

CHAPTER 6

Neurodiverse Students Learning How to Program Outside of the Classroom

6.1 Introduction

There are nearly 7 million students with disabilities—33% of whom have learning disabilities [167]. Students with cognitive, learning, and/or neurological disabilities can excel at computer programming with the appropriate scaffolding and support, but several things may dissuade them from learning how to program [198, 207, 354, 411]. In particular, challenges to providing computing education to students with disabilities include “teachers’ attitudes/expectations, pedagogical approaches, and accommodations/accessible materials, including technology” [207, p. 1]. In this study, I focus on the challenge of providing accommodations/accessible materials and learning technologies to students with cognitive, learning, and/or neurological disabilities hereby referred to as neurodiverse students.

Neurodiversity refers to individual variation in cognitive function, behavioral traits, and affect; it is an umbrella term considered a ‘moving target’ that is useful for how it helps us imagine the world from something other than a neurotypical perspective—thereby decentering cognitive norms about, for example, rates of learning [54, 65, 226, 322, 350]. Most research on learning design has focused on neurotypical individuals [226] and there is relatively little research in computing education on learners with cognitive disabilities [76, 198].

One framework used to teaching neurodiverse learners in computing courses and other STEM courses is Universal Design for Learning (UDL) [319, 330]. The goal is to improve equity and access for all learners [5]. Its principles inherently support the dispersion and neurodiversity that characterize cognition and learning [12], but most research has focused on K-12 learners [302]. Hence, in this study I explored how we can improve an eBooks design through UDL principles for neurodiverse learners in postsecondary education.

With the growth in online learning and abrupt movement to these environments as a way to adapt to global events like the current pandemic [85, 381], it is also increasingly important to an-

swer the call for research into learning how to program *outside* of the classroom [21]. Research about the quality of online computing education resources is scarce [21]. These resources support both self-directed learning (SDL) and self-regulated learning (SRL). Self-directed learning (SDL) is defined “a process in which individuals take the initiative without the help of others in diagnosing their learning needs, formulating goals, and identifying human and material resources” [193, p. 357]. By comparison, self-regulated learning (SRL) “is an active, constructive process whereby learners set goals for their learning and attempt to monitor, regulate and control their cognition, motivation, and behaviour, guided and constrained by their goals and contextual features in the environment” [325, p. 191]. In this study, I explore research questions about the needs of neurodiverse learners and the effectiveness of an interactive eBook. The goal is to design better adaptive and adaptable computing education materials for learners with and without disabilities who want to learn how to program outside of the classroom. Adaptability refers to the ability of the learner to control the system to, for example, (1) switch between problem type (between write-code problems and Parsons problems) and (2) change the difficulty of problems based on adaptive help-seeking strategies (i.e., interaction with help features to make the problem easier) [414]. And adaptivity refers to, for example, the system changing the difficulty of problems based on different measures (e.g. prior performance, cognitive load, self-efficacy, etc.) [414].

The interactive eBook that I use in this study contains Parsons problems which are considered a type of drag-and-drop/block-based programming problem; pieces of code must be ordered and indented correctly [89]. These problems comprise an adaptive learning system that dynamically adapts the learning experience to an individual’s ability via intra-problem and inter-problem adaptation (also known as adaptability and adaptivity respectively). Prior research has evidenced these kinds of adaptive programming practice problems can improve problem-solving efficiency, lower cognitive load, and that most undergraduate novice programmers find them useful for learning as part of a course (i.e., *inside* the classroom) [98, 103, 153]. But are these drag-and-drop/block-based computer programming practice problems accessible to novice programmers with disabilities?

Several researchers have posed this question [247, 197, 153], but most research has focused on novice programmers with visual or motor impairments [247, 248]. And, a recent study showed research in this area mainly focuses on K-12 learners with visual impairments and block-based environments that use the Scratch visual programming language [257]. To my knowledge, no one has studied the accessibility of adaptive Parsons problems for neurodiverse learners. In this study, I address the call to extend this research to how learners with cognitive disabilities interact with hybrid environments that provide both block-based (Parsons) and text-based (write-code) programming problems [197].

