OPEN ACCESS Freely available online

CourseSource

Lesson

Central Dogma, Dictionaries, and Functions: Using Programming Concepts to Simulate Biological Processes

Jyothi Kumar[1*], Fabio Gomez-Cano[2†], Seth W. Hunt[1†], Serena G. Lotreck[1,3,†], Davis T. Mathieu[4†], McKena L. Wilson[5†], and Tammy M. Long[1*]

[1]Department of Plant Biology, Michigan State University

[2]Department of Biochemistry and Molecular Biology, Michigan State University

[3]Department of Computational Mathematics, Science, and Engineering, Michigan State University

[4]Genetics & Genome Sciences Program, Michigan State University

[5]Department of Horticulture, Michigan State University

[†]Contributed equally

Abstract

Technologies like next-generation sequencing, proteomics, and high-throughput phenotyping have transformed the way we do biology. There is a continued need for scientists with computational skills to analyze biological data while understanding the underlying biological concepts. The Integrated training Model in Plant And ComputaTional Sciences (IMPACTS) is an interdisciplinary training program that trains doctoral students to employ computational and data science approaches to address grand challenges in plant biology. The first course in the curriculum, *Foundations in Computational Plant Science*, focuses on fundamental knowledge in computational and plant science through group learning and peer instruction while using real-world data. The lesson plan described here was developed by the 2019 cohort of IMPACTS trainees (authoring cohort) as part of a subsequent course on STEM teaching and learning. The authoring cohort collaborated to identify a gap in the *Foundations* curriculum and applied their learning about evidence-based instructional design to develop and subsequently teach the lesson in the next iteration of the course (2020). The lesson plan's goal was to develop students' abilities to apply dictionaries and functions as core tools in computational science to answer biological questions. The 2020 cohort that completed the lesson reported confidence in being able to effectively apply dictionaries and functions and provided feedback about modifications to improve lesson efficacy. This feedback was incorporated in the iterative version of this lesson. This lesson is designed to help bridge the gap between computer scientists and biologists by teaching them interdisciplinary concepts using real-world data.

Citation: Kumar J, Gomez-Cano F, Hunt SW, Lotreck SG, Mathieu DT, Wilson ML, Long TM. 2023. Central Dogma, Dictionaries, and Functions: Using Programming Concepts to Simulate Biological Processes. CourseSource 10. https://doi.org/10.24918/cs.2023.24

Editor: Abby Hare-Harris, Bloomsburg University

Received: 9/1/2022; **Accepted:** 4/12/2023; **Published:** 6/21/2023

Conflict of Interest and Funding Statement: This work was funded, in part, by NSF Research Traineeship Program, DGE- 8128149. None of the authors have a financial, personal, or professional conflict of interest related to this work.

***Correspondence to:** kumarjy1@msu.edu, longta@msu.edu

Learning Goals

Students will:

◊ apply in practice dictionaries and functions as core tools in computational science.

◊ explain affordances of computational methods in life sciences research.

◊ use computational approaches to answer molecular biological questions.

◊ From the Bioinformatics Learning Framework:

» What computational concepts are important in bioinformatics?

» Where are data about the genome found (*e.g.,* nucleotide sequence, epigenomics) and how are they stored and accessed?

» How do biologists employ software development as part of the scientific discovery process?

» What higher-level computational skills can be used in bioinformatics research?

Learning Objectives

Students will be able to:

◊ use variable and data structures (*e.g.,* lists, arrays, scalars, and functions).

◊ write a script to translate a DNA sequence to an amino acid sequence.

◊ utilize a dictionary to assess data.

◊ utilize a function for tasks like translation.

◊ explain the proper use of dictionaries and functions.

◊ explain when a dictionary is applicable compared to a list.

◊ describe the process of transcription and translation within biological systems and its role in the cellular flow of information from DNA to protein.

INTRODUCTION

The recent development of technologies such as next-generation sequencing and high-throughput phenotyping has changed the way biologists approach their scientific pursuits and introduced new challenges in data analysis (1–6). The amount of sequence data that is presently available is incomprehensibly large, with over 1.75 quadrillion base pairs in the NCBI database as of August 2022. The ubiquity of big data approaches such as genomics, phenomics, proteomics, lipidomics, etc. has provided biological disciplines with an overwhelming amount of publicly available data (7). Although the generation of this information is an incredible accomplishment, it has also driven the demand for both biological and computational experts that can analyze these data and make meaningful use of it (8, 9). This demand will require biologists and computational scientists to have a working knowledge of each other's disciplines to communicate and collaborate effectively (10). For biologists to use computational tools, they must possess the vocabulary and a fundamental understanding of computational systems in order to discuss biological data problems with computational experts when needed. Similarly, computational scientists should have a fundamental understanding of biology so they can comprehend the nature of the biological questions being asked, troubleshoot when problems arise, and be effectively critical of the computational output. Bridging these disciplines requires a change in curricula and training for graduate students of both fields and opportunities for students to integrate their disciplinary knowledge through authentic collaborations (11).

