Teaching Innovation Projects

Volume 8 | Issue 1 | 2018

Integrating Computer Programming into Introductory Physics Courses

Jeff VanKerkhove

The University of Western Ontario, jvankerk@uwo.ca

Follow this and additional works at: <https://ojs.lib.uwo.ca/index.php/tips>

Recommended Citation:

VanKerkhove, J. (2018) Integrating computer programming into introductory physics courses.
Teaching Innovation Projects, 8(1), 1-11. <https://doi.org/10.5206/tips.v8i1.6221>

ABSTRACT

Computing has become essential in virtually all physical fields, used for tasks such as modelling complex systems and analyzing data. As a result, computer programming competence is now considered a default requirement for physics research. Additionally, computer programming requires critical thinking and problem solving skills – both of which are also essential for physics and other rigorous disciplines. Thus, learning to program at the undergraduate level not only facilitates students' ability to apply physical principles to solving problems, but also boosts marketable skills valuable in a more general job market. However, little emphasis is placed on computer literacy in the introductory courses of undergraduate physics curricula. Physics students interested in pursuing undergraduate research will often need to either take a computer science course or learn a computer programming language independently. In either case, it takes the student a long time to gain an understanding of the language and be able to apply it to relevant problems. This workshop is geared toward instructors and teaching assistants in introductory undergraduate physics courses with a working understanding of and experience using at least one programming language (e.g., Python, MATLAB, C++) for scientific applications. The intention is to introduce methods and provide suggestions for more effectively introducing students to scientific programming and integrating it into the physics curriculum.

KEYWORDS

physics education; computer programming competence; problem-solving

LEARNING OUTCOMES

By the end of this workshop, participants will be able to:

- Identify the importance of programming in the development of technical and cognitive skills needed for solving complicated physical problems.
- Design lessons and tutorials for introducing students to programming in a way that is directly applicable to the content of their physics courses.
- Develop course activities and assessments requiring students to apply computational knowledge in solving physics problems.

ANNOTATED BIBLIOGRAPHY

Aiken, J. M., Caballero, M. D., Douglas, S. S., Burk, J. B., Scanlon, E. M., Thoms, B. D., & Schatz, M. F. (2013). Understanding student computational thinking with computational modeling. *PERC Proceedings*, 1513, 46-49. <https://doi.org/10.1063/1.4789648>

This paper discusses a pilot program for incorporating computational programming into a high school physics course. They argue that modern advances in science rely heavily on computation, making programming skills essential for a successful career in science. However, students are not usually exposed to programming in high school – if they are, it is in a computer science course that focuses more on programming than scientific problem solving. The outcomes are positive, with greater student computational competence and greater understanding of the physical concepts. Although this study focuses on high school students, it

still provides a useful approach to introducing physics students to programming that can be adapted for training students at the undergraduate level. This paper is used in an activity where participants discuss how to introduce undergraduate students in physics to programming.

Kluth, S., Pia, M. G., Schoerner-Sadenius, T., & Steinbach, P. (2015). How do particle physicists learn the programming concepts they need? *Journal of Physics: Conference Series*, 664(6), 062048. <https://doi.org/10.1088/1742-6596/664/6/062048>

This paper examines a training program for particle physicists and students in programming. As particle physics involves particle interaction simulations and analysis of huge datasets, a strong understanding of computing and programming is absolutely imperative in this field. Although most of the participants in this program have programming skills, this workshop aims to improve their capacity by training them in more high-level methods such as Test-Driven Development and object-oriented programming, boasting positive feedback. Although the training program itself is intended for professionals, covering many topics too advanced for lower year undergraduate students, the structure of the program and methods employed are worth considering. Additionally, entering undergraduate physics students have a wide range of experience when it comes to programming – it is important to be able to accommodate those with no prior experience, as well as those with significant experience who want to apply this skillset to physical applications.

Lockwood, J., & Mooney, A. (2018). Computational Thinking in Secondary Education: Where does it fit? A systematic literary review. *International Journal of Computer Science Education in Schools*, 2(1). <https://doi.org/10.21585/ijcses.v2i1.26>

This recent paper is a comprehensive literature review of efforts to promote “computational thinking” across many disciplines and age groups. Despite its generality, it does cover how to incorporate programming into physics courses at the undergraduate level. This review is particularly valuable, as it provides a list of programs and the corresponding tools developed to teach computational skills and programming to students. This content is used in the discussion on how to introduce programming to students, as well as in providing supplemental resources for the participants’ reference.

