Project-based introduction to scientific computing for physics majors

Jennifer Klay‡*

https://www.youtube.com/watch?v=eJhmMf6bHDU

Abstract—This paper presents an overview of a project-based course in computing for physics majors using Python and the IPython Notebook that was developed at Cal Poly San Luis Obispo. The course materials are made freely available on GitHub as a project under the Computing4Physics [C4P] organization.

Index Terms—physics, scientific computing, undergraduate education

Introduction

Computational tools and skills are as critical to the training of physics majors as calculus and math, yet they receive much less emphasis in the undergraduate curriculum. One-off courses that introduce programming and basic numerical problem-solving techniques with commercial software packages for topics that appear in the traditional physics curriculum are insufficient to prepare students for the computing demands of modern technical careers. Yet tight budgets and rigid degree requirements constrain the ability to expand computational course offerings for physics majors.

This paper presents an overview of a recently revamped course at California Polytechnic State University San Luis Obispo (Cal Poly) that uses Python and associated scientific computing libraries to introduce the fundamentals of open-source tools, version control systems, programming, numerical problem solving and algorithmic thinking to undergraduate physics majors. The spirit of the course is similar to the bootcamps organized by Software Carpentry [SWC] for researchers in science but is offered as a ten-week for-credit course. In addition to having a traditional in-class component, students learn the basics of Python by completing tutorials on Codecademy's Python track [Codecademy] and practice their algorithmic thinking by tackling Project Euler problems [PE]. This approach of incorporating online training may provide a different way of thinking about the role of MOOCs in higher education. The early part of the course focuses on skill-building, while the second half is devoted to application of these skills to an independent research-level computational physics project. Examples of recent student projects and their results will be presented.

* *Corresponding author: jklay@calpoly.edu*
‡ *California Polytechnic State University San Luis Obispo*

Background

California Polytechnic State University San Luis Obispo (Cal Poly) is one of the 23 campuses of the California State University system. The university has a "learn by doing" emphasis for the educational experience of its predominantly undergraduate population of approximately 19,000 students, encapsulated in its motto *discere faciendo*. Part of the university's mission is to provide students the opportunity to get directly involved in research at the frontiers of knowledge through interaction with faculty. The university is also committed to enhancing opportunities for underrepresented groups and is committed to fostering a diverse student body.

The College of Engineering enrolls the largest fraction of Cal Poly undergraduates (~28%). Due to the large number of engineering undergraduates at Cal Poly, the distribution of male (~54%) and female (~46%) students is opposite that of the national average.

The Department of Physics, in the College of Science & Mathematics, offers Bachelor of Science and Arts degrees in Physics, and minors in astronomy and geology, with approximately 150 students enrolled. There are roughly 30 tenure-track faculty, for a current student-to-faculty ratio of 1:5. In addition, there are typically 5-10 full-time lecturers and fifteen part-time and retired faculty teaching courses in physics and geology. A typical introductory physics course for scientists and engineers has 48 students, in contrast to typical class sizes of over a hundred at large public universities. The curriculum for physics majors includes a Senior Project which is often the continuation of paid summer internships undertaken with faculty members in the department who have funding to support student assistants. Some internal funding is made available to support these activities.

Cal Poly has one of the largest (in terms of degrees granted) and most successful undergraduate physics programs in the United States. Only about 5% of all physics programs in the United States regularly award more than 15 degrees per year, and most of those are at Ph.D. granting institutions. In 2013-2014, 28 B.S. and 1 B.A. degrees were awarded. The Cal Poly Physics Department is uniquely successful among four-year colleges. As a result, Cal Poly was one of 21 departments deemed to be "thriving" and profiled in 2002 by the SPIN-UP study (Strategic Programs for INnovation in Undergraduate Physics) sponsored by the American Association of Physics Teachers, the American Physical Society, and the American Institute of Physics [SPIN-UP]. The external reviewers from SPIN-UP made special mention of the strong

faculty-student interactions and of the success of the physics lounge (known as "h-bar") at making students feel welcome and at home in an intense academic environment. Cal Poly hosted the SPIN-UP Western Regional Workshop in June 2010 where faculty teams from 15 western colleges and universities came to learn how to strengthen their undergraduate physics programs.