Here, we describe a lesson plan that builds on the effectiveness of Jupyter Notebook to teach coding in Python, specifically functions and dictionaries, while also introducing core concepts related to the central dogma of biology. The novelty in this lesson's design is that it is the product of students enrolled in a graduate-level seminar course focused on STEM teaching and learning. The author cohort is affiliated with the Integrated training Model in Plant and CompuTational Sciences (IMPACTS; NSF DGE 1828149, PI: Shiu), a graduate training program at Michigan State University funded through the National Science Foundation Research Traineeship (NRT) program. IMPACTS cross-trains graduate students in plant biology and computational sciences to collaborate in research that addresses grand challenge problems in plant biology that require computational approaches. The first course in the IMPACTS curriculum is *Foundations in Computational Plant Science* (*Foundations*), which aims to provide foundational knowledge in plant computational science and to foster interdisciplinary, long-term, and collaborative partnerships among trainees. Five of the authors (FG-C, SWH, SGL, DTM and MLW) were students in the first iteration of *Foundations* and subsequently enrolled in F*orum on STEM Teaching and Learning* (*Forum*), a seminar course that is also part of the IMPACTS curriculum but focused on professional development and the scholarship of teaching and learning. In *Forum*, students were assessed on their ability to apply the principles of backward design and evidence-based instruction to the design and implementation of a lesson of their choice. The lesson described here is the product of that collaborative effort.

The author cohort focused their lesson design on a perceived gap in the *Foundations* curriculum—that of dictionaries and functions in Python. In addition, the authors chose the central dogma as the biological concept to contextualize the lesson. An overall goal of the IMPACTS program—and the *Foundations* course specifically—is to provide training that intersects principles of computational and plant sciences. In biology, the central dogma explains how genetic code is ultimately converted to functional proteins. In Python, dictionaries allow storing information associated with a key (like a word) and its information retrieval as the value (like its definition) when needed. A function is a segment of code which when called performs a specific task and returns data as a result. Thus, the mechanisms by which dictionaries and functions work in Python are uniquely analogous to the biological mechanisms underlying the central dogma, making it an especially productive context for this interdisciplinary lesson.

Intended Audience

This lesson was originally designed for first- and second-year graduate students in an interdisciplinary graduate program that focuses on plant and computational science. However, the lesson can be used in its current form or adapted to suit diverse student audiences at both graduate and undergraduate levels in any discipline with the aim of developing students' computational skills and biological knowledge. This lesson will be most effective in a computational lab environment in which students have had some prior exposure to basic computational skills in Python (*e.g.,* lists and loops) and terminology (*e.g.,* data types and Boolean logic). However, specific background knowledge on the topics covered in this lesson (dictionaries and functions, central dogma) are not essential for students to participate and achieve the lesson's objectives for learning. Given that students' prior knowledge and strengths will vary, the lesson is designed to be flexible in order to accommodate diverse levels of expertise.

Required Learning Time

In preparation for the in-class portion of this module, students are asked to watch the YouTube video: "DNA, Hot Pockets, & The Longest Word Ever: Crash Course Biology #11." Additionally, students will familiarize themselves with functions and dictionaries by reading the introductory information in the Student Jupyter Notebook, from "Central Dogma" to "Putting it together." For students unfamiliar with the central dogma, extra resources from Khan Academy have been included to support basic understanding. Finally, students will watch a 22-minute tutorial highlighting the computational components of the module and the process of translation: "Plants & Python, vol. 7: Functions, Dictionaries, and L-Systems," created by Dan Chitwood. Both videos are embedded within the student and instructor Jupyter Notebook lesson plans and are also available on YouTube. We recommend students allocate 90 minutes for class preparation.

The lesson was designed for an 80-minute class period (refer to Table 1, Class Session: Dictionaries and Functions), but can be readily adapted for different class times. The three-part lesson (also described in the In-Class Activities section) begins with dictionaries in the Student Jupyter Notebook at "Introduction to Dictionaries and Functions: Putting it into

practice." Students begin the class by working individually through the first section (5–8 minutes), "Introduction to Dictionaries" to create dictionaries highlighting their own personal identifiers. Students then transition to the next section, "Introduction to Functions" and work individually at their own pace (20 minutes) to read through examples and answer two questions. The final section, "Synthesis Question: Dictionaries and Functions," involves a synthesis question answered during class and a challenge question that can be finished in class or at home. The synthesis question will take approximately 55 minutes overall. Time during this component of the lesson is divided into two phases: first, students work in groups to create a framework for their computational coding and discuss the logic behind their solutions. This approach is referred to as "paper coding" in the notebook (20 minutes). Second, individuals or groups work to answer the synthesis question (35 minutes). Since the course is self-paced, students may continue to work on their solutions at home and turn in the notebook the following class period.

Prerequisite Student Knowledge

An understanding of fundamental concepts in biology is preferred but not required. The lesson uses central dogma as a context for teaching dictionaries and functions. Videos and resources introducing central dogma are provided in the notebook in order to prepare students who do not have a biology background and/or for those who would prefer to review relevant concepts (DNA, Hot Pockets, & The Longest Word Ever: Crash Course Biology #11, Intro to gene expression, Central dogma of molecular biology). This lesson is the seventh class period in the *Foundations* course and occurs in the first quarter of the semester. Students will have covered fundamental components of programming in Python in the first six lessons, including a working knowledge of Jupyter Notebook, data types (*e.g.,* string, number integers and floats, lists), for loops, boolean logic, if statements, and list indexing. A computer will be necessary for this lesson; therefore, it should be taught in a computer lab or students can use their own laptop if they have one. Anaconda (Anaconda Software Distribution; computer software; vers. 2-2.4.0; Anaconda, Nov. 2016) must be installed in advance and step-by-step instructions for installing these programs can be found in "Getting Started with Jupyter".

