Infusing Fundamental Competencies of Computational Science to the General Undergraduate Curriculum

Ana C. González-Ríos
University of Puerto Rico - Mayagüez
anacarmen.gonzalez@upr.edu

ABSTRACT
The growing need for a workforce that can analyze, model, and interpret real-world data strongly points to the importance of imparting fundamental concepts of computational and data science to the current student generation regardless of their intended majors. This paper describes the experiences in developing and implementing a course in computation, modeling, and simulation. The main goal of the course was to infuse fundamental competencies of computational science to the undergraduate curriculum. The course also aimed at making students aware that modeling and simulation have become an essential part of the research and development process in the sciences, social sciences, and engineering. The course was targeted to students of all majors.

Keywords
Computational science, Flipped classroom, Computational and data science literacy, General education, Python, Curriculum development

1. INTRODUCTION
Computational science (CS) is an interdisciplinary field that can be defined as the intersection of the domain area of the problem of interest, computer science, and mathematics. It requires mathematical modeling capability and the skills for its efficient implementation using computing techniques. CS allows us to build models, visualize phenomena, and conduct experiments difficult or impossible in the laboratory. So, it plays a critical role in the future of the scientific discovery process and engineering design [11, 8, 12].

It is important to reach not only the undergraduate curriculum in science, technology, engineering, and mathematics (STEM). All undergraduate students should be exposed to the fundamentals of CS. Hence, students can be better equipped to apply CS techniques in their fields and to better understand society and their environment [8, 12].

All students, independent of their area of study, should have access to a course that will help them develop fundamental CS competencies. After reviewing the curriculum at a public Minority Serving Institution, it was observed that the courses that address the topics of modeling, simulation, and data science have multiple advanced pre-requisites, so it is hard for many students to take those courses. Sometimes students who do have the pre-requisites cannot fit the courses in their curricula. It was also noted that these courses are offered randomly depending on enrollment and professor availability.

With the main goal of infusing fundamental concepts of computational science into the undergraduate general education curriculum; we proposed, developed, and implemented a new elective course with only college algebra as a pre-requisite.

All undergraduate students could benefit from this course including students who planned on participating in undergraduate research. Pre-service teachers were also an important target so that they would be better able to integrate CS in their K–12 courses. Computer science students had the opportunity to see how programming skills can be applied to science, social sciences, and engineering. The course could provide a panoramic view of the field and act as a catalyst to continue with more advanced courses.

This paper describes the set of basic learning outcomes upon which the course content rested, course description, course objectives, resources developed, and overall format of the course. Observations on students' perception of the course are also discussed.

2. COURSE LEARNING OBJECTIVES
The primary goal of the course was to introduce basic concepts of computational science to a diverse student body. The aim was not to produce experts in computational science but to provide the tools and skills that can benefit the personal and professional lives of individuals and allow them to better collaborate with computational scientists. With this mindset and the fact that the course would only require college algebra, we proceeded to determine the learning outcomes of the course. The learning outcomes represented the framework for the course content.

The computational science educational competencies that guided the course design derived from the work developed as part of a project at the Ohio Supercomputer Center sponsored by the National Science Foundation [17]. The competencies were created by the participating faculty and then reviewed by a business advisory committee [9].

The group identified seven competency areas shown in Table 1. As shown in Figure 1, each area was further subdivided to describe the level of the skills and knowledge necessary to master the competencies [17].

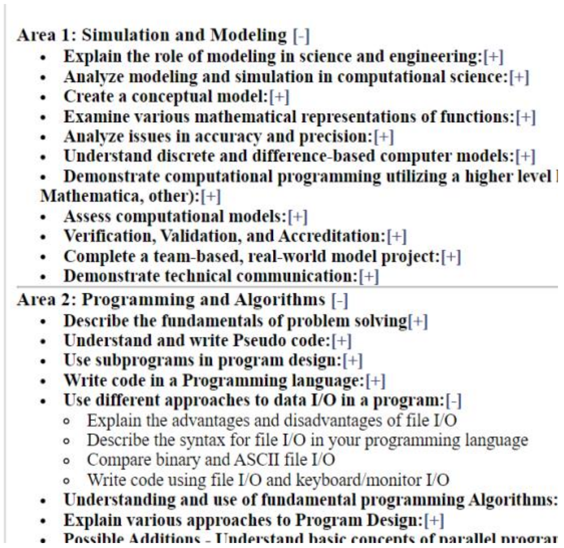
Consistent with [11, 7, 9], we believed that an introduction to computational science must equip learners with a basic understanding and an integration of both modeling and computer programming principles. Learners should be provided with experiences on how to translate the relationships within a system being modeled into a set of mathematical functions that accurately portray the behavior of that system and then translate the

