        else:
            print('MAYBE')

    a = A()
    if print('hi') is None:
        a.f()
    else:
        print('42')

a.f2 = lambda x: x+1
print(a.f2(41) in [1, 2, 42])

# What's the output of this code?
```

Exercise: Think about this puzzle and guess your output. Then, run the code and check whether you were right!

Did you struggle with the puzzle? No problem — Let's dive into all of these keywords to gain a better understanding of each.

Python Keywords

Learn 80% of the keywords in 20% of the time: these are the most important Python keywords.

[Python Keywords \[A Helpful 16-Minute Primer\]](#)

False, True

Data values from the Boolean data type

```
False == (1 > 2)
True == (2 > 1)
```

and, or, not

Logical operators:

- $(x \text{ and } y) \rightarrow$ both x and y must be True for the expression to be True
- $(x \text{ or } y) \rightarrow$ either x or y or both must be True for the expression to be True
- $(\text{not } x) \rightarrow$ x must be False for the expression to be True

```
x, y = True, False
(x or y) == True      # True
(x and y) == False    # True
(not y) == True       # True
```

break

Ends loop prematurely

```
while(True):
    break # no infinite loop
print("hello world")
```

continue

Finishes current loop iteration

```
while(True):
    continue
    print("43") # dead code
```

class

Defines a new class → a real-world concept

[\(object-oriented programming\) → https://blog.finxter.com/object-oriented-programming-terminology-cheat-sheet/](https://blog.finxter.com/object-oriented-programming-terminology-cheat-sheet/)

```
class Beer:
    def __init__(self):
        self.content = 1.0
    def drink(self):
        self.content = 0.0

becks = Beer() # constructor - create class
becks.drink() # beer empty: b.content == 0
```

def

Defines a new function or class method. For the latter, the first parameter (“[self](#) → <https://blog.finxter.com/self-in-python/>”) points to the class object. When calling the class method, the first parameter is implicit.

See previous code example.

if, elif, else

Conditional program execution: program starts with “if” branch, tries the “elif” branches, and finishes with “else” branch (until one branch evaluates to True).

```
x = int(input("your value: "))
if x > 3:
    print("Big")
elif x == 3:
    print("Medium")
else:
    print("Small")
```

for, while

Repeated execution of loop body.

```
# For loop declaration
for i in [0,1,2]:
    print(i)

# While loop - same semantics
j = 0
while j < 3:
    print(j)
    j = j + 1
```

in

Checks whether element is in sequence ([membership](#) → <https://blog.finxter.com/python-membership-in-operator/>):

```
42 in [2, 39, 42]
# True
```

is

Checks whether both elements point to the same object ([object](#) → <https://blog.finxter.com/python-is-operator/i> → <https://blog.finxter.com/python-is-operator/d> → <https://blog.finxter.com/python-is-operator/e> → <https://blog.finxter.com/python-is-operator/n> → <https://blog.finxter.com/python-is-operator/t> → <https://blog.finxter.com/python-is-operator/i> → <https://blog.finxter.com/python-is-operator/t> → <https://blog.finxter.com/python-is-operator/y> → <https://blog.finxter.com/python-is-operator/>)

```
y = x = 3
x is y # True
[3] is [3] # False
```

None

Empty value constant

```
def f():
    x = 2
f() is None # True
```

lambda

Function with no name ([anonymous function](https://blog.finxter.com/a-simple-introduction-of-the-lambda-function-in-python/) → <https://blog.finxter.com/a-simple-introduction-of-the-lambda-function-in-python/>)

```
(lambda x: x + 3)(3) # returns 6
```

return

Terminates execution of the function and passes the flow of execution to the caller. An optional value after the return keyword specifies the function result.