Computational physics at Cal Poly

The physics department has a strong record of preparing students for advanced degrees in physics, often at top tier research institutions. Between 2005 and 2009, at least 20% of Cal Poly physics graduates entered Ph.D. programs in physics and related disciplines with another 10% seeking advanced degrees in engineering, mathematics, law, and business.

The Cal Poly physics program provides a strong base in theoretical physics with the standard traditional sequence of courses while providing excellent experimental training of students in the laboratory, with a full year of upper division modern physics experiments and several additional specialty lab courses offered as advanced physics electives. Unfortunately, the department has not yet developed as cohesive and comprehensive of a program in computational physics. There has been one course "Physics on the Computer" on computational methods required for physics majors since 1996. The current catalog description of the course is

> *Introduction to using computers for solving problems in physics: differential equations, matrix manipulations, simulations and numerical techniques, nonlinear dynamics. 4 lectures.*

Students are encouraged to take the course in the Spring of their sophomore year, after completing their introductory physics and math courses. The original pre-requisites for the course were General Physics III: Electricity and Magnetism and Linear Analysis I (MATH), although in 1998 concurrent enrollment for Linear Analysis was allowed and in 2001 the phrase "and computer literacy" was added to the pre-requisites, although it was dropped when enforceable pre-requisites were introduced in 2011.

Despite the desire for students to come to this course with some "computer literacy", no traditional computer science courses have been required for physics majors (although they can be counted as free technical electives in the degree requirements). Each instructor selects the tools and methods used to implement the course. Early on, many numerical topics were covered using Excel because students typically had access and experience with it. Interactive computer algebra systems such as Maple and interactive computing environments such as MATLAB were also employed, but no open-source standard high level programming languages were used. Between 2007 and 2012 MATLAB was the preferred framework, although some use of Excel for introductory tasks was also included.

Although simple data analysis and graphing tasks are taught in upper division laboratories, there is no concerted effort to include computational or numerical techniques in upper division theory courses. Instructors choose to include this material at their own discretion. There is also currently no upper division computational physics elective in the catalog.

When I joined the faculty of Cal Poly in 2007 I quickly obtained external funding from the National Science Foundation to involve Cal Poly physics undergraduates in research at the CERN Large Hadron Collider with the ALICE experiment. My background in particle and nuclear physics has been very software intensive, owing to the enormous and complex datasets generated in heavy nucleus collisions. I have served as software coordinator for one of the ALICE detector sub-systems and I am the architect and lead developer of the offline analysis framework for the Neutron Induced Fission Fragment Tracking Experiment (NIFFTE). Most of my scientific software is written in C/C++, although I have experience with Pascal, Fortran, Java and shell scripting. I found it extremely challenging to engage students in my research because of the steep learning curve for these software tools and languages.

In 2012 I became interested in learning Python and decided to offer an independent study course called "Python 4 Physicists" so students could learn it with me. Over 30 eager students signed up for the course. We followed Allen Downey's "Think Python" book [Downey2002] for six weeks, largely on our own, but met weekly for one hour to discuss issues and techniques. For the second half of the course, the students were placed in groups of 3 and assigned one of two projects, either a cellular automaton model of traffic flow or a 3-D particle tracking algorithm for particle collision data reconstruction. All code and projects were version controlled with git and uploaded to GitHub. Examples can be found on GitHub [Traffic], [3DTracker]. At the end of the quarter the groups presented their projects to the class.

Not all groups were able to successfully complete the projects but this is likely due to competing priorities consuming their available coding time given that this was only a 1-unit elective course. Nevertheless, they were excited to work on a research-level problem and to be able to use their newly acquired programming skills to do so. Most of them gained basic programming proficiency and some students reported that the course helped them secure summer internships. It became clear to me that Python is an effective and accessible language for teaching physics majors how to program. When my opportunity to teach "Physics on the Computer" came in 2013-14, I decided to make it a project-based Python programming course that would teach best practices for scientific software development, including version control and creation of publication quality graphics, while giving a broad survey of major topics in computational physics.