Investigations into the usability of adaptive Parsons problems (i.e., learners satisfaction with them and understanding of their adaptability and adaptivity) has provided lots of insight, but these

studies were based on neurotypical individuals. Ericson et al. [98] interviewed secondary teachers to understand how they felt about solving Parsons problems and found teachers perceived these kinds of problems help them learn how to fix and write similar code and that they understood most but not all of the intra-problem adaptation (they did not understand implicit hints about indentation). Also, we asked undergraduate students open-ended questions about Parsons problems versus write-code problems and results showed all of the students perceived solving Parsons problems as easier than writing the equivalent code (see Chapter 4). Some students also found them helpful for learning yet, as long as students did not get stuck, they claimed to learn more when writing code from scratch. And some students with prior programming experience had strong negative reactions to Parsons problems [153]. As a result, we created a toggle feature so that students could either preview or switch to a Parsons problems while solving a write-code problem. Students also reported they found the adaptation confusing when it provided indentation or combined blocks that were already adjacent, but that they found the adaptation helpful when it removed distractors or combined blocks that were not already together [153]. Thus we, changed the adaptation process to no longer provide indentation and to combine blocks that are the furthest apart. These results show how Parsons problems, which can be considered either formative or summative computer programming assessments [79, 102], influence teachers and students' perceptions about learning how to program and how we design for adaptation, but they've mostly focused on neurotypical learners.

Furthermore, these assessments have the potential to lower cognitive load [312], but researchers have obtained mixed results based on the type of self-report/self-assessment instrument used to measure cognitive load. Computing education researchers (CERs) using the Paas scale have shown this instrument can be useful for understanding the mental effort imposed on novice programmers presented with Parsons problems that have distractor blocks [145, 146] and have observed significantly lower intrinsic cognitive load for solving Parsons problems with feedback and without distractors [24]. Prior research has also shown (1) significant results when using the Paas scale [153] and (2) no significant results when using the Computer Science Cognitive Load Component Survey (CS CLCS) [102] to study the impact on cognitive load of solving Parsons problems versus isomorphic fix-code and/or write-code problems. The Paas scale (considered a single measure) has proven reliable and the (CS CLCS) has showed mixed results, but researchers argue it is important to experiment with and develop differentiated measures such as the latter [191].

Moreover, educational researchers posit learners' attitudes about assessments influence educational outcomes [44, 42]. Hence, it is important to explore learners' perceptions of computer programming instruction and assessment materials to understand how they influence students' attitudes and thus learning outcomes.

It is also crucial for programmers to develop cognitive and metacognitive (self-assessment)

skills during problem-solving by asking learners to self-report on their attitudes about computing, cognitive load, self-efficacy, and sources of struggle and interference [109, 110, 223, 327, 386], yet work on how learners self-assess phenomena such as cognitive load or self-efficacy is limited [219, 224, 294]. Loksa et al. argue that “consistent, disciplined self-regulation during problem-solving” is necessary [223, p. 90]. Reducing cognitive load is an important endeavor for instructional designers in computing education [369], and students with positive attitudes and high self-efficacy about computing are expected to persist [300, 401]. Self-assessments have a strong influence on self-efficacy [130], and a recent review of surveys and/or questionnaires used in computer science education (CSEd) argue for further investigation into their validity [423]. In this study, I explore the computational practices, perspectives, and attitudes of neurodiverse learners and their self-assessment process through retrospective think-aloud sessions and four self-report/self-assessment instruments: an open-ended question about any accessibility barriers/challenges and benefits experienced while using the interactive eBook, a computer science attitude survey [36], the (CS CLCS) [253], and the self-efficacy portion of the Motivated Strategies for Learning Questionnaire (MSLQ) [289]. I chose to follow up with questions about the self-report measures via think-aloud sessions because research methodologists argue for using interviews to evaluate survey items [413]. My research questions were:

- RQ1:** What are the accessibility barriers/challenges and benefits reported when neurodiverse learners use an interactive eBook to learn how to program?
- RQ2:** What are their computational practices, perspectives, and attitudes?
- RQ3:** What are their perceptions of the usability and usefulness of adaptive Parsons problems versus write-code problems for learning how to program?
- RQ4:** What are their perceptions about the surveys and/or questionnaires used in the study?

