Design and implementation of a programming course embedded in civil engineering curriculum

Kaycie Lane[1,*] and Jason Hawkins[1]
[1]University of Nebraska-Lincoln, Lincoln, Nebraska, USA
[*]Corresponding author: kaycie.lane@unl.edu

Abstract

While coding classes have long been a component of engineering curricula, the integration of modern scripting language courses directly into civil engineering curriculum, taught by civil engineering professors has been slow to catch up to the big data needs of the industry as a whole. This study documents and briefly evaluated the design and preliminary implementation of a Python-based data analysis course into the sophomore-level design spine curriculum at the University of Nebraska-Lincoln. Using backwards design principles, this study used the principles of project and applications-based learning to identify a list of both engineering and coding skills and learning objectives for a new Civil Engineering Analysis II (CIVE 202) two-credit hour course. Learning objectives were then organized into five civil engineering specific projects across the civil engineering sub-disciplines of environmental, structural, transportation and geotechnical engineering to provide students with clear links between coding skills and potential real-world applications in their future careers. Each of the projects focused on a different data analysis skill (organization, analysis, visualization, interpretation and complete project integration) iteratively used by students in each subsequent project in the course. Class policies such as an AI policy and project report resubmission policy are also detailed in addition to a presentation of the overall engineering and coding skills that serve as the backbone for this new course. Finally, challenges, successes and opportunities for improvement are presented for future implementation of this course.

Keywords: Python programming, civil engineering, course design, programming in engineering

Introduction

In order for students to be competitive and successful graduates from civil and environmental engineering programs, students require the ability to understand and apply coding skills both to data analysis and engineering design calculations. Furthermore, graduates need to not only know how to code general syntax in a specific software coding language, but also a more comprehensive understanding of how to: (1) perform data analysis on large civil and environmental engineering data sets, (2) translate general engineering principles into automated functions for design calculations and (3) be able to interpret code written by other programmers, engineers and new artificial intelligence (AI) programs that will become ubiquitous for upcoming graduates. As a result, coding classes dissociated from engineering applications do not provide students with the necessary coding skills that translate to high performance in the field after graduation. There is therefore a need to develop coding courses embedded in engineering curricula that emphasize the application of coding skills to real-world civil and environmental engineering problems.

Currently, resources and studies on coding courses taught specifically within civil and environmental engineering curricula are sparse. A study performed by Grajdura and Neimeier, 2023, interviewed several civil and environmental engineering departments across the United States and found that historically, coding has been taught in MATLAB, C and C++ but that departments have been slow to

adopt modern scripting languages such as Python and Java [1]. Furthermore, there has been little integration of coding courses into the civil engineering curriculum, with coding courses taught by an adjacent department and not by civil engineering professors [1]. While previous data sets were manageable in spreadsheets and teaching spreadsheet classes is a norm across civil engineering departments [1], improvements in sensors that collect real time data and improvements in mapping large amounts of data (GIS), monitoring data (large EPA and USGS databases) and the overall emergence of big data sets has driven the need to teach students how to manipulate large datasets through flexible coding languages [2], [3]. Thus, there is currently dissonance between curriculum norms associated with coding and the future needs that have been identified in the civil engineering industry at large.

Literature also suggests that it is becoming increasingly critical to teach students how to understand algorithmic syntax, that is the structure of code and the process of data analysis, rather than simply how to generate a "correct" result from a single function [4]. Programming has been shown to help student find algorithmic solutions to problems when implemented correctly and Python provides a unique, general-purpose coding language with a small core of intuitive commands that allow even novice coders to access a wide range of functionality [5], especially when compared to MATLAB or C++. Most students struggle to explain how functions work in coding languages such as MATLAB and C++ [6], but the dynamic and intuitive nature of Python may contribute to a greater understanding of the purpose of the developed code [4]. Literature clearly points to the need to teach the integration of coding functions into the overall data analysis process [7] and that teaching in an algorithmic way may increase students acceptance of the need to code as engineers [4].

Literature from across engineering disciplines suggests that students "buy-in" to courses such as coding courses or fundamentals courses in engineering (such as calculus, physics, etc.) when the concepts from the course are directly tied to engineering applications, even at a basic level in freshmen and sophomore level courses [4], [5], [8]. At the University of Nebraska- Lincoln (UNL), the implementation of a new "design spine" curriculum embeds Python programming in the second semester sophomore class titled CIVE 202: Civil Engineering Analysis II. The design spine was created to integrate students into major-specific courses every semester within a four-year program, encourage relationship building amongst students and to teach industry-level skills identified by the department's advisory board and future employers that are becoming critical for young engineers entering the workforce. CIVE 202 was structured as an open-sourced coding class focused on using coding in general to solve civil engineering projects, giving it quick a wide range of possible skills and topics to cover. After reviewing relevant literature [4], [5], [8], [9], [10], the department selected Python as the coding language because it is open-source (Eliminating license fees normally not paid by consulting firms for proprietary software) and because Python is integrated in backend GIS programming and being currently integrated into Excel [11]. Prior to CIVE 202, students will have taken an introductory scripting course in Python through the computer science department (thereby exposing students to entry-level coding prior to civil engineering specific coding) and to visual basic code in CIVE 201 (Civil Engineering Analysis I) in the preceding semester. The overall aim of the design spine is to integrate skills such as Python coding into real-world civil engineering applications, focused on projects and applications-based learning, which provided the ethos for the design of CIVE 202 in this study.