Course Organization

The complete set of materials used for this course are available on GitHub under the Computing4Physics [C4P] organization and can be viewed with the IPython Notebook Viewer [nbviewer]. The learning objectives for the course are a subset of those developed and adopted by the Cal Poly physics department in 2013 for students completing a degree in physics:

- Use basic coding concepts such as loops, control statements, variable types, arrays, array operations, and boolean logic. (LO1)
- Write, run and debug programs in a high level language. (LO2)
- Carry out basic operations (e.g. cd, ls, dir, mkdir, ssh) at the command line. (LO3)
- Maintain a version controlled repository of your files and programs. (LO4)
- Create publication/presentation quality graphics, equations. (LO5)
- Visualize symbolic analytic expressions - plot functions and evaluate their behavior for varying parameters. (LO6)

- Use numerical algorithms (e.g. ODE solvers, FFT, Monte Carlo) and be able to identify their limitations. (LO7)
- Code numerical algorithms from scratch and compare with existing implementations. (LO8)
- Read from and write to local or remote files. (LO9)
- Analyze data using curve fitting and optimization. (LO10)
- Create appropriate visualizations of data, e.g. multidimensional plots, animations, etc. (LO11)

The course schedule and learning objective map are summarized in Table 1. Class time was divided into two 2-hour meetings on Tuesdays and Thursdays each week for ten weeks. For the first two weeks the students followed the Python track at Codecademy [Codecademy] to learn basic syntax and coding concepts such as loops, control statements, variable types, arrays, array operations, and boolean logic. In class, they were instructed about the command line, ssh, the UNIX shell and version control. Much of the material for the early topics came from existing examples, such as Software Carpentry [SWC] and Jake Vanderplas's Astronomy 599 course online [Vanderplas599]. These topics were demonstrated and discussed as instructor-led activities in which they entered commands in their own terminals while following along with me.

The IPython Notebook was introduced in the second week and their first programming exercise outside of Codecademy was to pair-program a solution to Project Euler [PE] Problem 1. They created their own GitHub repository for the course and were guided through the workflow at the start and end of class for the first several weeks to help them get acclimated. We built on their foundations by taking the Battleship game program they wrote in Codecademy and combining it with ipythonblocks [ipythonblocks] to make it more visual. We revisited the Battleship code again in week 4 when we learned about error handling and a subset of the students used ipythonblocks as the basis for their final project on the Schelling Model of segregation. The introduction, reinforcement and advanced application of programming techniques was employed to help students build lasting competency with fundamental coding concepts.

For each class session, the students were provided a "tour" of a specific topic for which they were instructed to read and code along in their own IPython Notebook. They were advised not to copy/paste code, but to type their own code cells, thinking about the commands as they went to develop a better understanding of the material. After finishing a tour they worked on accompanying exercises. I was available in class for consultations and questions but there was very little lecturing beyond the first week. Class time was activity-based rather than lecture-based. Along with the homework exercises, they completed a Project Euler problem each week to practice efficient basic programming and problem solving.

A single midterm exam was administered in the fifth week to motivate the students to stay on top of their skill-building and to assess their learning at the midway point. The questions on the midterm were designed to be straightforward and completable within the two-hour class time.

Assessment of learning

Figuring out how to efficiently grade students' assignments is a non-trivial task. Grading can be made more efficient by automatic output checking but that doesn't leave room for quality assessment and feedback. To deal with the logistics of grading, a set of UNIX shell scripts was created to automate the bookkeeping and communication of grades. Individual assignments were assessed personally by me while a grader was employed to evaluate the Project Euler questions. The basic grading rubric uses a 5-point scale for each assigned question, outlined in Table 2. Comments and numerical scores were recorded for each student and communicated to them through a script-generated email. Students' final grades in the course were determined by weighting the various course elements accordingly: Project Euler (10%), Exercises (30%), Midterm (20%), Project (30%), Demo (10%).

