JOURNAL OF CHEMICAL EDUCATION

Article

pubs.acs.org/jchemeduc

Scientific Computing for Chemists: An Undergraduate Course in Simulations, Data Processing, and Visualization

Charles J. Weiss*

Department of Chemistry, Wabash College, Crawfordsville, Indiana 47933, United States

Supporting Information

ABSTRACT: The Scientific Computing for Chemists course taught at Wabash College teaches chemistry students to use the Python programming language, Jupyter notebooks, and a number of common Python scientific libraries to process, analyze, and visualize data. Assuming no prior programming experience, the course introduces students to basic programming and applies these skills to solve a variety of chemical problems. The course is structured around Jupyter notebooks as easily shareable documents for lectures, homework sets, and projects; the software used in this course is free and open source, making it easily accessible to any school or research lab.

KEYWORDS: *Upper-Division Undergraduate, Curriculum, Interdisciplinary/Multidisciplinary, Computer-Based Learning, Mathematics/Symbolic Mathematics*

INTRODUCTION

In a time when research laboratories are generating increasing quantities of digital data,[1] the ability to efficiently and effectively process, analyze, and visualize digital data is becoming a critical skill for current and future generations of chemists. Spreadsheets are often used to handle digital data in many courses and research laboratories[2] because of their low cost, prevalence, and ease of use. While spreadsheets often contain a number of advanced features, there are still many tasks that even spreadsheets handle poorly or not at all (e.g., three-dimensional (3D) interpolation and image analysis). For these tasks, there are other, more advanced tools available to chemists. The Scientific Computing for Chemists course is designed to teach students to use the Python programming language,[3] Jupyter notebooks,[4] the iPython environment,[5] and a number of the common Python scientific libraries such as NumPy and SciPy[6] to process, analyze, and visualize data.

This course assumes no significant prior programming experience and begins by introducing chemistry students to programming with the Python programming language. The remainder of the course consists of units covering various scientific Python libraries and applying these tools to solve problems relevant to chemistry. Examples of problems addressed in this course include data interpolation, 3D plotting, data smoothing, image analysis, stochastic simulations, and automation of bulk data processing.

OBJECTIVES

The focus of this course is to enable students to more efficiently process, analyze, and visualize digital data and to extract more information from the available data. The primary objectives for this course are as follows:

1. Make students proficient in working with the Python programming language
2. Familiarize students with core scientific Python libraries and applying them to solve chemical problems
3. Teach students to use Jupyter notebooks as an interactive environment for solving chemical problems and to write clear explanatory text (see below for information on Jupyter notebooks)
4. Enable students to automate the processing and visualization of multiple, analogous data sets
5. Familiarize students with Python documentation

The first two objectives are met by giving students a brief introduction to the Python programming language along with a variety of Python scientific libraries, including NumPy, SciPy, Matplotlib,[7] scikit-image,[8] SymPy,[9] and pandas.[10] These libraries are applied to perform scientific tasks such as plotting data, running stochastic simulations, extracting data from images, and integrating sampled data. The course centers around Jupyter notebooks. Students are taught to include both code and documentation in the notebooks so that the notebooks can easily be shared with and modified by others. Students are also taught to automate the bulk processing of multiple data files. Finally, students are familiarized with reading documentation to get help or extend their abilities beyond the content covered in class.

RESOURCES

Book

Most of the reading assignments are from *Learning Scientific Programming with Python* by Christian Hill.[11] This text was chosen because it contains an introduction to programming in Python and Jupyter notebooks and also covers the NumPy, Matplotlib, and SciPy libraries. This text gives a condensed introduction to the Python programming language, focusing on topics relevant to scientific data analysis. The book provides background in plotting with Matplotlib, working with arrays of