Prerequisite Instructor Knowledge

The instructor needs to have an understanding of the biological concepts covered in the lesson, as well as a working knowledge of Python and Jupyter Notebook (12, 13). Instructors should be comfortable explaining the basics of the central dogma at both the molecular and cellular levels, and at a level conducive to students without a biology background. In terms of Python, the instructor should be able to read, interpret, and troubleshoot (debug) Python code in a Jupyter Notebook environment, such that they do not have to refer directly to the answer key if a student has taken a different algorithmic approach. Instructors should be comfortable with the basic programming concepts and syntax used in this lesson and are encouraged to complete the lesson on their own prior to coming to the classroom. This preparation will ensure the material is fresh in the instructor's mind and that they will be able to effectively coach students during the lesson. The instructor copy of the Jupyter Notebook can be used as a guide to assess possible solutions and to identify what the expected outputs should look like. Instructors should additionally be comfortable implementing common active learning methods (*e.g.,* think-pair-share), guiding whole class discussions, and facilitating students' self-directed learning (14–17).

SCIENTIFIC TEACHING THEMES

To teach scientifically means applying the same principles of evidence and reasoning regularly used in science to the context of teaching and learning. This lesson was conceived by students who participated in a common learning experience and perceived a gap in the curriculum that they predicted could impair students' course performance and learning outcomes. The lesson was created using a Backward Design (18) approach and by applying evidence-based principles about how people learn (19). Post-instructional data was collected to test assumptions about student understanding and was used to revise and iteratively improve lesson elements. In addition, this lesson was designed to teach programming concepts by intentionally identifying biological processes that they most closely simulate. Such integration was a direct outcome of engaging a diverse and multidisciplinary group of learners and reflects a more authentic approach to "tapping the interdisciplinary nature of science" (20) than merely using a biological context or dataset to teach a computational principle.

Active Learning

This lesson is designed for students to evaluate their learning through a self-paced Jupyter Notebook (13) that challenges them to support their computational solutions with algorithmic thinking and logic.

A flipped classroom approach (21) was chosen by instructors because it provides opportunity for self-paced learning and multiple modes of engaging with the material. Specifically, preparatory videos, practice exercises, and written explanations describing biological and computational concepts were provided to students before class to introduce the main ideas that would be discussed and applied during the class. Students could work individually or with classmates at their own pace to prepare for class.

Jupyter Notebooks are self-paced, accessible (13) and especially conducive for actively engaging students with Python as they learn. Students can annotate code, practice running scripts, and add or delete segments of code as they move through the lesson, thereby enabling practice and visualizing the consequences of modifying functioning elements.

Collaborative teams are a critical element of the lesson design. Ideally, teams would be composed of students that differ in their disciplinary expertise and prior experience in coding. Peer learners support one another as they troubleshoot errors and work through problems. Diverse teams bring differing perspectives about problem approaches and can promote students' development of more robust debugging skills critical in programming. Seating students together in groups or, in the virtual setting, separating students into breakout rooms, is preferred to encourage collaboration (22, 23).

"Paper coding" is an instructional approach in which a general algorithm is written down in preparation for writing true code. Paper coding acts as an opportunity for students to outline their Python scripts and brainstorm their methodology for solving a computational problem. This activity challenges students to apply concepts and transfer information learned in preparation for class. Analogous to hypothesis generation in science, paper coding requires students to propose an explanation or approach that can be tested empirically through enacted code. Paper coding is generally done as a think-pair-share, where students first conceive solutions on their own, then work with a partner (ideally, of a different disciplinary background) to discuss and refine their strategy.

Oral presentations by collaborative teams enable whole-class discussion about alternative approaches taken for solving problems. Feedback from instructors and classmates can help students identify affordances of different strategies and expand their understanding of both computational and biological concepts.

Inclusive Assessment

For a Jupyter Notebook-based tutorial with an active learning focus such as this, the completion of the task is the direct assessment (22, 24). The paper coding, think-pair-share activity assesses students' understanding of both the central dogma and the computational techniques introduced in this lesson. This assessment requires students to verbalize their understanding and synthesize a step-by-step design before beginning to code. The synthesis question and challenge question, available for extra practice, are designed to motivate students to apply their understanding of the lesson to complete the tasks efficiently and effectively. To assess the synthesis question, think-pair-share groups present their code to class, providing explanation for each cell in their notebook. In terms of learning outcomes, students should be able to develop a function that will take a gene and return a sequence string of the longest possible isoform, the length of the longest isoform, and the number of possible isoforms based on the number of exons. Students should also be able to respond to questions posed by the class as well as the instructors, which fosters student engagement.

The student's Jupyter Notebook also serves as an assessment of their prior knowledge and understanding of the concepts explained in the pre-class assignment. Instructors provide feedback on students' completed notebooks to clarify any missteps and suggest alternative strategies where appropriate. This feedback informs students of their personal understanding of the lesson and informs instructors about students' comprehension such that common challenges can be addressed and the pace or instruction can be altered in the following lesson.