```
def incrementor(x):
    return x + 1
incrementor(4) # returns 5
```

Put yourself on the road to mastery and download your free Python cheat sheets now, print them, and post them to your office wall!

Want more cheat sheets and free Python education? [Register for the free Finxter email academy](#) → <https://blog.finxter.com/subscribe/>. In the following, I'll present you a compilation of the best Python cheat sheets on the web. So, keep reading!

Best Python Cheat Sheets

But these are not all—the following Python cheat sheets will greatly improve your learning efficiency! Check out this compilation of the best Python cheat sheets!

5 Python Cheat Sheets Every Python Coder Must Own

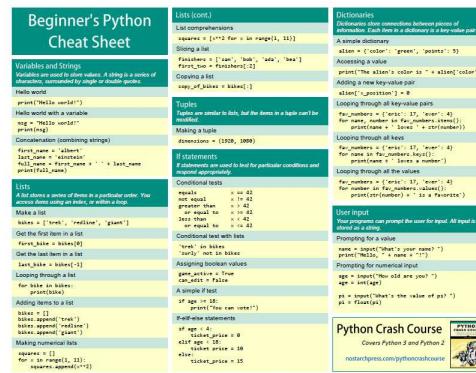
So let's dive into the best Python cheat sheets recommended by us.

Python 3 Cheat Sheet → https://perso.limsi.fr/pointal/_media/python:cours:mementopython3-english.pdf

Python 3 Cheat Sheet	
<pre>Lets you define your own types from typing import List, Tuple, Dict, Union, TypeVar, Type, overload # class Counter: # def __init__(self, start=0): # self._count = start # # def increment(self, n=1): # self._count += n # # def count(self): # return self._count</pre>	Custom Types
<pre># ordered list List[T] = List[TypeVar('T')] # tuple Tuple[T, U] = Tuple[TypeVar('T'), TypeVar('U')] # Non-modifiable tuple (immutable) FrozenList[T] = Tuple[TypeVar('T'), ...]</pre>	Container Types
<pre># key constants, as in prior code, but key access, so no __getitem__ Dict['K', 'V'] = Dict[TypeVar('K'), TypeVar('V')] # dict with type annotations dict[T, U] = Dict[TypeVar('T'), TypeVar('U')] # set with type annotations set[T] = Set[TypeVar('T')]</pre>	Set Types
<pre># keyable hashable type (has __hash__ method) keyable[T] = Hashable[TypeVar('T')]</pre>	Frozenset Imutable Set
Conversion	
<pre>int('123') = 123 float('1.23') = 1.23 float('1.23e-10') = 1.23e-10 round(15.5) = 15 round(15.5, 1) = 15.5 round(15.5, 2) = 15.50 str(123) = '123' str(123.45) = '123.45' str('123') = '123' str(123, 2) = '123.00' str(123.45, 2) = '123.45' hex(123) = '0x7b' hex(123.45) = '0x7b.647d89' oct(123) = '0o173' oct(123.45) = '0o173.357551'</pre>	
Variables Assignment	
<pre># assignment of variables x = y # x is now a reference to y y = 123 # y is now 123 x = y = 123 # both x and y are 123 a, b = 1, 2 # a=1, b=2 a = b = 1 # a=1, b=1 a, *b = [1, 2, 3] # a=1, b=[2, 3] a, *b = 1, 2, 3 # a=1, b=[2, 3] x = 1 x += 2 # x = 3 x -= 2 # x = 1 x *= 2 # x = 2 x /= 2 # x = 1 del x # remove name x</pre>	
For loops, strings, bytes, etc.	
<pre>for i in range(10): print(i) for i in range(10): print(i) else: print('Done') for item in [1, 2, 3]: print(item) for item in [1, 2, 3]: print(item) break for item in [1, 2, 3]: if item == 2: continue print(item)</pre>	
<pre>for file_name in os.listdir('.'): print(file_name) for line in open('file.txt'): print(line) for line in open('file.txt'): print(line) if line == 'quit': break</pre>	
<pre>for line, strng, byte in zip('abc', '123', b'123'): print(line, strng, byte) for item in reversed([1, 2, 3]): print(item) for item in reversed([1, 2, 3]): print(item) break for item in reversed([1, 2, 3]): if item == 2: continue print(item)</pre>	
Sequence Containers Index	
<pre>negative index: -1 = -1 positive index: 0 = 0 len([1, 2, 3]) = 3 len([1, 2, 3, 4, 5]) = 5 positive slice: [0:3] = [1, 2, 3] negative slice: [-3:-1] = [-2, -1] slice with step: [::2] = [1, 3, 5] slice with step: [1::2] = [2, 4] slice with step: [-1:-3:-1] = [-2, -3]</pre>	
Access to sub-sequences (list, dict, slice, str, bytes)	
<pre>list[-1:-10] = [] list[1:-10] = [] list[1:-10, 20, 30, 40] = [] list[1:-10, 20, 30, 40, 50] = [] list[1:-10, 20, 30, 40, 50, 60] = [] list[1:-10, 20, 30, 40, 50, 60, 70] = [] Missing slice index - from start to end On mutable sequences (list, dict, slice) with del: del list[1:-10] and del list[1:-10, 20, 30]</pre>	
Boolean Logic	
<pre>Comparisons: < > <= >= == != a and b # logical and, returns first a or b # logical or, returns either one not a # not a, returns True if a is False and and or and not: a and b is a or b and not a - ensures that a and b are booleans - a logical and True = True False = False</pre>	
Statements Blocks	
<pre>parent block: statement block 1... statement block 2... statement block 3... next statement after block 1 condition block 1: condition block 2: condition block 3: condition of an indent stat. condition of an indent stat.</pre>	
Methods	
<pre>str.format(*args, **kwargs) float('123.45') = 123.45 complex('1+2j') = (1+2j) max([1, 2, 3]) = 3 min([1, 2, 3]) = 1 abs(-123) = 123 round(12.345, 2) = 12.35 round(12.345, 2, 1) = 12.35 ord('A') = 65 ord('a') = 97 chr(65) = 'A' chr(97) = 'a'</pre>	
Module Variables Import	
<pre>import module from module import * from module import variable, function, class from module import variable, function as f, class as C from module import * as alias</pre>	
Conditional Statements	
<pre>if condition: statements block if condition: statements block... else: statements block... else: statements block</pre>	
Logical Conditions	
<pre>if condition1 and condition2: statements block if condition3: statements block... else: statements block... else: statements block</pre>	
Exception Handling	
<pre>try: code block except exception: statements block else: statements block finally: statements block</pre>	

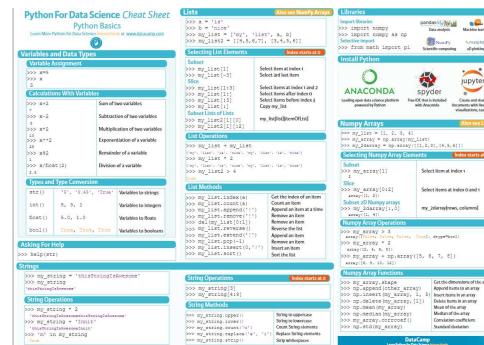
This is the best single cheat sheet. It uses every inch of the page to deliver value and covers everything you need to know to go from beginner to intermediate. Topics covered include container types, conversions, modules, [math](https://blog.finxter.com/python-math-module/) → <https://blog.finxter.com/python-math-module/>, conditionals, and formatting to name a few. A highly recommended 2-page sheet!

Python Beginner Cheat Sheet → https://github.com/ehmatthes/pcc/releases/download/v1.0.0/beginners_python_cheat_sheet_pcc_all.pdf



Some might think this cheat sheet is a bit lengthy. At 26 pages, it is the most comprehensive cheat sheets out there. It explains variables, data structures, exceptions, and classes – to name just a few. If you want the most thorough cheat sheet, pick this one.

Python for Data Science → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PythonForDataScience.pdf

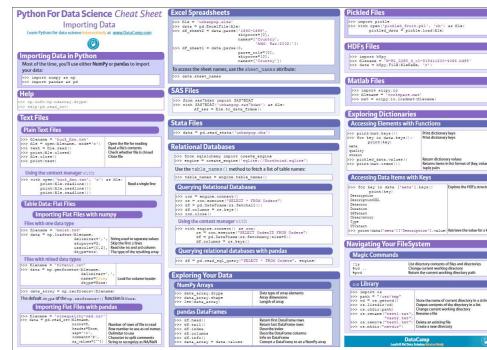


Some of the most popular things to do with Python are **Data Science and Machine Learning** → <https://blog.finxter.com/artificial-intelligence-machine-learning-deep-learning-and-data-science-whats-the-difference/>.

In this cheat sheet, you will learn the basics of Python and the most important scientific library: **NumPy** → <https://blog.finxter.com/numpy-tutorial/> (Numerical Python). You'll learn the basics plus the most important NumPy functions.

If you are using Python for Data Science, download this cheat sheet.

Python for Data Science (Importing Data) → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/CheatSheets/Importing_Data_Python_Cheat_Sheet.pdf



This Python data science cheat sheet from DataCamp is all about getting data into your code.

Think about it: importing data is one of the most important tasks when working with data. Increasing your skills in this area will make you a better data scientist—and a better coder overall!

Python Cheatography Cheat Sheet → <https://www.cheatography.com/davechild/cheat-sheets/python/pdf/>

This cheat sheet is for more advanced learners. It covers class, string, and [list methods](https://blog.finxters.com/python-list-methods/) → <https://blog.finxters.com/python-list-methods/> as well as system calls from the `sys` module.

Once you're comfortable defining basic classes and command-line interfaces (CLIs), get this cheat sheet. It will take you to another level.

The Ultimative Python Cheat Sheet Course (5x Email Series) → <https://blog.finxter.com/subscribe/>

Python Cheat Sheet - Keywords		
"A puzzle a day to learn, code, and play" → Visit finxter.com		
Keyword	Description	Code example
<code>False, True</code>	Boolean data types	<code>False == (1 > 0), True == (2 > 1)</code>
<code>None</code>	Empty value constant	<code>def f(): x = 2 f() == None # True</code>
<code>and, or, not</code>	Logical operators: <code>(x and y)</code> = both <code>x</code> and <code>y</code> must be True <code>(x or y)</code> = either <code>x</code> or <code>y</code> must be True <code>(not x)</code> = <code>x</code> must be false	<code>x, y = True, False (x and y) == True # True (x or y) == True # True (not y) == True # True</code>
<code>break</code>	Ends loop prematurely	<code>while True: break # no infinite loop print("Hello world")</code>
<code>continue</code>	Finishes current loop iteration	<code>while True: continue print("#") # dead code</code>
<code>class</code>	Defines a new class → a real-world concept (object oriented programming)	<code>class Beer: x = 1.0 # litre def drink(self): self.x -= 0.8 b = Beer() # creates class with constructor b.drink() # beer empty: b.x == 0</code>
<code>def</code>	Defines a new function or class method. For latter, first parameter ("self") points to the class object. When calling class method, first parameter is implicit.	<code>x = int(input("your value: ")) if x > 3: print("Big") elif x < 3: print("Medium") else: print("Small")</code>
<code>for, while</code>	# For loop declaration <code>for i in [0,1,2]: print(i)</code>	<code># While loop - same semantics x = 0 while x < 3: print(x) x += 1</code>
<code>in</code>	Checks whether element is in sequence	<code>42 in [2, 39, 42] # True</code>
<code>is</code>	Checks whether both elements point to the same object	<code>y = x = 3 x is y # True [3] is [3] # False</code>
<code>lambda</code>	Function with no name (anonymous function)	<code>(lambda x: x + 3)(3) # returns 6</code>
<code>return</code>	Result of a function	<code>def incrementor(x): return x + 1 incrementor(4) # returns 5</code>

finxter

Python Cheat Sheet - Basic Data Types

"A puzzle a day to learn, code, and play" → Visit finxter.com

	Description	Example
Boolean	The Boolean data type is a truth value, either True or False.	<pre>## 1. Boolean Operations x, y = True, False print(x and not y) # True print(not x and y or x) # True ## 2. If condition evaluates to False if None or 0 or 0.0 or '' or [] or {} or set(): print("Dead code") # Not reached</pre>
Integer, Float	<p>The Boolean operators ordered by priority:</p> <pre>not x # If x is False, then x, else y x and y # If x is False, then x, else y x or y # If x is False, then y, else x</pre> <p>These comparison operators evaluate to True:</p> <pre>1 < 2 and 0 < 1 and 3 > 2 and 2 >= 2 and 1 == 1 and 1 != 0 and 0 != True and 1 == 1 and 1 <= 0 and 0 >= False and</pre>	<pre>## 1. Arithmetic Operations x, y = 3, 2 print(x + y) # 5 print(x - y) # 1 print(x * y) # 6 print(x / y) # 1.5 print(x // y) # 1 print(x % y) # 1s print(x ** y) # 9 print(abs(x)) # 3 print(int(x)) # 3 print(float(x)) # 3.0 print(x ** y) # 9</pre>
String	<p>Python Strings are sequences of characters. The four main ways to create strings are the following:</p> <ul style="list-style-type: none"> 1. Single quotes 'Yes' 2. Double quotes "Yes" 3. Triple quotes (multi-line) '''Yes''' 4. String method <code>str(5) == "5" # True</code> <p>These are whitespace characters in strings.</p> <ul style="list-style-type: none"> • Newline <code>\n</code> • Space <code>\s</code> • Tab <code>\t</code> 	<pre>s = "The project page was 31 years old" print(s[0]) # 'T' print(s[1:3]) # 'he' print(s[-1]) # 'old' x = s.split() # creates string array of words print(x[-3] + " " + x[-1] + " " + x[2] + " ") # 11 old pages ## 5. Most Important String Methods y = "This is laziness" print(y.rstrip()) # Remove Whitespace: 'This is lazy' print("Dordre".lower()) # Lowercase: 'dordre' print("Attention".upper()) # Uppercase: 'ATTENTION' print("Phone".lstrip("P")) # Strip: 'hone' print("smartphone".endswith("phone")) # True print("another".find("other")) # Match index: 2 print("chest".replace("ch", "m")) # 'met' print("Rumpelstiltskin".ljust(17)) # String length: 17 print("Year in earth") # Contains: True</pre>

finxter

Python Cheat Sheet - Complex Data Types

"A puzzle a day to learn, code, and play" → Visit finxter.com

	Description	Example
List	A container data type that stores a sequence of elements. Unlike strings, lists are mutable: modification possible.	<pre>l = [1, 2, 3] print(l[0]) # 1</pre>
Adding elements	Add elements to a list with (i) append(), (ii) insert(), or (iii) list concatenation. The append operation is very fast.	<pre>[1, 2, 2].append(4) # [1, 2, 2, 4] [1, 2, 4].insert(2,2) # [1, 2, 2, 4] [1, 2, 2] + [4] # [1, 2, 2, 4]</pre>
Removal	Removing an element can be slower.	<pre>[1, 2, 2, 4].remove(1) # [2, 2, 4]</pre>
Reversing	This reverses the order of list elements.	<pre>[1, 2, 3].reverse() # [3, 2, 1]</pre>
Sorting	Sorts a list. The computational complexity of sorting is O(n log n) for n list elements.	<pre>[2, 4, 2].sort() # [2, 2, 4]</pre>
Indexing	Find the index of occurrence of an element in the list & return its index. Can be slow as the whole list is traversed.	<pre>[2, 2, 4].index(2) # Index of element 4 is "0" [2, 2, 4].index(2, 1) # Index of element 2 after pos 1 is "1"</pre>
Stack	Python lists can be used intuitively as stack via the two list operations append() and pop().	<pre>stack = [] stack.append(42) # [42] stack.pop() # 42 (stack: [42]) stack.pop() # [] (stack: [])</pre>
Set	A set is an unordered collection of elements. Each can exist only once.	<pre>basket = {'apple', 'eggs', 'banana', 'orange'} sane = set(basket) # basket: {'apple', 'eggs', 'banana', 'orange'}</pre>
Dictionary	The dictionary is a useful data structure for reading and writing elements by specifying the key within the brackets. Use the keys() and values() functions to access all keys and values of the dictionary.	<pre>calories = {'apple': 52, 'banana': 89, 'choco': 546} print(calories['apple']) < calories['choco'] # True calories['apple'] = 74 print(calories['banana'] + calories['capru']) # False print('apple' in calories.keys()) # True print(52 in calories.values()) # True</pre>
Dictionary Looping	You can loop over the (key, value) pairs of a dictionary with the items() method.	<pre>for k, v in calories.items(): print(v) if v < 500 else None # 'chocolate'</pre>
Membership operator	Check with the <code>in</code> keyword whether a key or value within the dictionary contains an element. Set containment is faster than list containment.	<pre>basket = {'apple', 'eggs', 'banana', 'orange'} print('egg' in basket) # True print('mushroom' in basket) # False</pre>
List and Set Comprehension	List comprehension is the concise Python way to create lists. Use brackets plus an expression. Set comprehension is similar to list comprehension.	<pre># List comprehension l = [(i, i * x) for i in [1, 2, 3] for x in [1, 2, 3] if i < x] l2 = [x * y for x in range(3) for y in range(3) if x > y] print(l) # [[0, 0, 2], [1, 2, 4], [2, 4, 6]] # Set comprehension squares = {x*x2 for x in [0, 2, 4] if x < 4} # {0, 4}</pre>

finxter

Python Cheat Sheet - Classes

"A puzzle a day to learn, code, and play" → Visit finxter.com

Description		Example
Classes		<pre>class Dog: """ Blueprint of a dog """ # class variable shared by all instances species = "canis lupus" def __init__(self, name, color): self.name = name self.state = "sleeping" self.color = color def command(self, cmd): if cmd == self.name: self.state = "sit" elif cmd == "sit": self.state = "sit" else: self.state = "wag tail" def bark(self, freq): for i in range(freq): print("{}: {}".format(self.name, i)) Bello = Dog("bello", "black") Alice = Dog("alice", "white") print(Bello.color) # black print(Alice.color) # white</pre>
Instances		<pre>Bello.command("sit") print("alice") # Alice # [alice]: sit Alice.command("woof") print("bello") # bello.state # [bello]: wag tail Bello.command("woof") # [bello]: woof! # [alice]: woof!</pre>
Instance		<p>You are an instance of the class <code>human</code>. An instance is a concrete implementation of a class: all attributes of an instance are <code>instance-specific</code>. Your hair is blond, brown, or black - but never unspecified.</p> <p>Each instance has its own attributes independent of other instances. Yet, class variables are different. These are static values associated with the class, not the instances. You can access them via the same class variable <code>species</code> in the example.</p>
Self		<p>The first argument when defining any method is always the <code>self</code> argument. This argument specifies the instance on which you call the method.</p> <p><code>self</code> gives the Python interpreter the information about the concrete instance. To define a method, you use <code>self</code> to modify the instance attributes. But to call an instance method, you do not need to specify <code>self</code>.</p>
Creation		<p>You can create classes "on the fly" and use them as logical units to store complex data types.</p> <pre>class Employee(): def __init__(self, name, salary): self.name = name self.salary = salary Alice = Employee() Alice.firstname = "alice" Alice.lastname = "wonderland" Alice.salary = 122000 print(Alice.firstname + " " + Alice.lastname + " " + str(Alice.salary) + "\$") # alice wonderland 122000\$</pre>

finxter

Python Cheat Sheet - Functions and Tricks

"A puzzle a day to learn, code, and play" → Visit finxter.com

Description		Example	Result
A D V A C E D	map(func, iter) map(func, *args) map(func, *args)	Executes the function on all elements of the iterable.	<code>['r', 'g', 'b']</code>
F U T S	filter(func, iterable)	Filters out elements in iterable for which function returns False (or 0).	<code>[18]</code>
C T N S	string.strip() sorted(iter) sorted(iter, key=key)	Removes leading and trailing whitespace of string Sorts iterable in ascending order Sorts according to the key function in ascending order	<code>print("\n \t 42 \n").strip()</code> <code>sorted([8, 3, 2, 42, 5])</code> <code>sorted([8, 3, 2, 42, 5], key=lambda x: x if x==42 else x)</code>
H I Z	help(func)	Returns documentation of func	<code>... to uppercase.'</code>
T I R C S	zip(*iterables) Unzip	Groups the i-th elements of iterators (i, 0 ... together Equal to: 1) unpack the zipped list, 2) zip the result	<code>list(zip(['Alice', 'Anna'], ['Bob', 'Tom', 'Frank']))</code> <code>list(zip([('Alice', 'Bob'), ('Anna', 'Tom')]))</code>
E N S	enumerate(iter)	Assigns a counter value to each element of the iterable	<code>list(enumerate([('Alice', 'Bob'), ('Anna', 'Tom')]))</code>
HTTP Requests		Share files between PC and phone? Run command in PC's shell: <code>python -m http.server</code> & open in the phone's browser. You can now browse the files in the PC directory.	
Read comic		Import comic series stored in your web browser	
Zen of Python		'...beautiful is better than ugly. Explicit is ...'	
Swapping numbers		Swapping variables is a breeze in Python. No offense, Java!	<code>a, b = 'Jane', 'Alice'</code> <code>a, b = b, a</code> <code>a = 'Alice'</code> <code>b = 'Jane'</code>
Unpacking arguments		Use a sequence as function arguments via asterisk operator *. Use a dictionary (key, value) via double asterisk operator **	<code>def f(x, y, z): return x + y * z</code> <code>f(*[1, 3, 4])</code> <code>f(**{'x': 1, 'y': 2, 'z': 3})</code>
Extended Unpacking		Use unpacking for multiple assignment feature in Python	<code>a, *b = [1, 2, 3, 4, 5]</code> <code>a = 1</code> <code>b = [2, 3, 4, 5]</code>
Merge two dictionaries		Use unpacking to merge two dictionaries into a single one	<code>x = {'Alice': 10, 'Bob': 27, 'Ann': 22}</code> <code>y = {'Bob': 27, 'Ann': 22}</code> <code>z = {**x, **y}</code>

finxter

Want to learn Python well, but don't have much time?

Then this course is for you. It contains 5 carefully designed PDF cheat sheets. Each cheat sheet takes you one step further into the rabbit hole.

You will learn practical Python concepts from the hand-picked examples and code snippets. The topics include basic keywords, simple and complex data types, crucial string and list methods, and powerful Python one-liners.

If you lead a busy life and do not want to compromise on quality, this is the cheat sheet course for you!

Dataquest Data Science Cheat Sheet – Python Basics → <https://s3.amazonaws.com/dq-blog-files/python-cheat-sheet-basic.pdf>



The wonderful team at Dataquest has put together this comprehensive beginner-level Python cheat sheet.

It covers all the basic data types, [looping](#), → <https://blog.finxtter.com/python-loops/> and [reading files](#) → <https://blog.finxtter.com/how-to-read-a-file-line-by-line-and-store-into-a-list/>. It's beautifully designed and is the first of a series.

Dataquest Data Science Cheat Sheet – Intermediate → <https://s3.amazonaws.com/dq-blog-files/python-cheat-sheet-intermediate.pdf>



This intermediate-level cheat sheet is a follow-up of the other Dataquest cheat sheet. It contains intermediate `dtype` methods, looping, and handling errors.

Dataquest NumPy → <https://s3.amazonaws.com/dq-blog-files/numpy-cheat-sheet.pdf>

LEARN DATA SCIENCE ONLINE
Start Learning For Free - www.dataquest.io

Data Science Cheat Sheet
NumPy

KEY Use these abbreviations in this cheat sheet: arr - A numpy array object	IMPORTS Import these to start: <code>import numpy as np</code>
IMPORTING/EXPORTING <code>np.loadtxt('file.txt')</code> - From a text file <code>np.savetxt('file.txt',arr,'delimiter')</code> - From a CSV file <code>np.savetxt('file.txt',arr,'delimiter',')</code> - Writes to a CSV file <code>np.savetxt('file.csv',arr,'delimiter',',')</code> - Writes to a CSV file	
CREATING ARRAYS <code>np.array([1,2,3])</code> - One dimensional array <code>np.array([(1,2,3),(4,5,6)])</code> - Two dimensional array <code>np.zeros(3)</code> - Array of length 3 of all values 0 <code>np.zeros((3,4))</code> - Array of shape (3,4) of all values 0 <code>np.eye(4)</code> - 4x4 array of 0 with 1 on diagonal (Identity matrix) <code>np.ones(4)</code> - Array of 4 evenly divided values from 0 to 1 <code>np.arange(0,10,2)</code> - Array of values from 0 to less than 10 with step size 2 (Ex: 0,2,4,...) <code>np.full((3,3),3)</code> - 3x3 array with all values 3 <code>np.random.rand(4,5)</code> - 4x5 array of random floats <code>np.random.randn(3,7)*100</code> - 3x7 array of random floats between -100 and 100 <code>np.random.randint(5,10*(10**2))</code> - 2x3 array with random ints between 0-4	
INSPECTING PROPERTIES <code>arr.size</code> - Returns number of elements in arr <code>arr.shape</code> - Returns dimensions of arr (rows, columns) <code>arr.dtype</code> - Returns type of elements in arr <code>arr.itemsize</code> - Get size of each element in arr <code>arr.tobytes()</code> - Convert arr to a Python int <code>arr.tolist()</code> - Convert arr to a Python list <code>arr.tostring()</code> - New documentation for np.bytes_	
COPYING/EDITING/RESHAPING <code>np.copy(arr)</code> - Copies arr to new memory <code>arr.view(dtype)</code> - Creates view of arr <code>arr.sort()</code> - Sorts arr <code>arr.reshape(4,4)</code> - Sets specific axis of arr <code>arr.T</code> - Transposes arr (rows become columns and vice versa)	
ARRAY MANIPULATION <code>arr[1]</code> - Returns arr[1] (row 1) <code>arr[[1,2,3]]</code> - Returns arr[1:4] (Rows 1-3) <code>arr[[1,2],[3,4]]</code> - Returns arr[1:2,3:4] (Rows 1-2, Columns 3-4) <code>arr[[1,2],[3,4],:]</code> - Returns arr[1:2,3:4,:] <code>arr[[1,2],[3,4],:]</code> - Returns arr[1:2,3:4,:]	
ADDING/REMOVING ELEMENTS <code>np.append(arr,[1,2,3],axis=0)</code> - Append values to end of arr <code>np.insert(arr,2,[1,2,3],axis=0)</code> - Inserts values into arr at index 2 <code>np.delete(arr,3,0)</code> - Deletes row on index 2 of arr <code>np.delete(arr,[3,4],1)</code> - Deletes columns on index 3 of arr	
COMBINING/FLATTENING <code>np.concatenate([arr1,arr2],axis=0)</code> - Adds arr2 to rows of arr1 <code>np.concatenate([arr1,arr2],axis=1)</code> - Adds arr2 to columns to end of arr1 <code>np.split(arr,-1)</code> - Splits arr into 2 sub-arrays <code>np.split(arr,2)</code> - Splits arr horizontally into the 2th index	
INDEXING/SLICING/SUBSETTING <code>arr[5]</code> - Returns the element at index 5 <code>arr[:2]</code> - Returns the 2D array element on index [1][1] <code>arr[1:2]</code> - Returns array element on index 1 to 2 <code>arr[1,1:2]</code> - Assigns array element on index [1][1] the value 10 <code>arr[0,1,2:3]</code> - Returns the 3D element at indices 0,1,2 (Or a 2D array returns rows 0,1,2) <code>arr[1:3,1:4]</code> - Returns the elements on rows 1,2 and columns 1,2,3,4 <code>arr[1:2]</code> - Returns the elements at indices 0,1 to all rows <code>arr[1:2,1:3]</code> - Returns an array with boolean values <code>arr[[1,0,1,0]]</code> - Returns an array with boolean values <code>arr[arr>1]</code> - Returns arr elements smaller than 1	
SCALAR MATH <code>np.add(arr,1)</code> - Add 1 to each array element <code>np.subtract(arr,-1)</code> - Subtract 1 from each array element <code>np.multiply(arr,2)</code> - Multiply each array element by 2 <code>np.divide(arr,4)</code> - Divide each array element by 4 (returns np.nan for division by zero) <code>np.mod(arr,2)</code> - Return each array element to its 10th power	
VECTOR MATH <code>np.add(arr1,arr2)</code> - Elementwise add arr1 to arr2 <code>np.subtract(arr1,arr2)</code> - Elementwise subtract arr2 from arr1 <code>np.multiply(arr1,arr2)</code> - Elementwise multiply arr1 by arr2 <code>np.divide(arr1,arr2)</code> - Elementwise divide arr1 by arr2 <code>np.abs(arr1+arr2)</code> - Elementwise case arr1 related to the power of arr2 <code>np.array_equal(arr1,arr2)</code> - Returns True if the arrays have the same shape and type <code>np.sqrt(arr)</code> - Squares root of each element in the array <code>np.log(arr)</code> - Natural log of each element in the array <code>np.log10(arr)</code> - Natural log of each element in the array <code>np.abs(arr)</code> - Absolute value of each element in the array <code>np.sum(arr)</code> - Sums up the values in arr <code>np.floor(arr)</code> - Rounds down to the nearest int <code>np.round(arr)</code> - Rounds to the nearest int	
STATISTICS <code>np.mean(arr, axis=0)</code> - Returns mean along 0 axis <code>arr.sum()</code> - Returns sum of arr <code>arr.min()</code> - Returns minimum value of arr <code>arr.max()</code> - Returns maximum value of specific axis <code>arr.var()</code> - Returns variance of arr <code>arr.std()</code> - Returns standard deviation of the elements in arr <code>arr.corrcoef()</code> - Returns correlation coefficient of arr	

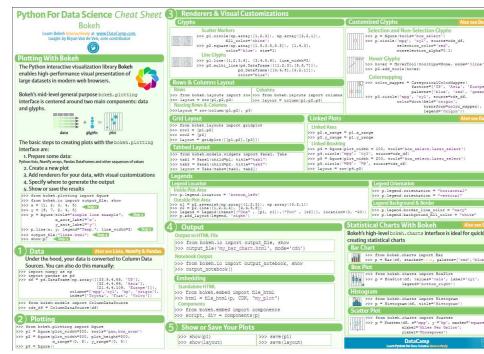
LEARN DATA SCIENCE ONLINE
Start Learning For Free - www.dataquest.io

NumPy → <https://blog.finxter.com/numpy-tutorial/> is at the heart of data science. Advanced libraries like [scikit-learn](#) → <https://blog.finxter.com/scikit-learn-cheat-sheets/>, [Tensorflow](#) → <https://blog.finxter.com/tensorflow-overview/>, [Pandas](#) → <https://blog.finxter.com/pandas-quickstart/>, and [Matplotlib](#) → <https://blog.finxter.com/best-matplotlib-cheat-sheet/> are built on NumPy arrays.

You need to understand NumPy before you can thrive in data science and machine learning. The topics of this cheat sheet are creating arrays, combining arrays, scalar math, vector math, and statistics.

This is only one great NumPy cheat sheet—if you want to get more, [check out our article on the 10 best NumPy cheat sheets](#) → <https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/>!

Python For Data Science (Bokeh) → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Bokeh_Cheat_Sheet.pdf



Want to master the visualization library Bokeh → <https://docs.bokeh.org/en/latest/index.html>? This cheat sheet is for you! It contains all the basic Bokeh commands to get your beautiful visualizations going fast!

Pandas Cheat Sheet for Data Science → https://drive.google.com/file/d/1UHK8wtWbADvHKXFC937IS6MTnISZC_zB/view



[Pandas](https://pandas.pydata.org) → <https://pandas.pydata.org> is everywhere. If you want to master “*the Excel library for Python coders*”, why not start with this cheat sheet? It’ll get you started fast and introduces the most important Pandas functions to you.

You can find a best-of article about the → <https://blog.finxter.com/pandas-cheat-sheets/7> → <https://blog.finxter.com/pandas-cheat-sheets/> best Pandas Cheat Sheets here. → <https://blog.finxter.com/pandas-cheat-sheets/>

Regular Expressions Cheat Sheet → <https://www.dataquest.io/wp-content/uploads/2019/03/python-regular-expressions-cheat-sheet.pdf>

DATAQUEST

LEARN DATA SCIENCE ONLINE
Start Learning For Free - www.dataquest.io

Data Science Cheat Sheet

Python Regular Expressions

SPECIAL CHARACTERS

- | Matches the expression to its right at the start or end. It matches every such instance before/behind the string.
- | Matches the expression to the left at the end of a string. It matches every such instance before/each in the string.
- \ Escapes special characters except line terminators like \n.
- \ Escapes special characters or denotes character class.
- \d Matches a digit 0-9.
- \w| Matches a word character or if A \n matched it, it is left unchanged.
- +| Greedily matches the expression to its left or more times.
- *| Non-greedily matches the expression to its left 0 or more times.
- ?| Matches the expression to its left 0 or 1 times. And, it is added to qualities to make it non-greedy. The pattern matches in a non-greedy manner.
- [e]| Matches the expression to its left once, and not less.
- [^e]| Matches the expression to its left = n times, and not less.
- [^e,?]| Matches the expression to its left = n times, and ignores > n. See above.

CHARACTER CLASSES

AUTHOR-SPECIFIED SEQUENCES

- \w\w\w| Matches three word characters, which mean a-z, A-z, and 0-9. It also matches the underscore, _.
- \d\d\d| Matches three digits, which means 0-9.
- \w\w\d| Matches two word characters followed by a digit.
- \w\w\w\d| Matches whitespace characters, which include the \t, \n, \v, and space characters.
- \w\w\w\w| Matches non-whitespace characters.
- \w\w\w\w\w| Matches five word characters starting at the start and end of the string, before and after \w and \d.
- \w\| Matches where \w does not, that is, the boundary of \w characters.

SETS

- []| Contains a set of characters to match.
- [a-e]| Matches either a, b, c, or d. It does not match e.
- [a-e]*| Matches any alphabet from a to z.
- [a-e]+| Matches a+, or it is - matches + (a sequence of escape sequences).
- [a-e] {n}| Matches, because - is not being used to indicate a series of characters.
- [a-e] {0, n}| Matches characters from a to z, n times.
- [a-e] {n, m}| Special characters become literal inside a set, so this matches {, +, and }.
- [a-e] {n, m}+| Adding + excludes any character in the set. If there is no +, it matches characters that are not a, b, c, or d.

GROUPS

- ()| Matches the expression inside the parentheses and groups it.
- (?i)| Makes matches like this, \w\w\w, as an ignore case operation. It matches on the character immediately to its right.
- (\w\w\w)| Matches the expression \w\w\w, and it can be accessed with the group name.
- (a|b)*| Matches a, a, L, L, s, u, u, and x are RegEx.
- a|—| Matches ASCII only
- | Ignore case
- L| Local dependent
- x| Multiline
- s| Matches all
- u| Matches unicode
- x| Verbose

POPULAR PYTHON RE MODULE FUNCTIONS

- re.findall(“\w”, “”) | Matches all instances of an expression in a string and returns them in a list.
- re.match(“\w”, “”) | Matches the first instance of an expression in a string, and returns it as a re.Match object.
- re.split(“\w”, “”) | Splits a string into a list using the delimiter A.
- re.sub(“\w”, “C”, “”) | Replace a \w in the string C.

LEARN DATA SCIENCE ONLINE
Start Learning For Free - www.dataquest.io

Regex to the rescue! [Regular expressions → https://blog.finxters.com/python-regex/](https://blog.finxters.com/python-regex/) are wildly important for anyone who handles large amounts of text programmatically (ask Google).

This cheat sheet introduces the most important Regex commands for quick reference. Download & master these regular expressions!

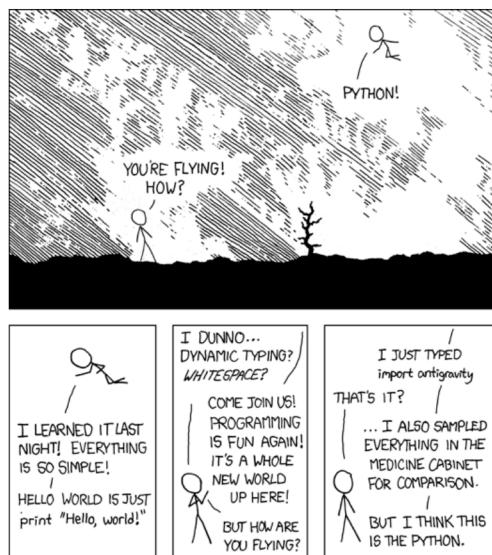
If you love cheat sheets, here are some interesting references for you (lots of more PDF downloads):

Related Articles:

- [Collection] 11 Python Cheat Sheets Every Python Coder Must Own → <https://blog.finxter.com/collection-5-cheat-sheets-every-python-coder-must-own/>
 - [Python OOP Cheat Sheet] A Simple Overview of Object-Oriented Programming → <https://blog.finxter.com/object-oriented-programming-terminology-cheat-sheet/>
 - [Collection] 15 Mind-Blowing Machine Learning Cheat Sheets to Pin to Your Toilet Wall → <https://blog.finxter.com/machine-learning-cheat-sheets/>
 - Your 8+ Free Python Cheat Sheet [Course] → <https://blog.finxter.com/python-cheat-sheets/>
 - Python Beginner Cheat Sheet: 19 Keywords Every Coder Must Know → <https://blog.finxter.com/python-cheat-sheet/>

- [Python Functions and Tricks Cheat Sheet](https://blog.finxter.com/python-cheat-sheet-functions-and-tricks/) → <https://blog.finxter.com/python-cheat-sheet-functions-and-tricks/>
- [Python Cheat Sheet: 14 Interview Questions](https://blog.finxter.com/python-interview-questions/) → <https://blog.finxter.com/python-interview-questions/>
- [Beautiful Pandas Cheat Sheets](https://blog.finxter.com/pandas-cheat-sheets/) → <https://blog.finxter.com/pandas-cheat-sheets/>
- [10 Best NumPy Cheat Sheets](https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-know/) → <https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-know/>
- [Python List Methods Cheat Sheet \[Instant PDF Download\]](https://blog.finxter.com/python-list-methods-cheat-sheet-instant-pdf-download/) → <https://blog.finxter.com/python-list-methods-cheat-sheet-instant-pdf-download/>
- [\[Cheat Sheet\] 6 Pillar Machine Learning Algorithms](https://blog.finxter.com/cheat-sheet-6-pillar-machine-learning-algorithms/) → <https://blog.finxter.com/cheat-sheet-6-pillar-machine-learning-algorithms/>

Programming Humor – Python



"I wrote 20 short programs in Python yesterday. It was wonderful. Perl, I'm leaving you." – [xkcd](https://imgs.xkcd.com/comics/python.png) → <https://imgs.xkcd.com/comics/python.png>

g

Where to Go From Here?

Enough theory. Let's get some practice!

Coders get paid six figures and more because they can solve problems more effectively using machine intelligence and automation.

To become more successful in coding, solve more real problems for real people. That's how you polish the skills you really need in practice. After all, what's the use of learning theory that nobody ever needs?

You build high-value coding skills by working on practical coding projects!

Do you want to stop learning with toy projects and focus on practical code projects that earn you money and solve real problems for people?

🚀 If your answer is **YES!**, consider becoming a [Python freelance developer](https://blog.finxter.com/become-python-freelancer-course/) → <https://blog.finxter.com/become-python-freelancer-course/>! It's the best way of approaching the task of improving your Python skills—even if you are a complete beginner.

If you just want to learn about the freelancing opportunity, feel free to watch my free webinar "[How to Build Your High-Income Skill Python](https://blog.finxter.com/webinar-freelancer/)" → <https://blog.finxter.com/webinar-freelancer/> and learn how I grew my coding business online and how you can, too—from the comfort of your own home.

[Join the free webinar now!](https://blog.finxter.com/webinar-freelancer/) → <https://blog.finxter.com/webinar-freelancer/>



While working as a researcher in distributed systems, [Dr. Christian Mayer](https://blog.finxter.com/about/) → <https://blog.finxter.com/about/> found his love for teaching computer science students.

To help students reach higher levels of Python success, he founded the programming education website [Finxter.com](https://finxter.com/) → <https://finxter.com/> that has taught exponential skills to millions of coders worldwide. He's the author of the best-selling programming books [Python One-Liners](https://amzn.to/2WAYeJE) → <https://amzn.to/2WAYeJE> (NoStarch 2020), [The Art of Clean Code](https://www.amazon.com/Art-Clean-Code-Practices-Complexity/dp/1718502184) → <https://www.amazon.com/Art-Clean-Code-Practices-Complexity/dp/1718502184> (NoStarch 2022), and [The Book of Dash](https://www.amazon.com/Python-Dash-Christian-Mayer-dp-1718502222/dp/1718502222) → <https://www.amazon.com/Python-Dash-Christian-Mayer-dp-1718502222/dp/1718502222> (NoStarch 2022). Chris also coauthored the [Coffee Break Python](http://coffeebreakpython.com/) → <http://coffeebreakpython.com/> series of self-published books. He's a computer science enthusiast, [freelancer](https://blog.finxter.com/become-python-freelancer-course/) → <https://blog.finxter.com/become-python-freelancer-course/>, and owner of one of the top 10 largest [Python blogs](https://blog.finxter.com/) → <https://blog.finxter.com/> worldwide.

His passions are writing, reading, and coding. But his greatest passion is to serve aspiring coders through Finxter and help them to boost their skills. You can [join his free email academy here](https://blog.finxter.com/email-academy/). → <https://blog.finxter.com/email-academy/>

3.12.2 Documentation

Source: <http://docs.python.org>

© Copyright → <http://docs.python.org/copyright.html> 2001-2024, Python Software Foundation.

This page is licensed under the Python Software Foundation License Version 2.

Examples, recipes, and other code in the documentation are additionally licensed under the Zero Clause BSD License.

See [History and License](#) → <http://docs.python.org/license.html> for more information.

The Python Software Foundation is a non-profit corporation. [Please donate](#), → <https://www.python.org/psf/donations/>

Last updated on Mar 11, 2024 (21:22 UTC). [Found a bug](#) → <http://docs.python.org/bugs.html>?

Created using [Sphinx](#) → <https://www.sphinx-doc.org/> 7.2.6.

The Python Tutorial

Source: <http://docs.python.org/tut/>

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.

For a description of standard objects and modules, see [The Python Standard Library → http://docs.python.org/library/index.html#library-index](http://docs.python.org/library/index.html#library-index). [The Python Language Reference → http://docs.python.org/reference/index.html#reference-index](http://docs.python.org/reference/index.html#reference-index) gives a more formal definition of the language. To write extensions in C or C++, read [Extending and Embedding the Python Interpreter → http://docs.python.org/extending/index.html#extending-index](http://docs.python.org/extending/index.html#extending-index) and [Python/C API Reference Manual → http://docs.python.org/c-api/index.html#c-api-index](http://docs.python.org/c-api/index.html#c-api-index). There are also several books covering Python in depth.

This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most noteworthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in [The Python Standard Library → http://docs.python.org/library/index.html#library-index](http://docs.python.org/library/index.html#library-index).

The [Glossary → http://docs.python.org/glossary.html#glossary](http://docs.python.org/glossary.html#glossary) is also worth going through.

- [1. Whetting Your Appetite → http://docs.python.org/tut/appetite.html](http://docs.python.org/tut/appetite.html)
- [2. Using the Python Interpreter → http://docs.python.org/tut/interpreter.html](http://docs.python.org/tut/interpreter.html)
 - [2.1. Invoking the Interpreter → http://docs.python.org/tut/interpreter.html#invoking-the-interpreter](http://docs.python.org/tut/interpreter.html#invoking-the-interpreter)
 - [2.1.1. Argument Passing → http://docs.python.org/tut/interpreter.html#argument-passing](http://docs.python.org/tut/interpreter.html#argument-passing)
 - [2.1.2. Interactive Mode → http://docs.python.org/tut/interpreter.html#interactive-mode](http://docs.python.org/tut/interpreter.html#interactive-mode)
 - [2.2. The Interpreter and Its Environment → http://docs.python.org/tut/interpreter.html#the-interpreter-and-its-environment](http://docs.python.org/tut/interpreter.html#the-interpreter-and-its-environment)
 - [2.2.1. Source Code Encoding → http://docs.python.org/tut/interpreter.html#source-code-encoding](http://docs.python.org/tut/interpreter.html#source-code-encoding)
- [3. An Informal Introduction to Python → http://docs.python.org/tut/introduction.html](http://docs.python.org/tut/introduction.html)
 - [3.1. Using Python as a Calculator → http://docs.python.org/tut/introduction.html#using-python-as-a-calculator](http://docs.python.org/tut/introduction.html#using-python-as-a-calculator)
 - [3.1.1. Numbers → http://docs.python.org/tut/introduction.html#numbers](http://docs.python.org/tut/introduction.html#numbers)
 - [3.1.2. Text → http://docs.python.org/tut/introduction.html#text](http://docs.python.org/tut/introduction.html#text)
 - [3.1.3. Lists → http://docs.python.org/tut/introduction.html#lists](http://docs.python.org/tut/introduction.html#lists)

- [3.2. First Steps Towards Programming](http://docs.python.org/tut/introduction.html#first-steps-towards-programming) → <http://docs.python.org/tut/introduction.html#first-steps-towards-programming>
- [4. More Control Flow Tools](http://docs.python.org/tut/controlflow.html) → <http://docs.python.org/tut/controlflow.html>
 - [4.1. if Statements](http://docs.python.org/tut/controlflow.html#if-statements) → <http://docs.python.org/tut/controlflow.html#if-statements>
 - [4.2. for Statements](http://docs.python.org/tut/controlflow.html#for-statements) → <http://docs.python.org/tut/controlflow.html#for-statements>
 - [4.3. The range\(\) Function](http://docs.python.org/tut/controlflow.html#the-range-function) → <http://docs.python.org/tut/controlflow.html#the-range-function>
 - [4.4. break and continue Statements, and else Clauses on Loops](http://docs.python.org/tut/controlflow.html#break-and-continue-statements-and-else-clauses-on-loops) → <http://docs.python.org/tut/controlflow.html#break-and-continue-statements-and-else-clauses-on-loops>
 - [4.5. pass Statements](http://docs.python.org/tut/controlflow.html#pass-statements) → <http://docs.python.org/tut/controlflow.html#pass-statements>
 - [4.6. match Statements](http://docs.python.org/tut/controlflow.html#match-statements) → <http://docs.python.org/tut/controlflow.html#match-statements>
 - [4.7. Defining Functions](http://docs.python.org/tut/controlflow.html#defineing-functions) → <http://docs.python.org/tut/controlflow.html#defineing-functions>
 - [4.8. More on Defining Functions](http://docs.python.org/tut/controlflow.html#more-on-defining-functions) → <http://docs.python.org/tut/controlflow.html#more-on-defining-functions>
 - [4.8.1. Default Argument Values](http://docs.python.org/tut/controlflow.html#default-argument-values) → <http://docs.python.org/tut/controlflow.html#default-argument-values>
 - [4.8.2. Keyword Arguments](http://docs.python.org/tut/controlflow.html#keyword-arguments) → <http://docs.python.org/tut/controlflow.html#keyword-arguments>
 - [4.8.3. Special parameters](http://docs.python.org/tut/controlflow.html#special-parameters) → <http://docs.python.org/tut/controlflow.html#special-parameters>
 - [4.8.3.1. Positional-or-Keyword Arguments](http://docs.python.org/tut/controlflow.html#positionals-or-keyword-arguments) → <http://docs.python.org/tut/controlflow.html#positionals-or-keyword-arguments>
 - [4.8.3.2. Positional-Only Parameters](http://docs.python.org/tut/controlflow.html#positional-only-parameters) → <http://docs.python.org/tut/controlflow.html#positional-only-parameters>
 - [4.8.3.3. Keyword-Only Arguments](http://docs.python.org/tut/controlflow.html#keyword-only-arguments) → <http://docs.python.org/tut/controlflow.html#keyword-only-arguments>
 - [4.8.3.4. Function Examples](http://docs.python.org/tut/controlflow.html#function-examples) → <http://docs.python.org/tut/controlflow.html#function-examples>
 - [4.8.3.5. Recap](http://docs.python.org/tut/controlflow.html#recap) → <http://docs.python.org/tut/controlflow.html#recap>
 - [4.8.4. Arbitrary Argument Lists](http://docs.python.org/tut/controlflow.html#arbitrary-argument-lists) → <http://docs.python.org/tut/controlflow.html#arbitrary-argument-lists>
 - [4.8.5. Unpacking Argument Lists](http://docs.python.org/tut/controlflow.html#unpacking-argument-lists) → <http://docs.python.org/tut/controlflow.html#unpacking-argument-lists>
 - [4.8.6. Lambda Expressions](http://docs.python.org/tut/controlflow.html#lambda-expressions) → <http://docs.python.org/tut/controlflow.html#lambda-expressions>
 - [4.8.7. Documentation Strings](http://docs.python.org/tut/controlflow.html#documentation-strings) → <http://docs.python.org/tut/controlflow.html#documentation-strings>
 - [4.8.8. Function Annotations](http://docs.python.org/tut/controlflow.html#function-annotations) → <http://docs.python.org/tut/controlflow.html#function-annotations>
 - [4.9. Intermezzo: Coding Style](http://docs.python.org/tut/controlflow.html#intermezzo-coding-style) → <http://docs.python.org/tut/controlflow.html#intermezzo-coding-style>
- [5. Data Structures](http://docs.python.org/tut/datastructures.html) → <http://docs.python.org/tut/datastructures.html>
 - [5.1. More on Lists](http://docs.python.org/tut/datastructures.html#more-on-lists) → <http://docs.python.org/tut/datastructures.html#more-on-lists>
 - [5.1.1. Using Lists as Stacks](http://docs.python.org/tut/datastructures.html#using-lists-as-stacks) → <http://docs.python.org/tut/datastructures.html#using-lists-as-stacks>
 - [5.1.2. Using Lists as Queues](http://docs.python.org/tut/datastructures.html#using-lists-as-queues) → <http://docs.python.org/tut/datastructures.html#using-lists-as-queues>
 - [5.1.3. List Comprehensions](http://docs.python.org/tut/datastructures.html#list-comprehensions) → <http://docs.python.org/tut/datastructures.html#list-comprehensions>
 - [5.1.4. Nested List Comprehensions](http://docs.python.org/tut/datastructures.html#nested-list-comprehensions) → <http://docs.python.org/tut/datastructures.html#nested-list-comprehensions>
 - [5.2. The del statement](http://docs.python.org/tut/datastructures.html#the-del-statement) → <http://docs.python.org/tut/datastructures.html#the-del-statement>
 - [5.3. Tuples and Sequences](http://docs.python.org/tut/datastructures.html#tuples-and-sequences) → <http://docs.python.org/tut/datastructures.html#tuples-and-sequences>
 - [5.4. Sets](http://docs.python.org/tut/datastructures.html#sets) → <http://docs.python.org/tut/datastructures.html#sets>
 - [5.5. Dictionaries](http://docs.python.org/tut/datastructures.html#dictionaries) → <http://docs.python.org/tut/datastructures.html#dictionaries>
 - [5.6. Looping Techniques](http://docs.python.org/tut/datastructures.html#looping-techniques) → <http://docs.python.org/tut/datastructures.html#looping-techniques>
 - [5.7. More on Conditions](http://docs.python.org/tut/datastructures.html#more-on-conditions) → <http://docs.python.org/tut/datastructures.html#more-on-conditions>
 - [5.8. Comparing Sequences and Other Types](http://docs.python.org/tut/datastructures.html#comparing-sequences-and-other-types) → <http://docs.python.org/tut/datastructures.html#comparing-sequences-and-other-types>

- [6. Modules](http://docs.python.org/tut/modules.html) → <http://docs.python.org/tut/modules.html>
 - [6.1. More on Modules](http://docs.python.org/tut/modules.html#more-on-modules) → <http://docs.python.org/tut/modules.html#more-on-modules>
 - [6.1.1. Executing modules as scripts](http://docs.python.org/tut/modules.html#executing-modules-as-scripts) → <http://docs.python.org/tut/modules.html#executing-modules-as-scripts>
 - [6.1.2. The Module Search Path](http://docs.python.org/tut/modules.html#the-module-search-path) → <http://docs.python.org/tut/modules.html#the-module-search-path>
 - [6.1.3. “Compiled” Python files](http://docs.python.org/tut/modules.html#compiled-python-files) → <http://docs.python.org/tut/modules.html#compiled-python-files>
 - [6.2. Standard Modules](http://docs.python.org/tut/modules.html#standard-modules) → <http://docs.python.org/tut/modules.html#standard-modules>
 - [6.3. The dir\(\) Function](http://docs.python.org/tut/modules.html#the-dir-function) → <http://docs.python.org/tut/modules.html#the-dir-function>
 - [6.4. Packages](http://docs.python.org/tut/modules.html#packages) → <http://docs.python.org/tut/modules.html#packages>
 - [6.4.1. Importing * From a Package](http://docs.python.org/tut/modules.html#importing-from-a-package) → <http://docs.python.org/tut/modules.html#importing-from-a-package>
 - [6.4.2. Intra-package References](http://docs.python.org/tut/modules.html#intra-package-references) → <http://docs.python.org/tut/modules.html#intra-package-references>
 - [6.4.3. Packages in Multiple Directories](http://docs.python.org/tut/modules.html#packages-in-multiple-directories) → <http://docs.python.org/tut/modules.html#packages-in-multiple-directories>
- [7. Input and Output](http://docs.python.org/tut/inputoutput.html) → <http://docs.python.org/tut/inputoutput.html>
 - [7.1. Fancier Output Formatting](http://docs.python.org/tut/inputoutput.html#fancier-output-formatting) → <http://docs.python.org/tut/inputoutput.html#fancier-output-formatting>
 - [7.1.1. Formatted String Literals](http://docs.python.org/tut/inputoutput.html#formatted-string-literals) → <http://docs.python.org/tut/inputoutput.html#formatted-string-literals>
 - [7.1.2. The String format\(\) Method](http://docs.python.org/tut/inputoutput.html#the-string-format-method) → <http://docs.python.org/tut/inputoutput.html#the-string-format-method>
 - [7.1.3. Manual String Formatting](http://docs.python.org/tut/inputoutput.html#manual-string-formatting) → <http://docs.python.org/tut/inputoutput.html#manual-string-formatting>
 - [7.1.4. Old string formatting](http://docs.python.org/tut/inputoutput.html#old-string-formatting) → <http://docs.python.org/tut/inputoutput.html#old-string-formatting>
 - [7.2. Reading and Writing Files](http://docs.python.org/tut/inputoutput.html#reading-and-writing-files) → <http://docs.python.org/tut/inputoutput.html#reading-and-writing-files>
 - [7.2.1. Methods of File Objects](http://docs.python.org/tut/inputoutput.html#methods-of-file-objects) → <http://docs.python.org/tut/inputoutput.html#methods-of-file-objects>
 - [7.2.2. Saving structured data with json](http://docs.python.org/tut/inputoutput.html#saving-structured-data-with-json) → <http://docs.python.org/tut/inputoutput.html#saving-structured-data-with-json>
- [8. Errors and Exceptions](http://docs.python.org/tut/errors.html) → <http://docs.python.org/tut/errors.html>
 - [8.1. Syntax Errors](http://docs.python.org/tut/errors.html#syntax-errors) → <http://docs.python.org/tut/errors.html#syntax-errors>
 - [8.2. Exceptions](http://docs.python.org/tut/errors.html#exceptions) → <http://docs.python.org/tut/errors.html#exceptions>
 - [8.3. Handling Exceptions](http://docs.python.org/tut/errors.html#handling-exceptions) → <http://docs.python.org/tut/errors.html#handling-exceptions>
 - [8.4. Raising Exceptions](http://docs.python.org/tut/errors.html#raising-exceptions) → <http://docs.python.org/tut/errors.html#raising-exceptions>
 - [8.5. Exception Chaining](http://docs.python.org/tut/errors.html#exception-chaining) → <http://docs.python.org/tut/errors.html#exception-chaining>
 - [8.6. User-defined Exceptions](http://docs.python.org/tut/errors.html#user-defined-exceptions) → <http://docs.python.org/tut/errors.html#user-defined-exceptions>
 - [8.7. Defining Clean-up Actions](http://docs.python.org/tut/errors.html#defining-clean-up-actions) → <http://docs.python.org/tut/errors.html#defining-clean-up-actions>
 - [8.8. Predefined Clean-up Actions](http://docs.python.org/tut/errors.html#predefined-clean-up-actions) → <http://docs.python.org/tut/errors.html#predefined-clean-up-actions>
 - [8.9. Raising and Handling Multiple Unrelated Exceptions](http://docs.python.org/tut/errors.html#raising-and-handling-multiple-unrelated-exceptions) → <http://docs.python.org/tut/errors.html#raising-and-handling-multiple-unrelated-exceptions>
 - [8.10. Enriching Exceptions with Notes](http://docs.python.org/tut/errors.html#enriching-exceptions-with-notes) → <http://docs.python.org/tut/errors.html#enriching-exceptions-with-notes>
- [9. Classes](http://docs.python.org/tut/classes.html) → <http://docs.python.org/tut/classes.html>
 - [9.1. A Word About Names and Objects](http://docs.python.org/tut/classes.html#a-word-about-names-and-objects) → <http://docs.python.org/tut/classes.html#a-word-about-names-and-objects>
 - [9.2. Python Scopes and Namespaces](http://docs.python.org/tut/classes.html#python-scopes-and-namespaces) → <http://docs.python.org/tut/classes.html#python-scopes-and-namespaces>

- [9.2.1. Scopes and Namespaces Example](http://docs.python.org/tut/classes.html#scopes-and-namespaces-example) → <http://docs.python.org/tut/classes.html#scopes-and-namespaces-example>
- [9.3. A First Look at Classes](http://docs.python.org/tut/classes.html#a-first-look-at-classes) → <http://docs.python.org/tut/classes.html#a-first-look-at-classes>
 - [9.3.1. Class Definition Syntax](http://docs.python.org/tut/classes.html#class-definition-syntax) → <http://docs.python.org/tut/classes.html#class-definition-syntax>
 - [9.3.2. Class Objects](http://docs.python.org/tut/classes.html#class-objects) → <http://docs.python.org/tut/classes.html#class-objects>
 - [9.3.3. Instance Objects](http://docs.python.org/tut/classes.html#instance-objects) → <http://docs.python.org/tut/classes.html#instance-objects>
 - [9.3.4. Method Objects](http://docs.python.org/tut/classes.html#method-objects) → <http://docs.python.org/tut/classes.html#method-objects>
 - [9.3.5. Class and Instance Variables](http://docs.python.org/tut/classes.html#class-and-instance-variables) → <http://docs.python.org/tut/classes.html#class-and-instance-variables>
- [9.4. Random Remarks](http://docs.python.org/tut/classes.html#random-remarks) → <http://docs.python.org/tut/classes.html#random-remarks>
- [9.5. Inheritance](http://docs.python.org/tut/classes.html#inheritance) → <http://docs.python.org/tut/classes.html#inheritance>
 - [9.5.1. Multiple Inheritance](http://docs.python.org/tut/classes.html#multiple-inheritance) → <http://docs.python.org/tut/classes.html#multiple-inheritance>
- [9.6. Private Variables](http://docs.python.org/tut/classes.html#private-variables) → <http://docs.python.org/tut/classes.html#private-variables>
- [9.7. Odds and Ends](http://docs.python.org/tut/classes.html#odds-and-ends) → <http://docs.python.org/tut/classes.html#odds-and-ends>
- [9.8. Iterators](http://docs.python.org/tut/classes.html#iterators) → <http://docs.python.org/tut/classes.html#iterators>
- [9.9. Generators](http://docs.python.org/tut/classes.html#generators) → <http://docs.python.org/tut/classes.html#generators>
- [9.10. Generator Expressions](http://docs.python.org/tut/classes.html#generator-expressions) → <http://docs.python.org/tut/classes.html#generator-expressions>
- [10. Brief Tour of the Standard Library](http://docs.python.org/tut/stdlib.html) → <http://docs.python.org/tut/stdlib.html>
 - [10.1. Operating System Interface](http://docs.python.org/tut/stdlib.html#operating-system-interface) → <http://docs.python.org/tut/stdlib.html#operating-system-interface>
 - [10.2. File Wildcards](http://docs.python.org/tut/stdlib.html#file-wildcards) → <http://docs.python.org/tut/stdlib.html#file-wildcards>
 - [10.3. Command Line Arguments](http://docs.python.org/tut/stdlib.html#command-line-arguments) → <http://docs.python.org/tut/stdlib.html#command-line-arguments>
 - [10.4. Error Output Redirection and Program Termination](http://docs.python.org/tut/stdlib.html#error-output-redirection-and-program-termination) → <http://docs.python.org/tut/stdlib.html#error-output-redirection-and-program-termination>
 - [10.5. String Pattern Matching](http://docs.python.org/tut/stdlib.html#string-pattern-matching) → <http://docs.python.org/tut/stdlib.html#string-pattern-matching>
 - [10.6. Mathematics](http://docs.python.org/tut/stdlib.html#mathematics) → <http://docs.python.org/tut/stdlib.html#mathematics>
 - [10.7. Internet Access](http://docs.python.org/tut/stdlib.html#internet-access) → <http://docs.python.org/tut/stdlib.html#internet-access>
 - [10.8. Dates and Times](http://docs.python.org/tut/stdlib.html#dates-and-times) → <http://docs.python.org/tut/stdlib.html#dates-and-times>
 - [10.9. Data Compression](http://docs.python.org/tut/stdlib.html#data-compression) → <http://docs.python.org/tut/stdlib.html#data-compression>
 - [10.10. Performance Measurement](http://docs.python.org/tut/stdlib.html#performance-measurement) → <http://docs.python.org/tut/stdlib.html#performance-measurement>
 - [10.11. Quality Control](http://docs.python.org/tut/stdlib.html#quality-control) → <http://docs.python.org/tut/stdlib.html#quality-control>
 - [10.12. Batteries Included](http://docs.python.org/tut/stdlib.html#batteries-included) → <http://docs.python.org/tut/stdlib.html#batteries-included>
- [11. Brief Tour of the Standard Library — Part II](http://docs.python.org/tut/stdlib2.html) → <http://docs.python.org/tut/stdlib2.html>
 - [11.1. Output Formatting](http://docs.python.org/tut/stdlib2.html#output-formatting) → <http://docs.python.org/tut/stdlib2.html#output-formatting>
 - [11.2. Templating](http://docs.python.org/tut/stdlib2.html#templating) → <http://docs.python.org/tut/stdlib2.html#templating>
 - [11.3. Working with Binary Data Record Layouts](http://docs.python.org/tut/stdlib2.html#working-with-binary-data-record-layouts) → <http://docs.python.org/tut/stdlib2.html#working-with-binary-data-record-layouts>
 - [11.4. Multi-threading](http://docs.python.org/tut/stdlib2.html#multi-threading) → <http://docs.python.org/tut/stdlib2.html#multi-threading>
 - [11.5. Logging](http://docs.python.org/tut/stdlib2.html#logging) → <http://docs.python.org/tut/stdlib2.html#logging>
 - [11.6. Weak References](http://docs.python.org/tut/stdlib2.html#weak-references) → <http://docs.python.org/tut/stdlib2.html#weak-references>
 - [11.7. Tools for Working with Lists](http://docs.python.org/tut/stdlib2.html#tools-for-working-with-lists) → <http://docs.python.org/tut/stdlib2.html#tools-for-working-with-lists>
 - [11.8. Decimal Floating Point Arithmetic](http://docs.python.org/tut/stdlib2.html#decimal-floating-point-arithmetic) → <http://docs.python.org/tut/stdlib2.html#decimal-floating-point-arithmetic>
- [12. Virtual Environments and Packages](http://docs.python.org/tut/venv.html) → <http://docs.python.org/tut/venv.html>
 - [12.1. Introduction](http://docs.python.org/tut/venv.html#introduction) → <http://docs.python.org/tut/venv.html#introduction>

- [12.2. Creating Virtual Environments](http://docs.python.org/tut/venv.html#creating-virtual-environments) → <http://docs.python.org/tut/venv.html#creating-virtual-environments>
- [12.3. Managing Packages with pip](http://docs.python.org/tut/venv.html#managing-packages-with-pip) → <http://docs.python.org/tut/venv.html#managing-packages-with-pip>
- [13. What Now?](http://docs.python.org/tut/whatnow.html) → <http://docs.python.org/tut/whatnow.html>
- [14. Interactive Input Editing and History Substitution](http://docs.python.org/tut/interactive.html) → <http://docs.python.org/tut/interactive.html>
 - [14.1. Tab Completion and History Editing](http://docs.python.org/tut/interactive.html#tab-completion-and-history-editing) → <http://docs.python.org/tut/interactive.html#tab-completion-and-history-editing>
 - [14.2. Alternatives to the Interactive Interpreter](http://docs.python.org/tut/interactive.html#alternatives-to-the-interactive-interpreter) → <http://docs.python.org/tut/interactive.html#alternatives-to-the-interactive-interpreter>
- [15. Floating Point Arithmetic: Issues and Limitations](http://docs.python.org/tut/floatingpoint.html) → <http://docs.python.org/tut/floatingpoint.html>
 - [15.1. Representation Error](http://docs.python.org/tut/floatingpoint.html#representation-error) → <http://docs.python.org/tut/floatingpoint.html#representation-error>
- [16. Appendix](http://docs.python.org/tut/appendix.html) → <http://docs.python.org/tut/appendix.html>
 - [16.1. Interactive Mode](http://docs.python.org/tut/appendix.html#interactive-mode) → <http://docs.python.org/tut/appendix.html#interactive-mode>
 - [16.1.1. Error Handling](http://docs.python.org/tut/appendix.html#error-handling) → <http://docs.python.org/tut/appendix.html#error-handling>
 - [16.1.2. Executable Python Scripts](http://docs.python.org/tut/appendix.html#executable-python-scripts) → <http://docs.python.org/tut/appendix.html#executable-python-scripts>
 - [16.1.3. The Interactive Startup File](http://docs.python.org/tut/appendix.html#the-interactive-startup-file) → <http://docs.python.org/tut/appendix.html#the-interactive-startup-file>
 - [16.1.4. The Customization Modules](http://docs.python.org/tut/appendix.html#the-customization-modules) → <http://docs.python.org/tut/appendix.html#the-customization-modules>

The Python Standard Library

Source: <http://docs.python.org/lib/>

While [The Python Language Reference](http://docs.python.org/reference/index.html#reference-index) → <http://docs.python.org/reference/index.html#reference-index> describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

In addition to the standard library, there is an active collection of hundreds of thousands of components (from individual programs and modules to packages and entire application development frameworks), available from the [Python Package Index](https://pypi.org/) → <https://pypi.org/>.

- [Introduction](http://docs.python.org/lib/intro.html) → <http://docs.python.org/lib/intro.html>
 - [Notes on availability](http://docs.python.org/lib/intro.html#notes-on-availability) → <http://docs.python.org/lib/intro.html#notes-on-availability>
- [Built-in Functions](http://docs.python.org/lib/functions.html) → <http://docs.python.org/lib/functions.html>
- [Built-in Constants](http://docs.python.org/lib/constants.html) → <http://docs.python.org/lib/constants.html>
 - [Constants added by the site module](http://docs.python.org/lib/constants.html#constants-added-by-the-site-module) → <http://docs.python.org/lib/constants.html#constants-added-by-the-site-module>
- [Built-in Types](http://docs.python.org/lib/stdtypes.html) → <http://docs.python.org/lib/stdtypes.html>
 - [Truth Value Testing](http://docs.python.org/lib/stdtypes.html#truth-value-testing) → <http://docs.python.org/lib/stdtypes.html#truth-value-testing>
 - [Boolean Operations — and, or, not](http://docs.python.org/lib/stdtypes.html#boolean-operations-and-or-not) → <http://docs.python.org/lib/stdtypes.html#boolean-operations-and-or-not>
 - [Comparisons](http://docs.python.org/lib/stdtypes.html#comparisons) → <http://docs.python.org/lib/stdtypes.html#comparisons>
 - [Numeric Types — int, float, complex](http://docs.python.org/lib/stdtypes.html#numeric-types-int-float-complex) → <http://docs.python.org/lib/stdtypes.html#numeric-types-int-float-complex>
 - [Boolean Type - bool](http://docs.python.org/lib/stdtypes.html#boolean-type-bool) → <http://docs.python.org/lib/stdtypes.html#boolean-type-bool>
 - [Iterator Types](http://docs.python.org/lib/stdtypes.html#iterator-types) → <http://docs.python.org/lib/stdtypes.html#iterator-types>
 - [Sequence Types — list, tuple, range](http://docs.python.org/lib/stdtypes.html#sequence-types-list-tuple-range) → <http://docs.python.org/lib/stdtypes.html#sequence-types-list-tuple-range>
 - [Text Sequence Type — str](http://docs.python.org/lib/stdtypes.html#text-sequence-type-str) → <http://docs.python.org/lib/stdtypes.html#text-sequence-type-str>
 - [Binary Sequence Types — bytes, bytearray, memoryview](http://docs.python.org/lib/stdtypes.html#binary-sequence-types-bytes-byt bytearray-memoryview) → <http://docs.python.org/lib/stdtypes.html#binary-sequence-types-bytes-byt bytearray-memoryview>
 - [Set Types — set, frozenset](http://docs.python.org/lib/stdtypes.html#set-types-set-frozenset) → <http://docs.python.org/lib/stdtypes.html#set-types-set-frozenset>
 - [Mapping Types — dict](http://docs.python.org/lib/stdtypes.html#mapping-types-dict) → <http://docs.python.org/lib/stdtypes.html#mapping-types-dict>
 - [Context Manager Types](http://docs.python.org/lib/stdtypes.html#context-manager-types) → <http://docs.python.org/lib/stdtypes.html#context-manager-types>

- [Type Annotation Types — Generic Alias, Union](http://docs.python.org/lib/stdtypes.html#type-annotation-types-generic-alias-union) → <http://docs.python.org/lib/stdtypes.html#type-annotation-types-generic-alias-union>
 - [Other Built-in Types](http://docs.python.org/lib/stdtypes.html#other-built-in-types) → <http://docs.python.org/lib/stdtypes.html#other-built-in-types>
 - [Special Attributes](http://docs.python.org/lib/stdtypes.html#special-attributes) → <http://docs.python.org/lib/stdtypes.html#special-attributes>
 - [Integer string conversion length limitation](http://docs.python.org/lib/stdtypes.html#integer-string-conversion-length-limitation) → <http://docs.python.org/lib/stdtypes.html#integer-string-conversion-length-limitation>
- [Built-in Exceptions](http://docs.python.org/lib/exceptions.html) → <http://docs.python.org/lib/exceptions.html>
 - [Exception context](http://docs.python.org/lib/exceptions.html#exception-context) → <http://docs.python.org/lib/exceptions.html#exception-context>
 - [Inheriting from built-in exceptions](http://docs.python.org/lib/exceptions.html#inheriting-from-built-in-exceptions) → <http://docs.python.org/lib/exceptions.html#inheriting-from-built-in-exceptions>
 - [Base classes](http://docs.python.org/lib/exceptions.html#base-classes) → <http://docs.python.org/lib/exceptions.html#base-classes>
 - [Concrete exceptions](http://docs.python.org/lib/exceptions.html#concrete-exceptions) → <http://docs.python.org/lib/exceptions.html#concrete-exceptions>
 - [Warnings](http://docs.python.org/lib/exceptions.html#warnings) → <http://docs.python.org/lib/exceptions.html#warnings>
 - [Exception groups](http://docs.python.org/lib/exceptions.html#exception-groups) → <http://docs.python.org/lib/exceptions.html#exception-groups>
 - [Exception hierarchy](http://docs.python.org/lib/exceptions.html#exception-hierarchy) → <http://docs.python.org/lib/exceptions.html#exception-hierarchy>
- [Text Processing Services](http://docs.python.org/lib/text.html) → <http://docs.python.org/lib/text.html>
 - [string — Common string operations](http://docs.python.org/lib/string.html) → <http://docs.python.org/lib/string.html>
 - [re — Regular expression operations](http://docs.python.org/lib/re.html) → <http://docs.python.org/lib/re.html>
 - [difflib — Helpers for computing deltas](http://docs.python.org/lib/difflib.html) → <http://docs.python.org/lib/difflib.html>
 - [textwrap — Text wrapping and filling](http://docs.python.org/lib/textwrap.html) → <http://docs.python.org/lib/textwrap.html>
 - [unicodedata — Unicode Database](http://docs.python.org/lib/unicodedata.html) → <http://docs.python.org/lib/unicodedata.html>
 - [stringprep — Internet String Preparation](http://docs.python.org/lib/stringprep.html) → <http://docs.python.org/lib/stringprep.html>
 - [readline — GNU readline interface](http://docs.python.org/lib/readline.html) → <http://docs.python.org/lib/readline.html>
 - [rlcompleter — Completion function for GNU readline](http://docs.python.org/lib/ricompleter.html) → <http://docs.python.org/lib/ricompleter.html>
- [Binary Data Services](http://docs.python.org/lib/binary.html) → <http://docs.python.org/lib/binary.html>
 - [struct — Interpret bytes as packed binary data](http://docs.python.org/lib/struct.html) → <http://docs.python.org/lib/struct.html>
 - [codecs — Codec registry and base classes](http://docs.python.org/lib/codecs.html) → <http://docs.python.org/lib/codecs.html>
- [Data Types](http://docs.python.org/lib/datatypes.html) → <http://docs.python.org/lib/datatypes.html>
 - [datetime — Basic date and time types](http://docs.python.org/lib/datetime.html) → <http://docs.python.org/lib/datetime.html>
 - [zoneinfo — IANA time zone support](http://docs.python.org/lib/zoneinfo.html) → <http://docs.python.org/lib/zoneinfo.html>
 - [calendar — General calendar-related functions](http://docs.python.org/lib/calendar.html) → <http://docs.python.org/lib/calendar.html>
 - [collections — Container datatypes](http://docs.python.org/lib/collections.html) → <http://docs.python.org/lib/collections.html>
 - [collections.abc — Abstract Base Classes for Containers](http://docs.python.org/lib/collections.abc.html) → <http://docs.python.org/lib/collections.abc.html>
 - [heappq — Heap queue algorithm](http://docs.python.org/lib/heappq.html) → <http://docs.python.org/lib/heappq.html>
 - [bisect — Array bisection algorithm](http://docs.python.org/lib/bisect.html) → <http://docs.python.org/lib/bisect.html>
 - [array — Efficient arrays of numeric values](http://docs.python.org/lib/array.html) → <http://docs.python.org/lib/array.html>
 - [weakref — Weak references](http://docs.python.org/lib/weakref.html) → <http://docs.python.org/lib/weakref.html>
 - [types — Dynamic type creation and names for built-in types](http://docs.python.org/lib/types.html) → <http://docs.python.org/lib/types.html>
 - [copy — Shallow and deep copy operations](http://docs.python.org/lib/copy.html) → <http://docs.python.org/lib/copy.html>
 - [pprint — Data pretty printer](http://docs.python.org/lib/pprint.html) → <http://docs.python.org/lib/pprint.html>
 - [reprlib — Alternate repr\(\) implementation](http://docs.python.org/lib/reprlib.html) → <http://docs.python.org/lib/reprlib.html>
 - [enum — Support for enumerations](http://docs.python.org/lib/enum.html) → <http://docs.python.org/lib/enum.html>
 - [graphlib — Functionality to operate with graph-like structures](http://docs.python.org/lib/graphlib.html) → <http://docs.python.org/lib/graphlib.html>
- [Numeric and Mathematical Modules](http://docs.python.org/lib/numeric.html) → <http://docs.python.org/lib/numeric.html>

- [numbers](http://docs.python.org/lib/numbers.html) — Numeric abstract base classes → <http://docs.python.org/lib/numbers.html>
 - [math](http://docs.python.org/lib/math.html) — Mathematical functions → <http://docs.python.org/lib/math.html>
 - [cmath](http://docs.python.org/lib/cmath.html) — Mathematical functions for complex numbers → <http://docs.python.org/lib/cmath.html>
 - [decimal](http://docs.python.org/lib/decimal.html) — Decimal fixed point and floating-point arithmetic → <http://docs.python.org/lib/decimal.html>
 - [fractions](http://docs.python.org/lib/fractions.html) — Rational numbers → <http://docs.python.org/lib/fractions.html>
 - [random](http://docs.python.org/lib/random.html) — Generate pseudo-random numbers → <http://docs.python.org/lib/random.html>
 - [statistics](http://docs.python.org/lib/statistics.html) — Mathematical statistics functions → <http://docs.python.org/lib/statistics.html>
- [Functional Programming Modules](http://docs.python.org/lib/functional.html) → <http://docs.python.org/lib/functional.html>
 - [itertools](http://docs.python.org/lib/itertools.html) — Functions creating iterators for efficient looping → <http://docs.python.org/lib/itertools.html>
 - [functools](http://docs.python.org/lib/functools.html) — Higher-order functions and operations on callable objects → <http://docs.python.org/lib/functools.html>
 - [operator](http://docs.python.org/lib/operator.html) — Standard operators as functions → <http://docs.python.org/lib/operator.html>
- [File and Directory Access](http://docs.python.org/lib/filesys.html) → <http://docs.python.org/lib/filesys.html>
 - [pathlib](http://docs.python.org/lib/pathlib.html) — Object-oriented filesystem paths → <http://docs.python.org/lib/pathlib.html>
 - [os.path](http://docs.python.org/lib/os.path.html) — Common pathname manipulations → <http://docs.python.org/lib/os.path.html>
 - [fileinput](http://docs.python.org/lib/fileinput.html) — Iterate over lines from multiple input streams → <http://docs.python.org/lib/fileinput.html>
 - [stat](http://docs.python.org/lib/stat.html) — Interpreting stat() results → <http://docs.python.org/lib/stat.html>
 - [filecmp](http://docs.python.org/lib/filecmp.html) — File and Directory Comparisons → <http://docs.python.org/lib/filecmp.html>
 - [tempfile](http://docs.python.org/lib/tempfile.html) — Generate temporary files and directories → <http://docs.python.org/lib/tempfile.html>
 - [glob](http://docs.python.org/lib/glob.html) — Unix style pathname pattern expansion → <http://docs.python.org/lib/glob.html>
 - [fnmatch](http://docs.python.org/lib/fnmatch.html) — Unix filename pattern matching → <http://docs.python.org/lib/fnmatch.html>
 - [linecache](http://docs.python.org/lib/linecache.html) — Random access to text lines → <http://docs.python.org/lib/linecache.html>
 - [shutil](http://docs.python.org/lib/shutil.html) — High-level file operations → <http://docs.python.org/lib/shutil.html>
- [Data Persistence](http://docs.python.org/lib/persistence.html) → <http://docs.python.org/lib/persistence.html>
 - [pickle](http://docs.python.org/lib/pickle.html) — Python object serialization → <http://docs.python.org/lib/pickle.html>
 - [copyreg](http://docs.python.org/lib/copyreg.html) — Register pickle support functions → <http://docs.python.org/lib/copyreg.html>
 - [shelve](http://docs.python.org/lib/shelve.html) — Python object persistence → <http://docs.python.org/lib/shelve.html>
 - [marshal](http://docs.python.org/lib/marshal.html) — Internal Python object serialization → <http://docs.python.org/lib/marshal.html>
 - [dbm](http://docs.python.org/lib/dbm.html) — Interfaces to Unix “databases” → <http://docs.python.org/lib/dbm.html>
 - [sqlite3](http://docs.python.org/lib/sqlite3.html) — DB-API 2.0 interface for SQLite databases → <http://docs.python.org/lib/sqlite3.html>
- [Data Compression and Archiving](http://docs.python.org/lib/archiving.html) → <http://docs.python.org/lib/archiving.html>
 - [zlib](http://docs.python.org/lib/zlib.html) — Compression compatible with gzip → <http://docs.python.org/lib/zlib.html>
 - [gzip](http://docs.python.org/lib/gzip.html) — Support for gzip files → <http://docs.python.org/lib/gzip.html>
 - [bz2](http://docs.python.org/lib/bz2.html) — Support for bzip2 compression → <http://docs.python.org/lib/bz2.html>
 - [lzma](http://docs.python.org/lib/lzma.html) — Compression using the LZMA algorithm → <http://docs.python.org/lib/lzma.html>
 - [zipfile](http://docs.python.org/lib/zipfile.html) — Work with ZIP archives → <http://docs.python.org/lib/zipfile.html>
 - [tarfile](http://docs.python.org/lib/tarfile.html) — Read and write tar archive files → <http://docs.python.org/lib/tarfile.html>
- [File Formats](http://docs.python.org/lib/fileformats.html) → <http://docs.python.org/lib/fileformats.html>
 - [csv](http://docs.python.org/lib/csv.html) — CSV File Reading and Writing → <http://docs.python.org/lib/csv.html>
 - [configparser](http://docs.python.org/lib/configparser.html) — Configuration file parser → <http://docs.python.org/lib/configparser.html>
 - [tomllib](http://docs.python.org/lib/tomllib.html) — Parse TOML files → <http://docs.python.org/lib/tomllib.html>
 - [netrc](http://docs.python.org/lib/netrc.html) — netrc file processing → <http://docs.python.org/lib/netrc.html>
 - [plistlib](http://docs.python.org/lib/plistlib.html) — Generate and parse Apple .plist files → <http://docs.python.org/lib/plistlib.html>
- [Cryptographic Services](http://docs.python.org/lib/crypto.html) → <http://docs.python.org/lib/crypto.html>

- [hashlib — Secure hashes and message digests](http://docs.python.org//lib/hashlib.html) → <http://docs.python.org//lib/hashlib.html>
 - [hmac — Keyed-Hashing for Message Authentication](http://docs.python.org//lib/hmac.html) → <http://docs.python.org//lib/hmac.html>
 - [secrets — Generate secure random numbers for managing secrets](http://docs.python.org//lib/secrets.html) → <http://docs.python.org//lib/secrets.html>
- [Generic Operating System Services](http://docs.python.org//lib/allos.html) → <http://docs.python.org//lib/allos.html>
 - [os — Miscellaneous operating system interfaces](http://docs.python.org//lib/os.html) → <http://docs.python.org//lib/os.html>
 - [io — Core tools for working with streams](http://docs.python.org//lib/io.html) → <http://docs.python.org//lib/io.html>
 - [time — Time access and conversions](http://docs.python.org//lib/time.html) → <http://docs.python.org//lib/time.html>
 - [argparse — Parser for command-line options, arguments and sub-commands](http://docs.python.org//lib/argparse.html) → <http://docs.python.org//lib/argparse.html>
 - [getopt — C-style parser for command line options](http://docs.python.org//lib/getopt.html) → <http://docs.python.org//lib/getopt.html>
 - [logging — Logging facility for Python](http://docs.python.org//lib/logging.html) → <http://docs.python.org//lib/logging.html>
 - [logging.config — Logging configuration](http://docs.python.org//lib/logging_config.html) → http://docs.python.org//lib/logging_config.html
 - [logging.handlers — Logging handlers](http://docs.python.org//lib/logging_handlers.html) → http://docs.python.org//lib/logging_handlers.html
 - [getpass — Portable password input](http://docs.python.org//lib/getpass.html) → <http://docs.python.org//lib/getpass.html>
 - [curses — Terminal handling for character-cell displays](http://docs.python.org//lib/curses.html) → <http://docs.python.org//lib/curses.html>
 - [curses.textpad — Text input widget for curses programs](http://docs.python.org//lib/curses.html#module-curses.textpad) → <http://docs.python.org//lib/curses.html#module-curses.textpad>
 - [curses.ascii — Utilities for ASCII characters](http://docs.python.org//lib/curses.ascii.html) → <http://docs.python.org//lib/curses.ascii.html>
 - [curses.panel — A panel stack extension for curses](http://docs.python.org//lib/curses.panel.html) → <http://docs.python.org//lib/curses.panel.html>
 - [platform — Access to underlying platform's identifying data](http://docs.python.org//lib/platform.html) → <http://docs.python.org//lib/platform.html>
 - [errno — Standard errno system symbols](http://docs.python.org//lib/errno.html) → <http://docs.python.org//lib/errno.html>
 - [ctypes — A foreign function library for Python](http://docs.python.org//lib/ctypes.html) → <http://docs.python.org//lib/ctypes.html>
- [Concurrent Execution](http://docs.python.org//lib/concurrency.html) → <http://docs.python.org//lib/concurrency.html>
 - [threading — Thread-based parallelism](http://docs.python.org//lib/threading.html) → <http://docs.python.org//lib/threading.html>
 - [multiprocessing — Process-based parallelism](http://docs.python.org//lib/multiprocessing.html) → <http://docs.python.org//lib/multiprocessing.html>
 - [multiprocessing.shared_memory — Shared memory for direct access across processes](http://docs.python.org//lib/multiprocessing.shared_memory.html) → http://docs.python.org//lib/multiprocessing.shared_memory.html
 - [The concurrent package](http://docs.python.org//lib/concurrent.html) → <http://docs.python.org//lib/concurrent.html>
 - [concurrent.futures — Launching parallel tasks](http://docs.python.org//lib/concurrent.futures.html) → <http://docs.python.org//lib/concurrent.futures.html>
 - [subprocess — Subprocess management](http://docs.python.org//lib/subprocess.html) → <http://docs.python.org//lib/subprocess.html>
 - [sched — Event scheduler](http://docs.python.org//lib/sched.html) → <http://docs.python.org//lib/sched.html>
 - [queue — A synchronized queue class](http://docs.python.org//lib/queue.html) → <http://docs.python.org//lib/queue.html>
 - [contextvars — Context Variables](http://docs.python.org//lib/contextvars.html) → <http://docs.python.org//lib/contextvars.html>
 - [thread — Low-level threading API](http://docs.python.org//lib/_thread.html) → http://docs.python.org//lib/_thread.html
- [Networking and Interprocess Communication](http://docs.python.org//lib/ipc.html) → <http://docs.python.org//lib/ipc.html>
 - [asyncio — Asynchronous I/O](http://docs.python.org//lib/asyncio.html) → <http://docs.python.org//lib/asyncio.html>
 - [socket — Low-level networking interface](http://docs.python.org//lib/socket.html) → <http://docs.python.org//lib/socket.html>
 - [ssl — TLS/SSL wrapper for socket objects](http://docs.python.org//lib/ssl.html) → <http://docs.python.org//lib/ssl.html>
 - [select — Waiting for I/O completion](http://docs.python.org//lib/select.html) → <http://docs.python.org//lib/select.html>
 - [selectors — High-level I/O multiplexing](http://docs.python.org//lib/selectors.html) → <http://docs.python.org//lib/selectors.html>
 - [signal — Set handlers for asynchronous events](http://docs.python.org//lib/signal.html) → <http://docs.python.org//lib/signal.html>
 - [mmap — Memory-mapped file support](http://docs.python.org//lib/mmap.html) → <http://docs.python.org//lib/mmap.html>
- [Internet Data Handling](http://docs.python.org//lib/netdata.html) → <http://docs.python.org//lib/netdata.html>
 - [email — An email and MIME handling package](http://docs.python.org//lib/email.html) → <http://docs.python.org//lib/email.html>

- [json](http://docs.python.org//lib/json.html) — JSON encoder and decoder → <http://docs.python.org//lib/json.html>
- [mailbox](http://docs.python.org//lib/mailbox.html) — Manipulate mailboxes in various formats → <http://docs.python.org//lib/mailbox.html>
- [mimetypes](http://docs.python.org//lib/mimetypes.html) — Map filenames to MIME types → <http://docs.python.org//lib/mimetypes.html>
- [base64](http://docs.python.org//lib/base64.html) — Base16, Base32, Base64, Base85 Data Encodings → <http://docs.python.org//lib/base64.html>
- [binascii](http://docs.python.org//lib/binascii.html) — Convert between binary and ASCII → <http://docs.python.org//lib/binascii.html>
- [quopri](http://docs.python.org//lib/quopri.html) — Encode and decode MIME quoted-printable data → <http://docs.python.org//lib/quopri.html>
- [Structured Markup Processing Tools](http://docs.python.org//lib/markup.html) → <http://docs.python.org//lib/markup.html>
 - [html](http://docs.python.org//lib/html.html) — HyperText Markup Language support → <http://docs.python.org//lib/html.html>
 - [html.parser](http://docs.python.org//lib/html.parser.html) — Simple HTML and XHTML parser → <http://docs.python.org//lib/html.parser.html>
 - [html.entities](http://docs.python.org//lib/html.entities.html) — Definitions of HTML general entities → <http://docs.python.org//lib/html.entities.html>
 - [XML Processing Modules](http://docs.python.org//lib/xml.html) → <http://docs.python.org//lib/xml.html>
 - [xml.etree.ElementTree](http://docs.python.org//lib/xml.etree.elementtree.html) — The ElementTree XML API → <http://docs.python.org//lib/xml.etree.elementtree.html>
 - [xml.dom](http://docs.python.org//lib/xml.dom.html) — The Document Object Model API → <http://docs.python.org//lib/xml.dom.html>
 - [xml.dom.minidom](http://docs.python.org//lib/xml.dom.minidom.html) — Minimal DOM implementation → <http://docs.python.org//lib/xml.dom.minidom.html>
 - [xml.dom.pulldom](http://docs.python.org//lib/xml.dom.pulldom.html) — Support for building partial DOM trees → <http://docs.python.org//lib/xml.dom.pulldom.html>
 - [xml.sax](http://docs.python.org//lib/xml.sax.html) — Support for SAX2 parsers → <http://docs.python.org//lib/xml.sax.html>
 - [xml.sax.handler](http://docs.python.org//lib/xml.sax.handler.html) — Base classes for SAX handlers → <http://docs.python.org//lib/xml.sax.handler.html>
 - [xml.sax.utils](http://docs.python.org//lib/xml.sax.utils.html) — SAX Utilities → <http://docs.python.org//lib/xml.sax.utils.html>
 - [xml.sax.xmlreader](http://docs.python.org//lib/xml.sax.xmlreader.html) — Interface for XML parsers → <http://docs.python.org//lib/xml.sax.xmlreader.html>
 - [xml.parsers.expat](http://docs.python.org//lib/pyexpat.html) — Fast XML parsing using Expat → <http://docs.python.org//lib/pyexpat.html>
- [Internet Protocols and Support](http://docs.python.org//lib/internet.html) → <http://docs.python.org//lib/internet.html>
 - [webbrowser](http://docs.python.org//lib/webbrowser.html) — Convenient web-browser controller → <http://docs.python.org//lib/webbrowser.html>
 - [wsgiref](http://docs.python.org//lib/wsgiref.html) — WSGI Utilities and Reference Implementation → <http://docs.python.org//lib/wsgiref.html>
 - [urllib](http://docs.python.org//lib/urllib.html) — URL handling modules → <http://docs.python.org//lib/urllib.html>
 - [urllib.request](http://docs.python.org//lib/urllib.request.html) — Extensible library for opening URLs → <http://docs.python.org//lib/urllib.request.html>
 - [urllib.response](http://docs.python.org//lib/urllib.request.html#module-urllib.response) — Response classes used by urllib → <http://docs.python.org//lib/urllib.request.html#module-urllib.response>
 - [urllib.parse](http://docs.python.org//lib/urllib.parse.html) — Parse URLs into components → <http://docs.python.org//lib/urllib.parse.html>
 - [urllib.error](http://docs.python.org//lib/urllib.error.html) — Exception classes raised by urllib.request → <http://docs.python.org//lib/urllib.error.html>
 - [urllib.robotparser](http://docs.python.org//lib/urllib.robotparser.html) — Parser for robots.txt → <http://docs.python.org//lib/urllib.robotparser.html>
 - [http](http://docs.python.org//lib/http.html) — HTTP modules → <http://docs.python.org//lib/http.html>
 - [http.client](http://docs.python.org//lib/http.client.html) — HTTP protocol client → <http://docs.python.org//lib/http.client.html>
 - [ftplib](http://docs.python.org//lib/ftplib.html) — FTP protocol client → <http://docs.python.org//lib/ftplib.html>
 - [poplib](http://docs.python.org//lib/poplib.html) — POP3 protocol client → <http://docs.python.org//lib/poplib.html>
 - [imaplib](http://docs.python.org//lib/imaplib.html) — IMAP4 protocol client → <http://docs.python.org//lib/imaplib.html>
 - [smtplib](http://docs.python.org//lib/smtplib.html) — SMTP.protocol client → <http://docs.python.org//lib/smtplib.html>
 - [uuid](http://docs.python.org//lib/uuid.html) — UUID objects according to **RFC 4122** → <http://docs.python.org//lib/uuid.html>
 - [socketserver](http://docs.python.org//lib/socketserver.html) — A framework for network servers → <http://docs.python.org//lib/socketserver.html>
 - [http.server](http://docs.python.org//lib/http.server.html) — HTTP servers → <http://docs.python.org//lib/http.server.html>
 - [http.cookies](http://docs.python.org//lib/http.cookies.html) — HTTP state management → <http://docs.python.org//lib/http.cookies.html>
 - [http.cookiejar](http://docs.python.org//lib/http.cookiejar.html) — Cookie handling for HTTP clients → <http://docs.python.org//lib/http.cookiejar.html>
 - [xmlrpc](http://docs.python.org//lib/xmlrpc.html) — XMLRPC server and client modules → <http://docs.python.org//lib/xmlrpc.html>
 - [xmlrpc.client](http://docs.python.org//lib/xmlrpc.client.html) — XML-RPC client access → <http://docs.python.org//lib/xmlrpc.client.html>
 - [xmlrpc.server](http://docs.python.org//lib/xmlrpc.server.html) — Basic XML-RPC servers → <http://docs.python.org//lib/xmlrpc.server.html>
 - [ipaddress](http://docs.python.org//lib/ipaddress.html) — IPv4/IPv6 manipulation library → <http://docs.python.org//lib/ipaddress.html>

- [Multimedia Services](http://docs.python.org//lib/mm.html) → <http://docs.python.org//lib/mm.html>
 - [wave](http://docs.python.org//lib/wave.html) — Read and write WAV files → <http://docs.python.org//lib/wave.html>
 - [colorsys](http://docs.python.org//lib/colorsys.html) — Conversions between color systems → <http://docs.python.org//lib/colorsys.html>
- [Internationalization](http://docs.python.org//lib/i18n.html) → <http://docs.python.org//lib/i18n.html>
 - [gettext](http://docs.python.org//lib/gettext.html) — Multilingual internationalization services → <http://docs.python.org//lib/gettext.html>
 - [locale](http://docs.python.org//lib/locale.html) — Internationalization services → <http://docs.python.org//lib/locale.html>
- [Program Frameworks](http://docs.python.org//lib/frameworks.html) → <http://docs.python.org//lib/frameworks.html>
 - [turtle](http://docs.python.org//lib/turtle.html) — Turtle graphics → <http://docs.python.org//lib/turtle.html>
 - [cmd](http://docs.python.org//lib/cmd.html) — Support for line-oriented command interpreters → <http://docs.python.org//lib/cmd.html>
 - [shlex](http://docs.python.org//lib/shlex.html) — Simple lexical analysis → <http://docs.python.org//lib/shlex.html>
- [Graphical User Interfaces with Tk](http://docs.python.org//lib/tk.html) → <http://docs.python.org//lib/tk.html>
 - [tkinter](http://docs.python.org//lib/tkinter.html) — Python interface to Tcl/Tk → <http://docs.python.org//lib/tkinter.html>
 - [tkinter.colorchooser](http://docs.python.org//lib/tkinter.colorchooser.html) — Color choosing dialog → <http://docs.python.org//lib/tkinter.colorchooser.html>
 - [tkinter.font](http://docs.python.org//lib/tkinter.font.html) — Tkinter font wrapper → <http://docs.python.org//lib/tkinter.font.html>
 - [Tkinter Dialogs](http://docs.python.org//lib/dialog.html) → <http://docs.python.org//lib/dialog.html>
 - [tkinter.messagebox](http://docs.python.org//lib/tkinter.messagebox.html) — Tkinter message prompts → <http://docs.python.org//lib/tkinter.messagebox.html>
 - [tkinter.scrolledtext](http://docs.python.org//lib/tkinter.scrolledtext.html) — Scrolled Text Widget → <http://docs.python.org//lib/tkinter.scrolledtext.html>
 - [tkinter.dnd](http://docs.python.org//lib/tkinter.dnd.html) — Drag and drop support → <http://docs.python.org//lib/tkinter.dnd.html>
 - [tkinter.ttk](http://docs.python.org//lib/tkinter.ttk.html) — Tk themed widgets → <http://docs.python.org//lib/tkinter.ttk.html>
 - [tkinter.tix](http://docs.python.org//lib/tkinter.tix.html) — Extension widgets for Tk → <http://docs.python.org//lib/tkinter.tix.html>
 - [IDLE](http://docs.python.org//lib/idle.html) → <http://docs.python.org//lib/idle.html>
- [Development Tools](http://docs.python.org//lib/development.html) → <http://docs.python.org//lib/development.html>
 - [typing](http://docs.python.org//lib/typing.html) — Support for type hints → <http://docs.python.org//lib/typing.html>
 - [pydoc](http://docs.python.org//lib/pydoc.html) — Documentation generator and online help system → <http://docs.python.org//lib/pydoc.html>
 - [Python Development Mode](http://docs.python.org//lib/devmode.html) → <http://docs.python.org//lib/devmode.html>
 - [doctest](http://docs.python.org//lib/doctest.html) — Test interactive Python examples → <http://docs.python.org//lib/doctest.html>
 - [unittest](http://docs.python.org//lib/unittest.html) — Unit testing framework → <http://docs.python.org//lib/unittest.html>
 - [unittest.mock](http://docs.python.org//lib/unittest.mock.html) — mock object library → <http://docs.python.org//lib/unittest.mock.html>
 - [unittest.mock](http://docs.python.org//lib/unittest.mock-examples.html) — getting started → <http://docs.python.org//lib/unittest.mock-examples.html>
 - [2to3](http://docs.python.org//lib/2to3.html) — Automated Python 2 to 3 code translation → <http://docs.python.org//lib/2to3.html>
 - [test](http://docs.python.org//lib/test.html) — Regression tests package for Python → <http://docs.python.org//lib/test.html>
 - [test.support](http://docs.python.org//lib/test.html#module-test.support) — Utilities for the Python test suite → <http://docs.python.org//lib/test.html#module-test.support>
 - [test.support.socket_helper](http://docs.python.org//lib/test.html#module-test.support.socket_helper) — Utilities for socket tests → http://docs.python.org//lib/test.html#module-test.support.socket_helper
 - [test.support.script_helper](http://docs.python.org//lib/test.html#module-test.support.script_helper) — Utilities for the Python execution tests → http://docs.python.org//lib/test.html#module-test.support.script_helper
 - [test.support.bytecode_helper](http://docs.python.org//lib/test.html#module-test.support.bytecode_helper) — Support tools for testing correct bytecode generation → http://docs.python.org//lib/test.html#module-test.support.bytecode_helper
 - [test.support.threading_helper](http://docs.python.org//lib/test.html#module-test.support.threading_helper) — Utilities for threading tests → http://docs.python.org//lib/test.html#module-test.support.threading_helper
 - [test.support.os_helper](http://docs.python.org//lib/test.html#module-test.support.os_helper) — Utilities for os tests → http://docs.python.org//lib/test.html#module-test.support.os_helper

- [test.support.import_helper](http://docs.python.org/lib/test.html#module-test.support.import_helper) — Utilities for import tests → http://docs.python.org/lib/test.html#module-test.support.import_helper
 - [test.support.warnings_helper](http://docs.python.org/lib/test.html#module-test.support.warnings_helper) — Utilities for warnings tests → http://docs.python.org/lib/test.html#module-test.support.warnings_helper
- [Debugging and Profiling](http://docs.python.org/lib/debug.html) → <http://docs.python.org/lib/debug.html>
 - [Audit events table](http://docs.python.org/lib/audit_events.html) → http://docs.python.org/lib/audit_events.html
 - [bdb](http://docs.python.org/lib/bdb.html) — Debugger framework → <http://docs.python.org/lib/bdb.html>
 - [faulthandler](http://docs.python.org/lib/faulthandler.html) — Dump the Python traceback → <http://docs.python.org/lib/faulthandler.html>
 - [pdb](http://docs.python.org/lib/pdb.html) — The Python Debugger → <http://docs.python.org/lib/pdb.html>
 - [The Python Profilers](http://docs.python.org/lib/profile.html) → <http://docs.python.org/lib/profile.html>
 - [timeit](http://docs.python.org/lib/timeit.html) — Measure execution time of small code snippets → <http://docs.python.org/lib/timeit.html>
 - [trace](http://docs.python.org/lib/trace.html) — Trace or track Python statement execution → <http://docs.python.org/lib/trace.html>
 - [tracemalloc](http://docs.python.org/lib/tracemalloc.html) — Trace memory allocations → <http://docs.python.org/lib/tracemalloc.html>
- [Software Packaging and Distribution](http://docs.python.org/lib/distribution.html) → <http://docs.python.org/lib/distribution.html>
 - [ensurepip](http://docs.python.org/lib/ensurepip.html) — Bootstrapping the pip installer → <http://docs.python.org/lib/ensurepip.html>
 - [venv](http://docs.python.org/lib/venv.html) — Creation of virtual environments → <http://docs.python.org/lib/venv.html>
 - [zipapp](http://docs.python.org/lib/zipapp.html) — Manage executable Python zip archives → <http://docs.python.org/lib/zipapp.html>
- [Python Runtime Services](http://docs.python.org/lib/python.html) → <http://docs.python.org/lib/python.html>
 - [sys](http://docs.python.org/lib/sys.html) — System-specific parameters and functions → <http://docs.python.org/lib/sys.html>
 - [sys.monitoring](http://docs.python.org/lib/sys_monitoring.html) — Execution event monitoring → http://docs.python.org/lib/sys_monitoring.html
 - [sysconfig](http://docs.python.org/lib/sysconfig.html) — Provide access to Python’s configuration information → <http://docs.python.org/lib/sysconfig.html>
 - [builtins](http://docs.python.org/lib/builtins.html) — Built-in objects → <http://docs.python.org/lib/builtins.html>
 - [_main_](http://docs.python.org/lib/_main_.html) — Top-level code environment → http://docs.python.org/lib/_main_.html
 - [warnings](http://docs.python.org/lib/warnings.html) — Warning control → <http://docs.python.org/lib/warnings.html>
 - [dataclasses](http://docs.python.org/lib/dataclasses.html) — Data Classes → <http://docs.python.org/lib/dataclasses.html>
 - [contextlib](http://docs.python.org/lib/contextlib.html) — Utilities for with-statement contexts → <http://docs.python.org/lib/contextlib.html>
 - [abc](http://docs.python.org/lib/abc.html) — Abstract Base Classes → <http://docs.python.org/lib/abc.html>
 - [atexit](http://docs.python.org/lib/atexit.html) — Exit handlers → <http://docs.python.org/lib/atexit.html>
 - [traceback](http://docs.python.org/lib/traceback.html) — Print or retrieve a stack traceback → <http://docs.python.org/lib/traceback.html>
 - [_future_](http://docs.python.org/lib/_future_.html) — Future statement definitions → http://docs.python.org/lib/_future_.html
 - [gc](http://docs.python.org/lib/gc.html) — Garbage Collector interface → <http://docs.python.org/lib/gc.html>
 - [inspect](http://docs.python.org/lib/inspect.html) — Inspect live objects → <http://docs.python.org/lib/inspect.html>
 - [site](http://docs.python.org/lib/site.html) — Site-specific configuration hook → <http://docs.python.org/lib/site.html>
- [Custom Python Interpreters](http://docs.python.org/lib/custominterp.html) → <http://docs.python.org/lib/custominterp.html>
 - [code](http://docs.python.org/lib/code.html) — Interpreter base classes → <http://docs.python.org/lib/code.html>
 - [codeop](http://docs.python.org/lib/codeop.html) — Compile Python code → <http://docs.python.org/lib/codeop.html>
- [Importing Modules](http://docs.python.org/lib/modules.html) → <http://docs.python.org/lib/modules.html>
 - [zipimport](http://docs.python.org/lib/zipimport.html) — Import modules from Zip archives → <http://docs.python.org/lib/zipimport.html>
 - [pkgutil](http://docs.python.org/lib/pkgutil.html) — Package extension utility → <http://docs.python.org/lib/pkgutil.html>
 - [modulefinder](http://docs.python.org/lib/modulefinder.html) — Find modules used by a script → <http://docs.python.org/lib/modulefinder.html>
 - [runpy](http://docs.python.org/lib/runpy.html) — Locating and executing Python modules → <http://docs.python.org/lib/runpy.html>
 - [importlib](http://docs.python.org/lib/importlib.html) — The implementation of import → <http://docs.python.org/lib/importlib.html>

- [importlib.resources](http://docs.python.org/lib/importlib_resources.html) – Package resource reading, opening and access → http://docs.python.org/lib/importlib_resources.html
 - [importlib.resources.abc](http://docs.python.org/lib/importlib_abc.html) – Abstract base classes for resources → http://docs.python.org/lib/importlib_abc.html
 - [importlib.metadata](http://docs.python.org/lib/importlib_metadata.html) – Accessing package metadata → http://docs.python.org/lib/importlib_metadata.html
 - [The initialization of the sys.path module search path](http://docs.python.org/lib/sys_path_init.html) → http://docs.python.org/lib/sys_path_init.html
- [Python Language Services](http://docs.python.org/lib/language.html) → <http://docs.python.org/lib/language.html>
 - [ast](http://docs.python.org/lib/ast.html) – Abstract Syntax Trees → <http://docs.python.org/lib/ast.html>
 - [symtable](http://docs.python.org/lib/symtable.html) – Access to the compiler’s symbol tables → <http://docs.python.org/lib/symtable.html>
 - [token](http://docs.python.org/lib/token.html) – Constants used with Python parse trees → <http://docs.python.org/lib/token.html>
 - [keyword](http://docs.python.org/lib/keyword.html) – Testing for Python keywords → <http://docs.python.org/lib/keyword.html>
 - [tokenize](http://docs.python.org/lib/tokenize.html) – Tokenizer for Python source → <http://docs.python.org/lib/tokenize.html>
 - [tabnanny](http://docs.python.org/lib/tabnanny.html) – Detection of ambiguous indentation → <http://docs.python.org/lib/tabnanny.html>
 - [pyclbr](http://docs.python.org/lib/pyclbr.html) – Python module browser support → <http://docs.python.org/lib/pyclbr.html>
 - [py_compile](http://docs.python.org/lib/py_compile.html) – Compile Python source files → http://docs.python.org/lib/py_compile.html
 - [compileall](http://docs.python.org/lib/compileall.html) – Byte-compile Python libraries → <http://docs.python.org/lib/compileall.html>
 - [dis](http://docs.python.org/lib/dis.html) – Disassembler for Python bytecode → <http://docs.python.org/lib/dis.html>
 - [pickletools](http://docs.python.org/lib/pickletools.html) – Tools for pickle developers → <http://docs.python.org/lib/pickletools.html>
- [MS Windows Specific Services](http://docs.python.org/lib/windows.html) → <http://docs.python.org/lib/windows.html>
 - [msvcrt](http://docs.python.org/lib/msvcrt.html) – Useful routines from the MS VC++ runtime → <http://docs.python.org/lib/msvcrt.html>
 - [winreg](http://docs.python.org/lib/winreg.html) – Windows registry access → <http://docs.python.org/lib/winreg.html>
 - [winsound](http://docs.python.org/lib/winsound.html) – Sound-playing interface for Windows → <http://docs.python.org/lib/winsound.html>
- [Unix Specific Services](http://docs.python.org/lib/unix.html) → <http://docs.python.org/lib/unix.html>
 - [posix](http://docs.python.org/lib/posix.html) – The most common POSIX system calls → <http://docs.python.org/lib/posix.html>
 - [pwd](http://docs.python.org/lib/pwd.html) – The password database → <http://docs.python.org/lib/pwd.html>
 - [grp](http://docs.python.org/lib/grp.html) – The group database → <http://docs.python.org/lib/grp.html>
 - [termios](http://docs.python.org/lib/termios.html) – POSIX style tty control → <http://docs.python.org/lib/termios.html>
 - [tty](http://docs.python.org/lib/tty.html) – Terminal control functions → <http://docs.python.org/lib/tty.html>
 - [pty](http://docs.python.org/lib/pty.html) – Pseudo-terminal utilities → <http://docs.python.org/lib/pty.html>
 - [fcntl](http://docs.python.org/lib/fcntl.html) – The fcntl and ioctl system calls → <http://docs.python.org/lib/fcntl.html>
 - [resource](http://docs.python.org/lib/resource.html) – Resource usage information → <http://docs.python.org/lib/resource.html>
 - [syslog](http://docs.python.org/lib/syslog.html) – Unix syslog library routines → <http://docs.python.org/lib/syslog.html>
- [Modules command-line interface \(CLI\)](http://docs.python.org/lib/cmdline.html) → <http://docs.python.org/lib/cmdline.html>
- [Superseded Modules](http://docs.python.org/lib/superseded.html) → <http://docs.python.org/lib/superseded.html>
 - [aifc](http://docs.python.org/lib/aifc.html) – Read and write AIFF and AIFC files → <http://docs.python.org/lib/aifc.html>
 - [audioop](http://docs.python.org/lib/audioop.html) – Manipulate raw audio data → <http://docs.python.org/lib/audioop.html>
 - [cgi](http://docs.python.org/lib/cgi.html) – Common Gateway Interface support → <http://docs.python.org/lib/cgi.html>
 - [cgitb](http://docs.python.org/lib/cgitb.html) – Traceback manager for CGI scripts → <http://docs.python.org/lib/cgitb.html>
 - [chunk](http://docs.python.org/lib/chunk.html) – Read IFF chunked data → <http://docs.python.org/lib/chunk.html>
 - [crypt](http://docs.python.org/lib/crypt.html) – Function to check Unix passwords → <http://docs.python.org/lib/crypt.html>
 - [imghdr](http://docs.python.org/lib/imghdr.html) – Determine the type of an image → <http://docs.python.org/lib/imghdr.html>
 - [mailcap](http://docs.python.org/lib/mailcap.html) – Mailcap file handling → <http://docs.python.org/lib/mailcap.html>
 - [msilib](http://docs.python.org/lib/msilib.html) – Read and write Microsoft Installer files → <http://docs.python.org/lib/msilib.html>
 - [nis](http://docs.python.org/lib/nis.html) – Interface to Sun’s NIS (Yellow Pages) → <http://docs.python.org/lib/nis.html>
 - [nntplib](http://docs.python.org/lib/nntplib.html) – NNTP protocol client → <http://docs.python.org/lib/nntplib.html>

- [optparse — Parser for command line options](http://docs.python.org//lib/optparse.html) → <http://docs.python.org//lib/optparse.html>
- [ossaudiodev — Access to OSS-compatible audio devices](http://docs.python.org//lib/ossaudiodev.html) → <http://docs.python.org//lib/ossaudiodev.html>
- [pipes — Interface to shell pipelines](http://docs.python.org//lib/pipes.html) → <http://docs.python.org//lib/pipes.html>
- [sndhdr — Determine type of sound file](http://docs.python.org//lib/sndhdr.html) → <http://docs.python.org//lib/sndhdr.html>
- [spwd — The shadow password database](http://docs.python.org//lib/spwd.html) → <http://docs.python.org//lib/spwd.html>
- [sunau — Read and write Sun AU files](http://docs.python.org//lib/sunau.html) → <http://docs.python.org//lib/sunau.html>
- [telnetlib — Telnet client](http://docs.python.org//lib/telnetlib.html) → <http://docs.python.org//lib/telnetlib.html>
- [uu — Encode and decode uuencode files](http://docs.python.org//lib/uu.html) → <http://docs.python.org//lib/uu.html>
- [xdrlib — Encode and decode XDR data](http://docs.python.org//lib/xdrlib.html) → <http://docs.python.org//lib/xdrlib.html>
- [Security Considerations](http://docs.python.org//lib/security_warnings.html) → http://docs.python.org//lib/security_warnings.html

The Python Language Reference

Source: <http://docs.python.org/ref/>

© Copyright → <http://docs.python.org/copyright.html> 2001-2024, Python Software Foundation.

This page is licensed under the Python Software Foundation License Version 2.

Examples, recipes, and other code in the documentation are additionally licensed under the Zero Clause BSD License.

See [History and License](#) → <http://docs.python.org/license.html> for more information.

The Python Software Foundation is a non-profit corporation. [Please donate](#), → <https://www.python.org/psf/donations/>

Last updated on Mar 11, 2024 (21:22 UTC). [Found a bug](#) → <http://docs.python.org/bugs.html>?

Created using [Sphinx](#) → <https://www.sphinx-doc.org/> 7.2.6.

PythonEditors - Python Wiki

Source: <https://wiki.python.org/moin/PythonEditors>

If you have anything to contribute -- e.g. configurations for editors, new editors, or opinion -- don't hesitate to edit or create pages.



Please keep wiki links as wiki links, use external links only if there is no existing page for the editor. Please add pages like [BoaConstructor](#) → <https://wiki.python.org/moin/BoaConstructor> also to page [IntegratedDevelopmentEnvironments](#) → <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>.

Name	Platform	Impl. Language	License	Notes
a8 → https://github.com/aliafshar/a8	Linux, FreeBSD	Python, GTK	GPLv3	Embed Vim. Little brother of PIDA → http://pida.co.uk/
Alphatk → http://www.purl.org/net/alphatk/about.html	Unix/X, Windows, Mac OS X	Tcl/Tk	Proprietary	Extensible in Tcl, Tk; Can interact with python.
Atom → https://atom.io/packages/language-python	Unix/X, Windows, Mac OS X	Python	MIT	Python language support for Atom-IDE, powered by the Python language server.
Code::Blocks → http://www.codeblocks.org/features	Linux, Mac OS X	C++, wxWidgets	GPLv3	class browser does not currently work for .py files, but it's still a nice IDE to use for python projects
Bluefish → http://bluefish.openoffice.nl/features.html	Linux, Mac OS X	C, GTK+	GPLv3	The link points to the features page.
Cream → http://cream.sourceforge.net/	Linux, FreeBSD	C	GPLv3	Cream is a free and easy-to-use configuration of the powerful and famous Vim → http://www.vim.org/ text editor for both Microsoft Windows and GNU/Linux.
Cssed → http://cssed.sourceforge.net/	Unix/X, Mac OS X	C, GTK+	GPLv2	CSS editor with syntax highlighting for Python, and embedded Python interpreter. Extensible through a Python API.
codeEditor → http://wiki.wxpython.org/PythonCardEditor	Unix/X, Mac OS X	Python, wxPython	BSD	Extensible in Python; part of PythonCard → http://pythoncard.sourceforge.net/ . Includes PyCrust → https://wiki.python.org/moin/PyCrust shell.
CodeLobster → http://www.codelobster.com/	Unix/X, Mac OS X	Python	Proprietary	Free version includes Python support.

eric → http://eric-ide.python-project.s.org/	Linux, Windows, Mac OS X	Python, PyQt → https://wiki.python.org/moin/PyQt	GPLv3	Complete IDE, very well integrated with PyQt development, but usable for any kind of project. Supports projects, debugging, auto-complete, syntax coloring, etc. It is extensible via plug-in system. Integrated version control interface for Git, Subversion and Mercurial through core plugins. eric6 requires Python 3 (and, if desired, PyQt5 → https://wiki.python.org/moin/PyQt5), and supports CxFreeze → https://wiki.python.org/moin/CxFreeze and PyInstaller → https://wiki.python.org/moin/PyInstaller. Django and Pyramid, PyLint → https://wiki.python.org/moin/PyLint and Vulture
CRIISP → http://www.crisp.demon.co.uk/	Unix/X, Windows, Mac OS X		Proprietary	BRIEF-compatible, supports Python syntax, in-buffer Python interpreter, supports lots of languages. Powerful macro language.
DRAKON_Editor → http://drakon-editor.sourceforge.net/	Windows, Mac OS X, Linux	Tcl/Tk	Public domain	DRAKON diagram editor with code generation in Python.
DreamPie → http://dreampie.sourceforge.net/	Windows, Linux and Mac/MacPorts	Python, PyGTK → http://macports.com/m/	GPLv3	Interactive shell with history box and code box, auto-completion of attributes and file names, auto-display of function arguments and documentation. Keeps your recent results, provides session history saving (optionally in HTML), interactive plotting with matplotlib. Extremely fast and responsive.
DrPython → http://https://wiki.python.org/moin/DrPython	Unix/X, Windows, Mac OS X	Python, wxPython	GPL	Simple, Highly Customizable Editor/Environment. A Tribute to DrScheme → https://wiki.python.org/moin/DrScheme.
Eclipse → http://www.eclipse.org/	Unix/X, Windows, Mac OS X	Java	EPL	Eclipse is ... an open extensible IDE for anything and nothing in particular." Support for Python can be obtained via the PyDEV → http://pydev.org// plugin.
EditPad Pro → http://www.editpadpro.com/editpadonline.html	Linux, Windows		Proprietary	Built-in Python syntax highlighting, Python class browsing, Python-compatible regular expressions, code folding, and extensive options for running external tools such as Python scripts.
Editra → http://www.editra.org/	Linux, Windows, Mac OS X	Python, wxPython	wxWindows	A general purpose developer's text editor written in Python/wxPython. It supports python syntax highlighting, auto-ident, auto-completion, classbrowser, and can run scripts from inside the editor. Extensible with plugins written in python.
EmacsEditor → https://wiki.python.org/moin/EmacsEditor	Unix/X, Windows, Mac OS	C, Lisp	GPLv3	Python support with EmacsPythonMode → https://wiki.python.org/moin/EmacsPythonMode. Extensible in Python using pymacs → https://github.com/pinard/pymacs

Epsilon	Linux, Windows, FreeBSD and OS/2	Proprietary	Customizable Python mode, syntax coloring, function tagging.	
ExCo	Linux, Windows, Mac OS	Python, PyQt4	Extensible editor written in Python, Python/C/Nim code tree browser, 3-window editing, text diff, multi-language support, Python REPL, manipulate editor text with Python code	
FTE Text Editor	Unix/X, Windows, DOS, OS/2	C++	GPL	Supports lots of languages, including Python; doesn't seem programmable
Geany	Unix/X, Windows	C, GTK+	GPLv2	A small and lightweight GTK+ IDE that supports lots of languages, including Python.
gedit	Unix/X, Windows, Mac OS	C, Python	GPLv2	gedit is the official text editor of the GNOME desktop environment, with Python syntax highlighting.
ideas	Unix/X, Windows, Mac OS	Python, PyQt	Proprietary	Ideas is a feature rich IDE that supports debugging, interpreting and project management.
J	Linux, Windows, Mac OS X	Java	GPLv2	syntax coloring for python, extensible with jython, supports many file formats, has folding, fully customisable, has sidebar for class and functions, fast for a Java application
jHepWork	Unix, Windows, Mac OS	Java	GPLv3	IDE with Jython Shell. Syntax coloring for python, extensible with jython, supports many file formats, fully customisable, has sidebar for class and functions
Jasspa's MicroEmacs	Unix, Windows	C	GPLv2	Supports Python syntax and a Python-specific menu.
JED	Unix, MSDOS, BeOS, QNX, and Windows.	VMS, C OS/2,	GPLv2	Syntax highlighting and indenting, (optional) emacs keybindings, programmable with s-lang. <i>Note: comment out "msw_help..." line in pymode.sl if you are having problems on Windows.</i>
jEdit	Unix, Windows, Mac OS	Java	GPLv2	Has three plugins - one for Jython and one for Python/Jython JpyDbg - interactive editing debugging, code browsing, highlighting.
JpyDbg	Linux, Mac OS	Java, Python	GPL	Netbeans jEdit cross IDE plugin

Jupp	Unix/OSX/Cygwin/Interix (curses), MS-DOS (DR DOS, FreeDOS, Win, OS/2 DOS-Box)	C	GPL	Versatile WordStar editor (JOE fork) with many coloured syntax-highlighting modes
kdevelop	Linux, Windows, Mac OS	C++	GPLv2	Kde main developing app. Code folding, syntax highlighting, navigator, projects, class browser, version control, customizable keybindings. Comes also with some typical python project models: pyton Qt app, Tkinter app and simple script.
Komodo IDE	Unix, Windows, Mac OS X	Proprietary		Komodo is an award winning Python IDE from ActiveState . Fully-integrated Python 3 support featuring code intelligence with autocomplete and calltips, Python debugger (includes remote debugging), interactive shell, remote file support, macros, templating, emacs command support and great help documentation. There is also an open-source version called Komodo Edit (Source), as part of the Open Komodo project started November 2007.
Komodo Edit	Unix/X, Windows, Mac OS X	Proprietary/Open Source		scaled-down version of Komodo IDE http://www.activestate.com/komodo-edit/compare-with-komodo-ide which also supports Python, but excludes the integrated debugger.
gEcrit	Unix/X, Windows, Mac OS X	Python, wxPython	GPLv3	Python IDE with focus is on simplicity and ease of use. It is fast and lightweight. It features Python indentation, line numbers, code folding, syntax highlighting, shell access, code completion, a program runner, a source browser, indentation guides, a white space indicator, autosaving, an edge line, multiple tabs, printing, jumping to a specific line, word searching, word replacement, zooming, undo/redo, pastebin.com code submission, Python syntax checking, the ability to change the indentation of many lines at once, autocompletion, and bad brace checking.
LeoEditor	Linux, Windows, Mac OS X	Python, PyQt	MIT	Outlining editor, fully scriptable and extensible, supporting literate programming . 100% pure Python code.

mmedit	→ http://m ooedit.sourceforge.net/	Unix, Windows	C, GTK+	LGPL	Gtk editor with python bindings, allows plugins written in python.
nano	→ https://w ww.nano-editor.or g/	Linux, Windows	C	GNU GPL	Small, terminal-based editor, Syntax highlighting
ne	→ http://ne.di unimi.it/	Unix, Windows, Mac OS X	C	GPLv3	Easy to use, small, powerful, fast, terminal-based editor. Supports UTF-8, syntax highlighting, undo, autocomplete, macros, regexes, bookmarks. v2.4 released 2012-04
NEdit	→ http://n edit.org/	Unix/X, Windows, Mac OS X	C	GPLv2	X-Based, Python support builtin.
Netbeans	→ ht tp://www.netbean s.org/	Unix/X, Windows, Mac OS X, others	Java	GPLv2	Netbeans is an open extensible cross platform IDE ; Support for Python can be obtained via the JpyDbg → http://jpydbg.sourceforge.net/ plugin. A new plugin exists for Netbeans 6.5 or newer - info → http://wiki.netbeans.org/Python .
NINJA-IDE	→ h tp://ninja-ide.org/	Linux, Windows, Mac OS	Python, PyQt	GPLv3	Lightweight and extensible editor. Class browser, project manager, PEP8 finder, virtualenv, plugin support
Ulipad	→ https:// github.com/limod ou/ulipad	Unix/X, Windows, Mac OS X	Python, wxPython	GPLv2	wxPython based editor. Can be easily extended with mixins and plugins, and has many features. Seems to be biased towards Python web development as contains FTP GUI, RSS aggregator and HTML preview.
PowerPad	→ ht tp://www.quickme diasolutions.com/ software/powerpa d/	Windows Linux (with 2)	NT, C++, wxWidgets	Freeware	Easy to use text editor with syntax highlighting for Python. Embeds the Python interpreter to extend functionality of the application
PyCharm	→ ht tp://jetbrains.com/ pycharm/	Linux, Windows, Mac OS X	Java	Apache 2.0	Full-featured IDE for Python. Has Free and Open Source edition fully supporting Python as well as proprietary Professional Edition with Django, Flask, Pyramid and Google App Engine support.
PyDev	→ http://p ydev.org/	Linux, Mac OS X	Windows, Java, Eclipse	EPL	Eclipse → http://www.eclipse.org/ plugin. Code-completion. Debugger. Under active development.

Pye	Linux, Windows	Python, Tk	GPLv3	Ultra-lightweight Python/text editor made in Python with Tk. Emphasis on easy customization and no-bloat attitude.
Pyedit	PyBoard	Python	MIT	Written in Python - simple small editor, suitable for MicroPython/circuitPython/Pycopy.
PyEdit	PyBoard	Unix, Windows, Mac OS X	X	
PyPE	Unix, Windows, Mac OS X	Python, wxPython	GPLv2	Written in Python - code folding, snippets, unicode, multiple documents, code completion, several languages, macros.
Python for VS Code	Linux, Windows, Mac OS X	Node.js	MIT	Free open-source - https://github.com/microsoft/vscode-python extension for Visual Studio Code. Supports syntax highlighting, debugging, code completion, code navigation, refactoring, with support for Django, multi threaded, local and remote debugging.
Python Toolkit (PTK)	Windows/Linux/Mac	Python	GPLv3	An interactive environment for python features include: Multiple independent python interpreters. Interactively program with different GUI toolkits (wxPython, TkInter - https://wiki.python.org/moin/Tkinter , pyGTK, pyQT4 and PySide - https://wiki.python.org/moin/PySide). Matlab style namespace/workspace browser. Object auto-completions, calltips and multi-line command editing in the console. Object inspection and python path management. Simple code editor and integrated debugger.
Pyzo	Unix/X, Windows, Mac OS X	Python, PySide	BSD License	Open-source Python IDE focused on interactivity and introspection, which makes it very suitable for scientific computing. Its practical design is aimed at simplicity and efficiency. Pyzo consists of two main components, the editor and the shell, and uses a set of pluggable tools to help the programmer in various ways: e.g. source structure, interactive help, workspace, file browser (with functionality for searching). Also includes a post-mortem debugger.

SciTE	Windows, Linux C++ (GTK+)	MIT-Like	A highly configurable light-weight source code editor (about 50 file formats) based on widely-used Scintilla rich text widget/control. Features: syntax highlighting, code folding, auto-indent, brace matching, code-page/unicode support, multiple documents, context help, code execution, output pane, external tools support, RegExp → https://wiki.python.org/moin/RegExp find/replace, text export (HTML, RTF, PDF, LaTeX, XML). Python API → http://scintilla.sourceforge.net/gen_python_api.zip for calltips and autocompletion available.	
SPE → http://wiki.python.org/moin/SPE	Windows, Linux, Mac OS X	Python, wxPython	GPLv3	Stani's Python Editor. Auto indentation, auto completion, call tips, syntax coloring/highlighting, UML viewer, class explorer, source index, auto todo list, sticky notes, integrated PyCrust → https://wiki.python.org/moin/PyCrust shell, Python file browser, recent file browser, drag&drop, context help. Blender → http://www.blender3d.org/ support with a Blender 3D object browser, runs interactively inside Blender. Ships with WxGlade → https://wiki.python.org/moin/WxGlade (GUI designer), PyChecker → https://wiki.python.org/moin/PyChecker (source code doctor) and Kiki → http://project5.fr/eezope.org/kiki/ (regular expression console). Extensible with WxGlade → https://wiki.python.org/moin/WxGlade .
Spyder → http://www.spyder-id.e.org/	Windows, macOS	Python, PyQt5	MIT	Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection and beautiful visualization capabilities of a scientific package. Furthermore, Spyder offers built-in integration with many popular scientific packages, including NumPy → https://wiki.python.org/moin/NumPy , SciPy → https://wiki.python.org/moin/SciPy , Pandas, IPython, QtConsole → https://wiki.python.org/moin/OtConsole , Matplotlib, SymPy → https://wiki.python.org/moin/SymPy , and more. Beyond its many built-in features, Spyder's abilities can be extended even further via first- and third-party plugins. It is conveniently integrated in the cross-platform Anaconda distribution → https://www.anaconda.com/ , and is the centerpiece of the Python(x,y) → http://www.pythony.com/ and WinPython → https://winpython.github.io/ distributions for Windows.

SlickEdit → http://www.slickedit.com/	Unix/X, Windows, Mac OS X	Proprietary	Syntax coloring, popup function arguments, class hierarchy browser, graphical debugger, and other nice Python features. Context tagging with class/module namespaces, code navigation, and smart indenting are also supported.	
Sublime Text → http://www.sublime-text.com/	Linux, Windows, Mac OS X	C++, Python	Proprietary	Beautiful interface, Python syntax highlighting, Python plugins.
Thonny → http://thonny.org/	Linux, Windows, Mac OS X	Python	MIT	For teaching/learning programming. Focused on program runtime visualization. Provides stepping both in statements and expressions, no-hassle variables view, separate mode for explaining references etc.
Vim → https://vi.python.org/moin/vim	Unix/X, Windows, Mac OS	C	Charity-ware	Highly configurable text editor built to enable efficient text editing. Syntax coloring, indenting, auto-completion, and source-navigation tools for Python. Can be scripted in Python. See also Cream → http://cream.sourceforge.net/ .
Wing IDE → http://wingware.com/wingide	Unix/X, Windows, Mac OS X	Python, wxPython	Proprietary	Powerful commercial IDE designed specifically for Python. Auto-completion, call tips, syntax highlighting, goto-definition, keyboard modes for emulating Visual Studio, VI/Vim, Emacs, & Brief, graphical debugger, code browser, integrated shell, scriptable in Python, and much more.
wxKonTEXT → http://soft.kawor.u.it/wxKonTEXTe.htm	Linux, Windows	Python, wxPython	GPL	Simple text editor written in python. Syntax highlighter, Code fold, Export code in HTML...

"IDEs" that don't integrate anything Python-specific go here.

"IDEs" that run on mobile devices and tablets.

IntegratedDevelopmentEnvironments - Python Wiki

Source: <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>



Please keep wiki links as wiki links, use external links only if there is no existing page for the IDE.

See also Wikipedia's [list of Python IDEs](http://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments#Python) → http://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments#Python and [ShowMeDo](https://wiki.python.org/moin>ShowMeDo) → <https://wiki.python.org/moin>ShowMeDo> videos for Wing 3 Professional, Python Development With SPE, Eclipse PyDev → <https://wiki.python.org/moin/PyDev> and IPython (see site page for updated information).

Name	Platform	Entry Updated	Notes
Thonny → http://thonny.org/	Windows, Linux, Mac OS X, more	2023	For teaching/learning programming. Focused on program runtime visualization. Provides stepping both in statements and expressions, no-hassle variables view, separate mode for explaining references etc.
Komodo → http://www.activestate.com/products/komodo-ide/	Windows/Linux/Mac OS X	2023	Multi-language IDE with support for Python 2.x and Python 3. Available as https://www.activestate.com/products/komodo-ide/ now open source only; last updated 2022
CodeLobster IDE → http://www.codelobster.com/	Windows/Linux/Mac OS X	2023	Multi-language IDE with free support for Python: code completion, navigation and highlighting etc.
LiClipse → http://www.liclips.com/	Linux/Mac OS X/Windows	2023	Commercial Eclipse-based IDE which provides a standalone bundling PyDev → http://pydev.org/ , Workspace Mechanic, Eclipse Color Theme, StartExplorer and AnyEdit, along with lightweight support for other languages, and other usability enhancements (such as multi-caret-edition).
NetBeans → http://wiki.python.org/moin/NetBeans	Linux, Mac, Solaris, Windows	2023	Python/Jython support in NetBeans → https://wiki.python.org/moin/NetBeans → Open source, allows Python and Jython Editing, code-completion, debugger, refactoring, templates, syntax analysis, etc. Note: the Python plugin as a community-supported project, and may trail behind. Currently it works for 8.1, does not appear to be available for 8.2
PyCharm → http://www.jetbrains.com/pycharm/	Linux/Mac OS X/Windows	2023	The <i>Community</i> edition is a free IDE with a smart Python editor providing quick code navigation, code completion, refactoring, unit testing and debugger. The commercial <i>Professional</i> edition fully supports Web development with Django, Flask, Mako and Web2Py → https://wiki.python.org/moin/Web2Py and allows to develop remotely. JetBrains offers free PyCharm Professional licenses for open-source projects under certain conditions https://www.jetbrains.com/buy/opensource/ . There is also free access for Student/Educational use.

Python for VS Code	Linux/Mac X/Windows	OS	2023	Free open-source → https://github.com/Microsoft/vscode-python extension for Visual Studio Code (now maintained by Microsoft). Supports syntax highlighting, debugging, code completion, code navigation, unit testing, refactoring, with support for Django, multi threaded, local and remote debugging.
KDevelop	Linux/Mac X/(Windows)	OS	2023	Free open-source IDE with a focus on static analysis-based code completion, navigation and highlighting. Also features a VI emulation mode.
PyDev	Eclipse		2023	Free, open-source plugin for Eclipse → http://www.eclipse.org/ -- Allows Python, Jython, and IronPython → https://wiki.python.org/moin/IronPython editing, code-completion, debugger, refactoring, quick navigation, templates, code analysis, unittest integration, Django integration, etc.
Wing	Windows, Mac OS X	Linux,	2023	Family of Python IDEs with advanced debugger, editor with vi, emacs, visual studio and other key bindings, auto-completion, auto-editing, import management, multi-selection, inline code warnings, snippets, goto-definition, find uses, refactoring, unit testing with code coverage, remote development, support for containers and clusters, array and dataframe viewer, bookmarking, project management with version control, Python environment creation with virtualenv, pipenv, conda, and Docker, Python package management with pip, pipenv, and conda, source browser, PEP 8 / Black / YAPF reformatting, and much more. Product levels, include free and paid versions with a fully functional trial and free licenses for educational use and unpaid open source developers. Documentation includes help for using Wing with Django, Flask, Docker, AWS, Vagrant, Matplotlib, Jupyter, Blender, Maya, any many other third party tools and packages. See product comparison → http://wingware.com/downloads and pricing → http://wingware.com/store/purchase for details.
PyScripter	Windows		2023	MIT licensed IDE written in Delphi with debugger, integrated unit testing, source browser, code navigation and syntax coloring/autocomplete editor.

Spyder → http://www.spyder-ide.org/	Windows/Linux/macOS	2023	A powerful, free/open-source scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. Features a unique combination of the advanced editing, analysis, debugging and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection and beautiful visualization capabilities of a scientific package. Furthermore, offers built-in integration with many popular scientific packages, including NumPy → https://wiki.python.org/moin/NumPy , SciPy → https://wiki.python.org/moin/SciPy , Pandas, IPython, QtConsole → https://wiki.python.org/moin/QtConsole , Matplotlib, SymPy → https://wiki.python.org/moin/SymPy , and more, and can be easily extended with plugins. It is conveniently integrated in the cross-platform Anaconda distribution → https://www.anaconda.com/ , and is the centerpiece of the Python(x,y) → https://python-xy.github.io/ and WinPython → https://winpython.github.io/ distributions.
IDLE	Windows/Linux/Mac OS X/All Tk Platforms	2023	Multi-window colorized source browser, autoindent, autocompletion, tool tips, code context panel, search in files, class and path browsers, debugger, executes code in clean separate subprocess with one key-stroke. 100% pure Python, part of Python 2.x and 3.x distributions (may be packaged separately in some situations).
IdleX → http://idlex.sourceforge.net/	Windows/Linux/Mac OS X/All Tk Platforms	2023	IdleX is a collection of over twenty extensions and plugins that provide additional functionality to IDLE, a Python IDE provided in the standard library. It transforms IDLE into a more useful tool for academic research and development as well as exploratory programming. Last updated 2022.
μ.dev → http://sakurastudio.yo-lasite.com/micro-dev.php	Windows Vista and XP	2023	An open-source IDE, created using Lazarus. It's only for Python. include syntax highlighting, project manager, and uses pdb for debugging. Development stopped in 2011.
Pyzo (formerly IEP) → http://www.pyzo.org/	Windows/Linux/Mac OS X	2023	Open-source Python IDE focused on interactivity and introspection, which makes it very suitable for scientific computing. Its practical design is aimed at simplicity and efficiency. Pyzo consists of two main components, the editor and the shell, and uses a set of pluggable tools to help the programmer in various ways: e.g. source structure, interactive help, workspace, file browser (with functionality for searching). Also includes a post-mortem debugger.

Python Toolkit	Win- (PTK) → http://pythontoolkit.org/moin/Komodo	2023	An interactive environment for python built around a matlab style console window and editor. It was designed to provide a python based environment similar to Matlab for scientists and engineers however it can also be used as a general purpose interactive python environment especially for interactive GUI programming. Features include: Multiple independent python interpreters. Interactively program with different GUI toolkits (wxPython, TkInter → https://wiki.python.org/moin/Tkinter , pyGTK, pyQT4 and PySide → https://wiki.python.org/moin/PySide). Matlab style namespace/workspace browser. Object auto-completions, calltips and multi-line command editing in the console. Object inspection and python path management. Simple code editor and integrated debugger. Last released in 2014.
Python Tools for Visual Studio	Windows	2023	Open-source plugin for Visual Studio 2010, 2012 onwards (now maintained by Microsoft). Supports syntax highlighting, debugging and rich intellisense, unit testing, refactoring, object browser, MPI cluster debugging, Django intellisense and debugging, development REPL window and a debugging REPL window. Supports mixed-mode Python/C/C++ debugging.
Exedore	Mac OS X	2015	Commercial with feature-limited free trial. A Mac-native, single-window IDE inspired by Xcode. Features integrated debugger, tabs, code completion with tab triggers, syntax highlighting themes, search and replace with regex, integrated REPL sessions, goto definition, file browser, integrated documentation browser. As of June 2015, does not support input() meaning any console input using this function is not supported.
Name	Platform	Updated	Notes
Komodo	Windows/Mac/Linux	2012	Komodo Edit → http://www.activestate.com/komodo-edit (open source, as part of the Open Komodo → http://www.openkomodo.com/ project). Little brother to Komodo IDE.
BlackAdder	Windows/Linux	2004	Commercial; integrated debugger; interfaces with Qt Designer
eric	Python + PyQt	2018	Open Source, interfaces with Qt Designer, Qt Linguist, unittest; integrated debugger
SPE	Windows, MacOsX, more	2008	Open-source with wxPython → http://www.wxpython.org/ interface. Code completion, call tips, class explorer, source index, auto todo list, Blender → http://www.blender.org/ support, integrated PyChecker → http://pychecker.sourceforge.net/ (source code doctor) and Kiki → http://project5.freezone.org/kiki (regex console). Download instructions → http://pythonide.blogspot.com/2007/02/how-to-download-latest-spe-from_26.html

Pida	→ http://pid.a.co.uk/	Linux, FreeBSD, ..., 2007 (2008 dev)	(Windows in progress)	Open-source with GTK interface, written in Python. Supports different languages, python trough rope	→ http://rope.sourceforge.net/ and pyflakes as well as rpdb2. Support different Editors (Vim, Medit, Emacs) Current Repos → http://pida.co.uk/trac/wiki/DeveloperRepos
SharpDevelop	.net CLR	26/7/2009		FOSS IDE uses IronPython	→ https://wiki.python.org/moin/IronPython to support making python module solutions.
NINJA-IDE	→ h tp://ninja-ide.org/	Python + PyQt	→ https://wiki.python.org/moin/PyQt	2011 + (Linux/Windows/Mac OS X)	NINJA-IDE (from: "Ninja Is Not Just Another IDE"), is a cross-platform integrated development environment specially design to build Python Applications.
Aptana Studio 3	Linux, Windows and Mac OS X	10/01/2012		Aptana Studio3 is a professional, open source development tool for the open web	
Pcode	→ https://github.com/fortharries/Pcode	Windows, Linux and Mac OSX	2014	Python 3x IDE with emphasis on power, usability and simplicity.	

IDLE

Source: <https://docs.python.org/3/library/idle.html>

Source code: [Lib/idlelib/ → https://github.com/python/cpython/tree/3.12/Lib/idlelib/](https://github.com/python/cpython/tree/3.12/Lib/idlelib/)

IDLE is Python's Integrated Development and Learning Environment.

IDLE has the following features:

- cross-platform: works mostly the same on Windows, Unix, and macOS
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features
- search within any window, replace within editor windows, and search through multiple files (grep)
- debugger with persistent breakpoints, stepping, and viewing of global and local namespaces
- configuration, browsers, and other dialogs

Editing and Navigation¶

Editor windows¶

IDLE may open editor windows when it starts, depending on settings and how you start IDLE. Thereafter, use the File menu. There can be only one open editor window for a given file.

The title bar contains the name of the file, the full path, and the version of Python and IDLE running the window. The status bar contains the line number ('Ln') and column number ('Col'). Line numbers start with 1; column numbers with 0.

IDLE assumes that files with a known .py* extension contain Python code and that other files do not. Run Python code with the Run menu.

Key bindings¶

The IDLE insertion cursor is a thin vertical bar between character positions. When characters are entered, the insertion cursor and everything to its right moves right one character and the new character is entered in the new space.

Several non-character keys move the cursor and possibly delete characters. Deletion does not put text on the clipboard, but IDLE has an undo list. Wherever this doc discusses keys, 'C' refers to the `Control` key on Windows and Unix and the `Command` key on macOS. (And all such discussions assume that the keys have not been re-bound to something else.)

- Arrow keys move the cursor one character or line.
- `[C]-[LeftArrow]` and `[C]-[RightArrow]` moves left or right one word.
- `Home` and `End` go to the beginning or end of the line.
- `Page Up` and `Page Down` go up or down one screen.
- `[C]-[Home]` and `[C]-[End]` go to beginning or end of the file.
- `Backspace` and `Del` (or `[C]-[d]`) delete the previous or next character.
- `[C]-[Backspace]` and `[C]-[Del]` delete one word left or right.
- `[C]-[k]` deletes ('kills') everything to the right.

Standard keybindings (like **[C]-[c]**) to copy and **[C]-[v]** to paste) may work. Keybindings are selected in the Configure IDLE dialog.

Automatic indentation

After a block-opening statement, the next line is indented by 4 spaces (in the Python Shell window by one tab). After certain keywords (break, return etc.) the next line is dedented. In leading indentation, **Backspace** deletes up to 4 spaces if they are there. **Tab** inserts spaces (in the Python Shell window one tab), number depends on Indent width. Currently, tabs are restricted to four spaces due to Tcl/Tk limitations.

See also the indent/dedent region commands on the [Format menu](#).

Search and Replace

Any selection becomes a search target. However, only selections within a line work because searches are only performed within lines with the terminal newline removed. If [x] Regular expression is checked, the target is interpreted according to the Python re module.

Completions

Completions are supplied, when requested and available, for module names, attributes of classes or functions, or filenames. Each request method displays a completion box with existing names. (See tab completions below for an exception.) For any box, change the name being completed and the item highlighted in the box by typing and deleting characters; by hitting **Up**, **Down**, **PageUp**, **PageDown**, **Home**, and **End** keys; and by a single click within the box. Close the box with **Escape**, **Enter**, and double **Tab** keys or clicks outside the box. A double click within the box selects and closes.

One way to open a box is to type a key character and wait for a predefined interval. This defaults to 2 seconds; customize it in the settings dialog. (To prevent auto popups, set the delay to a large number of milliseconds, such as 100000000.) For imported module names or class or function attributes, type ‘.’. For filenames in the root directory, type `os.sep` → <https://docs.python.org/3/library/os.html#os.sep> or `os.altsep` → <https://docs.python.org/3/library/os.html#os.altsep> immediately after an opening quote. (On Windows, one can specify a drive first.) Move into subdirectories by typing a directory name and a separator.

Instead of waiting, or after a box is closed, open a completion box immediately with Show Completions on the Edit menu. The default hot key is **[C]-[space]**. If one types a prefix for the desired name before opening the box, the first match or near miss is made visible. The result is the same as if one enters a prefix after the box is displayed. Show Completions after a quote completes filenames in the current directory instead of a root directory.

Hitting **Tab** after a prefix usually has the same effect as Show Completions. (With no prefix, it indents.) However, if there is only one match to the prefix, that match is immediately added to the editor text without opening a box.

Invoking ‘Show Completions’, or hitting **Tab** after a prefix, outside of a string and without a preceding ‘.’ opens a box with keywords, builtin names, and available module-level names.

When editing code in an editor (as oppose to Shell), increase the available module-level names by running your code and not restarting the Shell thereafter. This is especially useful after adding imports at the top of a file. This also increases possible attribute completions.

Completion boxes initially exclude names beginning with ‘_’ or, for modules, not included in ‘`__all__`’. The hidden names can be accessed by typing ‘_’ after ‘.’, either before or after the box is opened.

Calltips¶

A calltip is shown automatically when one types `(` after the name of an *accessible* function. A function name expression may include dots and subscripts. A calltip remains until it is clicked, the cursor is moved out of the argument area, or `)` is typed. Whenever the cursor is in the argument part of a definition, select Edit and “Show Call Tip” on the menu or enter its shortcut to display a calltip.

The calltip consists of the function’s signature and docstring up to the latter’s first blank line or the fifth non-blank line. (Some builtin functions lack an accessible signature.) A ‘/’ or ‘*’ in the signature indicates that the preceding or following arguments are passed by position or name (keyword) only. Details are subject to change.

In Shell, the accessible functions depends on what modules have been imported into the user process, including those imported by Idle itself, and which definitions have been run, all since the last restart.

For example, restart the Shell and enter `itertools.count()`. A calltip appears because Idle imports itertools into the user process for its own use. (This could change.) Enter `turtle.write()` and nothing appears. Idle does not itself import turtle. The menu entry and shortcut also do nothing. Enter `import turtle`. Thereafter, `turtle.write()` will display a calltip.

In an editor, import statements have no effect until one runs the file. One might want to run a file after writing import statements, after adding function definitions, or after opening an existing file.

Code Context¶

Within an editor window containing Python code, code context can be toggled in order to show or hide a pane at the top of the window. When shown, this pane freezes the opening lines for block code, such as those beginning with `class`, `def`, or `if` keywords, that would have otherwise scrolled out of view. The size of the pane will be expanded and contracted as needed to show the all current levels of context, up to the maximum number of lines defined in the Configure IDLE dialog (which defaults to 15). If there are no current context lines and the feature is toggled on, a single blank line will display. Clicking on a line in the context pane will move that line to the top of the editor.

The text and background colors for the context pane can be configured under the Highlights tab in the Configure IDLE dialog.

Shell window¶

In IDLE’s Shell, enter, edit, and recall complete statements. (Most consoles and terminals only work with a single physical line at a time).

Submit a single-line statement for execution by hitting `Return` with the cursor anywhere on the line. If a line is extended with Backslash (`\`), the cursor must be on the last physical line. Submit a multi-line compound statement by entering a blank line after the statement.

When one pastes code into Shell, it is not compiled and possibly executed until one hits `Return`, as specified above. One may edit pasted code first. If one pastes more than one statement into Shell, the result will be a `SyntaxError` → <https://docs.python.org/3/library/exceptions.html#SyntaxError> when multiple statements are compiled as if they were one.

Lines containing `RESTART` mean that the user execution process has been re-started. This occurs when the user execution process has crashed, when one requests a restart on the Shell menu, or when one runs code in an editor window.

The editing features described in previous subsections work when entering code interactively. IDLE’s Shell window also responds to the following:

- `(C-c)` attempts to interrupt statement execution (but may fail).
- `(C-d)` closes Shell if typed at a `>>>` prompt.

- (**[Alt]-[p]**) and (**[Alt]-[n]**) (**[C]-[p]**) and (**[C]-[n]**) on macOS) retrieve to the current prompt the previous or next previously entered statement that matches anything already typed.
- **Return** while the cursor is on any previous statement appends the latter to anything already typed at the prompt.

Text colors¶

Idle defaults to black on white text, but colors text with special meanings. For the shell, these are shell output, shell error, user output, and user error. For Python code, at the shell prompt or in an editor, these are keywords, builtin class and function names, names following `class` and `def`, strings, and comments. For any text window, these are the cursor (when present), found text (when possible), and selected text.

IDLE also highlights the `soft keywords` → https://docs.python.org/3/reference/lexical_analysis.html#soft-keywords `match` → https://docs.python.org/3/reference/compound_stmts.html#match, `case` → https://docs.python.org/3/reference/compound_stmts.html#match, and `wildcard patterns` → https://docs.python.org/3/reference/compound_stmts.html#wildcard-patterns in pattern-matching statements. However, this highlighting is not perfect and will be incorrect in some rare cases, including some `_s` in case patterns.

Text coloring is done in the background, so uncolorized text is occasionally visible. To change the color scheme, use the Configure IDLE dialog Highlighting tab. The marking of debugger breakpoint lines in the editor and text in popups and dialogs is not user-configurable.

Startup and Code Execution¶

Upon startup with the `-s` option, IDLE will execute the file referenced by the environment variables `IDLESTARTUP` or `PYTHONSTARTUP` → <https://docs.python.org/3/using/cmdline.html#envvar-PYTHONSTARTUP>. IDLE first checks for `IDLESTARTUP`; if `IDLESTARTUP` is present the file referenced is run. If `IDLESTARTUP` is not present, IDLE checks for `PYTHONSTARTUP`. Files referenced by these environment variables are convenient places to store functions that are used frequently from the IDLE shell, or for executing import statements to import common modules.

In addition, Tk also loads a startup file if it is present. Note that the Tk file is loaded unconditionally. This additional file is `.Idle.py` and is looked for in the user's home directory. Statements in this file will be executed in the Tk namespace, so this file is not useful for importing functions to be used from IDLE's Python shell.

Command line usage¶

