

# References for some of the Software Carpentry lessons

**Summary:** A list of Software Carpentry related free online resources that I have compiled along time. Online repository for this document <https://github.com/scw-ss/references>

## Contents

<b>bash</b>	<b>1</b>
<b>Git</b>	<b>2</b>
<a href="#">Using Git with LaTeX</a>	2
<a href="#">Making your code citable</a>	2
<b>Python</b>	<b>2</b>
<a href="#">Anaconda</a>	2
<a href="#">general references / guides</a>	3
<a href="#">Scientific Python</a>	3
<a href="#">pandas</a>	3
<a href="#">Data Science / Analysis</a>	3
<a href="#">IPython - Jupyter</a>	4
<a href="#">Python for Matlab users</a>	4
<a href="#">Graphical User Interface (GUI) application development tutorials</a>	4
<a href="#">Integrated Developmen Enviroments - IDEs</a>	4
<b>HPC</b>	<b>4</b>

## bash

For a tutorial like guide to bash <http://www.bash.academy/>

Also this interactive, very short lessons, bash tutorial is an easy way to get up your sleeves: [http://rik.smith-unna.com/command\\_line\\_bootcamp/](http://rik.smith-unna.com/command_line_bootcamp/)

Once you get more advanced, or just for reference or looking for specific things, the "Advanced Bash-Scripting Guide" is a very useful resource (and it comes first in google many times when searching for bash related subjects)

<http://tldp.org/LDP/abs/html/>

For an interactive explanation of bash commands see <https://explainshell.com/>. Try any of the classroom examples we did, or just the examples you can find there.

### **Note**

First, should you be using Windows, I do recommend take a look at MobaXterm, and "enhanced terminal for Windows with X11 server, tabbed SSH client, network tools and much more" <https://mobaxterm.mobatek.net/> .

It may seem a bit overwhelming with so many functionalities at first, but is incredibly useful, specially if you are connecting to remote machines. It has a "plugin system" with many prepackaged tools (Git, compilers, ...) and a "package manager" which allows you to install any Cygwin project <https://cygwin.com> linux utilities under windows.

Is closed source software but has a free personal version more than enough for most of a typical usage.

## Git

A good Mercurial introduction <http://hginit.com/index.html> . Mercurial is "the other" Distributed Version Control System, and its syntax is very similar to Git. All the concepts apply to both, and it has good basic insights on how Mercurial (and Git) works.

Later on, the "Pro Git book" has anything you need as a reference for Git: from basic commands, up to how setting up your own central Git repository server with ssh access (or locally) in a few lines <https://git-scm.com/book>

CodeAcademy free TryGit online tutorial, created by GitHub and useful for reviewing the course syllabus, and a bit more <https://try.github.io/levels/1/challenges/1>

Branching explained in an Interactive tutorial <https://learnitbranching.js.org/>

For a list of Git graphic user interface (GUI) clients see <https://git-scm.com/downloads/guis>

## Using Git with LaTeX

See some very good tips / guidelines in "Advice for writing LaTeX documents" <https://github.com/dspinelis/latex-advice>

Also take a look at `git-latexdiff` tool <https://gitlab.com/git-latexdiff/git-latexdiff>

For adding Git commit data inside the LaTeX document you can use the `gitinfo2` package <https://www.ctan.org/pkg/gitinfo2> aa:w

## Making your code citable

AKA assigning a DOI (Digital Object Identifier) to a Github release.

You can assign a DOI to a Github release of your project so that you can reference it in your papers etc. This is done through Zenodo (<https://zenodo.org/>) integration with Github, and it's a extremely simple process that is described here: <https://guides.github.com/activities/citable-code/>

## Python

### Anaconda

Learn how to decide which Anaconda / miniconda version you want, and how Python virtual environment work and why to use them <https://medium.freecodecamp.org/why-you-need-python-environments-and-how-to-manage-them-with-conda-85f155f4353c>

## general references / guides

If you want to jump straight into Python from zero, even if you know other programming languages, I strongly recommend "Python for Everybody", an evolution from "Think Python". A really clear approach and very easy to follow, it also gives the most affordable approach to Object Oriented Programming I've ever seen. It's free, but you can buy a hard copy really cheap. Also the author has a very good Coursera track based on the book.

<https://www.py4e.com/book.php>

Also, "The Hitchhiker's Guide to Python" <http://docs.python-guide.org/en/latest/> is a very good all-around Python reference and study guide.

"A Whirlwind Tour of Python" is a fast-paced introduction to essential components of the Python language for researchers and developers who are already familiar with programming in another language. Book and Jupyter notebooks.

