4-26-2021

Improving Skills in Computer Methods: Introductory Toolkit to Python for Undergraduate Physics Majors

Erin O'Donnell
Utah State University

Melissa Rasmussen
Utah State University

Improving Skills in Computer Methods: Introductory Toolkit to Python for Undergraduate Physics Majors.

Erin O'Donnell, Melissa Rasmussen, Dr. Maria Rodriguez
Department of Physics
Utah State University, Logan, Utah
April 26, 2021

The computer-based curriculum for undergraduate physics students before 2019 struggled to stay current and applicable. When the course was created in the early 2000s, MathCad was used daily by many physicists. As computer's and their computational abilities have grown exponentially, so have programming languages and applications. Now in the late 2010s/early 2020s, Excel, Python, and Mathematica are some of the most common computer appliques. To address this, we created toolkits to give undergraduate physicists the experience for future classes, careers, and graduate school. This paper focuses on the process for creating the Python toolkit and overall class setup. Run on Google Colabratory, no external programs need to be downloaded making the toolkit accessible to all with internet access. The toolkits the main points are: Functions, Loops, Downloading Data, etc. We based examples and problems on physics the students previously learned in their prerequisite classes (Snell's law, star classification, kinematics). After employing the toolkit in PHYS 2500: Computer Methods at Utah State University for Fall 2019 and Fall 2020, student knowledge on coding, and coding physics, significantly improved and overall satisfactory of the course improved.

I. Introduction

Teaching programming with a physics angle is a 3-sided issue. The first is access. Post-secondary access to education is still a barrier to many low-income students. Students exit high school with different levels of computer literacy and understanding of instruction, which may deter them from attending university [1]. When students with lower computer literacy begin taking classes, they are already behind their peers [2]. The second is that undergraduate students with higher computer literacy have rarely been introduced to the oft-used competency of handling large datasets, a.k.a. Big Data [3].

Whether undergraduates choose to directly enter the work force or attend graduate school, they will be faced with large amounts of data to make sense of. Introducing intermediate programming at this level expands students understanding and power of solving problems [4]. The last is the overwhelming amount of information taught in introductory computer classes. Unless students are planning to be a computer science major or minor, most of what they learn will be forgotten and unused. Is programming even needed to be a physicist?

The answer is yes [5]. Each of these obstacles contributes to the difficulty of determining how to teach big data to students with various computer backgrounds and necessities. The Toolkit I created with

Melissa Rasmussen through the guidance of Dr. Maria Rodriguez and funding from Utah State University's Department of Physics Howard L. Blood Scholarship and the National Science Foundation solves this issue.

II. Background

In March of 2019, Erin Rickenbach and I were chosen, supported, and funded by Dr. Maria Rodriguez and Dr. Oscar Varela to attended the Jurgen Ehlers Spring School for Undergraduate Physicists at the Max Planck Institute in Germany. We spent two weeks learning the basis of general relativity. In the second week, we completed multiple programming assignments beginning with the basics of Python. The end goal was to be able to find the chirp mass of Nobel-Prize winning Gravitational Wave 150914. To share files easier (and save time on downloading an environment), the institute used a cloud-based environment to run Jupyter Notebooks called Notebooks Azure. This gave the freedom of being able to work on multiple operating systems. By having previous programming knowledge, I was able to understand and gain knowledge from these assignments. The students with little or no programming knowledge struggled to understand the python portion of the assignments. They understood the physics behind gravitational waves but did not understand how the code connected. In previous coding classes I took (PHYS 2500: Computer Methods in Physics, CS1400: Introduction to Computer Science) I recognized the same trend among physicists.

In Fall 2017, Utah State University implemented a new programming language to be taught for their Introduction to Computer Science: Python. The previous Introduction to Computer Science I took (Spring 2017) was taught C++. Both courses taught the basics of programming: opening pop-up windows, fractal patterns, binary search trees, etc. None of these subjects I found were applicable to my major. They seemed like busy work compared to the programming I wanted to know. Looking over the new Python curriculum and hearing feedback from my colleagues, Python seemed to be a universal starter language everyone loved and more applicable to physicists than C++. As I took Computer Methods, a computer course teaching MathCad within the Department of Physics, my second year I discovered the same dissonance between learning programming and learning useful programming. I asked myself, "How can we learn programming applicable to us as physicists? What programming would I want to know to help my future research? What computer skills will make me marketable in the work force?"

