

Teaching Portfolio

Jason L. Weber, M.S.
CS PhD Student | Teaching Assistant

September 29, 2023

Table of Contents

1	Teaching Philosophy Statement	1
2	Teaching Techniques	4
3	Formative Assessment Practices	6
	Appendix A First Python Program Activity	7
	Appendix B From UML to Code Formative Assessment	12
	Appendix C Curriculum Vitae	14

1 Teaching Philosophy Statement

As a Computer Science (CS) educator, my goal is to facilitate student learning by creating an engaging and supportive classroom environment. I believe that CS education at its core is not just about acquiring computing knowledge, but about equipping students with a new method of perceiving and solving problems, often referred to as “computational thinking.” To achieve this, I embrace an andragogical approach, recognizing that adult learners have unique needs and motivations that should inform my teaching methods; that is, I focus on adults’ desire to have their learning grounded in real-world application and their move towards self-directed learning as their intrinsic motivation develops. I also utilize a variety of novel instructional techniques rooted in contemporary teaching theory.

The first of andragogy’s assumptions that I focus on in my teaching is that adult learners learn best when the content their learning is connected to applications in their career. To engage my students, I focus on creating authentic programming tasks that simulate real-world scenarios. I believe that students are more motivated to learn when they can see the relevance and practical applications of what they are studying. For this reason, I prioritize multi-week, scaffolded programming projects as the primary form of summative assessment. These projects allow students to develop their programming skills incrementally, building on their prior progress and knowledge as each week passes. By the end of the term, students have a sense of accomplishment and confidence in their ability to apply what they’ve learned to large, authentic problems.

In addition to these long-term summative assessments, I try to ground low-stakes, formative assessments in authentic performance tasks as well. For example, my “From UML to Code” (see Appendix B) activity tasks students with writing some code based off of a Unified Modeling Language (UML) class diagram. Since most CS majors will be seeking out careers in software engineering, I choose to give students a UML diagram that represents a possible project management system. The code students write for this assignment is very close to the actual code that would need to be written in a production setting. My students respond well to this, as many have reported that they prefer this theming of assignments over some more typical themings, especially over assignments that focus on solving problems from mathematics with computing.

I also recognize the andragogic principle that adults desire autonomy in

their education, that one size does not fit all. To provide students with a more personalized learning experience, I offer multiple options for large programming assignments. This allows students to choose the path that best suits their interests and career goals while still practicing the same computational thinking skills. For example, when giving students a formative assessment covering arrays, I provide them with three possible tasks to choose from: students can write a program which simulates heat diffusion across a wire (engineering theme), a virtual board game where the board is represented with an array (game theme), or a program which keeps a record of transaction amounts (business theme). While the actual requirements of these formative assessments differ slightly, they still require students to demonstrate their knowledge of arrays. By providing these options, my students have more ownership over their learning process, which I've seen results in more of my students getting genuinely invested and excited about the problems they are solving.

In my research, I've seen students stumble through the complex language of CS, struggling with both new vocabulary and familiar words with CS-specific contexts and connotations. To ensure that students feel comfortable with the discipline-specific language of computer science, I introduce new terms in a way that builds on their existing conceptualization of computing concepts. Over the course of the term, I build up a visual ontology of CS-specific terms to help students understand how new concepts relate to familiar ones, gradually expanding their vocabulary and conceptual schema. I believe that this approach minimizes the extraneous cognitive load that can come with learning new terminology and frees up working memory for students to process complex ideas. In my time as a TA, I saw many students confused when presented with words like function, method, and procedure. When I drew a visual which explicitly showed the differences and similarities between these words to my students, much of their confusion was cleared.