The objectives of this paper are therefore, as follows:

1. To clearly outline learning objectives that focus on both engineering and programming skills in line with the ethos of the UNL Design Spine
2. To design a clear and logical project-based course structure that can be adapted and enhanced in future iterations of the course.
3. To document observations from initial implementation of the course in the Fall 2024 semester in two locations (Omaha and Lincoln) and to provide other UNL faculty teaching upper-level courses with a list of skills students should theoretically have obtained after passing the course
4. To document challenges, successes and opportunities for future improvement

Course Development Methodology and Results

Course Structure

CIVE 202 is a 2-credit hour course taught as a lecture-lab format at UNL on both the Lincoln and Omaha campuses. The class meets for a 50-minute lecture once a week, followed by a two-hour lab once a week on the same day. The course is a requirement for civil engineering majors entering the major after Fall 2023 and was first taught in Spring 2024. The class is taught synchronously and in-person, with occasional guest lectures from industry as relevant to the content.

Course Design

CIVE 202 was designed in the Fall 2023 semester through a UNL workshop entitled "Learning by Design" presented by the Engineering and Computing Education Core. The workshop focused on backwards design principles [12] and was attended by one of the professors for the Spring 2024 CIVE 202 course. The Learning by Design workshop focused on three key steps in backwards design: (1) identify the desired results, (2) determine acceptable evidence, and (4) plan learning experienced and instruction, using concepts from "Understanding by Design" [12]. Over the course of the workshop, CIVE 202 instructors identified department-level requirements, accreditation-level requirements, general data analysis skills, and engineering skills and translated these requirements into a set of learning outcomes for CIVE 202. The instructors generated a list of coding and engineering skills using a previously developed syllabus for the class and their own personal experiences coding big data analysis projects in R and Python for past research and consulting projects. Instructors wanted to emphasize not only how to sequence and write code, but also how to present code and to understand what types of tasks students may need to code using real-world consulting and project-based examples that would require students to understand how to translate project prompts into code. Then, using these learning outcomes, the instructors assigned each learning outcome to one of the six levels on Bloom's Taxonomy of Cognitive skills [13]. These learning outcomes were then grouped into five data analysis skills: (1) organizing and collecting data, (2) analyzing data, (3) visualizing data, (4) interpreting data, and (5) integrating the previous four skills into a comprehensive project. The methodology for creating learning outcomes and assigning each learning outcome is summarized in Figure 1 for Project #1 (Organizing Data).

