Tailored Computing Instruction for Economics Majors

Richard Lawrence*
HPRC†
Texas A&M University
College Station, TX
rarensu@tamu.edu

Zhenhua He*
HPRC†
Texas A&M University
College Station, TX
happidence1@tamu.edu

Wesley Brashear
HPRC†
Texas A&M University
College Station, TX
wbrashear@tamu.edu

Ridham Patoliya
HPRC†
Texas A&M University
College Station, TX
ridhampatoliya@hprc.tamu.edu

Honggao Liu
HPRC†
Texas A&M University
College Station, TX
honggao@tamu.edu

Dhruva K. Chakravorty
HPRC†
Texas A&M University
College Station, TX
chakravorty@tamu.edu

ABSTRACT

Responding to the growing need for discipline-specific computing curricula in academic programs, we offer a template to help bridge the gap between informal and formal curricular support. Here, we report on a twenty-contact-hour computing course developed for economics majors at Texas A&M University. The course is built around thematic laboratories that each include learning objectives, learning outcomes, assignments, and assessments and is geared toward students with a high-school level knowledge of mathematics and statistics. Offered in an informal format, the course leverages the wide applicability of the Python programming language and scaffolding offered by discipline-specific, hands-on activities to introduce a curriculum that covers introductory topics in programming while prioritizing approaches that are more relevant to the discipline. The design leverages technology to offer classes in an interactive, Web-based format for both in-person and remote learners, ensuring easy access and scalability to other institutions as needed. To ensure easier adoption among faculty and offer differentiated learning opportunities for students, lectures are modularized to 10-minute segments that are mapped to other concepts covered during the entire course. Class notes, lectures, and exercises are pre-staged and leverage aspects of flipped classroom methods. The course concludes with a group project and follow-on engagements with instructors. In future iterations, curriculum can be extended with a capstone in a Web-based asynchronous certification process.

Keywords

Python, Cybertraining, Google Colab, Economics, Cyber infrastructure (CI)

1. INTRODUCTION

The success of broadening participation in computing efforts relies on effectively developing a continuum of informal and formal learning environments for computing instruction. Simultaneously, we have to offer programming education that should be tailored to the needs of the students based on their discipline and level of commitment, rather than apply a one-size-fits-all approach. Indeed, recruitment and engagement can be improved by offering computing training in which examples and exercises are tailored to the academic and professional interests of the student body. Scaffolding can be achieved by adopting a tailored approach that focuses on a subset of topics that lead to a discipline-relevant final project that offers a feeling of accomplishment. The programming language, learning platform, and technologies should be coupled with continuing activities to encourage the interested learner to continue the process well beyond the duration of the activity. Simultaneously, we have to teach in incremental modules that let students solve small problems that are tied to real-world examples. To benefit the larger community, we should incorporate Web-based interactive computing avenues that ensure scalability and reproducibility at the core. Finally, the approach should be grounded in best practices in cyberinfrastructure technologies and reviewed pedagogical approaches. Such an approach should ideally not be limited by the computing technologies that are available to the students, but rather focus on making the current generation of technologies accessible to the students.

Texas A&M High Performance Research Computing has a legacy of offering informal courses geared toward adoption of CI practices in the regional researcher community [1, 4, 6, 7]. These have extended from our "short courses" that cover several cyberinfrastructure topics using hands-on exercises. In previous works, we have studied means to promote programming at the early undergraduate level [2], relied on visualization to study cybersecurity, and have explored opportunities to expand computing to the middle and high school levels [3]. With a view toward improving student learning in remote learning environments, we have explored pedagogical approaches such as our peer-moderated and peer-taught "Primers" [5]. We have built software platforms to facilitate the use of CI technologies to improve reproducibility in the sciences and coupled them with technology enabling "Tech labs" to improve CI adoption in research [7]. These efforts have been tied into asynchronous approaches that leverage interactive computing and social media to further CI technology adoption at Texas A&M University.