Weatherford, S. A. (2011). *Student Use of Physics to Make Sense of Incomplete but Functional VPython Programs in a Lab Setting* (Unpublished doctoral dissertation). North Carolina State University, Raleigh, NC.

This PhD dissertation discusses a pilot study where basic computational programming activities were added to an introductory undergraduate physics course. Instead of requiring students to create original programs, the author developed assessments asking students to analyze programs that are functional, but incomplete. The students must draw on their understanding of physical principles to determine the parts of the program that are incorrect or missing. This approach reinforces students’ physics problem solving skills while also familiarizing them with programming, under limited time constraints. This approach is covered in more detail in the

Integrating programming into physics course: Using “skeleton” codes section of this workshop, and is also the motivation for Activity #2.

WORKSHOP CONTENT AND ORGANIZATION

DURATION (min)	SUBJECT	ACTIVITY	PURPOSE
5	Introduction	Lecture/Discussion: Briefly gauge the participants’ teaching and programming backgrounds. Ask the participants why they chose to attend the workshop, and what they hope to gain from it. Introduce the intended learning outcomes and the schedule.	To get a feel for participant motivation for attending the workshop.
5	Importance of Computer Programming	Lecture/Discussion: Ask participants to list why they think programming is important, leading into a brief lecture that outlines the significance of students learning programming, along with evidence of the benefits in securing jobs and improving physics problem solving skills (see Aiken et al., 2013, Undreiu et al., 2008, Weatherford, 2011).	To outline the value of programming in physics courses, backed by supporting evidence from physics education research papers.
15 (Activity) 5 (Lecture)	Introducing Physics Students to Programming	Activity 1: Ask participants to break into small groups to discuss the Aiken et al. (2013) paper and answer the following questions: <ol style="list-style-type: none"> 1) How do the authors introduce students to programming? 2) Was their method successful? Why or why not? 3) Do you have any concerns with their methods? 4) How would you adapt this approach for 1st year physics undergraduates? Lecture/Discussion: Call on volunteers for answers to the questions listed for Activity 1, leading into a discussion on some of the methods for training physics students	To provide some potential methods for appropriately introducing and teaching programming to students. To encourage participants to actively engage in discussion with each other.

		in a programming language (see Lockwood & Mooney, 2017, Redish & Wilson, 1993, Undreiu et al., 2008, Walck, 2015).	
5 (Lecture) 15 (Activity) 10 (Discussion)	Integrating Programming into Physics Courses	<p>Lecture: Introduce the use of “skeleton” codes (codes missing a few crucial pieces that the student must complete), using explanations and examples from Weatherford (2011).</p> <p>Activity 2: Break participants into groups (at least one laptop per group) and access the skeleton code distributed beforehand. They will then work together to fill in the missing content (see Appendix B).</p> <p>Discussion: Ask groups to come together and critique the activity (describe how valuable they think it would be for students; discuss any flaws and things that could be improved), and to discuss how a similar activity could be developed using another programming language of their choice.</p>	<p>To offer a specific method for establishing student programming skills in small bits (as learning to program is challenging and time-consuming), while also promoting their problem solving and conceptual physics skills.</p> <p>To give participants practice working with VPython, which is an ideal language for training those with little prior technical experience.</p>
5 (Lecture) 15 (Activity) 5 (Discussion)	Developing Programming- based Assessments	<p>Lecture: Once students have mastered the basics of the programming language, they can improve their skillset by constructing original programs to help solve more difficult problems in physics. Introduce programming synthesis assessments (homework assignments and projects requiring students to develop original code to solve the conceptual problems). See Appendix C for an example assessment.</p> <p>Activity 3: Participants will break into small groups to brainstorm a task for assessing student understanding of physics</p>	To brainstorm ideas for physics problems that can be solved using computational analysis.