In my in-person classes, I prioritize active learning strategies that encourage students to frequently engage with the material in meaningful ways, rather than just sitting through a lecture. While research supports segments of less than 15 minutes, I "chunk" lecture segments into a larger number of roughly three-to-five-minute sections, followed by short individual or group activities that allow students to apply and further explore what they've just learned. For online courses, I limit these lecture sections to three-minutes or less, as my students are more accustomed to consuming short-form media than longer forms of media. This allows for them to quickly absorb the in-

formation they need to get started on a task while also being able to easily find and re-watch videos without needing to search through a longer video. An example of one such assignment can be found in Appendix A. Additionally, I engage in back-and-forth, conversational dialogue with my students during the in-person lecture segments so that my students stay actively engaged throughout. This approach creates a more dynamic and interactive classroom environment, where students are asked to stay actively engaged with the material and with one another. While many of my students are initially resistant to this deviation from normal lecture sessions, they grow accustomed to the more active sessions, engaging with their peers without any hair-pulling on my part.

Lastly, I understand the importance of making mistakes in formative assessments in the learning process. To ensure that my students are continually learning and growing, I apply mastery-based grading by providing them with opportunities to revise and resubmit their work. By allowing them to receive feedback and improve upon their work, I facilitate an environment that encourages a growth mindset which supports their growing intrinsic motivation for learning. This approach also helps to reduce the emphasis on grades and encourages students to focus on their progress and learning journey. As with my short-form lecture style, students are initially resistant to this deviation from the standard grading scheme that they have learned how work around. However, by the end, my students report that this grading philosophy lowered their overall stress around their grade and incentivised them write better programs than they otherwise would have.

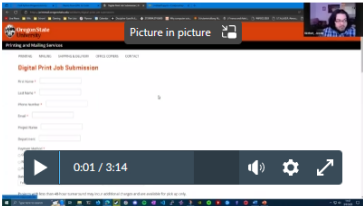
Ultimately, my teaching philosophy is centered on creating an interactive and personalized learning environment where my students can thrive in spite of mistakes. I believe that by embracing an andragogical approach, paying careful attention to the language I frame concepts with, and by using active learning strategies, I can help students develop the computational thinking skills and computing knowledge they need to succeed in our discipline.

2 Teaching Techniques

As mentioned briefly in my teaching philosophy statement above, I embrace a novel technique when it comes to developing lesson activities. Recent studies have shown that uninterrupted lecture for more than 15 minutes begins to severely lose its effectiveness, regardless of the class level. In online learning, the idea of “chunking” content into short videos has become increasingly popular, but I think that given students’ familiarity with short-form social media, they are much more familiar with consuming content in 1 to 3 minute, bite-sized chunks. Thus, I do my best to decompose lessons down into multiple short-form videos, tied together with programming tasks. An example of this is in my “First Python Program” learning activity (shown below) where the first video introduces students to programming with a simple “print” statement. Students are then asked to write a similar program based on the example in the video. A major component to active learning is reflection, so students are asked to reflect on anything that went wrong or on how they could modify their program. The full “First Python Program” activity can be found in Appendix A.

Question 1

1 pts



[Backup link to video](#)

It's time to write your first program! Almost everyone who has ever written a program started with the same, simple program, often called a *Hello World* program. You're going to be taking part in this tradition! A *Hello World* program is very simple. When run, the computer should "print" or display "Hello, World!" to the user. In your notebook go ahead and do this now using the `print()` function. If you get stuck, try starting from where I ended in the video, replacing "text" with "Hello, World!". When your short program is displaying the correct text, please respond to the prompts below.

1. Paste your code here.
2. Did your code work on the first try?
 - If so, try modifying it to say hello to yourself!
 - If not, what do you think went wrong?

Edit View Insert Format Tools Table

12pt Paragraph B I U A [color picker] [background color picker] T² [link icon] [image icon] [video icon] [audio icon] [code icon] [bulleted list icon] [numbered list icon] [undo icon] [redo icon]

Additionally, my learning activities are meant to be very low-stakes. They are graded based only on completion, and I focus more on students' reflections when I offer feedback here, as I understand that they are quite new to programming. In fact, the entire "First Python Program" lesson/activity is meant to be a quick and easy way for novice coders to write their first programs and have a really positive, relatively stress-free introduction to coding. Each of the topics briefly touched on in this lesson are the focus of the coming weeks of this class.