Inclusive Teaching

Students in this course have diverse backgrounds in biological and computational sciences. As such, this lesson has been designed for any student, regardless of academic background, outside the first lessons in the course. The collaborative design of this lesson and use of think-pair-share celebrates a diversity of student knowledge and encourages cooperation between peers to overcome challenges within the material (16, 24). Ideally, students work in small peer teams with diverse disciplinary expertise, such that they can collaborate to understand core concepts related to central dogma, functions, and dictionaries. However, the lesson is also conducive for individual learners, thereby promoting inclusion of students who participate remotely and/or asynchronously. The use of Jupyter Notebook allows a personalized learning experience for each student. Students can choose to process material before class, by coding alone or rewatching lecture videos, share code during class, add comments to their own notebook, and come to class ready to work through problems with others. The interdisciplinarity of the lesson encourages students to actively engage, include, and challenge ideas during group interaction. Since the students are from diverse disciplines the heterogeneity of the group creates an expansive learning environment. Students' start at different levels of initial knowledge, but with committed effort in working through the Jupyter Notebook, they are able to master the basics of dictionaries, functions, and central dogma by the end of the course.

In addition, the lesson plan was originally designed for in-person delivery in the fall semester of 2020. However, due to COVID-19, the course was taught online, and proved to be adaptable in a virtual classroom. We believe this lesson plan is conducive to in-person, virtual, and asynchronous teaching as well as alternative self-paced individual or team learning platforms.

There are elements of inclusive teaching inherent in the design of the course beyond the low background knowledge required and the in-class group work.

Paper coding serves as a way to help the peer groups learn together and creates opportunities for the novice learners to get their questions answered. The Plants and Python lesson available via GitHub creates access to the learning material to those unaffiliated with Michigan State University (MSU). Indeed, the Plants and Python courses are offered in a Universidad Nacional Autónoma de México (UNAM) course with Jupyter Notebooks and videos in Spanish language. Bridging the learning gap and making these resources widely accessible is a positive step towards inclusive teaching.

LESSON PLAN

The lesson is structured into three main introductory sections in the Jupyter Notebook. The three sections introduce the central dogma, dictionaries, and functions with examples of each. These three sections may or may not be necessary depending on a student's background in each respective topic. The fourth section has students follow along with a video lecture and checks understanding as students fill in their own custom dictionary with values about themselves, practice passing values to a prewritten function, and write their own function to perform simple statistical processes. The last part within the fourth section provides two challenge problems. Both problems have complex biological themes and require students to apply dictionaries and functions with a greater depth of understanding than the rest of the lesson. The first challenge problem provides students with a dictionary containing real genes in the *Physcomitrium patens* (moss) genome, and all elements within that dictionary refer to exonic and intronic space within those genes (25). Groups are expected to write a function that passes these dictionaries and returns a sequence

string for the longest possible version of each gene, the number of base pairs within that gene isoform, and the number of possible variations in which that gene could be spliced. The second challenge problem asks groups to write a function that identifies the start codons within a nucleotide sequence generated through putative gene models and translates each respective codon to their subsequent amino acid to produce a full length polypeptide.

This activity is designed for an 80-minute class period in which basic concepts of coding and terminology in Python have previously been covered. The class progression is outlined in the teaching timeline sections of Table 1 with central objectives (Preparation for Class) and estimated timing (Class Session: Dictionaries and Functions).

Classroom Context

This lesson is best taught in a room with grouped seating and large or shared monitors so that students can work together, ask each other questions, and easily share their code with one another throughout the class period.

Pre-Class Instructor Preparation

Before class, the instructors should familiarize themselves with the concepts of the lesson, prepare strategies to aid students through known or anticipated challenges, and have a plan for how groups will be determined (*e.g.,* self-formed or assigned based on expertise).

Pre-Class Student Preparation

Students are given the full notebook and asked to read the introductory information prior to class. In the first section, the central dogma is explained within a CrashCourse video, and additional resources are provided for students who would like more information. In the second section, dictionaries are then explained through a three step framework: what is a dictionary, why use a dictionary, and how to use a dictionary. The same framework is utilized for functions in the third section. To bring these concepts together in section 4, the students are asked to watch Dr. Dan Chitwood's video, "Plants & Python, vol. 7: Functions, Dictionaries, and L-Systems." We recommend that if a student has no prior knowledge of functions and dictionaries they read the introductory information and watch Dr. Dan Chitwood's video twice: first taking notes on computational concepts, and then working through the tutorial on their own Jupyter Notebook. If a student has no prior knowledge of the central dogma, we recommend the student watch both videos provided and utilize the resources listed at the beginning of the Notebook.