* Both authors contributed equally

† High Performance Research Computing

DOI: https://doi.org/10.22369/issn.2153-4136/13/1/6

As computing becomes more prevalent, programming is increasingly viewed as a critical career skill. Academic programs have moved to offer their graduates opportunities to develop skills in languages such as Python, R, MATLAB, or C++. Academic programs that do not have courses that directly tie into computing have relied on informal or certificate-based programs to help their students gain these skills. Here, we use our experience in offering CI-based training to develop a structured approach to teach "Python Programming" to graduate students in the Economics program at Texas A&M University. A template for adoption at institutions at other sizes is included here.

2. METHODS

The course was designed to be taught live in a mixed in-person and virtual hybrid modality and was structured to use active-learning methods. Curricular materials could be accessed via a Google Classroom (via Google Drive) for easy sharing. A course template was developed with unique branding and thematic elements to ensure that the course has a unique identity and to develop a sense of familiarity among students as we revisited old concepts. To ensure continuity during the classes, slide decks for the course were structured with learning objectives, concepts visited theory, hands-on exercises, and take-home assignments. To facilitate hands-on exercises while ensuring portability and equitable access to all students, Python examples and exercises were delivered in the form of Jupyter Notebooks as shown in Table 1, hosted in a Google Drive (students got a copy), and edited and run in Google Colab. The Notebooks contained a mix of informative lecture elements, interactive examples, and exercises. With a view toward student engagement, exercises used current real-world examples and relied on visualization approaches. Scaffolding was achieved by including a detailed description of topics covered in each coding block along with pointers to previously covered concepts. Assessments and assignments were offered using the notebooks. Curricular materials including presentation slides were offered before the class, allowing students to work on the exercises asynchronously. To facilitate a review of the previous thematic section, a video covering the major topics was offered before the lesson. Google Classroom allowed for seamless integration of all these technologies, with automatic distribution of the Jupyter Notebook files containing the course materials as well as recording student progress. From the students' perspective, this solution allowed the use of only a few clicks to both navigate the course and launch the Colab editor to do their coursework.

An initial list of course topics were identified via polling, informed discussions, and the general format used in Python education. Topics were developed in consultation with the Economics program at Texas A&M. Learning objectives, learning outcomes, and assignments were developed for each section. At its core, the format was modularized to enable easy adoption in teaching scenarios. Toward achieving this goal, the identified course topics were divided into thematic sections as described in Table 1. Each thematic section included three 50-minute lectures with hands-on exercises. Each lecture was divided into ten-minute modules that were mapped to other modules in the course, helping revisit topics later in the course. The last two thematic sections covered several topics introduced during the course. A detailed registry of each module's dependency on previous modules was developed. As such, a future instructor could mix and match these modules to create a new course or grab all the modules that lead to an advanced topic. The course included mandatory and optional assignments for each minute lecture. Assignments gradually built up in difficulty level, offering opportunities for differentiated learning. Students could choose to work on the take-home exercises or see the solutions by double-clicking on a cell. Participation was tracked at multiple points throughout each day, which determined course pass/fail for students. To facilitate student retention and participation, office hours were offered by the instructors during the week. To facilitate continued engagement on the conclusion of the course, HPRC leveraged its "Bring Your Own Science," a one-on-one researcher engagement service to work with student groups on their group projects. Several measures were considered to ensure that students participating using the remote option had an enriching experience. Adopting the best practices developed in our "Primers" and "Technology Laboratories" approaches, we maintained live chat via a peer moderator, online help offered via breakout rooms on Zoom, and the option to participate in remote office hours.

An important aspect of the development of this course was that it was not the sole work of any individual, but rather a collaborative effort of several instructors. This allowed for a diverse offering of teaching styles and helpd to ensure that relevant examples would be included at all stages of the course.

Table 1a. Concepts covered during the course. Each thematic section includes three 50-minute lectures that have accompanying, in-class, hands-on exercises and take-home assignments.