Finally, my mini-lecture videos are not heavily edited or re-shot (save for major mistakes), as I think that it is important for my students to see their instructors as people and not just as an authority figure. When my students see that people in their field, whether that be academics or industry professionals, are like them and can make similar mistakes as them, it serves a dual purpose: students are more easily able to visualize themselves being successful in their field, and it helps build rapport between myself and my students.

3 Formative Assessment Practices

Similarly to my lessons, I also want formative assessments to be low-stakes opportunities for my students to continue forming their own understanding of new concepts and the application thereof without fear of mistakes. To establish this low-stakes atmosphere, I always accompany all of my assignments with a short, informal video explaining the task and requirements. Additionally, I include a short optional reference reading that students may use to brush up on any concepts they are confused about when progressing through the assignment. An example of how this looks for my “From UML to Code” formative assessment is below. The full activity can be found in Appendix B.

⋮ Learning Materials	✓	⋮
⋮ Lectures	✓	⋮
⋮ Replit: From UML to Code - Activity Overview (Watch before starting activity)	✓	⋮
⋮ Readings	✓	⋮
⋮ Basics of a UML Class Diagram	✓	⋮
⋮ Code Examples	✓	⋮
⋮ Activities	✓	⋮
⋮ Replit: From UML to Code 20 pts	✓	⋮

To encourage collaboration, I do facilitate both pair-programming assignments and peer-review. Since formative assessments are based on mastery and can be revised for full credit, there is less pressure among the students in a pair when there is an experience-level difference. Additionally, peer-review is extra valuable, as students can apply this feedback on their re-submission.

On the topic of re-submission and mastery, re-submission comes with some terms attached. In order to resubmit, students must first have submitted something by the due date (however small). Students must also complete and submit a one-page narrative write-up alongside their revised work, discussing where their previous submission fell short of mastery, the steps they took to improve their new submission, and how they believe this submission new submission demonstrates their mastery of the associate learning outcome.

A First Python Program Activity

First Python Program

Purpose

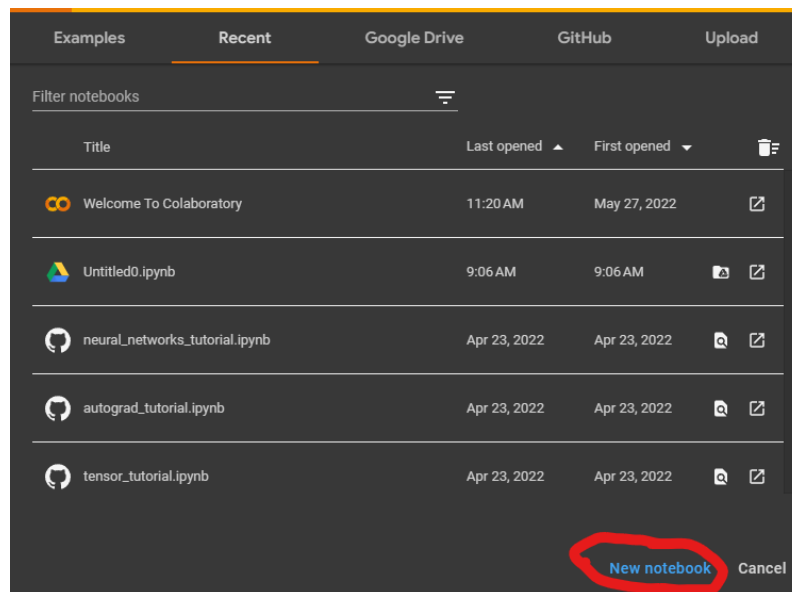
The goal of this activity is to write your first Python programs, demonstrating a basic understanding of data types, arithmetic operations, and conditional logic.