```
idle.py [-c command] [-d] [-e] [-h] [-i] [-r file] [-s] [-t title] [-] [arg] ...

-c command    run command in the shell window
-d             enable debugger and open shell window
-e             open editor window
-h             print help message with legal combinations and exit
-i             open shell window
-r file       run file in shell window
-s             run $IDLESTARTUP or $PYTHONSTARTUP first, in shell window
-t title      set title of shell window
-             run stdin in shell (- must be last option before args)
```

If there are arguments:

- If `-`, `-c`, or `r` is used, all arguments are placed in `sys.argv[1:...]` and `sys.argv[0]` is set to `' '`, `'-c'`, or `'-r'`. No editor window is opened, even if that is the default set in the Options dialog.
- Otherwise, arguments are files opened for editing and `sys.argv` reflects the arguments passed to IDLE itself.

Startup failure¶

IDLE uses a socket to communicate between the IDLE GUI process and the user code execution process. A connection must be established whenever the Shell starts or restarts. (The latter is indicated by a divider line that says ‘RESTART’). If the user process fails to connect to the GUI process, it usually displays a Tk error box with a ‘cannot connect’ message that directs the user here. It then exits.

One specific connection failure on Unix systems results from misconfigured masquerading rules somewhere in a system’s network setup. When IDLE is started from a terminal, one will see a message starting with `** Invalid host:`. The valid value is `127.0.0.1 (idlelib.rpc.LOCALHOST)`. One can diagnose with `tcpconnect -irv 127.0.0.1 6543` in one terminal window and `tcplisten <same args>` in another.

A common cause of failure is a user-written file with the same name as a standard library module, such as `random.py` and `tkinter.py`. When such a file is located in the same directory as a file that is about to be run, IDLE cannot import the stdlib file. The current fix is to rename the user file.

Though less common than in the past, an antivirus or firewall program may stop the connection. If the program cannot be taught to allow the connection, then it must be turned off for IDLE to work. It is safe to allow this internal connection because no data is visible on external ports. A similar problem is a network misconfiguration that blocks connections.

Python installation issues occasionally stop IDLE: multiple versions can clash, or a single installation might need admin access. If one undo the clash, or cannot or does not want to run as admin, it might be easiest to completely remove Python and start over.

A zombie `pythonw.exe` process could be a problem. On Windows, use Task Manager to check for one and stop it if there is. Sometimes a restart initiated by a program crash or Keyboard Interrupt (control-C) may fail to connect. Dismissing the error box or using Restart Shell on the Shell menu may fix a temporary problem.

When IDLE first starts, it attempts to read user configuration files in `~/.idlerc` (`~` is one’s home directory). If there is a problem, an error message should be displayed. Leaving aside random disk glitches, this can be prevented by never editing the files by hand. Instead, use the configuration dialog, under Options. Once there is an error in a user configuration file, the best solution may be to delete it and start over with the settings dialog.

If IDLE quits with no message, and it was not started from a console, try starting it from a console or terminal (`python -m idlelib`) and see if this results in an error message.

On Unix-based systems with tcl/tk older than 8.6.11 (see `About IDLE`) certain characters of certain fonts can cause a tk failure with a message to the terminal. This can happen either if one starts IDLE to edit a file with such a character or later when entering such a character. If one cannot upgrade tcl/tk, then re-configure IDLE to use a font that works better.

Running user code¶

With rare exceptions, the result of executing Python code with IDLE is intended to be the same as executing the same code by the default method, directly with Python in a text-mode system console or terminal window. However, the different interface and operation occasionally affect visible results. For instance, `sys.modules` starts with more entries, and `threading.active_count()` returns 2 instead of 1.

By default, IDLE runs user code in a separate OS process rather than in the user interface process that runs the shell and editor. In the execution process, it replaces `sys.stdin`, `sys.stdout`, and `sys.stderr` with objects that get input from and send output to the Shell window. The original values stored in `sys.__stdin__`, `sys.__stdout__`, and `sys.__stderr__` are not touched, but may be `None`.

Sending print output from one process to a text widget in another is slower than printing to a system terminal in the same process. This has the most effect when printing multiple arguments, as the string for each argument, each separator, the newline are sent separately. For development, this is usually not a problem, but if one wants to print faster in IDLE, format and join together everything one wants displayed together and then print a single string. Both format strings and `str.join()` – <https://docs.python.org/3/library/stdtypes.html#str.join> can help combine fields and lines.

IDLE's standard stream replacements are not inherited by subprocesses created in the execution process, whether directly by user code or by modules such as multiprocessing. If such subprocess use `input` from `sys.stdin` or `print` or `write` to `sys.stdout` or `sys.stderr`, IDLE should be started in a command line window. (On Windows, use `python` or `py` rather than `pythonw` or `pw`.) The secondary subprocess will then be attached to that window for input and output.

If `sys` is reset by user code, such as with `importlib.reload(sys)`, IDLE's changes are lost and input from the keyboard and output to the screen will not work correctly.

When Shell has the focus, it controls the keyboard and screen. This is normally transparent, but functions that directly access the keyboard and screen will not work. These include system-specific functions that determine whether a key has been pressed and if so, which.

The IDLE code running in the execution process adds frames to the call stack that would not be there otherwise. IDLE wraps `sys.getrecursionlimit` and `sys.setrecursionlimit` to reduce the effect of the additional stack frames.

When user code raises `SystemExit` either directly or by calling `sys.exit`, IDLE returns to a Shell prompt instead of exiting.

User output in Shell¶

When a program outputs text, the result is determined by the corresponding output device. When IDLE executes user code, `sys.stdout` and `sys.stderr` are connected to the display area of IDLE's Shell. Some of its features are inherited from the underlying Tk Text widget. Others are programmed additions. Where it matters, Shell is designed for development rather than production runs.

For instance, Shell never throws away output. A program that sends unlimited output to Shell will eventually fill memory, resulting in a memory error. In contrast, some system text windows only keep the last n lines of output. A Windows console, for instance, keeps a user-settable 1 to 9999 lines, with 300 the default.

A Tk Text widget, and hence IDLE's Shell, displays characters (codepoints) in the BMP (Basic Multilingual Plane) subset of Unicode. Which characters are displayed with a proper glyph and which with a replacement box depends on the operating system and installed fonts. Tab characters cause the following text to begin after the next tab stop. (They occur every 8 ‘characters’). Newline characters cause following text to appear on a new line. Other control characters are ignored or displayed as a space, box, or something else, depending on the operating system and font. (Moving the text cursor through such output with arrow keys may exhibit some surprising spacing behavior.)

```
>>> s = 'a\tb\ab<\x02><\r>\bc\nd' # Enter 22 chars.
>>> len(s)
14
>>> s # Display repr(s)
'a\tb\ab<\x02><\r>\x08c\nd'
>>> print(s, end='') # Display s as is.
# Result varies by OS and font. Try it.
```

The `repr` function is used for interactive echo of expression values. It returns an altered version of the input string in which control codes, some BMP codepoints, and all non-BMP codepoints are replaced with escape codes. As demonstrated above, it allows one to identify the characters in a string, regardless of how they are displayed.

Normal and error output are generally kept separate (on separate lines) from code input and each other. They each get different highlight colors.

For SyntaxError tracebacks, the normal '^' marking where the error was detected is replaced by coloring the text with an error highlight. When code run from a file causes other exceptions, one may right click on a traceback line to jump to the corresponding line in an IDLE editor. The file will be opened if necessary.

Shell has a special facility for squeezing output lines down to a 'Squeezed text' label. This is done automatically for output over N lines (N = 50 by default). N can be changed in the PyShell section of the General page of the Settings dialog. Output with fewer lines can be squeezed by right clicking on the output. This can be useful lines long enough to slow down scrolling.

Squeezed output is expanded in place by double-clicking the label. It can also be sent to the clipboard or a separate view window by right-clicking the label.

Developing tkinter applications¶

IDLE is intentionally different from standard Python in order to facilitate development of tkinter programs. Enter `import tkinter as tk; root = tk.Tk()` in standard Python and nothing appears. Enter the same in IDLE and a tk window appears. In standard Python, one must also enter `root.update()` to see the window. IDLE does the equivalent in the background, about 20 times a second, which is about every 50 milliseconds. Next enter `b = tk.Button(root, text='button');` `b.pack()`. Again, nothing visibly changes in standard Python until one enters `root.update()`.

Most tkinter programs run `root.mainloop()`, which usually does not return until the tk app is destroyed. If the program is run with `python -i` or from an IDLE editor, a >>> shell prompt does not appear until `mainloop()` returns, at which time there is nothing left to interact with.

When running a tkinter program from an IDLE editor, one can comment out the `mainloop` call. One then gets a shell prompt immediately and can interact with the live application. One just has to remember to re-enable the `mainloop` call when running in standard Python.

Running without a subprocess¶

By default, IDLE executes user code in a separate subprocess via a socket, which uses the internal loopback interface. This connection is not externally visible and no data is sent to or received from the internet. If firewall software complains anyway, you can ignore it.

If the attempt to make the socket connection fails, Idle will notify you. Such failures are sometimes transient, but if persistent, the problem may be either a firewall blocking the connection or misconfiguration of a particular system. Until the problem is fixed, one can run Idle with the `-n` command line switch.

If IDLE is started with the `-n` command line switch it will run in a single process and will not create the subprocess which runs the RPC Python execution server. This can be useful if Python cannot create the subprocess or the RPC socket interface on your platform. However, in this mode user code is not isolated from IDLE itself. Also, the environment is not restarted when Run/Run Module (F5) is selected. If your code has been modified, you must `reload()` the affected modules and re-import any specific items (e.g. `from foo import baz`) if the changes are to take effect. For these reasons, it is preferable to run IDLE with the default subprocess if at all possible.

Deprecated since version 3.4.

Help and Preferences¶

Help sources¶

Help menu entry “IDLE Help” displays a formatted html version of the IDLE chapter of the Library Reference. The result, in a read-only tkinter text window, is close to what one sees in a web browser. Navigate through the text with a mousewheel, the scrollbar, or up and down arrow keys held down. Or click the TOC (Table of Contents) button and select a section header in the opened box.

Help menu entry “Python Docs” opens the extensive sources of help, including tutorials, available at `docs.python.org/x.y`, where ‘x.y’ is the currently running Python version. If your system has an off-line copy of the docs (this may be an installation option), that will be opened instead.

Selected URLs can be added or removed from the help menu at any time using the General tab of the Configure IDLE dialog.

Setting preferences¶

The font preferences, highlighting, keys, and general preferences can be changed via Configure IDLE on the Option menu. Non-default user settings are saved in a `.idlerc` directory in the user’s home directory. Problems caused by bad user configuration files are solved by editing or deleting one or more of the files in `.idlerc`.

On the Font tab, see the text sample for the effect of font face and size on multiple characters in multiple languages. Edit the sample to add other characters of personal interest. Use the sample to select monospaced fonts. If particular characters have problems in Shell or an editor, add them to the top of the sample and try changing first size and then font.

On the Highlights and Keys tab, select a built-in or custom color theme and key set. To use a newer built-in color theme or key set with older IDLEs, save it as a new custom theme or key set and it will be accessible to older IDLEs.

IDLE on macOS¶

Under System Preferences: Dock, one can set “Prefer tabs when opening documents” to “Always”. This setting is not compatible with the tk/tkinter GUI framework used by IDLE, and it breaks a few IDLE features.

Extensions¶

IDLE contains an extension facility. Preferences for extensions can be changed with the Extensions tab of the preferences dialog. See the beginning of config-extensions.def in the idlelib directory for further information. The only current default extension is `zzdummy`, an example also used for testing.

idlelib¶

Source code: [Lib/idlelib → https://github.com/python/cpython/tree/3.12/Lib/idlelib](https://github.com/python/cpython/tree/3.12/Lib/idlelib)

The Lib/idlelib package implements the IDLE application. See the rest of this page for how to use IDLE.

The files in idlelib are described in idlelib/README.txt. Access it either in idlelib or click Help => About IDLE on the IDLE menu. This file also maps IDLE menu items to the code that implements the item. Except for files listed under ‘Startup’, the idlelib code is ‘private’ in sense that feature changes can be backported (see [PEP 434 → https://peps.python.org/pep-0434/](https://peps.python.org/pep-0434/)).

Powerful Python One-Liners - Python Wiki

Source: <https://wiki.python.org/moin/Powerful%20Python%20One-Liners>

This page is devoted to short programs that can perform powerful operations called *Python One-Liners*.

You may ask: why should I care? The answer is profound: if you cannot read and write one-liner code snippets, how can you ever hope to read and write more complicated codebases? Python one-liners can be just as powerful as a long and tedious program written in another language designed to do the same thing. In other languages (think: Java) this would be nearly *impossible*, but in Python, it's a lot easier to do. The trick is to think of something that will "do a lot with a little." Most importantly, reading and writing about Python one-liners (e.g., in this post) is a lot of fun! There's even a whole subculture around who can write the shortest code for a given problem.

It would be awesome if this page expanded to the point where it needs some sort of organization system. (*Edit: The one-liners are now sorted more or less by ease-of-understanding -- from simple to hard. Please use a "sorted insert" for your new one-liner.*)

The source code is contributed from different Python coders --- Thanks to all of them! Special thanks to the early contributor JAM. → <https://wiki.python.org/moin/JAM>

Of course, there is debate on whether one-liners are even *Pythonic*. As a rule of thumb: if you use one-liners that are confusing, difficult to understand, or to show off your skills, they tend to be *Unpythonic*. However, if you use well-established one-liner tricks such as list comprehension → <https://docs.python.org/3/tutorial/datastructures.html#list-comprehensions> or the ternary operator → <https://blog.finxter.com/python-one-line-ternary/>, they tend to be Pythonic.

So, use your one-liner superpower wisely!

Free Python One-Liners Learning Resources

- Free "Python One-Liners" videos & book resources → <https://pythononeliners.com/>
- Collection of "One-Liners" with interactive shell → <https://blog.finxter.com/10-python-one-liners/>
- Book "Python One-Liners" → <https://www.amazon.com/gp/product/B07ZY7XMX8>
- Interesting Quora Thread "Python One-Liner" → <https://www.quora.com/What-are-some-of-the-most-elegant-create-st-Python-one-liners>
- Python One-Line X → <https://blog.finxter.com/python-one-line-x/> - How to accomplish different tasks in a single line
- Subreddit "Python One-Liners" → <https://www.reddit.com/r/PythonOneLiners/>
- Github "Python One-Liners" → <https://github.com/finxter/PythonOneLiners/> - Share your own one-liners with the community

Overview: 10 one-liners that fit into a tweet

I visited this page oftentimes and I loved studying the one-liners presented above. Thanks for creating this awesome resource, JAM, and RJW! 😊

Because I learned a lot from studying the one-liners, I thought why not revive the page (after almost ten years since the last change happened)?

After putting a lot of effort into searching the web for inspiration, I created the following ten one-liners. Some of them are more algorithmic (e.g. Quicksort). Some day, I will add a detailed explanation here - but for now, you can read this blog article → <https://blog.finxter.com/10-python-one-liners/> to find explanations.

```

1
2 phrase.find(phrase[::-1])
3
4
5 a, b = b, a
6
7
8 sum(stock_prices[::2])
9
10
11 [line.strip() for line in open(filename)]
12
13
14 reduce(lambda x, y: x * y, range(1, n+1))
15
16
17 python -m cProfile foo.py
18
19
20 lambda l: reduce(lambda z, x: z + [y + [x] for y in z], l, [[]])
21
22
23 lambda x: x if x<=1 else fib(x-1) + fib(x-2)
24
25
26 lambda L: [] if L==[] else qsort([x for x in L[1:] if x< L[0]]) + L[0:1] +
qsort([x for x in L[1:] if x>=L[0]])
27
28
29 reduce( (lambda r,x: r-set(range(x**2,n,x)) if (x in r) else r),
range(2,int(n**0.5)), set(range(2,n)))

```

Find All Indices of an Element in a List

Say, you want to do the same as the `list.index(element)` method but return all indices of the element in the list rather than only a single one.

In this one-liner, you're looking for element 'Alice' in the list `lst = [1, 2, 3, 'Alice', 'Alice']` so it even works if the element is not in the list (unlike the `list.index()` method).

```

1
2 lst = [1, 2, 3, 'Alice', 'Alice']
3
4
5 indices = [i for i in range(len(lst)) if lst[i]=='Alice']
6
7
8 print(indices)
9

```

echo unicode character:

```
python -c "print unichr(234)"
```

This script echos "ê"

Reimplementing cut

Print every line from an input file but remove the first two fields.

```
python -c "import sys; [sys.stdout.write(' '.join(line.split(' ')[2:])) for line in sys.stdin]" < input.txt
```

Decode a base64 encoded file

```
import base64, sys; base64.decode(open(sys.argv[1], "rb"), open(sys.argv[2], "wb"))
```

Editing a list of files in place

I came up with this one-liner in response to an [article](http://linuxgazette.net/issue96/orr.html) → <http://linuxgazette.net/issue96/orr.html> that said it couldn't be done as a one-liner in Python.

What this does is replace the substring "at" by "op" on all lines of all files (in place) under the path specified (here, the current path).

- **Caution:** Don't run this on your home directory or you're going to get **all your text files edited**.

```
import sys,os,re,fileinput;a=[i[2] for i in os.walk('.') if i[2]] [0];[sys.stdout.write(re.sub('at','op',j)) for j in fileinput.input(a,inplace=1)]
```

Clearer is:
import os.path;
a=[f for f in os.listdir('.') if not os.path.isdir(f)]

Set of all subsets

- Function that returns the set of all subsets of its argument

```
f = lambda x: [[y for j, y in enumerate(set(x)) if (i >> j) & 1] for i in range(2**len(set(x)))]
```

```
>>>f([10,9,1,10,9,1,1,1,10,9,7])
[], [9], [10], [9, 10], [7], [9, 7], [10, 7], [9, 10, 7], [1], [9, 1], [10, 1],
[9, 10, 1], [7, 1], [9, 7, 1], [10, 7, 1], [9, 10, 7, 1]]
```

-RJW

Alternative (shorter, more functional version):

```
f = lambda l: reduce(lambda z, x: z + [y + [x] for y in z], l, [])
```

Terabyte to Bytes

Want to know many bytes a terabyte is? If you know further abbreviations, you can extend the list.

```
import pprint;pprint pprint(zip('Byte', 'KByte', 'MByte', 'GByte', 'TByte'), (1
<< 10*i for i in range(5)))
```

Largest 8-Bytes Number

And what's the largest number that can be represented by 8 Bytes?

```
print '\n'.join("%i Byte = %i Bit = largest number: %i" % (j, j*8, 256**j-1) for
j in (1 << i for i in range(8)))
```

Cute, isn't it?

Display List of all users on Unix-like systems

```
print '\n'.join(line.split(":",1)[0] for line in open("/etc/passwd"))
```

CSV file to json

```
python -c "import csv,json;print
json.dumps(list(csv.reader(open('csv_file.csv'))))"
```

Compress CSS file

```
python -c "import re,sys;print re.sub("\s*({};:})\s*", "\\\1", re.sub("/\*\.*?
\*/", "", re.sub("\s+", " ", sys.stdin.read())))"
```

Decode string written in Hex

```
python -c "print ''.join(chr(int(''.join(i), 16)) for i in zip(*[iter('474e552773204e6f7420556e6978')] * 2))"
```

Retrieve content text from HTTP data

```
python -c "import sys; print sys.stdin.read().replace('\r','').split('\n\n',2)[1]";
```

Broadcast magic packet to power on wakeonlan enabled computer

Small [Wikipedia Python script](https://en.wikipedia.org/wiki/Wake-on-LAN#Creating_and_sending_the_magic_packet) → https://en.wikipedia.org/wiki/Wake-on-LAN#Creating_and_sending_the_magic_packet squeezed out to 166 characters length one-liner:

```
python -c "import socket as S;s=S.socket(S.AF_INET,S.SOCK_DGRAM);s.setsockopt(S.SOL_SOCKET,S.SO_BROADCAST,1);s.sendto(b'\xff'*6+b'\x84\x47\x09\x0b\x7f\xfd'*16, ('<broadcast>',7))"
```

Uglier, with Python constant names replaced by their values, squeezed out to 123 characters only:

```
python -c "import socket as S;s=S.socket(2,2);s.setsockopt(1,6,1);s.sendto(b'\xff'*6+b'\x84\x47\x09\x0b\x7f\xfd'*16, ('<broadcast>',9))"
```

Replace mac address \x84\x47\x09\x0b\x7f\xfd with mac address of computer that should be powered on.

Prints file extension

```
print '~/python/one-liners.py'.split('.')[ -1]
```

Escapes content from stdin

This can be used to convert a string into a "url safe" string

```
python -c "import urllib, sys ; print urllib.quote_plus(sys.stdin.read())";
```

Reverse lines in stdin

```
python -c "import sys; print '\n'.join(reversed(sys.stdin.read().split('\n')))"
```

Print top 10 lines of stdin

```
python -c "import sys; sys.stdout.write(''.join(sys.stdin.readlines()[:10]))" </path/to/your/file
```

Apply regular expression to lines from stdin

```
[another command] | python -c "import sys,re;[sys.stdout.write(re.sub('PATTERN', 'SUBSTITUTION', line)) for line in sys.stdin]"
```

Modify lines from stdin using map

```
python -c "import sys; tmp = lambda x: sys.stdout.write(x.split()[0]+\t+str(int(x.split()[1])+1)+'\n'); map(tmp, sys.stdin);"
```

Cramming Python into Makefiles

A related issue is embedding Python into a Makefile. I had a really long script that I was trying to cram into a makefile so I automated the process:

```

1 import sys,re
2
3 def main():
4     fh = open(sys.argv[1],'r')
5     lines = fh.readlines()
6     print '\tpython2.2 -c "`printf \\\"if 1:\\\\n\\\''
7     for line in lines:
8         line = re.sub('[\\\\\'\\"]()','\\\\g<0>',line)
9
10
11     wh_spc_len = len(re.match('\\s*',line).group())
12
13     sys.stdout.write('\\t')
14     sys.stdout.write(wh_spc_len/4*'\\t'+line.rstrip().lstrip())
15     sys.stdout.write('\\n\\\\n')
16     print '\\t`"'
17
18 if __name__=='__main__':
19     main()

```

This script generates a "one-liner" from make's point of view.

They call it "The Pyed Piper" or `pyp`. It's pretty similar to the `-c` way of executing python, but it imports common modules and has its own preset variable that help with splitting/joining, line counter, etc. You use pipes to pass information forward instead of nested parentheses, and then use your normal python string and list methods. Here is an example from the homepage:

Here, we take a Linux long listing, capture every other of the 5th through the 10th lines, keep the username and filename fields, replace "hello" with "goodbye", capitalize the first letter of every word, and then add the text "is splendid" to the end:

```
ls -l | pyp "pp[5:11:2] | whitespace[2], w[-1] | p.replace('hello','goodbye') |
p.title(),'is splendid'"
```

and the explanation:

This uses `pyp`'s built-in string and list variables (`p` and `pp`), as well as the variable `whitespace` and its shortcut `w`, which both represent a list based on splitting each line on whitespace (`whitespace = w = p.split()`). The other functions and selection techniques are all standard Python. Notice the pipes ("|") are inside the `pyp` command.

<http://code.google.com/p/pyp/> <http://opensource.imageworks.com/?p=pyp>

Contributed Code

- [JAM/Code/PlatformFinder](https://wiki.python.org/moin/JAM/Code/PlatformFinder) → <https://wiki.python.org/moin/JAM/Code/PlatformFinder> - This program tells you what platform you are using.
- [JAM/Code/ComPYiler](https://wiki.python.org/moin/JAM/Code/ComPYiler) → <https://wiki.python.org/moin/JAM/Code/ComPYiler> - This program compiles every .py file in the Python directory.
- [Powerful Python One-Liners/Hostname](https://wiki.python.org/moin/Powerful%20Python%20One-Liners/Hostname) → <https://wiki.python.org/moin/Powerful%20Python%20One-Liners/Hostname> - This program tells you what your hostname is.

Powerful Python One-Liners (last edited 2023-07-19 19:23:56 by [HermannStammWilbrandt](#) → <https://wiki.python.org/moin/HermannStammWilbrandt>)

BeginnersGuide/Help - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Help>

Getting Help

If you experience problems using Python, your first step should be to check the documentation.

- The [Python documentation](https://www.python.org/doc/versions/) → <https://www.python.org/doc/versions/> describes the language and the standard modules in detail.
- Many common questions are answered in the [Frequently Asked Question lists](http://docs.python.org/faq/) → <http://docs.python.org/faq/>.
- If you suspect you've discovered a bug in the Python core, search the [Python Bug Tracker](http://bugs.python.org/) → <http://bugs.python.org/> to find out if it's already been reported.

If the documentation doesn't help, here's how to get help from real people:

- [comp.lang.python](#) → Post to the Python newsgroup, renowned for its friendly and helpful atmosphere. See [the guide to Python mailing lists](https://www.python.org/community/lists/#comp-lang:python) → <https://www.python.org/community/lists/#comp-lang:python> for more information.
- [tutor mailing list](#) → <http://mail.python.org/mailman/listinfo/tutor> A moderate-volume mailing list for folks who want to ask questions while learning computer programming with Python. See [the guide to Python mailing lists](https://www.python.org/community/lists/#tutor) → <http://www.python.org/community/lists/#tutor> for more information.
- help@python.org → <mailto:help@python.org> The Python help desk. You can ask a group of knowledgeable volunteers questions about all your Python problems. See [the guide to Python mailing lists](https://www.python.org/community/lists/#python-help) → <http://www.python.org/community/lists/#python-help> for more information.

webmaster is not a Python language support channel!!!

- The webmaster should only be contacted if you have found a problem with the python.org website such as a broken link, typographic error, or other problem.
- For such problems, also check [the github issue tracker for the website](https://github.com/python/pythondotorg/issues) → <https://github.com/python/pythondotorg/issues>

IntroductoryBooks - Python Wiki

Source: <https://wiki.python.org/moin/IntroductoryBooks>

The books on this page are all general introductions to the Python language. Most of these books will contain a few chapters on particular applications such as GUI interfaces or Web programming, but won't go into great detail on any one topic; refer to the [PythonBooks](#) → <https://wiki.python.org/moin/PythonBooks> page for lists of application-specific books. Experienced programmers who prefer a brief and condensed introduction should look at the list of [ReferenceBooks](#) → <https://wiki.python.org/moin/ReferenceBooks>.

[PyFlo - A Free, Interactive Guide to Python Programming](#) → <https://pyflo.net/>

Benjamin Dicken
Website released in 2023

Book Overview

Pyflo is a free, interactive guide to Python programming. It contains over 70 lessons that provide a high-quality foundation in computer programming. Key Features include:

1. Coverage of the core Python programming constructs including functions, conditions, loops, data structures, and file IO.
2. Short and modularized lessons that present the material in manageable chunks.
3. Interactivity in the form of embedded questions, which provide immediate feedback.
4. High-quality images, diagrams, and animations to help learners grasp concepts more quickly.
5. Guided projects to give learners the opportunity to apply what has been learned.

This completely free tool will help jump-start your Python programming career.

[Essential Python for Machine Learning](#) → <https://www.amazon.com/dp/B0C9SDNCP3>

Abhishek Singh
ISBN-13: 979-8852254672 572 Pages ((July 14, 2023))

Book Overview

"Essential Python for Machine Learning" is a comprehensive guide that equips readers with the fundamental Python programming skills necessary to kickstart their journey into the exciting world of machine learning. This book covers the essential topics and techniques required to understand and apply Python in the field of machine learning.

Starting with the basics of Python programming, readers learn about conditional statements, loops, data structures, and functions. The book then delves into important libraries like [NumPy](#) → <https://wiki.python.org/moin/NumPy> and Pandas, enabling readers to manipulate and analyze data efficiently. With a focus on practical application, readers explore topics such as file handling, regular expressions, and error handling, gaining essential skills for real-world machine learning projects.

Key Features:

1. Comprehensive coverage of Python programming fundamentals.
2. In-depth exploration of key libraries like [NumPy](#) → <https://wiki.python.org/moin/NumPy>, and Pandas for data manipulation and analysis.
3. Practical application of Python concepts to real-world machine learning scenarios.
4. Clear explanations, illustrative examples, and practical exercises to enhance learning.
5. Suitable for beginners and those with some programming experience looking to enter the field of machine learning.

Whether you are a student, aspiring data scientist, or a Python enthusiast looking to venture into machine learning, "Essential Python for Machine Learning" provides the essential knowledge and skills needed to get started. Gain a strong foundation in Python programming and unlock the power of machine learning with this practical and accessible guide.

[Master Python Using Version 3.11: Learn Python Like Never Before](https://www.amazon.com/dp/B0BW32CWD5) → <https://www.amazon.com/dp/B0BW32CWD5>

Abhishek Singh

ISBN-13: 979-8385523276 385 Pages (First edition, March 2023)

Book Overview:

One of the most popular programming languages used nowadays is called Python. It has just become more potent with the addition of a few additional features in its most recent iteration. Yet, not everyone is aware of its potency. The goal of this book is to introduce readers to Python programming from scratch. You are welcome to learn Python Programming like never before, whether you are a computer science undergraduate or not, a school-going programming geek or a beginner. You'll discover everything there is to know about Python, including how to use it for programming, when to employ the much-touted walrus operator and other brand-new features of version 3.11. Many solved programs that were used as interview questions at various firms are included in the book. Understanding the typical mistakes and how to fix them are covered in a separate chapter. The difficult topics have been simplified and expressed in simple terms. Beginners and intermediate readers should read the book.

[Python from the Very Beginning](https://pythonfromtheverybeginning.com/) → <https://pythonfromtheverybeginning.com/>

John Whitington

ISBN-13: 978-0-9576711-5-7 238 Pages (Second edition, May 2023)

Also available as a DRM-free PDF or EPUB.

Book Overview:

In *Python from the Very Beginning* John Whitington takes a no-prerequisites approach to teaching a modern general-purpose programming language. Each small, self-contained chapter introduces a new topic, building until the reader can write quite substantial programs. There are plenty of questions and, crucially, worked answers and hints.

Python from the Very Beginning will appeal both to new programmers, and to experienced programmers eager to explore a new language. It is suitable both for formal use within an undergraduate or graduate curriculum, and for the interested amateur.

Second edition, fully updated in 2023.

[Amazon printed and Kindle](https://www.amazon.com/Python-Very-Beginning-exercises-answers/dp/0957671156/) → <https://www.amazon.com/Python-Very-Beginning-exercises-answers/dp/0957671156/>

[PDF and EPUB eBook](https://pythonfromtheverybeginning.com/) → <https://pythonfromtheverybeginning.com/>

[The Art of Clean Code: Best Practices to Eliminate Complexity and Simplify Your Life](https://nostarch.com/art-clean-code) → <https://nostarch.com/art-clean-code>

Christian Mayer

Book Overview:

Most software developers waste thousands of hours working with overly complex code.

The eight core principles in *The Art of Clean Code* will teach you how to write clear, maintainable code without compromising functionality.

1. Complexity Problem
2. The 80/20 Principle
3. Build a Minimal Viable Product

4. Write Clean and Simple Code
5. Premature Optimization is the Root of All Evil
6. Flow
7. Do One Thing Well (Unix)
8. Less is More in Design

The book's guiding principle is simplicity: reduce and simplify, then reinvest energy in the important parts to save you countless hours and ease the often onerous task of code maintenance.

Amazon: [The Art of Clean Code](https://www.amazon.com/Art-Clean-Code-Practices-Complexity/dp/1718502184) → <https://www.amazon.com/Art-Clean-Code-Practices-Complexity/dp/1718502184>

[Python Dash: Build Data Analysis and Visualization Apps with Plotly](https://nostarch.com/python-dash) → <https://nostarch.com/python-dash>

Adam Schroeder, Christian Mayer, and Ann Marie Ward

Book Overview:

Create stunning interactive dashboard applications in Python with the Dash visualization and data analysis tool. Build interfaces that make sense of your data, and make it pretty.

A swift and practical introduction to building interactive data visualization apps in Python, known as dashboards. You've seen dashboards before; think election result visualizations you can update in real time, or population maps you can filter by demographic. With the Python Dash library you'll create analytic dashboards that present data in effective, usable, elegant ways in just a few lines of code.

The book is fast-paced and caters to those entirely new to dashboards. It will talk you through the necessary software, then get straight into building the dashboards themselves. You'll learn the basic format of a Dash app by building a Twitter analysis dashboard that maps the number of likes certain accounts gained over time. You'll build up skills through three more sophisticated projects. The first is a global analysis app that compares country data in three areas: the percentage of a population using the internet, percentage of parliament seats held by women, and CO₂ emissions. You'll then build an investment portfolio dashboard, and an app that allows you to visualize and explore machine learning algorithms.

In this book you will:

- Create and run your first Dash apps
- Use the pandas library to manipulate and analyze social media data
- Use Git to download and build on existing apps written by the pros
- Visualize machine learning models in your apps
- Create and manipulate statistical and scientific charts and maps using Plotly

Dash combines several technologies to get you building dashboards quickly and efficiently. This book will do the same.

Amazon: [Python Dash: Build Data Analysis and Visualization Apps with Plotly](https://www.amazon.com/Python-Dash-Build-Data-Analysis-Visualization-Apps-with-Plotly/dp/1718502222) → [https://www.amazon.com/Python-Dash-Build-Data-Analysis-and-Visualization-Apps-with-Plotly/dp/1718502222](https://www.amazon.com/Python-Dash-Build-Data-Analysis-Visualization-Apps-with-Plotly/dp/1718502222)

[Learn to Code by Solving Problems: A Python Programming Primer](https://nostarch.com/learn-code-solving-problems) → <https://nostarch.com/learn-code-solving-problems>

Daniel Zingaro

Book Overview:

Learn to code, right now. No coding experience required.

Don't try to memorize Python syntax with the hope of using it one day. Learn Python for good while you solve programming problems.

You get line by line walkthroughs, "check your understanding" questions, practice examples with answers, instant and accurate feedback using convenient online websites, discussion of common errors, and more.

This isn't just a Python book. You'll definitely learn Python here. But you'll also learn how to think and solve problems like programmers do. And you'll learn why slow code is slow and how to write better, faster code. You're building skills that will not only turn you into a Python programmer but will also help you with whatever programming language you might use in the future.

[Leaving the Rat Race with Python: An Insider's Guide to Freelance Developing](https://smile.amazon.com/gp/product/B08G1XLDNB/) → <https://smile.amazon.com/gp/product/B08G1XLDNB/>

Amazon Smile Link ;)

L. Rieger, C. Mayer

ASIN: B08G1XLDNB Finxter 233 Pages

Book Overview:

Leaving the Rat Race with Python shows you how to nurture, grow, and harness your work-from-home coding business online --- and live the good life!

As an insider's guide to freelance developing, *Leaving the Rat Race* helps you nurture, grow, and harness your new online coding business plant---even if you've got little or no experience as a programmer and business owner.

By following the instructions in this book, you'll create new joy, happiness, and a sense of independence and self-reliance. Because life's too short to work a company you don't believe in!

[Kindle Amazon](https://smile.amazon.com/gp/product/B08G1XLDNB/) → <https://smile.amazon.com/gp/product/B08G1XLDNB/> [eBook PDF](https://blog.finxter.com/book-leaving-the-rat-race-with-python/) → <https://blog.finxter.com/book-leaving-the-rat-race-with-python/> [Book + Resources Python Freelancing](https://python-freelancer.com/) → <https://python-freelancer.com/>

<https://www.amazon.com/dp/1094777978>

Abhishek Singh, Zohaib Hasan

ISBN-13: 978-1094777979 73 Pages (April 20, 2019)

Book Overview:

Awarded as one of the best books of regular expression in 2020 and 2021 by bookauthority.org. Regular Expression is one topic which is nightmare for many seasoned programmers also. And what better choice than Python could have been? An easy to understand language to explain an uneasy to understand topic. This book is designed for absolute beginners with elementary knowledge of Python language. Regular Expressions are considered as a tough topic and usually they are not covered in syllabus in much detail. Mostly, a chapter is given on this topic in syllabus or curriculum books. But here, a complete book is dedicated. This book covers Regular Expression in detail with plenty of examples covering password validation, email ID validation, string manipulations etc. and their step by step analysis. The examples are presented in such a way that one can feel like coding only by reading the codes itself. The book is written in a way to give stress free environment.

[Learning Regular Expression Page](https://regexusingpython.wordpress.com/) → <https://regexusingpython.wordpress.com/>

[Actionable Python 3.8](https://www.amazon.com/Python-3-8-Nat-Dunn/dp/1951959027) → <https://www.amazon.com/Python-3-8-Nat-Dunn/dp/1951959027>

Nat Dunn, Webucator

ISBN-13: 978-1951959029 536 Pages (March 15, 2020)

Book Overview:

Learning to program is hard. Don't let anyone tell you different. If you bought or are considering buying this book because someone told you that you should learn Python, then maybe you should reconsider. The big question to ask yourself is "*Do you want to learn Python?*" Certainly, this book will help you answer that question, but be prepared to struggle, and to make mistakes, and to get frustrated, and to bang your head against the keyboard. This is Pandora's box, baby, but...

If you get excited by solving hard problems... if you're into Sudoku, or crosswords, or jigsaw puzzles, or if you'll spend hours trying to figure out how to separate two metal looping thingamajigs that you found on someone's shelf..., or you like tinkering with engines, or figuring out the best recipe for chocolate chip cookies... in other words, if you like a challenge, then you might love Python programming. It's not easy, but it's easy enough to get started. Pick up a book (good start!), and start going through it. Be patient with yourself. Pretend you're the teacher as well as the student. When the student fails, the good teacher doesn't insult or chastise. The good teacher encourages. The good teacher knows that failure is a part of learning, and that overcoming failure is exciting and leads to more learning. **So, be a good teacher and treat yourself well.**

But be a good student too. When reading this book, you should be sitting at your computer. When you first begin, plan to spend at least two uninterrupted hours. You should do your best to get through the first two lessons in the first sitting. That will get you through the foundational learning. From there on out, you'll be hopping back and forth between the book and viewing, editing, and writing code. Go through the book slowly and methodically. Read every demo carefully. Work through every exercise. You cannot learn to code through reading alone. You must practice.

This book assume no prior knowledge of Python, but it introduces some pretty advanced topics, such as working with databases, scraping the web, and object-oriented Python.

[Learning IoT Page → https://www.learningiot.net/](https://www.learningiot.net/)

[Learning IoT with Python and Raspberry Pi → https://www.barnesandnoble.com/w/learning-iot-with-python-and-raspberry-pi-ei-horvath/1133345171?ean=9780578549361](https://www.barnesandnoble.com/w/learning-iot-with-python-and-raspberry-pi-ei-horvath/1133345171?ean=9780578549361)

E.I. Horvath E.A. Horvath

ISBN-13:9780578549361 760 Pages (September 1, 2019)

Book Overview:

The very nature of IoT requires a knowledge of a programming language and electronic circuits and a knowledge of networking in order to interface with the physical world and networking platforms. Learning IoT with Python and Raspberry Pi uses a cross-disciplinary approach when teaching IoT. Circuits which use light dependent resistors, LEDs, buzzers, passive infrared sensors, ultrasonic sensors, and other components will be used to explore and elucidate programming concepts. Users can monitor devices connected to the Raspberry Pi. Learning IoT with Python and Raspberry Pi allows the programmer to explore the Internet of Things (IoT) by writing code which will monitor temperature, pressure, humidity, and light levels and will determine when motion is detected, etc. Based upon these physical events, a text message is sent, an email is sent, a phone call is made, or a picture is taken.

- 300+ code listings illustrate Python programming concepts.
- Complete code listings in this textbook for all projects.
- 250+ homework problems ranging from short Python scripts to web site configuration projects.
- Labs on measuring voltage, current, and resistance using a multimeter.
- Introduction to basic Linux commands.
- Use the Sense HAT to read environmental data, such as temperature, pressure, and humidity.
- Send a text message if the light level detected by an LDR circuit falls below a threshold.
- Send an email if motion is detected using a PIR sensor circuit.
- Use an ultrasonic sensor in a proximity alert circuit and make a phone call if an intruder gets too close.
- Use a GPS HAT to get waypoints and store them in a file.
- Use an analog to digital converter to read in data from sensors.
- Control a servo using classes.
- Take a picture with the Raspberry Pi Camera.
- Install and configure an HTTP server with multiple virtual web sites.

- Install and configure an FTP server.
- Use SSH and VNC to remotely control the Raspberry Pi.
- Create a virtual machine in the cloud.
- Upload data to the cloud.
- Learn how to write multi-threaded applications.
- Play an audio file.
- Control a robot using a multi-threaded application.

Learning IoT Page → <https://www.learningiot.net/>

The Python Coding Book: Understanding what programming really is → <https://thepythoncodingbook.com/>

Stephen Grupetta

Freely available online book

Book Overview:

The key to mastering programming is to truly understand why we do things in the way we do them.

Learning the methods is not enough. My focus in the years I've spent teaching coding has been to get my students to think in the way the computer does. Only then can we speak its language. We need to understand the various tools in a coding language as if they're our best friends, not just mere acquaintances. The focus of this book is **real** understanding. 'Why' is just as important a question as 'How'. And I like analogies—a lot. So your journey through learning how to code will take you through The White Room and coffee machines, and you'll even climb a tree! This book will start from the basics, and in its second part will start to move towards areas of programming relevant to fields such as science, finance, and other data-driven fields. The Table of contents of the book is the following:

- 0 | How To Learn To Code: Preface
- 1 | Getting Started: Your First Project
- 2 | Loops, Lists and More Fundamentals
- 3 | Power-up Your Coding: Create Your Own Functions
- 4 | Data, Data Types, and Data Structures
- 5 | Dealing With Errors and Bugs
- ~ | The White Room: Understanding Programming
- 6 | Functions Revisited
- 7 | Object-Oriented Programming
- 8 | Numerical Python for Quantitative Applications Using numpy
- 9 | Dealing with Dates and Times in Python
- 10 | Basics of Data Visualisation Using matplotlib
- 11 | Functional Programming and Lambda Functions
- 12 | Analysing Data Using pandas

The Python Coding Book: Understanding what programming really is → <https://thepythoncodingbook.com/>

Python One-Liners → <https://www.amazon.com/gp/product/B07ZY7XMX8>

Christian Mayer

ASIN: B07ZY7XMX8 NoStarch → <https://wiki.python.org/moin/NoStarch> 256 Pages (May 2020)

Book Overview:

Python programmers will improve their computer science skills with these useful one-liners

Python One-Liners will show readers how to perform useful tasks with one line of Python code. Following a brief Python refresher, the book covers essential advanced topics like slicing, list comprehension, broadcasting, lambda functions, algorithms, regular expressions, neural networks, logistic regression and more. Each of the 50 book sections introduces a problem to solve, walks the reader through the skills necessary to solve that problem, then provides a concise one-liner Python solution with a detailed explanation.

[Book page \(free material\)](https://pythononeliners.com/) → <https://pythononeliners.com/> [Publisher's page](https://nostarch.com/pythononeliners) → <https://nostarch.com/pythononeliners>

[Learn Programming in Python with Cody Jackson](https://www.packtpub.com/application-development/learn-programming-python-cody-jackson) → <https://www.packtpub.com/application-development/learn-programming-python-cody-jackson>

Cody Jackson

ISBN 13: 9781789531947 Packt 304 Pages (November 2018)

Book Overview:

Kick-start your development journey with this end-to-end guide that covers Python programming fundamentals along with application development

Python is a cross-platform language used by organizations such as Google and NASA. It lets you work quickly and efficiently, allowing you to concentrate on your work rather than the language. Based on his personal experiences when learning to program, Learn Programming in Python with Cody Jackson provides a hands-on introduction to computer programming utilizing one of the most readable programming languages—Python. It aims to educate readers regarding software development as well as help experienced developers become familiar with the Python language, utilizing real-world lessons to help readers understand programming concepts quickly and easily.

The book starts with the basics of programming, and describes Python syntax while developing the skills to make complete programs. In the first part of the book, readers will be going through all the concepts with short and easy-to-understand code samples that will prepare them for the comprehensive application built in parts 2 and 3. The second part of the book will explore topics such as application requirements, building the application, testing, and documentation. It is here that you will get a solid understanding of building an end-to-end application in Python. The next part will show you how to complete your applications by converting text-based simulation into an interactive, graphical user interface, using a desktop GUI framework. After reading the book, you will be confident in developing a complete application in Python, from program design to documentation to deployment.

[Publisher's page](https://www.packtpub.com/application-development/learn-programming-python-cody-jackson) → <https://www.packtpub.com/application-development/learn-programming-python-cody-jackson>

[Python 3 Object-Oriented Programming - Third Edition](https://www.packtpub.com/application-development/python-3-object-oriented-programming-third-edition) → <https://www.packtpub.com/application-development/python-3-object-oriented-programming-third-edition>

Dusty Phillips

ISBN 13: 9781789615852 Packt 466 Pages (October 2018)

Book Overview:

Uncover modern Python with this guide to Python data structures, design patterns, and effective object-oriented techniques

Starting with a detailed analysis of object-oriented programming, you will use the Python programming language to clearly grasp key concepts from the object-oriented paradigm. You will learn how to create maintainable applications by studying higher level design patterns. The book will show you the complexities of string and file manipulation, and how Python distinguishes between binary and textual data. Not one, but two very powerful automated testing systems, unittest and pytest, will be introduced in this book. You'll get a comprehensive introduction to Python's concurrent programming ecosystem.

By the end of the book, you will have thoroughly learned object-oriented principles using Python syntax and be able to create robust and reliable programs confidently.

Publisher's page → <https://www.packtpub.com/application-development/python-3-object-oriented-programming-third-edition>

'Start Here: Python Programming for Beginners "Start Here: Python Programming for beginners" → <http://www.toonzcat.com/book.html> by Jody S. Ginther was written after the changes made in Python 3. It is aimed at total beginners and uses hands-on practice and humor to make learning to program more relaxing and enjoyable.

Software Development I: with Python

ISBN: 978-0-9754759-4-2, LCCN: 2010906917, 605 pages

"Software Development I: with Python" → <http://www.softbaugh.com/courses/python1/>

Tom Baugh

This course introduces the student to Python, cultivating professional software development skills along the way. Yet, this course requires no prior programming knowledge to learn or to teach. Organized as sixty-four lessons, sixteen tests and two exams, for a total of 82 lecture units, this course covers the following topics:

- Python variable, function, class and environment syntax.
- Custom module development.
- Software development skills and activities.
- Python dictionaries, lists, tuples and iterable types.
- Command console operations.
- List comprehensions.
- Python exception handling.
- IPython operations and commands.
- Essential built-in Python library features.
- An introduction to Matplotlib.
- An introduction to wxPython and windowed development.

By the end of the course, students will have learned how to write Python projects, including custom classes and modules. The course culminates with an exploration of windowed application development, including a bitmapped graphics project. The course material includes the student text and CD-ROM with instructor's guide for each lesson, tests, exams, solutions guide for exercises, tests and exams, student handouts and completed projects for each lesson. The format of the course is intended to support classroom use, homeschool use, or self-study.

Python Programming in Context

Bradley N. Miller and David L Ranum

Python Programming in Context is a clear, accessible introduction to the fundamental programming and problem solving concepts necessary for students at this level. The authors carefully build upon the many important computer science concepts and problem solving techniques throughout the text and offer relevant, real-world examples and exercises to reinforce key material. Programming skills throughout the text are linked to applied areas such as Image Processing, Cryptography, Astronomy, Music, the Internet, and Bioinformatics, giving students a well rounded look of its capabilities.

dead link

<http://www.jbpub.com/catalog/9780763746025/>

Python First: The Joy of Success

Atanas Radenski

ISBN: 978159526-713-9, 512 pages, Llumina Press, April 2007

This book is a paper companion to the "Python First" digital pack from studypack.com. The complete digital study pack features e-texts, slides, a wealth of detailed self-guided labs that learners can complete on their own, sample programs, and extensive quizzes. The book offers a printed version of the e-texts and self-guided labs in the same format as they appear in the online digital pack.

[Publisher's page → http://www.llumina.com/store/python.htm](http://www.llumina.com/store/python.htm)

Building Skills in Python

Steven F. Lott

How do you learn Python? By doing a series of exercises, each of which adds a single new feature of the language. This 250+ page book has 31 chapters that will help you build Python programming skills through a series of exercises. This book includes six projects from straight-forward to sophisticated that will help solidify your Python skills.

http://homepage.mac.com/s_lott/books/python.html

Building Skills in Programming How To Write Your Own Software Using Python

Steven F. Lott

How do you learn to solve your own programs by writing programs? By doing a series of exercises, each of which builds up a part of the skill set we call "computer programming". This book has 54 chapters that will help you build basic programming skills through a series of exercises that grow from simple identification of the parts of your computer through to statistical simulations.

http://homepage.mac.com/s_lott/books/nonprogrammer.html

A Byte of Python

- Swaroop C H

A Byte of Python is a book on programming using the Python language. It serves as a tutorial or guide to the Python language for anyone. If all you know is how to save text files, then you can learn Python using this book. If you are an expert programmer who loves C, Perl, Java or C#, you can also learn Python using this book.

This book can be read online or downloaded from <http://www.swaroopch.com/notes/Python>

Practical Python

- Magnus Lie Hetland

[1590590066 → http://www.amazon.com/exec/obidos/ISBN=1590590066](http://www.amazon.com/exec/obidos/ISBN=1590590066), 619 pages, APress (August 2002)

The first half of this book introduces the Python language, and the second half demonstrates its usage in various practical projects such as "automated document conversion, newsgroup administration, graphical PDF document generation, remote document maintenance, the creation of a peer-to-peer system with XML-RPC, database integration, and GUI and game development." A new edition of this book is available under the title **Beginning Python: From Novice to Professional**.

[Publisher's page → http://www.apress.com/book/bookDisplay.html?bID=93](http://www.apress.com/book/bookDisplay.html?bID=93)

Beginning Python: From Novice to Professional

- Magnus Lie Hetland

1-59059-519-x → <http://www.amazon.com/exec/obidos/ISBN=1-59059-519-x>, 640 pages, APress (September 2005)

This is an update of **Practical Python**.

Publisher's page → <http://www.apress.com/book/bookDisplay.html?bID=10013>

Computer Programming is Fun!

- David Handy

208 pages, Handy Software and Publishing (April 2005)

Written by a homeschooling Dad for teenage youth, this introductory computer programming book is for people who have no prior programming experience. Teaches the basic principles of programming using Python, with lots of examples. Small video game project at the end. Good for self-study or classroom use.

<http://www.handysoftware.com/cpif/>

Python Programming for the Absolute Beginner

- Michael Dawson

1592000738 → <http://www.amazon.com/exec/obidos/ISBN=1592000738> Premier Press, 456 pages (2003)

Simple intro accessible for beginners. Very hands-on and fun, utilizes development of games to teach each concept. Great for visual and hands-on learners alike.

Publisher's page → http://www.premierpressbooks.com/ptr_detail.cfm?group=Programming&isbn=1-59200-073-8

The Python Apprentice → <https://www.packtpub.com/application-development/python-apprentice>

Robert Smallshire, Austin Bingham

ISBN 13: 9781788293181 Packt Publishing 352 pages (June 2017)

Book Overview:

Before you master Python, you need to learn the culture and the tools to become a productive member of any Python project. This book will give you a practical and thorough introduction to Python programming, providing you with the insight and technical craftsmanship you need to be a productive member of any Python project. As a Python developer, know the tools, basic idioms and of course the ins and outs of the language, the standard library and other modules to be able to jump into most projects.

What you will learn:

- Learn the language of Python itself
- Get a start on the Python standard library
- Learn how to integrate 3rd party libraries
- Develop libraries on your own
- Become familiar with the basics of Python testing

Publisher's page → <https://www.packtpub.com/application-development/python-apprentice>

Learn Python in 7 Days → <https://www.packtpub.com/application-development/learn-python-7-days>

Mohit, Bhaskar N. Das

ISBN 13: 9781787288386 Packt Publishing 280 pages (May 2017)

Book Overview:

Python is a great language to get started in the world of programming and application development. This book will help you to take your skills to the next level having a good knowledge of the fundamentals of Python.

Begin with the absolute foundation, covering the basic syntax, type variables and operators. Then you can move on to concepts like statements, arrays, operators, string processing and I/O handling. You'll be able to learn how to operate tuples and understand the functions and methods of lists. Develop a deep understanding of list and tuples and learn python dictionary. As you progress through the book, you'll learn about function parameters and how to use control statements with the loop. You'll further learn how to create modules and packages, storing of data as well as handling errors. Later, dive into advanced level concepts such as Python collections and how to use class, methods, objects in Python.

By the end of this book, you will be able to take your skills to the next level having a good knowledge of the fundamentals of Python.

What you will learn:

- Use if else statement with loops and how to break, skip the loop
- Get acquainted with python types and its operators
- Create modules and packages
- Learn slicing, indexing and string methods
- Explore advanced concepts like collections, class and objects
- Learn dictionary operation and methods
- Discover the scope and function of variables with arguments and return value

Publisher's page → <https://www.packtpub.com/application-development/learn-python-7-days>

What You Need to Know about Machine Learning → [https://www.packtpub.com/packt/free-ebook/what-you-need-know-about-machine-learning2 \(Free eBook\)](https://www.packtpub.com/packt/free-ebook/what-you-need-know-about-machine-learning2)

By Gabriel A. Cánepa

ISBN 13: 9781786466211 Packt Publishing (July 2016)

Book Overview:

This book offers you the perfect place to lay the foundation for your work in the world of Machine Learning, providing the basic understanding, knowledge, and skills that you can build on with experience and time.

- Discover the various forms Machine Learning, and take that which will benefit you the most
- Pick up the Python tools you need to know: from pandas to scikit-learn
- Learn the relationship between machine learning and big data
- Understand and identify potential real-world scenarios where machine learning can be applied
- Lay the foundations to get started in the wide world of machine learning

Publisher's Page → <https://www.packtpub.com/packt/free-ebook/what-you-need-know-about-machine-learning2>

What You Need to Know about Python → [https://www.packtpub.com/packt/free-ebook/what-you-need-know-about-python2 \(Free eBook\)](https://www.packtpub.com/packt/free-ebook/what-you-need-know-about-python2)

By Pierluigi Riti

ISBN 13: 9781786465566 Packt Publishing (March 2016)

Book Overview:

With this book you'll experience the basics of Python, what it can be used for, and get a clear direction in what you need to really get the most from this great language.

- Discover the state of Python in 2016 and what it means for you
- Begin learning about the basics and unique features of Python
- Take on the world of cloud and web development with Django
- Let Jupyter handle your data visualization
- Get an introduction to the world of [DevOps](https://wiki.python.org/moin/DevOps) → <https://wiki.python.org/moin/DevOps>

[Publisher's Page](https://www.packtpub.com/packt/free-ebook/what-you-need-know-about-python2) → <https://www.packtpub.com/packt/free-ebook/what-you-need-know-about-python2>

[Learn Python](https://www.manning.com/books/get-programming) → <https://www.manning.com/books/get-programming>

By Ana Bell

- Manning Publications Co
- ISBN 9781617293788, 456 pages, printed in black & white
- Published Spring 2018

Book overview:

When you're ready to learn to program, Python is a great language to start with. It has a straightforward syntax, you'll find it easy to learn, and it's extremely versatile. Once you master the basics, you'll appreciate Python's very strong online community where you can continue learning and tinkering with more advanced programming techniques. All you need is some help to get going. Learning to program with Python doesn't have to be difficult, in fact it can even be fun. This book will get you started!

Learn Python is an introduction to programming using one of the most popular programming languages. The first few chapters give you a quick background to programming concepts and a hands-on guide to setting up your programming environment. You will then be gently introduced to the basics of programming, building up to writing your first programs! This easy-to-follow tutorial is full of exercises to practice and reinforce each new concept, as well as to give you confidence that you are ready to move on to the next lesson. As you progress, you'll learn programming topics and concepts that are ubiquitous across almost all other programming languages. By the end of the book, you'll have a solid grasp of how to write programs as well as programming best practices.

Who this book is written for:

Anyone without programming experience who wants an easy-to-follow book that guides you through short lessons and exercises.

Chapter 4 available for free: [Variables and Expressions: Giving Names and Values to Things](https://manning-content.s3.amazonaws.com/download/e/2d021c6-d4ed-40e5-8f6d-084f92ccbb86/SampleCh04.pdf) → <https://manning-content.s3.amazonaws.com/download/e/2d021c6-d4ed-40e5-8f6d-084f92ccbb86/SampleCh04.pdf>

Chapter 30 available for free: [Making your own Object Types](https://manning-content.s3.amazonaws.com/download/3/145ee71-1acd-4e02-a9b0-fb0f1892485e/SampleCh30.pdf) → <https://manning-content.s3.amazonaws.com/download/3/145ee71-1acd-4e02-a9b0-fb0f1892485e/SampleCh30.pdf>

[The Quick Python Book, 3rd Edition](https://www.manning.com/books/the-quick-python-book-third-edition) → <https://www.manning.com/books/the-quick-python-book-third-edition>

By Naomi Ceder

- Manning Publications Co
- ISBN 9781617294037 400 pages (estimated)
- MEAP began December 2016, Publication in Spring 2017 (estimated)

MEAP: early access to chapters as they are being written, available at the Manning Publications web site: [The Quick Python Book, 3rd Edition](https://www.manning.com/books/the-quick-python-book-third-edition) → <https://www.manning.com/books/the-quick-python-book-third-edition>

Book overview:

This third revision of Manning's popular The Quick Python Book offers a clear, crisp updated introduction to the elegant Python programming language and its famously easy-to-read syntax. Written for programmers new to Python, this latest edition includes new exercises throughout. It covers features common to other languages concisely, while introducing Python's comprehensive standard functions library and unique features in detail.

After exploring Python's syntax, control flow, and basic data structures, the book shows how to create and deploy full applications and larger code libraries. It addresses established Python features as well as the advanced object-oriented options available in Python 3. Along the way, you'll survey the current Python development landscape, including Pythonic best practices, data extraction and cleaning, database access, and web frameworks.

Who this book is written for:

This book is for someone who knows how to program, who wants to learn Python quickly and efficiently.

Chapter 1 available for free: [About Python](https://www.manning.com/books/the-quick-python-book-third-edition#downloads) → <https://www.manning.com/books/the-quick-python-book-third-edition#downloads>

[Python Machine Learning Cookbook](https://www.packtpub.com/big-data-and-business-intelligence/python-machine-learning-cookbook) → <https://www.packtpub.com/big-data-and-business-intelligence/python-machine-learning-cookbook>

By Prateek Joshi

ISBN 13: 9781786464477 Packt Publishing 304 pages (June 2016)

Book overview:

Machine learning is becoming increasingly pervasive in the modern data-driven world. It is used extensively across many fields such as search engines, robotics, self-driving cars, and more. With this book, you will learn how to perform various machine learning tasks in different environments. We'll start by exploring a range of real-life scenarios where machine learning can be used, and look at various building blocks. Throughout the book, you'll use a wide variety of machine learning algorithms to solve real-world problems and use Python to implement these algorithms. You'll discover how to deal with various types of data and explore the differences between machine learning paradigms such as supervised and unsupervised learning. We also cover a range of regression techniques, classification algorithms, predictive modeling, data visualization techniques, recommendation engines, and more with the help of real-world examples.

- Explore classification algorithms and apply them to the income bracket estimation problem
- Use predictive modeling and apply it to real-world problems
- Understand how to perform market segmentation using unsupervised learning
- Explore data visualization techniques to interact with your data in diverse ways
- Find out how to build a recommendation engine
- Understand how to interact with text data and build models to analyze it
- Work with speech data and recognize spoken words using Hidden Markov Models
- Analyze stock market data using Conditional Random Fields
- Work with image data and build systems for image recognition and biometric face recognition
- Grasp how to use deep neural networks to build an optical character recognition system

Who this book is written for:

This book is for Python programmers who are looking to use machine-learning algorithms to create real-world applications. This book is friendly to Python beginners, but familiarity with Python programming would certainly be useful to play around with the code.

[Publisher's page](https://www.packtpub.com/big-data-and-business-intelligence/python-machine-learning-cookbook) → <https://www.packtpub.com/big-data-and-business-intelligence/python-machine-learning-cookbook>

[Modular Programming with Python](https://www.packtpub.com/application-development/modular-programming-python) → <https://www.packtpub.com/application-development/modular-programming-python>

By Erik Westra

ISBN 13: 9781785884481 Packt Publishing 246 pages (May 2016)

Book overview:

This book will help readers develop readable, reliable, and maintainable programs in Python. Starting with an introduction to the concept of modules and packages, this book shows how you can use these building blocks to organize a complex program into logical parts and make sure those parts are working correctly together. Using clearly written, real-world examples, this book demonstrates how you can use modular techniques to build better programs. A number of common modular programming patterns are covered, including divide-and-conquer, abstraction, encapsulation, wrappers and extensibility. You will also learn how to test your modules and packages, how to prepare your code for sharing with other people, and how to publish your modules and packages on [GitHub](https://wiki.python.org/moin/GitHub) → <https://wiki.python.org/moin/GitHub> and the Python Package Index so that other people can use them. Finally, you will learn how to use modular design techniques to be a more effective programmer.

- Learn how to use modules and packages to organize your Python code
- Understand how to use the import statement to load modules and packages into your program
- Use common module patterns such as abstraction and encapsulation to write better programs
- Discover how to create self-testing Python packages
- Create reusable modules that other programmers can use
- Learn how to use [GitHub](https://wiki.python.org/moin/GitHub) → <https://wiki.python.org/moin/GitHub> and the Python Package Index to share your code with other people
- Make use of modules and packages that others have written
- Use modular techniques to build robust systems that can handle complexity and changing requirements over time

Who this book is written for:

This book is intended for beginner to intermediate level Python programmers who wish to learn how to use modules and packages within their programs. While readers must understand the basics of Python programming, no knowledge of modular programming techniques is required.

[Publisher's page](https://www.packtpub.com/application-development/modular-programming-python) → <https://www.packtpub.com/application-development/modular-programming-python>

[The Coder's Apprentice: Learning programming with Python 3](http://www.spronck.net/pythonbook) → <http://www.spronck.net/pythonbook>

By Pieter Spronck

~400 pages (July 2016)

Book overview:

"The Coder's Apprentice" is a course book, aimed at teaching Python 3 to students and teenagers who are completely new to programming. Contrary to many of the other books that teach Python programming, this book assumes no previous knowledge of programming on the part of the students, and contains numerous exercises that allow students to train their programming skills.

The book covers all imperative programming and object oriented programming aspects of Python 3, including:

- Installing, writing, and running Python
- Expressions, variables, conditions, loops, and functions
- Strings, tuples, lists, dictionaries, and sets
- Text files, binary files, and exceptions
- Object orientation, operator overloading, inheritance, iterators, and generators
- Regular expressions

The book also spends time on algorithm design and writing effective code.

The book is freely available as PDF.

[Learning Python for Forensics → https://www.packtpub.com/networking-and-servers/learning-python-forensics](https://www.packtpub.com/networking-and-servers/learning-python-forensics)

By Preston Miller, Chapin Bryce

ISBN 13: 9781783285235 Packt Publishing 488 pages (May 2016)

Book overview:

This book will illustrate how and why you should learn Python to strengthen your analysis skills and efficiency as you creatively solve real-world problems through instruction-based tutorials. The tutorials use an interactive design, giving you experience of the development process so you gain a better understanding of what it means to be a forensic developer.

- Discover how to perform Python script development.
- Update yourself by learning the best practices in forensic programming.
- Build scripts through an iterative design.
- Explore the rapid development of specialized scripts.
- Understand how to leverage forensic libraries developed by the community.
- Design flexibly to accommodate present and future hurdles.
- Conduct effective and efficient investigations through programmatic pre-analysis.

Who this book is written for:

If you are a forensics student, hobbyist, or professional that is seeking to increase your understanding in forensics through the use of a programming language, then this book is for you. You are not required to have previous experience in programming to learn and master the content within this book. This material, created by forensic professionals, was written with a unique perspective and understanding of examiners who wish to learn programming.

[Publisher's page → https://www.packtpub.com/networking-and-servers/learning-python-forensics](https://www.packtpub.com/networking-and-servers/learning-python-forensics)

[Practical Data Analysis Cookbook → https://www.packtpub.com/big-data-and-business-intelligence/practical-data-analysis-cookbook](https://www.packtpub.com/big-data-and-business-intelligence/practical-data-analysis-cookbook)

By Tomasz Drabas

ISBN 13: 9781783551668 Packt Publishing 384 pages (April 2016)

Book overview:

This book provides a rich set of independent recipes that dive into the world of data analytics and modeling using a variety of approaches, tools, and algorithms. You will learn the basics of data handling and modeling, and will build your skills gradually toward more advanced topics such as simulations, raw text processing, social interactions analysis, and more.

- Read, clean, transform, and store your data using Pandas and [OpenRefine](https://wiki.python.org/moin/OpenRefine) → <https://wiki.python.org/moin/OpenRefine>.
- Understand your data and explore the relationships between variables using Pandas and D3.js.
- Explore a variety of techniques to classify and cluster outbound marketing campaign calls data of a bank using Pandas, mlp, [NumPy](https://wiki.python.org/moin/NumPy) → <https://wiki.python.org/moin/NumPy>, and Statsmodels.
- Reduce the dimensionality of your dataset and extract the most important features with pandas, [NumPy](https://wiki.python.org/moin/NumPy) → <https://wiki.python.org/moin/NumPy>, and mlp.
- Predict the output of a power plant with regression models and forecast water flow of American rivers with time series methods using pandas, [NumPy](https://wiki.python.org/moin/NumPy) → <https://wiki.python.org/moin/NumPy>, Statsmodels, and scikit-learn.
- Explore social interactions and identify fraudulent activities with graph theory concepts using NetworkX and Gephi.

- Scrape Internet web pages using urllib and [BeautifulSoup](https://wiki.python.org/moin/BeautifulSoup) → <https://wiki.python.org/moin/BeautifulSoup> and get to know natural language processing techniques to classify movies ratings using NLTK .
- Study simulation techniques in an example of a gas station with agent-based modeling.

Who this book is written for:

This hands-on recipe guide is divided into three sections that tackle and overcome real-world data modeling problems faced by data analysts/scientist in their everyday work. Each independent recipe is written in an easy-to-follow and step-by-step fashion.

[Publisher's page](https://www.packtpub.com/big-data-and-business-intelligence/practical-data-analysis-cookbook) → <https://www.packtpub.com/big-data-and-business-intelligence/practical-data-analysis-cookbook>

[Python Projects for Kids](https://www.packtpub.com/application-development/python-projects-kids) → <https://www.packtpub.com/application-development/python-projects-kids>

Jessica Ingrassellino

ISBN: 9781782175063

192 pages, April 2016, Packt Publishing

Kids are always the most fast-paced and enthusiastic learners, and are naturally willing to build stuff that looks like magic at the end (when it works!). Programming can be one such magic. Being able to write a program that works helps them feel they've really achieved something. Kids today are very tech-savvy and cannot wait to enter the fast-paced digital world.

Because Python is one of the most popular languages and has a syntax that is quite simple to understand, even kids are eager to use it as a stepping stone to learning programming languages.

This book covers projects that are simple and fun, and teach kids how to write Python code that works.

The book will teach the basics of Python programming, installation, and so on and then will move on to projects. A total of three projects, with each and every step explained carefully, without any assumption of previous experience.

What You Will Learn:

- Start fiddling with Python's variables, build functions and interact with users
- Build your own calculator using the Math Library
- Train Python to make logical decisions
- Work with moving 2D objects on-screen
- Understand the Pygame Library and build your very own game!

This book is for kids (aged 10 and over). This is book is intended for absolute beginners who lack any knowledge of computing or programming languages and want to get started in the world of programming.

[Learn to Program with Minecraft](https://www.nostarch.com/pythonwithminecraft) → <https://www.nostarch.com/pythonwithminecraft>

Craig Richardson

ISBN: 978-1-59327-670-6

304 pages, December 2015, No Starch Press

You've bested creepers, traveled deep into caves, and maybe even gone to The End and back—but have you ever transformed a sword into a magic wand? Built a palace in the blink of an eye? Designed your own color-changing disco dance floor?

In *Learn to Program with Minecraft®*, you'll do all this and more with the power of Python, a free language used by millions of professional and first-time programmers!

Begin with some short, simple Python lessons and then use your new skills to modify Minecraft to produce instant and totally awesome results. Learn how to customize Minecraft to make mini-games, duplicate entire buildings, and turn boring blocks into gold.

You'll also write programs that:

- Take you on an automated teleportation tour around your Minecraft world
- Build massive monuments, pyramids, forests, and more in a snap!
- Make secret passageways that open when you activate a hidden switch
- Create a spooky ghost town that vanishes and reappears elsewhere
- Show exactly where to dig for rare blocks
- Cast a spell so that a cascade of flowers (or dynamite if you're daring!) follows your every move
- Make mischief with dastardly lava traps and watery curses that cause huge floods

Whether you're a Minecraft megafan or a newbie, you'll see Minecraft in a whole new light while learning the basics of programming. Sure, you could spend all day mining for precious resources or building your mansion by hand, but with the power of Python, those days are over!

Requires: Windows 7 or later; OS X 10.10 or later; or a Raspberry Pi.

Uses Python 3

Automate the Boring Stuff with Python → <https://www.nostarch.com/automatestuff>

Al Sweigart

ISBN: 978-1-59327-599-0

504 pages, April 2015, No Starch Press

If you've ever spent hours renaming files or updating hundreds of spreadsheet cells, you know how tedious tasks like these can be. But what if you could have your computer do them for you?

In *Automate the Boring Stuff with Python*, you'll learn how to use Python to write programs that do in minutes what would take you hours to do by hand—no prior programming experience required. Once you've mastered the basics of programming, you'll create Python programs that effortlessly perform useful and impressive feats of automation to:

- Search for text in a file or across multiple files
- Create, update, move, and rename files and folders
- Search the Web and download online content
- Update and format data in Excel spreadsheets of any size
- Split, merge, watermark, and encrypt PDFs
- Send reminder emails and text notifications
- Fill out online forms

Step-by-step instructions walk you through each program, and practice projects at the end of each chapter challenge you to improve those programs and use your newfound skills to automate similar tasks.

Don't spend your time doing work a well-trained monkey could do. Even if you've never written a line of code, you can make your computer do the grunt work. Learn how in *Automate the Boring Stuff with Python*.

Note: The programs in this book are written to run on Python 3.

Doing Math with Python → <https://www.nostarch.com/doingmathwithpython>

Amit Saha

ISBN: 978-1-59327-640-9

264 pages, August 2015, No Starch Press

In *Doing Math with Python* you'll learn to how to use the Python programming language as a tool to delve into math concepts. Python is easy to learn, and it's perfect for exploring topics like statistics, geometry, probability, and calculus. You'll learn to write programs to find derivatives, solve equations graphically, manipulate algebraic expressions, even examine projectile motion.

Rather than crank through tedious calculations by hand, you'll learn how to use Python functions and modules to handle the number crunching while you focus on the principles behind the math. Exercises throughout teach fundamental programming concepts, like using functions, handling user input, and reading and manipulating data. As you learn to think computationally, you'll discover new ways to explore and think about math, and gain valuable programming skills that you can use to continue your study of math and computer science.

If you're interested in math but have yet to dip into programming, you'll find that Python makes it easy to go deeper into the subject—let Python handle the tedious work while you spend more time on the math.

Suggested for ages 14 and up.

[Python Crash Course](https://www.nostarch.com/pythoncrashcourse) → <https://www.nostarch.com/pythoncrashcourse>

Eric Matthes

ISBN: 978-1-59327-603-4

624 pages, November 2015, No Starch Press

Solve Problems, Make Things, and Do Cool Stuff!

Python Crash Course is a fast-paced and thorough introduction to Python that will have you writing programs, solving problems, and making things that work, fast.

In the first half of the book you'll learn basic programming concepts, from installing Python to testing code, including how to use a text editor, write clean and readable code, store data in lists and dictionaries, create classes that simulate real-life objects, and write loops to perform actions on your data. You'll also learn how to make your programs interactive and test your code safely before adding it to your project. The exercises in chapters will help cement your learning so you can put that knowledge into practice in the second half, with three substantial projects: a Space-Invaders-inspired arcade game, data visualization with Python's super-handy libraries, and a simple Web app you can deploy online.

As you work through *Python Crash Course* you'll learn how to:

- Make games that respond to buttons and mouse clicks, and grow more difficult as you progress
- Work with data and generate interactive visualizations
- Utilize the most useful Python libraries and tools, including: matplotlib, numpy, and pygal
- Create and customize Web apps and deploy them safely online
- Deal with mistakes and errors, to better prepare you for solving your own programming problems later

Python Crash Course will teach you just what you need to know to start programming meaningful things quickly.

Uses Python 2 and 3

[Python for Kids](https://www.nostarch.com/pythonforkids) → <https://www.nostarch.com/pythonforkids>

Jason Briggs

ISBN: 978-1-59327-407-8

344 pages, December 2012, No Starch Press

The code in this book runs on almost anything: Windows, Mac, Linux, even an OLPC laptop or Raspberry Pi!

Python is a powerful, expressive programming language that's easy to learn and fun to use! But books about learning to program in Python can be kind of dull, gray, and boring, and that's no fun for anyone.

Python for Kids brings Python to life and brings you (and your parents) into the world of programming. The ever-patient Jason R. Briggs will guide you through the basics as you experiment with unique (and often hilarious) example programs that feature ravenous monsters, secret agents, thieving ravens, and more. New terms are

defined; code is colored, dissected, and explained; and quirky, full-color illustrations keep things on the lighter side.

Chapters end with programming puzzles designed to stretch your brain and strengthen your understanding. By the end of the book you'll have programmed two complete games: a clone of the famous Pong and "Mr. Stick Man Races for the Exit"—a platform game with jumps, animation, and much more.

As you strike out on your programming adventure, you'll learn how to:

- Use fundamental data structures like lists, tuples, and maps
- Organize and reuse your code with functions and modules
- Use control structures like loops and conditional statements
- Draw shapes and patterns with Python's turtle module
- Create games, animations, and other graphical wonders with tkinter

Why should serious adults have all the fun? *Python for Kids* is your ticket into the amazing world of computer programming.

For kids ages 10+ (and their parents).

Featuring original artwork by Miran Lipovača!

Teach Your Kids to Code → <https://www.nostarch.com/teachkids>

Bryson Payne

ISBN: 978-1-59327-614-0

336 pages, April 2015, No Starch Press

Teach Your Kids to Code is a parent's and teacher's guide to teaching kids basic programming and problem solving using Python, the powerful language used in college courses and by tech companies like Google and IBM.

Step-by-step explanations will have kids learning computational thinking right away, while visual and game-oriented examples hold their attention. Friendly introductions to fundamental programming concepts such as variables, loops, and functions will help even the youngest programmers build the skills they need to make their own cool games and applications.

Whether you've been coding for years or have never programmed anything at all, *Teach Your Kids to Code* will help you show your young programmer how to:

- Explore geometry by drawing colorful shapes with Turtle graphics
- Write programs to encode and decode messages, play Rock-Paper-Scissors, and calculate how tall someone is in Ping-Pong balls
- Create fun, playable games like War, Yahtzee, and Pong
- Add interactivity, animation, and sound to their apps

Teach Your Kids to Code is the perfect companion to any introductory programming class or after-school meet-up, or simply your educational efforts at home. Spend some fun, productive afternoons at the computer with your kids—you can all learn something!

Python for complete beginners → <http://www.amazon.com/Python-complete-beginners-friendly-experience/dp/1514376989/>

Dr. Martin Jones

ISBN: 978-1514376980

246 pages, 2015

Getting started with programming can be an intimidating challenge. Most books and tutorials assume lots of previous knowledge, skip over jargon and new concepts, and use examples that only make sense if you already understand programming.

This book is different. It gives you an introduction to programming in Python from the ground up, starting with tips on installation and setting up your programming environment, and moving through the core parts of the Python language in a logical order. Dr. Jones has drawn on his many years experience teaching programming to produce a book that will guide you through the language step by step in simple terms.

The book doesn't assume any previous knowledge, and introduces fundamental programming concepts like variables, loops and functions using simple terms and easy-to-follow examples that you can run and modify.

The book takes a unique approach to practical exercises. Rather than simply presenting you with the solutions, it shows you how large, complex programs are gradually built up from simple building blocks, explaining the role of every line. You can download the examples and exercise solutions - edit, modify and run them yourself.

Examples and exercise files available at the author's website: [Python for complete beginners](http://pythonforcompletebeginners.com/) → <http://pythonforcompletebeginners.com/>

[Geoprocessing with Python](http://www.manning.com/garrard/) → <http://www.manning.com/garrard/>

Chris Garrard

Manning Publications ISBN: 9781617292149, 400 pages, MEAP Began May 2014

Geospatial data is hard to ignore. Nearly every car, phone, or camera has a GPS sensor, and aerial photos, satellite imagery, and data representing political boundaries, roads, rivers, and streams are available for free download from many websites. Geoprocessing is the science of reading, analyzing, and presenting geospatial data programmatically. The Python language, along with dozens of open source libraries and tools, makes it possible to take on professional geoprocessing tasks without investing in expensive proprietary packages like ArcGIS and [MapInfo](https://wiki.python.org/moin/MapInfo) → <https://wiki.python.org/moin/MapInfo>.

Geoprocessing with Python teaches you how to use the Python programming language along with free and open source tools to read, write, and process geospatial data. You'll learn how to access available data sets to make maps or perform your own analyses using free and open source tools like the GDAL, Shapely, and Fiona Python modules. You'll master core practices like handling multiple vector file formats, editing and manipulating geometries, applying spatial and attribute filters, working with projections, and performing basic analyses on vector data. You'll also learn how to create geospatial data, rather than just consuming it. The book also covers how to manipulate and analyze raster data, such as aerial photographs, satellite images, and digital elevation models.

Chapter one available for free: [Introduction](http://www.manning.com/garrard/GeoPython_MEAP_ch01.pdf) → http://www.manning.com/garrard/GeoPython_MEAP_ch01.pdf

MEAP Available at Manning Publications's Site: [Geoprocessing with Python](http://www.manning.com/garrard/) → <http://www.manning.com/garrard/>

[Algorithms and Programming Course](https://books.google.com.br/books?id=HvSsCAAAQBAJ) → <https://books.google.com.br/books?id=HvSsCAAAQBAJ>

Guilherme Soares Lima

(2015, 100 pages)

The book is a gentle introduction to algorithms and programming with Python, specially for those that don't know how to program on any language. It has a progressive approach, it covers most of basic topics like programming logics, repetition structures. It is a small book, but yet a powerful tool for learning. It helps get good orientation from the beginning. It has only 100 pages and it's free.

Fundamentals of Python: First Programs → <http://www.cengagebrain.com/shop/isbn/9781111822705>

Ken Lambert

ISBN 13: 978-1-111-82270-5 (2012, 510 pages)

An introduction to programming and problem solving with Python (CS1). Covers algorithmic problem solving, basic data structures, functional design, and object-oriented design. Also includes an introduction to turtle graphics, image processing, GUIs, and networks.

1. Introduction.
2. Data Types and Expressions.
3. Control Statements.
4. Strings and Text Files.
5. Lists and Dictionaries.
6. Design with Functions.
7. Simple Graphics and Image Processing.
8. Design with Classes.
9. Graphical User Interfaces.
10. Multithreading, Networks, and Client/Server Programming.

Link to example programs and other information at the author's website → <http://home.wlu.edu/~lambertk/python/cs1python/index.html>.

Fundamentals of Python: Data Structures → <http://www.cengagebrain.com/shop/isbn/9781285752006>

Ken Lambert

ISBN 13: 978-1-285-75200-6 (2014, 496 pages)

An introduction to Python with data structures (CS2). Covers the design of collection classes with polymorphism and inheritance, multiple implementations of collection interfaces, and the analysis of space/time tradeoffs of different collections, all within a realistic collection framework.

1. Basic Python Programming.
2. An Overview of Collections.
3. Searching, Sorting, and Complexity Analysis.
4. Arrays and Linked Structures.
5. Interfaces, Implementations, and Polymorphism.
6. Inheritance and Abstract Classes.
7. Stacks.
8. Queues.
9. Lists.
10. Trees.
11. Sets and Dictionaries.
12. Graphs.

Link to example programs and other information at the author's website → <http://home.wlu.edu/~lambertk/python/cs2python/index.html>.

Easy GUI Programming in Python → <http://home.wlu.edu/~lambertk/breezypythongui/easyguibook.html>

Ken Lambert

ISBN-13: 978-0-9859567-0-7 (August, 2012, 110 pages, ebook only)

An introduction to GUI programming in Python. Topics include the basics of window layout, widget configuration, and responding to user events. The book uses a toolkit, [breezypythongui](http://home.wlu.edu/~lambertk/breezypythongui/index.html) → <http://home.wlu.edu/~lambertk/breezypythongui/index.html>, to explore GUI concepts and resources within a simple coding framework.

1. The Hello World Program
2. Windows, Layouts, and Window Components
3. Command Buttons and Responding to Events
4. Input and Output with Data Fields
5. Error Handling and Message Boxes
6. The Model/View/Controller Pattern
7. Check Buttons, Radio Buttons, and Menus
8. Scrolling List Boxes
9. Text Areas and File Dialogs
10. Dialogs
11. Canvases and Graphics Operations
12. Responding to Mouse Events in Canvases

Link to example programs and ebook vendor sites at the author's [website](http://home.wlu.edu/~lambertk/breezypythongui/easyguibook.html) → <http://home.wlu.edu/~lambertk/breezypythongui/easyguibook.html>.

[Building Machine Learning Systems with Python](http://www.packtpub.com/building-machine-learning-systems-with-python/book) → <http://www.packtpub.com/building-machine-learning-systems-with-python/book>

By Willi Richert, Luis Pedro Coelho

ISBN 13: 9781782161400 Packt Publishing 290 pages (July 2013)

Master the art of machine learning with Python and build effective machine learning systems with this intensive hands-on guide

- Master Machine Learning using a broad set of Python libraries and start building your own Python-based ML systems
- Covers classification, regression, feature engineering, and much more guided by practical examples
- A scenario-based tutorial to get into the right mind-set of a machine learner (data exploration) and successfully implement this in your new or existing projects

Who this book was written for

This book is for Python programmers who are beginners in machine learning, but want to learn Machine learning. Readers are expected to know Python and be able to install and use open-source libraries. They are not expected to know machine learning, although the book can also serve as an introduction to some Python libraries for readers who know machine learning. This book does not go into the detail of the mathematics behind the algorithms.

This book primarily targets Python developers who want to learn and build machine learning in their projects, or who want to provide machine learning support to their existing projects, and see them getting implemented effectively.

[Publisher's page](http://www.packtpub.com/building-machine-learning-systems-with-python/book) → <http://www.packtpub.com/building-machine-learning-systems-with-python/book>

[Learning to Program Using Python](https://www.createspace.com/3611970) → <https://www.createspace.com/3611970>

Cody Jackson

ISBN: 1461182050 (November 2011, 230 pages)

An introduction to computer programming, using the easy, yet powerful, Python programming language. Python, a cross-platform language used by such organizations as Google and NASA, lets you work quickly and efficiently, allowing you to concentrate on your work rather than the language. The core Python language (both versions 2.x and 3.x) is discussed, as well as an introduction to graphical user interface creation.

The ideas covered in this book provide the reader with many major programming topics, applicable to a wide variety of programming languages. After reading this book, the reader should be able to quickly create simple to medium-level programs and be prepared to tackle more complex programming tasks.

[Print version](https://www.createspace.com/3611970) → <https://www.createspace.com/3611970>

[Kindle version \(no DRM\)](http://www.amazon.com/dp/B0063ZM610) → <http://www.amazon.com/dp/B0063ZM610>

The book's website → <http://python-ebook.blogspot.com/> also provides alternative electronic versions for free, without DRM. In addition, source code files for programs shown in the book are also available.

[Python 2.6 Text Processing Beginner's Guide](https://www.packtpub.com/python-2-6-text-processing-beginners-guide/book) → <https://www.packtpub.com/python-2-6-text-processing-beginners-guide/book>

Jeff McNeil → <https://wiki.python.org/moin/McNeil>

ISBN 13: 978-1-84951-212-1 Packt Publishing 380 pages (December 2010)

The easiest way to learn how to manipulate text with Python:

- The easiest way to learn text processing with Python
- Deals with the most important textual data formats you will encounter
- Learn to use the most popular text processing libraries available for Python
- Packed with examples to guide you through

[Publisher's page](https://www.packtpub.com/python-2-6-text-processing-beginners-guide/book) → <https://www.packtpub.com/python-2-6-text-processing-beginners-guide/book>

Learning Python, 5th Edition

(Mark Lutz, O'Reilly Media, June 2013, 1600 pages)

An in-depth, tutorial introduction to Python core language fundamentals, based on 260 live classes taught by the author. This edition is updated to cover both Python 3.X and 2.X. It is specifically based on 3.3 and 2.7, but is applicable to other releases.

Links: [author](http://www.rmi.net/~lutz/about-lp5e.html) → <http://www.rmi.net/~lutz/about-lp5e.html>, [publisher](http://shop.oreilly.com/product/0636920028154.do) → <http://shop.oreilly.com/product/0636920028154.do>.

Programming Python, 4th Edition

(Mark Lutz, O'Reilly Media, December 2010, 1600 pages)

An in-depth, tutorial introduction to common Python application programming domains, and a follow-up to the core language coverage of Learning Python. This edition is updated to use Python 3.X (3.1 and 3.2 specifically), but is still largely applicable to most 2.X readers.

Links: [author](http://www.rmi.net/~lutz/about-pp4e.html) → <http://www.rmi.net/~lutz/about-pp4e.html>, [publisher](http://shop.oreilly.com/product/9780596158118.do) → <http://shop.oreilly.com/product/9780596158118.do>.

Python Pocket Reference, 4th Edition

(Mark Lutz, O'Reilly Media, September 2009, 200 pages)

A reference-only book, designed to serve as a companion to both Learning Python and Programming Python. This edition is updated to cover both Python 3.X and 2.X. It is specifically based on 3.1 and 2.6, but is applicable to other releases.

Links: [author](http://www.rmi.net/~lutz/about_pyref4e.html) → http://www.rmi.net/~lutz/about_pyref4e.html, [publisher](http://shop.oreilly.com/product/9780596158095.do) → <http://shop.oreilly.com/product/9780596158095.do>.

[Head First Python](http://oreilly.com/catalog/9781449382674/) → <http://oreilly.com/catalog/9781449382674/> by Paul Barry

- Published by: O'Reilly Media, November 2010, 496 pages.
- Print ISBN: 978-1-4493-8267-4, ISBN 10: 1-4493-8267-3
- Ebook ISBN: 978-1-4493-8268-1, ISBN 10: 1-4493-8268-1

Head First Python

Ever wished you could learn Python from a book? Head First Python helps you learn the language through a unique method that goes beyond syntax and how-to manuals. You'll quickly grasp Python's fundamentals, then move on to persistence, exception handling, web development, SQLite, data wrangling, and Google App Engine. You'll also learn how to write mobile apps for Android, all thanks to the power that Python gives you. Head First Python is a complete learning experience that will help you become a bona fide Python programmer.

This book is designed to get you up to speed with Python as quickly as possible.

See <http://python.itcarlow.ie> → <http://python.itcarlow.ie> and <http://www.headfirstlabs.com/books/hfpython/> for more details.

[Practical Programming](http://www.pragprog.com/titles/gwpy2/practical-programming) → <http://www.pragprog.com/titles/gwpy2/practical-programming> by Jennifer Campbell, Paul Gries, and Jason Montojo

Practical Programming (2nd edition): An Introduction to Computer Science Using Python 3

- ISBN: 978-1-93778-545-1
- 350 pages, August 2013

Computers are used in every part of science from ecology to particle physics. This introduction to computer science continually reinforces those ties by using real-world science problems as examples. Anyone who has taken a high school science class will be able to follow along as the book introduces the basics of programming, then goes on to show readers how to work with databases, download data from the web automatically, build graphical interfaces, and most importantly, how to think like a professional programmer.

[Practical Programming](http://www.pragprog.com/titles/gwpy/practical-programming) → <http://www.pragprog.com/titles/gwpy/practical-programming> by Jennifer Campbell, Paul Gries, Jason Montojo, and Greg Wilson

Practical Programming: An Introduction to Computer Science Using Python

- ISBN: 978-1-93435-627-2
- 350 pages, Apr 2009

Computers are used in every part of science from ecology to particle physics. This introduction to computer science continually reinforces those ties by using real-world science problems as examples. Anyone who has taken a high school science class will be able to follow along as the book introduces the basics of programming, then goes on to show readers how to work with databases, download data from the web automatically, build graphical interfaces, and most importantly, how to think like a professional programmer.

[Python 3 Object Oriented Programming](https://www.packtpub.com/python-3-object-oriented-programming/book) → <https://www.packtpub.com/python-3-object-oriented-programming/book>

Dusty Phillips

ISBN-13: 978-1-849511-26-1 Packt Publishing 404 pages (July 2010)

Harness the power of Python 3 objects:

Overview of Python 3 Object Oriented Programming

- Learn how to do Object Oriented Programming in Python using this step-by-step tutorial
- Design public interfaces using abstraction, encapsulation, and information hiding
- Turn your designs into working software by studying the Python syntax
- Raise, handle, define, and manipulate exceptions using special error objects
- Implement Object Oriented Programming in Python using practical examples

Who this book is written for If you're new to Object Oriented Programming techniques, or if you have basic Python skills and wish to learn in depth how and when to correctly apply Object Oriented Programming in Python, this is the book for you.

If you are an object-oriented programmer for other languages, you too will find this book a useful introduction to Python, as it uses terminology you are already familiar with.

Python 2 programmers seeking a leg up in the new world of Python 3 will also find the book beneficial, and you need not necessarily know Python 2.

[Publisher's page](https://www.packtpub.com/python-3-object-oriented-programming/book) → <https://www.packtpub.com/python-3-object-oriented-programming/book>

[The Practice of Computing Using Python](http://www.pearsonhighered.com/educator/product/Practice-of-Computing-using-Python-The/9780136110675.page) → <http://www.pearsonhighered.com/educator/product/Practice-of-Computing-using-Python-The/9780136110675.page> by Bill Punch and Rich Enbody

- ISBN-10: 0136110673
- ISBN-13: 9780136110675
- Addison-Wesley, 696 pp, 02/25/2010

The Computer Science Department of Michigan State University converted their Introduction to Programming Course [CSE 231](http://www.cse.msu.edu/~cse231) → <http://www.cse.msu.edu/~cse231> to Python in the Fall of 2007. One of the products of this change was this textbook, written as a general introduction to CS1 using Python. The book adopts the theme of "data manipulation" for its examples, focusing on using real-world datasets and manipulating them (averages, graphs, indicies, searches, etc.) in various ways.

The book covers the standard CS1 curriculum, and includes extensive algorithm development sections to help students in their study of computing. Supplemental material is also provided including: full set of power point slides, collaborative lab exercises, project homeworks and solutions to over 600 exercises in the book.

[Head First Programming](http://www.headfirstlabs.com/books/hfprog) → <http://www.headfirstlabs.com/books/hfprog> by Paul Barry and David Griffiths

A Learner's Guide to Programming, using the Python Language

- ISBN-10: 0596802374
- ISBN-13: 978-0596802370
- O'Reilly Media
- 440 pages (November 2009)

Ever wished you could learn how to program from a book? If you have no previous programming experience, you might be wondering where to start. Head First Programming introduces the core concepts of writing computer programs--variables, decisions, loops, functions, and objects--which apply regardless of the programming

language, but uses concrete examples and exercises in the dynamic and versatile Python language to apply and reinforce these concepts.

Learn the basic tools to start writing the programs that interests you, not the generic software someone else thinks you should have, and get a better understanding of what software can (and cannot) do. When you're finished, you'll have the necessary foundation to apply to whatever language or software project you need or want to learn.

[Programming in Python 3: A Complete Introduction to the Python Language \(Second Edition\)](http://www.pythontutorial.net/introduction-to-python/) → <http://www.pythontutorial.net/introduction-to-python/> by Mark Summerfield

- ISBN10: 0321680561
- Addison-Wesley Professional
- 648 pages (November 2009)

This book teaches you how to write programs using Python 3 in good Python 3 style.

The book will be useful to people who program professionally as part of their job, whether as full-time software developers, or those from other disciplines, including scientists and engineers, who need to do some programming in support of their work. It will also prove ideal for those Python 2 programmers who need to migrate (or prepare to migrate) to Python 3. The book is also suitable for students—the only prerequisite is some basic knowledge of programming in any language, for example, Basic or Java, or of course Python itself.

The book teaches solid procedural style programming, then builds on that to teach solid object-oriented programming, and then goes on to more advanced topics such as descriptors and class decorators. But even newcomers to Python 3 should be able to write useful (although small and basic) programs after reading chapter 1, and then go on to create larger and more sophisticated programs as they work through the chapters.

The book's web site → <http://www.pythontutorial.net/introduction-to-python/> lists the table of contents and has links to extracts. It also has all the examples and exercise solutions available for download.

[Python for Software Design: How to Think Like a Computer Scientist](http://www.cambridge.org/us/catalogue/catalogue.asp?isbn=9780521725965) → <http://www.cambridge.org/us/catalogue/catalogue.asp?isbn=9780521725965> by Allen B. Downey, Olin College of Engineering, Massachusetts

- ISBN-13: 9780521725965
- Cambridge University Press
- 272 pages (March 2009)

Python for Software Design is a concise introduction to software design using the Python programming language. Intended for people with no programming experience, this book starts with the most basic concepts and gradually adds new material. Some of the ideas students find most challenging, like recursion and object-oriented programming, are divided into a sequence of smaller steps and introduced over the course of several chapters. The focus is on the programming process, with special emphasis on debugging. The book includes a wide range of exercises, from short examples to substantial projects, so that students have ample opportunity to practice each new concept. Exercise solutions and code examples are available from thinkpython.com, along with Swamy, a suite of Python programs that is used in some of the exercises.

[Hello World! Computer Programming for Kids and Other Beginners](http://www.manning.com/sande) → <http://www.manning.com/sande> by Warren Sande and Carter Sande

- ISBN10: 1933988495
- Manning Publications

- 488 pages (January 2009)

"Computer programming is a powerful tool for children to 'learn learning,' that is, to learn the skills of thinking and problem-solving...Children who engage in programming transfer that kind of learning to other things."-- Nicholas Negroponte, the man behind the One Laptop Per Child project that hopes to put a computer in the hands of every child on earth, January 2008

Your computer won't respond when you yell at it. Why not learn to talk to your computer in its own language? Whether you want to write games, start a business, or you're just curious, learning to program is a great place to start. Plus, programming is fun!

Hello World! provides a gentle but thorough introduction to the world of computer programming. It's written in language a 12-year-old can follow, but anyone who wants to learn how to program a computer can use it. Even adults. Written by Warren Sande and his son, Carter, and reviewed by professional educators, this book is kid-tested and parent-approved. You don't need to know anything about programming to use the book. But you should know the basics of using a computer--e-mail, surfing the web, listening to music, and so forth. If you can start a program and save a file, you should have no trouble using this book.

Color ebook and black and white print book are both available from the publisher at www.manning.com/sande → <http://www.manning.com/sande>.

Hello Python! → <http://manning.com/briggs/> by Anthony Briggs

- Foreword by Steve Holden
- 424 pages (February, 2012)
- ISBN 9781935182085

Hello! Python fully covers the building blocks of Python programming and gives you an introduction to more advanced topics such as object-oriented programming, functional programming, network programming, and program design. New (or nearly new) programmers will learn most of what they need to know to start using Python immediately.

A more advanced book than Hello World!, it takes a different tack. You follow along as the author writes a practical project in every chapter. You'll see how programmers think about solving problems with programming, and there are hints on how to extend the projects to suit your own needs.

The Quick Python Book, 2nd edition → <http://www.manning.com/ceder/>

- Naomi Ceder
- [ISBN13:978-1935182207 → http://www.amazon.com/Quick-Python-Book-Second/dp/193518220X/ref=sr_1_1?ie=UTF8&qid=1309567984&sr=8-1](http://www.amazon.com/Quick-Python-Book-Second/dp/193518220X/ref=sr_1_1?ie=UTF8&qid=1309567984&sr=8-1)
- Manning Publications,
- 335 pages (January 2010)

This revision of Manning's popular The Quick Python Book offers a clear, crisp introduction to the elegant Python programming language and its famously easy-to-read syntax. Written for programmers new to Python, this updated edition covers features common to other languages concisely, while introducing Python's comprehensive standard functions library and unique features in detail.

After exploring Python's syntax, control flow, and basic data structures, the book shows how to create, test, and deploy full applications and larger code libraries. It addresses established Python features as well as the advanced object-oriented options available in Python 3. Along the way, you'll survey the current Python development landscape, including GUI programming, testing, database access, and web frameworks.

[Chapter 4](http://manning.com/ceder/SampleChapter-4.pdf) → <http://manning.com/ceder/SampleChapter-4.pdf> [Chapter 6](http://manning.com/ceder/SampleChapter-6.pdf) → <http://manning.com/ceder/SampleChapter-6.pdf>

[Review by Jim Kohli, dzone.com](http://books.dzone.com/reviews/quick-python-book-delivers) → <http://books.dzone.com/reviews/quick-python-book-delivers>

Ebook and print book are both available from the publisher at www.manning.com/ceder → <http://www.manning.com/ceder>.

[IronPython in Action](http://www.ironpythoninaction.com/) → <http://www.ironpythoninaction.com/> by Michael Foord

- ISBN10: 1933988339
- Manning Publications
- 480 pages (March 2009)

[IronPython](https://wiki.python.org/moin/IronPython) → <https://wiki.python.org/moin/IronPython> is an implementation of Python for the Microsoft .NET framework, Mono, and the Silverlight and Moonlight browser plugins. [IronPython in Action](https://wiki.python.org/moin/IronPython_in_Action) is an introduction to programming with [IronPython](https://wiki.python.org/moin/IronPython) → <https://wiki.python.org/moin/IronPython> for both .NET programmers interested in Python and Python programmers new to .NET.

[IronPython](https://wiki.python.org/moin/IronPython) → <https://wiki.python.org/moin/IronPython> in Action includes a swift paced Python tutorial, chapters introducing .NET libraries and structured application development with Python, integrating with other .NET languages like C# and VB.NET, server side web programming with ASP.NET and client side web programming with Silverlight, system administration, working with the WPF and Windows Forms user interface libraries and embedding the [IronPython](https://wiki.python.org/moin/IronPython) → <https://wiki.python.org/moin/IronPython> engine in .NET applications.

As well as covering specific topics and both Python and .NET libraries the book pays special attention to the nitty-gritty details of Python and .NET integration that previous experience with Python or C# won't necessarily have prepared you for.

Color ebook and black and white print book are both available from the publisher at www.manning.com/foord → <http://www.manning.com/foord>.

"Beginning Python: From Novice to Professional, Second Edition" Magnus Lie Hetland

ISBN13: 978-1-59059-982-2 Apress Inc, 688 pages (September 2008)

Gain a fundamental understanding of Python's syntax and features with the second edition of Beginning Python, an up-to-date introduction and practical reference. Covering a wide array of Python-related programming topics, including addressing language internals, database integration, network programming, and web services, you'll be guided by sound development principles. Ten accompanying projects will ensure you can get your hands dirty in no time.

Updated to reflect the latest in Python programming paradigms and several of the most crucial features found in the forthcoming Python 3.0 (otherwise known as Python 3000), advanced topics, such as extending Python and packaging/distributing Python applications, are also covered.

[Home Page](http://www.apress.com/book/view/9781590599822/) → <http://www.apress.com/book/view/9781590599822/> (includes electronic version)

Object-Oriented Programming in Python

- Michael H. Goldwasser, David Letscher

This is a textbook for an object-oriented introduction to computer science course (CS1) using Python.

[Home Page](http://prenhall.com/goldwasser) → <http://prenhall.com/goldwasser>

[0136150314](http://www.amazon.com/exec/obidos/ISBN=0136150314) → <http://www.amazon.com/exec/obidos/ISBN=0136150314>, Prentice Hall Publishing, 688 pages (October 2007)

An Introduction to Python (version 2.5)

- Guido van Rossum, and Fred L. Drake, Jr. (Editor)

[0954161769](http://www.amazon.com/exec/obidos/ISBN=0954161769) → <http://www.amazon.com/exec/obidos/ISBN=0954161769>, Network Theory Ltd, 164 pages (November 2006)

This is a printed edition of the official Python tutorial by Guido van Rossum. For each copy sold \$1 will be donated to the Python Software Foundation.

[Home Page](http://www.network-theory.co.uk/python/manual/) → <http://www.network-theory.co.uk/python/manual/> (includes electronic version)

Python for Dummies (version 2.5)

- Stef Maruch and Aahz Maruch

[0471778648](http://www.amazon.com/exec/obidos/ISBN=0471778648) → <http://www.amazon.com/exec/obidos/ISBN=0471778648>, For Dummies, 410 pages (August 2006)

[Home Page](http://www.pythongoodies.com/) → <http://www.pythongoodies.com/>

Following the usual Dummies style, this book takes a light-hearted approach to introducing Python. In addition to Python itself, *Python for Dummies* also gives lots of advice about good programming practices.

Introduction to Computing and Programming Using Python: A Multimedia Approach Mark Guzdial

ISBN: 0131176552 <http://coweb.cc.gatech.edu/mediaComp-teach>

An introduction to programming by manipulating digital media --- creating negatives and grayscale pictures, splicing sounds, implementing chromakey.

Core Python Programming

- Wesley J. Chun

ISBN: 0132269937 (2nd ed.); 0130260363 (1st ed.)

Prentice Hall PTR / Pearson Education

~1120 pages (Sep 2006); 810 pages (Jan 2001)

The main goal of this book is comprehensively teaching you the core of the Python language, much more than just its syntax (which you don't really need a book to learn, right?). Knowing more about how Python works under the covers, including the relationship between data objects and memory management, will make you a much more **effective** Python programmer coming out of the gate. The advanced topics chapters are meant as complete intros or "quick dives" into each of those distinct subjects. However, if moving towards those specific areas of development, they are more than enough to get you pointed in the right direction. We would say that the book is 40% introductory, 40% intermediate (in-depth core Python material plus advanced topics chapters), and 20% reference -- it is *not* meant to be a substitute for a pure reference such as the *Python Essential Reference* or *Python in a Nutshell*.

The new 2nd edition is expanded (300 new pages!) and updated through Python 2.5 as well as confirmed functionality for future versions! Also added are a few new chapters of advanced material. As in the 1st edition, a plethora of easy to advanced exercises can be found at the end of every chapter to hammer the concepts home. At the moment, this is the most complete and up-to-date Python book on the market today.

From an anonymous reviewer: "Very well written. It is the clearest, friendliest book I have come across yet for explaining Python, and putting it in a wider context. It does not presume a large amount of other experience. It may be too slow for more advanced people, but it does go into some important Python topics carefully and in

depth. Unlike too many beginner books, it never condescends or tortures the reader with childish hide-and-seek prose games. Not too many in-depth real-world examples in the book (hopefully he will do a followup volume), it sticks to gaining a solid grasp of Python syntax and structure."

[Home Page](http://corepython.com/) → <http://corepython.com/> (includes book reviews, errata, sample chapter, links to alternate editions, source code from the book, and more!)

Links regarding the 1st edition: [IBM Developer Works review](http://www-106.ibm.com/developerworks/linux/library/l-pbook3.html) → <http://www-106.ibm.com/developerworks/linux/library/l-pbook3.html> [Linux Journal review](http://www2.linuxjournal.com/lj-issues/issue85/4564.html) → <http://www2.linuxjournal.com/lj-issues/issue85/4564.html> [Chinese edition](http://www.china-pub.com/computers/common/info.asp?id=3097) → <http://www.china-pub.com/computers/common/info.asp?id=3097> [O'Reilly Safari electronic edition](http://safaribooksonline.com/main.asp?bookname=0130260363) → <http://safaribooksonline.com/main.asp?bookname=0130260363> [Indian edition \(English\)](http://www.prakashbooks.com/details.php3?id=5806) → <http://www.prakashbooks.com/details.php3?id=5806> [Korean edition](http://www.wowbook.com/computer/book/info/book_detail.asp?isbn=ISBN89-450-7052-4) → http://www.wowbook.com/computer/book/info/book_detail.asp?isbn=ISBN89-450-7052-4 [Association of C/C++ Users review](http://www.accu.org/bookreviews/public/reviews/c/c002320.htm) → <http://www.accu.org/bookreviews/public/reviews/c/c002320.htm> [Python Learning Foundation review](http://www.awaretek.com/CorePython.html) → <http://www.awaretek.com/CorePython.html> [Mississippi Python users group \(Useless Python\) review](http://uselesspython.com/) → <http://uselesspython.com/gettingstarted.html>

Python Programming: An Introduction to Computer Science

- John Zelle

[1887902996](http://www.amazon.com/exec/obidos/ISBN=1887902996) → <http://www.amazon.com/exec/obidos/ISBN=1887902996>, Franklin Beedle & Associates, December 2003

This is a textbook for a "traditional" introduction to computer science course (CS1) using Python.

[Home Page](http://mcsp.wartburg.edu/zelle/python) → <http://mcsp.wartburg.edu/zelle/python>

- **How to Think Like a Computer Scientist: Learning with Python**
- Allen Downey, Jeff Elkner and Chris Meyers

[0971677506](http://www.amazon.com/exec/obidos/ISBN=0971677506) → <http://www.amazon.com/exec/obidos/ISBN=0971677506>, Green Tea Press 288 pages

How to Think... is a free textbook available under the GNU Free Documentation License. It is a true beginners book. The ebook version is free and available in PDF, HTML and [PostScript](https://wiki.python.org/moin/PostScript) → <https://wiki.python.org/moin/PostScript>.

[Home Page](http://thinkpython.com/) → <http://thinkpython.com/>

- **Dive Into Python: Python for Experienced Programmers**
- Mark Pilgrim

[1590593561](http://www.amazon.com/exec/obidos/ISBN=1590593561) → <http://www.amazon.com/exec/obidos/ISBN=1590593561>, Apress 432 pages

Dive Into Python is a free Python ebook for experienced programmers available under the GNU Free Documentation License. A printed version has been published by Apress <http://www.apress.com/>, and is available through all major outlets.

[Home Page](http://www.diveintopython.net/) → <http://www.diveintopython.net/>

Learning Python

- Mark Lutz

[0596158068](http://www.amazon.com/exec/obidos/ISBN=0596158068) → <http://www.amazon.com/exec/obidos/ISBN=0596158068> O'Reilly & Associates, 1216 pages (September 2009)

Learning Python is meant for beginning Python programmers, and others seeking a quick introduction to the language. It focuses on core language fundamentals in depth, is based on Mark Lutz's Python training classes, and includes numerous exercises with solutions to guide the reader through a hands-on learning experience.

[O'Reilly's catalog page](http://www.oreilly.com/catalog/9780596158071/) → <http://www.oreilly.com/catalog/9780596158071/> [Author's book page](http://rmi.net/~lutz/about-lp4e.html) → <http://rmi.net/~lutz/about-lp4e.html> [Sample chapters](http://oreilly.com/catalog/9780596158071/toc.html) → <http://oreilly.com/catalog/9780596158071/toc.html> [Review of first edition by Phil Hughes in Linux Journal](http://www2.linuxjournal.com/j-issues/issue66/3541.html) → <http://www2.linuxjournal.com/j-issues/issue66/3541.html>

The Quick Python Book

- Daryl Harms, Kenneth McDonald → <https://wiki.python.org/moin/McDonald>

[1884777740](http://www.amazon.com/exec/obidos/ISBN=1884777740) → <http://www.amazon.com/exec/obidos/ISBN=1884777740>, Manning Publications, 275 pages (October 1999)

A clear and concise description of Python aimed at readers who are already familiar with programming in at least one other language.

[Two sample chapters](http://www.manning.com/getpage.html?project=harms&filename=Chapters.html) → <http://www.manning.com/getpage.html?project=harms&filename=Chapters.html> [Review by Francis Glassborow in C Vu](http://www.accu.org/bookreviews/public/reviews/q/q002082.htm) → <http://www.accu.org/bookreviews/public/reviews/q/q002082.htm> [Review by Phil Hughes](http://www2.linuxjournal.com/j-issues/issue73/3851.html) → <http://www2.linuxjournal.com/j-issues/issue73/3851.html> [Review by AMK](http://www.amk.ca/python/books/qpb.html) → <http://www.amk.ca/python/books/qpb.html>

A 2nd edition (complete rewrite moving it from Python 1.x to 3.x) was published Jan 2010.

[Publisher's page for The Quick Python Book, 2nd ed.](http://www.manning.com/ceder/) → <http://www.manning.com/ceder/>

Learn to Program Using Python

- Alan Gauld

[0201709384](http://www.amazon.com/exec/obidos/ISBN=0201709384) → <http://www.amazon.com/exec/obidos/ISBN=0201709384>, Addison-Wesley, 270 pages (December, 2000)

This book teaches programming in Python to true beginners. It started as a popular Web tutorial, and been expanded into a complete book.

[Home Page](http://www.freernetpages.co.uk/hp/alan.gauld/index.htm) → <http://www.freernetpages.co.uk/hp/alan.gauld/index.htm>

Programming Python (Third Edition)

- Mark Lutz

[0596009259](http://www.amazon.com/exec/obidos/ISBN=0596009259) → <http://www.amazon.com/exec/obidos/ISBN=0596009259>, O'Reilly & Associates, 1596 pages (August, 2006)

A Python classic, updated and expanded to cover Python 2.5. The first edition, published in 1996, was the first Python book project to be signed. Programming Python is about what you can do with Python after you've mastered the language fundamentals - it assumes you already know the core language, and focuses on applications programming in gradual tutorial fashion. It is designed to be a natural follow-up to the book Learning Python. This book includes 300 pages on GUIs, 500 on Internet programming, and more on databases, systems programming, text processing, Python/C integration, and other topics. Also available in PDF form from O'Reilly.

[O'Reilly's catalog page](http://www.oreilly.com/catalog/python3/) → <http://www.oreilly.com/catalog/python3/> [Author's book page](http://rmi.net/~lutz/about-pp3e.html) → <http://rmi.net/~lutz/about-pp3e.html> [Foreword by Guido van Rossum](http://www.python.org/doc/essays/foreword.html) → <http://www.python.org/doc/essays/foreword.html> [Review of the first edition by Greg Wilson](http://www.ercb.com/ddj/1997/ddj.9711.html) → <http://www.ercb.com/ddj/1997/ddj.9711.html> [Review of the first edition by Terry](http://www.ercb.com/ddj/1997/ddj.9711.html)

[Rooker in ;login:](http://www.usenix.org/publications/login/1998-4/python.html) → <http://www.usenix.org/publications/login/1998-4/python.html> Review of the first edition by [Danny Yee](http://dannyreviews.com/h/Python.html) → <http://dannyreviews.com/h/Python.html>

Python Pocket Reference (Third Edition)

- Mark Lutz

[0596009402](#) → <http://www.amazon.com/exec/obidos/ISBN=0596009402>, O'Reilly & Associates, 160 pages (February, 2005)

This handy reference guide summarizes Python statements, built-in functions, escape and formatting codes, and other prominent Python language features.

Python Standard Library

- [FredrikLundh](https://wiki.python.org/moin/FredrikLundh) → <https://wiki.python.org/moin/FredrikLundh>

[0596000960](#) → <http://www.amazon.com/exec/obidos/ISBN=0596000960>, O'Reilly & Associates, 250 pages (February 2001)

Based in part on 3,000 newsgroup articles written by Python veteran [FredrikLundh](https://wiki.python.org/moin/FredrikLundh) → <https://wiki.python.org/moin/FredrikLundh> over the last four and half years, this book provides sample scripts for all standard modules in the Python library. Also available in German.

[Author's book page](#) → <http://www.pythonware.com/people/fredrik/librarybook.htm> [Electronic edition \(archived snapshot\)](#) → <https://web.archive.org/web/20201017142948/http://effbot.org/zone/librarybook-index.htm>

Web Programming in Python: Techniques for Integrating Linux, Apache, and MySQL

- George K. Thiruvathukal, John Shafae and Thomas Christopher

[0130410659](#) → <http://www.amazon.com/exec/obidos/ISBN=0130410659>, Prentice Hall, 450 pages (October 2001)

The book has introductory chapters on Python, networking, Apache, Linux, and MySQL. It is a self-contained reference to Python and open-source programming that makes use of Python to develop real applications that are also available under an open source license.

Programming With Python

- Tim Altom with Mitch Chapman,

[0761523340](#) → <http://www.amazon.com/exec/obidos/ISBN=0761523340>, Prima Publishing, (October 1999)

[Review by Phil Hughes in Linux Journal](#) → <http://www2.linuxjournal.com/lj-issues/issue73/3851.html> [Review by Cary Miller](#) → <http://lists.tummy.com/pipermail/frpythoneers/2000-June/000085.html>

Python Developer's Handbook

- Andre Lessa

[0672319942](#) → <http://www.amazon.com/exec/obidos/ISBN=0672319942>, Sams, 600 pages (December 2000)

Python How to Program

- Harvey M Deitel, Paul J Deitel, Jonathan Liperi, Ben Wiedermann

0130923613 → <http://www.amazon.com/exec/obidos/ISBN=0130923613>, Prentice Hall, 1376 pages (2002)

In the renowned Deitel series "How to Program"

description → <http://vig.prenhall.com/catalog/academic/product/1.4096.0130923613.00.html>

Text Processing in Python

- David Mertz

The free text is available at:

<http://gnosis.cx/TPiP/>

Buy the dead-trees version at:

<http://tinyurl.com/jskh>

A review by Danny Yee at: http://dannyreviews.com/h/Text_Python.html

See also: [TextProcessingInPython](#) → <https://wiki.python.org/moin/TextProcessingInPython>

Python Programming Patterns Thomas Christopher

Prentice-Hall, 2001

ISBN: 0130409561

There's a sample chapter → <http://www.informit.com/articles/printfriendly.asp?p=28672&rl=1> at informIT

Reviews: ACCU review → <http://www.accu.org/bookreviews/public/reviews/p/p003210.htm> by Francis Glasborow; one of several books in Mertz's book roundup #3 → http://gnosis.cx/publish/programming/charming_python_b8.html

Python First: Introduction to Computing with Python

Atanas Radenski

The 'Python First' digital pack provides a gentle introduction to computer science. It is more than a book: Ten self-contained online chapters consist of e-texts, slides, 62 labs, tens of sample programs, and online quizzes. The 'Python First' pack includes a wealth of detailed self-guided labs that you can complete on your own.

Home Page → <http://studypack.com/comp/course/view.php?id=232>

Practical Programming (in Python) "Practical Programming.(in Python)" → <https://launchpad.net/practical-programming> is meant as a first programming course and is tightly aligned with the University of Otago → <http://www.otago.ac.nz/> introductory programming course called Practical Programming → <http://www.cs.otago.ac.nz/student/papers.php?name=COMP150>. The textbook is organized into 24 "lectures" that cover all the basics of programming (sequence, selection, iteration, functions etc), plus all the major data structures supplied by Python.

This textbook is a modified version of "How to Think Like a Computer Scientist: Learning with Python 2nd Edition" → <http://www.openbookproject.net/thinkcs/python/english2e/>, by Jeffrey Elkner, Allen B. Downey and Chris Meyers.

Books no longer available

How Would Pareto Learn Python!

[Rahul Verma → http://www.testingperspective.com/?page_id=1889](http://www.testingperspective.com/?page_id=1889)

Python is a very powerful high level object oriented language and has easy bindings with C,C++,Java etc. It can very well be the first and the only language needed by a software tester for routine test automation tasks as well as for building robust general purpose test automation frameworks.

This online book is written keeping beginners in Python in mind. It bridges the gap between very basic tutorials and comprehensive books.

The book is provided for free by [Talent Reboot Trainings and Assessments → http://www.talentreboot.com/](http://www.talentreboot.com/) and hosts the online version of the book as a part of its official website.

[Read Online Free Version - How Would Pareto Learn Python → http://www.talentreboot.com/publications/book-how-would-pareto-learn-python/](http://www.talentreboot.com/publications/book-how-would-pareto-learn-python/)

[CategoryPythonInEducation → https://wiki.python.org/moin/CategoryPythonInEducation](https://wiki.python.org/moin/CategoryPythonInEducation) [CategoryDocumentation → https://wiki.python.org/moin/CategoryDocumentation](https://wiki.python.org/moin/CategoryDocumentation)

[Learning Python → https://www.packtpub.com/packt/free-ebook/learning-python \(Free eBook\)](https://www.packtpub.com/packt/free-ebook/learning-python)

By Fabrizio Romano

ISBN 13: 9781783551712 Packt Publishing 442 pages (December 2015)

Book Overview:

This book goes deeper than simply showing you how to build a Python app, giving you the fundamentals of Python programming that every developer needs to know to make the most of the language. Packed with tutorials and examples this title features everything from data structures, writing reusable code, testing, paradigms, and how Python can be adapted. This free eBook will help transform you from a complete beginner to someone ready to bring the best out of their projects.

- Get Python up and running on Windows, Mac, and Linux in no time
- Grasp the fundamental concepts of coding, along with the basics of data structures and control flow.
- Write elegant, reusable, and efficient code in any situation
- Understand when to use the functional or the object oriented programming approach
- Create bulletproof, reliable software by writing tests to support your code
- Explore examples of GUIs, scripting, data science and web applications
- Learn to be independent, capable of fetching any resource you need, as well as dig deeper

Who this book is written for:

This book is meant for programmers who wants to learn Python programming from a basic to an expert level. The book is mostly self-contained and introduces Python programming to a new reader and can help him become an expert in this trade.

[Publisher's page → http://www.packtpub.com/packt/free-ebook/learning-python](http://www.packtpub.com/packt/free-ebook/learning-python)

[CategoryPythonWebsite → https://wiki.python.org/moin/CategoryPythonWebsite](https://wiki.python.org/moin/CategoryPythonWebsite) [CategoryPyCon2008 → https://wiki.python.org/moin/CategoryPyCon2008](https://wiki.python.org/moin/CategoryPyCon2008) [CategoryPythonInEducation → https://wiki.python.org/moin/CategoryPythonInEducation](https://wiki.python.org/moin/CategoryPythonInEducation)

BeginnersGuide/Examples - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Examples>

Python Examples and Sample Code

When you're learning, small examples can be very helpful.

- [The Python Standard Library](https://web.archive.org/web/20201017142948/http://effbot.org/zone/librarybook-index.htm) → <https://web.archive.org/web/20201017142948/http://effbot.org/zone/librarybook-index.htm> (archived copy), an electronically published book by Fredrik Lundh, examines most of the modules in Python's standard library, describing what the module does and giving a short example of its use. Note that this book is now relatively old and so misses a lot of the developments of the last two decades.
- The [Python Recipes](https://code.activestate.com/recipes/langs/python/) → <https://code.activestate.com/recipes/langs/python/>, from [ActiveState](https://wiki.python.org/moin/ActiveState) → <https://wiki.python.org/moin/ActiveState>, is a very large collection of code snippets, some elementary and some advanced.

PythonEvents - Python Wiki

Source: <https://wiki.python.org/moin/PythonEvents>

Python Events around the World

The following is a list of Python related events that are occurring around the world or online.

Event Listings

- For conference and user group events, please see the [PythonEventsCalendar](#) → <https://wiki.python.org/moin/PythonEventsCalendar> page for details. The calendars are available on the python.org website as:
 - [Python Conferences Events Calendar](#) → <https://www.python.org/events/python-events/>
 - [Python User Group Events Calendar](#) → <https://www.python.org/events/python-user-group/>

or combined as:

- [Python Events Calendar](#) → <https://www.python.org/events/>

The calendars are maintained by a group of volunteers. If you want to submit conference and user group events, please see the [submission section](#) → https://wiki.python.org/moin/PythonEventsCalendar#Submitting_an_Event of the project page. In recent years, the events calendars have grown to be a standard resource for people trying to find Python community related events, so we'd like to encourage your submission to those calendars.

- If you are looking for training events in upcoming months, please see the [PythonTraining/Events](#) → <https://wiki.python.org/moin/PythonTraining/Events> page.
- [TechVenue.com](#) → <http://calendars.techvenue.com/cgi-bin/techvenue.pl?CalendarName=Python> has another calendar of Python events that includes regularly scheduled user-group meetings.

Pointers

Python Conferences

See the [Python Conferences](#) → <https://wiki.python.org/moin/PythonConferences> page for a general overview of the different Python conferences.

If you want to organize a conference or event in your city or community, read [our guide to running a conference](#) → <https://wiki.python.org/moin/AdvocacyWritingTasks/RunningAConference> for helpful advice.

Looking for speakers at various events? Check out the [list of Python speakers](#) → [https://wiki.python.org/moin/PythonSpeakers!](https://wiki.python.org/moin/PythonSpeakers)

Python Training

Looking for a complete list of Python training options? Check out the [Python training](#) → <https://wiki.python.org/moin/PythonTraining> page!

If you want to set up your own training event, refer to the [Python Training](#) → <https://wiki.python.org/moin/PythonTraining> page on this site and add training dates to the [PythonTraining/Events](#) → <https://wiki.python.org/moin/PythonTraining/Events> page. You can also add yourself to [Cameron Laird's](#) → <https://wiki.python.org/moin/CameronLaird> page of [Python trainers](#) → http://phaseit.net/claird/comp.lang.python/python_training.html.

Python User Groups

Also see the [local user groups](https://wiki.python.org/moin/LocalUserGroups) → <https://wiki.python.org/moin/LocalUserGroups> list for recurring local events.

Calendar

This section is kept around for backwards compatibility

Since the calendar on this page was only showing training events in recent months, we have moved those event listings to a new [PythonTraining/Events](https://wiki.python.org/moin/PythonTraining/Events) → <https://wiki.python.org/moin/PythonTraining/Events> page.

Please check the above list of calendars for conference and user group events.

PythonTraining - Python Wiki

Source: <https://wiki.python.org/moin/PythonTraining>.

Python Training

If you are seeking upcoming open-to-the-public training events, check out the [PythonTraining/Events](https://wiki.python.org/moin/PythonTraining/Events) page.

Contents

1. [Python Training](#)
 1. [Adding entries](#)
 1. [Formatting of Entries](#)
 2. [USA](#)
 3. [Canada](#)
 4. [Europe](#)
 5. [Australia](#)
 6. [Latin America](#)
 7. [India](#)
 8. [Iran](#)
 9. [Israel](#)
 10. [Malaysia](#)
 11. [Nigeria](#)
 12. [Singapore](#)
 13. [New Zealand](#)
 14. [Turkey](#)
 15. [Internet - Python Online Training](#)

Adding entries

If you want to add information to this page and find that it is immutable, please sign up to the wiki and see the [FrontPage#use](#) for details on how to get editing permissions.

The sections tends to be ordered by "time since last person edited it" or by event date in the case of upcoming public courses. As such, entries towards the top will cycle slowly towards the bottom of the section over time. If you're offering an upcoming class or have updated your entry, it's fine to bump it back to the top, but be considerate of others and use some common sense. Entries with dead links may be removed.

Formatting of Entries

Please use the following format when adding entries to the wiki text version of the page:

```
* '''Trainer''' offers training on ...
```

and this format for the GUI version of the page:

- Trainer offers training on ...

Please avoid excessive emphasis and keep the entries short to give everyone a chance to promote their training classes.



Entries which are too long will be flagged as such from time to time to gently make the authors aware.

USA

* **Marilyn Davis, Ph.D.** → <http://www.pythontainer.com/> Marilyn specializes in Python training in corporate environments in the Silicon Valley, and anywhere, either through UCSC-Extension → <http://www.ucsc-extension.edu/>, where students earn University credit, or independently. She has taught Python at Apple, Cisco, Facebook, Google, Intuit, LLNL, NASA, Oracle, Plantronics, Skype, VMware, and more.

- All Online
- Python For Programmers → <https://course.ucsc-extension.edu/modules/shop/index.html?action=section&OfferingID=5591127&SectionID=7590317>, where students learn and practice core concepts and Pythonic thinking: **Sep 13, 2022 - Dec 13, 2022**.
- Here's a sample lecture. → <https://www.youtube.com/watch?v=t6pdG1oWFX4> PythonTrainer.Com → <http://www.pythontainer.com/> is a 100% woman-owned business, registered in SAM → <https://www.sam.gov/portal/public/SAM/>.

* **Skill Distillery** → <https://skilldistillery.com/> is a coding school providing career transformation programs for those pivoting careers in technology. Its award-winning online programs attract career changers worldwide. Skill Distillery also offers masterclasses for those wanting to continue their career growth. Masterclasses are offered in a convenient online evening format over 4 to 5 weeks.

- Upcoming Masterclasses:

- [Introduction to Python 3](https://skilldistillery.com/python-masterclass/) → <https://skilldistillery.com/python-masterclass/>: October 28 - November 23, 2021, Tuesday and Thursday 6-9 pm MT.

* **Webucator** → <https://www.webucator.com/> regularly provides live online Python training → <https://www.webucator.com/catalog/python-training/>.

- Upcoming Classes:

- [Python Data Analysis with JupyterLab Training](https://www.webucator.com/python-training/course/python-data-analysis-with-jupyterlab/) → <https://www.webucator.com/python-training/course/python-data-analysis-with-jupyterlab/> - Jun 15-16, 2023 10:00AM-5:00PM ET, Jul 13-14, 2023 10:00AM-5:00PM ET, Aug 10-11, 2023 10:00AM-5:00PM ET
- [Introduction to Django Training](https://www.webucator.com/django-training/course/introduction-django-training/) → <https://www.webucator.com/django-training/course/introduction-django-training/> - Jun 19-23, 2023 10:00AM-5:00PM ET, Jul 17-21, 2023 10:00AM-5:00PM ET, Aug 14-18, 2023 10:00AM-5:00PM ET
- [Advanced Django Training](https://www.webucator.com/django-training/course/advanced-django-training/) → <https://www.webucator.com/django-training/course/advanced-django-training/> - Jun 5-7, 2023 10:00AM-5:00PM ET, Jul 5-7, 2023 10:00AM-5:00PM ET, Jul 31-Aug 2, 2023 10:00AM-5:00PM ET
- [Python Essentials Training](https://www.webucator.com/python-training/course/python-essentials-training/) → <https://www.webucator.com/python-training/course/python-essentials-training/> - Jun 19-23, 2023 10:00AM-5:00PM ET, Jul 17-21, 2023 10:00AM-5:00PM ET, Aug 14-18, 2023 10:00AM-5:00PM ET
- [Introduction to Spark with Python](https://www.webucator.com/spark-training/course/introduction-spark-with-python/) → <https://www.webucator.com/spark-training/course/introduction-spark-with-python/> - Jun 5-7, 2023 10:00AM-5:00PM ET, Jul 5-7, 2023 10:00AM-5:00PM ET, Jul 31-Aug 2, 2023 10:00AM-5:00PM ET

* **DataCreative** → <https://datacreative.com/> delivers Python training classes → https://datacreative.com/classes/outline_s/python-programming-professional/ online → <https://datacreative.com/classes/online/python/> and in-person in Sacramento → <https://datacreative.com/classes/sacramento/python/> and San Jose → <https://datacreative.com/classes/san-jose/python/>.

* **The Hartmann Software Group** → <https://www.hartmannsoftware.com/> is an IT training and consulting organization based in Denver, Colorado that offers an extensive catalog of Python courses → <https://www.hartmannsoftware.com/training/python>. If you are looking to integrate Python with .NET, Java or C++, wish to employ Python's extensive Machine Learning libraries or hope to tackle many other out of the box development challenges, we can be of tremendous assistance either by way of training, mentoring and/or consulting.

- **Upcoming Classes:**

- [Python for Scientists](https://www.hartmannsoftware.com/hsg_a1067609284e8c671413401447289) → https://www.hartmannsoftware.com/hsg_a1067609284e8c671413401447289 - Dec 16-20, 2019 10AM-5PM ET
- [PYTHON FOR DATA SCIENTIST AND MACHINE LEARNING PRACTITIONERS Training](https://www.hartmannsoftware.com/hsg_8dd80d133416d4891517840294778) → [http://www.hartmannsoftware.com/hsg_8dd80d133416d4891517840294778](https://www.hartmannsoftware.com/hsg_8dd80d133416d4891517840294778) - Oct 28-Nov 1, 2019 10AM-5PM ET
- [PYTHON II: ADVANCED PYTHON 3](https://www.hartmannsoftware.com/PythonApplied) → <https://www.hartmannsoftware.com/PythonApplied> - Nov 14-15 10AM-5PM ET

* **David Beazley** → <http://www.dabeaz.com/>, author of the Python Essential Reference, 4th Ed (Addison-Wesley) and the Python Cookbook, 3rd Ed (O'Reilly Media), teaches ongoing **in-person** programming and computer science [Courses in Chicago](http://www.dabeaz.com/courses.html) → <http://www.dabeaz.com/courses.html>. If you've learned the basics of Python and want to advance your skills by applying what you've learned to more difficult projects, these courses provide an opportunity to do just that. Dave's courses are taught in round-table format that is **strictly limited to 6 students**--a size that fosters group discussion.

* **Web Age Solutions** → <https://www.webagesolutions.com/> offers instructor-led and virtual Python training → <https://www.webagesolutions.com/courses/python-training> in US and Canada. Our current offerings include [Introduction to Python Programming](https://www.webagesolutions.com/courses/TP1758-introduction-to-python-programming) → <https://www.webagesolutions.com/courses/TP1758-introduction-to-python-programming> and [Advanced Python Programming](https://www.webagesolutions.com/courses/TTPS4850-advanced-python-programming) → <https://www.webagesolutions.com/courses/TTPS4850-advanced-python-programming>. Web Age Solutions also provides customized training as per the needs of the organizations.

- **One Month Python** → <https://onemonth.com/courses/python/> is a 30 day online python course taught by Columbia University professor Mattan Griffel. In One Month Python you'll learn the basics of programming – variables, strings, lists, functions — as well as how to use tools like Jupyter, Pandas, and Flask. By the end of this course you will have created four projects: an Amazon Web Scraper, Weather API, Calculator, and a Twilio API script for sending SMS text messages from your code.

- [Enroll in One Month Python](https://onemonth.com/courses/python/) → <https://onemonth.com/courses/python/>
- [What will you learn in One Month Python?](https://www.youtube.com/watch?v=pfnPwnniE5Q) → [https://www.youtube.com/watch?v=pfnPwnniE5Q/](https://www.youtube.com/watch?v=pfnPwnniE5Q)

- **Enthought's** → <http://www.enthought.com/> experts train scientists, engineers, analysts, data scientists and geoscientists who would like to use Python more effectively. Enthought teaches public-open and private-onsite courses in cities around the US and EU. Current courses offered include:

- [Python Foundations](https://www.enthought.com/training/course/python-foundations/) → <https://www.enthought.com/training/course/python-foundations/>
- [Python for Scientists and Engineers](https://www.enthought.com/training/course/python-for-scientists-and-engineers/) → <https://www.enthought.com/training/course/python-for-scientists-and-engineers/>
- [Python for Data Science](https://www.enthought.com/training/course/python-for-data-science/) → <https://www.enthought.com/training/course/python-for-data-science/>
- [Python for Data Analysis](https://www.enthought.com/training/course/python-for-data-analysis/) → <https://www.enthought.com/training/course/python-for-data-analysis/>
- [Python for Machine Learning](https://www.enthought.com/training/course/python-for-machine-learning/) → <https://www.enthought.com/training/course/python-for-machine-learning/>

- [Machine Learning Mastery Workshop](https://www.enthought.com/training/course/machine-learning-mastery-workshop/) → <https://www.enthought.com/training/course/machine-learning-mastery-workshop/>
- [Pandas Mastery Workshop](https://www.enthought.com/training/course/pandas-mastery-workshop/) → <https://www.enthought.com/training/course/pandas-mastery-workshop/>
course is available upon request.
- Upcoming open classes are listed on the "Schedule" tab of each course's webpage → <https://www.enthought.com/python-training/>. If you are interested in scheduling a private-onsite course, please [contact us](https://www.enthought.com/contact/) → <https://www.enthought.com/contact/>. Enthought also has a set of 8 Pandas cheat sheets → <http://www.enthought.com/cheat-sheets-pandas-python-for-data-analysis>, a set of 3 machine learning cheat sheets → <http://www.enthought.com/cheat-sheets-machine-learning-in-python>, and a MATLAB to Python white paper → <http://www.enthought.com/white-paper-matlab-to-python-2/> available for free download.
- rmotr.com → <https://rmotr.com/> remote coding courses specialized in Python with real teachers and classmates. +50 hours of real coding, +12 projects in your Github profile, real mentors and classmates to work with. A great way to [learn python online](#) → <https://rmotr.com/learn-python-online>.
- Available courses
 - Intro course [Introduction to Programming with Python online course](https://rmotr.com/introduction-to-python-programming?utm_source=python_wiki) → https://rmotr.com/introduction-to-python-programming?utm_source=python_wiki. Some programming knowledge required.
 - Our most popular course, [Advanced Python Programming online course](https://rmotr.com/advanced-python-programming?utm_source=python_wiki) → https://rmotr.com/advanced-python-programming?utm_source=python_wiki, is a 4-week intensive course with a focus in advanced Python topics not taught anywhere else.
 - Finally, [Web Development with Django](https://rmotr.com/web-development-with-django?utm_source=python_wiki) → https://rmotr.com/web-development-with-django?utm_source=python_wiki. It's a highly practical course in which we focus on building REAL websites and APIs using Django.
- NEW! We've just released a [FREE Python Online Course](https://rmotr.com/free-python-online-course) → <https://rmotr.com/free-python-online-course> and a [FREE Flask Tutorial](https://flask-tutorial.com/) → <https://flask-tutorial.com/>.
- All the content we offer is **FREE** and **Open Source** and can be accessed in our own platform: learn.rmotr.com → <http://learn.rmotr.com/>.
- Check our reviews in [Course Report](https://www.coursereport.com/schools/rmotr-com/#reviews) → <https://www.coursereport.com/schools/rmotr-com/#reviews>
- [Naomi Ceder](mailto:naomi@naomiceder.tech) → <mailto:naomi@naomiceder.tech>: Naomi is available for on-site workshop-style Python training in the Americas and Europe. Courses range from introductory to advanced, can be customized to specific needs, and normally include large live coding and hands on elements. Course materials (typically in the form of Jupyter notebooks) are available after the classes for future use. Naomi is the author of the Quick Python Book, 2nd and 3rd editions and has experience teaching Python in various contexts at all levels in the US and Europe.
- [Yoav Ram, PhD](http://python.yoavram.com/) → <http://python.yoavram.com/> offers training in USA and Israel with focus on numerical, scientific, and statistical applications, including web application and user interfaces. To enhance participant learning, all course material is fully interactive, using Jupyter notebooks, and all lectures include hands-on exercises. Yoav is a postdoc at Stanford University, earned his PhD from Tel-Aviv University, working with Python since 2002, training and teaching Scientific Python since 2011, including [NumPy](https://wiki.python.org/moin/NumPy) → <https://wiki.python.org/moin/NumPy>, [SciPy](https://wiki.python.org/moin/SciPy) → <https://wiki.python.org/moin/SciPy>, Matplotlib, Jupyter, Pandas, Seaborn, Cython, scikit-image, scikit-learn, [TensorFlow](https://wiki.python.org/moin/TensorFlow) → <https://wiki.python.org/moin/TensorFlow>, Flask, Click, Tk. Experience with both small and enterprise companies.
- [Truthful Technology, LLC](http://truthful.technology/) → <http://truthful.technology/>: on-site workshop-style Python training taught by [Trey Hunner](#). Courses include **live coding** and **hands-on exercises** instead of slide-based lectures. Course curriculum comes in the form of a rich and descriptive documentation website in lieu of slides and speaker notes. All courses are tailored to your team's needs.

- **Firebox Training** consistently delivers [Python training courses](https://www.fireboxtraining.com/python) as live, instructor-led online, on-site, and classroom style training classes for individuals and groups. **Courses Offered:**
 - [Introduction to Advanced Python Programming Bootcamp 5-day - view detailed outline](https://www.fireboxtraining.com/python-training-bootcamps) → <https://www.fireboxtraining.com/python-training-bootcamps>
 - [Introduction to Python Programming Training 3-day - view detailed outline.](https://www.fireboxtraining.com/python-training-introduction-class) → <https://www.fireboxtraining.com/python-training-introduction-class>
 - [Advanced Python Programming Training 3-day - view detailed outline.](https://www.fireboxtraining.com/python-training-advanced-class) → <https://www.fireboxtraining.com/python-training-advanced-class>
 - [Python Programming Training for Scientist, Engineers and Analyst 5-Day - view detailed outline.](https://www.fireboxtraining.com/python-training-for-scientists-engineers-analyst) → <https://www.fireboxtraining.com/python-training-for-scientists-engineers-analyst>
 - [Python Programming Training for Scientist, Engineers and Analyst 3-Day - view detailed outline.](https://www.fireboxtraining.com/python-training-for-scientists-engineers-analyst) → <https://www.fireboxtraining.com/python-training-for-scientists-engineers-analyst>
- **Astro Code School** → <http://astrocodeschool.com/>, operated by **Caktus Group** → <http://www.caktusgroup.com/>, the world's largest Django development firm, is licensed by the State of North Carolina as a proprietary school and offers Django Bootcamps, Python and Django Web Engineering classes, and custom group trainings taught by qualified and experienced software educators. Current offerings are:
 - [Python and Django Web Engineering](http://astrocodeschool.com/classes/) → <http://astrocodeschool.com/classes/>, May 18 - August 10, 2015
 - [Python and Django Web Engineering](http://astrocodeschool.com/classes/) → <http://astrocodeschool.com/classes/>, September 22 - December 15, 2015
- **Six Feet Up** → <http://sixfeetup.com/> is a WBE dedicated to the development of Python based web applications and integration solutions for the enterprise. The company runs the local Indianapolis Python Meetup group and regularly gives public and private corporate training on a variety of Python based systems and web development.
- **Immersive Python, Django and Pyramid** → <https://elevenfifty.com/course-python/> is a 5-day hands on course that teaches students how to leverage Python & Django to build web applications and connect them to a server. The course begins with an overview of using Python for the web and then dives into Django. Students will create a simple blog site and then they will build the [ElevenNote](https://wiki.python.org/moin/ElevenNote) → <https://wiki.python.org/moin/ElevenNote> web application. Students will learn how to connect the [ElevenNote](https://wiki.python.org/moin/ElevenNote) → <https://wiki.python.org/moin/ElevenNote> web app they create to a Python web server and database. Students will even rebuild the [ElevenNote](https://wiki.python.org/moin/ElevenNote) → <https://wiki.python.org/moin/ElevenNote> app on the Pyramid microframework to understand the benefits of a leaner Python framework.
 - Next public class: [Dec 1-5th, Indianapolis, IN](#) → <https://elevenfifty.com/course-python/>
- **Batky-Howell Training** → <http://www.batky-howell.com/> has a 20-year history as an international learning organization publishing and delivering software development training content via classrooms, self-paced videos, and textbooks. Our [Python Training Courses](http://www.batky-howell.com/courses/python) → <http://www.batky-howell.com/courses/python> are continuously being delivered online and onsite to some of the largest and most recognized organizations in the world.
- **Hands On Technology Transfer, Inc. (HOTT)** offers a 5-day [Python Programming](http://www.traininghot.com/Courses/Python-Programming-Course.htm) → <http://www.traininghot.com/Courses/Python-Programming-Course.htm> course that teaches students how to rapidly develop and maintain effective Python programs. The course includes thorough coverage of Python syntax, built in data types and control constructs. Attendees will learn how to use Python to create scripts that manipulate data, automate tasks, perform error handling and store and retrieve data by using relational databases and XML files. Hands On Technology Transfer, Inc. (HOTT) offers competency-based IT training programs in more than 75 cities across the [United States](http://www.traininghot.com/Course-Schedules.htm) → <http://www.traininghot.com/Course-Schedules.htm>, [Canada](http://www.traininghot.com/Course-Schedules.htm) → <http://www.traininghot.com/Course-Schedules.htm>

[nghott.ca/](http://www.traininghott.ca/) and the [United Kingdom](http://www.traininghott.co.uk/) → <http://www.traininghott.co.uk/>, covering over 60 IT subject areas → <http://www.traininghott.com/Courses.htm>. These programs are designed with one main goal – making sure you and your staff will be competent and productive.

- [ITCourseware](http://www.itcourseware.com/) → <http://www.itcourseware.com/> provides quality Python Training Courseware → <http://www.itcourseware.com/catalogsearch/result/?q=python> for instructor-led Python Training → <http://www.batky-howell.com/courses/python/python-essentials.html>. This four day course leads the student from the basics of writing and running Python scripts to more advanced features such as file operations, regular expressions, working with binary data, and using the extensive functionality of Python modules.
- [Continuum Analytics](http://continuum.io/) → <http://continuum.io/> offers virtual, open, and custom on-site training courses for Python users of all levels and backgrounds. Driving core concept learning and in-depth understanding, Continuum's courses are led by Python experts and give students the ability to get their hands dirty, learn best practices, and ask questions about specific problems.
- [Practical Python Programming](https://store.continuum.io/cshop/practical%20python%20programming) → <https://store.continuum.io/cshop/practical%20python%20programming> is a 3-day introductory course into Python with a focus on applying Python to problems in scripting, data analysis, and systems programming. Geared toward quants and data analysts, the 4-day [Python for Finance](https://store.continuum.io/cshop/python%20for%20finance) → <https://store.continuum.io/cshop/python%20for%20finance> course provides a strong foundation for being able to work and prototype quicker. [Python for Science](https://store.continuum.io/cshop/python%20for%20science) → <https://store.continuum.io/cshop/python%20for%20science> is a 4-day course for scientists and engineers, which gives students a strong foundation of best practices and the tools available for technical computing.
 - Upcoming open and virtual classes can be found on the [Continuum Training page](https://store.continuum.io/cshop/training) → <https://store.continuum.io/cshop/training>. Custom on-site courses are available worldwide, all year round by contacting sales@continuum.io → <mailto:sales@continuum.io>.
- [DevelopIntelligence](http://www.developintelligence.com/) → <http://www.developintelligence.com/> is the leader in delivering highly-customized learning solutions to software teams. We offer more than 150 [developer training courses](http://www.developintelligence.com/catalog/it-training) → <http://www.developintelligence.com/catalog/it-training>. Including instructor-led [Python training](http://www.developintelligence.com/catalog/open-source-training/python) → <http://www.developintelligence.com/catalog/open-source-training/python>, [Apache training](http://www.developintelligence.com/catalog/apache-training) → <http://www.developintelligence.com/catalog/apache-training>, and other [open source training](http://www.developintelligence.com/catalog/open-source-training) → <http://www.developintelligence.com/catalog/open-source-training> courses.
- [CyberWeb Consulting](http://cyberwebconsulting.com/) → <http://cyberwebconsulting.com/> features courses delivered by principal consultant WesleyChun → <https://wiki.python.org/moin/WesleyChun>, another well-known Python trainer. Wesley is author of Prentice Hall's bestselling, [Core Python](http://corepython.com/) → <http://corepython.com/> books, the video lecture course, [Python Fundamentals](http://corepython.com/pf) → <http://corepython.com/pf> ([LiveLessons](http://mylivelessons.com/) → <http://mylivelessons.com/> DVD), and co-author of [Python Web Development with Django](http://withdjango.com/) → <http://withdjango.com/>. In addition to being a software architect and Developer Advocate at Google, he runs [CyberWeb](http://cyberwebconsulting.com/) → <http://cyberwebconsulting.com/>, a consultancy specializing in Python training. Wesley has over 25 years of programming, teaching, and writing experience, including more than a decade of Python. While at Yahoo!, Wesley helped create Yahoo!Mail using Python. He has delivered courses to Google, Cisco, VMware, Avaya, Hitachi, UC Santa Barbara, UC Santa Cruz, and Foothill College as well as tutorials and sessions at O'Reilly's [OSCON](http://oscon.com/) → <http://oscon.com/> and [PyCon](http://pycon.org/) → <http://pycon.org/>. Wesley holds degrees in Computer Science, Mathematics, and Music from the University of California. He is available to travel globally. **REGISTRATION OPEN for 2012 Aug 1-3 "Intro+Intermediate Python" course in San Francisco**. Go to [WEBSITE](http://cyberwebconsulting.com/) → <http://cyberwebconsulting.com/> for more info & registration.
- [Accelebrate](http://www.accelebrate.com/) offers [Python courses](http://www.accelebrate.com/python-training) → <http://www.accelebrate.com/python-training> at client sites all over the US and worldwide. Accelebrate's Python classes prepare beginning and intermediate Python users for developing sophisticated scripts and applications in the Python programming language. Our current offerings include [Introduction to Python training](http://www.accelebrate.com/training/python) → <http://www.accelebrate.com/training/python> and [Advanced Python training](http://www.accelebrate.com/training/python-advanced) → <http://www.accelebrate.com/training/python-advanced>, as well as [Python Programming for Scientists training](http://www.accelebrate.com/training/python-scientists) → <http://www.accelebrate.com/training/python-scientists>. Accelebrate also offers [PHP courses](http://www.accelebrate.com/php-training) → <http://www.accelebrate.com/php-training>.

- **Python Training** → <http://www.learntoprogram.tv/> with Mark Lassoff of LearnToProgram → <https://wiki.python.org/moin/LearnToProgram>.tv. Mark offers training in several programming languages including Python. Known for a promoting a dynamic classroom environment, Mark has been lauded for his ability to break down complex technical information into digestable chunks. Mark provides classes onsite at your company location and provides a lab based environment that encourages retention. Mark can be reached at 203-292-0689
- **Develop & Deploy: Django** → <http://c2etraining100201.eventbrite.com/> is five full days of intensive hands-on training in the Django framework. Django is a Python-based web framework that eases the creation of complex database driven websites. A combination of instruction, demos, and hands-on activities will immerse the developer in Django for a full week. The teaching style is casual but focused, with short informative lectures followed by exercises designed to give you immediately relevant understanding of each aspect of the technology you're learning. February 1 - February 5, 2010 at the Hacker Dojo → <http://www.hackerdojo.com/> in Mountain View, California
- **Mark Lutz** → <http://learning-python.com/> is probably the most experienced Python trainer in the world. He provides private on-site classes at your company or organization. Since 1997 he has taught roughly 250 Python classes to some 4,000 students in the US and abroad. Today, his classes cover both Python 2.X and 3.X, and present current best Python practice.
- **Dave Kuhlman** → http://www.rexx.com/~dkuhlman/course_descriptions.html teaches introductory Python courses.
- **Calvin Spealman** → <http://www.ironfroggy.com/blog/> offers private tutoring for general programming and problem specific assistance.
- **Stratolab** → <http://www.stratolab.com/> teaches Python for the purpose of writing video games. Clients are mostly middle and high school students. Stratolab is located in San Francisco.
- **Cameron Laird** → <https://wiki.python.org/moin/CameronLaird> occasionally teaches classes on advanced Python topics--Python for process control, Python with LDAP, and so on.
- **Pradeep Gowda** → <http://www.bbytes.com/training/> offers on site Python training courses. Pradeep has more than four years python programming experience including Django, Zope, Plone, Windows COM, Jython and Shell scripting. He has published Python articles in DeveloperIQ magazine. He has given python training beginners as well as experienced programmers across India. Pradeep currently lives in Indianapolis, IN.
- **CLEVERtrain** → <http://www.cleverdevil.org/train/> is a four-day Python training course taught by Jonathan LaCour → <http://www.cleverdevil.org/resume>, who is based out of Atlanta, GA.
- **Noah Gift** → <http://noahgift.com/> is based out of San Francisco and provides Python training in the Bay Area.
- **GreyCampus** → <http://www.greycampus.com/django-unchained-with-python-training-instructor-led> is a Dallas based training provider of Python.

Canada

- **Python Training Courses by Bernd Klein** → <http://www.python-training-courses.com/>: Open enrolment training courses in Toronto.
- **Computer Science Circles** → <http://www.cemc.uwaterloo.ca/resources/cscircles> is a series of fully-online lessons designed for beginners to Python programming. This can include groups of students working with a teacher, or individual students/adults. There are programming exercises which are automatically graded, and the content runs from 'Hello World' to recursion and efficiency.
- **An introductory workshop at Acadia University to encourage students, professors, and researchers** → http://euler.acadiau.ca/~064881j/py_wkshop/ alike to use the power of Python for their software endeavors.
- **Savoir-faire Linux** → <http://www.savoirfairelinux.com/> provides Python training courses → <http://training.savoirfairelinux.com/en/cours.php?id=SFL-DEV401> in english and french in Montreal, Quebec City, Ottawa and on-site. Customized Python training are also available on any Python-related subject. Contact training@savoirfairelinux.com → <mailto:training@savoirfairelinux.com> for more information.

- **SIAST GIS Certificate Program** → <http://www.siast.sk.ca/programsites/woodland/irm/gis.html> has a Introduction to Python course for Geographic Information Systems.
 - **Bodenseo** → <http://www.bodenseo.ca/> offers **Python Courses** → <http://ca.bodenseo.com/courses.php?topic=Python> in cooperation with Laila Zichmanis in the Toronto area.
- * **Web Age Solutions** → <https://www.webagesolutions.com/> offers instructor-led and virtual **Python training** → <http://www.webagesolutions.com/courses/python-training> in US and Canada. Our current offerings include Introduction to Python Programming → <https://www.webagesolutions.com/courses/TP1758-introduction-to-python-programming> and Advanced Python Programming → <https://www.webagesolutions.com/courses/TTPS4850-advanced-python-programming>. Web Age Solutions also provides customized training as per the needs of the organizations.

Europe

- **PythonSherpa** → <https://www.pythonsherpa.com/> offers Python training in The Netherlands. The instructor, Joris Hoendervangers → <https://www.linkedin.com/in/jorishoendervangers/>, has a background in financial markets.
- **eGenix.com** → <http://www.egenix.com/services/coaching/> offers customized Python training and coaching in Europe and around the world, with each course specifically designed to meet your particular needs. Please contact us → <http://www.egenix.com/company/contact/> for details.
- **Martin Jones** → <http://pythonforbiologists.com/index.php/training/>, author of **Python for Biologists** → <http://pythonforbiologists.com/index.php/introduction-to-python-for-biologists/>, specializes in Python training for students with a scientific background. He is available to teach both introductory and advanced Python courses. He has standard courses for beginners, or can put together bespoke courses tailored to your needs and interests. He is based in Scotland but will travel to deliver on-site training.
- **NEW! cmt GmbH** → <https://www.cmt.de/> offers high-quality python courses → <https://www.cmt.de/kategorien/it-trainings/programmierung/python> specially tailored to your needs. We are located in Germany, Austria and Switzerland. For further information please contact us via E-Mail (info@cmt.de → <mailto:info@info@cmt.de>), give us a call +49800 71 20000 (from within Germany) or visit our website.
- **Enthought's** → <http://www.enthought.com/> experts can train scientists, engineers, financial analysts, data scientists and geoscientists who would like to use Python more effectively. Enthought teaches public-open and private-onsite courses in cities around the US and EU. Current Enthought courses offered include:
 - **Python Foundations** → <https://www.enthought.com/training/course/python-foundations/>
 - **Python for Scientists and Engineers** → <https://www.enthought.com/training/course/python-for-scientists-and-engineers/>
 - **Python for Data Science** → <https://www.enthought.com/training/course/python-for-data-science/>
 - **Python for Data Analysis** → <https://www.enthought.com/training/course/python-for-data-analysis/>
 - **NEW! Machine Learning Mastery Workshop** → <https://www.enthought.com/training/course/machine-learning-mastery-workshop/>
 - **Pandas Mastery Workshop** → <https://www.enthought.com/training/course/pandas-mastery-workshop/> course is available upon request.
- Upcoming open classes are listed on the "Schedule" tab of each course's webpage → <https://www.enthought.com/python-training/>. If you are interested in scheduling a private-onsite course, please contact us → <https://www.enthought.com/contact/>. Enthought also has a set of 8 Pandas cheat sheets → <https://www.enthought.com/training/course/pandas-mastery-workshop/#pandas-cheat-sheet-download>, a set of 3 machine learning cheat sheets → <https://www.enthought.com/training/course/python-for-data-science/#ML-cheat-sheet-download>, and a MATLAB to Python white paper → <http://www.enthought.com/white-paper-matlab-to-python-2/> available for free download.

- **QA** → <http://www.qa.com/>, has two standard Python courses, one for **Python 2** → <http://www.qa.com/training-courses/technical-it-training/python/python-2-programming/> and one for **Python 3** → <http://www.qa.com/training-courses/technical-it-training/python/python-3-programming/> that run public schedules in their own premises in the UK. QA also provide customized on-site training which can include advanced topics.
- **Russel Winder** → <http://www.russel.org.uk/>, joint author of **Python for Rookies** → <http://www.pythonforrookies.org/> and other books, undertakes training and consultancy on Python programming in UK, Europe and worldwide. **SCons** → <http://www.scons.org/> (a Python-based build toolkit for C, C++, Fortran, D, Java, etc.) training and consultancy is also available.
- **PythonCourses.com** → <http://www.pythontcourses.com/> are teaching Python courses at various levels and for various audiences. For complete beginners to advanced users, from general python concepts to custom training designed to you specific needs or projects - we teach it. All our courses consist of lectures explaining the necessary theory without getting lost in IT jargon, followed by many hands-on exercises. We keep course groups small and put strong emphasis on personal interaction.
- **Antonio Cuni** → <http://antocuni.eu/> offers training and courses about Python and **PyPy** → <https://wiki.python.org/moin/PyPy> in Italy and Europe.
- **NobleProg Python Training** → <http://www.nobleprog.co.uk/python/training> provides public (UK, Poland and Worldwide), on-site and on-line instructor-led Python courses.
- **Mark Lutz** → <http://learning-python.com/> also teaches classes in Europe, Canada, Mexico, and other locations.
- **Michael Foord** → <http://www voidspace.org.uk/> is available for Python and **IronPython** → <https://wiki.python.org/moin/IronPython> training in the UK, Europe and the USA.
- **Graham Ellis** → <http://www.grahamellis.co.uk/> teaches public and private courses in the UK and private courses elsewhere in Europe on behalf of **Well House Consultants** → <http://www.wellho.net/>
- **Clockwork Software Systems** → <http://www.clocksoft.com/> run public scheduled and private courses mainly in the UK. Their sister company **The Linux Emporium** → http://www.linuxemporium.co.uk/products/training/python_training/ offer *no-frills* Python training courses which can be purchased on-line. The tutors for these courses, John Pinner and David Chan, are practising Python programmers.
- **Enable AI** → <https://enable-ai.de/> offers training courses in Artificial Intelligence, machine learning and data science in Germany. We are offering public courses, inhouse trainings and online webinars including **Introduction in Python** → <https://enable-ai.de/kurse/python-programmieren-lernen-schulung/>, **Machine Learning workshop** → <https://enable-ai.de/kurse/machine-learning-schulung:python>, **data science bootcamp** → <https://enable-ai.de/kurse/python-weiterbildung-data-science-bootcamp> or a **data science course with pandas and scikit-learn** → <https://enable-ai.de/kurse/python-data-science-schulung>. Advanced trainings are offered for **computer vision with deep learning (Keras)** → <https://enable-ai.de/kurse/deep-learning-ki-bilderkennung:python-schulung>, **introduction to deep learning (Keras)** → <https://enable-ai.de/kurse/deep-learning-neuronale-netze-python-ki-schulung> or an **introduction to artificial intelligence for executives** → <https://enable-ai.de/kurse/ki-manager-schulung>.

* **Python Academy** → <http://www.python-academy.com/> specializes in Python training and offers a wide variety of public as well customized in-house courses. The courses are held in **German** → <http://www.python-academy.de/> as well as in **English** → <http://www.python-academy.com/> at Python Academy's teaching center in Leipzig, Germany, in-house across Europe as well as **online**. Training topics include:

- Learning the Language
 - **Python for Non-Programmers** → http://www.python-academy.com/courses/python_course_nonprogrammers.html
 - **Python for Programmers** → http://www.python-academy.com/courses/python_course_programmers.html
 - **Advanced Python** → http://www.python-academy.com/courses/python_course_advanced.html
 - **Design Patterns in Python** → http://www.python-academy.com/courses/python_course_patterns.html
 - **Migration from Python 2 to 3** → http://www.python-academy.com/courses/python_course_migration_2to3.html

- [Functional Programming with Python](http://www.python-academy.com/courses/python_course_functional.html) → http://www.python-academy.com/courses/python_course_functional.html
- Data Science, Scientific Computing and Engineering
 - [Python for Scientists and Engineers](http://www.python-academy.com/courses/python_course_scientists.html) → http://www.python-academy.com/courses/python_course_scientists.html
 - [Image Processing with Python](http://www.python-academy.com/courses/python_course_image_processing.html) → http://www.python-academy.com/courses/python_course_image_processing.html
 - [Python for Data Analysis](https://www.python-academy.com/courses/python_course_data_analysis.html) → https://www.python-academy.com/courses/python_course_data_analysis.html
 - [Machine Learning with Python - A Comprehensive Introduction](https://www.python-academy.com/courses/python_course_machine_learning_intro.html) → https://www.python-academy.com/courses/python_course_machine_learning_intro.html
 - [High-Performance Computing with Python](https://www.python-academy.com/courses/python_course_hpc.html) → https://www.python-academy.com/courses/python_course_hpc.html
 - [Optimizing Python Programs](https://www.python-academy.com/courses/python_course_optimizing.html) → https://www.python-academy.com/courses/python_course_optimizing.html
 - [Python-Extensions with Other Languages](https://www.python-academy.com/courses/python_course_extensions.html) → https://www.python-academy.com/courses/python_course_extensions.html
 - [Cython in Depth](https://www.python-academy.com/courses/python_course_cython.html) → https://www.python-academy.com/courses/python_course_cython.html
- Software Development
 - [Professional Testing with Python](https://www.python-academy.com/courses/python_course_testing.html) → https://www.python-academy.com/courses/python_course_testing.html
 - [High Performance XML with Python](https://www.python-academy.com/courses/python_course_xml.html) → https://www.python-academy.com/courses/python_course_xml.html
 - [Database Programming with SQLAlchemy](https://www.python-academy.com/courses/python_course_sqlalchemy.html) → https://www.python-academy.com/courses/python_course_sqlalchemy.html
 - [Threads and Processes in Python](https://www.python-academy.com/courses/python_course_threads.html) → https://www.python-academy.com/courses/python_course_threads.html
- Web Development
 - [Introduction to Django](https://www.python-academy.com/courses/django_course_introduction.html) → https://www.python-academy.com/courses/django_course_introduction.html
 - [Advanced Django](https://www.python-academy.com/courses/django_course_advanced.html) → https://www.python-academy.com/courses/django_course_advanced.html
- **Mark Summerfield** → <http://www.qtrac.eu/>, author of [Programming in Python 3](http://www.qtrac.eu/py3book.html) → <http://www.qtrac.eu/py3book.html> and [Rapid GUI Programming with Python and Qt](http://www.qtrac.eu/pyqtbook.html) → <http://www.qtrac.eu/pyqtbook.html>, provides Python and PyQt → <https://wiki.python.org/moin/PyQt> training, research, and consultancy in English, primarily in the UK.
- **Christopher Arndt** → <http://chrisarndt.de/about.html> is an independent software developer and Linux consultant based in [Cologne, Germany](#) → <http://wiki.python.de/pyCologne>. He provides general Python, [TurboGears](#) → <https://wiki.python.org/moin/TurboGears> and Linux training in English or German language all over Europe on request. For further information write an email to chris at chrisarndt dot de.
- Despite [this notice](#) → http://www.deitel.com/training/python/python_1.html **Deitel & Associates** are not themselves offering Python training. Inquiries about Python training that they receive are forwarded to other training organizations.
- **Logilab** → <http://www.logilab.fr/formations/> offers Python training in France and Europe.
- **Klosterdata** → <http://www.kloster.se/kurser/k7026.html> offers Python training in Sweden and other Nordic Countries. The teacher, Fredrik Lundh, has long and deep experience from the Python language. He has taken part in the development of the language and used it for over ten years.
- **Philipp von Weitershausen** → <http://worldcookery.com/Training> offers Zope training all over Europe and the Americas. He is the author of the standard Zope textbook [Web Component Development with Zope 3](#) → <http://worldcookery.com/>. Courses can be held in English, German or Spanish.
- **Simon Willison** → <http://simonwillison.net/> is available for both [Django](#) → <http://www.djangoproject.com/> and Python training in the UK.

- **Ceintec** → <http://www.ceintec.com/> offers [Python programming](http://www.ceintec.com/curso_de_programacion_python_presencial_en_bilbao_bilbo_vizcaya_bizkaia_108.html) → http://www.ceintec.com/curso_de_programacion_python_presencial_en_bilbao_bilbo_vizcaya_bizkaia_108.html training classes in Spain for beginners and advanced developers.
- **Marcin Kaszynski** → <http://marcinkaszynski.com/> teaches Python and Django in English or Polish → <http://warsztatyit.pl/>.
- **Kristian Rother** → <http://www.rubor.de/> teaches Python, Biopython → <http://www.biopython.org/>, and software development practises in English, German, and Polish.
- **Svilen Dobrev** → <http://www.svilendobrev.com/> offers teaching and hands-on mentoring on python or software in general, accenting on efficiency and flexibility and team-work, in Bulgaria or anywhere. Languages: Bulgarian, English, Russian.
- **Python training at JBI Training UK** → <http://www.jbinternational.co.uk/Python-application-development-training-course.html> instructor-led classroom training (public and custom on-site courses) from Intro to Advanced level. Call +44 (0) 208 446 7555 and ask for Tom for more details and info on current offers.
- **Anaska / Alter Way Group** → <http://www.anaska.fr/formation-python-zope-plone.php> offers Python and Plone training courses and coaching in France and Europe.
- **Python training by IT-Schulungen.com** → <http://www.it-schulungen.com/> IT-Schulungen.com is a portal for IT-Trainings and offers training and consulting for Python from project experienced trainers as well as other open source technologies in Germany, Austria and Swiss. For further information please call +49 911 6500 8 222 (from within Germany) or visit our website.
- <http://www.it-seminare.de> → <http://www.it-seminare.de/> Python training for individuals or companies - For more than 10 years, we offer more than 1.000 different trainings - for more information, please visit our website.
- **Bodenseo** → <https://www.bodenseo.com/courses.php?topic=Python>, - a training centre situated in the heart of Europe on the shores of Lake Constance, where Switzerland, Austria and Germany meet - offers public scheduled courses at Lake Constance and on-site seminars all over Europe (e.g. Switzerland, Austria, Germany, France, Luxemburg and the UK) in English, German and French. The Python courses of Bodenseo aim both at [beginners](https://www.bodenseo.com/course/python_training_course.html) → https://www.bodenseo.com/course/python_training_course.html and [advanced learners](https://www.bodenseo.com/course/python_training_course_intermediate.html) → https://www.bodenseo.com/course/python_training_course_intermediate.html of Python. The courses are also offered in various levels: [Python Course: Level I](https://www.bodenseo.com/course/python_course_level_i.html) → https://www.bodenseo.com/course/python_course_level_i.html, [Python Course: Level II](https://www.bodenseo.com/course/python_course_level_ii.html) → https://www.bodenseo.com/course/python_course_level_ii.html and [Python Course: Level III](https://www.bodenseo.com/course/python_course_level_iii.html) → https://www.bodenseo.com/course/python_course_level_iii.html. There are special training courses on [Text Processing with Python](https://www.bodenseo.com/course/python_text_processing_course.html) → https://www.bodenseo.com/course/python_text_processing_course.html, [Python Course for Data Analysis and Machine Learning](https://www.bodenseo.com/course/python_data_analysis_machine_learning.html) → https://www.bodenseo.com/course/python_data_analysis_machine_learning.html, [Python and Bash Programming](https://www.bodenseo.com/course/python_and_bash_programming.html) → https://www.bodenseo.com/course/python_and_bash_programming.html, [Python Shell](https://www.bodenseo.com/course/python_shell.html) → https://www.bodenseo.com/course/python_shell.html, [Django Python Web Application Framework](https://www.bodenseo.com/course/django_and_python_course.html) → https://www.bodenseo.com/course/django_and_python_course.html. For those who want to learn both C++ and Python Bodenseo offers [Python and C++ Course](https://www.bodenseo.com/course/python_c_course.html) → https://www.bodenseo.com/course/python_c_course.html. Other cross-thematic are are [Introduction to Tkinter](https://www.bodenseo.com/course/python_tkinter.html) → https://www.bodenseo.com/course/python_tkinter.html and [Python and XML Course](https://www.bodenseo.com/course/python_and_xml_course.html) → https://www.bodenseo.com/course/python_and_xml_course.html.
- **Framework Training Ltd** → <https://www.frameworktraining.co.uk/> delivers hands-on [Python training courses](https://www.frameworktraining.co.uk/python-training-course-uk/) → [http://www.frameworktraining.co.uk/python-training-course-uk/](https://www.frameworktraining.co.uk/python-training-course-uk/) and [Python Data Science Analysis & ML training courses](https://www.frameworktraining.co.uk/courses/data/data-science/data-science-python-training-course/) → <https://www.frameworktraining.co.uk/courses/data/data-science/data-science-python-training-course/> and many other [Coding training courses](https://www.frameworktraining.co.uk/courses/coding/) → <https://www.frameworktraining.co.uk/courses/coding/> at their London training centre. All courses can be customised for virtual classroom / remote / on-site delivery (UK, Europe, Worldwide), tailored to take into account existing skills, learning goals and project requirements. Call on +44 (0) 20 3137 3920 for more information.
- **G FU IT Schulung** → <http://www.gfu.net/> offers open and company trainings in the category of [Python Schulung](http://www.gfu.net/python/python-schulung.html) → <http://www.gfu.net/python/python-schulung.html> seminars all over Europe.

- **Tomasz Puton** → <http://www.tomasz-puton.pl/> teaches Python, **Django** → <http://www.djangoproject.com/> including **GeoDjango** → <https://wiki.python.org/moin/GeoDjango>, scientific programming in Python and software development.
- **Sixty North** → <http://sixty-north.com/> is a software development company based in Norway. We offer Python consulting and training services worldwide, with specific expertise in scientific, oil & gas, and geospatial topics as well as C++/Python integration.
- **CBT Training & Consulting** → <http://www.cbt-training.de/> as an IT Training Company offers open and company trainings in the category of **Python Schulung** → <http://www.cbt-training.de/Seminare/Programmierung.html> seminars in Germany.
- **RoPython** → <http://ropython.org/> does trainings, workshops and conferences in Romania.
- **Bernd Klein** → http://www.python-course.eu/python_classes.php offers Python training courses in Germany, Austria, Switzerland, France, Luxembourg, Belgium, The Netherlands, England, and Canada.
- **HECKER CONSULTING** → <https://www.hco.de/> offers Live-Online / Remote **Python training courses** → <https://www.hco.de/beratung-coaching-workshop-training/python>.
 - **Python Programming for Beginners** → <https://www.hco.de/leistungen/python-fur-einsteiger-beratung-coaching-workshop-training>
 - **Python Programming Basics** → <https://www.hco.de/leistungen/python-grundlagen-beratung-coaching-workshop-training>
 - **Automation (RPA) with Python** → <https://www.hco.de/leistungen/automatisierung-rpa-mit-python-beratung-coaching-workshop-training>
 - **Data Science with Python** → <https://www.hco.de/leistungen/data-science-mit-python-beratung-coaching-workshop-training>
 - **Machine Learning (ML) with Python** → <https://www.hco.de/leistungen/maschinelles-lernen-ml-mit-python-beratung-coaching-workshop-training>

Australia

- **Python Charmers** → <http://pythoncharmers.com/> specializes in Python training for scientists and engineers in the Asia-Pacific region, based in Melbourne and Singapore. The principal trainer is Dr Edward Schofield. Ed is a well-known contributor in the **NumPy** → <https://wiki.python.org/moin/NumPy> and **SciPy** → <https://wiki.python.org/moin/SciPy> communities; he was the release manager for five releases of **SciPy** → <https://wiki.python.org/moin/SciPy> in 2005-6 and the author of the maximum entropy and parts of the sparse matrix module. He has 20 years of experience in programming, teaching, and public speaking, including 9 years of experience with Python. Ed holds a PhD in statistical pattern recognition from Imperial College London and an MA in mathematics and computer science from Cambridge University. He and Python Charmers' other topic specialists are available to travel globally. [More info](http://pythoncharmers.com/training) → <http://pythoncharmers.com/training>.
- **Peter Lovett** of **Plus Plus** → <http://wwwplusplus.com.au/> offers **Introductory** → <http://wwwplusplus.com.au/prodpy.htm>, **Intermediate** → <http://wwwplusplus.com.au/prodpyi.htm> and **Advanced** → <http://wwwplusplus.com.au/prodpya.htm> level Python courses (as well as other languages). He is an accomplished trainer, running computer programming courses since 1985, in C, C++, Perl, Python, Java and XML around Australia, New Zealand and England, and is available for travel. As well as being highly accomplished technically, Peter is also highly effective at communicating, and gets glowing reviews for his courses.

Latin America

- **Facundo Batista** → <http://www.taniquetil.com.ar/facundo/cursoPython.html> teaches introduction and advanced courses of Python to different types of organization (universities, schools, enterprises) in Argentina and other locations.

- **LucianoRamalho** → <https://wiki.python.org/moin/LucianoRamalho> teaches Python in the context of Zope and Plone training offered by **Simples Consultoria** → <http://simplesconsultoria.com.br/>. Ramalho also created the **Turing Club** → <http://turing-club.org/>, which promotes computer programming as a hobby and uses Python in many of its activities.
- **Alfonso de la Guarda** → <http://www.cosperu.com/> The COS is the first Center that offers free online flash courses about FLOSS, including streaming video on real time. The courses are many: python basics, advance, turbogears, django, gtk, wxglade, etc. Their director, Alfonso de la Guarda, is python programmer since 1990 and has develop many GPL applications as: sisventi, sisgerpy, itv, edukt and recently is working with OLPC.
- **Trianguli Software Livre** → <http://www.trianguli.com.br/> A free software company, offers Python training in many skills (from basic to advanced or custom courses to special needs), Django, Twisted and other Python services. The founder, Christiano Anderson is Python programmer and free software evangelist.
- **Menttes** → <http://www.menttes.com/> A free software company, offers high quality Python, Zope and Plone development, consulting, coaching and training. The directors are **Roberto Allende** → <http://robertoallende.com/>, member of **Plone Foundation** → <http://plone.org/> and Emanuel Sartor. Both are co-founders of **Plone Cono Sur** → <http://plone.org/countries/conosur>.

India

- **Kiran Chandra Chitmallu** → <http://www.indiatrainings.in/technology/python/> conducts online & classroom training classes for Graduate students in various python frameworks. **KiranChandra** → <https://wiki.python.org/moin/KiranChandra> has got 6 years experience in the field of python data structures, algorithms and networking. He Currently Runs **IndiaTrainings** → <http://www.indiatrainings.in/> as a platform for improving career opportunities for graduates by **training python in hyderabad, Telangana** → <http://www.indiatrainings.in/technology/python>.
- **Anuraag Singh** → <http://www.seoclick.com/python/> conducts training and classes for CBSE Class XI Python language, which is introduced from this year 2013 as per new syllabus changes in CBSE computer science subject. Anuraag has got 12 years experience in the field of programming and is founder of many software companies.
- **Rahul Verma** → <http://www.testingperspective.com/> conducts **Python training for Software testers, in India** → <http://www.testingperspective.com/?p=926>, focused on Implementation of general purpose test automation frameworks and day-to-day test automation requirements. You can find more details about his courses at **Testing Perspective website** → http://www.testingperspective.com/?page_id=634 and on Python specifically at **Python Scripting for Software Testers** → <http://www.testingperspective.com/?p=926>. Rahul has presented at various conferences including Google Test Automation Conference, PyCon → <https://wiki.python.org/moin/PyCon> India, CONQUEST Germany and all Indian testing conferences. He is the author of **How Would Pareto Learn Python** → <http://hwplibpython.testingperspective.com/> and **Design Patterns in Python** → <http://dppip.testingperspective.com/> which are available as free online books on his website. He is also a regular columnist with Testing Experience magazine on the subject of test automation.
- **Vasudev Ram (Dancing Bison Enterprises)** → <https://vasudevram.github.io/>, a Fellow → <https://www.python.org/p-sf/members/> of the **Python Software Foundation** → <http://www.python.org/psf>, offers Python (and other) training courses in person or worldwide via the Internet using email and shared documents for lesson delivery, and VoIP and IM tools for interaction. He's a developer with many years experience, has published articles on Packt Publishing, IBM developerWorks and Linux For You (now Open Source For You), has a blog with lots of **Python posts** → <https://jugad2.blogspot.com/search/label/python> and is the creator of some **software products** → <https://vasudevram.github.io/products.html>, including **xtopdf** → <https://vasudevram.github.io/products.html#xtopdf>, a Python toolkit for PDF creation from other data formats. xtopdf is used by some well-known organizations and others. This is his **LinkedIn profile**, → <http://www.linkedin.com/in/vasudevram>

- Anand Chitipothu → <http://anandology.com/> offers Python training courses in Bangalore, India. He is the co-author of web.py → <http://webpy.org/>, an open-source web framework in python. He is a software consultant and trainer with more than 10 years of programming experience, including 6+ years in Python. He conducts public python trainings → <http://anandology.com/trainings/> in Bangalore on semi-regular basis.
- Biztech Academy → <http://biztech.co.in/> offers Python training courses in Rajkot,Gujarat, India. they are also providing open source courses on Linux, Apache,Mysql,PHP,Perl,Python,Ajax.
- Noufal Ibrahim → <https://thelycaeum.in/> is the founder of "PyCon" → <https://wiki.python.org/moin/PyCon India>" and has been using Python since 2002. He offers corporate training, campus workshops, public classes and mentors fresh engineering graduates to help them find jobs via his company Lycaeum → <https://thelycaeum.in/> based in Kozhikode, Kerala.

Iran

- GoToClass.ir → <http://gotoclass.ir/> offers online traning for Persian Speakers, this time we offer free online python course for biginners, this course is a "Introduction to programming using python" → <http://gotoclass.ir/courses/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D9%BE%D8%A7%DB%8C%D8%AA%D9%88%D9%86/>, JADI → <http://jadi.net/> teach you all that need to start programming in python in a interactive course, by sign up in this course you have access to videos, documents, assessments, quizes, exams and a discusion for being in touch with our teacher and classmates. if you are a persion speaker and you need to start programming using Python, don't miss this course.

Israel

- Yoav Ram, PhD → <http://python.yoavram.com/> offers training in USA and Israel with focus on numerical, scientific, and statistical applications, including web application and user interfaces. To enhance participant learning, all course material is fully interactive, using Jupyter notebooks, and all lectures include hands-on exercises. Yoav is a postdoc at Stanford University, earned his PhD from Tel-Aviv University, working with Python since 2002, training and teaching Scientific Python since 2011, including NumPy → <https://wiki.python.org/moin/NumPy>, SciPy → <https://wiki.python.org/moin/SciPy>, Matplotlib, Jupyter, Pandas, Seaborn, Cython, scikit-image, scikit-learn, TensorFlow → <https://wiki.python.org/moin/TensorFlow>, Flask, Click, Tk. Experience with both small and enterprise companies.
- Udi Oron (10x.org.il) → <https://www.10x.org.il/training/> offers on-site python training for a wide range of participants, from non-programmers to experienced software developers. Complete syllabuses are available here → <https://www.10x.org.il/training/>.
- Avner Ben → <http://www.swskilltree.org/Design/1/2/44.html>, author of the "Skill-Driven" software design method, has been instructing object-oriented programming and design since 1988, design patterns since 1999 and, recently - "Python for designers". This four day course is intended for programmers with previous knowledge of object oriented programming. It concentrates on features that make Python the designer's language of choice for both explorative development and implementation of object-oriented design. See course foils → <http://www.swskilltree.org/Python/0.html>.
- Ram Rachum, Python Workshops → <http://pythonworkshops.co/> Hi! My name is Ram Rachum, and I've been working with Python since 2009. As an escape from my life of corporate imprisonment, I like to give workshops in which I teach basic and advanced Python usage. I usually teach teams of 10-20 developers / QA people / database people. Sometimes companies bring me to teach experienced programmers who need to up their Python game, and sometimes complete beginners who have never programmed before and want

to get started with Python. Email me at ram@rachum.com so we could talk about what kind of workshop would be the best for your team. Thanks! (Hebrew website – <http://pythonworkshops.co.il/>.)

Malaysia

- R. Ramesh Kumar → <mailto:rameshkumar@techdynamics.com.my> provides Python classes through his company **Tech Dynamics** → <http://www.techdynamics.com.my>. Primarily a developer, he has a passion for teaching and sharing his knowledge and experience. Ramesh also teaches other programming languages, and Linux/Unix courses.
- Pytech Resources → <http://pytechresources.com/> conducts introductory and advanced Python courses in Malaysia and internationally. Its principal trainer, **P.C. Boey** → <mailto:pcboey@yahoo.com>, has more than 12 years programming experience with Python and his courses on Python programming have been very well-received. He is also available to do web development, training and consultancy work with Django.
- Low Kian Seong → <mailto:kianseong@gmail.com> conducts beginner, intermediate and advanced Python courses in Malaysia under his own company Squid Consulting and Integration affectionately known as **SqCI** → <http://www.sqci.biz>. He has been using Python since 2000 starting from Zope ending up in Django and have been deploying projects in MNCs all around the country.

Nigeria

- <http://www.ascomt.com> → <http://www.ascomt.com> ASCOMT NIGERIA offers training and consulting for Python in Nigeria and the rest of Africa for small and large companies with focus on numerical, scientific, and statistical applications, including data mining and artificial intelligence. Our training service include **NumPy** → <https://wiki.python.org/moin/NumPy>, **SciPy** → <https://wiki.python.org/moin/SciPy>, Matplotlib, Pandas, Seaborn, Scikit-learn, and other popular Python packages. Our consulting service include offering innovative, value driven solutions that help clients manage and analyse data. For further information please visit our websites – <http://www.ascomt.com>.

Singapore

- **Python Charmers** → <http://pythoncharmers.com/> specializes in Python training for scientists and engineers in the Asia-Pacific region, based in Melbourne and Singapore. The principal trainer is Dr Edward Schofield. Ed is a well-known contributor in the **NumPy** → <https://wiki.python.org/moin/NumPy> and **SciPy** → <https://wiki.python.org/moin/SciPy> communities; he was the release manager for five releases of **SciPy** → <https://wiki.python.org/moin/SciPy> in 2005-6 and the author of the maximum entropy and parts of the sparse matrix module. He has 20 years of experience in programming, teaching, and public speaking, including 9 years of experience with Python. Ed holds a PhD in statistical pattern recognition from Imperial College London and an MA in mathematics and computer science from Cambridge University. He and Python Charmers' other topic specialists are available to travel globally. **More info** → <http://pythoncharmers.com/training>.

New Zealand

- Peter Lovett of **Plus Plus** → <http://www.plusplus.com.au> offers **Introductory** → <http://www.plusplus.com.au/prodpy.htm>, **Intermediate** → <http://www.plusplus.com.au/prodpyi.htm> and **Advanced** → <http://www.plusplus.com.au/prodpya.htm> level Python courses (as well as other languages). He is an accomplished trainer, running

computer programming courses since 1985, in C, C++, Perl, Python, Java and XML. Although based in Sydney, Australia, he regularly travels to New Zealand, and is available for training. As well as being highly accomplished technically, Peter is also highly effective at communicating, and gets glowing reviews for his courses.

Turkey

- [Hakan Ozen](http://hakanozen.com/) → <http://hakanozen.com/> is available for teaching Python.

Internet - Python Online Training

* **Marilyn Davis, Ph.D.** → <http://www.pythontrainer.com/> Marilyn specializes in Python training in corporate environments in the Silicon Valley, and anywhere, either through [UCSC-Extension](http://www.ucsc-extension.edu/) → <http://www.ucsc-extension.edu/>, where students earn University credit, or independently. She has taught Python at Apple, Cisco, Facebook, Google, Intuit, LLNL, NASA, Nokia, Oracle, Plantronics, Skype, VMware, and more.

- [Python For Programmers](https://course.ucsc-extension.edu/modules/shop/index.html?action=section&OfferingID=5591127&SectionID=7590317) → <https://course.ucsc-extension.edu/modules/shop/index.html?action=section&OfferingID=5591127&SectionID=7590317>, where students learn and practice core concepts and Pythonic thinking: **Sep 13, 2022 - Dec 13, 2022**.
- Here's a [sample lecture](https://www.youtube.com/watch?v=t6pdG1oWFX4). → <https://www.youtube.com/watch?v=t6pdG1oWFX4> [PythonTrainer.Com](http://www.pythontrainer.com/) → <http://www.pythontrainer.com/> is a 100% woman-owned business, registered in [SAM](#) → <https://www.sam.gov/portal/public/SAM/>.

Thinkful → <http://www.thinkful.com/> offers training in Python for web programming and data science, Courses are generally student-led with regular one-on-one meetings with a mentor, but a bootcamp option is also available.

Firebox Training consistently delivers [Python training courses](#) → <https://www.fireboxtraining.com/python> as live, instructor-led online, on-site, and classroom style training classes for individuals and groups. **Courses Offered:**

- [Introduction to Advanced Python Programming Bootcamp 5-day - view detailed outline](https://www.fireboxtraining.com/python-training-bootcamps). → <https://www.fireboxtraining.com/python-training-bootcamps>
- [Introduction to Python Programming Training 3-day - view detailed outline](https://www.fireboxtraining.com/python-training-introduction-class). → <https://www.fireboxtraining.com/python-training-introduction-class>
- [Advanced Python Programming Training 3-day - view detailed outline](https://www.fireboxtraining.com/python-training-advanced-class). → <https://www.fireboxtraining.com/python-training-advanced-class>
- [Python Programming Training for Scientist, Engineers and Analyst 5-Day - view detailed outline](https://www.fireboxtraining.com/python-training-for-scientists-engineers-analyst). → <https://www.fireboxtraining.com/python-training-for-scientists-engineers-analyst>
- [Python Programming Training for Scientist, Engineers and Analyst 3-Day - view detailed outline](https://www.fireboxtraining.com/python-training-for-scientist-engineers-and-analyst-3-day-view-detailed-outline). → <https://www.fireboxtraining.com/python-training-for-scientist-engineers-and-analyst-3-day-view-detailed-outline>

* **Batky-Howell** → <http://www.batky-howell.com/> has been training IT Professional's for over 20 years training over 40,000 students in over 3,200 courses with live instructor led online training. At [Batky-Howell](http://www.batky-howell.com/) → <http://www.batky-howell.com/> we offer three unique live online instructor led [Python training courses](#) → <http://www.batky-howell.com/courses/python.html>. Our 4 day intro course, [Python Training I: Essentials](#) → <http://www.batky-howell.com/courses/python/python-i-essentials.html> leads the student from the basics of writing and running [Python](#) → <http://www.batky-howell.com/courses/python.html> scripts to more advanced features such as file operations, regular expressions, working with binary data, and using the extensive functionality of [Python Training](#) → <http://www.batky-howell.com/courses/python.html> modules. We offer two additional advanced courses: [Python Training II: Applied Python](#) → <http://www.batky-howell.com/courses/python-ii-applied-python.html>, a 4 day course that puts extra emphasis placed on features unique to [Python](#) → <http://www.batky-howell.com/courses/python.html>, such as tuples, array slices, and output

formatting and our newest release [Python Training for Scientists and Engineers](http://www.batky-howell.com/courses/python/python-for-scientists-and-engineers.html) → <http://www.batky-howell.com/courses/python/python-for-scientists-and-engineers.html> that is an intense 5 day course that teaches scientists and engineers the [Python](http://www.batky-howell.com/courses/python.html) → <http://www.batky-howell.com/courses/python.html>, [NumPy](https://wiki.python.org/moin/NumPy) → <https://wiki.python.org/moin/NumPy>, [SciPy](https://wiki.python.org/moin/SciPy) → <https://wiki.python.org/moin/SciPy>, [SymPy](https://wiki.python.org/moin/Sympy) → <https://wiki.python.org/moin/Sympy>, and other skills they need to work with data, manipulating arrays, performing statistical calculations, and plotting results.

- See [Computer Science Circles](http://cemclinux1.math.uwaterloo.ca/~cscircles/wordpress/) → <http://cemclinux1.math.uwaterloo.ca/~cscircles/wordpress/> above, which are online-only.
- The [Open Technology Group, Inc](http://www.otg-nc.com/python-training) → <http://www.otg-nc.com/python-training> offers all its courses for virtual LIVE instructor-led delivery via the Internet. Students may attend courses from anywhere, worldwide, provided they have a broadband Internet connection available. [Python Bootcamps](http://www.otg-nc.com/python-bootcamp) → <http://www.otg-nc.com/python-bootcamp>, [Advanced Python](http://www.otg-nc.com/advanced-python-training) → <http://www.otg-nc.com/advanced-python-training>, [Introduction to Django](http://www.otg-nc.com/introduction-to-django-training) → <http://www.otg-nc.com/introduction-to-django-training>, and [GeoDjango training](http://www.otg-nc.com/geodjango-training) → <http://www.otg-nc.com/geodjango-training>. courses are scheduled to run **Mar 7-11, Apr 18-22, May 23-27, and June 20-24**. *All courses are available for both in-person as well as Virtual LIVE instructor-led delivery, which may be attended from ANYWHERE.* Contact us at info@otg-nc.com → <mailto:info@otg-nc.com> for global, customized, on-site delivery.

[Python Academy](http://www.python-academy.com/) → <http://www.python-academy.com/> specializes in Python training and offers a wide variety of public as well customized in-house courses. Training topics include a solid [Introduction to Python](http://www.python-academy.com/courses/python_course_programmers.html) → http://www.python-academy.com/courses/python_course_programmers.html and [Advanced Python](http://www.python-academy.com/courses/specialtopics/python_course_advanced.html) → http://www.python-academy.com/courses/specialtopics/python_course_advanced.html to dive deeper into the language. Furthermore, courses are available for [Cython](http://www.python-academy.com/courses/specialtopics/python_course_cython.html) → http://www.python-academy.com/courses/specialtopics/python_course_cython.html as well as [Testing with pytest and tox](http://www.python-academy.com/courses/specialtopics/python_course_testing.html) → http://www.python-academy.com/courses/specialtopics/python_course_testing.html. Scientists and engineers can learn about what offers Python in the course [Python for Scientists and Engineers](http://www.python-academy.com/courses/python_course_scientists.html) → http://www.python-academy.com/courses/python_course_scientists.html. Python can be very fast as the course [High Performance Computation with Python](http://www.python-academy.com/courses/python_course_high_performance.html) → http://www.python-academy.com/courses/python_course_high_performance.html proves. [Introduction to Django](http://www.python-academy.com/courses/django_course_introduction.html) → http://www.python-academy.com/courses/django_course_introduction.html provides all the basics to get started developing professional web applications. Even more insight into this powerful Python web framework offers [Advanced Django](http://www.python-academy.com/courses/django_course_advanced.html) → http://www.python-academy.com/courses/django_course_advanced.html. These courses are held **online**.

A growing wealth of other "Audio/Video Instructional Materials for Python" → <http://www.python.org/doc/av/>" are becoming available, including a variety of podcasts and screencasts.

* [GoSkills](http://www.goskills.com/) → <http://www.goskills.com/> is an online learning platform that helps anyone learn business skills to reach their personal and professional goals. With a [GoSkills](https://wiki.python.org/moin/GoSkills) → <https://wiki.python.org/moin/GoSkills>.com subscription, members have access to over 80+ courses consisting of bite-sized and interactive content.

[GoSkills](https://wiki.python.org/moin/GoSkills) → <https://wiki.python.org/moin/GoSkills> has three beginner-friendly, bite-sized Python courses: [Intro to Python](https://www.goskills.com/Course/Intro-Python) → <https://www.goskills.com/Course/Intro-Python>, [Introduction to Data Analysis with Python](https://www.goskills.com/Course/Python-Data-Analysis) → <https://www.goskills.com/Course/Python-Data-Analysis>, and [Python with Excel](https://www.goskills.com/Course/Python-Excel) → <https://www.goskills.com/Course/Python-Excel>.

* [DataCamp](http://www.datacamp.com/) → <http://www.datacamp.com/> is an interactive platform to learn data science that has already trained over 600,000 students and offers more than 250 hours of course content, which has been built out with academic and corporate partners such as Continuum Analytics, RStudio and Microsoft, and experts from companies like Pfizer, Liberty Mutual, H2O, DataRobot, and many other leaders.

DataCamp has recently been building a collection of cheat sheets for Python learners: right now, they have a [Python for Data Science Cheat Sheet](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PythonForDataScience.pdf) → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PythonForDataScience.pdf, a [Pandas Cheat Sheet](https://assets.datacamp.com/blog_assets/PandasPythonForDataScience.pdf) → https://assets.datacamp.com/blog_assets/PandasPythonForDataScience.pdf and a [Bokeh Cheat Sheet](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Bokeh_Cheat_Sheet.pdf) → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Bokeh_Cheat_Sheet.pdf available.

Python Trainers, Promote Thyself

Many trainers are individuals or small companies, and it can be hard to get the attention of the big IT houses. While skill credentials and a portfolio of past training gigs are important, perhaps one of the best promoters is when someone has actually experienced one of your classes. They gain insight into your speaking style, how you relate to the students and your ability to explain complex technical subjects in an approachable way. No class syllabus can convey that. The Python community has a valuable resource that can give you the next best thing.

Marilyn Davis, Ph.D., of [PythonTrainer](https://wiki.python.org/moin/PythonTrainer) → <https://wiki.python.org/moin/PythonTrainer>, offers a free course: **Python For Programmers** Check it out:<https://www.youtube.com/watch?v=e7J0e0ovYAI>

||Screencasting is a multimedia creation that focuses on the instructor's desktop, with voiceover guidance. It can be in the format of an online slideshow, a guided sourcecode walkthrough or a follow-along interactive session. They can be as long or short as you wish and they have opportunities for branding, by using custom wallpaper behind your talks desktop and musical lead-in/fade-out.

Screencasts can be hosted on video.google.com → <http://video.google.com/>. They can also be embedded in your website while hosted elsewhere, as shown in our **5-Minutes with Python** → <http://www.python.org/doc/av/5minutes/> series.

But perhaps you're really busy on current projects and short on time. Consider doing an audio interview about an upcoming seminar you're offering and releasing it as a podcast. Ron Stephens of Python 411 makes available an excellent collection of podcasts and may be interested in hosting yours.

Unlike face-to-face presentation opportunities, screencasts/podcasts have the additional benefit that they promote your training offerings while you're busy on other gigs. It's almost like cloning yourself and having more time for promotion. It's all about leverage. Use it. ||

Public training courses presented by these trainers are listed on the [PythonEvents](https://wiki.python.org/moin/PythonEvents) → <https://wiki.python.org/moin/PythonEvents> page.

Before creation of this page, [CameronLaird](https://wiki.python.org/moin/CameronLaird) → <https://wiki.python.org/moin/CameronLaird> maintained [a private one](http://phaseit.net/claird/comp.lang.python/python_training.html) → http://phaseit.net/claird/comp.lang.python/python_training.html on the same subject.

[CategoryPythonInBusiness](https://wiki.python.org/moin/CategoryPythonInBusiness) → <https://wiki.python.org/moin/CategoryPythonInBusiness>

BeginnersGuide/Download - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Download>

Downloading Python

On many systems Python comes pre-installed, you can try running the `python` command to start the Python interpreter to check and see if it is already installed. On windows you can try the `py` command which is a launcher which is more likely to work. If it is installed you will see a response which will include the version number, for example:

```
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit  
(AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.
```

If you don't see this, you will need to install Python on your system.

If the version number is Python 2.x.y (where x and y are any number) you are using Python 2 which is no longer supported and is not a good choice for development. You can try running `python3` to see if there is also a Python 3.x.y version installed, if not you'll want to install the latest version of Python.

If you do not have Python installed or need a newer version you can go to:

<https://www.python.org/downloads/>

which will provide a button to download an installer for your particular system. The Python documentation also has a detailed guide on how to install and setup Python here:

<https://docs.python.org/3/using/index.html>

Below are some system specific notes to keep in mind.

Windows

On Windows the most stable build is available from the official download page

<https://www.python.org/downloads/>

You should download and run the installer from that page to get the latest version of Python for your system. You can refer to the Python documentation for more details on the installation process and getting started:

<https://docs.python.org/3/using/windows.html>

Mac

For macOS 10.9 (Jaguar) up until 12.3 (Catalina) the operating system includes Python 2, which is no longer supported and is not a good choice for development. You should go to do the downloads page: <https://www.python.org/downloads/> and download the installer.

For newer versions of macOS, Python is no longer included by default and you will have to download and install it. You can refer to the Python documentation for more details on the installation process and getting started:

<https://docs.python.org/3/using/mac.html>

Linux

On most Linux distributions Python comes pre-installed and/or available via the distribution's package managers. Below are some common examples, but refer to your specific distribution's documentation and package list to get the most up to date instructions.

If you'd like to download and build Python from source (or your distribution's package manager does not include a version of Python you need) you can download a source tarball from the general download page: <https://www.python.org/downloads/>

Red Hat, CentOS, or Fedora

```
dnf install python3 python3-devel
```

Debian or Ubuntu

```
apt-get install python3 python3-dev
```

Gentoo

```
emerge dev-lang/python
```

Arch Linux

```
pacman -S python3
```

BeginnersGuide/Examples - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Examples>

Python Examples and Sample Code

When you're learning, small examples can be very helpful.

- [The Python Standard Library](https://web.archive.org/web/20201017142948/http://effbot.org/zone/librarybook-index.htm) → <https://web.archive.org/web/20201017142948/http://effbot.org/zone/librarybook-index.htm> (archived copy), an electronically published book by Fredrik Lundh, examines most of the modules in Python's standard library, describing what the module does and giving a short example of its use. Note that this book is now relatively old and so misses a lot of the developments of the last two decades.
- The [Python Recipes](https://code.activestate.com/recipes/langs/python/) → <https://code.activestate.com/recipes/langs/python/>, from [ActiveState](https://wiki.python.org/moin/ActiveState) → <https://wiki.python.org/moin/ActiveState>, is a very large collection of code snippets, some elementary and some advanced.

BeginnersGuide/Help - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Help>

Getting Help

If you experience problems using Python, your first step should be to check the documentation.

- The [Python documentation](https://www.python.org/doc/versions/) → <https://www.python.org/doc/versions/> describes the language and the standard modules in detail.
- Many common questions are answered in the [Frequently Asked Question lists](http://docs.python.org/faq/) → <http://docs.python.org/faq/>.
- If you suspect you've discovered a bug in the Python core, search the [Python Bug Tracker](http://bugs.python.org/) → <http://bugs.python.org/> to find out if it's already been reported.

If the documentation doesn't help, here's how to get help from real people:

- [comp.lang.python](#) → Post to the Python newsgroup, renowned for its friendly and helpful atmosphere. See [the guide to Python mailing lists](https://www.python.org/community/lists/#comp-lang:python) → <https://www.python.org/community/lists/#comp-lang:python> for more information.
- [tutor mailing list](#) → <http://mail.python.org/mailman/listinfo/tutor> A moderate-volume mailing list for folks who want to ask questions while learning computer programming with Python. See [the guide to Python mailing lists](https://www.python.org/community/lists/#tutor) → <http://www.python.org/community/lists/#tutor> for more information.
- help@python.org → mailto:help@python.org The Python help desk. You can ask a group of knowledgeable volunteers questions about all your Python problems. See [the guide to Python mailing lists](https://www.python.org/community/lists/#python-help) → <http://www.python.org/community/lists/#python-help> for more information.

webmaster is not a Python language support channel!!!

- The webmaster should only be contacted if you have found a problem with the python.org website such as a broken link, typographic error, or other problem.
- For such problems, also check [the github issue tracker for the website](https://github.com/python/pythondotorg/issues) → <https://github.com/python/pythondotorg/issues>

BeginnersGuide/Mathematics - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Mathematics>

BeginnersGuide/Mathematics (last edited 2019-06-20 20:13:37 by [IgorRocha](#) → <https://wiki.python.org/moin/IgorRocha>)

BeginnersGuide/NonProgrammers - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers>

Python for Non-Programmers

If you've never programmed before, the tutorials on this page are recommended for you; they don't assume that you have previous experience. If you have programming experience, also check out the [BeginnersGuide/Programmers](#) → <https://wiki.python.org/moin/BeginnersGuide/Programmers> page.

Books

Each of these books can be purchased online but is also available as free textual, website, or video content.

- **Automate the Boring Stuff with Python - Practical Programming for Total Beginners** by *Al Sweigart* is "written for office workers, students, administrators, and anyone who uses a computer to learn how to code small, practical programs to automate tasks on their computer." ||website → <https://automatetheboringstuff.com/> ||print version → <http://www.amazon.com/gp/product/1593275994/> ||
- **How To Think Like a Computer Scientist** is a classic open-source book by *Allen Downey* with contributions from *Jeffrey Elkner* and *Chris Meyers*. It was updated to Python 3 by *Peter Wentworth*. ||website → <http://openbookproject.net/thinkcs/python/english3e/> ||print version → <http://openbookproject.net/thinkcs/python/english3e/> ||
- **Making Games with Python & Pygame** by *Al Sweigart* introduces the Pygame framework for novices and intermediate programmers to make graphical games. ||website → <http://inventwithpython.com/pygame> ||print version → <http://www.amazon.com/Making-Games-Python-Pygame-Sweigart/dp/1469901730?ie=UTF8&tag=playwithpyt-20&linkCode=as2&camp=1789&creative=9325&creativeASIN=0982106017> ||
- **Python One-Liners** by *Christian Mayer* teaches you how to read and write "one-liners": concise statements of useful functionality packed into a single line of code. ||website with free one-liner explainer videos → <http://pythononeliners.com/> ||print version → <https://www.amazon.com/gp/product/B07ZY7XMX8> ||
- **Think Python** by *Allen B. Downey* teaches you how to think like a computer scientist. ||website → <http://greenteapress.com/thinkpython/html/index.html> ||print version → <https://www.amazon.com/Think-Python-Like-Computer-Scientist/dp/1491939362/> ||

You can find many free Python books online. For example, check out [this article with 101 free Python books](#) → <https://blog.finxter.com/free-python-books/>.

Interactive Courses

These sites give you instant feedback on programming problems that you can solve in your browser.

- [A beginner-friendly and free Python tutorial](#) → <https://python.land/python-tutorial> with interactive code examples, explaining the Python language in an easy-to-understand way.
- [A beginner-friendly Python course](#) → <https://programiz.pro/learn/master-python> that teaches to learn to code through bite-size lessons, quizzes and 100+ challenges.
- [CheckiO](#) → <http://www.checkio.org/> is a gamified website containing programming tasks that can be solved in Python 3.
- [Codéex](#) → <https://www.codedex.io/> is a learn to code platform for K-12 and college students.

- [Codecademy](https://www.codecademy.com/search?query=pythonPython) (→ <https://www.codecademy.com/search?query=pythonPython>)
- [Code the blocks](https://codetheblocks.com/) → <https://codetheblocks.com/> combines Python programming with a 3D environment where you "place blocks" and construct structures. It also comes with Python tutorials that teach you how to create progressively elaborate 3D structures.
- [Codevisionz Python](https://codevisionz.com/learn:python:programming/) → <https://codevisionz.com/learn:python:programming/> 10+ hrs of Python learning material - Learn common programming concepts through code examples, quizzes, and challenges
- [Computer Science Circles](http://cscircles.cemc.uwaterloo.ca/) → <http://cscircles.cemc.uwaterloo.ca/> has 30 lessons, 100 exercises, and a message system where you can ask for help. Teachers can use it with their students. It is also available in Dutch, French, German, and Lithuanian.
- [DataCamp Python Tutorial](https://www.datacamp.com/courses/intro-to-python-for-data-science) → <https://www.datacamp.com/courses/intro-to-python-for-data-science> Unlike most other Python tutorials, this 4 hour tutorial by [DataCamp](https://www.datacamp.com/) → <https://www.datacamp.com/> focuses on Python specifically for Data Science. It has 57 interactive exercises and 11 videos.
- [Finxter](https://finxter.com/) → <https://finxter.com/> - How good are your Python skills? Test and Training with >300 hand-picked Python puzzles.
- [HackInScience](https://hackinscience.org/) → <https://hackinscience.org/> - 50+ Python exercises on a free, adless, simple, and open-source platform.
- [How to Think Like a Computer Scientist: Interactive Edition](https://runestone.academy/ns/books/published/t_hinkcsipy/index.html) → https://runestone.academy/ns/books/published/t_hinkcsipy/index.html is an interactive reimagination of Elkner, Downey and Meyer's book with visualizations and audio explanations.
- [LearnPython](https://www.learnpython.org/) → <https://www.learnpython.org/> is an interactive Python tutorial that is suitable for absolute beginners.
- [Learn Python](https://learn-python.adamemory.dev/) → <https://learn-python.adamemory.dev/> - A no install Python course with interactive exercises powered by Pyodide.

Resources for Younger Learners

(This section was previously called "K-12 Oriented", K-12 being a USA-centric term which refers to the primary and secondary educational stages; through level 3 on the UNESCO ISCED education levels list.)

- [Guido van Robot](http://gvr.sourceforge.net/) → <http://gvr.sourceforge.net/> A teaching tool in which students write simple programs using a Python-like language to control a simulated robot. Field-tested at Yorktown High School, the project includes a lesson plan.
- [Python for Kids](http://jasonrbriggs.com/python-for-kids/index.html) → <http://jasonrbriggs.com/python-for-kids/index.html> by Jason R Briggs. Book with sample code and puzzles.
- [PythonTurtle](http://pythonturtle.org/) → <http://pythonturtle.org/> A learning environment for Python suitable for beginners and children, inspired by Logo. Geared mainly towards children, but known to be successful with adults as well.
- [Webucator's self-paced Python 3 course](https://www.webucator.com/self-paced-training/index.cfm#?courseId=P_YT111) → https://www.webucator.com/self-paced-training/index.cfm#?courseId=P_YT111 free for homeschoilers and other students (use HOMESCHOOL as the coupon code when checking out). This course is appropriate for students 13 and up. **From our experience, these students can learn at least as quickly as adults new to programming.**

Tutorials and Websites

- [A Byte of Python](https://python.swaroopch.com/) → <https://python.swaroopch.com/>, by Swaroop C.H., is also an introductory text for people with no previous programming experience.
- [Afternerd](https://www.afternerd.com/) → <https://www.afternerd.com/>, by Karim Elghamrawy, is a Python tutorials blog that is geared towards Python beginners.
- [Ask Python](https://askpython.com/) → <https://askpython.com/> Absolute Beginners Python Tutorial.

- [Hands-on Python Tutorial](http://anh.cs.luc.edu/handsonPythonTutorial/) → <http://anh.cs.luc.edu/handsonPythonTutorial/> Beginners' Python, graphics, and simple client/server introduction, with videos.
- [Learning to Program](http://www.alan-g.me.uk/l2p2) → <http://www.alan-g.me.uk/l2p2> An introduction to programming for those who have never programmed before, by Alan Gauld. It introduces several programming languages but has a strong emphasis on Python. (Python 2 and 3)
- [ItsMyCode](https://itsmycode.com/) → <https://itsmycode.com/> A Python Blog and tutorials built for developers who love coding
- [After Hours Programming Python 3 Tutorial](https://www.afterhoursprogramming.com/tutorial/Python/Overview/) → <https://www.afterhoursprogramming.com/tutorial/Python/Overview/>
- [Letsfindcourse - Python](http://letsfindcourse.com/python) → <http://letsfindcourse.com/python>: Best Python tutorials and courses recommended by experts.
- [The Wikibooks Non-Programmer's Tutorial for Python by Josh Cogliati](http://en.wikibooks.org/wiki/Non-Programmer's_Tutorial_for_Python_3.0) → http://en.wikibooks.org/wiki/Non-Programmer's_Tutorial_for_Python_3.0
- [Online Python Courses](https://www.coursesonline.co.uk/courses/python/) → <https://www.coursesonline.co.uk/courses/python/> Compare online Python courses from learning providers from across the UK
- [Learn Python](https://overiq.com/python/3.4/intro-to-python/) → <https://overiq.com/python/3.4/intro-to-python/> An Introductory yet in-depth tutorial for Python beginners.
- The [Python tips](http://pythontips.com/) → <http://pythontips.com/> blog includes Python tips and tutorials for beginners and professional programmers.
- [Python Tutorial in Python's documentation set](http://docs.python.org/py3k/tutorial/) → <http://docs.python.org/py3k/tutorial/>. It's not written with non-programmers in mind, but it will give you an idea of the language's flavor and style.
- [The Python-Course.eu's extensive tutorial for complete beginners](http://www.python-course.eu/python3_course.php) → http://www.python-course.eu/python3_course.php, with lots of illustrations.
- [Pythonspot Tutorials](https://www.pythonspot.com/) → <https://www.pythonspot.com/> Python tutorials.
- [The Python Guru](http://thepythonguru.com/) → <http://thepythonguru.com/> A beginner-friendly guide for aspiring programmers.
- [CodersLegacy](https://coderslegacy.com/) → <https://coderslegacy.com/> A website + blog geared towards both new and experienced programmers. Mainly focused on teaching Python.
- [Discover Python & Patterns with game programming](https://www.patternsgameprog.com/series/discover-python-and-patterns/) → <https://www.patternsgameprog.com/series/discover-python-and-patterns/> Discover Python by programming video games.
- [QuizCure: A Python Learning Platform](https://www.quizcure.com/topic/python/) → <https://www.quizcure.com/topic/python/> Contains a list of Commonly asked Python Questions and Answers with Examples.

Tutorial Aggregators / lists

- [Gitconnected Python](https://gitconnected.com/learn/python) → <https://gitconnected.com/learn/python> tutorials submitted and ranked by Python developers with the best rising to the top
- [Coursesity - Python](https://coursesity.com/best-tutorials-learn/python) → <https://coursesity.com/best-tutorials-learn/python> - Curated list of the best python courses and tutorials for beginners.
- [Classpert - Python](https://classpert.com/python-programming) → <https://classpert.com/python-programming> - A large collection of free and paid Python online courses, from a wide range of providers.
- [Hackr.io - Python](https://hackr.io/tutorials/learn-python) → <https://hackr.io/tutorials/learn-python>: Programming community-recommended best Python tutorials and courses

Tutorials for Scientific Audiences

These websites are written in support of science courses but are general enough that anyone can learn from them.

- [Beginning Python for Bioinformatics](http://www.onlamp.com/pub/a/python/2002/10/17/biopython.html) → <http://www.onlamp.com/pub/a/python/2002/10/17/biopython.html> by Patrick O'Brien. An introduction to Python aimed at biologists that introduces the [PyCrust](https://wiki.python.org/moin/PyCrust) → <https://wiki.python.org/moin/PyCrust> shell and Python's basic data types.

- [Python for Number Theory](http://illustratedtheoryofnumbers.com/prog.html) → <http://illustratedtheoryofnumbers.com/prog.html> is a series of Python notebooks (for Jupyter) for applications to number theory and cryptography. They assume no prior programming experience and are suitable for someone learning elementary number theory at the same time. They conclude with an introduction to primality testing and cryptography (Diffie-Hellman, RSA).

Apps

- [Programiz App to Learn Python](https://www.programiz.com/learn-python) → <https://www.programiz.com/learn-python> - A beginner-friendly app on Android and iOS to learn Python step by step with an in-built interpreter and quizzes.

Videos

- [Python Programming Tutorials for Beginners](https://youtu.be/uCzFUKWtzgA?list=PLboXykqtm8dy_DNg1NZiS08Dnyj35PWxw) → https://youtu.be/uCzFUKWtzgA?list=PLboXykqtm8dy_DNg1NZiS08Dnyj35PWxw: Installation, IDE, variables, functions, strings, lists, OOP
- The [Young Programmers Podcast](http://youngprogrammers.blogspot.com/search/label/python) → <http://youngprogrammers.blogspot.com/search/label/python> contains video lessons on Python, Pygame, Jython, Scratch, Alice, Java, and Scala (somewhat outdated content!)

Email Academies

- [Finxter Email Computer Science Academy](https://blog.finxter.com/email-academy/) → <https://blog.finxter.com/email-academy/>: 20+ free Python and computer science courses delivered in email video lessons. **Content:** cheat sheets, Python basics, data structures, [NumPy](https://wiki.python.org/moin/NumPy) → <https://wiki.python.org/moin/NumPy>, data science, career advancement, coding productivity, and machine learning.
 - [Thonny, Python IDE for beginners](http://thonny.org/) → <http://thonny.org/>
-

[CategoryPythonInEducation](https://wiki.python.org/moin/CategoryPythonInEducation) → <https://wiki.python.org/moin/CategoryPythonInEducation> [CategoryPythonInEducation](https://wiki.python.org/moin/CategoryPythonInEducation) → <https://wiki.python.org/moin/CategoryPythonInEducation>

BeginnersGuide/NonProgrammersChinese - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/NonProgrammersChinese>

目次

[Beginners-Guide/Programmers](#) → <https://wiki.python.org/moin/BeginnersGuide/Programmers>

目次

/ .

- - *Al Sweigart* “ ” || → <https://automatetheboringstuff.com/> ||
→ <http://www.amazon.com/gp/product/1593275994/> ||
 - *Peter Wentworth.* *Python3* || → <http://openbookproject.net/thinkcs/python/english3e/> ||
→ <http://openbookproject.net/thinkcs/python/english3e/> ||
 - **Python Pygame** *Al Sweigart* *Pygame* || → <http://inventwithpython.com/pygame> || → <http://www.amazon.com/Making-Games-Python-Pygame-Sweigart/dp/1469901730?ie=UTF8&tag=playwithpyth-20&linkCode=as2&camp=1789&creative=9325&creativeASIN=0982106017> ||
 - **Python** *Christian Mayer* : ||
→ <https://pythontoneliners.com/> || → <https://www.amazon.com/gp/product/B07ZY7XMX8> ||
 - **Python** *Allen B. Downey* || → <http://greenteapress.com/thinkpython/html/index.html> || → <https://www.amazon.com/Think-Python-Like-Computer-Scientist/dp/1491939362/> ||
- Python 101 Python → <https://blog.finxter.com/free-python-books/>.

目次

- [Python](#) → <https://python.land/python-tutorial> Python
- [Python](#) → <https://programiz.pro/learn/master-python> 100
- [CheckIO](#) → <http://www.checkio.org/> Python3
- [Codédex](#) → <https://www.codedex.io/> K-12
- [Codecademy](#) → <https://www.codecademy.com/search?query=python>(Python)
- [Code the blocks](#) → <https://codetheblocks.com/> 3D
Python
- [Codevisionz Python](#) → <https://codevisionz.com/learn-python-programming/> 10 +Python -
- [Computer Science Circles](#) → <http://cscircles.cemc.uwaterloo.ca/> 30 100
- [DataCamp Python Tutorial](#) → <https://www.datacamp.com/courses/intro-to-python-for-data-science>

- [Finxter](https://finxter.com/) → <https://finxter.com/> - Python 300 Python
- [HackInScience](https://hackinscience.org/) → <https://hackinscience.org/> - 50 Python
- [How to Think Like a Computer Scientist: Interactive Edition](https://runestone.academy/ns/books/published/thinkcsjava/index.html) → <https://runestone.academy/ns/books/published/thinkcsjava/index.html> is an interactive reimagining of Elkner, Downey and Meyer's book with visualizations and audio explanations.
- [LearnPython](https://www.learnpython.org/) → <https://www.learnpython.org/> Python

၂၀၁၅ ခုနှစ်

- | | | |
|---|---|---------|
| ISCED | K-12 | /UNESCO |
| | | |
| • Guido van Robot → http://gvr.sourceforge.net/
Yorktown High School | Python | |
| • Python for Kids → http://jasonbriggs.com/python-for-kids/index.html | Jason R Briggs | |
| • PythonTurtle → http://pythonturtle.org/ | Python | Logo |
| • Young Coders tutorial → http://www.letslearnpython.com/learn/ | PyCon → https://wiki.python.org/moin/PyCon | 10 |

၂၀၁၆

- [A Byte of Python](https://python.swaroopch.com/) → <https://python.swaroopch.com/> Swaroop C.H.
- [Afternerd](https://www.afternerd.com/) → <https://www.afternerd.com/> Karim Elghamrawy Python Python
- [Ask Python](https://askpython.com/) → <https://askpython.com/> Python
- [Hands-on Python Tutorial](http://anh.cs.luc.edu/handsonPythonTutorial/) → <http://anh.cs.luc.edu/handsonPythonTutorial/> Python /
- [Learning to Program](http://www.alan-g.me.uk/l2p2) → <http://www.alan-g.me.uk/l2p2> Alan Gauld
Python Python2 Python3
- [ItsMyCode](https://itsmycode.com/) → <https://itsmycode.com/> Python
- [After Hours Programming Python 3 Tutorial](https://www.afterhoursprogramming.com/tutorial/Python/Overview/) → <https://www.afterhoursprogramming.com/tutorial/Python/Overview/>
- [Letsfindcourse - Python](http://letsfindcourse.com/python/) → <http://letsfindcourse.com/python/> Python
- [The Wikibooks Non-Programmer's Tutorial for Python by Josh Cogliati](http://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3.0) → http://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3.0
- [Online Python Courses](https://www.coursesonline.co.uk/courses/python/) → <https://www.coursesonline.co.uk/courses/python/> Python
- [Learn Python](https://overiq.com/python/3.4/intro-to-python/) → <https://overiq.com/python/3.4/intro-to-python/> Python
- [The Python tips](http://pythontips.com/) → <http://pythontips.com/> Python
- [Python Tutorial in Python's documentation set](http://docs.python.org/py3k/tutorial/) → <http://docs.python.org/py3k/tutorial/>.
Python
- [Python-Course.eu's](http://www.python-course.eu/python3_course.php) → http://www.python-course.eu/python3_course.php
- [Pythonspot Tutorials](https://www.pythonspot.com/) → <https://www.pythonspot.com/> Python
- [The Python Guru](http://thepythonguru.com/) → <http://thepythonguru.com/>
- [CodersLegacy](https://coderslegacy.com/) → <https://coderslegacy.com/> Python
- [Python](https://www.patternsgameprog.com/series/discover-python-and-patterns/) → <https://www.patternsgameprog.com/series/discover-python-and-patterns/>
- [QuizCure: A Python Learning Platform](https://www.quizcure.com/topic/python/) → <https://www.quizcure.com/topic/python/> Python

၂၁၂၂၂၂၂

- [Gitconnected Python](https://gitconnected.com/learn/python) → <https://gitconnected.com/learn/python> Python
- [Coursesity - Python](https://coursesity.com/best-tutorials-learn/python) → <https://coursesity.com/best-tutorials-learn/python> - Python
- [Classpert - Python](https://classpert.com/python-programming) → <https://classpert.com/python-programming> - / Python
- [Hackr.io - Python](https://hackr.io/tutorials/learn-python) → <https://hackr.io/tutorials/learn-python>: Python

၂၃၂၃၂၃၂

- [Python](http://www.onlamp.com/pub/a/python/2002/10/17/biopython.html) → <http://www.onlamp.com/pub/a/python/2002/10/17/biopython.html> Patrick O'Brien. PyCrust → <https://wiki.python.org/moin/PyCrust> Python
- [Python](http://illustratedtheoryofnumbers.com/prog.html) → <http://illustratedtheoryofnumbers.com/prog.html> Python notebooks Jupyter , (Diffie-Hellman, RSA).

Apps

- [Programiz App to Learn Python](https://www.programiz.com/learn-python) → <https://www.programiz.com/learn-python> - Python iOS

၂၄

- [Python](https://youtu.be/uCzFUKWtzgA?list=PLboXykqtm8dy_DNg1NZiS08Dnyj35PWXw) → https://youtu.be/uCzFUKWtzgA?list=PLboXykqtm8dy_DNg1NZiS08Dnyj35PWXw: IDE OOP
- The [Python](http://youngprogrammers.blogspot.com/search/label/python) → <http://youngprogrammers.blogspot.com/search/label/python> Python Pygame Jython Scratch Alice Java Scala

၂၅၂၅၂၅

- [Finxter](https://blog.finxter.com/email-academy/) → <https://blog.finxter.com/email-academy/>: 20+ Python : Python NumPy → <https://wiki.python.org/moin/NumPy>.

၂၆

- [Thonny](http://thonny.org/), Python IDE → <http://thonny.org/>

[CategoryPythonInEducation](https://wiki.python.org/moin/CategoryPythonInEducation) → <https://wiki.python.org/moin/CategoryPythonInEducation> [CategoryPythonInEducation](https://wiki.python.org/moin/CategoryPythonInEducation) → <https://wiki.python.org/moin/CategoryPythonInEducation>
BeginnersGuide/NonProgrammersChinese (last edited 2022-12-29 03:29:12 by [AtmanAn](mailto:AtmanAn@twinsant@gmail.com) → <mailto:twinsant@gmail.com>)

BeginnersGuide/Overview - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Overview>

Python is a clear and powerful object-oriented programming language, comparable to Perl, Ruby, Scheme, or Java.

Some of Python's notable features:

- Uses an elegant syntax, making the programs you write easier to read.
- Is an easy-to-use language that makes it simple to get your program working. This makes Python ideal for prototype development and other ad-hoc programming tasks, without compromising maintainability.
- Comes with a large standard library that supports many common programming tasks such as connecting to web servers, searching text with regular expressions, reading and modifying files.
- Python's interactive mode makes it easy to test short snippets of code. There's also a bundled development environment called IDLE.
- Is easily extended by adding new modules implemented in a compiled language such as C or C++.
- Can also be embedded into an application to provide a programmable interface.
- Runs anywhere, including Mac OS X → <https://www.python.org/downloads/mac-osx/>, Windows → <https://www.python.org/downloads/windows/>, Linux → <https://docs.python.org/3/using/unix.html>, and Unix → <https://docs.python.org/3/using/unix.html>, with unofficial builds also available for Android → <https://wiki.python.org/moin/Android> and iOS.
- Is free software in two senses. It doesn't cost anything to download or use Python, or to include it in your application. Python can also be freely modified and re-distributed because while the language is copyrighted it's available under [an open-source license](#) → <http://www.python.org/psf/license/>.

Some programming-language features of Python are:

- A variety of basic data types are available: numbers (floating point, complex, and unlimited-length long integers), strings (both ASCII and Unicode), lists, and dictionaries.
- Python supports object-oriented programming with classes and multiple inheritances.
- Code can be grouped into modules and packages.
- The language supports raising and catching exceptions, resulting in cleaner error handling.
- Data types are strongly and dynamically typed. Mixing incompatible types (e.g. attempting to add a string and a number) causes an exception to be raised, so errors are caught sooner.
- Python contains advanced programming features such as generators and list comprehensions.
- Python's automatic memory management frees you from having to manually allocate and free memory in your code.

See the [SimplePrograms](#) → <https://wiki.python.org/moin/SimplePrograms> collection of short programs, gradually increasing in length, which shows off Python's syntax and readability.

Writing Pythonic code is not hard---but you have to get used to the (PEP) code style rules. You can test, check, and improve your code style at online resources such as [Pythonchecker.com](#) → <http://pythonchecker.com/>.

Translations:

- [ChineseOverview](#) → <https://wiki.python.org/moin/Todo>

BeginnersGuide/OverviewChinese - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/OverviewChinese>

Python Python	Perl Ruby Scheme Java
• • • • Python • C C++ • • Mac OS X Windows Linux Unix	Python IDLE iOS
Python	Python Free Python
Python	Python ASCII Unicode
• • Python • • • • generators	list comprehensions Python — PEP Pythonchecker.com

[SimpleProgramsChinese](#) → <https://wiki.python.org/moin/SimpleProgramsChinese>

BeginnersGuide/Programmers - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Programmers>

Please Note

This is a Wiki page. Users with edit rights can edit it. You are, therefore, free to (in fact, encouraged to) add details of material that other Python users will find useful. It is **not** an advertising page and is here to serve the whole Python community. Users who continually edit pages to give their own materials (particularly commercial materials) prominence, or spam the listing with multiple entries which point to resources with only slightly altered material, may subsequently find their editing rights disabled. *You have been warned.* On a cheerier note - there is a constant stream of new and updated information on Python as the language is exploding in popularity. Only enthusiastic volunteers can keep this page current, so if something helps you, feel free to link it here.

The tutorials on this page aim at people with previous experience with other programming languages (C, Perl, Lisp, Visual Basic, etc.). Also of potential interest are such related Beginners Guides as [BeginnersGuide/Overview](https://wiki.python.org/moin/BeginnersGuide/Overview) → <https://wiki.python.org/moin/BeginnersGuide/Overview> and [BeginnersGuide/NonProgrammers](https://wiki.python.org/moin/BeginnersGuide/NonProgrammers) → <https://wiki.python.org/moin/BeginnersGuide/NonProgrammers>, and the tips in [MovingToPythonFromOtherLanguages](https://wiki.python.org/moin/MovingToPythonFromOtherLanguages) → <https://wiki.python.org/moin/MovingToPythonFromOtherLanguages>.

Books, Websites, Tutorials (non-interactive)

Resources

- [A beginner-friendly Python course](https://programiz.pro/learn/master-python) → <https://programiz.pro/learn/master-python> with interactive, bite-size lessons, and over 100 challenges.
- [A beginner-friendly Python tutorial](https://python.land/python-tutorial) → <https://python.land/python-tutorial> that starts with the absolute basics but also covers more advanced stuff like Python software deployment.
- [A Byte of Python](https://python.swaroopch.com/) → <https://python.swaroopch.com/>, by Swaroop C.H. An introductory text for beginners and experienced programmers looking to learn Python.
- [After Hours Programming's Python Introduction](http://www.afterhoursprogramming.com/tutorial/Python/introduction/) → <http://www.afterhoursprogramming.com/tutorial/Python/introduction/> A beginners introduction into Python.
- [Awesome Python](https://awesome-python.com/) → <https://awesome-python.com/> A curated list of awesome Python frameworks, libraries, software and resources.
- [CheckiO interactive learning resource](https://www.checkio.org/) → <https://www.checkio.org/> Creative way to improve Python skills with interesting tasks, it also supports Python 3|2.
- [Classpert - Python](https://classpert.com/python-programming) → <https://classpert.com/python-programming> - A collection of free and paid Python online courses from a wide range of providers.
- [Codedex](https://www.codedex.io.com/) → <https://www.codedex.io.com/> - A learn to code platform for K-12 and college students.
- [CodersLegacy](https://coderslegacy.com/) → <https://coderslegacy.com/> A website + blog geared towards both new and experienced programmers. Mainly focused on teaching Python.
- [Dive Into Python 3](https://diveintopython3.problemsolving.io/) → <https://diveintopython3.problemsolving.io/> by Mark Pilgrim.
- [Elements of Python Style](https://github.com/ambianti/elements-of-python-style) → <https://github.com/ambianti/elements-of-python-style> This document goes beyond PEP8 to cover the core of what the author thinks of as great Python style.
- [Finxter](https://finxter.com/) → <https://finxter.com/> - Solve Python puzzles and test your Python skill level (beginner to grandmaster level).
- [Full Stack Python](https://www.fullstackpython.com/) → <https://www.fullstackpython.com/> Once you know the basics, learn how to build, deploy and operate Python Applications.

- [ItsMyCode](https://itsmycode.com/) → <https://itsmycode.com/> A Python Programming Blog which teaches Python basics and helps to solve various issues which developers face in day to day Programming
- [Python 3 Patterns, Recipes, and Idioms](https://python-3-patterns-idioms-test.readthedocs.io/en/latest/) → <https://python-3-patterns-idioms-test.readthedocs.io/en/latest/> by Bruce Eckel and Friends.
- [Learn Python Step by Step](https://www.techbeamers.com/python-tutorial-step-by-step/) → <https://www.techbeamers.com/python-tutorial-step-by-step/> - Start learning python from the basics to pro-level and attain proficiency.
- [Learn Python OverIQ](https://overiq.com/python/3.4/intro-to-python) → <https://overiq.com/python/3.4/intro-to-python> - An entry-level course to get you started with Python Programming.
- [Learn Python - Tutorial for Beginners](https://www.programiz.com/python-programming) → <https://www.programiz.com/python-programming> A comprehensive Python guide to get started, Python tutorials, and examples for beginners.
- [Free python tips and tutorials](http://freepythontips.wordpress.com/) → <http://freepythontips.wordpress.com/> Python tips and tutorials for beginners and professional programmers.
- [Intro to Python](http://stromberg.dnsalias.org/~dstromberg/Intro-to-Python/) → <http://stromberg.dnsalias.org/~dstromberg/Intro-to-Python/> - A Brief Presentation about Python mainly aimed at experienced programmers. Might be nice as a first pass over the language.
- [Learn Python in 10 minutes](https://www.stavros.io/tutorials/python/) → <https://www.stavros.io/tutorials/python/>
- [Python Course](http://www.python-course.eu/) → <http://www.python-course.eu/> - This online Python course is aiming at beginners and with advanced topics at experienced programmers as well.
- [Python Koans](https://github.com/gregmalcolm/python_koans) → https://github.com/gregmalcolm/python_koans Learn Python through TDD
- [Python Programming for Beginners](http://www.linuxjournal.com/lj-issues/issue73/3946.html) → <http://www.linuxjournal.com/lj-issues/issue73/3946.html> A short introduction to writing command-line applications in Python by Jacek Artymiak.
- [PythonSpeed.com](https://pythonspeed.com/performance/) → <https://pythonspeed.com/performance/> Great resource with insightful ways to speed up your Python code
- [Python Essential Reference](https://www.dabeaz.com/per.html) → <https://www.dabeaz.com/per.html> (book) If you want a highly compressed K&R-style 'just the facts' overview, David Beazley's "Python Essential Reference" covers practically all of the language in about a hundred pages. A version that covers Python 3.7 is in progress.
- [Resources for Learning Python](http://codecondo.com/10-ways-to-learn-python/) → <http://codecondo.com/10-ways-to-learn-python/> 10 of the most popular / recommended platforms in the World when it comes to learning Python, either as a complete beginner or someone who knows their way around.
- [Python Tutorial](http://docs.python.org/tut/) → <http://docs.python.org/tut/> This tutorial is part of Python's documentation set and is updated with each new release.
- [Wikiversity:Python](http://en.wikiversity.org/wiki/Topic:Python) → <http://en.wikiversity.org/wiki/Topic:Python> The Wiki(anything) information about Python.
- [Python Programming Tutorials](https://pythonspot.com/) → <https://pythonspot.com/> Python programming tutorials.
- [Python Tutorials](http://thepythonguru.com/getting-started-with-python/) → <http://thepythonguru.com/getting-started-with-python/> Python in plain English.
- [Learn Python - Programming Made Easy](http://www.trytoprogram.com/python-programming/) → <http://www.trytoprogram.com/python-programming/> Simplified tutorials for beginners (Learn with relevant examples).
- [Pandas Cookbook](https://tutswiki.com/pandas-cookbook/) → <https://tutswiki.com/pandas-cookbook/> A newbie friendly introduction to pandas with real-life examples.
- [Ultimate Python study guide](https://github.com/huangsam/ultimate-python) → <https://github.com/huangsam/ultimate-python> Ultimate Python study guide for newcomers and professionals alike.
- [Learn coding with Python notebooks](https://www.nbshare.io/) → <https://www.nbshare.io/> A place where users can learn lot about Python coding with Python notebooks.
- [Learn Python Programming](https://www.scaler.com/topics/python/) → <https://www.scaler.com/topics/python/> Easy to understand Python tutorial explained with examples for beginners and professionals alike.
- [Computer Science Circles](http://cscircles.cemc.uwaterloo.ca/) → <http://cscircles.cemc.uwaterloo.ca/>
- [HackInScience: free and open-source Python training website](https://hackinscience.org/) → <https://hackinscience.org/>
- [Learn Python](https://learn-python.adamemory.dev/) → <https://learn-python.adamemory.dev/> - A no install Python course with interactive exercises powered by Pyodide.

- [Python Editor](https://python-editor.adamemery.dev/) → <https://python-editor.adamemery.dev/> - A web app for writing and running basic Python scripts
- [Python visualizer tool](http://people.csail.mit.edu/pgbovine/python/tutor.html) → <http://people.csail.mit.edu/pgbovine/python/tutor.html>
- [Thonny, Python IDE for beginners. Has intuitive features for program runtime visualization](http://thonny.org/) → <http://thonny.org/>
- [PyFlo](https://pyflo.net/) → <https://pyflo.net/> - A free, interactive guide to becoming a Python Programmer

Python Video Tutorials

- <https://www.youtube.com/watch?v=rfscVS0vtbw> - Python Beginners Course by [FreeCodeCamp](#) → <https://wiki.python.org/moin/FreeCodeCamp> (4 hours)
- <https://www.youtube.com/watch?v=HGOBQPFzWKO> - Python Intermediate Course by [FreeCodeCamp](#) → <https://wiki.python.org/moin/FreeCodeCamp> (6 hours)
- <https://www.webucator.com/django-training/course/writing-your-first-django-app/> - A free Django course with videos based on the Django Software Foundations official tutorial.
- [Crawl the Web With Python](https://code.tutsplus.com/courses/crawl-the-web-with-python) → <https://code.tutsplus.com/courses/crawl-the-web-with-python> - learn to build a web crawler and scraper (paid/commercial).
- [Django Basics](https://overiq.com/django/1.10/intro-to-django/) → <https://overiq.com/django/1.10/intro-to-django/> - An introductory course to learn basics of Django framework in great detail.
- [MIT's Introduction to Computer Science and Programming in Python](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-videos/) → <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-videos/>
- [Khan Academy computer science](https://www.khanacademy.org/computing/computer-science) → <https://www.khanacademy.org/computing/computer-science> playlist teaches Python.
- [Python Exception Handling for beginners](https://youtu.be/HJSLyzm4j6Y) → <https://youtu.be/HJSLyzm4j6Y> - Exception handling with Python.
- [Python Lists and Object Tutorial for Beginners](https://www.youtube.com/watch?v=DFLD3JjsvJo&list=PLboXykqtm8dy_DNg1NZIS08Dnyj35PWXw) → https://www.youtube.com/watch?v=DFLD3JjsvJo&list=PLboXykqtm8dy_DNg1NZIS08Dnyj35PWXw - Sorting Objects with Python.
- [Python OOP Tutorial for Beginners](https://youtu.be/RZF17FfRllo) → <https://youtu.be/RZF17FfRllo> - Getting started with OOP programming with Python.
- [Python Screencasts](https://www.youtube.com/playlist?list=PLlgoYPTU6ljCEggReCMF0m0760QTot9Qz) → <https://www.youtube.com/playlist?list=PLlgoYPTU6ljCEggReCMF0m0760QTot9Qz> (36 videos)

Python video tutorial (commercial/paid)

- [Getting Started With Django](https://code.tutsplus.com/courses/getting-started-with-django) → <https://code.tutsplus.com/courses/getting-started-with-django> - learn the Django back-end framework from scratch (paid/commercial)
- [Build a News Aggregator With Django](https://code.tutsplus.com/courses/build-a-news-aggregator-with-django) → <https://code.tutsplus.com/courses/build-a-news-aggregator-with-django> - learn advanced Django skills with a hands-on project (paid/commercial)
- [Data Handling With Python](https://code.tutsplus.com/courses/data-handling-with-python) → <https://code.tutsplus.com/courses/data-handling-with-python> - learn the basics of handling data in the Python language (paid/commercial).

Free Python Courses

- [Free Python 3 email course](https://blog.finxters.com/subscribe) → <https://blog.finxters.com/subscribe> (almost daily Python lesson + cheat sheets, email required)

Other Python Resource Aggregators

- [Learn Python - Best Python Tutorials and Courses](https://hackr.io/tutorials/learn-python) → <https://hackr.io/tutorials/learn-python> Python tutorials & courses recommended by the programming community.
 - [Learn Python - Best Python Courses](https://gitconnected.com/learn/python) → <https://gitconnected.com/learn/python> Python tutorials submitted and ranked by Python developers with the best rising to the top
 - [Paid Python 3 course](https://www.codecademy.com/learn/learn-python-3) → <https://www.codecademy.com/learn/learn-python-3> (almost daily Python lesson + cheat sheets)
-

[CategoryPythonWebsite](https://wiki.python.org/moin/CategoryPythonWebsite) → <https://wiki.python.org/moin/CategoryPythonWebsite> [CategoryCategory](https://wiki.python.org/moin/CategoryCategory) → <https://wiki.python.org/moin/CategoryCategory>

attachment:Cpp2Python.pdf of BeginnersGuide/Programmers - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Programmers?action=AttachFile&do=view&target=Cpp2Python.pdf>.

Download → <https://wiki.python.org/moin/BeginnersGuide/Programmers?action=AttachFile&do=get&target=Cpp2Python.pdf>

Attached Files

To refer to attachments on a page, use **attachment:filename**, as shown below in the list of files. Do NOT use the URL of the

[get]

link, since this is subject to change and can break easily.

- [get → <https://wiki.python.org/moin/BeginnersGuide/Programmers?action=AttachFile&do=get&target=Cpp2Python.pdf> |
view → <https://wiki.python.org/moin/BeginnersGuide/Programmers?action=AttachFile&do=view&target=Cpp2Python.pdf>] (2007-03-19 02:52:30, 37.0 KB) [[attachment:Cpp2Python.pdf]]

All files | Selected Files: delete move to page copy to page

You are not allowed to attach a file to this page.

BeginnersGuide/Programmers/SimpleExamples - Python Wiki

Source: <https://wiki.python.org/moin/BeginnersGuide/Programmers/SimpleExamples>

Here are some samples to help get a better idea of Python's syntax:

Hello World (the traditional first program)

String formatting

```
name = 'Monty'
print('Hello, %s' % name)
print('Hello, {}'.format(name))
```

Defining a function

```
def add_one(x):
    return x + 1
```

Testing variable equality

```
x = 1
y = 2
print 'x is equal to y: %s' % (x == y)
z = 1
print 'x is equal to z: %s' % (x == z)
names = ['Donald', 'Jake', 'Phil']
words = ['Random', 'Words', 'Dogs']
if names == words:
    print 'Names list is equal to words'
else:
    print "Names list isn't equal to words"
new_names = ['Donald', 'Jake', 'Phil']
print 'New names list is equal to names: %s' % (new_names == names)
```

Defining a class with two methods

```
class Talker(object):
    def greet(self, name):
        print 'Hello, %s!' % name
    def farewell(self, name):
        print 'Farewell, %s!' % name
```

Defining a list

```
dynamic_languages = ['Python', 'Ruby', 'Groovy']
dynamic_languages.append('Lisp')
```

Defining a dictionary

```
numbered_words = dict()
numbered_words[2] = 'world'
numbered_words[1] = 'Hello'
numbered_words[3] = '!'
```

Defining a while loop

```
while True:
    if value == wanted_value:
        break
    else:
        pass
```

Defining multiline strings

```
string = '''This is a string with embedded newlines.
Also known as a tripled-quoted string.
    Whitespace at the beginning of lines is included,
so the above line is indented but the others are not.
'''
```

Splitting a long string over several lines of source code

```
string = ('This is a single long, long string'
          ' written over many lines for convenience'
          ' using implicit concatenation to join each'
          ' piece into a single string without extra'
          ' newlines (unless you add them yourself).')
```

Defining a for loop

```
for x in xrange(1, 4):
    print ('Hello, new Python user!'
          'This is time number %d') % x
```

List comprehension

```
l = [x**2 for x in range(4)]  
print(l)
```

Set comprehension with condition