The problems encountered at the school and my own programming background inspired this project which aims to:

- Allow students to learn and work at their own pace,
- Merge the lines between teaching computer programming and teaching physics,
- Provide self-guided lessons appropriate for those with no programming knowledge,
- Give students who already know Python a challenge,
- Create a rotational curriculum easily altered for the future discoveries,
- Introduce students to Big Data manipulation, and
- Be accessible to students no matter the technological background.

After returning from Germany, Dr. Maria Rodriguez and I expanded upon my ideas and concerns. After a month of brainstorming, we finalized the project and my application to the Howard. L. Blood Scholarship. These lead to the creation of the Toolkit(s) and its subsequent use starting in Fall 2019 for PHYS 2500, Computer Methods in Physics (see Figure 1 for access). For the Excel and Mathematica toolkits, Dr. Rodriguez created the material. For the Fall 2020 iteration of the course, it was agreed to form the toolkits in the same style as the Python Toolkit. Collaboration on the updated material existed between Dr. Rodriguez, graduate student Jacob Ciafre, and myself. We met once a month during the summer and into the school year beginning in June 2020.

Fig 1. This QR Code links to the Fall 2020 version of the Python Introductory Toolkit in Google Colaboratory.

III. Creation Process

In the Sumer of 2019, Melissa Rasmussen, a Physics and Computer Science double major, and I began collaboration with Dr. Maria. Rodriguez's guidance. We based the Toolkit loosely on the assignments from the Spring School. The same Cloud-based program (Notebooks Azure) was used to run the Toolkit. We stuck with this cloud-based format to benefit student's relationship with technology after gaining feedback.

Since everything can be saved within the online-environment and compatible with all operating systems, students did not need to go out of their way to purchase a new laptop or tablet. Even further, some students were using their phones in class to code, or public computers in the school library. This availability of multiple sources allowed students to find what worked best for them and be able to fully focus on class material. To help aid the students access Notebooks Azure, Melissa Rasmussen created a one page pdf walking students through the steps of being able to use Notebooks Azure.

For 12 weeks beginning in the second week of May, Melissa and I met once a week on Skype, and then once a week also with Dr. Rodriguez until the project was finished. Relying on our previous coding knowledge and several coding textbooks, we condensed a semester's worth of Python into a 12-15 hour course (depending on skill level) [6, 7, 8]. Melissa and I browsed our previous physics textbooks and coding assignments to curate a list of topics undergraduate students should know the most in Python [9, 10, 11]. These topics were: Variables and computations, Functions, Loops, Libraries, and Files. Each broad topic became its own assignment broken into subsections for easy learning.

We designed each lesson around the concept of *learning by doing*: students interact with the material after each new concept is introduced [12]. This style of lesson is illustrated in Figure 2. Hands-on learning allowed the students to engage their problem-solving skills directly and better retain class material. Another benefit was that students were able to discover what material they were struggling on immediately rather than during assignments.

The first assignment taught the basics of Python syntax so we assigned rudimentary problems like simple algebra and changing variable types. To help students with the plethora of short math examples, a

"cheat sheet" of mathematical and common functions were given to students to assist them. This allowed for the overall length of the toolkit to be shorter and allow the students to have a list to refer to directly instead of the internet. Further along, we included more difficult problems (See Figure 3 and 4) that were the Python equivalent of classic physics homework problems [7][10].