Course Learning Outcomes

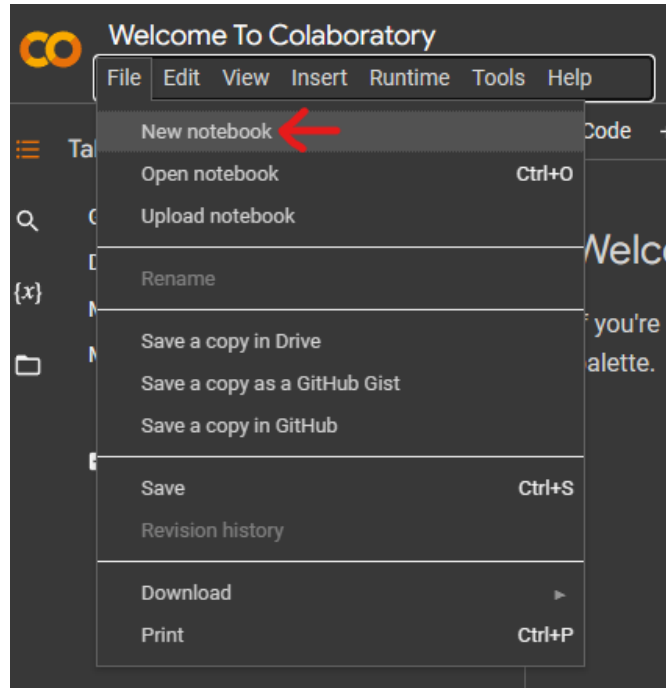
This assignment supports progress towards CLO1: After successfully completing this course, you will be able to *design, code, and test small Python programs*, and CLO2: After successfully completing this course, you will be able to *use top-down design to develop solutions to real-world problems*.

Getting Started

1. Google Colab is an online notebook service that allows us to run Python code alongside our notes. Begin by navigating to [Google Colab](#) and opening a new notebook.
 - a. This can be done by selecting the "New notebook" button if the following window is shown.



- b. If the window is *not* shown, select "File" in the top left corner of the page. Then select "New notebook".



2. Keep your notebook open and work your way through each activity!

Part 1

Please start by watching [this video](#).

It's time to write your first program! Almost everyone who has ever written a program started with the same, simple program, often called a *Hello World* program. You're going to be taking part in this tradition! A *Hello World* program is very simple. When run, the computer should "print" or display "Hello, World!" to the user. In your notebook go ahead and do this now using the `print()` function. If you get stuck, try starting from where I ended in the video, replacing "text" with "Hello, World!". When your short program is displaying the correct text, please respond to the prompts below.

1. Paste your code here.
2. Did your code work on the first try?

- If so, try modifying it to say hello to yourself!
- If not, what do you think went wrong?

Part 2

Please start by watching [this video](#).

Let's do some math! Think of a simple equation from your field of study and write some code to have the computer evaluate it for you. For example, to convert a temperature in Celsius, say 100 degrees, to Fahrenheit, I would write `print((100 * 1.8) + 32)`. If you cannot think of an equation, feel free to use the formula for converting from Fahrenheit to Celsius. Check your answer by hand, with a calculator, or with an online solve. When your program is printing the correct answer, please respond to the prompts below.

1. Paste your code here.
2. Did your code work on the first try?
 - If so, try modifying the calculation to use a really large input, one that would be difficult to calculate by hand. In my case, I might change my input from 100 degrees to 92,921,352 degrees.
 - If not, what do you think went wrong?
3. Based only on what you learned from today's activities, do you think that what you've learned is useful? Or are there still easier ways to accomplish the same calculations?
4. What kind of features might we need in order for our program to become more useful?

Part 3

Please start by watching [this video](#).

It's time to make our calculation more useful! Like we did in the video, ask the user for any input(s) that your calculation needs and save it to a new variable. If your formula has multiple inputs, you'll need to make a new variable for each and ask the user for each. Make sure to update the `print()` statement to use your new variable(s). Test out your program with a few different inputs. Once it is behaving as you expect, [watch the next video](#).

To wrap things up, let's have the computer make a decision based on the result of your calculation. Create an if-statement that checks whether or not the result of your calculation is above or below some number. If it is, print something out. If it is not, print something different out. For example, in the video we told the user to wear a coat if the temperature was below 60 degrees F. If not, we told them not to wear a coat. When you are satisfied that your code is working as you intend, respond to the prompts below.