Received: January 29, 2017
Revised: March 16, 2017
Published: April 4, 2017

DOI: 10.1021/acs.jchemed.7b00078
J. Chem. Educ. 2017, 94, 592−597

Table 1. Course Outline with Topics and Reading Assignments

Meeting	Topics	Assignment[a]
1	Course Introduction, Jupyter Notebooks	–
2	Numbers, Variables, Boolean Logic, Strings	Ch. 2, pp 11–23, 27–34
3	List, Tuples, Loops	Ch. 2, pp 41–53
4	Loops, Conditions	Ch. 2, pp 56–62
5	File I/O, Functions	Ch. 2, pp 66–80; Ch. 6, pp 216–217
6	Basic Plotting with Matplotlib	Ch. 3, pp 84–100
7	Basic NumPy and Arrays	Ch. 6, pp 184–202
8	NumPy Array Methods	Ch. 6, pp 203-212, 225–228, 262–270
9	Multifigure Plots and 3D Plots	Ch. 6, pp 300–304, 327–330
10	Diffusion, Kinetics	Monte Carlo article[b]
11	Polymer Simulations (hand out final project assignment)	Polymer article[c]
12	Fitting and Interpolation, *os* Module	Ch. 4, pp 134–135; Ch. 6, pp 240–245; Ch. 8, pp 374–380
13	Peak Finding	–
14	Smoothing Data	Ch. 6, pp 272–277; Smoothing Web site[d]
15	Basic Image Structures	Scikit-image Web site[e]
16	Feature Detection	–
17	Basic Pandas	Pandas Web site "Pandas in 10 minutes"[f]
18	Student Presentations	–
19	Symbolic Math (SymPy)	SymPy tutorial[g]
20	Integration (SciPy)	–

[a]Chapter and page numbers are from the texbook by Hill (ref 11). [b]See ref 20. [c]See ref 21. [d]See ref 22. [e]See ref 23. [f]See ref 24. [g]See ref 25.

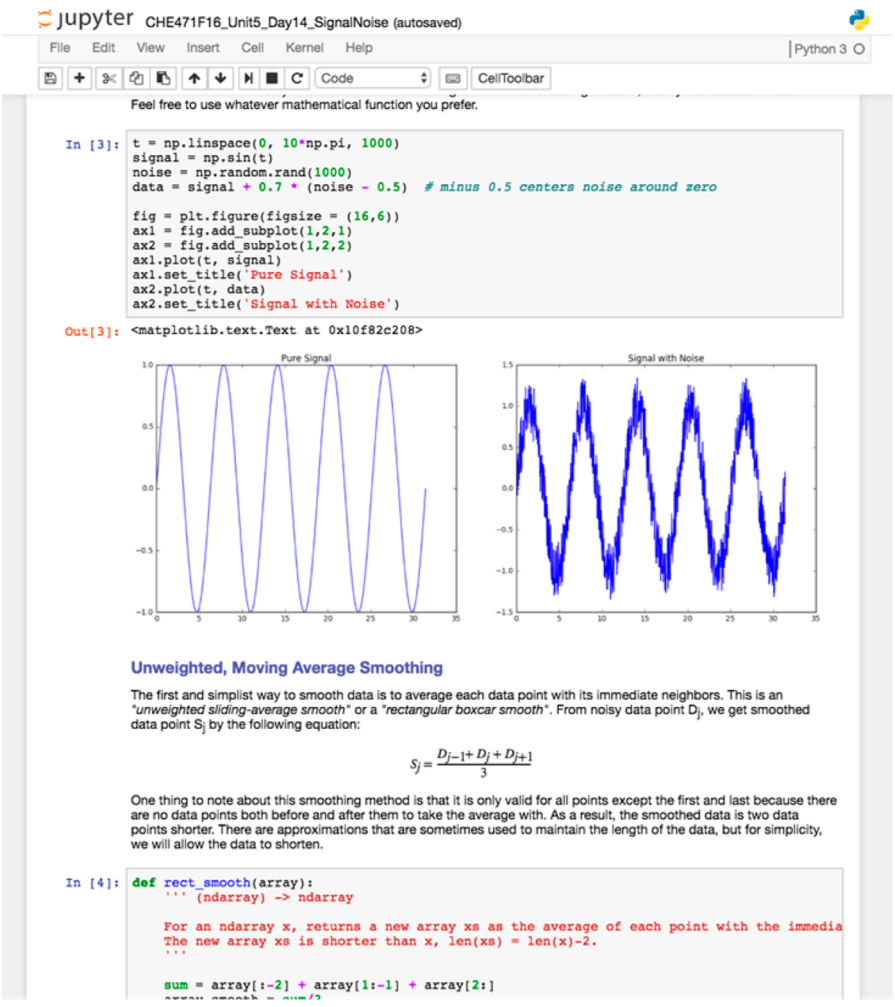

Figure 1. Screenshot of a Jupyter notebook from class containing explanatory text, Python code, and output of the code.