		material and programming skills by filling in the handout in Appendix D. Discussion: The groups will come together again, with some volunteering to share what ideas they came up with.	
5	Summary and Conclusion	Discussion: Summarize the key points of the workshop, reiterating the importance of programming for physics students' intellectual and technical development. Address any questions and make available a list of relevant resources (e.g., Lockwood & Mooney, 2017).	To summarize the findings from the workshop and offer additional resources participants can use in the pursuit of integrating programming into their courses.
Total Time: 90 minutes			

PRESENTATION STRATEGIES

Workshop Room Requirements

The workshop will involve a digital presentation and should be held in a room with a projector, screen, and AV connections. Since participants will be encouraged to bring their personal laptops, the room should also have Wi-Fi access and ample outlets. It would be ideal to have tables set up in pods to better accommodate the planned activities.

A Few Days Before the Workshop

The workshop facilitator or coordinator should email the prospective participants to inform them about the prerequisite tasks. Before the workshop, participants should read the Aiken et al. (2013) paper and be prepared to discuss it in small groups. Participants should also download VPython (vpython.org/contents/download_windows.html) onto their laptop and bring it to the workshop. VPython is a version of the Python programming language with built-in visual modules, which makes it easy to generate 3D graphics for simulating physical conditions. This is an open source software that works for Windows, Mac, and Linux operating systems. (Visit vpython.org for more details). The facilitator should also send out a VPython script that will be used in Activity 2.

Lectures and Discussion

The facilitator will present a few short lectures over the course of the workshop. The lectures are intended to introduce some key topics, such as computational thinking, using “skeleton” codes, and creating programming-based physics assessments. Discussions, generally held after the activities, will require more active engagement from the participants, including independent reflection and small/whole group discussions.

Activities

The activities are intended to be the highlight of the workshop, where participants can engage in active learning. The first activity is a small group discussion that focuses on the prerequisite reading. The facilitator should post the Activity 1 questions, urging participants to contemplate the questions independently and then work in small groups to discuss the paper and the corresponding questions. At the end of the activity, participants will have a chance to share their reflections with the larger group.

The second activity will involve the participants breaking into groups to work on the “skeleton” code. Group members will retrieve their laptops and load VPython as well as the script distributed beforehand. The groups can choose to share one laptop or each use their own; in either case, they are encouraged to discuss the problem. While the groups are working, the facilitator will walk around the room in case the groups have questions or difficulties. After the activity, individuals can share their reflections on the experience with the larger group. VPython was chosen for this activity, as it is easy to use, open source, and has proven to be an effective tool for connecting an understanding of physics with computational capabilities (Aiken et al., 2013; Lockwood & Mooney, 2018; Weatherford, 2011). The facilitator may also choose to adapt this activity to employ another language, or even multiple languages, if they desire. However, it is recommended that the language used is open source (so it can be easily accessed by anyone attending the workshop), has graphing applications, and has access to scientific libraries.

For the final activity, small groups brainstorm a scenario that can be represented using basic physical principles, such as tracking projectiles or charged particle interactions. The group members would then consider the governing physics concepts in this setup, and come up with questions for the student, which require a computational solution. During this time, group members will fill in the handout (see Appendix D) for personal reference. The facilitator should walk around the room fielding questions. Afterward, groups can share their ideas, if they are comfortable.

Extending the Workshop

This workshop can easily be expanded to 120 minutes if there is interest in pursuing more in-depth discussions, or if participants wish to have more time to work on the activities.

ADDITIONAL REFERENCES

- Redish, E. F., & Wilson, J. M. (1993). Student programming in the introductory physics course: M.U.P.P.E.T. *American Journal of Physics*, 61(3), 222-232. doi:[10.1119/1.17295](https://doi.org/10.1119/1.17295)
- Undreiu, L., Schuster, D., & Undreiu, A. (2008). Interactive problem solving tutorials through visual programming. *AIP Conference Proceedings*, 215-218. doi:[10.1063/1.3021258](https://doi.org/10.1063/1.3021258)
- Walck, S. N. (2014). Learn physics by programming in Haskell. *Electronic Proceedings in Theoretical Computer Science*, 170, 67-77. doi:[10.4204/eptcs.170.5](https://doi.org/10.4204/eptcs.170.5)