Week	Topics	Learning Objectives
1	Programming Bootcamp	LO1, LO2, LO3, LO4
2	Programming Bootcamp	LO1-4, LO11
3	Intro to NumPy/SciPy, Data I/O	LO1-4, LO9, LO11
4	Graphics, Animation and Error handling	LO1-4, LO5, LO6, LO11
5	Midterm Exam, Projects and Program Design	LO1-4, LO5, LO6, LO9
6	Interpolation and Differentiation	LO1-4, LO5, LO6, LO7, LO8, LO11
7	Numerical Integration, Ordinary Differential Equations (ODEs)	LO1-4, LO5, LO6, LO7, LO8, LO11
8	Random Numbers and Monte-Carlo Methods	LO1-4, LO5, LO6, LO7, LO8, LO11
9	Linear Regression and Optimization	LO1-11
10	Symbolic Analysis, Project Hack-a-thon!	LO1-4, LO5, LO6, LO11
Final	Project Demos	LO1-11

TABLE 1: *Course schedule of topics and learning objectives*

Points	Description
5	Goes above and beyond. Extra neat, concise, well-commented code, and explores concepts in depth.
4	Complete and correct. Includes an analysis of the problem, the program, verification of at least one test case, and answers to questions, including plots.
3	Contains a few minor errors.
2	Only partially complete or has major errors.
1	Far from complete.
0	No attempt.

TABLE 2: *Grading rubric for assigned exercises.*

Projects

Following the midterm exam one class period was set aside for presenting three project possibilities and assigning them. Two of the projects came from Stanford's NIFTY assignment database [Nifty] - "Schelling's Model of Segregration" by Frank McCown [McCown2014] and "Estimating Avogadro's Number from Brownian Motion" by Kevin Wayne [Wayne2013]. The Schelling Model project required students to use IPython widgets and ipythonblocks to create a grid of colored blocks that move according to a set of rules governing their interactions. Several recent physics publications on the statistical properties

of Schelling Model simulations and their application to physical systems [Vinkovic2006], [Gauvin2009], [DallAsta2008] were used to define research questions for the students to answer using their programs. For estimating Avogadro's number, the students coded a particle identification and tracking algorithm that they could apply to the frames of a movie showing Brownian motion of particles suspended in fluid. The initial test data came from the Nifty archive, but at the end of the quarter the students collected their own data using a microscope in the biology department to image milkfat globules suspended in water. The challenges of adapting their code to the peculiarities of a different dataset were part of the learning experience. They used code from a tour and exercise they did early in the quarter, based on the MultiMedia programming lesson on Software Carpentry, which had them filter and count stars in a Hubble image.

The third project was to simulate galaxy mergers by solving the restricted N-body problem. The project description was developed for this course and was based on a 1972 paper by Toomre and Toomre [Toomre1972]. They used SciPy's *odeint* to solve the differential equations describing the motion of a set of massless point particles (stars) orbiting a main galaxy core as a disrupting galaxy core passed in a parabolic trajectory. The students were not instructed on solving differential equations until week 7, so they were advised to begin setting up the initial conditions and visualization code until they had the knowledge and experience to apply *odeint*.

The projects I selected for the course are ones that I have not personally coded myself but for which I could easily outline a clear algorithmic path to a complete solution. Each one could form a basis for answering real research questions. There are several reasons for this approach. First, I find it much more interesting to learn something new through the students' work. I would likely be bored otherwise. Second, having the students work on a novel project is similar to how I work with students in research mentoring. My interactions with them are much more like a real research environment. By not already having one specific solution I am able to let them choose their own methods and algorithms, providing guidance and suggestions rather than answers to every problem or roadblock they encounter. This gives them the chance to experience the culture of research before they engage in it outside of the classroom. Finally, these projects could easily be extended into senior projects or research internship opportunities, giving the students the motivation to keep working on their projects after the course is over. As a consequence of these choices, the project assessment was built less on "correctness" than on their formulation of the solution, documentation of the results, and their attempt to answer the assigned "research question". The rubric was set up so that they could earn most of the credit for developing an organized, complete project with documentation, even if their results turned out to be incorrect.