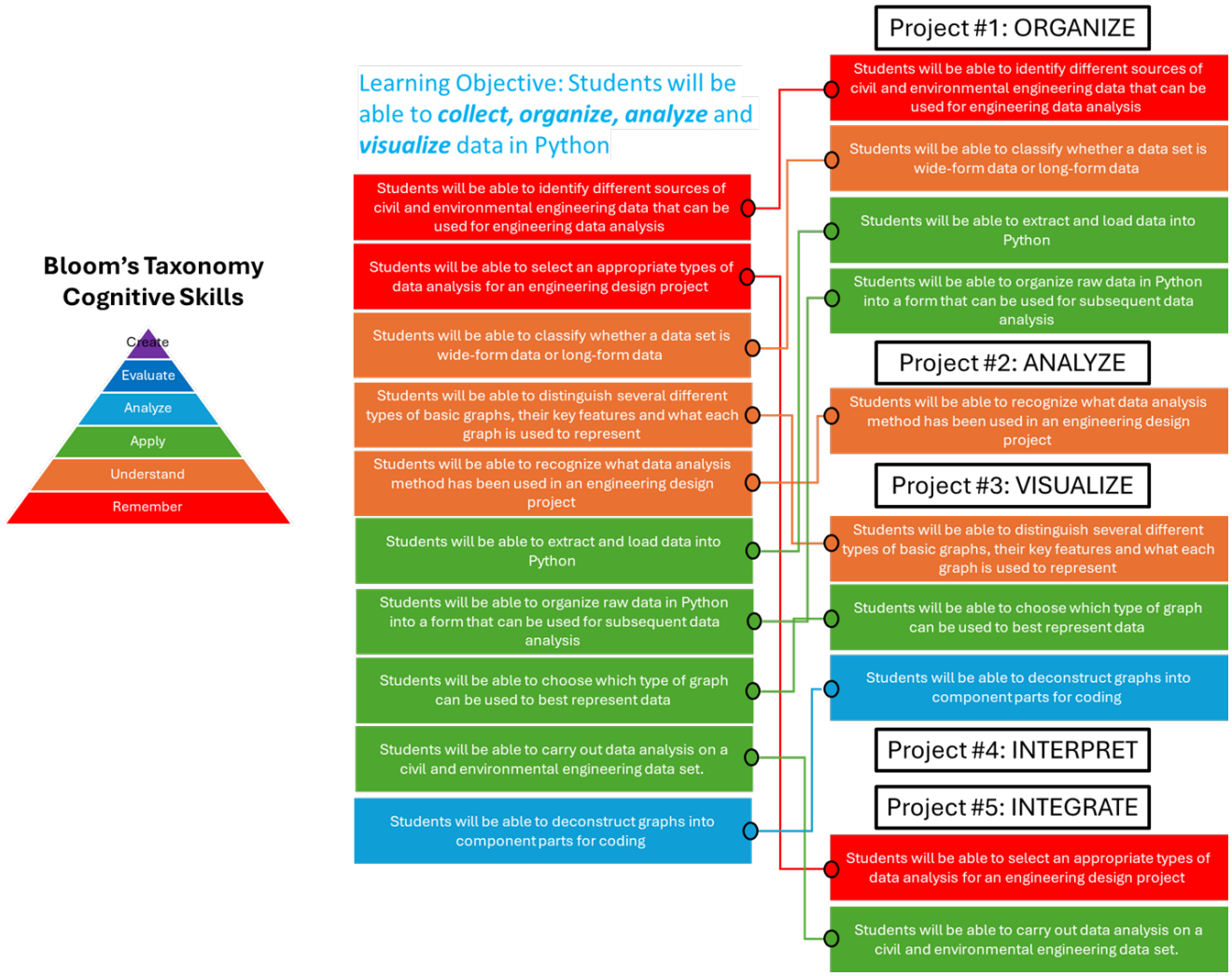

Figure 1: Methodology used to create learning outcomes and then assign learning outcomes into each project by the data analysis skill. This figure presents only Project #1 for brevity, but a similar process was used for the additional four projects.

Each of the learning objectives for both programming and engineering skills were next aligned with 5 project topics related to data analysis workflow: (1) collecting and organizing data, (2) analyzing data, (3) visualizing data, (4) interpreting data, and (5) integrating the previous four steps into a cohesive coded solution for a client. We organized learning objectives into each project and then organized the engineering skills in a logical order, adding one engineering skill per project and iterating the engineering skills in each subsequent project to ensure students were comfortable with sequencing the steps in an engineering project by the end of the semester. Figure 2 shows the engineering and programming skills associated with each project. For example, in Project #3, we introduced engineering timesheets. A scope of work was introduced in Project #1 and a Gantt Chart was introduced in Project #2; therefore, for Project #3 final report to the "client" the students needed to submit a scope of work, Gantt Chart and an engineering timesheet.

Next, using the learning objectives, engineering and coding skills, we ordered the learning objectives using both Bloom's Taxonomy of cognitive skills and our own experiences teaching data analysis within the three-week time frame we allocated for each of the five projects. An example of this structure from

Project #1 is shown in Table 1. Then, using the learning objectives for each week, we outlined the structure of each project, focusing on deliverables we wanted students to produce by the end of each project. Then, knowing the final deliverables, we planned lecture and lab activities to match and provide students with sufficient exposure to adequately complete the client prompt detailed in the project. Based on the lecture and lab activities, we then designed pre-lecture and pre-ab activities (described in more detail below) designed to help students practice engineering and coding skills outside of class time. Finally, to ensure each project was relevant to and grounded in civil engineering practice, we assigned a civil engineering sub-discipline to each project. Because students were sophomore-level in this course, we attempted to cover a breadth of topics to expose students to the types of civil engineering problems that can be solved in Python.

Table 1: Outline of activities for Project #1 based on learning objectives, project topic and engineering and programming skills.

Week	Learning Goals	Lecture Content	Lab Content
1	Students will be able to ***match*** engineering operations (such as add, subtract, multiply, divide, integrate, differentiate, etc.) to their corresponding Python function syntax.	Review of Python basic functions Engineering skill building – writing a scope of work, outlining key functions to code	Practices operations such as mean, median, etc., That are needed for the final project report on a clean set of data (a different state in the US) Translate the project prompt into a scope of work (have the professor pose as the client and teach students how to ask clarifying questions to their "client") Turn in by end of lab: code showing mean, max, min and median calculations on practice data set. Turn in before next week: individually draft a scope of work
2	Students will be able to ***identify*** different sources of civil and environmental engineering data that can be used for engineering data analysis. Students will be able to ***classify*** whether a data set is wide-form data or long-form data. Students will be able to ***extract*** and load data into Python. Students will be able to ***organize*** raw data in Python into a form that can be used for subsequent data analysis	Preparing and locating civil engineering data Organizing data	Have students pull relevant data from SDWIS. -use code to translate from long to wide form data -Learn how to set up working directories and projects -different ways of importing data into Python Organize project data to be able to replicate lab activity #1. Turn in by end of lab: -Code showing students have translated long to wide form data -Screenshot of project directory and file structure -Draft scope of work for comments
3	Students will be able to ***outline*** the key functions that their Python code is designed to perform. Students will be able to ***outline*** a basic scope of work that details "client" desires and objectives for a Python engineering project. Students will be able to ***calculate*** average, median, maximum, minimum and standard deviation in Python	Organizing data	Time to work on project before deadline. Turn in by the end of lab: -Draft python code for project with student annotations in Jupyter notebook

Course instructors met biweekly during the Fall 2023 semester to determine how the course would be evaluated, structured and implemented in Spring 2024. The Design Spine at UNL has an emphasis on promoting essential teamwork, communication and holistic thinking skills in an effort to produce more engineering graduates who are prepared for the future challenges facing the engineering industry. Furthermore, as this course is new to the civil engineering major, the instructor also wanted to ensure sufficient opportunities for student feedback coupled with flexibility to alter the schedule as needed as implementation occurred. The instructors therefore focused on five semester projects with clearly delineated rubric components in alignment with learning outcomes. Details of each project are provided in the result section and sample rubrics are provided in the Supplemental Information for Project #1.

Course Implementation

CIVE 202 was taught for the first time in Spring 2024. In Omaha, class consisted of one section with thirteen students and in Lincoln, class consisted of a combined lecture and two lab sections with a total of 28 students. Each location had one graduate teaching assistant who primarily assisted with debugging errors and helping answer questions during working time in the lab sections. The course spanned a 16-week semester, with three weeks allocated for each project and no final exam or assessment beyond the last project.