This work contributes to the literature on design principles for neurodiverse and neurotypical learners related to (1) computing education interactive eBooks, (2) computer programming assessments such as Parsons and write-code problems, and (3) computer science surveys and/or questionnaires used to understand novice programmers attitudes, cognitive load, and self-efficacy.

6.2 Methods

To answer the research questions, I conducted an exploratory multiple case study following Yin’s case study guidelines [421]. Case study methodology was originally intended for exploratory purposes [115]. Exploratory case studies are designed to discover what’s happening, search for new insights, and generate ideas and working hypotheses for future research [321, 421]. In particular, a

multiple-case study enables researchers to explore differences within and between cases. Furthermore, most investigations into how learners with cognitive disabilities learn to program are case studies [76].

6.2.1 Participants

I recruited novice programmers from a post-secondary research institution in the northern Midwest in the United States via a flyer sent to several listservs for students with disabilities (see Figure 6.1). Participants were eligible if they had a cognitive, learning, and/or neurological disability as categorized by [W3C's Web Accessibility Initiative \(WAI\)](#) and defined by the Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition (DSM-V). They were asked to fill out a sign-up questionnaire used to screen participants which included a prior programming experience survey (see Appendix D) [163, 345] and then a disability questionnaire if they were eligible (see Figure 6.2). Each participant picked their own pseudonym and received a \$500.00 stipend for completing the study. Ten people volunteered for this study. Five completed it. I only report on four of the participants; this was what my committee and I agreed to when the study was proposed. These four participants had completed the study by the time I started analyzing the data. Demographic information about each participation is shown in Table 6.1.

Of the ten participants, four never started the study. One participant was asked why they dropped out of the study; they had completed at least one set of the reading and programming practice problems in the eBook. Atype was a 25 year-old female of mixed ethnicity (American Indian or Alaska Native, African American, and White). She was a dual masters student in studying general epidemiology (in public health) and integrated health (in social work) who openly identified as having an autism spectrum disorder (ASD), intellectual disability, learning disability, mental health disability, and memory impairment. Atype reported that requirements for other classes, illness, and social/personal life issues significantly interfered with her ability to finish the study.

6.2.1.1 Amanda

Amanda was a 28 year-old Caucasian who identifies as female. She was a doctoral candidate in psychology with a focus on personality & social contexts who openly identified as having a disability and/or chronic condition. Her research was on teacher well-being and resilience; she was working on her dissertation part-time. Amanda did not have an Individualized Education Program (IEP) or 504 plan in high school. She was currently registered with the university's Office of Services for Students with Disabilities (SSD) for a few accommodations: assistance with editing, a medical alert service dog, and extended time on assignments. Amanda described herself

Learn how to code!

For beginners.



Do you like solving puzzles or learning foreign languages? Do you enjoy working with computers? If so, you might like to learn Python, a beginner-friendly programming language.



Who: We are looking to recruit **students with cognitive, learning, and/or neurological disabilities** with **no prior programming experience in Python**.



What: This is a **research study**. You will learn the basics of Python using an eBook. Participants will earn a stipend of **\$500**.



Where: Work **online** remotely and asynchronously through the eBook. Participate in four remote interviews **via Zoom**.



When: Rolling start dates.

For more information, contact Carl Haynes-Magyar and Dr. Barbara Ericson at ericsonlabs@umich.edu



Figure 6.1: Recruitment Flyer

Q2. How would you describe your cognitive, learning, and/or neurological disability? (You can select more than one.)