Thematic Section	Topics Covered
Introduction	Google Colaboratory, Variables
	Files, Data Types
	Dates and Times, how to use Functions, User Input
Algorithms	Operations
	Blocks, Control Structures
	Control Structures, Errors
Data Structures	Lists and Strings
	Lists, Loops, Dictionaries, Classes
	Arrays
Data Tools	Python libraries, Scatter plot, Line plot, Subplot, Candlestick plot
	Series, Index, Values, DataFrame Creation
	DataFrame Entry Retrieval, Filtering, Sorting
Data Analysis	DataFrame histogram, Missing and duplicate data handling
	Merge DataFrame (left, right, outer, inner)
	Linear regression, Train data, Test data, Predict, Accuracy

Table 1b (continuation of Table 1a).

Thematic Section	Topics Covered
Data Scavenging	Web-scraping, HTML, Tags, Browser Inspect
	Requests library, FRED API (short for Application Programming Interface)
	Beautiful Soup

3. RESULTS

The course was offered in Fall 2021 to the first-year graduate students enrolled in the Economics program at Texas A&M University. Enrollment was limited to 70, with the majority of students preferring to attend the sessions in-class. 69 students attended the first day of classes, with 47 students completing the course. The course was evenly structured in two learning components. In the first half of the course, we offered three thematic sections that covered the Python programming basics over ten contact hours. The second half covered different applications of Python for students of Economics over another ten hours. This format lent the course to two continued education credits. Details of each section are described in Table 1, and the course syllabus is included as supporting information. The course culminated with capstone exercises that used the Beautiful Soup library to "Web scrape" a website with economic data and used finance and plotting libraries to generate candlestick charts. These exercises offered an opportunity to reinforce concepts that the participating students had learned during the thematic sections.

3.1 Learning Outcomes

3.1.1 Orientation and Introduction

- Motivate the use of Python for Economics
- Familiarity with Jupyter IDE via Google Colab
- Understand general programming concepts
- Know what Python is, where it comes from, how to use

3.1.2 Programming Skills

- Import from external libraries (e.g. numpy)
- Use the assignment operator
- Inspect variables with print()
- Read and write a simple file
- Handle common types of data
- Inspect variables with type()
- Using functions with arguments
- Handle dates and times with numpy.datetime
- Get user input
- Apply mathematical rules (order of operations)
- Apply logical rules (comparisons)
- Define control structures with whitespace
- Use functions, loops, and conditionals to implement algorithms
- Handle errors
- Organize data into simple data structures (string, list, array)
- Interact with data structures (index, slice, mask)
- Integrate data structures into program control (loops, array operations)
- Organize data into advanced data structures (dictionary, class)
- Read HTML to locate data in web page code

3.1.3 Data Skills

- Visualize data with Matplotlib
- Create scatter plot, color map, best-fit-line
- Organize data with Pandas data structures
- Manipulate data with Pandas data methods
- Handle missing data
- Analyze data with Pandas data methods
- Create linear regression models with Scikit-Learn
- Retrieve data from the Web
- Parse HTML format to extract data
- Organize Web-scavenged data into data structures
- Make observations about data and adapt algorithms to match

Table 2a. In-class exercises and take-home assignments for additional learning. * indicates a take-home exercise.

Topics and exercises covered	Topics and exercises covered
Example Assignment	National Economics Data*
Hello World	Classes demo*
Your First Variables	Talking Cats*
Variables Quiz*	Array Basics
Text Files (preview)	Array Operations*
Common Variable Types	Scatter Plot
How to use Functions	Line Plot
Datetime	Subplots
The Droid*	Color Plots
Data Type Quiz*	Series
User Input*	Creating a DataFrame method
Story Generator*	Retrieve and Drop Rows
User Input Quiz*	Select, Filter, and Sort Rows
Arithmetic and Comparisons	Read/Write files
Units of Time*	Group data
Operations Quiz*	DataFrame plots and histogram
Functions	Missing and duplicate data*
Conditionals	Merge data*
More Conditionals	Pandas DataFrame*
Compute Pi*	Matplotlib Pandas*
Control Structures Quiz*	Candlestick Plot*
Errors and Files*	Linear Regression
Calculator	HTML
String Index	Pandas HTML
List Properties	Requests
List Logic	FRED API
List Loops	Regular Expression
Capstone for Lists and Strings	Beautiful Soup and Pandas for web-scraping

Table 2b. (Continuation of Table 2a).