© 2021 Journal of Computational Science Education
DOI: https://doi.org/10.22369/issn.2153-4136/12/3/3

mathematics into computer code that correctly simulates those relationships.

Table 1. Competency areas [17].

1	Simulation and Modeling
2	Programming and Algorithms
3	Differential Equations and Discrete Dynamical Systems
4	Numerical Methods
5	Optimization
6	Parallel Programming
7	Scientific Visualization



Area 1: Simulation and Modeling [-]
- **Explain the role of modeling in science and engineering:**[+]
- **Analyze modeling and simulation in computational science:**[+]
- **Create a conceptual model:**[+]
- **Examine various mathematical representations of functions:**[+]
- **Analyze issues in accuracy and precision:**[+]
- **Understand discrete and difference-based computer models:**[+]
- **Demonstrate computational programming utilizing a higher level**
Mathematica, other):[+]
- **Assess computational models:**[+]
- **Verification, Validation, and Accreditation:**[+]
- **Complete a team-based, real-world model project:**[+]
- **Demonstrate technical communication:**[+]

Area 2: Programming and Algorithms [-]
- **Describe the fundamentals of problem solving**[+]
- **Understand and write Pseudo code:**[+]
- **Use subprograms in program design:**[+]
- **Write code in a Programming language:**[+]
- **Use different approaches to data I/O in a program:**[-]
  - Explain the advantages and disadvantages of file I/O
  - Describe the syntax for file I/O in your programming language
  - Compare binary and ASCII file I/O
  - Write code using file I/O and keyboard/monitor I/O
- **Understanding and use of fundamental programming Algorithms:**
- **Explain various approaches to Program Design:**[+]
- **Possible Additions - Understand basic concepts of parallel progra**

Figure 1. Subset of competency subdivisions [17].

Table 2. Course learning outcomes.

1.	Explain the role of modeling in the sciences and engineering.
2.	Explain the terms of modeling in the sciences and engineering.
3.	Create a conceptual model.
4.	Write code in a programming language.
5.	Use subprograms in program design.
6.	Understand and write pseudocode.
7.	Describe the fundamentals of problem solving.
8.	Use different approaches to data I/O in a program.
9.	Understanding and use of fundamental programming algorithms.
10.	Understand discrete and difference-based computer models.
11.	Understand the use of empirical data.
12.	Understand the modeling process.
13.	Verification and validation.
14.	Technical communication.
15.	Demonstrate computational programming.

We believed that an introductory course should cover the skills and knowledge described in the first two areas: simulation and modeling, programming, and algorithms. The selected learning outcomes as shown in Table 2 represented what the students would be able to do at course completion.

Guided by the HPC University [17] competencies, the learning outcomes as shown in Tables 3a and 3b were further subdivided for a more detailed description of the skills and knowledge that needed to be acquired.

Table 3a. Learning outcomes subdivision.