To review implementation progress, both instructors met weekly to discuss how the previous week was received by students and to prepare and plan for subsequent week activities. Because the UNL civil engineering major is taught on two campuses, the instructors focused on ensuring content was delivered as consistently as possible in both locations with minor variations based on scheduling changes and student reception and performance. Instructors documented adaptations to the course and used Microsoft Teams to communicate and share files. All course documents are clearly labelled in Microsoft Teams and available to other faculty within the department for future iterations of the course and to provide instructors teaching 300 and 400-level courses with insights into what types of Python skills students entering their course should have after having completed CIVE 202. We taught Python using online Jupyter Notebooks, having each student create an Anaconda Cloud account using their GitHub accounts established in their freshmen-level coding course.

Types of assessment activities

Pre-lecture and pre-lab activities

Because class only met once a week, we designed to small activities to ideally be completed between class periods: (1) a pre-lecture activity designed to have students interact with the client prompt to prepare for the project or to practice engineering skills such as writing a scope of work and (2) a pre-lab activity designed to help students practice coding skills learned in the previous class period. Both activities were normally less than 50 points and could be a group or individual assignment. To encourage students to practice coding on their own instead of relying on group members, we attempted to make the majority of pre-lab activities individual to encourage students to work through coding and debugging procedures on their own. Sometimes the activities asked students to set up a Jupyter notebook on their own for la or to draft a scope of work that would then be used in class lecture or lab; we used this structure to try to

encourage student preparedness in class. Pre-lecture and pre-labs contributed the most to individual student grades in the course.

Projects

The five projects constituted the majority of students' grades for the semester. Each project lasted for 3 weeks with 4 weeks allocated for Project #5 when time is available in future semesters. The following list presents each project with the corresponding civil engineering sub-discipline and a brief description of the project:

1. Project #1: Environmental Engineering – students worked with the Safe Drinking Water Information System (SDWIS) database to explore drinking water quality violations in Nebraska by source water type, community water system type and population served.
2. Project #2: Structural Engineering – students learned how to use functions to automate Statics calculations, then used their functions to simulate and report three different sets of conditions (force magnitude, direction, etc.).
3. Project #3: Transportation Engineering – students learned how to graph bar plots, scatter plots, boxplots and histograms of discrete and continuous data from the National Household Transportation Survey (NHTS).
4. Project #4: Infrastructure Access – students proposed a revised set of criteria to evaluate sanitation access in two different countries and compared their analysis to the Joint Monitoring Programme (JMP) guidelines and methods using JMP data (a collaboration between WHO and UNICEF).
5. Project #5: Student's choice of topics – students could choose among eight prescribed projects: additional SDWIS data review, automating steel beam calculations, automating concrete mix calculations (geotechnical engineering), additional NHTS analysis or using JMP data to look at either drinking water or hygiene facility access. Each group's final code and report had to include data filtering, a function students created, at least two proper graphs and a detailed description of the assumptions students used to conduct the analysis.

Each project consisted of the following elements provided to students in the first week of the project: a client prompt outlining deliverables a student group needed to provide, a rubric for the project report and any cleaned data sets needed for the project. We elected not to teach students intensive data techniques throughout the semester to keep the focus on necessary skill building related to each project; we did talk about how to deal with "NA" values and how to format CSV files and name variables for loading into Python as was necessary in each project, but overall tried to provide students with relatively "clean" datasets that they either downloaded from a repository or database or were provided by the instructors. Students were then asked to generate a scope of work for each project; in the first two projects, these largely restated the client prompt, but as skills such as constructing Gantt charts were added, we wrote client prompts more open-ended and asked for more specific elements in the scope of work to attempt to promote more student autonomy and get students used to articulating client goals in their own words. An engineering skill was added to each subsequent project throughout the semester to keep students practicing their skills during the semester (shown in Figure 2).

During the course of each project, lecture time was used to introduce how to complete each engineering skill and focused on broader concepts related to the project topics. For example, Project #4 introduced

structure to a large water, sanitation and hygiene (WaSH) database and asked them to look at both how sanitation access is assessed currently by the Joint Monitoring Programme (JMP) and to propose and evaluate a new assessment paradigm. In this project, we wanted students to understand that how we qualitatively categorize data can introduce unintentional bias when we need to interpret data to make policy decisions. Thus, lectures for this project covered the following: (1) what data is currently collected and how it is analyzed by the JMP, asking students to investigate past reports and resources online to teach them how to search for relevant information, (1) how bias can be defined in coding and the mechanisms by which intentional or unintentional bias appears in the results of a program and (3) a reflection on what students noticed about their own results compared to literature. Each project had a similar structure, focused on discussing the importance of the project to civil engineering, introducing new concepts, and then a reflection on how the coding functions and process students are learning relate to their future careers.