When this course was piloted in 2013, project demonstrations were not included, as they had been for the 2012 independent study course. I was disappointed in the effort showed by the majority of students in the 2013 class, many of whom ultimately gave up on the projects and turned in sub-standard work, even though they were given additional time to complete them. For 2014, the scheduled final exam time was used for 5-7 minute project demonstrations by each individual student. Since the class was divided into three groups, each working on a common project, individual students were assigned a personalized research question to answer with their project code and present during their demo. The students were advised that they needed to present *something*, even if their code didn't function as expected. Only one student out of 42 did not make a presentation. (That student ultimately failed the course for turning in less than 50% of assignments and not completing the project.) The rest were impressive, even when unpolished.

It was clear from the demos that the students were highly invested in their work and were motivated to make a good impression. The project demos were assessed using a peer evaluation oral presentation rubric that scored the demos on organization, media (graphics, animations, etc. appropriate for the project), delivery, and content. Presenters were also asked to evaluate their own presentations. Grades were assigned using the average score from all peer evaluation sheets. The success of the project demos strongly suggest that they are an essential part of the learning experience for students. This is supported in the literature. See for example, Joughin and Collom [Joughin2003].

Project Examples

The most impressive example from 2014 came from a student who coded the Galaxy Merger project [Parry2014]. Figure 1 shows a still shot from an animated video he created of the direct passage of an equal mass diruptor after the interaction has begun. He also uploaded Youtube videos of his assigned research question (direct passage of an equal mass diruptor) from two perspectives, the second of which he coded to follow his own curiosity - it was not part of the assignment. The main galaxy perspective can be viewed here: http://www.youtube.com/watch?v=vavfpLwmT0o and the interaction from the perspective of the disrupting galaxy can be viewed here: http://www.youtube.com/watch?v=iy7WvV5LUZg

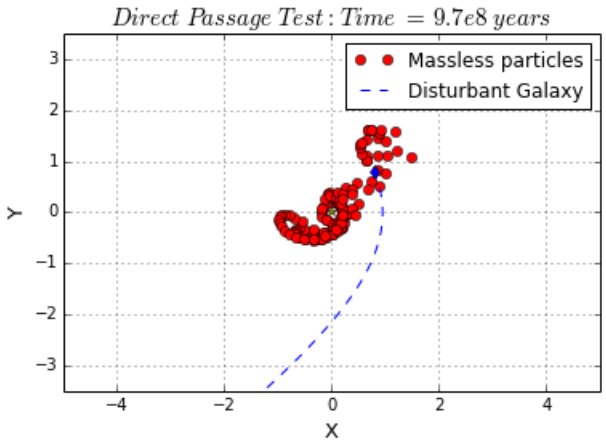

Fig. 1: *Direct passage of an equal mass disruptor galaxy shortly after the disrupting galaxy passes the minimum distance of approach. [Parry2014]*

There were also two other good Youtube video examples of the galaxy merger project, although the solutions exhibited pathologies that this one did not.

The best examples from the Schelling Model either did an excellent analysis of their research question [Nelson2014] or created the most complete and useful interactive model [Parker2014].

Highlights from 2013

Although no project demos were required in 2013, students who submitted excellent projects were invited to collaborate together

on a group presentation of their work at the 2013 annual meeting of the Far West Section of the American Physical Society held at Sonoma State University Nov. 1-2, 2013 [Sonoma2013]. Two talks were collaborations among four students each, one talk was a pair collaboration, and one was given as a single author talk.

The single author talk came from the best project submitted in 2013, an implementation of a 3-D particle tracking code [VanAtta2013] for use with ionization chamber data from particle collision experiments. Figure 2 shows an example of the output from his tracker with the voxels associated with different trajectories color coded. The notebook was complete and thorough, addressing all the questions and including references. Although the code could be better organized to improve readability, the results were impressive and the algorithm was subsequently adapted into the NIFFTE reconstruction framework for use in real experiments.