Topics and exercises covered	Topics and exercises covered
Capstone with Dictionaries and Libraries	Candlestickplots

3.2 Description of Course Content by Thematic Section

3.2.1 Introduction

Students were first introduced to Python using the Jupyter Notebook provided by Google Colaboratory. These exercises built up their understanding of programming concepts and Python language syntax. The most important topics were covered in class, while a few were provided as take-home assignments. All of the exercises were directly related to future assignments that depended on these fundamentals. Particular emphasis was placed on the Numpy datetime64 data type as an example to support the future lessons on time series data. The use of files was introduced early because they would be critical for data analysis exercises later. Modules were introduced early despite not being traditionally considered a fundamental topic because of their massive importance in future lessons. Students were shown how modules could be imported (i.e., import <module_name>) and incorporated within their own code. In this course, the Numpy (the module/library introduced earlier) arrays were extensively used to generate data in Pandas and the Matplotlib exercises.

3.2.2 Algorithms

Students practiced with several topics that share a dependence on the Python language syntax element called indentation. These are the control structures: functions, conditionals, and loops. In practice, control structures are primarily used for the implementation of algorithms, which is not a primary focus of Economics research. Thus, these topics were less connected to future exercises. However, we still made use of them, when possible, to reinforce that learning. The take-home exercises for this session reinforced the use of files and built upon one another to form a sort of mini-lesson in themselves.

3.2.3 Data Structures

The most relevant topic for Economics research is the Data Structure, which is a strategy for keeping large amounts of data organized for effective processing. In Python, these structures are the List, Dictionary, and Array (with NumPy). These exercises depended on existing knowledge of Python fundamentals, most especially data typing and interacting with files. The exercises here were in the next session to build a useful network of tools that can handle time series data.

3.2.4 Data Tools

Students were introduced to two commonly used data handling libraries: Matplotlib [10] for visualization and Pandas [9] for data structure. Matplotlib was introduced first, leaning heavily on the Arrays lesson previously covered in order to handle the data that was to be visualized. The example data to be visualized was selected from publicly available Housing market data for relevance to Economics. This same data would be revisited in a future session featuring linear regression tools. Pandas includes two data structures, the simpler of which is the Series, a one-dimensional labeled array, and multiple Series together form a DataFrame, which is the most used structure. These data structures directly combined the elements taught in the previous session, especially the lessons on Dictionaries and Arrays. The primary kind of data in Economics research is a time series, so an emphasis was placed on those by including them as examples in many lessons. This built upon the Numpy [8] datetime64 introduced in the first session. This data type integrates with the Pandas [9] dataframe custom index feature, which is a staple method of handling time series data. The time series concept was revisited in an advanced exercise, the candlestick plot, a time series finance data visualization from the popular mplfinance [11, 12] library that builds upon the Pandas time series data structure.

3.2.5 Data Analysis

In addition to the data structure provided by the Pandas library, students were taught how to perform several basic data analysis operations using the same library. This included manipulation of a dataset to collect data into groups, add and remove elements, and filter to search for elements meeting certain logical criteria. This built upon concepts introduced in the second session where the operations that were previously used with conditional control structures were recycled for this new purpose. To tie in with economic research, the example data for manipulation practice in this session were chosen to be a sample of National Economic Data, which reprised the data set previously introduced in a Dictionary-focused take-home assignment.

Linear regression is one of the primary techniques economists use to determine correlations between different variables. We introduced students to linear regression using visualizations and a practical example. Students were also introduced to a machine learning package (Scikit-learn) and its functions. Apart from Scikit-learn, the rest of the code was built upon the modules that students learned during preceding sessions (Pandas, NumPy, and Matplotlib). A Housing Price dataset example was used to give participants a feel for visualizing, cleaning, and manipulating real-world economic data using Pandas.