DOI: 10.1021/acs.jchemed.7b00078
J. Chem. Educ. 2017, 94, 592–597

data using NumPy, and data processing using various modules from SciPy. Reading assignments from the Web sites of the libraries not covered in this book, including pandas, scikit-image, and SymPy (see Table 1), are also assigned later in the course. Because this book is written for a general scientific audience, including examples from a range of fields such as biology, chemistry, physics, and engineering, chemistry-specific examples for lectures and homework sets were generated by the instructor (see the Supporting Information for examples).

Software

This course relies entirely on free software. Python 3.5 and all of the scientific libraries used in this course are free and open-source software (FOSS).[12] FOSS benefits not only the department and school but also ensures that students have continued access to the software outside the course and after graduation. Anyone with Internet access can download a copy of the software free of charge, whereas some of the proprietary alternatives may have a prohibitively large cost barrier that prevents a person from using them if a research group or employer is unable or unwilling to purchase a license. In addition to cost and availability, the Python programming language was also chosen for this course because of a wealth of resources for learning Python, a large and supportive user community, the availability of numerous field-specific libraries, and the versatility of the Python programming language.[13] The use of open-source software also allows any individual with the ability and interest to inspect and adapt the code to suit his or her needs. During one of the assignments in this course, students are asked to write their own functions to convert a color image to black-and-white and then compare it to the code of a built-in function from the scikit-image library that performs the same conversion. All of the software is also cross-platform, running on Mac, Windows, and Linux systems, making it convenient for installation on any school computers or for students to run on their personal desktop or laptop computer if desired.

While there are numerous programming environments available for the Python programming language, this course utilizes the Jupyter notebook environment (Figure 1). The Jupyter notebook, formerly known as the iPython notebook,[14] is an open document that can contain documentation, code, interactive elements, and the output of the code such as plots and numerical values. The Jupyter notebook was chosen for this course because it is an interactive environment ideal for data processing and visualization that is well-suited for sharing with others.[15] All of the course meetings take place in a computer lab with Python and all other required software installed on the systems. At the beginning of each class, the students download a folder containing a Jupyter notebook and any external data used during that class period. The notebooks created by the instructor contain documentation and occasional sections of code. As the instructor demonstrates coding and data processing in a notebook projected on a screen at the front of the room, the students are able to follow along on their individual lab computers in their own copies of the notebook.[16]

■ COURSE STRUCTURE AND CONTENT

This course is structured as a half-semester course[17] to fit as a Wabash College special topics course but can be expanded into a full-semester course with the addition of content and applications.[18] The prerequisites for this course are general chemistry and an intermediate-level course on chemical structure and reactivity.[19] These prerequisites are required in order to ensure that students have adequate chemical background and quantitative skills to apply the tools taught in this course to solve chemical problems. This course is divided into eight units designed to give students background in the Python programming language, exposure to a variety of Python scientific libraries, and experience solving a variety of problems relevant to chemistry. The course begins with a brief introduction to the Python programming language followed by units introducing students to a variety of libraries and tools in Python and methods to deal with various types of data or common problems in chemistry. An outline of the the course meetings is shown in Table 1 along with the reading assignments students are required to complete before coming to class.

Unit 1: Basic Python

The first unit in this course is a five-session introduction to the Python programming language. This is a rapid exposure to programming with the assumption that the students have little or no significant programming experience. It covers numbers, variables, Boolean logic, lists, tuples, loops, conditions, importing/exporting data from/to text files, and functions. To fit within the time constraints of this half-semester course, this introduction focuses on topics relevant to numerical data and does not go into depth on subjects such as text formatting, dictionaries, and sets. Some subjects in Chapter 4 of Hill,[11] such as dictionaries and list comprehension, are introduced later in the course as needed.