In-Class Activities

1. **Introduction to Dictionaries:** Students utilize dictionaries to create their own personal identifiers. This section allows students to apply the knowledge learned from the introductory information in the second section, "An introduction to dictionaries in Python," of the notebook.
2. **Introduction to Functions:** Beginning with two examples, students are able to analyze the provided functions and then write their own to calculate the area of a circle. Taking it one step further, students will then write their own function to perform a set of statistical operations. Again, this section is asking students to utilize their pre-class preparation and build upon their knowledge of how functions can be utilized by actively implementing their own (refer to "An introduction to functions" in the second section of the notebook).
3. **Synthesis Question: Dictionaries and Functions:** First, students will form groups and then utilize think-pair-share to paper code their solutions. By paper coding the steps and solutions for the synthesis question first, students focus on the process of computational coding and discover the logic behind each step. After groups discuss their frameworks with one another, groups will move on, and work to define and apply functions that identify the start codon of a DNA sequence and translate it to a polypeptide chain. While it was not included in the original implementation of the lesson, a challenge question is now included at the end of the notebook for students who finish the assignment quickly or those who would like to further develop their skills regarding functions after class (refer to "Introduction to Dictionaries and Functions: Putting it into practice" in the notebook). Students must develop a function that identifies splice variants and construct possible isoforms that could be generated from an mRNA transcript. Further reflection for the inclusion of this challenge question can be found in the Discussion.

TEACHING DISCUSSION

Effectiveness at Achieving the Stated Learning Goals and Objectives

This lesson was inspired by the authors' collective experiences in the *Foundations* computational course, which merged students from computational and biological disciplines. The course content for *Foundations* is available here. As with other lessons in the course and many other preliminary Python courses, our lesson uses Jupyter Notebook and a flipped class approach (21, 26–29). Jupyter notebook is a 'computational notebook' that has broad support in the data science community in both professional and education contexts. Jupyter notebook is also effective in multimedia integration (*i.e.,* code, text, images, and other scaffolds) that facilitate data exploration, allows students to clearly make note and annotate their scripts, and has a growing acceptance worldwide (21, 26–29). In our design, prior to class, students watch video lectures to introduce concepts, practice simple applications using working models and prewritten scripts, and conclude with basic problem sets where newly introduced concepts are used in applicable scenarios. This modality prior to class was conducive for an interdisciplinary course with learners that varied in their prior experience; novice learners can aptly pace themselves and revisit information while experts can skip already familiar information. In class, students receive peer and instructor support as they collaborate to solve problems that test their ability to apply multidimensional concepts. An unforeseen advantage of our design was the seamless transition to online instruction during the pandemic. Although the course was designed for in-person instruction, shifting to online required few changes to our original design.

An explicit aim of our lesson was to leverage the expertise of both the biological and the computational science students in order to teach foundational concepts of both disciplines. Our lesson provides training on functions and dictionaries together

in the context of the central dogma. Dictionaries in Python organize data and variables that help the logical flow of code and reduce computational demand of the system. They accomplish these two tasks because (i) data can be paired, where each entry has a **key** and a corresponding **value**, which other data structures (*e.g.,* lists) cannot, and (ii) dictionaries use an under-the-hood approach to search specific terms instead of loading every object in a list sequentially. While many new coding students are content with lists to organize their data, the limitation of exclusively doing so is quickly discovered as datasets get larger and more complicated. In biological contexts, where datasets are often large and include paired data (*e.g.,* gene name: gene sequence), dictionaries can optimize computing power and provide a useful tool for organizing their code more effectively. Like dictionaries, functions are a fundamental way to organize and reuse code, which drastically improves the readability of scripts. When students first learn to code, they frequently engage in procedural programming, in which code is written in a stream-of-consciousness style, with segments of code copy and pasted multiple times to perform similar tasks. However, this methodology often becomes untenable when working with real-world data. Functional programming involves compartmentalizing code that performs the same task into functions, which allows for a segment of code that performs a specific computational task (*e.g.,* a statistical analysis) to be appropriately named after its action and referenced repeatedly in a script. Using functions improves efficiency by (i) allowing algorithms to be reused, (ii) reducing complexity and redundancy in executable scripts, and (iii) improving fidelity where properly functioning algorithms are not at risk of copy-and-paste errors. The central dogma refers to the collective processes by which biological information encoded in DNA (*i.e.,* a gene) is transcribed to the intermediate molecule (mRNA) and then translated/expressed as a functional protein. The central dogma, sometimes referred to as 'information flow,' is broadly regarded as a core concept for basic biological literacy (23, 24).

Although functions and dictionaries could each be taught independently and acontextually, to do so would miss an opportunity to provide students an authentic interdisciplinary learning experience. Our synthesis and challenge questions ask students to use these computational tools to translate and splice mRNA, similar to the basic principles observed within the cell. We believe that this unique combination of tools and the context for solving such a task are particularly conducive for fostering learning about core principles across disciplines. While dictionaries and functions are likely already familiar concepts for computational students, using them together for this application simulates the machinery and nuances of gene expression, and thereby promotes a more mechanistic understanding of how and why biological information is expressed in nature. For biologists, the applications of dictionaries and functions are immediately relevant due to their combined ability to process paired data with high levels of repetition and fidelity. However, biologists can gain a deeper understanding of the mechanics underlying the computational tools because they can mirror and apply these tools to familiar biological processes.

As technology rapidly continues to advance and computational power grows so will possible applications for analyzing biological data. With this unavoidable future, we identified the essentiality of bridging communication and expertise between computational scientists and biologists. Maximizing the benefits from this collaboration requires fundamental understanding of and limitations of the complementary discipline. We believe our lesson helps bridge communication and expertise between computational scientists and biologists by teaching and applying foundational, interdisciplinary concepts with real world data.