1	Explain the role of modeling in the sciences and engineering. a) Describe the importance of modeling to science and engineering. b) Describe the history and need for modeling. c) Describe the cost effectiveness of modeling. d) Describe the time-effect of modeling.
2	Explain the terms of modeling in the sciences and engineering. a) Define modeling terms. b) List questions that would check/validate model results. c) Describe future trends and issues in science and engineering. d) Identify specific examples of modeling in science.
3	Create a conceptual model. a) Utilize the modeling process to identify key parameters of a model. b) Estimate model outcomes. c) Use Python to implement the mathematical representation of the model.
4	Write code in a programming language. a) Understand the concept of syntax in a programming language. b) Describe the syntax of the programming language constructs. c) Understand the difference between a compiled and interpreted language. d) Write and run basic programs in the language of choice. e) Understand how to de-bug code. f) Understand the numerical limits of various data types and the implications for numerical accuracy of results.
5	Use subprograms in program design. a) Describe how logical tasks can be implemented as subprograms. b) Explain the control flow when a function is called. c) Explain how function output is used. d) Understand how languages handle passed data into functions and subprograms, especially one- and two-dimensional arrays.
6	Understand and write pseudocode. a) List the basic programming elements of pseudocode. b) Explain the logic behind an if/then/else statement. c) Understand the iterative behavior of loops. d) Describe the difference between several looping constructs. e) Write pseudocode to solve basic problems.

Table 3b. Learning outcomes subdivision.

7	Describe the fundamentals of problem solving. a) Understand top-down thinking and program design. Discuss breaking up a problem into its component tasks. Understand how tasks acquire data. b) Describe how tasks should be ordered. c) Represent tasks in a flow-chart style format.
8	Use different approaches to data I/O in a program. a) Explain the advantages and disadvantages of file I/O. b) Describe the syntax for file I/O in your programming language. c) Write code using file I/O and keyboard/monitor I/O.
9	Understanding and use of fundamental programming algorithms. a) Explain an algorithm as an ordered series of solution steps. b) Describe an algorithm for a simple programming problem. c) Describe what a software library is. d) Construct difference-based computer models.
10	Understand discrete and difference-based computer models. a) Write simple Python programs performing numerical calculations as needed for modeling and simulation. b) Implement finite difference modeling equations and create simulations in Python.
11	Understand the use of empirical data. a) Visualize empirical data and the fitting function using Python. b) Use data science techniques to illustrate data relevant to social change issues and interpret the results.
12	Understand the modeling process. a) Identify different types of models and simulations. b) Describe iterative development of a model. c) Explain use of models and simulation for hypothesis testing.
13	Verification and validation. a) Discuss methods for reviewing models their verification and validation. b) Describe the differences between predictions of model, actual results, and relevance of these differences to the problem. c) Suitability/limits of models.
14	Technical communication. a) Document the development and implementation of a model and present it in oral and written form.
15	Demonstrate computational programming. a) Describe the computational programming system environment. b) Define elementary representations, functions, matrices and arrays, script files. c) Explain relational operations, logical operations, condition statements, loops, debugging programs. d) Create tabular and visual outputs. e) Translate the conceptual models to run with this system and assess the model results.

3. COURSE DESCRIPTION, COURSE OBJECTIVES

Based on the selected learning outcomes, the course description as stated in the University's Course Catalog was as follows:

Introduction to the principles of modeling and simulation; progressive introduction of programming principles and skills using a high-level programming language; application of programming skills to the solution of different classes of models.

The course did not require programming experience and the mathematics pre-requisite was college algebra.

The course objectives were:

1. Provide a background for more advanced modeling courses.
2. Provide the students with an introduction to modeling and its importance to current practices in different subject domains; like science, social sciences, and engineering.
3. Introduce programming principles and apply them to the solution of different classes of models.
4. Provide an overview of the modeling process and the terminology associated with modeling and simulation.
5. Study the mathematical representation of different classes of models.
6. Introduce techniques for fitting a function to an experimental data set.
7. Provide the opportunity for students to document the development and implementation of a model and present it in oral and written from.

4. COURSE ELEMENTS

Tables 4a and 4b show a mapping of the course topics and course learning outcomes. Each topic was covered as a unit that included instructional materials, learning activities, and assessments.

Table 4a. Course topics and learning outcome mapping.

	Topic	Learning Outcomes
1	Introduction to modeling; modeling concepts and definitions	1a,b,c,d 2a,b,c,d 13c 12c
2	Introduction to the Programming Environment	4a,b,c,d,e 9c
3	Deterministic Linear Models	3a,b,c 12a,b 4f
4	Array Mathematics and Python	15a,b 5d
5	Plotting in Python	11a,b 15d
6	Problem solving	7a,b,c 9a,b 6a,e
7	Conditional Statements	6b,15c
8	Iteration and Loops	4b,6c,d,e
9	Nonlinear and Dynamic Models	10a,b 12a,b 9d

Table 4b. Course topics and learning outcome mapping.