```
squares = {x**2 for x in [0,2,4] if x < 4}  
print(squares)
```

[CategoryDocumentation](https://wiki.python.org/moin/CategoryDocumentation) → <https://wiki.python.org/moin/CategoryDocumentation>

BeginnersGuide/Programmers/SimpleExamples (last edited 2020-03-20 17:37:52 by [MarcAndreLemburg](https://wiki.python.org/moin/MarcAndreLemburg) → <http://wiki.python.org/moin/MarcAndreLemburg>)

Be on the Right Side of Change

Source: <https://finxter.com>

Q: What is Finxter?

Finxter is a learning platform dedicated to helping you enhance your programming skills, particularly in Python, and prompting skills, particularly in OpenAI's API.

We offer a range of resources including [free email academy](#) → <https://blog.finxter.com/email-academy/>, [cheat sheets](#) → <https://blog.finxter.com/subscribe/>, [books](#) → <https://blog.finxter.com/finxter-books/>, and [courses](#) → <https://academy.finxter.com/> to help you advance your career and become a proficient coder.

Most importantly, our main goal is to help you stay on the right side of change by leveraging AI and large language models to help you create more value with less effort. 

Q: How can I get started with learning Python at Finxter?

It's simple! Just [download our free Python and OpenAI Prompting cheat sheets and subscribe to our free email academy](#) → <https://blog.finxter.com/subscribe/>. These resources will provide you with valuable insights and knowledge to kickstart your tech education and Python learning journey.

Q: Are all the resources on Finxter free?

99% of all Finxter material is free. We provide free Python cheat sheets, books, and a free email academy to all our users. Additionally, we offer premium [coding books](#) → <https://blog.finxter.com/finxter-books/> and [premium courses](#) → <https://academy.finxter.com/> for more advanced learning.

Our flagship program is the [Finxter Premium Membership](#) → <https://blog.finxter.com/finxter-premium-membership/> that is eventually purchased by roughly 2% of our students. 98% of our students keep learning with our free material, so we recommend you start there.

Q: Can Finxter help me find freelancing opportunities?

Absolutely! Finxter guides you on how to [start making money as a freelancer](#) → <https://academy.finxter.com/university/finxter-freelancer-course/>, allowing you to work from anywhere and build a high-income skill as a software developer.

Q: What kind of courses does Finxter offer?

Finxter offers over 40 courses covering a wide range of topics including OpenAI Prompt Engineering, Python, Machine Learning (ML), Artificial Intelligence (AI), Data Science (DS), Cryptocurrency, and freelancing. You can scroll through our course library [here](#) → <https://academy.finxter.com/>. Finxter Premium members can access all courses for free!

Q: How can Finxter help me become a Grandmaster of Code?

By solving [Python puzzles on Finxter](#) → <https://app.finxter.com/learn/computer/science/>, you not only have fun but also learn and practice coding, helping you to eventually become a Grandmaster of Code.

Q: What do other students say about Finxter?

Our students love the content and resources we provide. We receive hundreds of [positive testimonials](https://blog.finxter.com/what-our-users-say/) → <https://blog.finxter.com/what-our-users-say/> every week. Our growth comes exclusively from word of mouth, we spend zero dollars on advertising! Most of our users find our Python cheat sheets excellent for quick direction and daily motivation, and they enjoy solving puzzles and reading our books to enhance their Python skills.

Q: How can I stay up to date with the latest coding tutorials?

By subscribing to Finxter's [free email academy](https://blog.finxter.com/email-academy/) → <https://blog.finxter.com/email-academy/>, you will receive regular updates on the latest coding tutorials, ensuring you are always up to date with the most current information and best practices in programming.

Q: How do I stay on the right side of change - is my job safe?

Keep learning and applying new technologies. We share new technological disruptions like ChatGPT, Llama 2, Bitcoin almost daily in our [free email newsletter](https://blog.finxter.com/email-academy/) → <https://blog.finxter.com/email-academy/>.

Also, to stay on the right side of change, you need to become an owner of scarce and desirable assets (e.g., "*the infinity of AI meets the scarcity of Bitcoin*"), so we encourage the creation of and investment into business systems and crypto assets like Bitcoin. We're not financial advisors though but coders and tech enthusiasts like you.

Your job is not safe, none is. The world will look vastly different in 10 years. The only constant is change. The Finxter mission is to help you stay on the right side of change.  

[Join the Finxter Movement for Free](https://blog.finxter.com/email-academy/)  → <https://blog.finxter.com/email-academy/>

CheckiO - coding games and programming challenges for beginner and advanced

Source: <http://www.checkio.org>



island



island

CheckiO

[Python → https://py.checkio.org/](https://py.checkio.org/) [TypeScript → https://js.checkio.org/](https://js.checkio.org/)

Coding games for beginners and advanced programmers

where you can improve your coding skills by solving engaging challenges and fun tasks using [Python → https://py.checkio.org/](https://py.checkio.org/) and [TypeScript → https://js.checkio.org/](https://js.checkio.org/)

Teachers all over the world use CheckiO as an extra-tool during their courses so that students could practice their skills when learning new material

Partners:



The Python IDE for Professional Developers. - online programming games



WebStorm is a powerful JavaScript IDE that makes you productive. Try free. - websites to practice coding

For Teachers

Partners:



The Python IDE for Professional Developers. - online programming games



WebStorm is a powerful JavaScript IDE that makes you productive. Try free. - websites to practice coding

Python Quiz - After Hours Programming

Source: <http://www.afterhoursprogramming.com/tutorial/Python/Python-Quiz/>

The Standard Python Quiz

It's Python quiz time. You have finally finished our current tutorials for Python. No worries, more tutorials are still being developed, but for now, we have to rest. I would first like to congratulate you on working through those tutorials. Your dedication to learning a new language like Python is impressive, but you might be wondering exactly what you learned or how well you paid attention. That is precisely why we have crafted a quiz. The questions that compose this quiz are designed to test if you understand the basics of Python, but it will not quiz advanced topics. However, this quiz is probably the beginning of a series of quizzes for each tutorial category.

[Start the Python Quiz](https://www.afterhoursprogramming.com/tests/python/) → <https://www.afterhoursprogramming.com/tests/python/>

Python Tests and Exams

Currently, we only have a standard quiz that is not intended to be a professional test. There is a possibility to integrate a testing system into After Hours Programming, but we are not actively working towards that goal at the moment. For now, we have created a [simple quiz to check your understanding of Python](#) → <https://www.afterhoursprogramming.com/tests/python/>. Good luck and I hope you enjoyed our Python tutorial. Please contact us if you would like more Python tutorials.

Link/cite this page

If you use any of the content on this page in your own work, please use the code below to cite this page as the source of the content.

- Python Quiz
- Stewart, Suzy. "Python Quiz". *After Hours Programming*. Accessed on February 16, 2024. <https://www.afterhoursprogramming.com/tutorial/python/python-quiz/>.
- Stewart, Suzy. "Python Quiz". *After Hours Programming*, <https://www.afterhoursprogramming.com/tutorial/python/python-quiz/>. Accessed 16 February, 2024.
- Stewart, Suzy. Python Quiz. After Hours Programming. Retrieved from <https://www.afterhoursprogramming.com/tutorial/python/python-quiz/>.

Source: <http://www.pythongui.com>

Python Tests / Quizes

Source: <https://pythonspot.com/python-tests-quizes/>

Python hosting: [Host, run, and code Python in the cloud!](https://www.pythonanywhere.com/?affiliate_id=00535ced) → https://www.pythonanywhere.com/?affiliate_id=00535ced
If you finished the Python beginner series, you could do this quiz to test your understanding of Python. Don't worry, more tutorials are being created on a weekly and sometimes daily basis.

We hope you like our tutorials and enjoy the quizzes!

Related Course: [Python Programming Bootcamp: Go from zero to hero](https://gum.co/dcsp) → <https://gum.co/dcsp>

Python Beginner Quiz

A quiz to understand your understanding of Python. This quiz only covers parts of the beginners series and not all of the topics covered on this site including variables, functions, classes, objects and many more. We recommend you do the beginner series first if you have not done so yet.

The quiz consists of 20 questions.

Python Beginner Quiz 2

Another Python quiz to test your knowledge of Python for the beginners series. The quiz includes some general questions on the Python programming language.

You may be surprised to find some of the answers.

The test Consists of 10 questions.

Python Syntax Checker PEP8

Source: <http://pythonchecker.com/>

PythonChecker Makes Your Code Great Again

PyPI · The Python Package Index

Source: <http://pypi.python.org/pypi>

The Python Package Index (PyPI) is a repository of software for the Python programming language.

PyPI helps you find and install software developed and shared by the Python community. [Learn about installing packages → https://packaging.python.org/tutorials/installing-packages/](https://packaging.python.org/tutorials/installing-packages/).

Package authors use PyPI to distribute their software. [Learn how to package your Python code for PyPI → https://packaging.python.org/tutorials/packaging/projects/](https://packaging.python.org/tutorials/packaging/projects/).

Welcome to Python.org

Source: <http://www.python.org/search/>

Notice: While JavaScript is not essential for this website, your interaction with the content will be limited. Please turn JavaScript on for the full experience.

[Skip to content](#)

- [Python](#) → <http://www.python.org/>
- [PSF](#) → <https://www.python.org/psf/>
- [Docs](#) → <https://docs.python.org/>
- [PyPI](#) → <https://pypi.org/>
- [Jobs](#) → <http://www.python.org/jobs/>
- [Community](#) → <http://www.python.org/community-landing/>



[Donate](#) → <https://psfmember.org/civicrm/contribute/transact?reset=1&id=2>

[≡ Menu](#)

Search This Site

- [Socialize](#)
 - [LinkedIn](#) → <https://www.linkedin.com/company/python-software-foundation/>
 - [Mastodon](#) → <https://fosstodon.org/@ThePSF>
 - [Chat on IRC](#) → <http://www.python.org/community/irc/>
 - [Twitter](#) → <https://twitter.com/ThePSF>
- [About](#) → <http://www.python.org/about/>
- [Downloads](#) → <http://www.python.org/downloads/>
- [Documentation](#) → <http://www.python.org/doc/>
- [Community](#) → <http://www.python.org/community/>
- [Success Stories](#) → <http://www.python.org/success-stories/>
- [News](#) → <http://www.python.org/blogs/>
- [Events](#) → <http://www.python.org/events/>

Search Python.org

Welcome to Python.org

Source: <http://www.python.org/doc/faq/>

Notice: While JavaScript is not essential for this website, your interaction with the content will be limited. Please turn JavaScript on for the full experience.

Error 404: File not Found

We couldn't find what you were looking for. This error has been reported and we will look into it shortly. For now,

- Try using the search box.
- Python.org went through a redesign and you may have followed a broken link. Sorry for that, see if [the same page on the legacy website → http://legacy.python.org/doc/faq/](#) works.
- Lost in an exotic function call? Read [the docs → http://www.python.org/doc/](#). Looking for a package? Check the [Package Index → https://pypi.org/](#). Need a specific version of Python? The [downloads section → http://www.python.org/downloads/](#) has it.

Welcome to Python.org

Source: <http://www.python.org/search/>

Notice: While JavaScript is not essential for this website, your interaction with the content will be limited. Please turn JavaScript on for the full experience.

[Skip to content](#)

- [Python](#) → <http://www.python.org/>
- [PSF](#) → <https://www.python.org/psf/>
- [Docs](#) → <https://docs.python.org/>
- [PyPI](#) → <https://pypi.org/>
- [Jobs](#) → <http://www.python.org/jobs/>
- [Community](#) → <http://www.python.org/community-landing/>



[Donate](#) → <https://psfmember.org/civicrm/contribute/transact?reset=1&id=2>

[≡ Menu](#)

Search This Site

- [Socialize](#)
 - [LinkedIn](#) → <https://www.linkedin.com/company/python-software-foundation/>
 - [Mastodon](#) → <https://fosstodon.org/@ThePSF>
 - [Chat on IRC](#) → <http://www.python.org/community/irc/>
 - [Twitter](#) → <https://twitter.com/ThePSF>
- [About](#) → <http://www.python.org/about/>
- [Downloads](#) → <http://www.python.org/downloads/>
- [Documentation](#) → <http://www.python.org/doc/>
- [Community](#) → <http://www.python.org/community/>
- [Success Stories](#) → <http://www.python.org/success-stories/>
- [News](#) → <http://www.python.org/blogs/>
- [Events](#) → <http://www.python.org/events/>

Search Python.org

Source: <http://pydoc.net/>

Google

Source: <http://www.google.com>

Search Images → <https://www.google.com/imghp?hl=en&tab=wi> Maps → <http://maps.google.com/maps?hl=en&tab=wi> Play Web History → <http://www.google.com/history/optout?hl=en> | Settings → <http://www.google.com/preferences?hl=en> | .

[Collection] 11 Python Cheat Sheets Every Python Coder Must Own – Be on the Right Side of Change

By Chris

Source: <https://blog.finxter.com/collection-5-cheat-sheets-every-python-coder-must-own/>

Don't have much time to learn Python? Cheat sheets to the rescue!

Cheat sheets are among the most efficient ways to acquire knowledge. A great cheat sheet focuses on the key learning material and skips the rest. If you read over them every day, you'll quickly learn all the basics you need to know to master Python.

In this article, we'll share the top 11 Python cheat sheets with you. Download them, print them, put them on your wall and watch your Python skills grow!

If you just want to download your free PDF Python cheat sheet and be done with it, click the following PDF symbol. It's a direct PDF download link of the great Python 3 cheat sheet by the French Python star Laurent Pointal:



5 Python Cheat Sheets Every Python Coder Must Own

[ATTENTION] While this free cheat sheet is the best single cheat sheet you'll find in the world ? (maybe even in the universe), it's not enough to become a great coder. I LOVE cheat sheets ? and believe in the power of the 80/20 principle. So, I've developed a method for training basic Python skills by sending you regular Python cheat sheets. → <https://blog.finxter.com/subscribe/>

What's better than one Python cheat sheet? Eleven Python cheat sheets! ? Grab them, print them, and stick them all to your wall. Then watch yourself to become a Python master. Register for my free Python cheat sheet email course → <https://blog.finxter.com/subscribe/> here:

The rest of the article gives you an overview of the best Python cheat sheets in the world!

Ultra short summary:

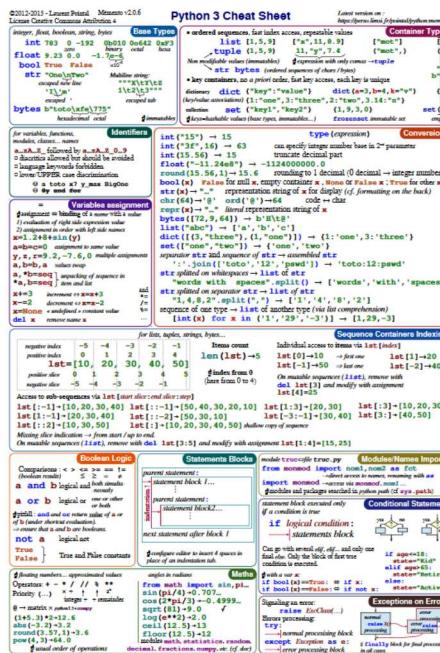
- You want the most dense cheat sheet in Python? Check out this cheat sheet! → https://perso.limsi.fr/pointal/_media/python:cours:memento/python3-english.pdf
- You want the most comprehensive cheat sheet course? Check out this cheat sheet course! → <https://blog.finxter.com/subscribe/>

Have you downloaded the cheat sheet of Laurent Pointal via the PDF button and registered for my free Python course? Good, then move on into the wonderful world of condensed programming knowledge! ??

All given links open in a new browser tab!

So let's dive into the best Python cheat sheets recommended by us.

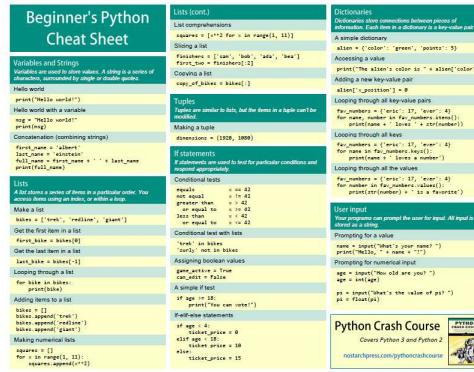
(1) Python 3 Cheat Sheet → https://perso.limsi.fr/pointal/_media/python:cours:mementopython3-english.pdf



This is the best single cheat sheet. It uses every inch of the page to deliver value and covers everything you need to know to go from beginner to intermediate. Topics covered include container types, conversions, modules, maths, conditionals, and formatting, to name a few. A highly recommended 2-page sheet!

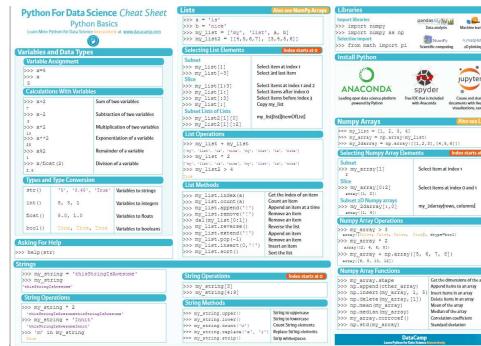
👉 Recommended: [Python Beginner Cheat Sheet: 19 Keywords Every Coder Must Know](https://blog.finxter.com/python-cheat-sheet/) → <https://blog.finxter.com/python-cheat-sheet/>

(2) Python Beginner Cheat Sheet → https://github.com/ehmatthes/pcc/releases/download/v1.0.0/beginners_python_cheat_sheet_pcc_all.pdf



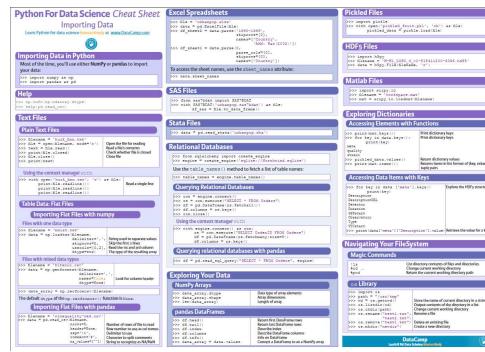
Some might think this cheat sheet is a bit lengthy. At 26 pages, it is the most comprehensive cheat sheets out there. It explains variables, data structures, exceptions, and classes – to name just a few. If you want the most thorough cheat sheet, pick this one.

(3) Python for Data Science → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PythonForDataScience.pdf



Some of the most popular things to do with Python are [Data Science and Machine Learning](https://blog.finxter.com/artificial-intelligence-machine-learning-deep-learning-and-data-science-whats-the-difference/) → <https://blog.finxter.com/artificial-intelligence-machine-learning-deep-learning-and-data-science-whats-the-difference/>. In this cheat sheet, you will learn the basics of Python and the most important scientific library: [NumPy](https://blog.finxter.com/numpy-tutorial/) → <https://blog.finxter.com/numpy-tutorial/> (Numerical Python). You'll learn the basics plus the most important NumPy functions. If you are using Python for Data Science, download this cheat sheet.

(4) Python for Data Science (Importing Data) → https://s3.amazonaws.com/assets.datacamp.co/m/blog_assets/Cheat+Sheets/Importing_Data_Python_Cheat_Sheet.pdf



This Python data science cheat sheet from DataCamp is all about getting data into your code. Think about it: importing data is one of the most important tasks when working with data. Increasing your skills in this area will make you a better data scientist—and a better coder overall!

(5) Python Cheatography Cheat Sheet – <https://www.cheatography.com/davechild/cheat-sheets/python/pdf/>

This cheat sheet is for more advanced learners. It covers class, string and list methods as well as system calls from the `sys` module. Once you're comfortable defining basic classes and command line interfaces (CLIs), get this cheat sheet. It will take you to another level.

(6) The Ultimative Python Cheat Sheet Course (5x Email Series) → <https://blog.finxter.com/subscribe/>

Keyword	Description	Code example
<code>False, True</code>	Boolean data types	<code>False == (1 > 0), True == (2 > 1)</code>
<code>None</code>	Empty value constant	<code>def f(): x = 2 f() == None # True</code>
<code>and, or, not</code>	Logical operators: <code>(x and y)</code> = both <code>x</code> and <code>y</code> must be True <code>(x or y)</code> = either <code>x</code> or <code>y</code> must be True <code>(not x)</code> = <code>x</code> must be False	<code>x, y = True, False (x and y) == True (x or y) == True (not x) == True</code>
<code>break</code>	Ends loop prematurely	<code>while True: break # no infinite loop print("Hello world")</code>
<code>continue</code>	Finishes current loop iteration	<code>while True: continue print("#") # dead code</code>
<code>class</code>	Defines a new class → a real-world concept (object oriented programming)	<code>class Beer: x = 1.0 # litre def drink(self): self.x -= 0.8 b = Beer() # creates class with constructor b.drink() # beer empty: b.x == 0</code>
<code>def</code>	Defines a new function or class method. For latter, first parameter ("self") points to the class object. When calling class method, first parameter is implicit.	<code>x = int(input("Your value: ")) if x > 3: print("Big") elif x < 3: print("Medium") else: print("Small")</code>
<code>for, while</code>	# For loop declaration <code>for i in [0,1,2]: print(i)</code>	<code># While loop - same semantics x = 0 while x < 3: print(x) x = x + 1</code>
<code>in</code>	Checks whether element is in sequence	<code>42 in [2, 39, 42] # True</code>
<code>is</code>	Checks whether both elements point to the same object	<code>y = x = 3 y is x # True [3] is [3] # False</code>
<code>lambda</code>	Function with no name (anonymous function)	<code>(lambda x: x + 3)(3) # returns 6</code>
<code>return</code>	Result of a function	<code>def incrementor(x): return x + 1 incrementor(4) # returns 5</code>

finxter

Python Cheat Sheet - Basic Data Types

"A puzzle a day to learn, code, and play" → Visit finxter.com

	Description	Example
Boolean	The Boolean data type is a truth value, either <code>True</code> or <code>False</code> .	<pre># 1. Boolean Operations print(x > y) # True print(x and not y) # True print(not(x and y) or x) # True # 2. If condition evaluates to False if None or 0 or 0.0 or [] or {} or set(): print("Dead code") # Not reached</pre>
	The Boolean operators used for <code>python</code> :	
	<code>x > y</code> if <code>x</code> is False, then <code>y</code> , else <code>x</code> <code>x and y</code> if <code>x</code> is False, then <code>x</code> , else <code>y</code> <code>x or y</code> if <code>"x</code> is False, then <code>y</code> , else <code>x"</code>	
	These comparison operators evaluate to <code>True</code> :	
	<code>1 < 2 and 0 < 1 and 3 < 2 and 2 >= 2</code> and <code>1 == 1 and 1 != 0</code> <code>&&</code> <code>True</code>	
Integer, Float	An integer is a positive or negative number without floating point (e.g. 3). A float is a positive or negative number with floating point precision (e.g. <code>3.141592653589793</code>).	<pre># 3. Arithmetic Operations x = 1.5 y = 2.5 print(x + y) # 4.0 print(x * y) # 3.75 print(x - y) # -1.0 print(x // y) # 1 print(x % y) # 1 print(x ** y) # 16.41005964949558</pre>
	The <code>//</code> operator performs integer division.	
	The result is an integer value that is rounded towards the smaller integer number (<code>(0.5 // 2 == 0)</code>).	
String	Python Strings are sequences of characters.	<pre># 4. Indexing and Slicing print("Hello world"[0]) # 'H' print("Hello world"[1:5]) # 'Hello' print("Hello world"[-1]) # 'd' print("Hello world"[-5:-1]) # 'old' x = "Hello" print(x[0:1]) # 'H' creates string array of words print(x[0:3] + " " + x[1:3] + " " + x[2:3] + " ") # 'Hello world' spaces print(x[0:3] + " " + x[1:3] + " " + x[2:3] + " ") # 'Hello world' spaces</pre>
	The four main ways to create strings are the following:	
	1. Single quotes 'Yes'	
	2. Double quotes "Yes"	
	3. Triple quotes (multi-line) ''' Me can'''	
	4. String method <code>str() == "S" # True</code>	
	<code>isinstance("Hu", str) == "MahaTma"</code>	
	These are whitespace characters in strings.	<pre># 5. Most Important Methods x = " this is lisy " print(x.lstrip()) # Remove Whitespace: 'this is lisy' print(x.lower()) # 'this is lisy' print(x.upper()) # 'THIS IS Lisy' print(x.replace(' ', '_')) # 'this_is_lisy' print(x.startswith('t')) # True print(x.endswith('ny')) # True print(x.replace(' ', '_').replace('i', 'I')) # 'tHIS_IS_lNy' print(x.replace(' ', '_').replace('i', 'I')) # 'tHIS_IS_lNy' print(len("Rumpleteazer")) # String Length: 15 print("ear" in "Karthik") # Contains: True</pre>

finxter

Python Cheat Sheet - Complex Data Types

"A puzzle a day to learn, code, and play" → Visit finxter.com

	Description	Example
List	A container data type that stores a sequence of elements. Unlike strings, lists are mutable; modification possible.	<code>1 + [1, 2, 3] print([1]*1) # [1]</code>
Adding elements	Adds elements to a list with (i) append, (ii) insert, or (iii) list concatenation. The append operation is very fast.	<code>[1, 2, 3].append(4) # [1, 2, 3, 4] [1, 2, 3].insert(2, 4) # [1, 2, 2, 4] [1, 2, 3] + [4] # [1, 2, 2, 4]</code>
Removal	Removing an element can be slower.	<code>[1, 2, 2, 4].remove(2) # [1, 2, 4]</code>
Reversing	This reverses the order of list elements.	<code>[1, 2, 3].reverse() # [3, 2, 1]</code>
Sorting	Sorts a list. The computational complexity of sorting is $O(n \log n)$ for n list elements.	<code>[1, 2, 4, 2].sort() # [1, 2, 4]</code>
Indexing	Finds the first occurrence of an element in the list & returns its index. Can be slow as the whole list is traversed.	<code>[1, 2, 4, 2].index(2) # index of element 4 is "8" [1, 2, 4].index(2, 1) # index of element 2 after pos 1 is "1"</code>
Stack	Python lists can be used intuitively as stack via the two operations append() and pop().	<code>stack = [3] stack.append(4) # [1, 2] stack.pop() # [1] stack.pop() # []</code>
Set	A set is an unordered collection of elements. Each can exist only once.	<code>basket = {'apple', 'orange', 'banana', 'orange'} same = set(['apple', 'eggs', 'banana', 'orange']) different = {'apple', 'banana', '89', 'choco'} # TypeError</code>
Dictionary	The dictionary is a useful data structure for storing (key, value) pairs.	<code>print(calories['apple']) < calories['choco']) # True calories['apple'] = 740 print(calories['apple']) < calories['apple']) # False print(len(calories)) # True print(SZ in calories.values()) # True</code>
Reading and writing elements	Read and write elements by specifying the key within the brackets. Use the <code>key</code> and <code>value</code> of the dictionary to access all keys and values of the dictionary.	<code>for k, v in calories.items(): print(k) # prints all keys else None # 'chocolate'</code>
Dictionary Looping	You can loop over the (key, value) pairs of a dictionary with the <code>items()</code> method.	<code>basket = {'apple', 'eggs', 'banana', 'orange'} print('eggs' in basket) # True print('mushroom' in basket) # False</code>
Membership operator	Check with the "in" keyword whether the set, list, or dictionary contains an element. Set comprehension is faster than list comprehension.	<code># List comprehension 1 = [x * x for x in ['Alice', 'Bob', 'Peter']] 1[0] # ['Hi Alice', 'Hi Bob', 'Hi Peter'] 1[2] # 'Hi Peter' 1[3] # IndexError: list index out of range # Set comprehension squares = {x ** 2 for x in [8, 2, 4]} # {8, 4}</code>
List and Set Comprehension	List comprehension is the concise Python way to create lists. Use brackets plus an expression, followed by a for clause. Close with zero or more for if clauses.	
	Set comprehension is similar to list comprehension.	

finxter

Python Cheat Sheet - Classes

"A puzzle a day to learn, code, and play" → Visit finxter.com

Description		Example
Classes	A class encapsulates data and functionality: data as attributes, and functionality as methods. It is a blueprint to create concrete instances in the memory.	<pre>class Dog: """ Blueprint of a dog """ # class variable shared by all instances species = "canis lupus" def __init__(self, name, color): self.name = name self.state = "sleeping" self.color = color def command(self, x): if x == self.name: self bark() elif x == "sit": self.state = "sit" else: self.state = "wag tail" def bark(self, freq=1): for i in range(freq): print("[" + self.name + "] woof!") Bello = Dog("bello", "black") Alice = Dog("alice", "white") print(Bello.color) # black print(Alice.color) # white</pre>
Instances		<pre>Bello.command("sit") print("[alice] " + Alice.state) # [alice]: sit Alice.command("woof") print("[bello] " + Bello.state) # [bello]: wag tail Alice.command("woof") # [alice]: woof! # [alice]: woof!</pre>
Instance	You are an instance of the class <code>human</code> . An instance is a concrete implementation of a class: all attributes of an instance are <code>instance variables</code> . Your hair is blond, brown, or black - but never unspecified.	
	Each instance has its own attributes independent of other instances. Yet, class variables are different. These are data values associated with the class, not the instances. All instances share the same class variable <code>species</code> in the example.	
Self	The first argument when defining any method is always the <code>self</code> argument. This argument specifies the instance on which you call the method.	
	<code>self</code> gives the Python interpreter the information about the concrete instance. To define a method, you use <code>self</code> to modify the instance attributes. But to call an instance method, you do not need to specify <code>self</code> .	
Creation	You can create classes "on the fly" and use them as logical units to store complex data types.	<pre>class Employee(): pass Alice = Employee() Alice.firstname = "alice" Alice.lastname = "wonderland" Alice.salary = 122000 Alice.species = "human" print(Alice.firstname + " " + Alice.lastname + " " + str(Alice.salary) + "\$") # alice wonderland 122000\$</pre>

finxter

Python Cheat Sheet - Functions and Tricks

"A puzzle a day to learn, code, and play" → Visit finxter.com

Description		Example	Result
A D V A C E	<code>map(func, iter)</code>	Executes the function on all elements of the iterable.	<code>['r', 'g', 'b']</code>
	<code>map(func, *it, *ik)</code>	Executes the function on all k elements of the k iterables.	<code>['@ apple', '2 oranges', '2 bananas']</code>
	<code>string.join(iter)</code>	Concatenates iterable elements separated by <code>string</code>	<code>'Alice' + join(list(['Alice', 'Bob']))</code>
F U N T O R	<code>filter(func, iterable)</code>	Filters out elements in iterable for which function returns False (or 0).	<code>[18]</code>
	<code>string.strip()</code>	Removes leading and trailing whitespace of string	<code>print("\n \t 42 \n").strip()</code>
	<code>sorted(iter)</code>	Sorts iterable in ascending order	<code>[2, 3, 5, 8, 42]</code>
	<code>sorted(iter, key=key)</code>	Sorts according to the key function in ascending order	<code>[42, 2, 3, 5, 8]</code>
	<code>help(func)</code>	Returns documentation of func	<code>... to uppercase.</code>
	<code>zip(*l1, *l2, ...)</code>	Groups the i-th elements of iteratos l1, l2 ... together	<code>[[{'Alice': 'Bob'}, {'Ann': 'Bob'}, {'Tom': 'Frank'}]]</code>
	<code>Unzip</code>	Equal to: 1) unpack the zipped list, 2) zip the result	<code>[[{'Alice': 'Bob'}, {'Ann': 'Bob'}, {'Tom': 'Bob'}], [{"Alice": "Ann"}, {"Tom": "Ann"}]]</code>
	<code>enumerate(iter)</code>	Assigns a counter value to each element of the iterable	<code>[[0, 'Alice'], (1, 'Bob'), (2, 'Tom')]</code>
T H I C K S		Share files between PC and phone? Run command in PC's shell: <code>adb push</code> Is any port number 0-65535. Type <IP address of PC>:port in the phone's browser. You can now browse the files in the PC directory.	
	<code>Read comic</code>	Import comic series stored in your web browser	
	<code>Zen of Python</code>	<code>'...beautiful is better than ugly. Explicit is ...'</code>	
	<code>Swapping numbers</code>	Swapping variables is a breeze in Python. No offense, Java!	<code>a, b = 'Jane', 'Alice' a, b = b, a a = 'Alice' b = 'Jane'</code>
	<code>Unpacking arguments</code>	Use a sequence as function arguments via asterisk operator *. Use a dictionary {key, value} via double asterisk operator **	<code>def f(x, y, z): return x + y * z f(1, 3, 4) f(*[1, 3, 4]) f(**{'x': 1, 'y': 2, 'z': 3})</code>
	<code>Extended Unpacking</code>	Use unpacking for multiple assignment feature in Python	<code>a, *b = [1, 2, 3, 4, 5] a = 1 b = [2, 3, 4, 5]</code>
	<code>Merge two dictionaries</code>	Use unpacking to merge two dictionaries into a single one	<code>x = {'Alice': 18, 'Bob': 27, 'Ann': 22} y = {'Bob': 27, 'Ann': 22} z = {**x, **y}</code>

finxter

Want to learn Python well, but don't have much time? Then this course is for you. It contains 5 carefully designed PDF cheat sheets. Each cheat sheet takes you one step further into the rabbit hole. You will learn practical Python concepts from the hand-picked examples and code snippets. The topics include basic keywords, simple and complex data types, crucial string and list methods, and powerful Python one-liners. If you lead a busy life and do not want to compromise on quality, this is the cheat sheet course for you!

(7) Dataquest Data Science Cheat Sheet – Python Basics → <https://s3.amazonaws.com/dq-bl/og-files/python-cheat-sheet-basic.pdf>

BASICS, PRINTING AND GETTING HELP

```
x = 2 # Assign 2 to the variable x
help(x) # Show documentation for the str data type
print(x) # Print the value of x
help(print) # Show documentation for the print() function
type(x) # Return the type of the variable x (in this case, it'll be int)
```

READING FILES

```
f = open("my_file.txt","r") # Open the file "my_file.txt" in read mode
f.read() # Read the contents of the file
f.close() # Close the file
with open("my_file.txt") as f: # Open the file "my_file.txt" and assign its contents to a variable named f
    f.read() # Read the contents of the file
    f.close() # Close the file
with open("my_dataset.csv") as f: # Open the CSV file my_dataset.csv and assign its data to the list of lists csv_my_list
    csv_my_list = []
    for line in f:
        csv_my_list.append(line)
    f.close()
```

STRINGS

```
s = "Hello".upper() # Assign the string "Hello" to the variable s and convert it to uppercase
s[0] # Return the first character in s
s[-1] # Return the last character in s
s[2:4] # Return a slice of s containing the second and third values of s
len(s) # Return the length of s
s.count("o") # Return the number of characters in s that equal "o"
s.replace("o","oo") # Replace all occurrences of "o" replaced with "oo"
s.split(",") # Split the string "s" into a list of strings based on the character ","
s[1:-1] # Return all the characters in s except the first and last character and return that list
```

NUMERIC TYPES AND MATHEMATICAL OPERATIONS

```
i = int("5") # Convert the string "5" to the integer 5
f = float("2.5") # Convert the string "2.5" to the float value 2.5 and assign the result to f
x = i + f # Addition
x = i - f # Subtraction
x = i * f # Multiplication
x = i / f # Division
x = i % f # Modulo operation
x ** 2 # Raise x to the power of 2 (or x2)
27 ** (1/3) # The third root of 27 (or  $\sqrt[3]{27}$ )
x = 100 # Assign the value of x = 100
x // 10 # Assign the value of x // 10 to x
```

LISTS

```
l = [100, 21, 45, 7] # Assign a list containing the integers 100, 21, 45, and 7 to the variable l
l[0] # Return the first item in l
l[1:3] # Return a slice (list) containing the second and third values of l
len(l) # Return the length of l
sum(l) # Return the sum of the values in l
min(l) # Return the minimum value from l
max(l) # Return the maximum value from l
l.append(1) # Append the value 1 to the end of l
l.insert(1, 2) # Insert the value 2 at index 1
l.pop(1) # Remove the first value in l
l.remove(2) # Remove the first occurrence of 2 from l
l.extend([3, 4]) # Extend the list l with the list [3, 4]
```

DICTIONARIES

```
d = {"Carrie": "Bratton", "Mike": "Bratton", "Carmichael": "Bratton", "Oscar": "Bratton", "Dora": "Bratton"} # Create a dictionary of "Carrie", "Mike", and "Oscar" and corresponding values of "Bratton", "Bratton", "Bratton", and "Bratton"
d["Oscar"] # Return the value associated with the key "Oscar"
d.get("Oscar") # Return the value from the dictionary d if the key "Oscar" exists, otherwise return None
d.get("Mike", "Sorry") # Return the value from the dictionary d if the key "Mike" is not found in d
d.keys() # Return a list of the keys from d
d.values() # Return a list of the values from d
d.items() # Return a list of (key, value) pairs from d
for d in d: # Loop through the keys in d
```

MODULES AND FUNCTIONS

```
import random # Import the module random
from math import sqrt # Import the function sqrt from the module math
```

IF STATEMENTS AND LOOPS

```
If the body of statements and loops are defined under indentation
if x > y:
    print("x is greater than y")
else:
    print("x is less than y")
```

BOOLEAN COMPARISONS

```
x == y # Test whether x is equal to y
x != y # Test whether x is not equal to y
x < y # Test whether x is less than y
x <= y # Test whether x is less than or equal to y
x > y # Test whether x is greater than or equal to y
x >= y # Test whether x is greater than or equal to y
x >= y or name == "alpha" # Test whether x is greater than or equal to y or name equals "alpha"
x <= y and name == "alpha" # Test whether both x is less than or equal to y and name equals "alpha"
x >= y or name == "alpha" # Test whether either x is greater than or equal to y or name equals "alpha"
x >= y and name == "alpha" # Test whether both x is greater than or equal to y and name equals "alpha"
```

PRINT STATEMENT

```
print("Hello") # Print the value "Hello" to the console
print("x = ", x) # Print the value of the variable x and run the code below based on the value of x
print("x = ", x) # Print the value of the variable x and run the code below based on the value of x
print("x = ", x) # Print the value of the variable x and run the code below based on the value of x
print("x = ", x) # Print the value of the variable x and run the code below based on the value of x
```

LEARN DATA SCIENCE ONLINE

Start Learning For Free: www.dataquest.io

The wonderful team at Dataquest have put together this comprehensive beginners cheat sheet. It covers all the basic data types, looping and reading files. It's beautifully designed and is the first of a series.

(<https://s3.amazonaws.com/dq-blog-files/python-cheat-sheet-intermediate.pdf>) **Dataquest Data Science Cheat Sheet - Intermediate** (<https://s3.amazonaws.com/dq-blog-files/python-cheat-sheet-intermediate.pdf>)



This intermediate-level cheat sheet is a follow-up of the other Dataquest cheat sheet. It contains intermediate dtype methods, looping and handling errors.

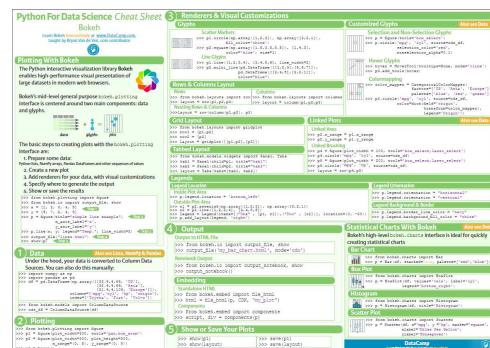
(9) Dataquest NumPy → <https://s3.amazonaws.com/dq-blog-files/numpy-cheat-sheet.pdf>



NumPy → <https://blog.finxter.com/numpy-tutorial/> is at the heart of data science. Advanced libraries like scikit-learn, Tensorflow, Pandas, and Matplotlib all built on NumPy arrays. You need to understand NumPy before you can thrive in data science and machine learning. The topics of this cheat sheet are creating arrays, combining arrays, scalar math, vector math and statistics.

This is only one great NumPy cheat sheet—if you want to get more, [check out our article on the 10 best NumPy cheat sheets](https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/) → <https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/>!

(10) Python For Data Science (Bokeh) → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Bokeh_Cheat_Sheet.pdf



Want to master the visualization library [Bokeh](https://docs.bokeh.org/en/latest/index.html) → <https://docs.bokeh.org/en/latest/index.html>? This cheat sheet is for you! It contains all the basic Bokeh commands to get your beautiful visualizations going fast!

(11) Pandas Cheat Sheet for Data Science → https://drive.google.com/file/d/1UHK8wtWbADvHKXFC937IS6MTnISZC_zB/view

Data Science Cheat Sheet	
Pandas	
KEY This key shows definitions of the cheat sheet: df - A DataFrame object s - A pandas Series object	
IMPORTS	Import these to use Pandas: import pandas as pd import numpy as np
IMPORTING DATA	
pd.read_csv(filename) - From a CSV file	df[col] - Returns column with label col as Series
pd.read_excel(filename) - From a Excel file	df[[col1, col2]] - Returns columns as a array
pd.read_excel(filename) - From an Excel file	df.loc[1] - Selection by position
pd.read_json(url) - From a JSON URL	df.loc[1:, 1] - Selection by index
pd.read_json(json_string) - Reads from a JSON string	df.iloc[1] - From index
pd.read_html(url) - Parses an HTML, using it's and extracts tables to a list of dataframes	df.iloc[1, 1] - First element of first column
pd.read_sql(sql, con) - From a database	
pd.read_sql_table(sql, connection, schema)	
pd.read_sql_query(sql, connection, schema)	
pd.read_csv(file) - Reads from a file	
pd.read_csv(file) - Reads from a file	
pd.read_json(file) - Writes to a JSON file	
pd.to_sql(table_name, connection, df) -	
pd.to_json(file) - Writes to a file in JSON format	
pd.to_html(file) - Writes to an HTML table	
pd.to_clipboard() - Writes to the clipboard	
TRANSFORMING DATA	
df.to_csv(file) - Writes to a CSV file	df.fillna(value) - Returns a copy of the DataFrame with null values filled.
df.to_excel(filename) - Writes to an Excel file	df.fillna(0) - Fills all null values with zero.
df.to_sql(table_name, connection, df) -	df.fillna(method='ffill') - Fills all null values with previous value.
df.to_json(file) - Writes to a file in JSON format	df.fillna(method='bfill') - Fills all null values with next value.
pd.DataFrame(df) - From a dict, keep for reference	df.fillna(method='ffill', limit=1) - Fills the average across all columns for every unit.
pd.read_csv(file) - Reads from a CSV file	df.fillna(method='bfill', limit=1) - Fills the average across all columns for every unit.
pd.read_excel(filename) - From an Excel file	df.fillna(method='ffill', limit=1) - Fills the average across all columns for every unit.
pd.read_json(file) - Reads from a JSON file	df.fillna(method='bfill', limit=1) - Fills the average across all columns for every unit.
pd.read_html(url) - Parses an HTML, using it's and extracts tables to a list of dataframes	df.fillna(method='ffill', limit=1) - Fills the average across all columns for every unit.
pd.read_sql(sql, con) - From a database	df.fillna(method='bfill', limit=1) - Fills the average across all columns for every unit.
pd.read_sql_table(sql, connection, schema)	
pd.read_sql_query(sql, connection, schema)	
pd.read_csv(file) - Reads from a file	
pd.read_csv(file) - Reads from a file	
pd.read_json(file) - Writes to a JSON file	
pd.to_sql(table_name, connection, df) -	
pd.to_json(file) - Writes to a file in JSON format	
pd.to_html(file) - Writes to an HTML table	
pd.to_clipboard() - Writes to the clipboard	
CREATE TEST OBJECTS	
df[0] - First row of the DataFrame	df.append(df2) - Adds the rows in df2 to the end of df1 (rows should be identical).
df[0:10] - First 10 rows of the DataFrame	pd.concat([df1, df2], axis=0) - Adds the rows in df2 to the end of df1 (rows should be identical).
df.shape - Number of rows and columns	df[0:10] - First 10 rows of the DataFrame
df.info() - Index, Datatype and Memory	df[0:10] - First 10 rows of the DataFrame
df.describe() - Summary statistics for numerical columns	df[0:10] - First 10 rows of the DataFrame
df.kurtosis(dropna=False) - From kurtosis values and counts	df[0:10] - First 10 rows of the DataFrame
df.agg(['sum', 'mean', 'count']) - Unique values and counts for all columns	df[0:10] - First 10 rows of the DataFrame
df.apply(pd.Series.value_counts) - Unique values and counts for all columns	df[0:10] - First 10 rows of the DataFrame
VIEWING/INSPECTING DATA	
df.loc[1] - Find one row of the DataFrame	df[0:10].head() - Returns the first 10 rows of the DataFrame
df.loc[1:10] - Find 10 rows of the DataFrame	df[0:10].tail() - Returns the last 10 rows of the DataFrame
df.shape - Number of rows and columns	df[0:10].head(n) - Returns the first n rows of the DataFrame
df.info() - Index, Datatype and Memory	df[0:10].tail(n) - Returns the last n rows of the DataFrame
df.describe() - Summary statistics for numerical columns	df[0:10].head(n) - Returns the first n rows of the DataFrame
df.kurtosis(dropna=False) - From kurtosis values and counts	df[0:10].tail(n) - Returns the last n rows of the DataFrame
df.agg(['sum', 'mean', 'count']) - Unique values and counts for all columns	df[0:10].head(n) - Returns the first n rows of the DataFrame
df.apply(pd.Series.value_counts) - Unique values and counts for all columns	df[0:10].tail(n) - Returns the last n rows of the DataFrame
FILTER, SORT, & GROUPBY	
df[df[0] > 0.5] - Rows where the col0 value is greater than 0.5	df.sort_index(ascending=False) - Sorts values by index in descending order
df[df[0] > 0.5 & df[1] < 0.7] - Rows where col0 > 0.5 and col1 < 0.7	df.sort_index(ascending=True) - Sorts values by index in ascending order
df.groupby('category').sum() - Groups values by category	df.sort_values([col1, col2], ascending=[True, False]) - Sorts values by col1 in ascending order and col2 in descending order
df.groupby('category').sum().reset_index() - Groups values and counts	df.sort_values([col1, col2], ascending=[False, True]) - Sorts values by col1 in descending order and col2 in ascending order
df.groupby('category').sum().groupby('category').sum() - Groups values and counts for all categories	df.groupby('category').sum() - Groups values by category
df.groupby('category').sum().groupby('category').sum().reset_index() - Groups values and counts for all categories	df.groupby('category').sum().reset_index() - Groups values by category
STATISTICS	
df.agg(np.all) - Applied to a series as well.	df.describe() - Summary statistics for numerical columns
df.describe() -	df.mean() - Returns the mean of all columns
df.describe() -	df.corr() - Returns the correlation between columns
df.describe() -	df.count() - Returns the number of non-null values in all DataFrame columns
df.describe() -	df.isnull().sum() - Returns the sum of each column's missing values
df.describe() -	df.sum() - Returns the sum of each column's values
df.describe() -	df.max() - Returns the largest value in each column
df.describe() -	df.median() - Returns the median of each column
df.describe() -	df.std() - Returns the standard deviation of each column

Pandas → <https://pandas.pydata.org/> is everywhere. If you want to master “the Excel library for Python coders”, why not start with this cheat sheet? It’ll get you started fast and introduces the most important Pandas functions to you.

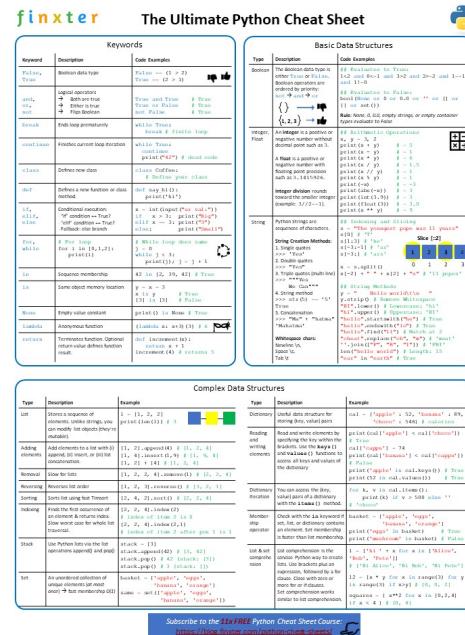
You can find a best-of article about the → <https://blog.finxter.com/pandas-cheat-sheets/7> → <https://blog.finxter.com/pandas-cheat-sheets/> best Pandas Cheat Sheets here. → <https://blog.finxter.com/pandas-cheat-sheets/>

(12) Regular Expressions Cheat Sheet → <https://www.dataquest.io/wp-content/uploads/2019/03/python-regular-expressions-cheat-sheet.pdf>



Regex to the rescue! [Regular expressions](https://blog.finxter.com/python-regex/) → <https://blog.finxter.com/python-regex/> are wildly important for anyone who handles large amounts of text programmatically (ask Google). This cheat sheet introduces the most important Regex commands for quick reference. Download & master these regular expressions!

(13) The World's Most Concise Python Cheat Sheet → https://blog.finxter.com/wp-content/uploads/2020/07/Finxter_WorldsMostDensePythonCheatSheet.pdf



Python Ultimate Cheat Sheet

This cheat sheet is the most concise Python cheat sheet in the world. It contains keywords, basic data structures, and complex data structures—all in a single 1-page PDF file. If you’re lazy, this cheat sheet is a must!

If you love cheat sheets, here are some interesting references for you (lots of more PDF downloads):

Related articles:

- [Collection] 10 Best NumPy Cheat Sheets Every Python Coder Must Own → <https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/>
 - [Python OOP Cheat Sheet] A Simple Overview of Object-Oriented Programming → <https://blog.finxter.com/object-oriented-programming-terminology-cheat-sheet/>
 - [Collection] 15 Mind-Blowing Machine Learning Cheat Sheets to Pin to Your Toilet Wall → <https://blog.finxter.com/machine-learning-cheat-sheets/>
 - Your 8+ Free Python Cheat Sheets → <https://blog.finxter.com/python-cheat-sheets/>
 - Python Beginner Cheat Sheet: 19 Keywords Every Coder Must Know → <https://blog.finxter.com/python-cheat-sheet/>
 - Python Cheat Sheet – Functions and Tricks → <https://blog.finxter.com/python-cheat-sheet-functions-and-tricks/>
 - Python Cheat Sheet — Coding Interview Questions → https://blog.finxter.com/cheatsheet-python-6_coding-interview-questions/

What Cheat Sheets are NOT a Replacement For

Your wall could be full of cheat sheets. You shelves full of the most up to date coding books. You could be subscribed to all the best online Python courses. But none of this matters if you don't put in the work.

To become a coder, you need to sit down and actually write code. It's best to do it every day. This ensures you are making continuous progress. Even 20 minutes per day adds up to 121 hours over a year – 3 working weeks!

It doesn't matter what you do. You can read over a cheat sheet, work through a course, or sit down and try to build something you think is cool. Whatever it is, the most important thing is that you typing Python into your computer. Cheat sheets are here to make that process easier, not as a replacement for it.

Now let's look at some other things to help you on your Python journey.

Cheat Sheet Alternatives

We all learn in different ways. What works well for one person may be terrible for another. So here are some alternatives to cheat sheets.

Flashcards

Flashcards took me from amateur to professional. I recommend that you write down all the information you want to remember on flashcards. You could create them in real life. But far better is the free app [Anki](https://apps.ankiweb.net/) → <https://apps.ankiweb.net/>. It works on your phone, tablet and computer. They use a scientifically proven spaced repetition system to help you learn. This means it shows you flashcards you find difficult more often than the ones you find easy. Instead of scrolling through social media next time you're waiting in line, why not answer some coding questions that you wrote?

Personalised Cheat Sheets

The cheat sheets in this article are great to give you inspiration. But they may include information that you already know. Plus, you're 10x more likely to remember something if you've written it down and created it yourself. So, at the end of a topic you've learned, make a 1-page cheat sheet of all the most important info. This ensures you only include information that's relevant to you and that it's valuable every time you look at it.

Projects

At Finxter, we believe in the power of project-based learning. Programming is supposed to be fun and interesting. What's more fun than building something you think is amazing?

Come up with an idea and start creating it. If you get stuck, google is your best friend. There is so much content out there and so many wonderful communities. They are happy to help aspiring coders like you. Start typing and playing and see what happens. You'll learn loads and have a lot of fun too!

If you want to learn Python as quickly as possible through project-based learning, check out our [Python Freelancer course](https://www.digistore24.com/redir/261950/adsmurphy/) → <https://www.digistore24.com/redir/261950/adsmurphy/>.

The 5 Best Python Cheat Sheets

Summary

Cheat sheets are a wonderful way to apply the 80/20 principle. You only need to put in 20% of the learning effort and you get 80% of the results. These helped me when I was first learning Python and I know they will do the same for you. Best of luck!

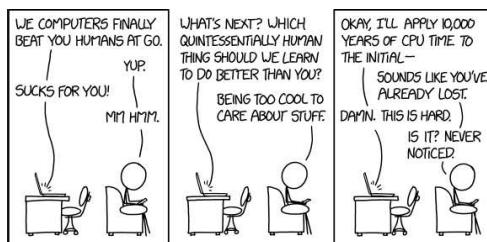
Do you want the most dense cheat sheet in Python? [Click here! → https://perso.limsi.fr/pointal/_media/python:courses/memento/python3-english.pdf](https://perso.limsi.fr/pointal/_media/python:courses/memento/python3-english.pdf)

Do you want the most comprehensive cheat sheet course? [Sign up now! → https://blog.finxter.com/subscribe/](https://blog.finxter.com/subscribe/)

Related Articles:

- [Collection] [11 Python Cheat Sheets Every Python Coder Must Own](https://blog.finxter.com/collection-5-cheat-sheets-every-python-coder-must-own/) → <https://blog.finxter.com/collection-5-cheat-sheets-every-python-coder-must-own/>
- [Python OOP Cheat Sheet] [A Simple Overview of Object-Oriented Programming](https://blog.finxter.com/object-oriented-programming-terminology-cheat-sheet/) → <https://blog.finxter.com/object-oriented-programming-terminology-cheat-sheet/>
- [Collection] [15 Mind-Blowing Machine Learning Cheat Sheets to Pin to Your Toilet Wall](https://blog.finxter.com/machine-learning-cheat-sheets/) → <https://blog.finxter.com/machine-learning-cheat-sheets/>
- [Your 8+ Free Python Cheat Sheet \[Course\]](https://blog.finxter.com/python-cheat-sheets/) → <https://blog.finxter.com/python-cheat-sheets/>
- [Python Beginner Cheat Sheet: 19 Keywords Every Coder Must Know](https://blog.finxter.com/python-beginner-cheat-sheet-19-keywords-every-coder-must-know/) → <https://blog.finxter.com/python-beginner-cheat-sheet-19-keywords-every-coder-must-know/>
- [Python Functions and Tricks Cheat Sheet](https://blog.finxter.com/python-functions-and-tricks-cheat-sheet/) → <https://blog.finxter.com/python-functions-and-tricks-cheat-sheet/>
- [Python Cheat Sheet: 14 Interview Questions](https://blog.finxter.com/python-cheat-sheet-14-interview-questions/) → <https://blog.finxter.com/python-cheat-sheet-14-interview-questions/>
- [Beautiful Pandas Cheat Sheets](https://blog.finxter.com/beautiful-pandas-cheat-sheets/) → <https://blog.finxter.com/beautiful-pandas-cheat-sheets/>
- [10 Best NumPy Cheat Sheets](https://blog.finxter.com/10-best-numpy-cheat-sheets/) → <https://blog.finxter.com/10-best-numpy-cheat-sheets/>
- [Python List Methods Cheat Sheet \[Instant PDF Download\]](https://blog.finxter.com/python-list-methods-cheat-sheet-instant-pdf-download/) → <https://blog.finxter.com/python-list-methods-cheat-sheet-instant-pdf-download/>
- [[Cheat Sheet](https://blog.finxter.com/cheat-sheet-6-pillar-machine-learning-algorithms/)] [6 Pillar Machine Learning Algorithms](https://blog.finxter.com/cheat-sheet-6-pillar-machine-learning-algorithms/) → <https://blog.finxter.com/cheat-sheet-6-pillar-machine-learning-algorithms/>

Programmer Humor



It's hard to train deep learning algorithms when most of the positive feedback they get is sarcastic. — from [xkcd](https://imgs.xkcd.com/comics/computers_vs_humans.png) → https://imgs.xkcd.com/comics/computers_vs_humans.png



While working as a researcher in distributed systems, [Dr. Christian Mayer](https://blog.finxter.com/about/) → <https://blog.finxter.com/about/> found his love for teaching computer science students.

To help students reach higher levels of Python success, he founded the programming education website [Finxter.com](https://finxter.com/) → <https://finxter.com/> that has taught exponential skills to millions of coders worldwide. He's the author of the best-selling programming books [Python One-Liners](https://amzn.to/2WAYeJE) → <https://amzn.to/2WAYeJE> (NoStarch 2020), [The Art of Clean Code](https://www.amazon.com/Art-Clean-Code-Practices-Complexity/dp/1718502184) → <https://www.amazon.com/Art-Clean-Code-Practices-Complexity/dp/1718502184> (NoStarch 2022), and [The Book of Dash](https://www.amazon.com/Python-Dash-Christian-Mayer-dp-1718502222/dp/1718502222) → <https://www.amazon.com/Python-Dash-Christian-Mayer-dp-1718502222/dp/1718502222> (NoStarch 2022). Chris also coauthored the [Coffee Break Python](http://coffeebreakpython.com/) → <http://coffeebreakpython.com/> series of self-published books. He's a computer science enthusiast, [freelancer](https://blog.finxter.com/become_python-freelancer-course/) → https://blog.finxter.com/become_python-freelancer-course/, and owner of one of the top 10 largest [Python blogs](https://blog.finxter.com/) → <https://blog.finxter.com/> worldwide.

His passions are writing, reading, and coding. But his greatest passion is to serve aspiring coders through Finxter and help them to boost their skills. You can [join his free email academy here](https://blog.finxter.com/email-academy/). → <https://blog.finxter.com/email-academy/>

NumPy - Python Wiki

Source: <https://wiki.python.org/moin/NumPy>

NumPy is Python's fundamental package for scientific computing. It is a library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

NumPy is used at the core of many popular packages in the world of Data Science and machine learning.

Learning NumPy

- The [official NumPy documentation](https://numpy.org/doc/stable/) → <https://numpy.org/doc/stable/> offers multiple thorough manuals including a getting started manual.
- Python Land offers a short and free [getting started with NumPy tutorial](https://python.land/data-science/numpy) → <https://python.land/data-science/numpy> and a comprehensive paid [NumPy course](https://python.land/product/numpy-course) → <https://python.land/product/numpy-course> called 'a gentle, hands-on introduction to NumPy'
- There's a complete but outdated (2006) manual by the principal author of Numpy, Travis Oliphant, is [available](http://csc.ucdavis.edu/~chaos/courses/nlp/Software/NumPyBook.pdf) → <http://csc.ucdavis.edu/~chaos/courses/nlp/Software/NumPyBook.pdf> for free (although donations are accepted).

NumPy examples

Many examples of NumPy usage can be found at http://wiki.scipy.org/Numpy_Example_List

- [numpy Example](#)

```
from numpy import *
from PIL import Image

ar = ones((100,100),float32)
ar = ar * 100

for i in range(0,100):
    ar[i,:] = 100 + (i * 1.5)

im = Image.fromarray(ar,"F")
```

[Collection] 10 Best NumPy Cheat Sheets Every Python Coder Must Own – Be on the Right Side of Change

By Emily Rosemary Collins

Source: <https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/>

Little time to learn NumPy? This article shows you the ten most amazing NumPy cheat sheets. Download them, print them, and pin them to your wall — and watch your data science skills grow! ??

All NumPy cheat sheets in this article are 100% free. All links open in a new tab (so feel free to click all links without worrying about losing this page).

Here's a quick summary if you don't have time reading all cheat sheets:

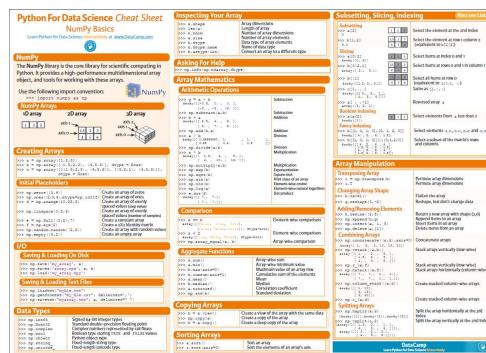
- [The visually most appealing NumPy cheat sheet \(incl. cheat sheet video\)](https://blog.finxter.com/numpy-cheat-sheet/) → <https://blog.finxter.com/numpy-cheat-sheet/>
- [The most comprehensive NumPy cheat sheet](http://hyperpolyglot.org/numerical-analysis) → <http://hyperpolyglot.org/numerical-analysis>
- [The most general data science cheat sheet](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Numpy_Python_Cheat_Sheet.pdf) → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Numpy_Python_Cheat_Sheet.pdf

Here's a quick download for you: I created this cheating sheet to explain some important NumPy concepts to [my coding students](#) → <https://blog.finxter.com/subscribe/>.

[NumPy](#) → <https://docs.scipy.org/> is a widely used Python scientific computing package. It simplifies linear algebra, matrix computations, and speeds up data analysis. Knowing NumPy is a prerequisite for other Python packages like pandas or Scikit-Learn.

This article should serve as the ultimate NumPy reference. The cheat sheets are diverse and range from one page to multiple pages. They also involve cross-language comparison cheat sheets. While some resources are great beginner's references, others are involved and require high-level expertise.

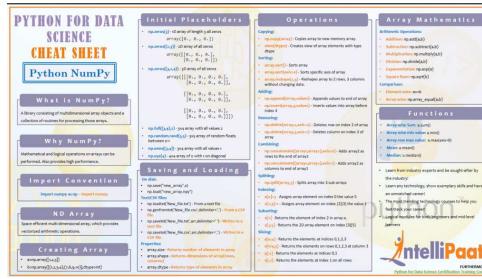
Cheat Sheet 1: DataCamp NumPy → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Numpy_Python_Cheat_Sheet.pdf



DataCamp is an online platform that offers data science training through videos and [coding exercises](#) → <https://finxter.com/>. This cheat sheet is one of the most comprehensive one-page cheat sheets available. In a way, it adds to the previous cheat sheet with more examples and more functions. It is a good summary of creating arrays and basic array design. This cheat sheet provides functions for the specific datatypes. At the end of the sheet is more advanced stuff like [slicing](#) → <https://blog.finxter.com/introduction-to-slicing-in-python/> and indexing. There are also some introductory tools for data analysis and array manipulation. Though overall, this is a fantastic resource, the one drawback is the color palette. The bright orange is a distraction from the content. If you like the color palette, this could be your comprehensive go-to list of the [NumPy basics](#) → <https://blog.finxter.com/numpy-tutorial/>.

- **Pros:** Comprehensive, graphics, very dense
- **Cons:** Overwhelming, color may distract

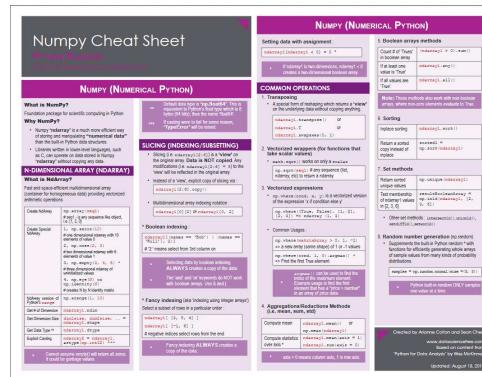
Cheat Sheet 2: Basic NumPy → <https://cdn.intellipaat.com/mediaFiles/2018/12/Python-NumPy-Cheat-Sheet-.pdf>



This is a useful resource for the NumPy basics. It provides a summary of creating arrays and some basic operations. It is minimalist, with a good overview of many basic functions. The sheet is divided into sections with headers for easier orientation. On the left-hand side of the sheet, the NumPy import convention is mentioned `import numpy as np`. A [one-line explanation](#) → <https://blog.finxter.com/python-one-liners-sampling-a-2d-python-list-to-speed-up-machine-learning/> follows each function. The biggest advantage of this list is good readability. This enables a quick search for the right function.

- **Pros:** Lightweight, basics, minimalist, readable
- **Cons:** Shallow, distracting background image, no visuals

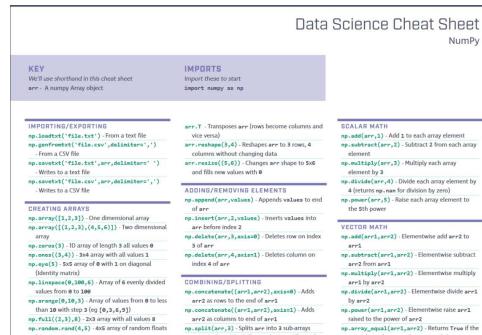
Cheat Sheet 3: A Little Bit of Everything → <http://datasciencefree.com/numpy.pdf>



The cheat sheet is divided into four parts. The first part goes into details about [NumPy arrays](https://blog.finxter.com/how-to-convert-list-of-lists-to-numpy-array/) → <https://blog.finxter.com/how-to-convert-list-of-lists-to-numpy-array/>, and some useful functions like `np.arange()` → <https://blog.finxter.com/numpy-arange/> or finding the number of dimensions. The 2nd part focuses on slicing and indexing, and it provides some delightful examples of [Boolean indexing](https://blog.finxter.com/how-to-use-numpy-boolean-indexing-to-uncover-instagram-influencers/) → <https://blog.finxter.com/how-to-use-numpy-boolean-indexing-to-uncover-instagram-influencers/>. The last two columns are a little bit disconnected. They provide a wide range of functions, ranging from matrix operations like transpose to [sorting an array](https://blog.finxter.com/python-list-sort/) → <https://blog.finxter.com/python-list-sort/>. However, the last two columns are not necessarily grouped conveniently. The advantage of this sheet is that it also includes Boolean and not only the [numerical types](https://blog.finxter.com/decimal-outputs-float-trap-and-how-to-solve-it/) → <https://blog.finxter.com/decimal-outputs-float-trap-and-how-to-solve-it/>.

- **Pros:** Focus on slicing and dimensionality
 - **Cons:** Disconnected content, slightly confusing, no graphics

Cheat Sheet 4: Data Science → <https://s3.amazonaws.com/dq-blog-files/numpy-cheat-sheet.pdf>



Dataquest is a similar online platform to DataCamp. It offers a variety of [data science](https://blog.finxter.com/codetutorial-break-numpy/) – <https://blog.finxter.com/codetutorial-break-numpy/> tracks and lessons, followed by coding exercises. This is another good resource of the most important NumPy functions and properties. The cheat sheet is readable with distinct sections, and each section has a clear title. Besides the sheet organization and excellent readability, it provides a range of functions and operations. Also, compared to the previous two cheat sheets, there is a [math](https://blog.finxter.com/python-math-module/) – <https://blog.finxter.com/python-math-module/> and statistics section. It divides the math sections into scalar and vector math, and there is a statistics section at the bottom.

- **Pros:** Statistics, HTML-based, well-structured, comprehensive, readable
 - **Cons:** No graphics
-

Cheat Sheet 5: NumPy for Matlab Users → <http://mathesaurus.sourceforge.net/matlab-numpy.html>

NumPy for MATLAB users		
Help		
MATLAB/Octave	Python	Description
doc	help()	Browse help interactively
help -i % browses with Info		
help for doc doc	help	Help on using help
help plot	help(plot) OR ?plot	Help for a function
help splines OR doc splines	help(splines)	Help for a toolbox/library package
demo		Demonstration examples
Searching available documentation		
MATLAB/Octave	Python	Description
lookfor plot		Search help files
help	help(); modules [Numeric]	List available packages
which plot	help(plot)	Locate functions
Using interactively		
MATLAB/Octave	Python	Description
clear	ipython -pylab	Start session
cls OR cls	CLS	Auto completion
foo.m	execfile('foo.py') OR run foo.py	Run code from file
history	hist -n	Command history
diary on [...] diary off	CTRL-D	Save command history
exit OR quit	CTRL-C	End session

If you are a Matlab user and need a quick introduction to [Python](https://blog.finxter.com/python-crash-course/) – <https://blog.finxter.com/python-crash-course/> and NumPy, this could be your go-to. The sheet contains three columns – the first column is the Matlab/Octave, the second column are the Python and NumPy equivalents, and the third is a description column. The sheet's focus is not solely on NumPy, but there are many Python basics listed. Since it is not a single sheet, the content is organized into separate sections. It provides [math](https://blog.finxter.com/python-math-module/) – <https://blog.finxter.com/python-math-module/>, logical and boolean operators, roots and round offs, complex numbers, extensive linear algebra, [reshaping](https://blog.finxter.com/numpy-reshape/) – <https://blog.finxter.com/numpy-reshape/> and [indexing](https://blog.finxter.com/indexing/) – <https://blog.finxter.com/indexing/>, some basic [plots](https://blog.finxter.com/plots/) – <https://blog.finxter.com/plots/>, calculus, and [statistics](https://blog.finxter.com/statistics/) – <https://blog.finxter.com/statistics/>.

- **Pros:** Matlab, Python background, a wide spectrum of related content
 - **Cons:** Unfocused, not visually too appealing, font choice strange
-

Cheat Sheet 6: The Matrix → https://sebastianraschka.com/blog/2014/matrix_cheatsheet_table.html

Task	MATLAB/Octave	Python NumPy	R	Julia	Task
CREATING MATRICES					
Creating Matrix (new 2d matrix)	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	Creating Matrix (new 2d matrix)
Creating an column vector (new 1d matrix)	$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$	Creating a column vector (new 1d matrix)			

This cheat sheet provides the equivalents for four different languages – MATLAB/Octave, Python and NumPy, R, and Julia. The list is not a single PDF sheet, but it is a scrollable document. On each far left-hand and the right-hand side of the document, there are task descriptions. This is an extensive sheet, and it is extra useful because the output of each task is given. The sheet covers creating and designing of matrices, matrix shape manipulation, and some basic and more advanced matrix operations. The advanced section is particularly interesting because it lists many useful functions in data analysis, like finding a covariance and eigenvalues and creating random normally distributed variables.

- **Pros:** Wide spectrum of related topics (Matlab, Python, NumPy, R, Julia), advanced-level features
- **Cons:** Bad readability, no PDF download

Cheat Sheet 7: Numerical Analysis → <http://hyperpolyglot.org/numerical-analysis>

Matrix	C	Matrix	Java
newdouble A[3][3]; double a[3][3]; a[0][0] = 1.0;	float a[3][3]; a[0][0] = 1.0;	float a[3][3]; a[0][0] = 1.0;	float a[3][3]; a[0][0] = 1.0;
a[1][0] = 2.0; a[2][0] = 3.0;			
a[0][1] = 4.0; a[1][1] = 5.0;			
a[2][1] = 6.0; a[0][2] = 7.0;			
a[1][2] = 8.0;	a[1][2] = 8.0;	a[1][2] = 8.0;	a[1][2] = 8.0;
for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { System.out.println(a[i][j]); } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { System.out.println(a[i][j]); } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { System.out.println(a[i][j]); } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { System.out.println(a[i][j]); } }

Matrix	C	Matrix	Java
for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { a[i][j] = 0.0; } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { a[i][j] = 0.0; } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { a[i][j] = 0.0; } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { a[i][j] = 0.0; } }
a[0][0] = 1.0; a[1][0] = 2.0; a[2][0] = 3.0;			
a[0][1] = 4.0; a[1][1] = 5.0;			
a[0][2] = 6.0; a[1][2] = 7.0;			
a[2][1] = 8.0;	a[2][1] = 8.0;	a[2][1] = 8.0;	a[2][1] = 8.0;
for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { System.out.println(a[i][j]); } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { System.out.println(a[i][j]); } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { System.out.println(a[i][j]); } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { System.out.println(a[i][j]); } }

Matrix	C	Matrix	Java
for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { a[i][j] = 0.0; } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { a[i][j] = 0.0; } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { a[i][j] = 0.0; } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { a[i][j] = 0.0; } }
a[0][0] = 1.0; a[1][0] = 2.0; a[2][0] = 3.0;			
a[0][1] = 4.0; a[1][1] = 5.0;			
a[0][2] = 6.0; a[1][2] = 7.0;			
a[2][1] = 8.0;	a[2][1] = 8.0;	a[2][1] = 8.0;	a[2][1] = 8.0;
for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { System.out.println(a[i][j]); } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { System.out.println(a[i][j]); } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { System.out.println(a[i][j]); } }	for (int i = 0; i < 3; i++) { for (int j = 0; j < 3; j++) { System.out.println(a[i][j]); } }

This is the most comprehensive sheet on the list. Not only that includes side-to-side equivalents between MATLAB, R, NumPy, and Julia; and it also covers everything from functions and syntax, to loops and I/O. The most interesting and useful component is that certain lines like a function definition are given for MATLAB, R, and Julia, but not for NumPy because of the lack of that functionality. That makes it easy to compare and contrast and to find the best fit for a project → <https://blog.finxtre.com/how-real-freelancers-earn-money-in-2019-10-practical-python-projects/>.

- **Pros:** Extremely comprehensive
 - **Cons:** Not well-structured, poor design, no PDF download
-

Cheat Sheet 8: NumPy for R (and S-plus) Users → <http://mathesaurus.sourceforge.net/r-numpy.html>

NumPy for R (and S-Plus) users

Help		
R/S-Plus	Python	Description
help.start()	help()	Browse help interactively
help()	help	Help on using help
help(plot) OR ?plot	help(plot) OR ?plot	Help for a function
help(package="splines")	help(pylab)	Help for a toolbox/library package
demo()		Demonstration examples
example(plot)		Example using a function

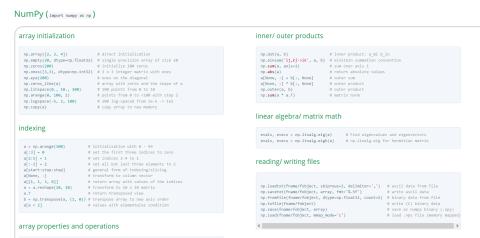
Searching available documentation		
R/S-Plus	Python	Description
help.search("plot")	search help files	
argspace("plot")	Find objects by partial name	
library()	help(); modules [Numeric]	List available packages
find(plot)	help(plot)	Locate functions
methods(plot)		List available methods for a function

Using interactively		
R/S-Plus	Python	Description
gui	ipython -pylab	Start session
source("foo.R")	execfile('foo.py') OR run foo.py	Auto completion
history()	histat -n	Run code from file
savehistory(file=".Rhistory")		Command history
q(save="no")	CTRL-D	Save command history
	CTRL-Z # windows	End session
	sys.exit()	

Although there are other comparison cheat sheets in this collection, this one lists some [advanced features → http://blog.finxtter.com/the-top-18-best-python-tricks/](http://blog.finxtter.com/the-top-18-best-python-tricks/). As the title says, it is a comparison between R(and S-plus) and NumPy. It is very detailed for each family of operations. For example, the sorting section provides eight ways to [sort an array → https://blog.finxtter.com/how-to-sort-in-one-line/](#). Some operations are not possible in both languages, so it is easy to find the right function. This is the only cheat sheet in the collection that provides detailed plots and graphs. Moreover, some advanced math and statistics were given, like differential equations and Fourier analysis.

- **Pros:** Advanced-level, comparison-based (R vs. NumPy), detailed, plots and graphs
- **Cons:** Confusing, not focused

Cheat Sheet 9: Scientific Python → https://ipgp.github.io/scientific_python_cheat_sheet/?utm_content=buffer7d821&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer#numpy-import-numpy-as-np



This is not a NumPy specific sheet. It covers many Python data science topics, but also some Python basics. It is easily navigated through because of the contents given in the beginning. The NumPy section is comprehensive. It covers NumPy basics like the array properties and operations. Also, it contains an extensive list of math functions and linear algebra functions. Some of the useful linear algebra functions are finding inner and outer products and eigenvalues. Others are functions for rounding off and generating [random → https://blog.finxter.com/python-random-module/variables](https://blog.finxter.com/python-random-module/variables).

- **Pros:** NumPy + science, maths, linear algebra, beautiful design
 - **Cons:** No PDF download, no visuals
-

Cheat Sheet 10: Finxter NumPy → <https://blog.finxter.com/numpy-cheat-sheet/>

Python Cheat Sheet: NumPy		
"A puzzle a day to learn, code, and play" → Visit finxter.com		
Name	Description	Example
<code>a.shape</code>	The shape attribute of NumPy arrays keeps a tuple of integers. Each integer describes the number of elements of the axis.	<pre>a = np.array([[1,2],[1,1],[0,0]]) print(np.shape(a)) # [3, 2]</pre>
<code>a.ndim</code>	The ndim attribute is equal to the length of the shape tuple.	<pre>print(np.ndim(a)) # 2</pre>
<code>*</code>	The asterisk (*) operator performs the Hadamard product, i.e., multiplies two matrices with equal shape element-wise.	<pre>a = np.array([[1, 0], [0, 1]]) b = np.array([[1, 1], [1, 1]]) print(*a*b) # [[1 0] [0 1]]</pre>
<code>np.matmul(a,b), a@b</code>	The standard matrix multiplication operator. Equivalent to the @ operator.	<pre>print(np.matmul(a,b)) # [[1 2] [2 2]]</pre>
<code>np.arange([start,]stop, [step,])</code>	Creates a new 1D numpy array with evenly spaced values.	<pre>print(np.arange(0,10,2)) # [0 2 4 6 8]</pre>
<code>np.linspace(start, stop, num=10)</code>	Creates a new 1D numpy array with evenly spread elements within the given interval	<pre>print(np.linspace(0,10,5)) # [0. 3. 6. 9.]</pre>
<code>np.average(a)</code>	Averages over all the values in the numpy array.	<pre>a = np.array([[1, 0], [0, 2]]) print(np.average(a)) # 1.0</pre>
<code>cslice[::val]</code>	Replace the <code>val</code> as selected by the slicing operator with the value <code>val</code> .	<pre>a = np.array([0, 1, 0, 0, 0]) a[::2] = 2 print(a) # [2 1 2 0 2]</pre>
<code>np.var(a)</code>	Calculates the variance of a numpy array.	<pre>a = np.array([1, 0]) print(np.var(a)) # 4.0</pre>
<code>np.std(a)</code>	Calculates the standard deviation of a numpy array.	<pre>print(np.std(a)) # 2.0</pre>
<code>np.diff(a)</code>	Calculates the difference between subsequent values in NumPy array <code>a</code> .	<pre>rbs = np.array([0, 1, 1, 2, 3, 5]) print(np.diff(rbs, n=1)) # [1 0 1 2 3]</pre>
<code>np.cumsum(a)</code>	Calculates the cumulative sum of the elements in NumPy array <code>a</code> .	<pre>print(np.cumsum(np.arange(5))) # [0 1 3 6 10]</pre>
<code>np.sort(a)</code>	Creates a new NumPy array with the values from a (increasing).	<pre>a = np.array([10,3,7,1,0]) print(np.sort(a)) # [0 1 3 7 10]</pre>
<code>np.argsort(a)</code>	Returns the indices of a NumPy array so that the indexed values would be sorted.	<pre>a = np.argsort([10,3,7,1,0]) print(np.argsort(a)) # [4 3 2 0]</pre>
<code>np.max(a)</code>	Returns the maximal value of NumPy array <code>a</code> .	<pre>a = np.array([10,3,7,1,0]) print(np.max(a)) # 10</pre>
<code>np.argmax(a)</code>	Returns the index of the element with maximal value in the NumPy array <code>a</code> .	<pre>a = np.argmax([10,3,7,1,0]) print(np.argmax(a)) # 0</pre>
<code>np.nonzero(a)</code>	Returns the indices of the nonzero elements in NumPy array <code>a</code> .	<pre>a = np.array([10,3,7,1,0]) print(np.nonzero(a)) # [0 1 2 3]</pre>

finxter

The [Finxter → https://finxter.com/cheat sheet](https://finxter.com/cheat sheet) is different from all the previously mentioned sheets because it's visually the clearest. It gives a detailed description of each function and lists the examples along with the outcome. The good thing about the visible outcome is that looking at it can help if you're unsure about the name of the function. Along with the cheat sheet, there is an accompanying video with further detailed examples and explanations.

- **Pros:** Easy, simple, visually clean, and focused on the most important functions
 - **Cons:** No graphics
-

Bonus Cheat Sheet: From NumPy to xtensor → <https://xtensor.readthedocs.io/en/latest/numpy.html>

From numpy to xtensor	
Containers	
Two container types are provided. <code>varray</code> (dynamic number of dimensions) and <code>xtensor</code> (static number of dimensions).	
Python 3 - numpy	C++ 14 - xtensor
<code>np.array([[1, 4], [5, 6]])</code>	<code>xt::varray<double>({{1, 4}, {5, 6}})</code>
<code>arr.reshape([1, 4])</code>	<code>arr.reshape({1, 4})</code>
<code>arr.astype(np.float64)</code>	<code>xt::cast<double>(arr)</code>
Initializers	
Lazy helper functions return tensor expressions. Return types don't hold any value and are evaluated upon access or assignment. They can be assigned to a container or directly used in expressions.	
Python 3 - numpy	C++ 14 - xtensor
<code>np.linspace(1.0, 10.0, 100)</code>	<code>xt::linspace<double>(1.0, 10.0, 100)</code>
<code>np.linspace(2.0, 3.0, 4)</code>	<code>xt::linspace<double>(2.0, 3.0, 4)</code>
<code>np.arange(3, 7)</code>	<code>xt::arange(3, 7)</code>

xtensor is a C++ library, similar to NumPy, made for numerical analysis. The cheat sheet provides a two-column view, where the first column is NumPy, and the second column contains the xtensor equivalents. The sheet focuses on array initialization, reshaping, and slicing functions. Further, it continues with array manipulation like transpose or rotation functions. There are a lot of tensor operations, but the sheet is missing the descriptions. So, it's not always easily deducible what a certain function does.

- **Pros:** Only xtensor sheet, simple, comparison-based
- **Cons:** Too specific for most coders

Programmer Humor

Q: What is the object-oriented way to become wealthy?



A: Inheritance.

Attribution

This article is contributed by Finxter user **Milica Cveticovic**. Milica is also a writer on Medium — check out her profile → https://medium.com/@l_am_milica/the-beginners-guide-to-downloading-twitter-data-using-tweepy-4ec981eaba77.

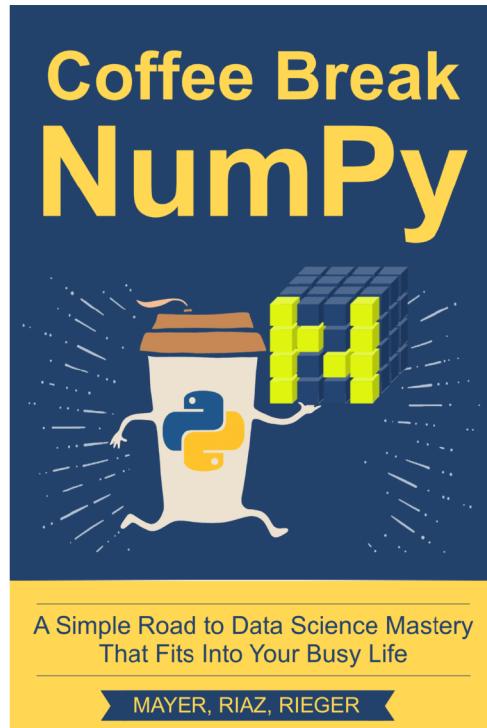
Where to Go From Here?

A thorough understanding of the NumPy basics is an important part of any data scientist's education. NumPy is at the heart of many advanced machine learning and data science libraries such as Pandas, TensorFlow, and Scikit-learn.

If you struggle with the NumPy library — fear not! Become a NumPy professional in no time with our new coding textbook “Coffee Break NumPy”. It’s not only a thorough introduction into the NumPy library that will increase your value to the marketplace. It’s also fun to go through the large collection of code puzzles in the

book.

[Get your Coffee Break NumPy! → https://blog.finxter.com/coffee-break-numpy/](https://blog.finxter.com/coffee-break-numpy/)



Related Articles:

- [Collection] 11 Python Cheat Sheets Every Python Coder Must Own → <https://blog.finxter.com/collection-5-cheat-sheets-every-python-coder-must-own/>
- [Python OOP Cheat Sheet] A Simple Overview of Object-Oriented Programming → <https://blog.finxter.com/object-oriented-programming-terminology-cheat-sheet/>
- [Collection] 15 Mind-Blowing Machine Learning Cheat Sheets to Pin to Your Toilet Wall → <https://blog.finxter.com/machine-learning-cheat-sheets/>
- Your 8+ Free Python Cheat Sheet [Course] → <https://blog.finxter.com/python-cheat-sheets/>
- Python Beginner Cheat Sheet: 19 Keywords Every Coder Must Know → <https://blog.finxter.com/python-cheat-sheet/>
- Python Functions and Tricks Cheat Sheet → <https://blog.finxter.com/python-cheat-sheet-functions-and-tricks/>
- Python Cheat Sheet: 14 Interview Questions → <https://blog.finxter.com/python-interview-questions/>
- Beautiful Pandas Cheat Sheets → <https://blog.finxter.com/pandas-cheat-sheets/>
- 10 Best NumPy Cheat Sheets → <https://blog.finxter.com/collection-10-best-numpy-cheat-sheets-every-python-coder-must-own/>
- Python List Methods Cheat Sheet [Instant PDF Download] → <https://blog.finxter.com/python-list-methods-cheat-sheet-instant-pdf-download/>

- [Cheat Sheet] 6 Pillar Machine Learning Algorithms → <https://blog.finxter.com/cheat-sheet-6-pillar-machine-learning-algorithms/>



Emily Rosemary Collins is a tech enthusiast with a strong background in computer science, always staying up-to-date with the latest trends and innovations. Apart from her love for technology, Emily enjoys exploring the great outdoors, participating in local community events, and dedicating her free time to painting and photography. Her interests and passion for personal growth make her an engaging conversationalist and a reliable source of knowledge in the ever-evolving world of technology.

[PDF Collection] 7 Beautiful Pandas Cheat Sheets — Post Them to Your Wall — Be on the Right Side of Change

Source: <https://blog.finxter.com/pandas-cheat-sheets/>



Python Logo

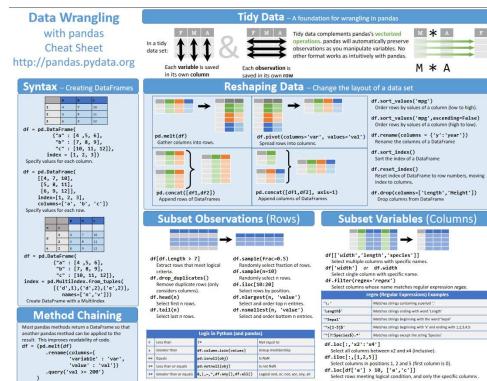
[PDF Collection] 7 Beautiful Pandas Cheat Sheets — Post Them to Your Wall

[Pandas](https://pandas.pydata.org/) → <https://pandas.pydata.org/> is an open-source Python library that is powerful and flexible for data analysis. If there is something you want to do with data, the chances are it will be possible in pandas. There are a vast number of possibilities within pandas, but most users find themselves using the same methods time after time. In this article, we compiled the best cheat sheets from across the web, which show you these core methods at a glance.

The primary data structure in pandas is the [DataFrame](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html) → <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html> used to store two-dimensional data, along with a label for each corresponding column and row. If you are familiar with Excel spreadsheets or SQL databases, you can think of the DataFrame as being the pandas equivalent. If we take a single column from a DataFrame, we have one-dimensional data. In pandas, this is called a [Series](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html) → <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html>. DataFrames can be created from scratch in your code, or loaded into Python from some external location, such as a CSV. This is often the first stage in any data analysis task. We can then do any number of things with our DataFrame in Pandas, including removing or editing values, filtering our data, or combining this DataFrame with another DataFrame. Each line of code in these cheat sheets lets you do something different with a DataFrame. Also, if you are coming from an Excel background, you will enjoy the performance pandas has to offer. After you get over the learning curve, you will be even more impressed with the functionality.

Whether you are already familiar with pandas and are looking for a handy reference you can print out, or you have never used pandas and are looking for a resource to help you get a feel for the library- there is a cheat sheet here for you!

1. The Most Comprehensive Cheat Sheet → https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf



https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

This one is from the pandas guys, so it makes sense that this is a comprehensive and inclusive cheat sheet. It covers the vast majority of what most pandas users will ever need to do to a DataFrame. Have you already used pandas for a little while? And are you looking to up your game? This is your cheat sheet! However, if you are newer to pandas and this cheat sheet is a bit overwhelming, don't worry! You definitely don't need to understand everything in this cheat sheet to get started. Instead, check out the next cheat sheet on this list.

2. The Beginner's Cheat Sheet → <https://www.dataquest.io/blog/pandas-cheat-sheet/>



<https://www.dataquest.io/blog/pandas-cheat-sheet/>

Dataquest is an online platform that teaches Data Science using interactive coding challenges. I love this cheat sheet they have put together. It has everything the pandas beginner needs to start using pandas right away in a friendly, neat list format. It covers the bare essentials of each stage in the data analysis process:

- Importing and exporting your data from an Excel file, CSV, HTML table or SQL database
- Cleaning your data of any empty rows, changing data formats to allow for further analysis or renaming columns
- Filtering your data or removing anomalous values
- Different ways to view the data and see its dimensions
- Selecting any combination of columns and rows within the DataFrame using loc and iloc
- Using the .apply method to apply a formula to a particular column in the DataFrame

- Creating summary statistics for columns in the DataFrame. This includes the median, mean and standard deviation
- Combining DataFrames

3. The Excel User's Cheat Sheet → <https://www.shanelynn.ie/using-pandas-dataframe-creating-editing-viewing-data-in-python/>

The Pandas DataFrame – loading, editing,
and viewing data in Python

44 Comments / blog, data science, Data Visualisation, Pandas, python, Tutorials / By shanelynn



Starting out with Python Pandas DataFrames

<https://www.shanelynn.ie/using-pandas-dataframe-creating-editing-viewing-data-in-python/>

Ok, this isn't quite a cheat sheet, it's more of an entire manifesto on the pandas DataFrame! If you have a little time on your hands, this will help you get your head around some of the theory behind DataFrames. It will take you all the way from loading in your trusty CSV from Microsoft Excel to viewing your data in Jupyter and handling the basics. The article finishes off by using the DataFrame to create a histogram and bar chart. For migrating your spreadsheet work from Excel to pandas, this is a fantastic guide. It will teach you how to perform many of the Excel basics in pandas. If you are also looking for how to perform the pandas equivalent of a VLOOKUP in Excel, check out Shane's [article](#) → <https://www.shanelynn.ie/merge-join-dataframes-python-pandas-index-1/> on the merge method.

4. The Most Beautiful Cheat Sheet → <https://www.enthought.com/wp-content/uploads/Enthought-Python-Pandas-Cheat-Sheets-1-8-v1.0.2.pdf>

Plotting with Pandas Series and DataFrames

Pandas uses Matplotlib to generate figures. Once a figure is generated with Pandas, all of Matplotlib's functions can be used to modify the title, labels, legend, etc. In a Jupyter notebook, all plotting calls for a given plot should be in the same cell.

Parts of a Figure

An Axes object is what you think of as a "plot". It has a title and two Axis objects with their own limits. Each Axis can have a label. There can be multiple Axis objects in a Figure.

Figure

Setup

Import packages:

```
> import pandas as pd
> import matplotlib.pyplot as plt
```

Select this at IPython prompt to display figures in new windows:

```
> %matplotlib
```

Use this in Jupyter notebooks to display static images inline:

```
> %matplotlib inline
```

Use this in Jupyter notebooks to display zoomable images inline:

```
> %matplotlib notebook
```

Plotting with Pandas Objects

Series	Dataframe	Labels
With a Series, Pandas plots values against the index.	With a Dataframe, Pandas creates one line per column.	Use Matplotlib to override or add annotations.

<https://www.enthought.com/wp-content/uploads/Enthought-Python-Pandas-Cheat-Sheets-1-8-v1.0.2.pdf>

If you're more of a visual learner, try this cheat sheet! Many common pandas tasks have intricate, color-coded illustrations showing how the operation works. On page 3, there is a fantastic section called 'Computation with Series and DataFrames', which provides an intuitive explanation for how DataFrames work and shows how the index is used to align data when DataFrames are combined and how element-wise operations work in contrast to operations which work on each row or column. At 8 pages long, it's more of a booklet than a cheat sheet, but it can still make for a great resource!

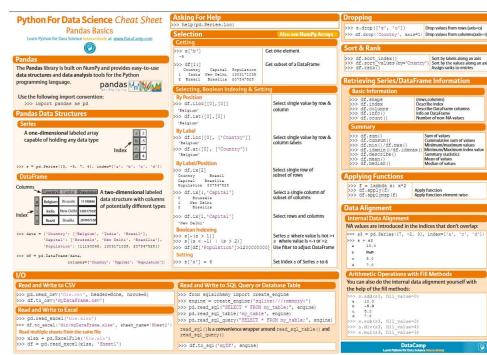
5. The Best Machine Learning Cheat Sheet → <https://elitedatascience.com/python-cheat-sheet>



<https://elitedatascience.com/python-cheat-sheet>

Much like the other cheat sheets, there is comprehensive coverage of the pandas basic in here. So, that includes filtering, sorting, importing, exploring, and combining DataFrames. However, where this Cheat Sheet differs is that it finishes off with an excellent section on [scikit-learn](https://scikit-learn.org/stable/) → <https://scikit-learn.org/stable/>, Python's machine learning library. In this section, the DataFrame is used to train a machine learning model. This cheat sheet will be perfect for anybody who is already familiar with machine learning and is transitioning from a different technology, such as R.

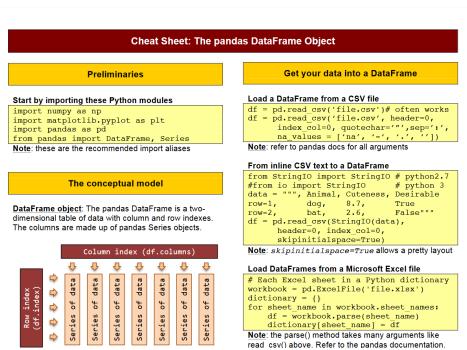
6. The Most Compact Cheat Sheet → <http://datacamp-community-prod.s3.amazonaws.com/dbed353d-2757-4617-8206-8767ab379ab3>



<http://datacamp-community-prod.s3.amazonaws.com/dbed353d-2757-4617-8206-8767ab379ab3>

Data Camp is an online platform that teaches Data Science with videos and coding exercises. They have made cheat sheets on a bunch of the most popular Python libraries, which you can also check out [here](https://www.datacamp.com/community/data-science-cheatsheets) → <https://www.datacamp.com/community/data-science-cheatsheets>. This cheat sheet nicely introduces the DataFrame, and then gives a quick overview of the basics. Unfortunately, it doesn't provide any information on the various ways you can combine DataFrames, but it does all fit on one page and looks great. So, if you are looking to stick a pandas cheat sheet on your bedroom wall and nail home the basics, this one might be for you! The cheat sheet finishes with a small section introducing NaN values, which come from NumPy. These indicate a null value and arise when the indices of two Series don't quite match up in this case.

7. The Best Statistics Cheat Sheet → <https://www.webpages.uidaho.edu/~stevel/504/pandas%20dataframe%20notes.pdf>



<https://www.webpages.uidaho.edu/~stevel/504/pandas%20dataframe%20notes.pdf>

While there aren't any pictures to be found in this sheet, it is an incredibly detailed set of notes on the pandas DataFrame. This cheat shines with its complete section on time series and statistics. There are methods for calculating covariance, correlation, and regression here. So, if you are using pandas for some advanced statistics or any kind of scientific work, this is going to be your cheat sheet.

Where to go from here?

For just automating a few tedious tasks at work, or using pandas to replace your crashing Excel spreadsheet, everything covered in these cheat sheets should be entirely sufficient for your purposes.

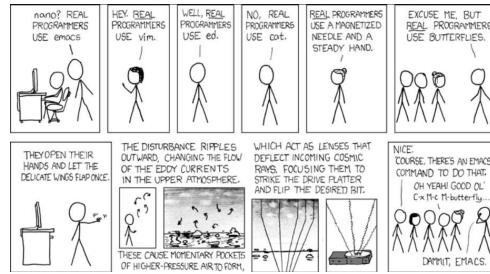
If you are looking to use pandas for Data Science, then you are only going to be limited by your knowledge of statistics and probability. This is the area that most people lack when they try to enter this field. I highly recommend checking out [Think Stats](https://greenteapress.com/thinkstats2/thinkstats2.pdf) → <https://greenteapress.com/thinkstats2/thinkstats2.pdf> by Allen B Downey, which provides an introduction to statistics using Python.

For those a little more advanced, looking to do some machine learning, you will want to start taking a look at the scikit-learn library. Data Camp has a great [cheat sheet](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Scikit_Learn_Cheat_Sheet_Python.pdf) → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Scikit_Learn_Cheat_Sheet_Python.pdf for this. You will also want to pick up a linear algebra textbook to understand the theory of machine learning. For something more practical, perhaps give the famous [Kaggle Titanic machine learning competition](https://www.kaggle.com/c/titanic) → <https://www.kaggle.com/c/titanic>.

Learning about pandas has many uses, and can be interesting simply for its own sake. However, Python is massively in demand right now, and for that reason, it is a high-income skill. At any given time, there are thousands of people searching for somebody to solve their problems with Python. So, if you are looking to use

Python to work as a freelancer, then check out the [Finxter Python Freelancer Course](https://blog.finxter.com/become-a-python-freelancer-course/) → <https://blog.finxter.com/become-a-python-freelancer-course/>. This provides the step by step path to go from nothing to earning a full-time income with Python in a few months, and gives you the tools to become a six-figure developer!

Programmer Humor



"Real programmers set the universal constants at the start such that the universe evolves to contain the data they want." — [xkcd](https://img0.xkcd.com/comics/real_programmers.png) → https://img0.xkcd.com/comics/real_programmers.png

Best 15+ Machine Learning Cheat Sheets to Pin to Your Toilet Wall – Be on the Right Side of Change

By Chris

Source: <https://blog.finxter.com/machine-learning-cheat-sheets/>

This article compiles for you the 15 best cheat sheets in the web that help you get started with machine learning. If you're short on time, here are the 15 direct PDF links (open in a new tab):

1. Supervised Learning → <https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/cheatsheet-supervised-learning.pdf> (Afshine Amidi)
2. Unsupervised Learning → <https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/cheatsheet-unsupervised-learning.pdf> (Afshine Amidi)
3. Deep Learning → <https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/cheatsheet-deep-learning.pdf> (Afshine Amidi)
4. Machine Learning Tips and Tricks → <https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/cheatsheet-machine-learning-tips-and-tricks.pdf> (Afshine Amidi)
5. Probabilities and Statistics → <https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/refresher-probabilities-statistics.pdf> (Afshine Amidi)
6. Linear Algebra and Calculus → <https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/refresher-algebra-calculus.pdf> (Afshine Amidi)
7. Comprehensive Stanford Master Cheat Sheet → <https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/super-cheatsheet-machine-learning.pdf> (Afshine Amidi)
8. Data Science Cheat Sheet → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PythonForDataScience.pdf (Datacamp)
9. Keras Cheat Sheet → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Keras_Cheat_Sheet_Python.pdf (Datacamp)
10. Deep Learning with Keras Cheat Sheet → <https://github.com/rstudio/cheatsheets/raw/master/keras.pdf> (RStudio)
11. Visual Guide to Neural Network Infrastructures → <http://www.asimovinstitute.org/wp-content/uploads/2016/09/neuralnetworks.png> (Asimov Institute)
12. Scikit-Learn Python Cheat Sheet → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Scikit_Learn_Cheat_Sheet_Python.pdf (Datacamp)
13. Scikit-learn Cheat Sheet: Choosing the Right Estimator → https://scikit-learn.org/stable/tutorial/machine_learning_map/ (Scikit-learn.org)
14. Tensorflow Cheat Sheet → <https://github.com/kailashahirwar/cheatsheets-ai/blob/master/PDFs/Tensorflow.pdf> (Altoros)
15. Machine Learning Test Cheat Sheet → <https://www.cheatography.com/lulu-0012/cheat-sheets/test-ml/pdf/> (Cheatography)

Each cheat sheet link points directly to the PDF file. So don't lose any more time, and start learning faster with these 15 ML cheat sheets.

In the following video, I quickly describe you all 15 cheat sheets and their pros and cons:

[Collection] 15 Mind-Blowing Machine Learning Cheat Sheets

(Article reading time: 12 minutes ||| Or watch the video)

Cheat sheets are the 80/20 principle applied to coding: learn 80% of the relevant material in 20% of the time.

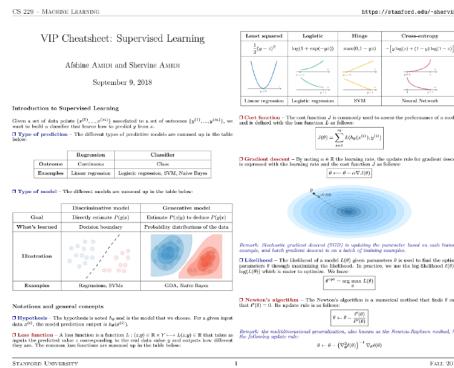
If you love learning with cheat sheets, [join my free cheat sheet academy → https://blog.finxtre.com/subscribe/](https://blog.finxtre.com/subscribe/):

This article compiles the list of all the best cheat sheets for machine learning. Are you a practitioner and want to move towards machine learning and data science? Are you a young data scientist just starting out with your career? Or are you a computer science student struggling to find a clear path of how to master the intimidating area of machine learning? Then check out these cheat sheets to make your life easier.

ALL LINKS OPEN IN A NEW TAB!

Supervised Learning → https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/cheatsheet-supervised-learning.pdf (Afshine Amidi)

This cheat sheet is the first part of a series of cheat sheets created for the Stanford Machine Learning Class. It gives you a short and concise **introduction to supervised learning**.



Topics include the following:

- Supervised learning notations,
- Linear regression,
- Classification,
- Logistic regression,
- Generalized linear models,
- Support vector machines,
- Generative learning,
- Gaussian discriminant analysis,
- Naive Bayes,
- Tree-based and ensemble methods, and
- General learning theory.

Unsupervised Learning → <https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/cheatsheet-unsupervised-learning.pdf> (Afshine Amidi)

This cheat sheet is the second part of the introductory series for the Stanford Machine Learning Class. It provides a concise **introduction to unsupervised learning**.

The VIP Cheatsheet: Unsupervised Learning is a comprehensive guide to unsupervised learning methods. It includes:

- Expectation-Maximization (EM):** A table showing the setting, latent variable, and comment for different EM variants:

Setting	Latent variable	Comment
Mixture of k Gaussians	$\text{Mult}(x_i \mu_j, \Sigma_j)$	$\mu_j \in \mathbb{R}^n, \Sigma_j \in \mathbb{R}^{n \times n}$
Factor analysis	$\text{Normal}(x_i \mu + Ax)$	$\mu \in \mathbb{R}^n, A \in \mathbb{R}^{n \times p}$
- K-means clustering:** A diagram illustrating the iterative steps: Initialize clusters, Assign points to nearest cluster, Compute centroid, and Convergence.
- Hierarchical clustering:** A diagram illustrating the iterative steps: Initialize clusters, Merge closest clusters, Compute distances, and Convergence.

You will learn about these topics:

- Expectation-maximization (EM),
- K-means clustering,
- Hierarchical clustering,
- Clustering assessment metrics,
- Principal component analysis, and
- Independent component analysis.

Deep Learning → <https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/cheatsheet-deep-learning.pdf> (Afshine Amidi)

This is the third part of the cheat sheet series provided by the Stanford Machine Learning Class. The cheat sheet is packed with dense information about deep learning. This cheat sheet offers a promising **kickstart into the hot topic of deep learning**.

The cheat sheet addresses topics such as

- Introduction to neural networks,
- Entropy,
- Convolutional neural networks,
- Recurrent neural networks,
- Reinforcement learning, and
- Control.

Of course, this covers only a subspace of the broad field of deep learning, but it will give you a short and effective start into this attractive area.

Machine Learning Tips and Tricks → [https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/cheatsheet-machine-learning-tips-and-tricks.pdf\(Afshine Amidi\)](https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/cheatsheet-machine-learning-tips-and-tricks.pdf)

The fourth part of the cheat sheet series provided as part of the Stanford Machine Learning Class promises **small tips and tricks in machine learning**. Although the author calls it that way (“Tips and Tricks”), I believe this is merely an understatement. In reality, this cheat sheet gives you valuable insights from a highly-skilled practitioner in the field.

CS 229 - MACHINE LEARNING <https://stanford.edu/~shervine>

VIP Cheatsheet: Machine Learning Tips
Afshine Amidi and Shervine Amidi
September 9, 2018

Metrics
Given a set of data points $\{x^{(1)}, \dots, x^{(m)}\}$, where each $x^{(i)}$ has a feature, associated to a set of outcomes $\{y^{(1)}, \dots, y^{(m)}\}$, let us say it is a prior classifier that learns how to predict a item.

Classification
In classification, there are two main metrics that are important to look for in order to evaluate the performance of the model:
TPR (True Positive Rate) = the probability of correctly identifying a positive sample while assessing the performance of a model. It is defined as follows:

Predicted class		
Actual class	TP	FP
	TN	FN

→ True Positives → False Positives → Type I error
→ False Negatives → True Negatives → Type II error

Main metrics – The following metrics are commonly used to assess the performance of a model:

Metric	Formula	Interpretation
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Overall performance of model
Precision	$\frac{TP}{TP + FP}$	How accurate the positive predictions are
Recall	$\frac{TP}{TP + FN}$	Coverage of actual positive sample
Sensitivity	$\frac{TP}{TP + FN}$	Coverage of actual negative sample
Specificity	$\frac{TN}{TN + FP}$	Coverage of actual negative sample
F1 score	$\frac{2 \cdot TP}{TP + FP + FN}$	Hybrid metric useful for imbalanced classes

ROC – The receiver operating curves, also noted ROC, is the plot of TPR versus FPR by varying the threshold τ .

Metric	Formula	Interpretation
True Positive Rate	$\frac{TP}{TP + FN}$	Recall
False Positive Rate	$\frac{FP}{FP + TN}$	1 - Specificity

AUC – The area under the receiver operating curve, also noted AUC or ROCUC, is the area below the ROC as shown in the following figure:

Ridgeless regression – Given a linear regression model f_θ , the following metrics are commonly used to assess the performance of the model:

Metric	Formula	Interpretation
Total sum of squares	$\sum_{i=1}^n (y_i - \bar{y})^2$	
Explained sum of squares	$\sum_{i=1}^n (\hat{y}_i - \bar{y})^2$	
Residual sum of squares	$\sum_{i=1}^n (y_i - \hat{y}_i)^2$	

Coefficient of determination – The coefficient of determination, often noted R^2 or r^2 , is the ratio of the explained sum of squares to the total sum of squares as explained by the model and is defined as follows:

$$R^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Model metrics – The following metrics are commonly used to assess the performance of a model:

Metric	Formula	Interpretation
Logistic C-P	$\frac{\log(1 + e^{-\lambda})}{\lambda}$	
AIC	$-2[\ln(\hat{y}) + \lambda \cdot \ln(k)]$	
BIC	$\log(n) + \lambda \cdot \ln(k)$	
Adjusted R ²	$1 - \frac{(1 - R^2)(n-1)}{n-k-1}$	

The topics are not only limited to

- Metrics,
- Classification,
- Regression,
- Model selection, and
- Diagnostics.

A must-read for upcoming data scientists.

Probabilities and Statistics → [https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/refresher-probabilities-statistics.pdf \(Afshine Amidi\)](https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/refresher-probabilities-statistics.pdf)

The fifth part of the cheat sheet series of the Stanford Machine Learning Class gives you a quick start (they call it a “refresher”) in the crucial area of **probability theory and statistics**. No matter in which field you will end up working, statistics will always help you on your path to becoming a machine learning professional. This refresher is definitely worth a read (and an investment of your printer ink).

CS 229 – MACHINE LEARNING
VIP Refresher: Probabilities and Statistics
Afshine Amidi and Shervine Amidi
August 6, 2018

Remark: for any event B in the sample space, we have $P(B) = \sum_i P(A_i)P(A_i|B)$.

3 Extended from Bayes' rule: Let $\{A_i\}_{i \in [n]}$ be a partition of the sample space. We have:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_j P(B|A_j)P(A_j)}$$

7 Independence: Two events A and B are independent if and only if we have: $P(A \cap B) = P(A)P(B)$

Introduction to Probability and Combinatorics

1 Sample space: The set of all possible outcomes of an experiment is known as the sample space.

2 Event: Any subset E of the sample space is known as an event. That is, as event is a set of outcomes. If E is an event, then \bar{E} is also an event. If E is an event, then $P(E)$ denotes the probability of event E , that is we say that E has occurred.

3 Permutation: A permutation is an arrangement of n objects from a pool of d objects, where the order does not matter. The number of such arrangements is given by $P(n, d)$, defined as:

$$P(n, d) = \frac{d!}{(d-n)!} = \frac{d(d-1)\dots(d-n+1)}{(d-n)!}$$

4 Combination: A combination is an arrangement of n objects from a pool of d objects, where the order does not matter. The number of such arrangements is given by $C(n, d)$, defined as:

$$C(n, d) = \frac{P(n, d)}{d!} = \frac{n!}{(n-d)!d!} = \frac{n(n-1)\dots(n-d+1)}{d!(n-d)!}$$

Remark: we note that for $0 \leq d \leq n$, we have $P(n, d) \geq C(n, d)$.

5 Conditional Probability: The events A and B such that $P(A \cap B) > 0$ are known as **conditional events**. We have:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Remark: we have $P(A|B) = P(A \cap B)/P(B) = P(A|B)/P(A|B) + P(A^c|B)$.

6 Partition: Let $\{A_i\}_{i \in [n]}$ be such that for all $i, A_i \neq \emptyset$. We say that $\{A_i\}_{i \in [n]}$ is a partition if we have:

$$\bigcup_{i \in [n]} A_i = \Omega \quad \text{and} \quad \bigcup_{i \in [n]} A_i = \emptyset$$

7 Bayes' rule: The events A and B such that $P(A \cap B) > 0$ are known as **Bayesian events**. We have:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_j P(B|A_j)P(A_j)}$$

8 Random Variables:

9 Random variable: A random variable, often called X , is a function that maps every element in a sample space to a real line.

10 Cumulative distribution function (CDF): The cumulative distribution function F_x which is monotonically non-decreasing and is such that $\lim_{x \rightarrow -\infty} F_x(x) = 0$ and $\lim_{x \rightarrow \infty} F_x(x) = 1$, is defined as:

$$F_x(x) = P(X \leq x)$$

Remark: we have $P(a < X \leq b) = F(b) - F(a)$.

11 Probability density function (PDF): The probability density function f_x is the probability that a random variable X takes a value in a small interval around x .

12 Related concepts involving the PDF and CDF: Here are the important properties to know about the relationship between the PDF and CDF:

Case	CDF F_x	PDF f_x	Properties of PDF
(I)	$F_x(x) = \sum_{a < x} P(X = a)$	$f_x(x) = P(X = x)$	$0 \leq f_x(x) \leq 1$ and $\int_{-\infty}^{\infty} f_x(x)dx = 1$
(II)	$F_x(x) = \int_{-\infty}^x f_x(s)ds$	$f_x(x) = \frac{dF}{dx}$	$F(x) \geq 0$ and $\int_{-\infty}^{\infty} f_x(x)dx = 1$

13 Variance: The variance of a random variable, often denoted $V(X)$ or σ^2 , is a measure of the spread of its distribution function. It is calculated as follows:

$$[V(X)] = E[(X - \mu)^2] = E[X^2] - \mu^2$$

Remark: the standard deviation of a random variable, often called σ , is a measure of the spread of the spread of the distribution function, which is compatible with the name of the standard normal variable. It is denoted as $\sigma = \sqrt{V(X)}$.

Here are the topics addressed in this cheat sheet:

- Introduction to probability and combinatorics,
- Conditional probability,
- Random variables,
- Joint distributions, and
- Parameter estimation.

Get this cheat sheet now!

Linear Algebra and Calculus → <https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/refresher-algebra-calculus.pdf> (Afshine Amidi)

Although the sixth part of the popular cheat sheet series of the Stanford Machine Learning Class does not sound too sexy, it teaches a fundamental area each machine learning professional knows well: **linear algebra**.

CS 229 - MACHINE LEARNING <https://stanford.edu/~shervine/>

VIP Refresher: Linear Algebra and Calculus
Afsiné Amidi and Shervine Amidi
October 6, 2018

General notations

• **Vector** - We note $x \in \mathbb{R}^n$ a vector with n entries, where $x_i \in \mathbb{R}$ is the i^{th} entry:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$$

• **Matrix** - We note $A \in \mathbb{R}^{m \times n}$ a matrix with m rows and n columns, where $A_{ij} \in \mathbb{R}$ is the entry located in the i^{th} row and j^{th} column:

$$A = \begin{pmatrix} A_{11} & \cdots & A_{1n} \\ A_{21} & \cdots & A_{2n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

• **Identity matrix** - The identity matrix $I \in \mathbb{R}^{n \times n}$ is a square matrix with ones in the diagonal and zeros everywhere else.

• **Inverse** - For all matrices $A \in \mathbb{R}^{n \times n}$, we have $A^{-1}A = I_n = A^{-1}$.

• **Determinant** - The determinant of a square matrix $A \in \mathbb{R}^{n \times n}$ is a scalar with nonzero values if its columns and rows are linearly independent.

• **Trace** - We also note $\text{tr}(A)$ as $\text{tr}(\text{diag}(A)) = \sum_{i=1}^n A_{ii}$.

• **Matrix operations**

• **Vector-matrix multiplication** - There are two types of vector-matrix products:

- **Inner product**: for $x \in \mathbb{R}^n$, we have

$$x^T g = \sum_{i=1}^n x_i g_i \in \mathbb{R}$$

• **Outer product**: for $a \in \mathbb{R}^m$, $b \in \mathbb{R}^n$, we have

$$ab^T \in \mathbb{R}^{m \times n} = \begin{pmatrix} a_1 b_1 & \cdots & a_1 b_n \\ a_2 b_1 & \cdots & a_2 b_n \\ \vdots & \ddots & \vdots \\ a_m b_1 & \cdots & a_m b_n \end{pmatrix} \in \mathbb{R}^{m \times n}$$

• **Matrix-vector multiplication** - The product of matrix $A \in \mathbb{R}^{m \times n}$ and vector $x \in \mathbb{R}^n$ is a vector of size \mathbb{R}^m , such that

$$Ax = \begin{pmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_m^T x \end{pmatrix} \in \mathbb{R}^m$$

where a_i^T are the vector rows and a_i are the vector columns of A , and a_i are the entries of A .

• **Matrix-matrix multiplication** - The product of matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$ is a matrix of size $\mathbb{R}^{m \times p}$, such that

$$AB = \begin{pmatrix} a_1^T b_1 & \cdots & a_1^T b_p \\ a_2^T b_1 & \cdots & a_2^T b_p \\ \vdots & \ddots & \vdots \\ a_m^T b_1 & \cdots & a_m^T b_p \end{pmatrix} \sum_{i=1}^m \sum_{j=1}^p a_i b_j \in \mathbb{R}^{m \times p}$$

where $a_i^T b_j$ are the vector rows and a_i, b_j are the vector columns of A and B respectively.

• **Transpose** - The transpose of a matrix $A \in \mathbb{R}^{m \times n}$, noted A^T , is such that its entries are swapped:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}^T = A^T$$

Remark: for matrices A, B , we have $(AB)^T = B^T A^T$.

• **Inverse** - The inverse of an invertible square matrix A is noted A^{-1} and is the only matrix such that

$$AA^{-1} = A^{-1}A = I_n$$

Remark: not all square matrices are invertible. Also, for matrices A, B , we have $(AB)^{-1} = B^{-1}A^{-1}$.

• **Trace** - The trace of a square matrix A , noted $\text{tr}(A)$, is the sum of the diagonal entries.

$$\text{tr}(A) = \sum_{i=1}^n A_{ii}$$

Remark: for matrices A, B , we have $\text{tr}(AB) = \text{tr}(BA) = \text{tr}(A)B$.

• **Determinant** - The determinant of a square matrix $A \in \mathbb{R}^{n \times n}$, noted $\det(A)$ or $|A|$, is expressed in terms of $A_{i,j}$, which is the matrix A without i^{th} row and j^{th} column.

Do you struggle understanding this critical topic? Your lack of understanding will cost you weeks as soon as you start implementing practical machine learning algorithms. Simply put: you have to master linear algebra, there is no way around. So do it now and do it well.

What are the precise topics included in this cheat sheet?

- Standard matrix notation,
- Matrix operations,
- Matrix properties, and
- Matrix calculus (gradient operations).

You see, it's all about matrices. Before you even consider diving in practical libraries used in machine learning (such as Python's numpy, check out my [HUGE numpy tutorial](https://blog.finxtre.com/numpy-tutorial) → <https://blog.finxtre.com/numpy-tutorial>), study this cheat sheet first.

Comprehensive Stanford Master Cheat Sheet → <https://github.com/afshinea/stanford-cs-229-machine-learning/blob/master/en/super-cheatsheet-machine-learning.pdf> (Afshine Amidi)

This cheat sheet comprises six cheat sheets of the Stanford Machine Learning Class. It is an awesome resource, packed with information in many important subfields in Machine Learning. I highly recommend downloading this resource and studying it a whole day. It will boost your machine learning skills in little time.

Super VIP CheatSheet: Machine Learning	
Afshine Ashtari and Shervine Ashtari	10
October 6, 2018	10
Machine Learning Tips and Tricks	
4.1 Metrics	10
4.1.1 Classification	10
4.1.2 Regression	10
4.2 Model selection	11
4.3 Decision trees	11
References	
5.1 Probabilities and Statistics	12
5.1.1 Introduction to Probability and Combinatorics	12
5.1.2 Conditional Probability	12
5.1.3 Jointly Distributed Random Variables	13
5.2 Linear Algebra and Calculus	13
5.2.1 Linear Algebra	14
5.2.2 Matrix operations	14
5.2.3 Matrix properties	15
5.2.4 Determinants	16
Contents	
1 Supervised Learning	1
1.1 Introduction to Supervised Learning	1
1.2 Notation and general concepts	1
1.3 Linear models	1
1.3.1 Linear regression	1
1.3.2 Logistic regression	1
1.3.3 Generalized Linear Models	1
1.4 Tree-based methods	1
1.4.1 Generative Learning	1
1.4.2 Gaussian Discriminant Analysis	1
1.4.3 Naive Bayes	1
1.5 One-handy and ensemble methods	1
1.5.1 Random Forest	1
1.5.2 Boosting	1
1.5.3 Gradient boosting	1
1.5.4 AdaBoost	1
1.5.5 XGBoost	1
1.5.6 LightGBM	1
1.5.7 CatBoost	1
1.6 Loss Function	5
2 Unsupervised Learning	6
2.1 Clustering	6
2.2 Expectation–maximization	6
2.2.1 K-means clustering	6
2.2.2 Hierarchical clustering	6
2.2.3 Dimensionality reduction	6
2.3 Dimension reduction	7
2.3.1 Principal component analysis	7
2.3.2 Independent component analysis	7
3 Deep Learning	8
3.1 Deep Networks	8
3.2 Convolutional Neural Networks	8
3.3 Recurrent Neural Networks	8
3.4 Reinforcement Learning and Control	8

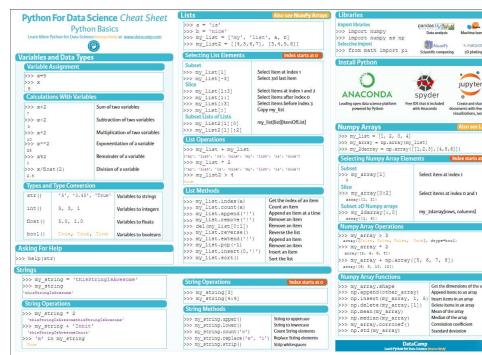
The widely distributed topics of this 16-page cheat sheet include

- Supervised Learning,
- Unsupervised learning,
- Deep learning,
- Machine learning tips and tricks,
- Probabilities and statistics, and
- Linear algebra and calculus.

Don't lose any more time reading the rest of this article and download this cheat sheet. Thanks, Afshine, for this awesome resource!

Data Science Cheat Sheet → https://s3.amazonaws.com/assets.datacamp.co/m/blog_assets/PythonForDataScience.pdf (Datacamp)

The datacamp cheat sheets are always worth a look. However, I would recommend this cheat sheet only for absolute beginners in the field of data science. If you focus on learning core machine learning concepts and you already have some experience, please skip this cheat sheet. But if you are just starting out with data science and machine learning – and you want to use **Python** as your programming language – this 1-page **data science** cheat sheet is for you.



The basic topics of this cheat sheet are

- Installing Python,
- Python variables and data types,
- Strings and string operations,
- Lists and list methods, and
- Basic numpy functionality (numpy is the Python library for basic linear algebra and matrix operations).

Keras Cheat Sheet → https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Keras_Cheat_Sheet_Python.pdf (Datacamp)

This 1-page cheat sheet is worth your time if you are looking into the specialized machine learning tool Keras. I have not yet used Keras myself but it is considered to be the best abstraction layer for deep learning and neural networks.



Wikipedia → <https://en.wikipedia.org/wiki/Keras> defines Keras as follows.

“Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible”.

With such a broad applicability, I am so convinced, I will check out Keras after finishing this blog post. Will you, too?

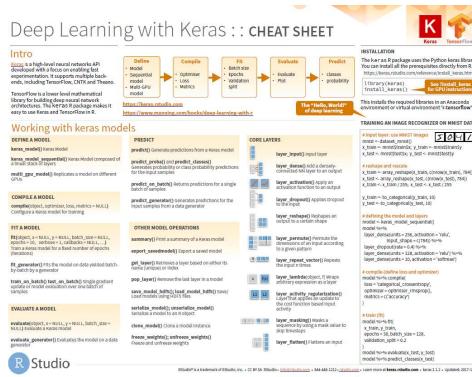
If you’re interested in Keras, feel free to watch this video and read the [associated blog article on the income levels of Keras developers](#) → <https://blog.finxter.com/keras-developer-income-and-opportunity/>:

From Keras to \$65/h+ on Upwork... Juan

The Keras Cheat Sheet addresses the following points (from a code-centric perspective).

- Basic usage,
- Data and data structures,
- Preprocessing,
- Multilayer perceptron,
- Convolutional neural networks,
- Recurrent neural networks, and
- Model training, inference, & fine-tuning.

Deep Learning with Keras Cheat Sheet → <https://github.com/rstudio/cheat-sheets/raw/master/keras.pdf> (RStudio)

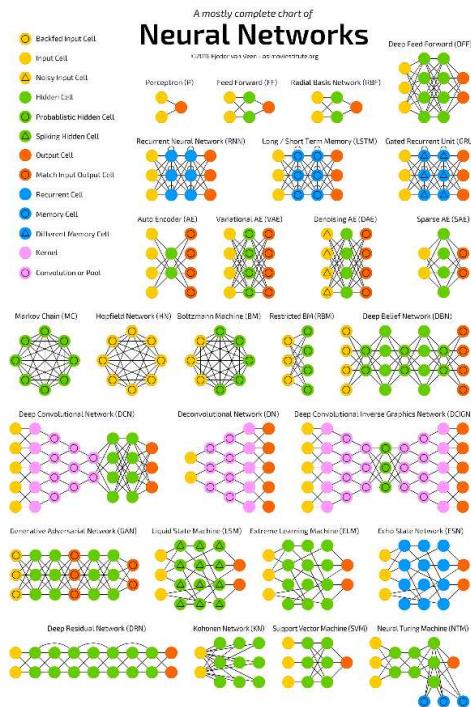


Simply put: I love this cheat sheet. It's about deep learning with the open-source neural network library Keras. It is visual, to the point, comprehensive, and understandable. I highly recommend checking out this cheat sheet!

- The 2-page cheat sheet gives you a quick overview of the Keras pipeline for deep learning.
- It shows you how to work with models (e.g. definition, training, prediction, fitting, and evaluation).
- Furthermore, it gives you a visual overview of how to access the diverse layers in the neural network.
- Finally, it provides a short but insightful example of the standard demo problem of handwriting recognition.

Visual Guide to Neural Network Infrastructures → <http://www.asimovinstitute.org/wp-content/uploads/2016/09/neuralnetworks.png> (Asimov Institute)

This 1-page visual guide gives you a quick overview of all the most common **neural network infrastructures** that you will find in the wild. The sheet showcases 27 different architectures. As a machine learning newbie, you will not get much out of this sheet. However, if you are a practitioner in the field of neural networks, you will like it.



The cheat sheet shows 27 neural network architectures including

- Perceptron,
- Feedforward, Radial basis network, Deep feedforward,
- Recurrent neural network, long / short term memory (LSTM), gated recurrent unit,
- Autoencoder, variational autoencoder, denoising autoencoder, sparse autoencoder,
- Markov chain, Hopfield network,
- Boltzmann machine, restricted Boltzmann machine, deep belief network, and
- Finally, deep convolutional network, deconvolutional network, deep convolutional inverse graphics network, generative adversarial network, liquid state machine, extreme learning machine, echo state network, deep residual network, kohonen network, support vector machine, and neural turing machine.

Pheww, what a list!

Scikit-Learn Python Cheat Sheet → https://s3.amazonaws.com/assets.datcamp.com/blog_assets/Scikit_Learn_Cheat_Sheet_Python.pdf (Datacamp)

Another 1-page PDF cheat sheet that gives you a headstart in **Python's library for machine learning scikit-learn → <https://scikit-learn.org/stable/>**. This library is the best single-CPU, general-purpose libraries for machine learning in Python. Python is the most popular programming language in the field of machine learning, so this cheat sheet gives you a lot of value. Get this cheat sheet if you use Python for machine learning.



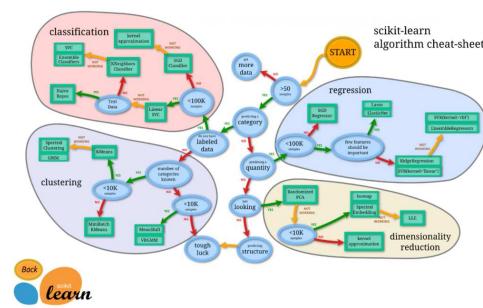
The topics include

- Basic functionality such as loading and preprocessing the training data,
- Creating the model,
- Model fitting,
- Prediction and inference, and
- Evaluation metrics such as classification metrics, regression metrics, clustering metrics, cross-validation, and model tuning.

Be warned that these concepts are not explained in detail. It only shows how to use them in the scikit-learn library.

Scikit-learn Cheat Sheet: Choosing the Right Estimator → [https://scikit-learn.org/stable/tutorial/machine_learning_map/ \(Scikit-learn.org\)](https://scikit-learn.org/stable/tutorial/machine_learning_map/)

This cheat sheet is so valuable – I cannot even describe it in words. Thanks, scikit-learn creators, for posting this awesome piece of art!



It helps you figure out which algorithm to use for which kind of problem. You simply follow the questions in the cheat sheet. As a result, you will reach the recommended algorithm for your problem at hand. This is why I love cheat sheets – they can deliver complex information in little time.

The cheat sheet divides the estimators into four classes:

- Classification,

- Clustering,
- Regression, and
- Dimensionality reduction.

Although those classes are not explored in depth, you will already know in which direction to look further. Of course, if you are already an experienced practitioner, the provided information may be too simplistic – but isn't this true for every cheat sheet?

Build your own opinion now! (Do it.)

Tensorflow Cheat Sheet → <https://github.com/kailashahirwar/cheatsheets-ai/blob/master/PDFs/Tensorflow.pdf> (Altoras)

Although this cheat sheet is not the most sophisticated one, it is still valuable being one of the few TensorFlow cheat sheets out there.



You know TensorFlow, don't you? TensorFlow is one of the most popular Github projects and it's created by Google. Its machine learning API is tailored to deep learning on a heterogeneous computing environment (including GPUs). Nowadays, if you push in the field of deep learning, there is no way you can avoid TensorFlow.

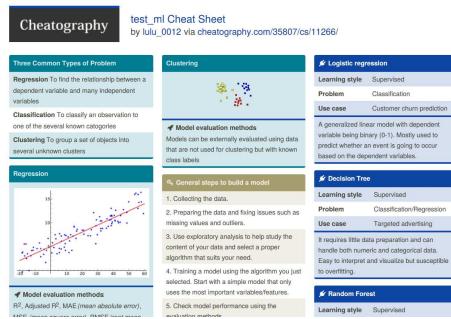
Get a first impression with this cheat sheet and then [dive into Google's TensorFlow system](https://www.tensorflow.org/) → <https://www.tensorflow.org/>. By the way, you can also use Keras on top of TensorFlow as a more high-level abstraction layer. Check out the Keras cheat sheet described earlier.

The cheat sheet gives you hints about

- The correct installation method,
- Helper functions,
- The name of some important functions in TensorFlow, and
- Estimators.

To be frank, I would not recommend learning TensorFlow with this cheat sheet. Why? Because it is not focused on education. Yet, I felt obliged to include the link because there are no better alternatives for TensorFlow. If you know a better resource, please let me know.

Machine Learning Test Cheat Sheet → <https://www.cheatography.com/lulu-0012/cheat-sheets/test-ml/pdf/> (Cheatography)



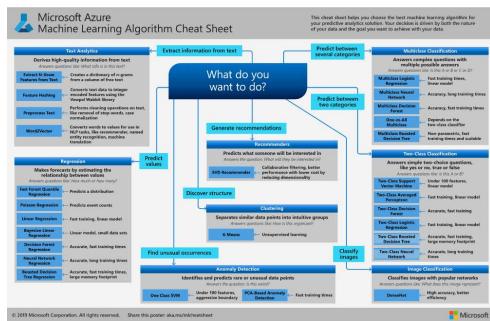
Do you know cheatography? It's like Wikipedia for cheat sheets. Everybody can submit cheat sheets (user-generated content).

After going through most machine learning cheat sheets at Cheatography, I found that this one will be most helpful for most of our readers. It is a well-structured overview of some important machine learning algorithms.

- It shows you that there are three common problems in machine learning: regression, clustering, and classification.
- It gives you the general steps for training a model.
- Finally, it glances over a collection of specific algorithms that you should know when starting out in the field of machine learning. Those are logistic regression, decision tree, random forest, k-means, naive Bayes, k nearest neighbors, and support vector machines.

I know that it is only a first dip into the ocean. But if you are a beginner or intermediate machine learning practitioner, this may just be what you have looked for.

Microsoft's Machine Learning Algorithm Cheat Sheet → <https://blog.fintecher.com/machine-learning-cheat-sheets/16.%20Microsoft's%20Machine%20Learning%20Algorithm%20Cheat%20Sheet> (Azure)



This excellent cheat sheet provides you a quick overview of the most important algorithms and how they are interrelated. It's a great way to gain an overview of the field of artificial intelligence and machine learning.

Did you enjoy this collection of the best machine learning cheat sheets on the web? I recommend to download all sheets, print them and work through each of them. This will give you a first overview of the field of machine learning. Later, you can decide in which area to dive in further.

Bonus: Many hot machine learning systems (e.g. TensorFlow) require excellent Python programming skills. Do you know all the features, tips, and tricks of Python? If not, I recommend to check out [this free Python cheat sheet email course](https://blog.finxter.com/subscribe/) → <https://blog.finxter.com/subscribe/>.

The email course will not only provide you with 5 Python cheat sheets (80% of the learning in 20% of the time, remember?) but also with a constant stream of Python programming lectures. It's 100% free, you can unsubscribe at any time, and I will not spam you. It's pure value (and occasionally I will send you information about my books and courses). So check it out!

Best Python Cheat Sheet → <https://blog.finxter.com/collection-5-cheat-sheet-s-every-python-coder-must-own/>

Python is at the core of machine learning today. It has the best library support for machine learning among all programming languages. So, to become a better ML engineer, you may need to study Python. What better way than to download a cheat sheet PDF?

NumPy Cheat Sheets: Tips and Tricks

What's the Meaning of Axis and Shape properties?

1D NumPy Array	2D NumPy Array	3D NumPy Array
<code>a.ndim = 1</code> <code>a.shape = [5,]</code> <code>a.size = 5</code>	<code>a.ndim = 2</code> <code>a.shape = [5, 4]</code> <code>a.size = 20</code>	<code>a.ndim = 3</code> <code>a.shape = [5, 4, 3]</code> <code>a.size = 60</code>

What's Broadcasting?

Goal: bring arrays with different shapes into the same shape during arithmetic operations. NumPy does that for you!

```
import numpy as np
salary=np.array([2000, 4000, 6000])
salaries=salary*1000
print(salary + salary * 1000)
# [2000 4000 6000]
```

- For any dimension where the first axis has size of one, NumPy conceptually copies its data until the size of the second axis is reached.
- If nothing is completely missing for array B, it is simply copied along the missing dimension.

When Boolean Indexing

Goal: create a new array that contains a fine-grained element selection of the old array

```
import numpy as np
a=np.array([[1,2],[3,4],[5,6]])
indices=np.array([[False, False, True],
                  [False, False, False],
                  [True, False, False]])
print(a[indices])
#[[3, 4]

```

We create two arrays "a" and "index". The first array contains two-dimensional numerical data (3x2 array). The second array contains Boolean values (3x3 array) (Boolean array). You can use this indexing array for fine-grained selection of elements from the first array. We select the data array containing only those elements for which the indexing array contains "True". Boolean values at which the indexing array contains "True" Boolean values at which the respective array position, therefore, the resulting array will contain the three values 3, 4, and 5.

How to Search Arrays? The np.nonzero() Trick

Goal: find elements that meet a certain condition in a NumPy array

- Step 1: Understanding np.nonzero()

```
import numpy as np
X=np.array([[1,0,0],[0,1,0],[0,0,1]])
print(np.nonzero(X))
#(array([0, 1, 2], dtype=int64), array([0, 1, 0], dtype=int64))
```

The result is a tuple of two NumPy arrays. The first array gives the row indices of non-zero elements, the second array gives the column indices of non-zero elements.

- Step 2: Use np.nonzero() and broadcasting to find elements

```
import numpy as np
# Data: air quality index(AQI) data (row x city)
X=np.array([
    [1, 2, 3, 4, 5, 6], # Hong Kong
    [3, 2, 1, 2, 3, 4], # New York
    [5, 1, 3, 1, 1, 5], # Berlin
    [1, 1, 1, 1, 1, 1], # Montreal
])
cities=['HK','NY','Berlin']
# Find cities with above average pollution
polluted=np.all(np.nonzero(X > np.average(X)))
print(polluted)
```

The Boolean expression "`X > np.average(X)`" uses broadcasting to bring both together to the same shape. Then it performs an element-wise comparison. If the respective measurement contains "True" if the respective measurement observed an average value or higher. The np.all() function then applies the average AQI value over all NumPy array elements. Boolean indexing accesses all city rows with above average pollution values.

A Puzzle A Day to Learn, Code, and Play!

(Click to download PDF) → <https://blog.finxter.com/wp-content/uploads/2019/04/11-Numpy-Cheat-Sheet.pdf>

NumPy → <https://docs.scipy.org/> is a widely used Python scientific computing package. It simplifies linear algebra, matrix computations, and speeds up data analysis. Knowing NumPy is a prerequisite for other Python packages like pandas or Scikit-Learn.

Best Scikit-Learn Cheat Sheet → <https://blog.finxter.com/scikit-learn-cheat-sheets/>

Python For Data Science Cheat Sheet

Scikit-Learn

Scikit-Learn is an open source Python library that implements a variety of machine learning, regression, and classification algorithms and methods for data mining and data analysis.

Create Your Model

Supervised Learning Estimators

```
from sklearn import datasets, linear_model
# Load dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
# Create linear regression object
regr = linear_model.LinearRegression()
# Train the model using the training sets
regr.fit(X, y)
# Make predictions using the testing set
y_pred = regr.predict(X)
# Print out the coefficients
print(regr.coef_)
```

Unsupervised Learning Estimators

```
# Import K-Means clustering
from sklearn.cluster import KMeans
# Create K-Means clustering object
kmeans = KMeans(n_clusters=3)
# Fit the model to data
kmeans.fit(X)
# Print out cluster centers
print(kmeans.cluster_centers_)
```

Model Fitting

Supervised learning

```
# Fit the model to the data
regr.fit(X, y)
# Print out the coefficient
print(regr.coef_)
```

Prediction

Supervised Estimators

```
# Predict labels for X
y_hat = regr.predict(X)
# Print out the coefficient
print(y_hat)
```

Preprocessing The Data

You don't need to reshape and convert Numpy arrays to be fully sparse matrices. You can use the built-in functions in Dataframe, and also available:

```
# Fit StandardScaler to data
scaler = StandardScaler()
scaler.fit(X)
# Transform data
X_scaled = scaler.transform(X)
```

Standardization

```
# Fit StandardScaler to data
scaler = StandardScaler()
scaler.fit(X)
# Transform data
X_scaled = scaler.transform(X)
```

Normalization

```
# Fit MinMaxScaler to data
scaler = MinMaxScaler()
scaler.fit(X)
# Transform data
X_normalized = scaler.transform(X)
```

Imputation

```
# Fit SimpleImputer to data
imputer = SimpleImputer(strategy='mean')
imputer.fit(X)
# Transform data
X_imputed = imputer.transform(X)
```

Encoding

Encoding Categorical Features

```
# Fit OneHotEncoder to data
encoder = OneHotEncoder()
encoder.fit(X)
# Transform data
X_oh = encoder.transform(X)
```

Encoding Numerical Features

```
# Fit StandardScaler to data
scaler = StandardScaler()
scaler.fit(X)
# Transform data
X_scaled = scaler.transform(X)
```

Encoding Missing Values

```
# Fit SimpleImputer to data
imputer = SimpleImputer(strategy='mean')
imputer.fit(X)
# Transform data
X_imputed = imputer.transform(X)
```

Generating Polynomial Features

```
# Fit PolynomialFeatures to data
poly = PolynomialFeatures(degree=2)
poly.fit(X)
# Transform data
X_poly = poly.transform(X)
```

Evaluate Your Model's Performance

Classification Metrics

```
# Accuracy Score
accuracy_score(y_true, y_pred)
# Confusion Matrix
confusion_matrix(y_true, y_pred)
# Precision-Recall Curve
precision_recall_curve(y_true, y_scores)
# ROC-AUC Score
roc_auc_score(y_true, y_scores)
```

Regression Metrics

```
# Root Mean Squared Error
mean_squared_error(y_true, y_pred)
# R2 Score
r2_score(y_true, y_pred)
```

Classification Thresholding

Adjusting Threshold

```
# Fit DecisionBoundaryVisualizer to data
visualizer = DecisionBoundaryVisualizer()
visualizer.fit(X, y)
# Plot decision boundary
visualizer.show()
```

Tune Your Model

Grid Search

```
# Fit GridSearchCV to data
grid_search = GridSearchCV(estimator=RandomForestClassifier(),
                          param_grid=param_grid,
                          cv=cv)
grid_search.fit(X, y)
```

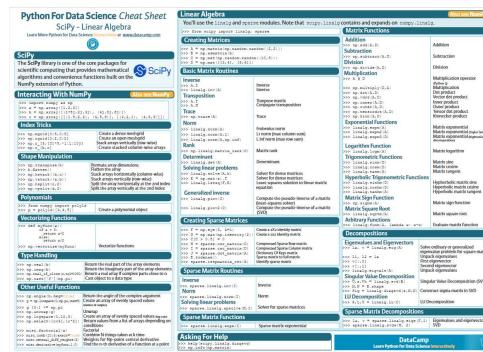
Constrained Parameter Optimization

```
# Fit ConstrainedGridSearchCV to data
grid_search = ConstrainedGridSearchCV(
    estimator=RandomForestClassifier(),
    param_grid=param_grid,
    constraints=constraints,
    cv=cv)
grid_search.fit(X, y)
```

DataCamp

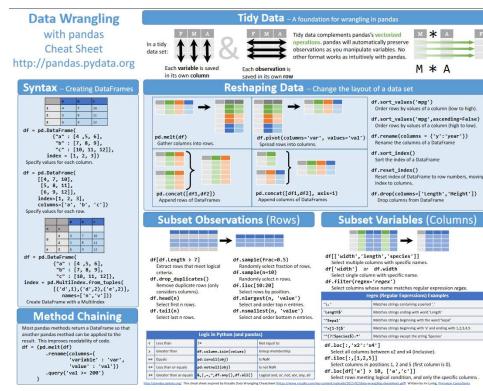
This Scikit-Learn cheat sheet from DataCamp will kick start your data science project by introducing you to the basic concepts of machine learning algorithms successfully. This cheat sheet is for those who have already started to learn Python packages and for those who would like to take a quick look to get a first idea of the basics for total beginners!

Best Scipy Cheat Sheet → <https://blog.finxtre.com/best-10-scipy-cheat-sheet/>



The cheat sheet is from DataCamp.com and is chock full of information for you to consume. You will learn to interact with Numpy and know which functions and methods to use for linear algebra and of course a help section. This is one I would hang behind my monitor behind the wall!

Best Pandas Cheat Sheet → <https://blog.finxtre.com/pandas-cheat-sheets/>



This one is from the pandas guys, so it makes sense that this is a comprehensive and inclusive cheat sheet. It covers the vast majority of what most pandas users will ever need to do to a DataFrame. Have you already used pandas for a little while? And are you looking to up your game? This is your cheat sheet! However, if you are newer to pandas and this cheat sheet is a bit overwhelming, don't worry! You definitely don't need to understand everything in this cheat sheet to get started.