1. Paste your code here. Don't worry if the formatting gets messed up; starting next week we'll set up accounts with a service that allows us to share code more easily.
2. Did your code work on the first try?
 - If so, try adding a second if-statement and test some inputs out. Does the program work as intended? If not, why do you think that is?
 - If not, what do you think went wrong?
3. Now that you've completed today's activities, has your mind changed about the usefulness of what you've learned? Why or why not?
4. Can you think of any features that you would need to be able to make more useful programs?

B From UML to Code Formative Assessment

Replit: From UML to Code

Purpose

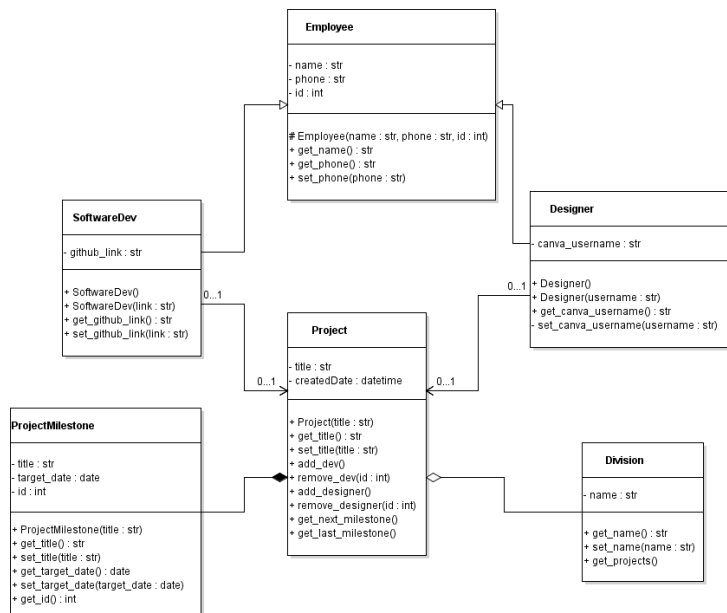
The goal of this activity is to write Python code based on the specifications represented in a UML Class Diagram.

Course Learning Outcome

This assignment supports progress towards CLO2: After successfully completing this course, you will be able to implement in code the requirements specified by UML Class Diagrams.

Instructions

1. Begin by signing into [Replit](#) with the same account you've been using this term.
2. Navigate to the "From UML to Code" project.
3. Select a class from the UML diagram to start with and implement it. Remember to start with classes that don't depend on other classes. If you do not do so, you may have difficulty testing your code incrementally!
4. Continue implementing the other classes.
5. Instructor Unit Tests will be released on Wednesday, so make sure your code passes all of them before you submit. You will be graded based on the percentage of unit tests that pass.



C Curriculum Vitae



Curriculum Vitae

Personal

Name Jason Weber
Address Irvine, CA
Phone number (503) 922-9180
Email jasonweber99@gmail.com

Education and Qualifications

- Aug 2023 - Present • **PhD in Computer Science**
University of California, Irvine, Irvine
Advisors: Jennifer Wong-Ma and Sergio Gago-Masague
Research Focus: Artificial Intelligence in CS Education
- Sep 2021 - Jun 2023 • **MS in Computer Science**
Oregon State University, Corvallis
Advisor: Jennifer Parham-Mocello
Research Group: Computer Science Education
Thesis: Staying Consistent: Discipline-Specific Language Used in Introductory CS Curricula
- Sep 2022 - Jun 2023 • **Graduate Certificate in College and Undergraduate Teaching**
Oregon State University, Corvallis
Capstone Internship: Created and administered multiple active learning assignments in ENGR 103, an introductory programming class required for all engineering majors at OSU. Implementation justified by referencing contemporary learning theories, like Andragogy and Cognitivism.
- Sep 2018 - Jun 2021 • **BS in Computer Science/Mathematics**
Western Oregon University, Monmouth
summa cum laude