10	Estimating Models from Empirical Data	3a,b,c 8a,b,c 11a,b 15d,e
11	Stochastic Models	12a,b
12	Functions in Python	5a,b,c,d
13	Verification, Validation, and Errors	13a,b,c 12c
14	Project implementation, Students are given in-class time to work in groups in their final project presentations	14a

4.1 Course Textbook
The textbook selected for the course was *Introduction to Modeling and Simulation with MATLAB® and Python* by Steven I. Gordon and Brian Guilfoos [11]. The reason for selecting the book was that its content focuses on meeting the set of basic modeling and simulation competencies as defined in HPC University [17], and it uses a just-in-time approach to introduce both programming and modeling concepts.

4.2 Python Programming Language
Python [15] and the Spyder integrated development environment [19] were selected as the platform to develop the programming skills.

Python is a popular general-purpose programming language; it is Free and Open-Source Software that can be acquired and used at no cost. It provides powerful tools and libraries like SciPy and NumPy for scientific computing and Matplotlib and Seaborn for data visualization. Python has a rich web ecosystem of pedagogical resources.

5. PEDAGOGICAL MODEL
The course was designed to combine face-to-face interventions with online, web-enabled strategies, a pedagogical model known as flipped classroom [6]. The flipped classroom switches the typical face-to-face lectures and homework elements of a course. Instead of attending the traditional lecture, students engage with short video presentations or other multimedia content asynchronously before the class period [13]. The in-class period is spent in student-centered learning strategies such as active learning activities like discussions, assignments, laboratories, and mini projects; allowing the instructor to spend more time guiding and supporting the students' progress [16]. This teaching approach has been documented to benefit students' learning outcomes [10, 2, 1].

6. RESOURCES DEVELOPED
The development of all course materials was guided by the course learning outcomes and by taking into consideration the expected students' mathematical background. The created instructional material consisted of videos (equivalent to a lecture but divided in segments of 5-to-15 minutes), practice exercises, walkthroughs, multiple choice questions, and recorded demonstrations. These as well as the course evaluations were made available through the course management system, Moodle [14]. The course and instructional materials were designed following the best practices for using technology and the theory for the creation of courses [5, 4]. The center of resources for distance education, CREAD [3], at the author's institution offered the necessary guidance and support during the design and development of the instructional materials and assessments.

7. COURSE FORMAT
The course met twice a week, Tuesdays and Thursdays, for two hours each time. New material was made available in cycles of one week. A new cycle started at the end of the class period on Thursdays. During the in-class time of the second half of a cycle (Thursdays), the students took a 15-minute, multiple-choice quiz of the material on the current cycle. The idea was to provide an incentive to the students to stay on top of the material.

The students also had to work and submit what was denoted as the module's activity (a list of which is provided in Appendix A). The purpose was to practice and develop the skills introduced in the module. On occasion, the activity was a mini project in which the students organized and developed their work to present/share at the next in-class time. The students were given an outline of what was expected in the presentation, for example, to include four slides and the topic to discuss in each slide.

The students were allowed to work by themselves or in groups of at most three students. They were encouraged to share ideas with one another. The instructor was always available to answer questions and give feedback as needed.

A solution was discussed at the end of the period. On occasion, a student volunteer presented his/her solution to the rest of the class.

During the first half of the cycle (Tuesdays), the students were expected, but not required, to have studied all the online materials. At the beginning of the period, the students could ask questions about the module's materials. The instructor also delivered a short lecture, of at most 15 minutes, that covered core concepts and examples from the videos. This lecture also aimed at focusing and guiding the students on how to study videos and other online materials. Students could spend time working on the practice activities. Practice exercises were similar to what was discussed in the video. The students were also asked to modify the mathematical models or code and to comment on the results. On the other hand, the module's activity and mini project could be described as an open-ended problem. The students needed to recognize the mathematical model and corresponding coding from the videos and other online materials to develop their solution.