Unit 2: Matplotlib

The second unit is an introduction to the Matplotlib plotting library. Matplotlib is a common Python plotting library with the capability to make scatter plots, histograms, stream plots, and surface plots, among many others. The first session of this unit covers basic two-dimensional plots and includes practice importing data from external comma-separated value (CSV) files to plot. The second session includes plotting multiple plots in the same figure, depicting 3D data in a 2D plot, and surface plots. While the two sessions for this unit can be done consecutively, the second session is introduced after the NumPy unit because a knowledge of NumPy simplifies the creation of surface plots.

Unit 3: NumPy

Sandwiched between the two sessions of plotting is an introduction to NumPy. The NumPy library is the foundation of many of the Python scientific libraries and provides *n*-dimensional array objects for efficient handling of large amounts of data, vectorization and broadcasting between multiple arrays, and an extensive library of mathematical functions. The first session of this unit covers the creation and manipulation of data in NumPy arrays, including topics such as slicing and reshaping. The second session introduces vectorization and broadcasting between arrays as well as various NumPy methods including statistical methods and generating distributions of pseudorandom numbers.

Unit 4: Simulations

This unit exposes students to both deterministic and stochastic simulations. Using pseudorandom number generators from the NumPy library, the class generates basic simulations of Brownian motion, diffusion, and chemical kinetics for simple reactions and compares these stochastic simulations to the theoretical models or deterministic simulations. As an in-class

DOI: 10.1021/acs.jchemed.7b00078
J. Chem. Educ. 2017, 94, 592−597

activity, students also perform a stochastic simulation of chain-growth polymerization to examine polymer weight distributions (see below).

Unit 5: Signal

Three sessions are spent covering various aspects of signal processing and feature detection in data using the NumPy and SciPy libraries. The first session is spent fitting data to equations and interpolating 2D and 3D data, while the second session covers methods of finding peaks in spectra. In the first session, students are also introduced to using the native Python *os* module to navigate the computer file system. This allowed students to modify Python scripts to run through a folder of data files and perform the same processing and information extraction on multiple files. The third session of the signal unit covers noise in data. The class utilizes various methods of smoothing data[26] through moving averages and the Savitzky–Golay filter, and students learn to perform fast Fourier transforms on data, including audio data, to observe the signal and noise frequencies present. The moving average algorithms are coded by the students, while the Savitzky–Golay filtering and the fast Fourier transforms are performed by calling built-in functions from the SciPy modules *signal* and *fftpack*, respectively.

Unit 6: Images

Two sessions are spent covering the processing and extraction of information from images using the scikit-image library. Most of the first session covers basic image structure such as color channels, how the data is structured in Python, and how to work efficiently with the data in NumPy arrays. The class then uses NumPy methods to quantify the relative amounts of nanoparticle coverage in two scanning electron microscopy (SEM) images. In the second session of the images unit, the class explores a few examples of scikit-image methods to quantify entropy in an image and automate the counting of nanoparticles in SEM images.

Unit 7: Pandas

A single session is spent introducing the class to the pandas library, which adds extra data analysis tools on top of NumPy, including the ability to call data based on text labels, better handling of heterogeneous and missing data, and extra tools for working with data.[10b] This unit can potentially be incorporated at any point after NumPy is introduced, but it is scheduled near the end of the course to avoid delay in introducing students to more central topics. While introducing the pandas library is not strictly necessary for the course, it does add useful features for data processing and analysis that students anecdotally appear to appreciate.

Unit 8: Mathematics

The final unit consists of two sessions of solving symbolic mathematics and integration problems using SymPy and the *integrate* module of SciPy. The first session is spent covering SymPy, a Python library for performing symbolic mathematics, such as algebraic manipulations, differentiation, and solving both definite and indefinite integrals. The second session begins with a short introduction to using the *integrate* module from SciPy to integrate under sampled data followed by a small-group activity in which these tools are applied to calculate the entropy of a compound (see below).

STUDENT WORK

Problem Sets and Quizzes

In addition to in-class activities and reading assignments, students are required to complete problem sets after each class meeting and three quizzes throughout the term. The purpose of the exercises is to give the students additional experience with Python and practice solving chemical problems using the tools from class. The homework sets are distributed in the form of Jupyter notebooks containing the questions, with external data sometimes provided in external files such as CSV files or images. The students code their solutions directly in the Jupyter notebooks and provide answers to any questions in markdown cells directly in the notebook. This makes grading exercises simple and gives students valuable practice coding and writing in Jupyter notebooks.