<https://github.com/jakevdp/WhirlwindTourOfPython>

A quite extensive community list of resources for the Python avid ones. From the very basic for people new to programming, to specific occupations as "Python for the Humanities" references: <https://www.fullstackpython.com/best-python-resources.html>

The Real Python webpage <https://realpython.com/> has a long list of very good tutorials for all levels and covering many different subjects which are worth taking a look. They publish new tutorials often. One of them is "The Best Python Books" <https://realpython.com/best-python-books/>.

A curated collection of Jupyter/IPython notebooks that are notable, from the Jupyter project GitHub page itself <https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks>

## Scientific Python

For a more scientific oriented goal, perhaps for people with some at least basic python knowledge, the "Scipy Lecture Notes" are a great resource. With lessons on how to interface Python with C and Fortran, and other more advanced topics.

<http://www.scipy-lectures.org/>

Notebooks from J.R. Johansson, "Scientific Computing with Python", cover a wide variety of subjects, even some advanced ones as HPC or interfacing Python with C and Fortran

<https://github.com/jrjohansson/scientific-python-lectures>

For those of you using Matlab, see below the section "Python for Matlab users"

## pandas

"Pandas in a nutshell" notebook <http://kanoki.org/2017/07/16/pandas-in-a-nutshell/>

A Beginner's Guide to Optimizing Pandas Code for Speed

<https://engineering.upside.com/a-beginners-guide-to-optimizing-pandas-code-for-speed-c09ef2c6a4d6?gi=789797286edf>

## Data Science / Analysis

Python Data Science Book, from Jake VanderPlas

<https://jakevdp.github.io/PythonDataScienceHandbook/>

Data science Python notebooks: Deep learning (TensorFlow, Theano, Caffe, Keras), scikit-learn, Kaggle, big data (Spark, Hadoop MapReduce, HDFS), matplotlib, pandas, NumPy, SciPy, Python essentials, AWS, and various command lines.

<https://github.com/donnemartin/data-science-ipython-notebooks>

## IPython - Jupyter

Slideshow about IPython and Jupyter, 34 slides, very good

<http://eueung.github.io/python/ipython-intro>

28 Jupyter Notebook tips, tricks and shortcuts

<http://www.pybloggers.com/2016/10/28-jupyter-notebook-tips-tricks-and-shortcuts>

Building Interactive Dashboards with Jupyter

<https://blog.dominodatalab.com/interactive-dashboards-in-jupyter/>

IPython Interactive Computing and Visualization Cookbook, Second Edition (2018), by Cyrille Rossant, contains over 100 hands-on recipes on high-performance numerical computing and data science in the Jupyter Notebook. <https://ipython-books.github.io/>

IPython Interactive Computing and Visualization Cookbook, Second Edition (2018), by Cyrille Rossant, contains over 100 hands-on recipes on high-performance numerical computing and data science in the Jupyter Notebook. Most of the book is freely available on this website (CC-BY-NC-ND license) <https://ipython-books.github.io/>

## Python for Matlab users

If you are transitioning from MATLAB to Python check the White Paper "MATLAB to Python: A Migration Guide", from ENTHOUGHT, the creators of the first scientific Python distribution (now overpassed by Anaconda):

<https://www.enthought.com/wp-content/uploads/Enthought-MATLAB-to-Python-White-Paper.pdf> , and the related online webinar at <https://www.youtube.com/watch?v=YkCegitoHFQ>

See also RealPython "MATLAB vs Python: Why and How yo Make the Switch" at <https://realpython.com/matlab-vs-python/>

The "Mathesaurus" has a "NumPy for MATLAB users" table of equivalences <http://mathesaurus.sourceforge.net/>

## Graphical User Interface (GUI) application development tutorials

Real Python'2 "Python and PyQt: Buidling a GUI Desktop Calculator" <https://realpython.com/python-pyqt-gui-calculator/>

## Integrated Developmen Enviroments - IDEs

The most popular IDEs for Python are

- Pycharm: proprietary software with a good-enough community version, specifically designed for Python development.
- Microsoft's Visual Studio Code: open source general IDE, with goog Python support through its plugins system.
- Atom and SublimeText: text editors which can become kind of IDEs by means of pluggins and addons.
- VIM (I had to do it :-)

By the moment there is no R-Studio equivalent.

## HPC

SGE - SLURM migration guide / table

<https://confluence.csiro.au/display/SC/Reference+Guide%3A+Migrating+from+SGE+to+SLURM>