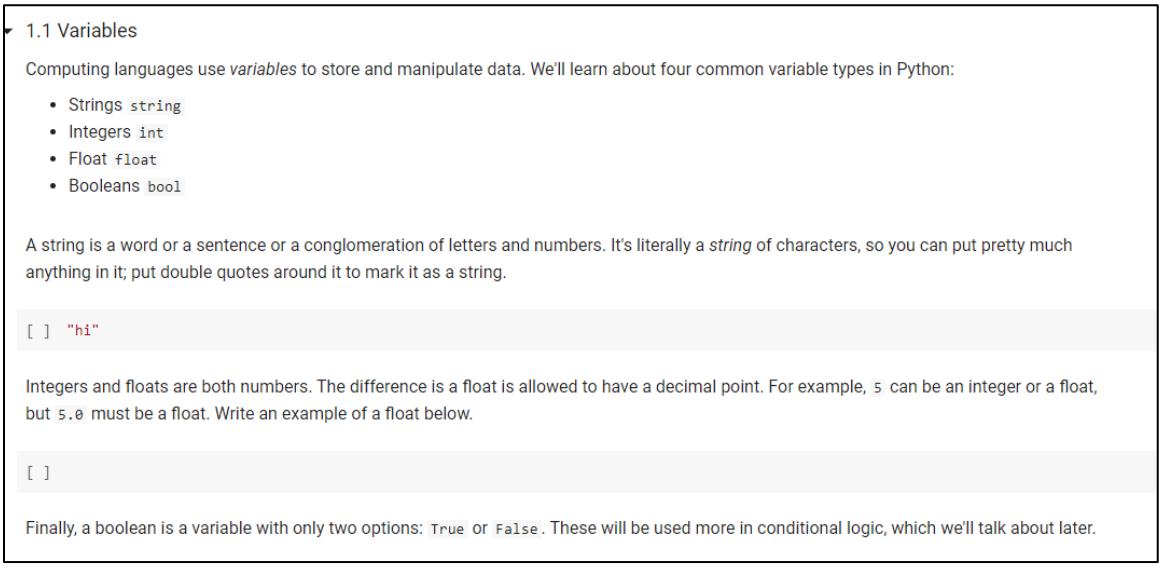

Fig 2. This section teaching the concept of variables exemplifies the learning by doing *strategy. Students first learn about types of variables in Python and then are prompted to write a certain type.*

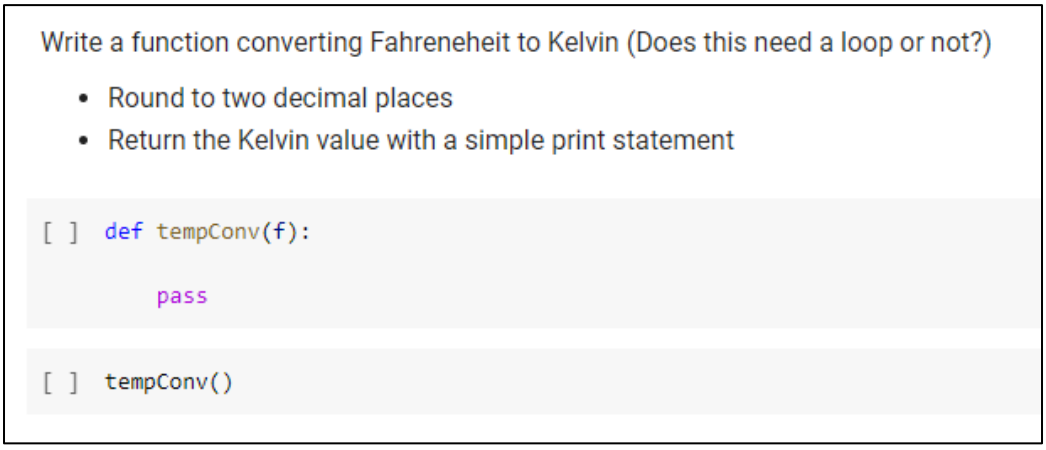

Fig 3. A problem demonstrating the conversion between Fahrenheit and Celsius in Assignment 3, Loops.