3.2.6 Data Scavenging

Nowadays, most data, for example economics data, live online. Web scraping is an efficient method to collect data for research, sales, and marketing, popularity comparison, etc. Students were introduced to common Web data concepts including HTML basics, HTTP requests, and the JSON data structure. These naturally built upon previously taught skills, especially functions and data structures, including Pandas.

Federal Reserve Economic Data (FRED) [13] is a public resource hosted by the St. Louis Federal Reserve bank. This is commonly used by economics researchers as a free source of economics data. This specific API was also requested by faculty in the department. FRED API is a traditional Web API which can be handled easily in Python. Data is downloaded in the JSON format, which integrates easily into the Python dictionary data structure. This example was used as an advanced exercise building upon previous Python concepts while also priming students for their future studies using the same data source.

While APIs offer a straightforward means to retrieve data, we note that most websites do not have APIs for data extraction. For more general Web-data extraction, we covered the Beautiful Soup [11] library. Beautiful Soup is a Python library to pull data out of HTML files. The example use case for this lesson was salary statistics of a given job position in one or more cities.

3.3 Feedback from Economics Graduate Students

Surveys released each week collected student impressions of the material covered that day. Student impressions were generally positive about the relevance of the materials, interest in the topics, and opportunities offered by the course. There was a significant interest in converting this learning opportunity into a certificate that could be digitally displayed on professional social media sites such as LinkedIn. The students were able to follow along and use the Google Classroom platform effectively. From the survey responses, the most requested change was to slow down, because the syllabus was too ambitious in its pacing. Incorporating this request into the course plan resulted in omission of the capstone project and a few other exercises scattered throughout. We were pleasantly surprised to receive no comments criticizing the relatively advanced nature of the API data retrieval and Web-scraping exercises.

For the learning outcomes, 100% of the students who participated in the survey agreed that they were able to learn what they expected. Take-home assignments for continued learning were read but not graded. We observed that assignments were largely attempted the night before the previous session. Given the opportunity, we learned that offering solutions on the Web platform may dissuade students from working on the problems. To encourage more discussion, this feature was removed in later weeks. Students were now asked to attend office hours to get the solutions to the advanced problems. As such, student activity with regard to take-home assignments for continued learning was found to be unenthusiastic. Many students did not attempt the at-home exercises, citing that their other courses kept them too busy. This is an unfortunate but predictable result of the choice to do participation-only grading. This in turn had a negative impact on attendance at our hours, too, since the harder take-home assignments were designed to foster a discussion during office hours. In the future, homework should not represent a significant part of the curriculum unless true grading can be offered.

4. CONCLUSION

During this course, students were introduced to fundamental competencies and subject-specific applications in Python programming. This 20-contact-hour course built on an easy-to-use Web-based platform and represented a scalable opportunity for programming language instruction targeted at students of specific disciplines. Here, we showed that as we work to broaden participation in computing, tailoring examples and exercises to the interests of the student body increases student-engagement and facilitates student learning. The format gave us opportunities to include modules that increased awareness of cyberinfrastructure practices such as code optimization and parallel programming that are common in research computing. This will dovetail into our current series of HPRC courses that cover topics in Artificial Intelligence and Machine Learning. A tailored approach can focus on a subset of topics that lead to a discipline-relevant final project, which offers a feeling of accomplishment while serving as a capstone exercise in a certification-styled effort. An asynchronous version of the course is under development, which will include videos that accompany class slides and the Google Colab notebooks. This version will be made available to the community for wider adoption. A version with mandatory assignments and a capstone will be offered for certification.

5. SUPPORTING INFORMATION

The slide decks and some of the training materials used in this study are available to the community via the Texas A&M HPRC website [14]. The course syllabus is included as supporting information. Please send us feedback about your adoption experience via an email to help@hprc.tamu.edu.