7.1 Examples of Course Materials
Some examples of course materials are presented in Appendix C. Individuals interested in reviewing additional materials or resources from other topics are encouraged to contact the author.

8. COURSE EVALUATIONS
Two types of evaluations of the students were offered: those that required the students' individual efforts and those that could be done collaboratively.

The individual evaluations were in the form of timed, multiple-choice questions quizzes. One 15-minute quiz was offered at the end of each cycle. Toward the end of the semester, the students took a 60-minute, multiple-choice exam that covered all the material discussed to the that point. This type of evaluation assessed students' knowledge on terminology and basic concepts.

The collaborative evaluations included the module's activity, the mini projects, and the end of semester project.

9. FINAL PROJECT: COMMENTS
As a final evaluation, students were required to develop, document, and present a project. Keeping in mind that this is a foundational course, the aim of the project was to provide the means for students

consciously to go through all the steps of the modeling process. The objective of the project was to build and test a model of a system and use that model to derive insights into the system behavior [11]. The project specifications indicated that the code be written in Python and that students must use graphs to discuss part of the results.

Two weeks before the end of the semester, the students had to submit a project proposal. In the proposal, the students explained the goals and motivations of the project, why the project was of interest, and what question(s) they would like to answer or what problem(s) they were trying to solve.

To help guide in the project topic selection and development, students were directed to the course textbook where the last chapter provides a list of project topics and a set of reference materials for each topic [11]. Other reference materials were available at Shodor [18]. The students could design, develop, and implement the final project either in groups or individually. A written report and an oral presentation were required. A list of project topics is presented in Appendix B.

10. FINAL DISCUSSION

There have been two offerings of the course, the first during spring 2019 and the second during spring 2020. The first time, it was offered in the format as described in the paper. The second time, due to the COVID-19 pandemic, it was offered completely online. On both occasions, the course was advertised through the University internal e-mail system.

In spring 2019, 16 students registered to take the course, and only one withdrew due to health conditions. For the spring 2020 cohort, there were a total of 13 students. The breakdown by year is shown is Table 5.

Table 5. Year breakdown

Year	Spring 2019	Spring 2020
Freshman	0	0
Sophomore	3	5
Junior	6	7
Senior	7	1

At the beginning of the semester, the students were asked to write two or three paragraphs describing why they had taken the course. Of the spring 2019 group, there was one student from the Physics Department, one student from the Economics Department, and the rest of the students were from the Computer Science program. Their answers to why they were taking the course can be described as follows: the student from the Physics Department mentioned that the course would provide an introduction to more advanced courses in the topic of modeling and simulation and would help develop programming skills. The student from the Economics department mentioned that, according to the search she had done on necessary skills for the modern workforce, this course would help her acquire some of those competencies. Seven of the students from the Computer Science program mentioned that they had taken the course to fulfill a required elective and secondly to learn about Python. The rest of the students also took the course to fulfill an elective requirement and said they were curious about the modeling and simulation topics and felt that learning the skills of problem solving and Python would be advantageous for their careers.

In the spring 2020 group, there were six students from the Physics department, one student from the Business faculty, one student from Mathematics, and the rest of the students from the Computer Science program. Their answers to why they were taking the course can be described as follows. The students from the Physics department were taking the course because it was recommended by other students who had taken the course or had heard about the course. According to their comments, these students were looking for an introductory course in computational science that would provide experience in implementing solutions using a programming language rather than focusing on the programming language itself. The rest of the students were taking the course to fulfill an elective requirement. They also mentioned being interested in learning about the programming language Python.

A list of selected questions from the end-of-semester student survey is shown in Table 6. The answers to the first two questions enumerate the preferred activities and course concepts most frequently mentioned. From the answers to the last three questions, it can be said that by the end of the semester the students preferred the flipped classroom model to face-to-face. It was often mentioned that they appreciated the immediate feedback and guidance from the instructor. The high percentages of affirmative answers to recommending the course to other students and mentioning in the resume having taken the course suggest that they valued and recognized the importance of the skills and concepts acquired through the course.