Publications

- Weber, J. L., Parham-Mocello, J. (2023 October). Staying Consistent: Discipline-Specific Language Used in a Well-known AP CS A Curriculum. *Frontiers in Education (FIE 2023)*. (primary contributor, wrote paper, analyzed data)
- Parham-Mocello, J., Erwig, M., Niess, M., Weber, J., Barrett, G., and Smith, M. (2023 March). Putting Computing on the Table: Using Physical Games to Teach Computer Science. *Special Interest Group in Computer Science Education (SIGCSE 2023)*. (~30% acceptance rate) (co-developed coding guide, co-analyzed data, co-wrote the results section of paper)
- Parham-Mocello, J., Erwig, M., Niess, M., Nelson, A., Weber, J., and Barrett, G. (2022 October). Using a Text-Based, Functional Board Game Language to Teach Middle School Programming. *Frontiers in Education (FIE 2022)*. (co-developed coding guide, co-analyzed data, co-wrote the results section of paper)

Work/Leadership Experience

- Sep 2023 - Present • **Teaching Assistant**
University of California, Irvine, Irvine, CA
- Jan 2023 - Jun 2023 • **Graduate Teaching Assistant**
Oregon State University, Corvallis
 - Assistant to various instructors in ENGR 102 (a required Excel class for 1st year engineering students) and ENGR 103 (a required CS1 class for 1st year engineering students).

	<ul style="list-style-type: none"> • Led two studio sections of ENGR 102, lecturing for 15 minutes and facilitating effective group work for the remaining hour and a half. • Led all four studio sections of ENGR 103, frequently designing new studio assignments. • Collaborated with instructor and the rest of instructing team to revise studio slides and assignments to ensure student tasks were clear and informative.
Dec 2021 - Dec 2022	<ul style="list-style-type: none"> • Graduate Research Assistant <i>Oregon State University, Corvallis</i> <ul style="list-style-type: none"> • Assistant to Dr Jennifer Parham-Mocello • Led a small team of undergraduate researchers developing and revising curriculum for active 6th and 7th grade computer science classes taught at a local middle school under the Childsplay approach, funded by an NSF CS for All grant. • Taught and demonstrated effective data analysis quantitative and qualitative techniques and Agile project management to undergraduates in order to help them develop professional skills. • Interviewed prospective undergraduate researchers to ensure that they were qualified and understood the type of work they would be doing. • Wrote an (accepted) REU supplement proposal to hire an undergraduate researcher to join our research team for a summer.
Sep 2021 - Dec 2021	<ul style="list-style-type: none"> • Graduate Teaching Assistant <i>Oregon State University, Corvallis</i> <ul style="list-style-type: none"> • Assistant to Dr Eduardo Cotilla-Sanchez's new ENGR 100 course. • Led two studio sections, lecturing for 15 minutes and facilitating effective group work for the remaining hour and a half. • Collaborated with Dr Eduardo Cotilla-Sanchez and the rest of his instructing team to revise studio slides and assignments to ensure studio tasks were clear and informative.
Sep 2020 - Jun 2021	<ul style="list-style-type: none"> • Club President <i>Western Oregon University Mathematics Club, Monmouth</i> <ul style="list-style-type: none"> • Held weekly club meetings over Zoom to discuss upcoming club events and to provide peers with a respite from the stresses of college and the Covid-19 pandemic. • Organized the creation, purchase, and distribution of club merchandise to club members and interested faculty. • Assisted in the planning and execution of the annual Estimathon event, including reaching out to internal university sponsors to fund a prize pool for the winning teams.
Sep 2019 - Jun 2021	<ul style="list-style-type: none"> • Mathematics Grader <i>Western Oregon University, Monmouth</i> <ul style="list-style-type: none"> • Worked with a variety of professors and instructors to grade homework assignments. • Endeavored to understand student's mistakes and leave detailed feedback so that they could easily identify their errors. • Identified homework problems that were repeatedly incorrect and reported them to professors to help them tailor future lectures and assignments to address deficiencies in student progress.
Sep 2018 - Jun 2021	<ul style="list-style-type: none"> • Computer Science and Mathematics Tutor <i>Self Employed, Monmouth</i> <ul style="list-style-type: none"> • Tutored peers at and below my year in computer science and mathematics. • Created sample problems and demonstrated solutions in order to give students a step-by-step guide without giving them homework answers. • Reviewed old homework assignments to assist student in identifying and understanding their deficiencies in knowledge of course content.