Lab time during each project was used to introduce students to the relevant coding functions they needed to complete each project and as work time where groups could meet to work on their projects. In the first two weeks of each project, the instructor provided a template Jupyter Notebook with a similar dataset to the project and demonstrated how to code specific sequences needed to complete the project for about one hour of the lab. The second hour of lab was reserved as time for students to try to apply the concepts to their project datasets with their groups and the instructors and teaching assistants were available to debug and answer questions. For example, in Project #3, students were asked to create a bar plot from a transportation dataset with the caveats that the bar plot had to correctly show counts of discrete data and must be customized using color or different formatting arguments than the base graphs generated by Python. The lab for the first week showed students how to plot four different types of plots using a similar dataset and reiterated concepts from lecture related to the difference between discrete, categorical and continuous data. Lab in the second week covered labeling graphs, customizing graph appearance and generally making plots of high quality to include in technical reports. Templates were used in each project to help provide students with structure and to communicate norms for how clean Jupyter Notebooks should be submitted in class. We observed in other coding courses that when students start with a blank notebook or shell to code within, students get easily frustrated and tend to reinforce their own preconceived notions of their ability to code. We thus provided templates in an attempt to keep students engaged and focused on the specific concepts presented in class, which hopefully translates to more time understanding how the code applies to civil engineering.

Reflections

At the end of each project, we asked students to complete a 1-2 paragraph reflection that focused on what skills students learned during the project (either engineering or coding skills) and how the project and these skills relate to their future careers as engineers. Reflections were individual and provided students with the opportunity for more metacognitive thinking about how they were learning coding. We observed other coding courses students take largely focus on coding correct syntax and not on the "why" of the syntax, which leads to a dissociation between the code and the intended outcome. We hope to perform theme and code analysis [14] on the result of the reflections in a later paper pending IRB approval. Reflections provided us as instructors with a wealth of timely feedback that were used to adapt developed materials to our individual students' needs which led to more tailored learning experiences in the Omaha and Lincoln locations. While we do not discuss specific reflection results in this paper, we will note that

we observed students providing honest feedback about their participation in group projects and how well they understood the class content.

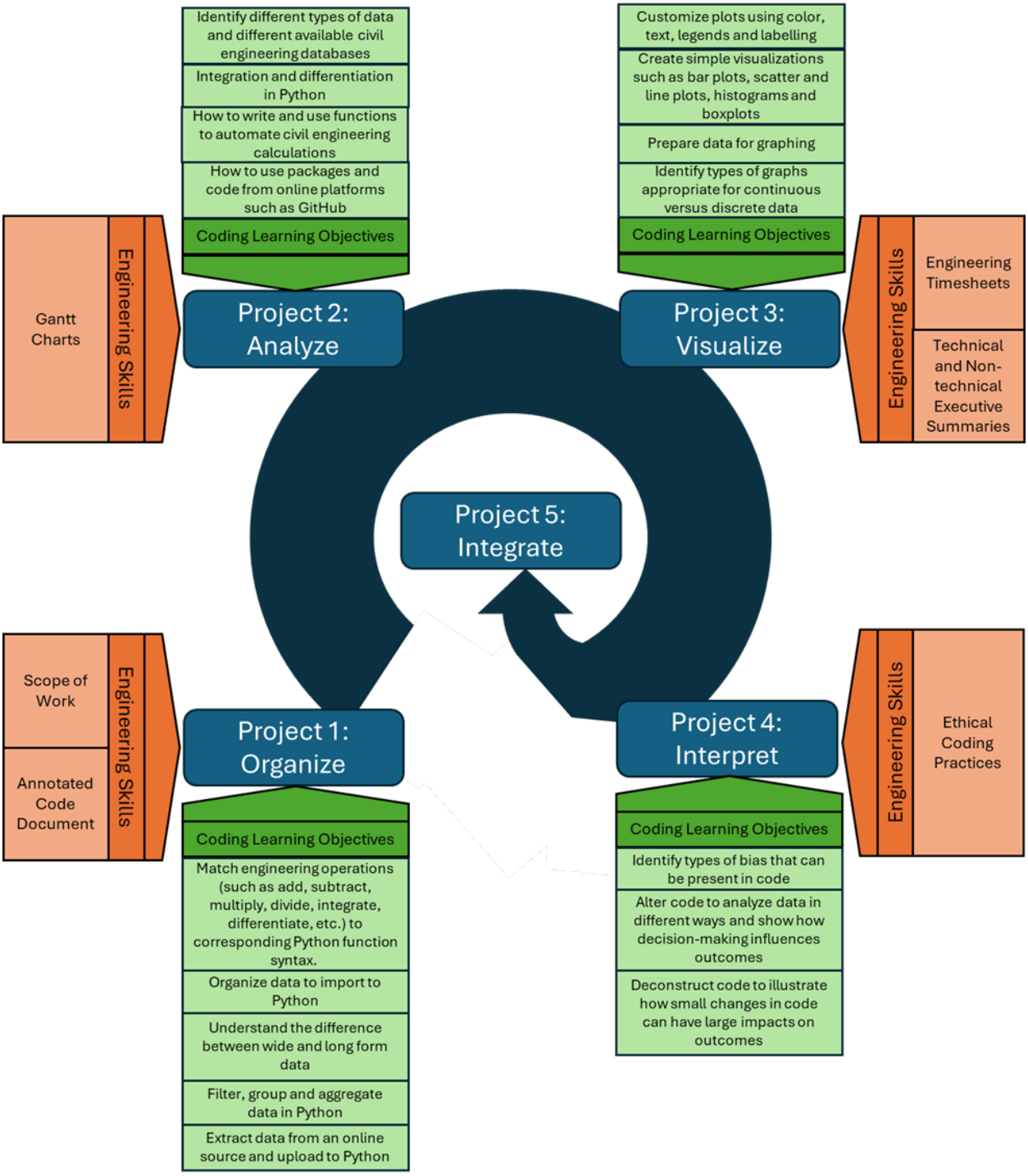

Figure 2: Final course design featuring 5 key projects within 5 skills relevant to data analysis. Each Project emphasizes a different set of engineering skills and a different set of coding learning objectives.