- Attention deficit hyperactivity disorder (ADHD)** (formerly "attention deficit disorder (ADD)") – involves difficulty focusing on a single task, focusing for longer periods, or being easily distracted.
- Autism spectrum disorder (ASD)** (includes "autism," "Asperger syndrome," and "pervasive developmental disorder" (PDD)) – involves impairments of social communication and interaction abilities, and sometimes restricted habits and interests.
- Intellectual disability** (sometimes called "learning disabilities" in Europe and some other countries, and "developmental disabilities" in other regions) – involves impairments of intelligence, learning more slowly, or difficulty understanding complex concepts. Down syndrome is one among many different causes of intellectual disabilities.
- Learning disability** – is a functional term rather than a medical condition, and is not uniformly defined. In Europe and some other countries, it refers to intellectual disabilities, while in Australia, Canada, the U.S., and some other countries it refers to perceptual disabilities.
- Mental health disability** – includes anxiety, delirium, depression, paranoia, schizophrenia, and many other disorders. These conditions may cause difficulty focusing on information, processing information, or understanding it. In particular medication for these disorders may have side effects including blurred vision, hand tremors, and other impairments.
- Memory impairment** – involves limited short-term memory, missing long-term memory, or limited ability to recall language. Dementia is one among many different causes of memory impairments.
- Multiple sclerosis** – causes damage to nerve cells in the brain and spinal cord, and can affect auditory, cognitive, physical, or visual abilities, in particular during relapses.
- Neurodiverse** – is a societal rather than medical term to describe the natural diversity in neurocognitive functioning, like gender, ethnicity, sexual orientation, and disability.
- Perceptual disability** (sometimes called "learning disabilities" in Australia, Canada, the U.S., and some other countries) – involves difficulty processing auditory, tactile, visual, or other sensory information. This can impact reading (dyslexia), writing (dysgraphia), processing numbers (dyscalculia), or spatial and temporal orientation.
- Seizure disorder** – includes different types of epilepsy and migraines, which may be in reaction to visual flickering or audio signals at certain frequencies or patterns.

Figure 6.2: Disability Questionnaire

Table 6.1: Participant Demographics

Pseudonym	Age	Gender	Ethnicity	Case	Programming Experience
Amanda	28	Female	Caucasian	Mental Health Disability; Neurodiverse; Seizure Disorder	SPSS, R
Claire	21	Female	Caucasian	Attention Deficit Hyperactivity Disorder (ADHD); Neurodiverse	SPSS, R
User	38	Agender	Russian-Yakut	ADHD; Mental Health Disability	None
Sophia	26	Female, Pansexual	Latina	Learning Disability; Mental Health Disability; Memory Impairment	Mplus, SPSS, R

as having a seizure disorder, a mental health disability, and being neurodiverse. She was diagnosed with epilepsy two years ago. In particular, she experiences focal seizures which can “affect specific cognitive functions including receptive and expressive language, memory and mood regulation” [303, p. 145]. Her seizures came from her right temporal lobe (common for focal seizures) where she used to have a tumor before surgery. The main features of these seizures include a “variety of feelings, emotions, and thoughts” [303, p. 145]. Amanda said, “When I first started having seizures, I didn’t know they were seizures because I wasn’t having convulsions. And when I tried to explain it to my doctors. I kept saying it’s like an out-of-body experience. It’s a really weird feeling, and sometimes it almost feels like a dream....that’s how I described it. It’s kind of like an out-of-body experience that feels weird for about 30 seconds to two minutes and then you come back.”

When asked how her disability impacted how she read and learned from print and/or digital books, Amanda said, “With digital, I need to be careful because of seizure triggers and migraines”. She preferred print books. She said, “Print has less of a chance of giving me a migraine or a seizure.”

When asked why she wanted to learn how to program, Amanda said, “What really appealed to me about the study was the disability portion...I try to participate in research as much as I can...that appealed to me more than the computer part.” She went on to mention that she had prior programming experience with R and SPSS, but initially responded that she had no prior programming experience when filling out the sign-up questionnaire. This was permissible because

she did not have any Python experience. Interestingly, a couple of participants did not view their experience with languages used in statistics courses as prior programming experience.

When asked how she thought her disability would affect her during the research study, Amanda said, “The only thing I can think of is that if it’s a lot of reading, I would need to space it out. If I get the document and I see that there’s 100 pages and I want to get it done that day, I would do one section in the morning, one section in the afternoon, and one section at night.” She said it would be helpful to have estimated reading times for each of the chapters and practice in the eBook. As a result, I used a Chrome extension called [Read Time Estimator](#) to estimate all of the reading assignments in the eBook and sent them to her (see Table 6.2). Future tables of contents should explore adding reading time estimates to the eBook and account for its interactivity; these times may not be accurate for all readers.