Table 6. Selected questions.

Questions	Most frequent answers	
1. From the module activities and mini projects, which were the ones that you enjoyed the most?	Concept map, linear models plotting with Python, linear regression, curve fit, modeling process.	
2. List three concepts that you remember from the course.	The Traffic Model, calculating prices of car rentals, compound interest, designing an app for the best gas station selection.	
	Affirmative Answers	
	Spring 2019	Spring 2020
3. Do you prefer the flipped model vs. the face-to-face?	77%	N/A
4. Would you recommend this course to other students?	100%	90%
5. Do you think that you can benefit from mentioning in your resume that you have taken this course?	92%	80%

After offering the course in both the flipped classroom and online modalities, two final observations are described. First, for the end-of-semester project, the students had to submit both written and oral reports. The oral presentation needed to be supported by a PowerPoint (PPT) presentation. For spring 2019, the oral presentation was face-to-face. It was noticed that most of the students for the PPT did a copy and paste of parts of the written report, tended to read from the PPT, and had problems keeping to the presentation time limits. For spring 2020, the students formed virtual groups, and for the oral presentation, they had to submit an audio recording. This worked well because they were forced to reflect about their work and organize their thoughts to prepare a script before recording. Each member of the group had to submit his/her own audio recording.

Secondly, as a distance course, it was observed that only a few students made use of the virtual office hours, and the main feedback they received was from the graded evaluations as opposed to the immediate feedback obtained from the flipped classroom model.

To conclude, the experience of developing and implementing a course in computation, modeling, and simulation has been very gratifying. We believe that such a course provides the means to facilitate, to a diverse student body, the access to the much-needed fundamental CS competencies. From the students' comments, we believe that they recognize the importance of having knowledge in modeling and simulation as well as the leading edge it provides in entering today's workforce. The students also valued the pedagogical model that allowed for active learning activities as well as immediate feedback from the instructor. We are hoping that for our next cohort we can impact pre-service teachers as well as provide professional development to K–12 teachers.

11. ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation funded Grant CyberTraining: CUI: Computational and Data Science Literacy Exchange (CSE-1829717). The author would like to acknowledge the support of Linda Akli from SURA and Katharine Cahill from OSC and the rest of the C^{2}Exchange team for providing a five-star space for our curriculum development discussions.

12. REFERENCES

[1] Netravathi Basavaraj Angadi, Avinash Kavi, Kimi Shetty, and Nayana Kamalnayan Hashilkar. 2019. Effectiveness of flipped classroom as a teaching-learning method among undergraduate medical students — An interventional study. *Journal of Education and Health Promotion*, 8, Article 211 (Oct. 2019). DOI: https://doi.org/10.4103/jehp.jehp_163_19

[2] R. Brewer and S. Movahedazarhouligh. 2018. Successful stories and conflicts: A literature review on the effectiveness of flipped learning in higher education. *Journal of Computer Assisted Learning* 34 (Feb. 2018), 409–416. DOI: https://doi.org/10.1111/jcal.12250

[3] Centro de Recursos para la Educación a Distancia. 2019. CREAD. Retrieved September 8, 2021 from https://www.uprm.edu/cread/

[4] Centro de Recursos para la Educación a Distancia. 2019. Mejor Prácticas. Retrieved June 11, 2021 from https://www.uprm.edu/cread/mejores10practicas/

[5] José Ferrer. 2017. *Diseño y creación de materiales educativos: Guía de la mínimo a lo óptimo para cursos en línea* (Spanish Edition).

[6] Flipped Learning Network (FLN). 2014. What Is Flipped Learning? Retrieved September 8, 2021 from https://flippedlearning.org/wp-content/uploads/2016/07/FLIP_handout_FNL_Web.pdf

[7] Steven I. Gordon. 2010. Creating Computational Science Programs for the Existing and Future Workforce. In *Proceedings ACM / IEEE SC2010 — International Conference for High Performance Computing, Networking, Storage and Analysis SC10*, Nov. 13–19, 2010, New Orleans, LA.