The unique nature of previous students developing this lesson intends to set the precedent that curriculum design should continue to build upon itself where students and educators incorporate feedback and play an active role in designing the next iteration when possible.

Students and educators alike can identify the necessity for effective collaboration between computational scientists and biologists for the future of answering biological questions (9, 10, 30). The necessity to successfully instill an effective education for students who pursue these goals is what motivated the cohort of student authors to actively design this lesson plan. The steep learning curve for computational skills is intimidating and discouraging to new students so ensuring that peers, educators, and curriculum all work together for providing the best education is critical to getting more students involved and effective at using computational tools in biology.

Student Reaction

Student feedback was gathered through Google Forms surveys and informed subsequent revisions to the lesson. A survey (Supporting File S1) was administered after the course asking for feedback on the activity from the sixteen enrolled students. Overall, students reported positively when surveyed about their experiences with the lesson during the Fall 2020 semester. Of the eight students who responded, a majority of six students self-reported that they were able to grasp the biological concepts prior to the start of the lesson but the responses varied in the self-reported understanding of the computational concepts prior to the lesson. One student stated that "*The lesson was very easy to follow and I think even beginners can understand*." Another claimed that they were able to grasp the biological and computational concepts "*very well*" and further said, "*I think using the example of the translation process to introduce dictionaries and writing functions was a well taken opportunity*." Since the lesson provides students with a challenging task that is scaffolded so that they may become familiar with key concepts while grappling with the coding complexities, some students struggled with the level of difficulty of this task. This was evident when one student said, "*I grasped the biological concepts well because I have experience in them, but I got lost during the coding session. The majority of the notebook was easy to understand, but I wish I had worked on the challenge at home by myself first.*" Four of the students specifically included in their open-ended response that the challenge question was helpful in understanding the concept of a dictionary. While the majority of students reported having a positive experience with the notebook, which helped to solidify computational and biological concepts, they also provided constructive feedback to drive future iterations of the course.

Suggestions for Improvement

In the original iteration of this lesson, only one challenge question was provided. Nearly all groups were able to finish

this challenge problem and present their code to the rest of the class. Students were also able to respond appropriately to questions posed by both the teaching team and other students during the presentations of their scripts. However, based on the survey responses, adding degrees of difficulty to challenge problems could help the students to grasp computational concepts. This would also create opportunities for more student collaboration and reinforce transfer of knowledge. Increased complexity (*e.g.,* isolating exons, removing introns, determining splice variants, etc.) with the challenge problem will also further engage those with more coding skill while also relying on the biological knowledge that students with opposing expertise might possess. Suggestions from the survey also included providing links to Stack Overflow and similar resources in the lesson plan to aid in the self-learning activities like troubleshooting the Python code.

An important factor would be to do an inventory of the students' disciplinary backgrounds and expertise levels in both computational and biological concepts at the beginning of the class in a survey. This would help to form collaborative groups with heterogeneous expertise resulting in greater sharing of knowledge between diverse learners (31). This is especially important because of the range of expertise of the students and disciplinary backgrounds.

This lesson was tested within a fully-online, synchronous graduate-level course. The authors observed that the use of Jupyter Notebook in conjunction with a video conferencing platform that has breakout sessions and screen-sharing capabilities worked well. While this lesson plan was originally designed to be taught in-person, the technology used within this lesson allowed for an easy transition to an online format.

SUPPORTING MATERIALS

- S1. Central Dogma, Dictionaries, and Functions – Survey Questions

ACKNOWLEDGMENTS

This work is supported in part by Michigan State University and National Science Foundation Research Traineeship (NRT) Program (DGE-1828149; PI: Shiu). The NRT-IMPACTS (Integrated Training Model in Plant and Computational Science) program at Michigan State University provided a context for this work and financial support to authors: JK, DTM, SGL, MLW, SWH, and FG-C. We thank Blake Trygestad for his early investment and contributions to the design of this lesson plan. We thank Daniel Chitwood and Robert VanBuren for their input during the writing of this manuscript and the initial implementation of this lesson during their course. We thank all the students for their participation in working through this lesson plan during the Fall 2020 *HRT-841: Foundations in Computational Plant Science* course.

Authors, FG-C, SWH, SGL, DTM, and MLW, contributed equally to the writing and editing of this publication.

All student surveys were conducted with prior approval from the MSU Institutional Review Board for Human Subjects (IRB #STUDY00000978).