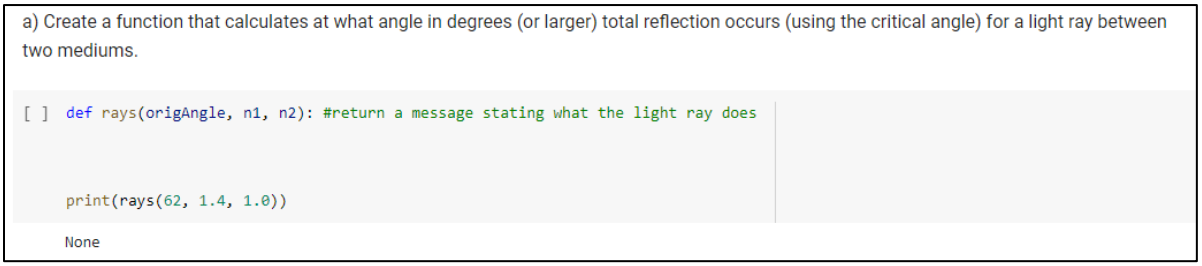

Fig 4. A Problem that asks students to calculate what angle total reflection occurs for a light ray between two media in Assignment 2, Functions and Conditionals.

Since the Spring School was one of the motivations for the project, we adapted the final gravitational wave problem to fit the toolkit. We made sure all functions and necessary syntax needed to complete this problem were taught. After all the assignments were completed, the following gravitational wave problem was given (See Figure 5, 6, and 7 for examples).

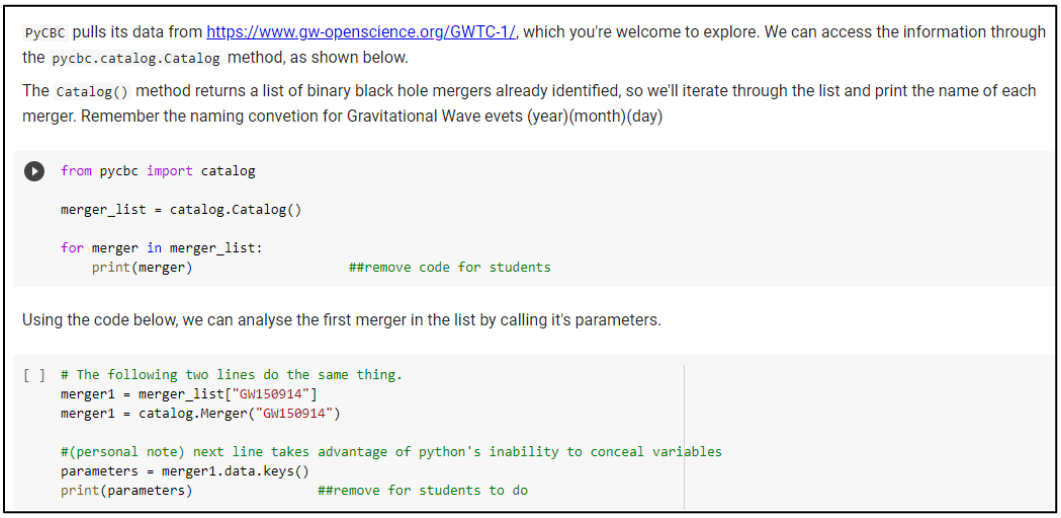

Fig 5. Shows students how to access the LIGO Data of the 150914 gravitational wave. This code was given at Jurgen Ehlers Spring School in 2019. A Green comment is written in the first block of code shows the change made for the students of the Fall 2019 Computer Methods class.

Filters

There's no signal that we can recognize in the data above, so we'll have to run it through a series of filters below.

```
[ ]  filtered = {}

     for ifo in data:
         # this adds white noise to data
         filtered[ifo] = data[ifo].whiten(4,4)

         # we know the frequency of our signal is between 30 Hz and 250 Hz
         # this removes frequencies below 30 Hz
         filtered[ifo] = filtered[ifo].highpass_fir(30, 512)
         # this removes frequencies above 250 Hz
         filtered[ifo] = filtered[ifo].lowpass_fir(250, 512)
```

Fig 6. A fully written code showing students how to filter the data and which lines are responsible for what filtering.