Three take-home quizzes are also distributed throughout the course to test student abilities in solving chemical problems using Python. This provides feedback to the instructor on student understanding of and abilities derived from content in recent units. Similar to the problem sets, the quizzes are distributed in Jupyter notebooks with data sometimes provided in external files. Students are allowed to use notes, the book, and other resources (except other people) to solve the problems. The quizzes are open-note in order to place the emphasis of the course on the students' ability to apply Python to chemical problems in an environment analogous to research rather then testing student memorization of an extensive list of functions.

Oral Reports and Final Projects

In addition to problem sets and quizzes, students present an oral report in small groups and submit an individual, final project in Jupyter notebooks. The oral reports are short presentations where students report on and demonstrate a Python scientific library or module not otherwise covered in the course. Each report is expected to include a short background of the library or module, an explanation of what it adds to the scientific Python ecosystem, and a live demonstration of the library or module to solve a problem relevant to chemistry or biochemistry.

There are a number of goals for this presentation. First, the presentation requires students to learn to use a library or module not taught in the course. Second, students will also become more familiar with Python documentation as they use it to prepare the presentation. The report also gives students an opportunity to explore problems of unique interest and relevance to those students. For example, one group interested in biochemistry chose to present biopython,[27] a scientific library useful for biochemistry, computational biology, and bioinformatics, to analyze DNA and protein sequences. Finally, the presentation gives students experience with speaking in front of a group of their peers on a technical subject.

A final project is also required for the course in the form of a Jupyter notebook as a demonstration of student abilities to use the Python programming language to process, visualize, and analyze scientific data. This project is fundamentally a lab report in a Jupyter notebook format. While students outside of this course typically use a word processor and spreadsheet to perform analysis, visualization, and presentation of their problem, method, data, and conclusions, the final project in this course requires students to performed this in a single Jupyter notebook document that can be easily shared with others. Students who have performed research outside of this

DOI: 10.1021/acs.jchemed.7b00078
J. Chem. Educ. 2017, 94, 592–597

course are welcome to use research data for this project, but students are cautioned against making this project about doing original research instead of the intended purpose of demonstrating their ability to solve a chemical problem using the skills learned in this course.

Activities

The course includes two in-class small-group activities. The first activity is scheduled for the second session of the simulations unit, when students are given a Jupyter notebook with instructions to stochastically simulate chain-growth polymerization and use the results to examine the distribution of polymer chain lengths. Small groups of up to three students devise methods for simulating the growth of a large number of polymer chains while randomly terminating chains at different points in the simulation using a pseudorandom number generator from NumPy. The results are then compared to the theoretical model for polymer weight distributions.[21]

The second activity is conducted on the last day of class, when students calculate the entropy in one mole of a substance at a given temperature using specific heat capacity and enthalpy of phase transition data.[28] Again, students are given Jupyter notebooks containing instructions and the data in two CSV files. This activity requires students to import data from external files, perform basic calculations using pandas or NumPy, and integrate under sampled data using functions from the *integrate* module in SciPy. Because no experimental specific heat data are typically available for substances a temperatures near 0 K, values at extremely low temperatures are estimated by integrating the Debye equation using SymPy or SciPy.

EVALUATION

A course survey issued on the last day of class in Fall 2016 indicates that the students overwhelmingly found the course to be valuable for both academics and research (see the Supporting Information). Almost all of the students indicated that they felt the content would be useful in future courses, research, and their careers, but possibly the most encouraging response was that seven out of nine students indicated that they had already applied what they learned in this course to other courses and research even before the course had concluded. In addition, almost all of the students felt that the course introduced them to ways to perform valuable data analysis they did not realize they could otherwise perform, and all of the students felt the course improved their abilities to process and visualize data.

Students enjoyed the course and found the content interesting. Anecdotally, the students appeared to be excited by the content and the ability to solve new types of problems. In the survey, most of the students reported that they enjoyed using the Python programming language, and almost all of the students reported that the course increased their interest in data analysis and computer programming. Nearly all of the students felt that the course increased their confidence in working with digital data.