APPENDIX A: Introducing Physics Students to Programming - Activity 1

Participants will read the following paper prior to attending the workshop:

Aiken, J. M., Caballero, M. D., Douglas, S. S., Burk, J. B., Scanlon, E. M., Thoms, B. D., & Schatz, M. F. (2013). Understanding student computational thinking with computational modeling. *PERC Proceedings*, 1513, 46-49. doi:[10.1063/1.4789648](https://doi.org/10.1063/1.4789648)

During the workshop, break participants into small groups to discuss the following questions:

- 1) How do the authors introduce students to programming?
- 2) Was their method successful? Why or why not?
- 3) Do you have any concerns with their methods?
- 4) How would you adapt this approach for 1st year physics undergrads?

Key talking points for the facilitator:

- High school students are capable of using computational thinking to address questions in physics.
- Students who were able to generate an animation of a baseball's motion with the correct physical result possessed an understanding of the causal connection between force and motion, as well as a computational iterative view of the changes in motion.
- Students who were able to generate an animation of the baseball's motion but with an incorrect result possessed an iterative view, but were able to clearly connect force with motion.
- "While it is important for students to write programs correctly, programming is not computational thinking," (Aiken et al. 2013, pp. 2). Computational thinking requires the ability to fully comprehend the connection between physical processes (in this case, the relation between force and motion) and the iterative process (how the computer is able to represent the physical process iteratively).

APPENDIX B: Integrating Programming into Physics Courses - Sample Script for Activity 2

The code snippet below, adapted from a sample program from the VPython library, simulates a ball bouncing:

```
from visual import *
#Creates floor object
floor = box(length=10, height=0.5, width=4, color=color.blue)
#Creates ball object and initial velocity
ball = sphere(pos=(0,4,0), color=color.red)
ball.velocity = vector(0,-1,0)
#Time step
dt = 0.01
while 1:
    rate(100)
    #Position equation
    ball.pos = ball.pos + ball.velocity*dt
    #Velocity equation
    if ball.y < 1:
        ball.velocity.y = -ball.velocity.y
    else:
        ball.velocity.y = ball.velocity.y
```

If one runs this program, they will see the ball drop, hit the floor, and then bounce up. However, one should notice that the ball is moving at a constant speed – surely it should accelerate as it is being dropped! Looking at the code, one sees that the velocity equation is constant. To properly account for gravity, the velocity equation should be $v_f = v_i - g \cdot t$ instead of $v_f = v_i$. Thus, this can be fixed by simply changing the last line of code to:

```
ball.velocity.y = ball.velocity.y - 9.8*dt
```

Running the program again shows the ball bouncing more naturally. This can also be adapted to account for air resistance.

APPENDIX C: Developing Programming-based Assessments - Programming Synthesis Assessment Example

Below is an example of a problem to assess student understanding of basic physics (i.e., motion and forces), as well as their ability to synthesize code to accomplish these tasks. This can be a homework problem or part of a project. It can be solved using any programming language that has graphing capabilities.

1. You are a golfer who can hit a golf ball with an initial speed of 160 km/hr.
 - a. Ignoring air resistance, compute and graph the ball's height, velocity, acceleration, and horizontal distance as a function of time until the ball lands. Assume the ball is launched at an angle of 30° .
 - b. You want to hit the ball as far as possible. How would changing the launch angle affect the range? Using the code from part a, iterate for launch angles between 5° and 60° .
 - c. Now, repeat parts *a* and *b*, but include air resistance. Assume the golf ball is a sphere with a diameter of the ball is 4cm. How does this change the maximum range of the ball? Explain conceptually why this change occurs?

APPENDIX D: Developing Programming-based Assessments - Handout for Activity 3

In small groups, you will consider a problem that can be solved by introductory physics students using a program they have created or modified.

1. Think of a topic from an introductory physics course (e.g., projectile motion, charged particle motion in a magnetic field).
2. Identify a problem related to the above topic that can be solved via programming.
3. What could the structure of the code be (can use pseudocode here to outline it)?
4. How would you evaluate the accuracy and/or quality of the student's solution?
5. Do you foresee any issues with assigning an assessment like this to your students? If so, explain.