References

If you can know everything about anything, it is not worth knowing.
—Robertson Davies

Starting from the foundation in this tutorial, you may be able to get the advanced material you need for a particular problem just from web searches. Some other references that we found helpful appear below.

As your skills develop, you may start writing longer and more complex codes. Then it will pay off to make an additional investment in learning about basic software engineering practices (for example, in Scopatz & Huff, 2015) and specifically about user-defined classes (for example, in Downey, 2012; Guttag, 2016; Scopatz & Huff, 2015). The PEP 8 Style guide for Python also provides guidelines for writing Python code in a standard, readable way: `www.python.org/dev/peps/pep-0008/`.

This book has alluded to the free LaTeX typesetting system, in the context of graph labels and Jupyter notebooks. You can obtain it, and its documentation, from `www.latex-project.org/get/`.

A blog accompanies this book. It can be accessed via `press.princeton.edu/titles/11349.html`, or directly at `physicalmodelingwithpython.blogspot.com/`. Here you will find data sets, code samples, errata, additional resources, and extended discussions of the topics introduced in this book.

A counterpart to this tutorial covers similar techniques, but with the MATLAB programming language (Nelson, 2015).

Berendsen, H J C. 2011. *A student's guide to data and error analysis.* Cambridge UK: Cambridge Univ. Press.

Cromey, D W. 2010. Avoiding twisted pixels: Ethical guidelines for the appropriate use and manipulation of scientific digital images. *Sci. Eng. Ethics*, **16**(4), 639–667.

Downey, A. 2012. *Think Python.* Sebastopol CA: O'Reilly Media Inc. `www.greenteapress.com/thinkpython/thinkpython.pdf`.

Guttag, J V. 2016. *Introduction to computation and programming using Python.* 2d ed. Cambridge MA: MIT Press.

Hill, C. 2015. *Learning scientific programming with Python.* Cambridge UK: Cambridge Univ. Press.

Ivezic, Ž, Connolly, A J, VanderPlas, J T, & Gray, A. 2014. *Statistics, data mining, and machine learning in astronomy: A practical Python guide for the analysis of survey data.* Princeton NJ: Princeton Univ. Press.

Landau, R H, Páez, M J, & Bordeianu, C C. 2015. *Computational physics: Problem solving with computers.* 3rd ed. New York: Wiley-VCH. physics.oregonstate.edu/~rubin/Books/CPbook/index.html.

Langtangen, H P. 2016. *A primer on scientific programming with Python.* 5th ed. Berlin: Springer.

Libeskind-Hadas, R, & Bush, E. 2014. *Computing for biologists: Python programming and principles.* Cambridge UK: Cambridge Univ. Press.

Lutz, M. 2014. *Python pocket reference.* 5th ed. Sebastopol CA: O'Reilly Media Inc.

Nelson, P. 2015. *Physical models of living systems.* New York: W. H. Freeman and Co.

Nelson, P. 2017. *From photon to neuron: Light, imaging, vision.* Princeton NJ: Princeton Univ. Press.

Newman, M. 2013. *Computational physics.* Rev. and expanded ed. CreateSpace Publishing.

Pérez, F, & Granger, B E. 2007. IPython: A system for interactive scientific computing. *Computing in Science and Engineering*, **9**(3), 21–29.

Pine, D. 2014. *Introduction to Python for science.* github.com/djpine/pyman.

Rougier, N P, Droettboom, M, & Bourne, P E. 2014. Ten simple rules for better figures. *PLoS Comput. Biol.*, **10**(9), e1003833.
journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003833.

Scopatz, A, & Huff, K D. 2015. *Effective computation in physics.* Sebastopol CA: O'Reilly Media.

Shaw, Z. 2017. *Learn Python 3 the hard way: A very simple introduction to the terrifyingly beautiful world of computers and code.* Upper Saddle River NJ: Addison-Wesley.

Wang, J. 2016. *Computational modeling and visualization of physical systems with Python.* Hoboken NJ: Wiley.

Zemel, A, Rehfeldt, F, Brown, A E X, Discher, D E, & Safran, S A. 2010. Optimal matrix rigidity for stress-fibre polarization in stem cells. *Nature Physics*, **6**(6), 468–473.