SUMMARY

This course introduces undergraduate chemistry students to advanced data processing and visualization tools and applications of these tools in chemistry. Students learn to extract information from a variety of data sources, including text documents, audio files, and images, and they are able to automate the bulk processing and visualization of multiple data files. The student responses to the course are very positive, and many students began applying skills learned in this course to research and other courses within the same term.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available on the ACS Publications website at DOI: 10.1021/acs.jchemed.7b00078.

Course syllabus, results of a student questionnaire, final project and oral report handouts, information on the software used in the course, and course handouts (PDF)
Examples of lecture notebooks and a homework set (ZIP)

AUTHOR INFORMATION

Corresponding Author

*E-mail: weissc@wabash.edu.

ORCID

Charles J. Weiss: 0000-0003-4102-7683

Notes

The author declares no competing financial interest.

ACKNOWLEDGMENTS

The author thanks Wabash College and the Department of Chemistry for their support and the students for their feedback, hard work, and enthusiasm in this course. Dr. Lon A. Porter, Jr., and Dr. Paul D. Schmitt of Wabash College are thanked for sharing spectroscopic data for use in this course.

REFERENCES

(1) (a) Baraniuk, R. G. More Is Less: Signal Processing and the Data Deluge. *Science* **2011**, *331*, 717−719. (b) Mullin, R. Breaking Big. *Chem. Eng. News* **2013**, *91* (42), 19−21.

(2) For examples, see: (a) Kadjo, A. F.; Stamos, B. N.; Shelor, C. P.; Berg, J. M.; Blount, B. C.; Dasgupta, P. K. Evaluation of Amount of Blood in Dry Blood Spots: Ring-Disk Electrode Conductometry. *Anal. Chem.* **2016**, *88* (12), 6531−6537. (b) Boeker, P.; Leppert, J.; Mysliwietz, B.; Lammers, P. S. Comprehensive Theory of the Deans' Switch As a Variable Flow Splitter: Fluid Mechanics, Mass Balance, and System Behavior. *Anal. Chem.* **2013**, *85* (19), 9021−9030. (c) Blanck, H. F. Using Spreadsheets To Emulate Diffusion and Thermal Conductivity. *J. Chem. Educ.* **2009**, *86* (5), 651. (d) Brown, J. H. Development and Use of a Cyclic Voltammetry Simulator To Introduce Undergraduate Students to Electrochemical Simulations. *J. Chem. Educ.* **2015**, *92* (9), 1490−1496.

(3) Python Home Page. https://www.python.org/ (accessed March 2017).

(4) (a) Jupyter Home Page. https://jupyter.org/ (accessed March 2017). (b) Rossant, C. *Learning IPython for Interactive Computing and Data Visualization*; Packt Publishing Ltd.: Birmingham, U.K., 2013. (c) Hill, C. *Learning Scientific Programming with Python*, 1st ed.; Cambridge University Press: Cambridge, U.K., 2016; pp 160−183. (d) Vaingast, S. *Beginning Python Visualization: Crafting Visual Transformation Scripts*; Apress: New York, 2014; pp 41−42. (e) Nelli, F. *Python Data Analytics*; Apress: New York, 2015; pp 26−27.

(5) (a) Pérez, P.; Granger, B. E. IPython: A System for Interactive Scientific Computing. *Comput. Sci. Eng.* **2007**, *9* (3), 21−29. (b) IPython Home Page. https://ipython.org/ (accessed March 2017).

(6) (a) Van Der Walt, S.; Colbert, S. C.; Varoquaux, G. The NumPy Array: A Structure for Efficient Numerical Computation. *Comput. Sci.*