Class Policies

To provide additional context to class implementation, in this section we include a description of policies unique to this course that did influence the challenges we observed during implementation and may have contributed to some of the highlighted successes. Of primary note is the resubmission policy. In order to encourage proactive behavior related to receiving feedback, students were allowed to submit each assignment multiple times prior to the due date to receive instructor feedback but may not submit for revisions after the deadline. This policy was in place to mimic typical consulting and public sector project deadlines: engineers can consult with their clients on questions and feedback prior to the deadline for a request for proposals or projects, but in industry, the deadline is considered a hard deadline and failure to meet the deadline results in loss of a project, contract or potential client. Students in Omaha had been exposed to this policy in previous design spine courses, but the policy was new to Lincoln students. In general, we observed students used this policy to their advantage which resulted not only in higher grades but also better project reports in general. This policy applied not only to the final project deliverables but to pre-lecture and pre-lab activities as well.

Second, we specifically outlined an AI policy in the syllabus and provided a lecture on debugging techniques and the AI policy specifically to clearly delineate expectations (AI policy available in the SI). We outlined the following key items for students:

1. AI can NOT be used to write reports or generate significant quantities of the final code in a project.
2. AI can be used to debug or help write code but must be clearly referenced in the report and the annotated code document clearly.

To help us as instructors determine how much students used AI and also understood their code. We asked students to submit an annotated code document (example provided in the SI). In the annotated code document, students were asked to code lines of code into a table and explain each section of the code in their own words to demonstrate they understood what each function was contributing to their overall code. If AI was used to help write a function or specific component of syntax, we required students to cite the AI with a date and time accessed along with the explanation of the code. When designing this policy, we decided to acknowledge the fact that students will have access to AI tools in their future careers and wanted to put the emphasis on using AI to help write quality code students understand instead of banning AI's use altogether in the class.

Discussion

Based on our initial implementation of CIVE 202 in Spring 2024, we have provided a brief list of challenges, successes and improvements related to the course structure presented above. A more in-depth analysis of the results from implementation is currently being conducted according to IRB protocols, however, we provide a summary of our key observations to help readers evaluate whether they wish to adopt elements of the CIVE 202 course structure within their own teaching practice.

Challenges implementing CIVE 202

We observed the following challenges implementing CIVE 202 in Spring 2024 based on the structure presented above:

1. The heavy emphasis on group-projects occasionally detracted from individual student's ability to code by the end of the semester as many students relied on peers to complete coding. Using the pre-lecture and pre-lab assignment to test individual coding skills provided a temporary work-around during the semester to alleviate this issue.
2. Not enough structure was provided to teach students how to write technical reports based on the results of their code. While students do take a writing course in freshman year, they have not yet been exposed specifically to the kind of technical writing needed to write reports based on coding outputs.
3. Five projects taught over sixteen weeks with only one day of class meeting times provided a challenge in terms of pacing project due dates and lab activities. This was alleviated in part by the submission policy outlined above but perhaps fewer projects with more depth may be more beneficial to student learning long-term.

Successes from implementing CIVE 202

We observed the following successes implementing CIVE 202 in Spring 2024 based on the structure presented above:

1. Providing students with templates to work through in lab was a very helpful tool to build student confidence in their own ability to code. We observed many students did not think they could code or didn't like coding but found that templates helped them to focus on the important elements in each lab and provided a good resource they could refer to when working on their projects outside of class time.
2. Students tended to enjoy projects where they found the civil engineering discipline the most relevant to the career they want to pursue, resulting in more student engagement on specific projects. For example, students interested in transportation engineering found they enjoyed making different types of plots of the transportation survey data because they could see how the data would potentially relate to work they are interested in performing as future engineers. Having civil engineering specific projects changed student attitudes towards coding. Many students can into the semester with the preconceived notion that they would never use coding as an engineer but remarked that having a reason to code that was related to their future careers made the material learned in class more relevant.
3. The sequence of projects from organization to integration provided an iterative learning environment where students practiced skills learned in previous projects in each subsequent project, reinforcing skills learned over time.

Opportunities for improvements

In future CIVE 202 iterations, our priority improvements include:

1. Providing clearer guidance on how to write technical reports and prepare presentations based on the results of code. A clearer structure and set of expectations for students would have resulted in a better student understanding of why we need to be able to translate what we coded into a technical report.
2. Add additional small metacognitive exercises, similar to the reflections but on a smaller scale throughout the semester, to help students think about their learning processes. We noticed

through an initial review of the reflections that students seemed to have developed an awareness of when they were contributing to code and when they were letting others do the work and think that more metacognitive activities might help these students to take more control over their learning process in class.

Conclusions

Backwards design and thoughtful collaboration between the two instructors teaching CIVE 202 in the semester preceding course implementation were critical to creating a solid course structure that can be used to improve future course iterations. The overall project-based course structure with its emphasis on civil engineering applications provided with students with a clear link between syntactic coding in Python and its relevance to their future careers, helping to shift student's perceptions of how coding relates to the field of civil engineering. Providing students with templates and structured labs that teach skills relevant to specific tasks helped to put more emphasis on the process of coding as a civil engineer rather than on completing a prescribed set of steps. With additional future studies of student reflections and other relevant student data, we hope to more conclusively determine whether the civil engineering specific projects helped to shift students from thinking of CIVE 202 as a "coding class" to a course that is part of the civil engineering curriculum. While challenges such as timing, balancing individual and group assignments and unbalanced working groups did arise, many of these challenges can be improved in future iterations of the course. In future iterations of this course, we hope to improve how technical writing is taught and to provide students with more small metacognitive activities to further develop students' ability to recognize patterns in how they learn code.