[8] Steven I. Gordon and Katharine Cahill. 2020. The State of Undergraduate Computational Science Programs. *Journal of Computational Science Education* (Apr. 2020), 7–11. DOI: https://doi.org/10.22369/issn.2153-4136/11/2/2

[9] Steven I. Gordon, Kate Carey, and Ignatios Vakalis. 2008. A Shared, Interinstitutional Undergraduate Minor Program in Computational Science., *Comput. Sci. Eng.*, 10, 5 (Aug. 2008), 12–16. DOI: https://doi.org/10.1109/MCSE.2008.127

[10] Steven I. Gordon, James Demmel, Lizanne Destefano, and Lorna Rivera. 2015. Implementing a Collaborative Online Course to Extend Access to HPC Skills, *Comput. Sci. Eng.* 18, 1 (Dec. 2015), 73–79. DOI: https://doi.org/10.1109/MCSE.2016.6

[11] Steven I. Gordon and Brian Guilfoos. 2017. *Introduction to Modeling and Simulation with MATLAB® and Python* (1st. ed.). Chapman and Hall/CRC. London, UK.

[12] Scott A. Lathrop, Katharine Cahill, Steven I. Gordon, Jennifer Houchins, Robert M. Panoff, and Aaron Weeden. 2020. Preparing a Computationally Literate Workforce. *Comput. Sci. Eng.* 22, 4 (May 2020), 7–16. DOI: https://doi.org/10.1109/MCSE.2020.2994763

[13] Michigan State University. What, Why, and How to Implement a Flipped Classroom Model. Retrieved September 8, 2021 from https://omerad.msu.edu/teaching/teaching-skills-strategies/27-teaching/162-what-why-and-how-to-implement-a-flipped-classroom-model

[14] Moodle Pty Ltd. 2021. Moodle: Online Learning with the World's Most Popular LMS. Retrieved September 8, 2021 from https://moodle.com/

[15] Python Software Foundation. 2021. Welcome to Python.org. Retrieved September 14, 2021 from https://www.python.org/

[16] Ashley Radder-Renter. 2020. The Flipped Classroom Model: What It Is and How It Works. (Sept. 2020). Retrived September 8, 2021 from https://www.yeseep.org/blog/the-flipped-classroom-model-what-it-is-and-how-it-works

[17] Shodor. 2011. HPC University: Minor Program in Computational Science Competency/Topic Overview. Retrieved June 7, 2021 from http://hpcuniversity.org/educators/undergradCompetencies/

[18] Shodor. 2021. Shodor: A National Resourse for Computational Science Education. Retrieved August 26, 2021 from http://www.shodor.org/

[19] Spyder Website Contributors. 2021. Home — Spyder IDE. Retrieved September 14, 2021 from https://www.spyder-ide.org/

APPENDIX A
List of Module Activities and Mini Projects

1. Watch a video and:
    a. Describe how the scientific method and simulation were mentioned in the video
    b. Describe the hypothesis mentioned in the video
    c. Describe if experiments are used to confirm or deny the hypothesis
2. Given an open-ended problem (best gas station option), design an app, and state:
    a. List of assumptions, concept map, mathematical model
3. Converting mathematical expressions to Python expressions
    a. Creating and editing Matrices.
    b. Using IDE Spyder.
    c. Using Numpy arrays.
4. Deterministic Linear Models: Modeling and Implementing a Traffic Model
5. Arrays Mathematics in Python using NumPy
6. Visualization with Python, storytelling with a dataset
7. Describe an algorithm for the problem of solving the real roots of a quadratic equation, create a flow chart as part of the solution, Calculating car rental price
8. Implementing different situations, where repetition structures are required, implement a guessing game.
9. Practice exercises on population growth, bank account interest rates, use of visualization to answer what-if questions.
10. Best fit from empirical data: implementing a model to predict the weight of a dog at any time during its life.
11. Using a dataset find the coefficients for a linear model using at least two of the different procedures in Python.