REFERENCES

1. Leonelli S. 2019. Philosophy of biology: The challenges of big data biology. Elife 8:e47381. doi:10.7554/eLife.47381.
2. Fillinger S, de la Garza L, Peltzer A, Kohlbacher O, Nahnsen S. 2019. Challenges of big data integration in the life sciences. Anal Bioanal Chem 411:6791–6800. doi:10.1007/s00216-019-02074-9.
3. Bayat A. 2002. Science, medicine, and the future: Bioinformatics. BMJ. 324:1018–1022. doi:10.1136/bmj.324.7344.1018.
4. Kahvejian A, Quackenbush J, Thompson JF. 2008. What would you do if you could sequence everything? Nat Biotechnol 26:1125–1133. doi:10.1038/nbt1494.
5. Ekblom R, Galindo J. 2011. Applications of next generation sequencing in molecular ecology of non-mode organisms. Heredity 107:1–15. doi:10.1038/hdy.2010.152.
6. Gupta AK, Gupta UD. 2014. Next generation sequencing and its applications, p 345–367. *In* Verma AS, Singh A (ed), Anima biotechnology. Academic Press, Waltham, MA.
7. Boekel J, Chilton JM, Cooke IR, Horvatovich PL, Jagtap PD, Käll L, Lehtiö J, Lukasse P, Moerland PD, Griffin TJ. 2015. Multi-omic data analysis using Galaxy. Nat Biotechnol 33:137–139. doi:10.1038/nbt.3134.
8. Maloney M, Parker J, LeBlanc M, Woodward CT, Glackin M, Hanrahan M. CBE Life Sci Educ 9:172–174. doi:10.1187/cbe.10-03-0038.
9. Robeva RS, Jungck JR, Gross LJ. 2020. Changing the name of quantitative biology education: Data science as a driver. Bull Math Biol 82:127. doi:10.1007/s11538-020-00793-0.
10. Cechova M. 2020. Ten simple rules for biologists initiating a collaboration with computer scientists. PLoS Comput Biol 16:e1008281. doi:10.1371/journal.pcbi.1008281.
11. Friesner J, Assmann SM, Bastow R, Bailey-Serres J, Beynon J, Brendel V, Buell CR, Bucksch A, Busch W, Demura T, Dinneny JR, Doherty CJ, Eveland AL, Falter-Braun P, Gehan MA, Gonzales M, Grotewold E, Gutierrez R, Kramer U, Krouk G, Ma S, Markelz RJC, Megraw M, Meyers BC, Murray JAH, Provart NJ, Rhee S, Smith R, Spalding EP, Taylor C, Teal TK, Torii KU, Town C, Vaughn M, Vierstra R, Ware D, Wilkins O, Williams C, Brady SM. 2017. The next generation of training for Arabidopsis researchers: bioinformatics and quantitative biology. Plant Physiol 175:1499–1509. doi:10.1104/pp.17.01490.
12. Kluyver T, Ragan-Kelley B, Pérez F, Granger B, Bussonnier M, Frederic J, Kelley K, Hamrick J, Grout J, Corlay S, Ivanov P, Avila D, Abdalla S, Willing C, Jupyter development team. 2016. Jupyter Notebooks – A publishing format for reproducible computational workflows, p 87–90. *In* Loizides F, Scmidt B (ed), Positioning and power in academic publishing: Players, agents and agendas. IOS Press, Fairfax, VA.
13. Reades J. 2020. Teaching on Jupyter. REGION. 7:21–34. doi:10.18335/region.v7i1.282.
14. Ebert-May D, Brewer C, Allred S. 1997. Innovation in large lectures: Teaching for active learning. Bioscience 47:601–607. doi:10.2307/1313166
15. Freeman S, Eddy SL, McDonough M, Smith MK, Okoroafor N, Jordt H, Wenderoth MP. 2014. Active learning increases student performance in science, engineering, and mathematics. PNAS 111:8410–8415.
16. Lyman F. 1981. The responsive classroom discussion: The inclusion of all students, p 109–113. *In* Anderson A (ed), Mainstreaming digest. University of Maryland, College Park, MD.
17. Wandersee JH, Mintzes JJ, Novak JD. 1994. Research on alternative conceptions in science, p 177–201. *In* Gabel D (ed), Handbook of research on science teaching and learning. MacMillan, New York, NY.
18. Wiggins G, McTighe J. 2005. Understanding by design, 2nd ed. Association for Supervision and Curriculum Development, Alexandria, VA.
19. Bransford JD, Brown AL, Cocking RR. 2000. How people learn. National Academy Press, Washington, DC.
20. Rebmann A, Beuther A, Fettkey P. 2020. Hands-on process discovery with Python - Utilizing Jupyter Notebook for the digital assistance in higher education, p 65–76. Joint Proceedings of Modellierung 2020 Short, Workshop and Tools & Demo Papers, Vienna, Austria.
21. Awidi IT, Paynter M. 2019. The impact of a flipped classroom approach on student learning experience. Comput Educ 128:269–283. doi:10.1016/j.compedu.2018.09.013.
22. Black P, Wiliam D. 1998. Assessment and classroom learning. Assess Educ 5:7–74.
23. Barkley E, Cross K, Major C. 2014. Collaborative learning techniques: A handbook for college faculty. John Wiley & Sons, San Francisco, CA.
24. Momsen J, Offerdahl E, Kryjevskaia M, Montplaisir L, Anderson E, Grosz N. 2013. Using assessments to investigate and compare the nature of