When asked how well she’d done in the past in math, science, etc., Amanda said she “would classify [herself] as more of a math and science person versus a humanities person based on feedback through schooling.” What motivated her to learn in general has changed over the years. She said, “When I was an undergrad, it was definitely outcomes like getting a degree. Now, especially in grad school, as I know what career I’m going into—a teaching focused career—what motivates me to learn is becoming a better teacher, becoming a better instructor. I love when I can learn something and then use it in my practice. That’s really motivating. I’m teaching at Eastern Michigan University right now and I have a lot of students who major in computer programming and gaming. I always try to learn what students’ interest are. I have a lot of students who are very interested in coding and computer science.”

6.2.1.2 Claire

Claire was a 21 year-old Caucasian who identified as female. She was undergraduate senior in environmental studies planning to attend law school. She openly identified as having a disability and/or chronic condition. Claire did not have an IEP or 504 plan in high school, but was currently registered with the university’s SSD office for a two accommodations: time-and-a-half and private locations for exams. She describe herself as having an attention-deficit/hyperactivity disorder (ADHD) and being neurodiverse. When asked how her disability impacted how she read and learned from print and/or digital books, Claire said, “I have a lot of trouble processing what I read. Even if I am physically reading something, I often am not processing the information and am usually thinking about something else entirely. I’m also easily distracted by any sort of outside stimuli, which can include moving advertisements on a web page, movement occurring around me, or even a screen being too bright.” She preferred print books. She said, “I’ve found it helps me to stay focused and also remember information if I can physically see where in the reading I am (what page, what point on a page, etc.). I also like to be able to make notes in the margins of what

I am reading, and I remember the information better if this is done by hand. I'm also less easily distracted with print, because digital books are usually found on the internet, which can cause me to become sidetracked very easily."

When asked why she wanted to learn how to program, Claire said, "I've learned that it's a useful skill employers want to see and just for different projects that come up in the workplace. I have pretty much no experience with it." When asked if she had experience with R or SPSS, Claire said, "Yeah, I worked with [both] just never *coding*. She initially responded that she had no prior programming experience when filling out the online survey.

When showed a portion of Runestone and asked how she thought her disability would affect her during the research study, Claire said, "I think one thing that I struggle a lot with is reading long blocks of information at a time. I just can't really sit and process that much at a time. I can read it, but I don't actually understand what I'm reading. And working on any task that takes [a lot of] brainpower if I'm not taking medication is very frustrating and I just don't do it. I like seeing the interactive questions. I also like noticed that it followed rather short blocks of information so I didn't have to read as much before like switching to a different task which definitely helps me." When asked a followup question about how she deals with distractions, Claire said, "Sometimes I have to get up and walk and just take a break. Sometimes I also have to change how I'm sitting...I'll sit on the floor, and like, start reading...and sometimes that works."

When asked how well she'd done in the past in math, science, etc., Claire said, "I did quite well in math and science courses throughout high school and in college. I really hated math except for statistics. Looking back, I think I would have liked it, but I just didn't want to put in the time to figure things out. And so I did my homework or just copied it and crammed for exams. [I like statistics because it's] more logical and like a puzzle."

What motivated her to learn was learning. She said, "I just like to know things. Not in a weird creepy way, but I just think it's fun to learn about new things...it's not any deeper than that."

6.2.1.3 User

User was a 38 year-old Russian-Yakut who identified as agender. They had just started the graduate program in social work and openly identified as having a disability and/or chronic condition. User attended high school in Russia (where they do not have IEP or 504 plans) and had become disabled in their mid-thirties. They were currently registered with the university's Office of Services for Students with Disabilities for a few accommodations: flexible attendance requirements, flexible deadlines, and regular class breaks.

They describe themselves as having an attention-deficit/hyperactivity disorder (ADHD) and a mental health disability. When asked how their disability impacted how they read and learned from print and/or digital books, User said, "I find it hard to concentrate and have to reread portions of

the text slowly.” They preferred both print and digital books. They said, “It’s actually not a big deal for me and I can go either way, but it’s nice when I can take a break from looking at the screen.”