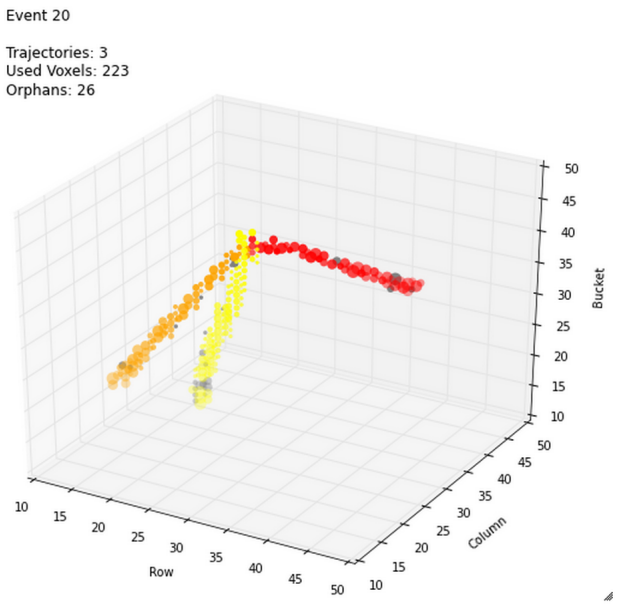

Fig. 2: *Matplotlib 3d plot of particle trajectories reconstructed from ionization trails left by charged particles in a gaseous drift detector. [VanAtta2013]*

One of the students from the pair collaboration turned his project from 2013 into a Cal Poly senior project recently submitted [Rexrode2014]. He extended his initial work and created an open library of code for modeling the geometry of nuclear collisions with the Monte Carlo Glauber model. The project writeup and the code can be found on GitHub under the [MCGlauber] organization.

Pre- and Post- Assessment

In order to assess the course's success at achieving the learning objectives, both a pre-learner survey and course evaluations were administered anonymously. The pre-learner survey, adapted from a similar Software Carpentry example, was given on the first day of class with 100% participation, while the course evaluation was given in the last week. Some in class time was made available for the evaluations but students were also able to complete it on their own time. Course evaluations are conducted through the Cal Poly "SAIL" (Student Assessment of Instruction and Learning) online system. SAIL participation was 82%. Some questions were common to both the pre and post assessment, for comparison.

Learning Objective	Completely or mostly	Neutral or partially	Not met
LO1	33/36	3/36	0/36
LO2	31/36	5/36	0/36
LO3	33/36	2/36	0/36
LO4	31/36	5/36	0/36
LO5	32/36	4/36	0/36
LO6	31/35	4/35	0/35
LO7	25/35	10/35	0/35
LO8	27/35	7/35	1/35
LO9	30/35	5/35	0/35
LO10	26/35	9/35	0/35
LO11	30/35	5/35	0/35

TABLE 3: *Student evaluation of how well the course met the learning objectives.*

Language	Pre-	Post-
Fortran	0/42	1/34
C	5/42	7/34
C++	6/42	5/34
Perl	0/42	0/34
MATLAB	5/42	1/34
Python	3/42	31/34
R	1/42	1/34
Java	7/42	5/34
Others (list)	7/42 Labview	1/34
None	20/42	2/34

TABLE 4: With which programming languages could you write a program from scratch that reads a column of numbers from a text file and calculates mean and standard deviation of that data? (Check all that apply)

The first question on the post-assessment course evaluation asked the students to rate how well the course met each of the learning objectives. The statistics from this student-based assessment are included in Table 3.

Students were also asked to rate the relevance of the learning objectives for subsequent coursework at Cal Poly and for their career goals beyond college. In both cases, a majority of students rated the course as either "Extremely useful, essential to my success" (21/34 and 20/34) or "Useful but not essential" (12/34 and 11/34) and all but one student out of 34 expected to use what they learned beyond the course itself. Almost all students indicated that they spent at least 5-6 hours per week outside of class doing work for the course, with half (17/34) indicating they spent more than 10 hours per week outside of class.