APPENDIX B
List of some of the topics of the final project

1. Finding the fastest route from home to UPRM, using traffic data and precipitation data
2. Effect of altitude on projectile moving at fast speeds
3. Ball Toss
4. Modeling predator prey population
5. SIR Model of COVID-19 in the USA
6. Prey-Predator Model with carrying capacity
7. Natural Disaster Model for Future Economic Losses

APPENDIX C
Topic 1
Activity 1
Watch the video in the provided link:
https://youtu.be/T9qoU9_tGhA

And answer the following questions:

1. Describe how the scientific method and simulation are mentioned in this video.
2. What is the hypothesis mentioned in the video?
3. Is it possible to make experiments to confirm or deny the hypothesis? Explain.

Activity 2
(based on an Example presented by Daniel Teague of North Carolina School of Science and Mathematics)

Suppose there is a local radio station that broadcasts the locations and prices for all the gas stations in your area. The question is, which should you buy from?

Make a PPT. Imagine you want to make an app for the phone, that can be used to answer that question. Turn in five PPT slides to illustrate your ideas.

Slide 1: List the assumptions.

Slide 2: Show the concept map.

Slide 3: Show the mathematical model.

Slides 4 & 5:

Create screens for an app based on your model. The first screen should request essential information from the user, and the second should show the app's response.

Attach your PPT document in the Moodle platform. In addition to the PPT, each student will also submit a script consisting of the explanation of your work. It is recommended that you also submit an audio version of the script.

Topic 2
Sample exercise:

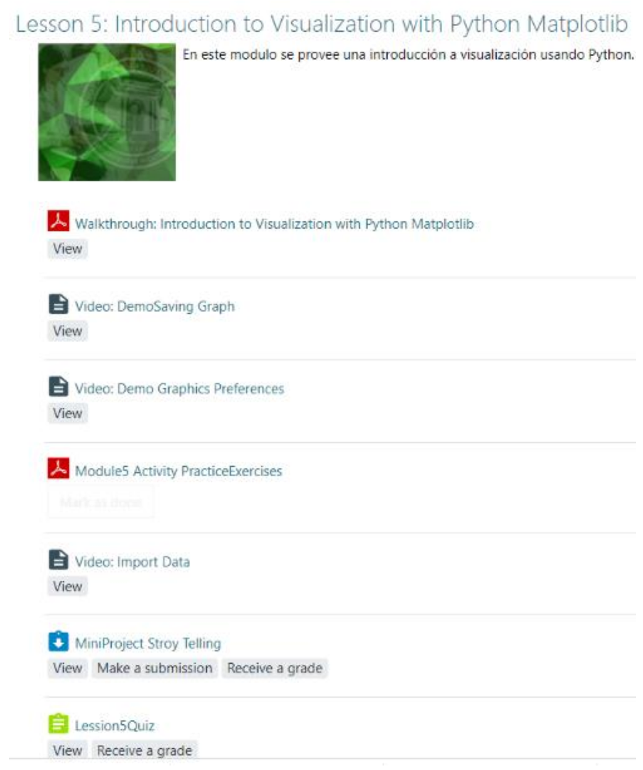
Using IDE Spyder, write code to:

1. Create a NumPy array with values 1, 7, 13, 105 and determine the size of the memory occupied by the array.
2. Create and print a NumPy array of integers from 30 to 70 in steps of 10.

Convert a list of numeric values into a one-dimensional NumPy array.

Topic 5

A figure of what is shown in Moodle



Lesson 5: Introduction to Visualization with Python Matplotlib

En este modulo se provee una introducción a visualización usando Python.

Walkthrough: Introduction to Visualization with Python Matplotlib
View

Video: DemoSaving Graph
View

Video: Demo Graphics Preferences
View

Module5 Activity PracticeExercises
Mark as done

Video: Import Data
View

MiniProject Stroy Telling
View Make a submission Receive a grade

Lession5Quiz
View Receive a grade

Topic 7

If else demo:

https://youtu.be/F8QwAhOiuq4

Topic 8

Sample of video (while):

https://youtu.be/a72XiszmKE8

Topic 10

Sample of videos:

https://youtu.be/bsopgKEdBF0

https://youtu.be/BzvYO13HDuk

https://youtu.be/6I9d7FVGEoo