When asked how they thought their disability would affect them during the research study, User said, “Reading, in general, is kind of difficult because I have problems with concentration and it depends on how tired I am or what kind of mental state I’m in. Sometimes I can go through several paragraphs and that’s fine. And sometimes, I need to reread things several times.”

When asked how well they’d done in the past in math, science, etc., User said, “regarding STEM classes, I only got them in high school and after that I specialized in humanities and arts. I was always an A student. I loved algebra and geometry. My main problem in calculus was that I would misread the equations. I would see a plus instead of a multiplication sign and then do the procedure correctly, but the answer was wrong because I misread the [equation]...that was very typical for me. It was very easy for me to actually follow the rules, but my attention was suffering a little bit....I always loved physics. But after that stage, I went into sort of a vocational college for music as a pianist....there was just music and none of the STEM [courses]. And then I went to the conservatory as a musicologist and there weren’t any STEM [courses] for my bachelor’s. Then I got a masters in musicology and a PhD in musicology. I was very deeply in the humanities. I have a girlfriend who tried to teach me Python, but none of that stuck. But it’s something that I want to do that I think would be enjoying to learn.”

What motivated them to learn was reconnecting with old interests and gaining new skills. They said, “First, I think it would be fun. I did always love these kinds of mathy things. Now, my life is very far from them and I think it would be fun to kind of reconnect with them. And then, in terms of job search, it is always good to have another skill on my resume.”

6.2.1.4 Sophia

Sophia was a 26 year-old Latina who identified as female. She was a doctoral student in her sixth year of psychology and women’s and gender studies. Sophia had four years of prior programming experience in Mplus, SPSS, and R which she gained through three semesters of psychology statistics courses and research. She was the only one who counted these languages as experience via the survey. She openly identified as having a disability and/or chronic condition. Sophia did not have a IEP or 504 plan in high school. She became disabled later on in life due to a traumatic brain injury in June 2021. She was currently registered with the university’s SSD office for accommodations: lectures with closed captions and/or recordings of them for her to re-watch later. She said, “I use otter which records and transcribes words as [a lecture is] happening. Regardless, the ability to see the words helps me process a conversation with another person.”

Sophia describe herself as experiencing memory impairment and having a learning and mental health disability. She said, “I have trouble concentrating, and speaking. When listening to audio

or a video it helps me a lot to see captions. I lose my place constantly when I read, so I have to mark where I am at. When speaking, I often stutter or have a speech delay, the patience of others often helps with this. But it makes it difficult to use dictator in Microsoft Word (speech-to-text). When trying to make a presentation, I now need to write everything down as opposed to relying on memory.”

When asked how her disability impacted how she read and learned from print and/or digital books, Sophia said, “I lose focus a lot and if there are any distractions (too much color, different types of text size, too many pictures) then my eyes wander to other things besides the text. Concentration and processing are much slower for me, so I will be stuck on one word or a sentence because I keep rereading it or can’t comprehend the meaning. My short-term memory is a part of concentration. [Sometimes] I am in the middle of reading a sentence and forget what came before, or the starting part of the sentence. I also can’t rely on speaking while I read because I have a speech delay (I am thinking of the word but my mouth/voice cannot form my thoughts). I used to be able to read from digital books with a lot of practice (I didn’t like it at first because I wasn’t taught that way in school), but my disability has led me to only be able to read print books. Because I can interact better with text on a paper.”

Sophia preferred print books. She said, “Personally, I prefer reading/learning from a printed book/article. It helps me focus because I can touch it and interact with it better than an article online. Many online documents now have the ability to highlight and write but I constantly have to customize it. For example, online I could be writing notes and have to consistently change the font. Or maybe I want to underline some text but the wrong text keeps underlining so I have to delete it and redo it. I often lose focus when reading and end up reading the same text over and over again. I can mark it when it’s on paper or use my finger to help me read. This interaction has been really helpful even before I was disabled but even more so now.”

When asked why she wanted to learn how to program, Sophia said, “I would say that it’s part of the field so it’s kind of required. Most people do quantitative work. Programs are what we use to analyze data, but I haven’t really been the best. It takes me a really long time to learn it. And after I my concussion, it is even worse.”