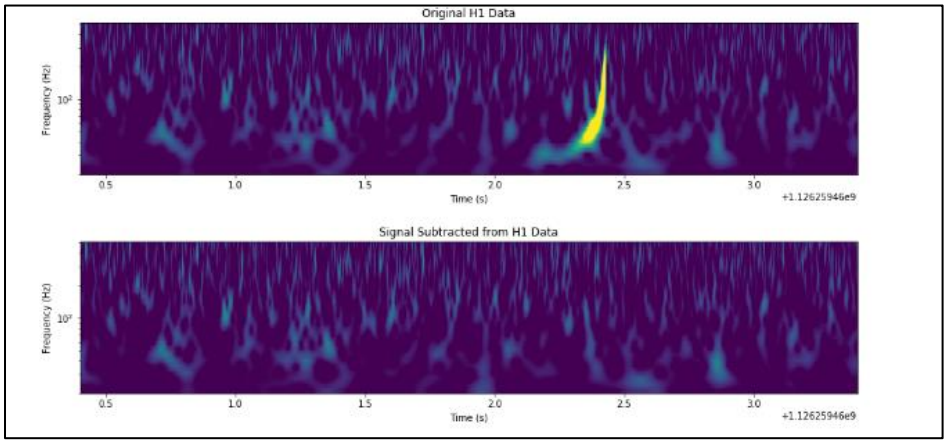

Fig 7. After filtering, the chirp mass of the gravitational wave is found. The top half is the signal with the wave. The bottom figure shows the noise within the signal.

For the 2020 school year and following the retirement of Notebooks Azure on October 9th, 2020, we transferred the Python Toolkit from Notebooks Azure to Google Colabratory. Given the time constraints, skill level, and change in teaching environment, the gravitational wave problem was retired for Fall 2020 and future classes.

IV. Application in the Classroom

For both Fall 2019 and Fall 2020, PHYS 2500 was split evenly into 4 curriculum sections based on the toolkits created: Excel, Python, Mathematica, and Final Project. To follow the *learning by doing* approach, assignments were created with both descriptions and hands-on learning. The Excel semi-toolkit

has students manipulating “small” sample sizes and creating different types of graphs. Mathematica was run on a cloud-based version called Wolfram Cloud and introduced students to helpful mathematical tools such as summation calculations for future classes and careers. Like the Python Toolkit, Students were given math problems from prerequisite classes and shown how to transfer them to Mathematica. Instructions were given in a more traditional way by listening to lecture (lectures were posted later that day) and then doing the separate, subsequent assignments. The final project gave students a chance to see how much they improved from the beginning of the course. The final projects were chosen at random for each student. The goal was to solve Intermediate-level Physics problems using either Python or Mathematica. After presenting their code and findings, students can see the value in using different languages for different tasks.

In Fall of 2019, classes were twice a week, followed by office hours. Each class was an hour, with half an hour of lecture, and half hour to work on the assigned problems within the Toolkit. The second half of the class allows students to work at their own pace, while being able to ask the undergraduate TA (Rob Smith), graduate TA (John Mojica), or professor (Dr. Maria Rodriguez) questions as they get stuck. As students asked questions, they were able to be guided through Low vs High order questions to further their comprehension on the subject matter [13].

In Fall 2020, COVID-19 necessitated classes be held remotely over Zoom. This required a modified structure of the syllabus and material. Each class still began with lecture, however during the second half of class, students were split into three breakout rooms. Each breakout room was led by either the undergraduate TA (Erin O’Donnell), the graduate TA (Jacob Ciafre) or the professor (Dr. Maria Rodriguez). The separate rooms consisted of 3-4 kids each, depending on how many attended remotely. While the classroom setup, curriculum speed and programming environment changed, the overall lecture setup and material was the same. The professor for Computer Methods, Dr. Rodriguez found Google Colaboratory much easier to use than Notebooks Azure. Due to the popularity of Google products, both students and teachers appeared to be more familiar with the new environment. Students were able to log in using their university or personal email and save all their work directly in the application. With Notebooks Azure, students had to create a fresh account, and go through multiple steps to upload the file and download the correct file. The switch reduced the initial Python setup time from an hour in Fall 2019 to 20 minutes in Fall 2020.

Seven lectures (eight for Fall 2020) were set aside to teach Python, accompanied by five assignments. The week’s assignments were given out on Monday lectures and due the following Monday. Lecture for both semesters including working through a few problems from the assignment in real time, and students were encouraged to finish the assignments directly after lecture. Given the independent nature of the Toolkit, students were able to finish the extra problems during the second half of the lecture or by the evening.