The four questions that were common to both the pre- and post- evaluations and their corresponding responses are included in Tables 4, 5, 6, and 7.

It is worth noting that the 7/42 students who indicated they could complete the programming task with Labview at the beginning of the course probably came directly from the introductory electronics course for physics majors, which uses Labview heavily.

Of the free response comments in the post-evaluation, the most common was that more lecturing by the instructor would have enhanced their learning and/or helped them to better understand some of the coding concepts. In future offerings, I might add

Answer	Pre-	Post-
I could not complete this task.	19/42	3/34
I could complete the task with documentation or search engine help.	22/42	13/34
I could complete the task with little or no documentation or search engine help.	1/42	18/34

TABLE 5: In the following scenario, please select the answer that best applies to you. A tab-delimited file has two columns showing the date and the highest temperature on that day. Write a program to produce a graph showing the average highest temperature for each month.

Answer	Pre-	Post-
I could not complete this task.	42/42	2/34
I could complete the task with documentation or search engine help.	0/42	17/34
I could complete the task with little or no documentation or search engine help.	0/42	15/34

TABLE 6: In the following scenario, please select the answer that best applies to you. Given the URL for a project's version control repository, check out a working copy of that project, add a file called notes.txt, and commit the change.

a brief mini-lecture to the beginning of each class meeting to introduce and discuss concepts but I will keep the focus on student-centered active learning.

Conclusion

This paper presented an example of a project-based course in scientific computing for undergraduate physics majors using the Python programming language and the IPython Notebook. The complete course materials are available on GitHub through the Computing4Physics [C4P] organization. They are released under a modified MIT license that grants permission to anyone the right to use, copy, modify, merge, publish, distribute, etc. any of the content. The goal of this project is to make computational tools for training physics majors in best practices freely available. Contributions and collaboration are welcome.

The Python programming language and the IPython Notebook are effective open-source tools for teaching basic software skills. Project-based learning gives students a sense of ownership of their work, the chance to communicate their ideas in oral live software demonstrations and a starting point for engaging in physics research.

References

[C4P] All course materials can be obtained directly from the Computing4Physics organization on GitHub at https://github.com/Computing4Physics/C4P

[SWC] "Software Carpentry: Teaching lab skills for scientific computing", http://software-carpentry.org/, accessed 2 July 2014.

Answer	Pre-	Post-
I could not create this list.	35/42	3/34
I would create this list using "Find in Files" and "copy and paste"	2/42	0/34
I would create this list using basic command line programs.	4/42	2/34
I would create this list using a pipeline of command line programs.	1/42	2/34
I would create this list using some Python code and the ! escape.	N/A	19/34
I would create this list with code using the Python 'os' and 'sys' libraries.	N/A	8/34

TABLE 7: How would you solve this problem? A directory contains 1000 text files. Create a list of all files that contain the word "Drosophila" and save the result to a file called results.txt. ***Note:*** *the last two options on this question were included in the post-survey only.*

[Codecademy] "Codecademy: Learn to code interactively, for free.", http://www.codecademy.com/, accessed 2 July 2014.

[PE] "ProjectEuler.net: A website dedicated to the puzzling world of mathematics and programming", https://projecteuler.net/, accessed 2 July 2014.

[SPIN-UP] "American Association of Physics Teacher: Strategic Programs for Innovations in Undergraduate Physics", http://www.aapt.org/Programs/projects/spinup/, accessed 2 July 2014.

[Downey2002] Allen B. Downey, Jeffrey Elkner, and Chris Meyers, "Think Python: How to Think Like a Computer Scientist", Green Tea Press, 2002, ISBN 0971677506, http://www.greenteapress.com/thinkpython/thinkpython.html

[Traffic] D.Townsend, J. Fernandes, R. Mullen, and A. Parker, GitHub repository for the cellular automaton model of traffic flow created for the Spring 2012 PHYS 200/400 course at Cal Poly, https://github.com/townsenddw/discrete-graphic-traffic, accessed 2 July 2014.