6. ACKNOWLEDGEMENTS

This work was supported by the Department of Economics at Texas A&M University, the National Science Foundation (NSF) award number 1925764, “CC* Cyberteam SWEETER”, NSF award number 2019129, “MRI:FASTER”, NSF award number 1730695, “CyberTraining: CIP: CiSE-ProS: Cyberinfrastructure Security Education for Professionals and Students,” NSF award number 1818253, “Frontera: Computing for the Endless Frontier,” and NSF award number 2019136, “CC* BRICCs: Building Research Innovation at Community Colleges.”

7. REFERENCES

[1] Chakravorty, D. K., Pennings, M., Liu, H., Rodriguez, D. M., Jordan, L. T., Ghaffari, N., and Le, S. D. 2019. Effectively Extending Computational Training Using Informal Means at Larger Institutions. *Journal of Computational Science Education* 10, 7 (Jan. 2019), 40–47. DOI: https://doi.org/10.22369/issn.2153-4136/10/1/7

[2] Chakravorty, D. K., Pennings, M., Liu, H., Wei, Z., Rodriguez, D. M., Jordan, L. T., McMullen D.F., Ghaffari, N., and Le, S. D. 2019. Evaluating Active Learning Approaches for Teaching Intermediate Programming at an Early Undergraduate Level. *Journal of Computational Science Education* 10, 7 (Jan. 2019), 61–66. DOI: https://doi.org/10.22369/issn.2153-4136/10/1/10

[3] Chakravorty, D. K., Pennings, M., Liu, H., Thomas, X., Rodriguez, and Perez, L. M. 2020. Incorporating Complexity in Computing Camps for High School Students - A Report on the Summer Computing Academy Program at Texas A&M University. *Journal of Computational Science Education* 11, 1 (Jan. 2020), 12–20. DOI: https://doi.org/10.22369/issn.2153-4136/11/1/3

[4] Chakravorty, D. K., and Pham, M.T. 2020. Evaluating the Effectiveness of an Online Learning Platform in Transitioning Users from a High Performance Computing to a Commercial Cloud Computing Environment. *Journal of Computational Science Education* 11, 1 (Jan. 2020), 26–28. DOI: https://doi.org/10.22369/issn.2153-4136/11/1/5

[5] Chakravorty, D. K., Perez, L. M., Liu, H., Yosko, B., Jackson, K., Rodriguez, D.M., Trivedi, S.H., Jordan, L.T., and Le, S.D. 2021. Exploring Remote Learning Methods for User Training in Research Computing. *Journal of Computational Science Education* 12, 2 (Feb. 2021), 11–17. DOI: https://doi.org/10.22369/issn.2153-4136/12/2/2

[6] Lawrence, R.M., Pham, T. M., Au, P. T., Yang, X., Hsu, K., Trivedi, S.H., Perez, L.M., and Chakravorty, D. K. In press. Expanding Interactive Computing to Facilitate Informal Instruction in Research Computing. *Journal of Computational Science Education*.

[7] He, Z., Tao, J., Perez, L.M., and Chakravorty, D. K. In press. Technology Laboratories: Facilitating Informal Instruction for Cyberinfrastructure infused Data Sciences in Virtual Settings. *Journal of Computational Science Education*.

[8] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … and Oliphant, T. E. 2020. Array programming with NumPy. *Nature* 585, (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[9] McKinney, W. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference* 445, 51–56. DOI: https://doi.org/10.25080/Majora-92bf1922-00a

[10] Hunter, J. D. 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering* 9, 3 (May–June 2007), 90–95. DOI: https://doi.org/10.1109/MCSE.2007.55

[11] Richardson, L. 2007. Beautiful Soup Documentation.

[12] Goldfarb, D. 2019. Mplfinance Documentation.

[13] 1997. FRED, Federal Reserve Economic Data. St. Louis, MO: Federal Reserve Bank of St. Louis. Software, E-Resource. Retrieved from the Library of Congress, https://lccn.loc.gov/98802805.

[14] Texas A&M High Performance Research Computing. Python for Economics Graduate Students. Retrieved from https://hprc.tamu.edu/events/workshops/2021-09-10-PyEcon.html

APPENDIX: REPRODUCIBILITY

Figure 1. "Hello world" notebook in Google Colab, first few cells.