DOI: 10.1021/acs.jchemed.7b00078
J. Chem. Educ. 2017, 94, 592−597

Eng. **2011**, *13* (2), 22−30. (b) SciPy Home Page. https://scipy.org/ (accessed March 2017. (c) NumPy Home Page. http://www.numpy.org/ (accessed March 2017).
(7) (a) Hunter, J. D. Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.* **2007**, *9* (3), 90−95. (b) Matplotlib Home Page. http://matplotlib.org/ (accessed March 2017).
(8) (a) van der Walt, S.; Schönberger, J. L.; Nunez-Iglesias, J.; Boulogne, F.; Warner, J. D.; Yager, N.; Gouillart, E.; Yu, T. Scikit-Image: Image Processing in Python. *PeerJ* **2014**, *2*, e453. (b) Scikit-Image Home Page. http://scikit-image.org/ (accessed March 2017).
(9) (a) SymPy Home Page. http://www.sympy.org/en/index.html (accessed March 2017).
(10) (a) McKinney, W. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*; van der Walt, S., Millman, J., Eds.; SciPy: Austin, TX, 2010; pp 51−56. (b) McKinney, W. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Ipython*, 1st ed.; O'Reilly: Sebastopol, CA, 2012. (c) Pandas Home Page. http://pandas.pydata.org/ (accessed March 2017).
(11) Hill, C. *Learning Scientific Programming with Python*, 1st ed.; Cambridge University Press: Cambridge, U.K., 2016.
(12) Open Source Initiative. https://opensource.org/osd (accessed March 2017).
(13) Perkel, J. M. Pick Up Python. *Nature* **2015**, *518*, 125−126.
(14) The notebook component of the original iPython project developed into the Jupyter notebook, which supports programming languages beyond just Python. The iPython project continues to develop and support the IPython kernel for the Jupyter notebook. See https://ipython.org/ (accessed March 2017).
(15) For examples of the use of Jupyter notebooks, see: (a) Srnec, M. N.; Upadhyay, S.; Madura, J. D. Teaching Reciprocal Space to Undergraduates via Theory and Code Components of an IPython Notebook. *J. Chem. Educ.* **2016**, *93* (12), 2106−2109. (b) Cruzeiro, V. W. D.; Roitberg, A.; Polfer, N. C. Interactively Applying the Variational Method to the Dihydrogen Molecule: Exploring Bonding and Antibonding. *J. Chem. Educ.* **2016**, *93* (9), 1578−1585.
(16) The Jupyter notebooks can be accessed from any web browser. The software may be installed on individual computers (as was done for this course) or on a server that students can access remotely via their web browsers.
(17) Because of scheduling issues, the course met only twice a week instead of the typical thrice. As a result, the course extended beyond the middle of the semester. Meeting only twice a week had the advantage of giving students more time to think over problems.
(18) Additional topics could include optimization algorithms, machine learning, and processing of NMR spectra, among others.
(19) This course follows general chemistry and introduces thermodynamics, inorganic chemistry, coordination chemistry, and analytical chemistry at an intermediate level.
(20) Rabinovitch, B. The Monte Carlo Method: Plotting the Course of Complex Reactions. *J. Chem. Educ.* **1969**, *46* (5), 262−268.
(21) Horta, A.; Pastoriza, M. A. The Molecular Weight Distribution of Polymer Samples. *J. Chem. Educ.* **2007**, *84*, 1217−1221.
(22) Smoothing Data Guide. https://terpconnect.umd.edu/~toh/spectrum/Smoothing.html (accessed March 2017).
(23) A Crash Course on NumPy for Images. http://scikit-image.org/docs/stable/user_guide/numpy_images.html (accessed March 2017).
(24) 10 Minutes to pandas. http://pandas.pydata.org/pandas-docs/stable/10min.html (accessed March 2017).
(25) SymPy Tutorial. http://docs.sympy.org/latest/tutorial/intro.html (accessed March 2017).
(26) O'Haver, T. C. An introduction to signal processing in chemical measurement. *J. Chem. Educ.* **1991**, *68* (6), A147−A150.
(27) Biopython Project Home Page. http://biopython.org (accessed March 2017).
(28) (a) Zielinski, T. J.; Hamilton, T. M. Conventional Entropies and the Third Law of Thermodynamics. *J. Chem. Educ.* **2009**, *86* (12), 1468. (b) Osborne, D. W.; Flotow, H. E.; Dallinger, R. F.; Hoekstra, H. R. *J. Chem. Thermodyn.* **1974**, *6* (8), 751−756.

DOI: 10.1021/acs.jchemed.7b00078
J. Chem. Educ. 2017, 94, 592−597