[3DTracker] R.Cribbs, K. Boucher, R. Campbell, K. Flatland, and B. Norris, GitHub repository for the 3-D pattern recognition tracker created for the Spring 2012 PHYS 200/400 course at Cal Poly, https://github.com/Rolzroyz/3Dtracker, accessed 2 July 2014.

[nbviewer] "nbviewer: A simple way to share IPython Notebooks", http://nbviewer.ipython.org, accessed 2 July 2014.

[Vanderplas599] Jake Vanderplas, "Astronomy 599: Introduction to Scientific Computing in Python", https://github.com/jakevdp/2013_fall_ASTR599/, accessed 2 July 2014.

[ipythonblocks] "ipythonblocks: code + color", http://ipythonblocks.org/, accessed 2 July 2014.

[Nifty] "Nifty Assignments: The Nifty Assignments session at the annual SIGCSE meeting is all about gathering and distributing great assignment ideas and their materials.", http://nifty.stanford.edu/, accessed 2 July 2014.

[McCown2014] Frank McCown, "Schelling's Model of Segregation", http://nifty.stanford.edu/2014/mccown-schelling-model-segregation/, accessed 2 July 2014.

[Wayne2013] Kevin Wayne, "Estimating Avogadro's Number", http://nifty.stanford.edu/2013/wayne-avogadro.html, accessed 2 July 2014.

[Vinkovic2006] D.Vinkovic and A.Kirman, Proc.Nat.Acad.Sci., vol. 103 no. 51, 19261-19265 (2006). http://www.pnas.org/content/103/51/19261.full

[Gauvin2009] L.Gauvin, J.Vannimenus, J.-P.Nadal, Eur.Phys.J. B, Vol. 70:2 (2009). http://link.springer.com/article/10.1140%2Fepjb%2Fe2009-00234-0

[DallAsta2008] L.Dall'Asta, C.Castellano, M.Marsili, J.Stat.Mech. L07002 (2008). http://iopscience.iop.org/1742-5468/2008/07/L07002/

[Toomre1972] A.Toomre and J.Toomre, Astrophysical Journal, 178:623-666 (1972). http://adsabs.harvard.edu/abs/1972ApJ...178..623T

[Joughin2003] G.Joughin and G.Collom, "Oral Assessment. The Higher Education Academy", (2003) http://www.heacademy.ac.uk/resources/detail/resource_database/id433_oral_assessment, retrieved 2 July 2014.

[Parry2014] B.W. Parry, "Galaxy Mergers: The Direct Passage Case", http://nbviewer.ipython.org/github/bwparry202/PHYS202-S14/blob/master/GalaxyMergers/GalaxyMergersFinal.ipynb, accessed 2 July 2014.

[Nelson2014] P.C. Nelson, "Schelling Model", http://nbviewer.ipython.org/github/pcnelson202/PHYS202-S14/blob/master/IPython/SchellingModel.ipynb, accessed 2 July 2014.

[Parker2014] J.Parker, "Schelling Model", http://nbviewer.ipython.org/github/jparke08/PHYS202-S14/blob/master/SchellingModel.ipynb, accessed 2 July 2014.

[Sonoma2013] "2013 Annual Meeting of the American Physical Society, California-Nevada Section", http://epo.sonoma.edu/aps/index.html, accessed 2 July 2014.

[VanAtta2013] John Van Atta, "3-D Trajectory Generation in Hexagonal Geometry", http://nbviewer.ipython.org/github/jvanatta/PHYS202-S13/blob/master/project/3dtracks.ipynb, accessed 2 July 2014.

[Rexrode2014] Chad Rexrode, "Monte-Carlo Glauber Model Simulations of Nuclear Collisions", http://nbviewer.ipython.org/github/crexrode/PHYS202-S13/blob/master/SeniorProject/MCGlauber.ipynb, accessed 2 July 2014.

[MCGlauber] "MCGlauber: An Open-source IPython-based Monte Carlo Glauber Model of Nuclear Collisions", https://github.com/MCGlauber, accessed 2 July 2014.