Acknowledgements

We would like to thank Tareq Daher and the Engineering and Computing Educational Core (ECEC) at the University of Nebraska-Lincoln for facilitating the Learning by Design workshop that was used to help design the course presented in this paper.

References

[1] S. Grajdura and D. Niemeier, "State of Programming and Data Science Preparation in Civil Engineering Undergraduate Curricula," *J. Civ. Eng. Educ.*, vol. 149, no. 2, p. 04022010, Apr. 2023, doi: 10.1061/(ASCE)EI.2643-9115.0000076.

[2] A. Katal, M. Wazid, and R. H. Goudar, "Big data: Issues, challenges, tools and Good practices," in *2013 Sixth International Conference on Contemporary Computing (IC3)*, Noida, India: IEEE, Aug. 2013, pp. 404–409. doi: 10.1109/IC3.2013.6612229.

[3] A. P. McAfee and E. Brynjolfsson, "Big data: the management revolution.," *Harv. Bus. Rev.*, vol. 90 10, pp. 60–6, 68, 128, 2012.

[4] B. Bettin, M. Jarvie-Eggart, K. S. Steelman, and C. Wallace, "Preparing First-Year Engineering Students to Think About Code: A Guided Inquiry Approach," *IEEE Trans. Educ.*, vol. 65, no. 3, pp. 309–319, Aug. 2022, doi: 10.1109/TE.2021.3140051.

[5] A. Talaat, M. Kohail, and S. M. Ahmed, "Programming in The Context of Civil Engineering Education," Aug. 11, 2022. doi: 10.21203/rs.3.rs-1802246/v1.

[6] P. Von Lockette, "Algorithmic Thinking And Matlab In Computational Materials Science," in *2006 Annual Conference & Exposition Proceedings*, Chicago, Illinois: ASEE Conferences, Jun. 2006, p. 11.168.1-11.168.13. doi: 10.18260/1-2--725.

[7] A. Robins, J. Rountree, and N. Rountree, "Learning and Teaching Programming: A Review and Discussion," *Comput. Sci. Educ.*, vol. 13, no. 2, pp. 137–172, Jun. 2003, doi: 10.1076/csed.13.2.137.14200.
[8] Q. Cao, L. H. I. Lim, V. Dale, and N. Tasler, "EXPERIENCES IN PYTHON PROGRAMMING LABORATORY FOR CIVIL ENGINEERING STUDENTS WITH ONLINE COLLABORATIVE PROGRAMMING PLATFORM," presented at the 14th annual International Conference of Education, Research and Innovation, Online Conference, Nov. 2021, pp. 5784–5791. doi: 10.21125/iceri.2021.1305.
[9] D. G. Balreira, T. L. T. D. Silveira, and J. A. Wickboldt, "Investigating the impact of adopting Python and C languages for introductory engineering programming courses," *Comput. Appl. Eng. Educ.*, vol. 31, no. 1, pp. 47–62, Jan. 2023, doi: 10.1002/cae.22570.
[10] G. List, "Introducing Civil Engineering Analysis Through Programming," in *2007 Annual Conference & Exposition Proceedings*, Honolulu, Hawaii: ASEE Conferences, Jun. 2007, p. 12.961.1-12.961.10. doi: 10.18260/1-2--2303.
[11] Microsoft Excel, "Announcing Python in Excel: Combining the power of Python and the flexibility of Excel." Accessed: Jul. 30, 2024. [Online]. Available: https://techcommunity.microsoft.com/t5/excel-blog/announcing-python-in-excel-combining-the-power-of-python-and-the/ba-p/3893439
[12] G. P. Wiggins and J. McTighe, *Understanding by Design*. in Gale Reference. Association for Supervision and Curriculum Development, 2005. [Online]. Available: https://books.google.com/books?id=N2EfKlyUN4QC
[13] N. E. Adams, "Bloom's taxonomy of cognitive learning objectives," *J. Med. Libr. Assoc. JMLA*, vol. 103, no. 3, pp. 152–153, Jul. 2015, doi: 10.3163/1536-5050.103.3.010.
[14] J. W. Creswell and J. C. Báez, *30 Essential Skills for the Qualitative Researcher*. SAGE Publications, 2020. [Online]. Available: https://books.google.com/books?id=Mgr2DwAAQBAJ

First Author Biography

Dr. Kaycie Lane is an assistant professor of practice at the University of Nebraska-Lincoln, teaching in the civil and environmental engineering department. Her technical expertise centers around drinking water and wastewater management and assessment, with over seven years of experience performing data analysis on big data sets from the water, sanitation and hygiene (WaSH) field in the R statistical programming language course. As a teacher, she emphasizes application-based learning through project-driven courses, the application of the engineering design process to unique scenarios and the addition of socio-technical thinking into engineering education. She teaches a wide variety of classes within the environmental engineering curriculum including Mass and Energy Balances, Environmental Microbiology and Water and Wastewater Treatment Design and in the broader civil engineering curriculum on the topics of data analysis and sustainability.