V. Results

Compared to semesters prior to Fall 2019, overall classroom attitude improved significantly. As the undergraduate TA in Fall 2020, I directly interacted with students. I heard their concerns and saw real-time how the students were interacting with the Toolkit(s). They did not know I established the Python portion of the curriculum or that I contributed to the other sections. To the students I was simply a TA

who took the course previously. From Fall 2019 and Fall 2020, a qualitative trend is seen within the open-ended comment section of the *End of Semester IDEA Course Survey*'s sent out by USU administration each semester (See Figure 8 below).

Fall 2019	Fall 2020
Enjoyed the Introductions to the various angles we could approach same problem different ways	Lots of good material and great ways to learn. Skills already have come in handy in jobs.
Relevant to real world scenarios, applicable to future careers.	Necessary class to learn how to use the tools that my future classes will use.
All the projects can be references for later	Always had resources to turn to

Fig 8. Qualitative comments answering the question: What aspects of the teaching or content of this course do you feel were especially good? From End-of-semester IDEA surveys. Both semesters had the same positive topics to comment on, showing a positive trend in the satisfactory of PHYS 2500.

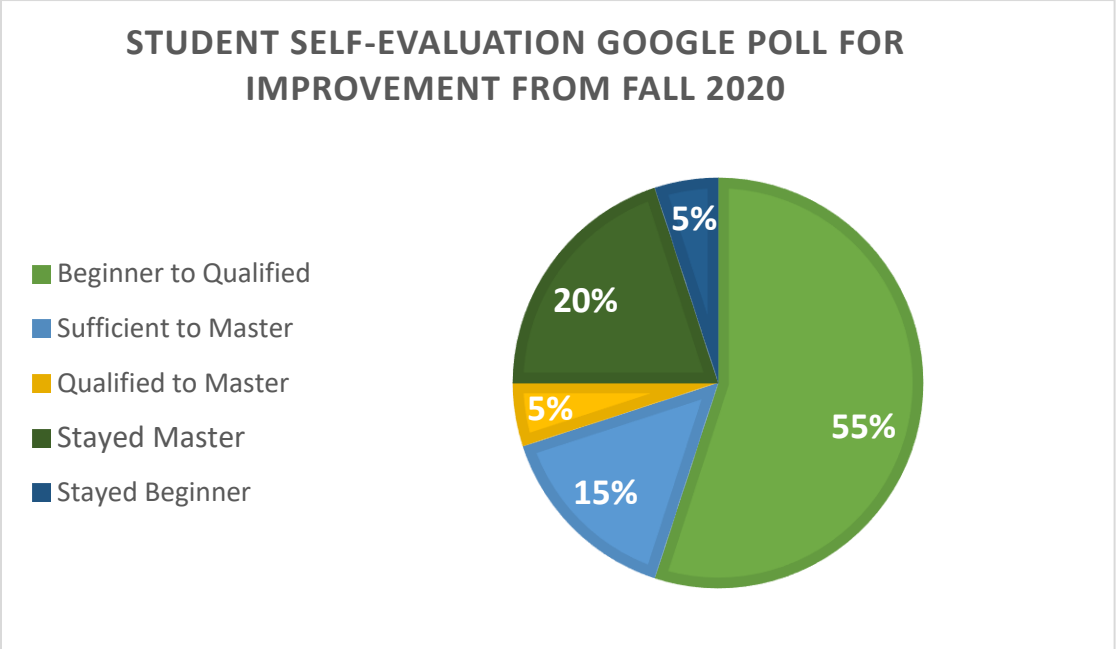

Fig95. Student Self Evaluation Polls on how their skills changed over the Python section of the course using the scale Beginner, Sufficient, Qualified and Master.

Thirteen out of fourteen students in the Fall 2020 iteration of Computer Methods answered a short google poll at the beginning of the Python Toolkit and at the end of the Final Project. Figure 9 shows the results of this poll. Over 75% felt they improved their Python skills: 55% of students transitioned from Beginner to Qualified, 15% from Sufficient to Master, and 5% from Qualified to Master. From the 25% that responded they felt they did not improve, 20% had "Mastered" Python and one single student responded their Python skills stayed at Beginner. The students who had previously experience with Python expressed that the Python section allowed them to visualize how physicists apply coding to their specific research.