learning in undergraduate science courses. CBE Life Sci Educat 12:239–249. doi:10.1187/cbe.12-08-0130.
25. Lang D, Ullrich KK, Murat F, Fuchs J, Jenkins J, Haas FB, Piednoel M, Gundlach H, Van Bel M, Meyberg R, Vives C, Morata J, Syemeonidi A, Hiss M, Muchero W, Kamisugi Y, Saleh O, Blanc G, Decker EL, van Gessel N, Grimwood J, Hayes RD, Graham SW, Gunter LE, McDaniel SF, Hoernstein SNW, Larsson A, Li F-W, Perroud P-F, Phillips J, Ranjan P, Rokshar DS, Rothfeis CJ, Schneider L, Shu S, Stevenson DW, Thümmler F, Tillich M, Villarreal Aguilar JC, Widiez T, Wong GK-S, Wymore A, Zhang Y, Zimmer AD, Quatrano RS, Mayer KFX, Goodstein D, Casacuberta JM, Vandepoele K, Reski R, Cuming AC, Tuskan GA, Maumus F, Salse J, Schmutz J, Rensing SA. 2017. The *Physcomitrella patens* chromosome-scale assembly reveals moss genome structure and evolution. Plant J 93:515–533. doi:10.1111/tpj.13801.
26. Cardoso A, Leitão J, Teixeira C. 2019. Using the Jupyter Notebook as a tool to support the teaching and learning processes in engineering courses, p 227–236. *In* Auer M, Tsiatsos T (ed), The challenges of the digital transformation in education. Proceedings of the 21st International Conference on Interactive Collaborative Learning (ICL2018) - Volume 2. Springer, Cham, Switzerland.
27. Perkel JM. 2018. Why Jupyter is data scientists' computational notebook of choice. Nature 563:145–146. doi:10.1038/d41586-018-07196-1.
28. Nwulu N, Damisa U, Gbadamosi S. 2021. Students' perception about the use of Jupyter notebook in power system education. Int J Eng Pedagog 11:78–86. doi:10.3991/ijep.v11i1.14769.
29. Carrano D, Chugunov I, Ayazifar B. 2020. Self-contained Jupyter notebook labs promote scalable signal processing education, p 1409–1416. Sixth International Conference on Higher Education Advances. Editorial Universitat Politècnica de València, València, Spain. doi:10.4995/HEAd20.2020.11308.
30. National Research Council. 2009. A new biology for the 21st century. The National Academies Press, Washington, DC.
31. American Association for the Advancement of Science (AAAS). 2011. Vision and change in undergraduate biology education: A call to action. AAAS, Washington, DC.
32. Camara GS, Camboim S, Bravo JVM. 2021. Using Jupyter Notebooks for viewing and analysing geospatial data: Two examples for emotional maps and education data. Int Arch Photogramm Remote Sens Spatial Inf Sci XLVI-4/W2:17–24. doi:10.5194/isprs-archives-XLVI-4-W2-2021-17-2021.

Table 1. Teaching timeline.

Activity	Description	Estimated Time	Notes
Preparation for Class			
1. Watch "DNA, Hot Pockets, & The Longest Word Ever: Crash Course Biology #11"	Introduction to the central dogma (Section 1 in Jupyter Notebook).	Self-paced. Actual time to complete activities 1–3 depends on students' prior training, but ranges from 30 minutes to 2 hours	Prepare students with relevant background information about core concepts related to the central dogma and gene-to-protein phenomena. Additional resources are included in the notebook for students without a biological background.
2. Read Sections 2 and 3 in the Jupyter Notebook, Dictionaries and Functions	Overview of functions and dictionaries.	Self-paced. Actual time to complete activities 1–3 depends on students' prior training, but ranges from 30 minutes to 2 hours	Provide basic information about dictionaries and functions, including vocabulary, definitions, and examples.
3. Watch the first 22 minutes of "Plants & Python, vol. 7: Functions, Dictionaries, and L-Systems"	Apply dictionaries and functions to the biological process of translation (mRNA to protein) as an example.	Self-paced. Actual time to complete activities 1–3 depends on students' prior training, but ranges from 30 minutes to 2 hours	Reinforce and extend learning about functions and dictionaries by connecting to the biological example of translation. Enable students to (a) contrast the roles and purposes of dictionaries and functions, and (b) identify applications of each using translation as an example.
Class Session: Dictionaries and Functions			
1. Introduction to Dictionaries (refer to In-Class Activities in the article)	Construct and utilize their own dictionary.	10 min	Intended to help students understand dictionary syntax and when to use it, and to elicit and correct common misconceptions.
2. Introduction to Functions (refer to In-Class Activities in the article)	Develop and apply their own functions to answer questions.	20 min	Activities 1–2 are intended to help students understand syntax and when to use it. Additionally, these exercises are designed to elicit and correct common misconceptions.
3. Synthesis Question: Dictionaries and Functions – Paper Code (refer to In-Class Activities in the article)	Write down the steps to successfully answer the challenge question. This paper code will serve as a framework for the computational solution.	20 min	The goal of section 3 is to push students to think about the process of coding, beyond syntax, and assess the quality of students' algorithmic thinking as well as central dogma.
4. Synthesis Question: Dictionaries and Functions – Synthesis and Challenge question (refer to In-Class Activities in the article)	Utilizing their paper code developed during the previous step, students will incorporate both dictionaries and functions in order to isolate isoforms and find the longest CDS in the DNA given.	35+ min	In section 4 students will utilize algorithmic thinking, dictionaries, functions, etc. to answer the synthesis and challenge questions.
Assessment			
Download and complete notebook	After completion, the notebook will be downloaded and submitted to the course management system for the instructor's review. Feedback on the challenge question will be given.	Self-paced homework	As an assessment, students' completed notebooks will provide evidence of their grasp of concepts that were a focus of the lesson, their ability to apply algorithmic thinking, and their progress in developing their coding skills generally.