Second Author Biography

Supplemental Information

PROJECT #1 – ORGANIZE AND SUMMARIZE
Client Prompt:

As part of a proposal your engineering firm is submitting to compete for a drinking water treatment project, the proposal asks for an evaluation of drinking water treatment technical capacity across the state of Nebraska. The request for proposal (RFP) from the client, Zenith Data Analysis, asks potential firms submitting to the request to use the United States Environmental Protection Agency (USEPA) Safe Drinking Water Information System (SDWIS) database to use basic statistics and data analytics to describe the current compliance status, operation and function of small water systems (serving less than 10,000 people) that fall under the "community water systems" category. The RFP specifically asks that the firm includes the following items in their analysis:

1. An overall evaluation of the number of violations experienced by each type of water system (community water system, non-transient non-community water system, and transient non-community water system) that includes the mean, minimum, maximum, and median number of violations by system type.
2. A summary of the types of contaminants in drinking water most often in compliance violation by water system type (as characterized above). This summary should also include a category for population. Population categories used by the USEPA can be found at this link: https://echo.epa.gov/help/drinking-water-qlik-dashboard-help#:~:text=System%20Size&text=Very%20Small%3A%20500%20or%20less,Large%3A%2010%2C001%20%2D%20100%2C000
3. A more detailed evaluation that focuses only on small community water systems and summarizes the total number of systems using the following identifying information:
    a. The type of source water used by the community water system.
    b. Only considers active systems.
    c. Only considers Maximum contaminant level (MCL) violations.
    d. Reports the data by contaminant name in a clearly labelled table.

Your firm asks you to use your coding skills in Python over the next 3 weeks to complete the data analysis requested in the RFP and provide a summary of the results to your professional engineer (PE) supervisor so your PE can then use this analysis in the final proposal your firm writes. The summary of the deliverables requested by your PE and the deadline are listed below.

Deliverables:

1. Raw data files used in the Python Code file.
    a. This allows your professor to run your code and provide you with detailed feedback.
2. Python code file that documents your code used to generate your solution.
3. A well-written scope of work related to the client prompt that defines the tasks undertaken by your group to complete the RFP.
4. An annotated code document that explains each line of code in detail (https://uofnelincoln-my.sharepoint.com/:w:/g/personal/klane11_unl_edu/EWQY3dadECBAppqDxlIi2UwByWjgxcJOOCsRPVudJPMq6Q?e=RDILxC)
5. A written document that summarizes the findings contained in the code that can be handed to your firm for use in a competitive application to the RFP.

All files must be submitted to Canvas by Monday February 12th, 2024, at 5:00pm CT.

Skills Students will learn:

- Review basic Python functions.
- Write a scope of work.
- How to outline functions/steps in code
- Identify sources of data
- Identify types of data
- Extract and load data into Python.
- Basic operations in Python to compute statistics.
- How to organize raw data

Figure S1: Example Project #1 client prompt given to students at the beginning of Project #1

Table S 1: Example Rubric from Project #1

Component	Description	Points Possible	Points earned	Instructor Feedback
Raw data files formatted and submitted	Raw data files used in the Python Code file are formatted and submitted to Canvas	5		
Python code file	Python code file (.py file) is submitted to Canvas, can easily be opened and accessed by the instructor and contains all of the code used to generate the solutions in other deliverables	10		
Scope of Work	See scope of work rubric for more details	50		
Annotated Code Document	Please see the rubric for the annotated code document on Canvas	20		
Written summary	Written summary contains: -Tables of final results -Clearly written paragraphs explaining the methods used to generate the results -Clearly written paragraphs explaining the results of the study	35		
TOTAL		120		

Table S 2: Example annotated code document format.

Python Code	Annotations
import pandas as pd ne_facilities = pd.read_csv('ne_facilities_summary.csv') ne_raw_facilities = pd.read_csv('ne_raw_facilities.csv')	This section of code imports the "pandas" package using the import function. Then two different CSV documents are uploaded to the Python environment using the pd.read_csv() function
print(ne_facilities.columns())	This line of code prints the column headers of the data to see what data exists in the dataframe

ne_raw_facilities[ne_raw_facilities.treatment_objective == 'Inorganics removal']	This line filters the data set for only treatment objectives that are equal to Inorganics removal

Policy on Use of AI

In this class, we will be using code and computers extensively. Artificial intelligence (AI) generators such as ChatGPT and many other platforms all can help make your assignments easier but need to be used responsibly. In this class, we allow the following when it comes to AI generators:

1. AI generators may be used to check your solution to a problem and to troubleshoot errors in your code. AI generators may not be used to generate the body of code used in your solution – you need to attempt to write the code first and then use AI to help you improve your code.
    a. Please note that ChatGPT can tell us as your instructors if your code was generated by Chat GPT or another AI. We will be checking your code on each project.
2. Any time AI generators are used to generate any part of the solution to a problem, you must reference the AI generator, date and time that the solution was obtained and clearly delineate what pieces of code in your solution were generated by the AI generator. This reference must appear in both an in-text citation in a report and in a reference list.
    a. References must adhere to the APA citation style, 7th edition: https://owl.purdue.edu/owl/research_and_citation/apa_style/apa_formatting_and_style_guide/general_format.html
3. AI generators may NEVER be used to generate the text in a report – all of the written text in your report needs to be written by you so we can verify you understand your code and can accurately present the results of your code to real-world engineering clients. The purpose of this class is to teach you how to use coding practices to your advantage so that you can solve engineering problems. To that end, you need to also be able to communicate with your co-workers and clients how your code works.

Figure S2: AI policy used in CIVE 202