Computer methods is a prerequisite for Intermediate and Advanced Lab here at USU. The lab professor noted the students who had taken the Fall 2019 version of Computer Methods had increased fluency in Python and Excel compared to previous semesters. Since the previous semesters of Computer Methods taught MathCad, students were learning Excel and programming for the first time. In Fall 2020 Intermediate Lab, students spent more time focused on learning lab materials rather than the data applications.

VI. Looking Forward

While the goals given at the beginning were base achievements to strive for, there are still objectives for the Toolkit (and the whole class) to meet over the next few years. The goals of the Toolkit that were met were allowed students to work at their own pace and allow the toolkits to be accessible to all students. Some students enjoyed that they could completely work on their own while others loved that they could choose between group work and pairing up. This style of learning allowed students to work in their own comfortable environment. For the future, this is the most important goal to keep.

Tested on multiple student's devices, school computers, and cellphones, the Python Toolkits can be accessed if there is an internet connection. Whether students are typing on a cellphone, tablet or keyboard all students had an equal chance at all materials. Being able to learn in this kind of environment gave students the confidence to ask questions without hesitation. For the other two toolkits, students had the same satisfaction.

I do believe the lines between teaching computer programming and teaching physics were merged the past two years but I am hoping for them to completely homogenize. To homogenize, the rest of the objectives set for the Introductory Python Toolkit should be satisfied. This can be done with a second toolkit: Intermediate Python Toolkit. In this secondary toolkit, students would be learning new physics at the same time as learning new code to allow for homogenization and muscle memory to develop. Big data manipulation can also be introduced. Data can be download from multiple research labs across the world and used. Students who already have a background in Python/coding could take this course instead to give themselves a challenge. It would also challenge students who have worked through the Introductory Python Toolkit. The toolkit could introduce upper-level topics such as neutrino detection, chemical analysis, partial differential equations, and gravitational waves.

VII . Acknowledgements

I would like to give a big thank you to the Department of Physics for funding this research by awarding me with the Howard L. Blood Scholarship, as well as Dr. Rodriguez for providing Melissa with NSF funding for her work as well. Travel to and from the Undergraduate Jurgen Ehler's Spring School was graciously supported by Dr. Maria Rodriguez and Dr. Oscar Varela

---

[1] Rebecca D. Cox (2016) *Complicating Conditions: Obstacles and Interruptions to Low-Income Students' College "Choices*", The Journal of Higher Education, 87:1, 1-26, DOI: 10.1080/00221546.2016.117777

[2] William A Howe, P. L. (2012). *Becoming a Multicultural Educator.* Sage Publications Inc.

[3] Lin, X.-W. C. (2014, April 20). *Big Data Deep Learning: Challenges and Perspectives. IEEE Access*, pp. 514-525.

[4] Edward F. Redish, J. M. W. (1993) *Student programming in the introductory physics course: M.U.P.P.E.T*, American Journal of Physics, 61:222, 222-232, DOI: 10.1119/1.17295

[5] F. James, (1986) *Do Physicists Need Software Engineering?.* Computer Physics Communications, 41, 205-216

[6] John P. Flynt, D. K. (2012). *Mathematics & Physics for Programmers* . Course Technology, Cengage Learning.

[7] Liang, Y. D. (2013). *Introduction to Programming using Python.* Pearson.

[8] Springer. (2020). *Essential Python for the Physicists* . Springer.

[9] Moche, D. L. (2015). *Astronomy, A Self - Teaching Guide.* Nashville: Turner Publishing.

[10] Pearson. (2017). *Physics for Scientists and Engineers.* Pearson.

[11] Ratcliffe, M. (2009). *Cosmology and the Evolution of the Universe.* Greenwood Press.

[12]Yuchiro Anzai, H. A. S. (1979). *The Theory of Learning by Doing*. Psychological Review, 86: 2, 124-140.

[13] Juliana Saxton, C. M. (2018). *Asking Better Questions: Teaching and Learning for a Changing World*. Pembroke Publishers.