When asked how she thought her disability would affect her during the research study, Sophia said, “I would probably say it might take me a lot longer.”

When asked how well they’d done in the past in math, science, etc., Sophia said, “so high school, I didn’t go to a well funded school. I didn’t have any type of class like statistics until college. And even then, I only took one college class on statistics and we did SPSS which is less coding and more just point and click. But once I got into grad school, I started doing a lot more different programming. [In high school math and science,] I didn’t excel like some of my classmates. But there were other factors. Most of us had parents that didn’t graduate high school,

so that support I didn't have....I did average in science and math. I struggled a lot."

When asked what motivated her to learn, Sophia said, "I've experienced a lot of microaggressions. A lot of it was that the partner I had, who was male, at the time was seen as smarter than I was and because I didn't excel in those math and science classes as he did that it was like I was less smart, less intelligent, so my reason for learning is to stop that stereotype. I'm Latina and people that I grew up with had a lot of negative stuff to say about us. It's to stop these stereotypes, do better for my people, and then really just secure a decent job so I can help my family."

6.2.2 Materials

The eBook used in this study is an interactive version of Dr. Charles Severance's *Python for Everybody* static eBook [336]. It features typical instructional material (text, pictures, videos) and interactive features with immediate feedback (write-code problems, fix code problems, adaptive mixed-up code (Parsons) problems, multiple-choice questions, fill-in-the-blank questions, and matching questions) [103]. The eBook covers programming fundamentals (strings, variables, loops, conditionals, functions), data structures (lists, tuples, dictionaries), object-oriented programming, and an introduction to data science (Files, Beautiful Soup, APIs, Databases). The chapters contain worked examples with interleaved practice problems. "A worked example is a detailed description or demonstration of how to solve a problem, including both the problem statement and all steps of the problem solution" [102]. Concluding each chapter there are multiple-choice questions, mixed-up code (Parsons) problems, and write-code problems.

I used one open-ended question, three surveys, one questionnaire, and one scale in this study. First, participants answered an open-ended question about accessibility at the end of each reading and practice assignment. It asked the to "Please read over the strategies, standards, and resources for making the Web accessible to people with disabilities by clicking on this [link](#) and then answer the following question. Did you encounter any accessibility barriers or challenges while using the interactive eBook to today? If so, please explain?"

Second, I used a computer science attitude survey to measure how participants attitudes toward computing changed after each reading and practice assignment (see Appendix G) [36]. The survey has 40 items that comprise four factors: problem solving - fixed mindset (Factor 1), gender equity (Factor 2), importance (Factor 3), problem-solving - strategies (Factor 4), gender bias (Factor 5), and personal interest (Factor 6). It asks respondents to rate each statement using a 5-point Likert scale from "*strongly disagree*" to "*strongly agree*". Bockman et al. tested the reliability of each factor and found each had high reliable alpha scores similar to Dorn et al.'s study [88].

Third, I used the Computer Science Cognitive Load Component Survey (CS CLCS) to capture the overall cognitive load experienced by participants after each reading and practice assignment

(see Appendix H); it's been demonstrated to be internally reliable [253, 424]. The survey has 10 items that comprise three factors: intrinsic load (Factor 1), extraneous load (Factor 2), and germane load (Factor 3). It asks respondents to rate each statement using a 10-point Likert scale from “*not at all*” to “*completely the case*”.

Fourth, I used the self-efficacy portion of the Motivated Strategies for Learning Questionnaire (MSLQ) to measure participants beliefs about their ability comprehend and complete the reading and practice assignments (see Appendix I) [289]. It asks five questions rated using a 7-point Likert scale from “not at all true of me” to “very true of me”. It's been validated several times across different populations [70, 166, 174].

Fifth, the Paas scale was administered after some of the practice problems (see Appendix C) [271]. This question uses a 9-point Likert scale that asks respondents to rate how much mental effort they invested in solving the previous problem from “*very, very low mental effort*” to “*very, very high mental effort*”.

And finally, participants were asked to fill out a survey about the relationship between life and learning outside of the classroom (see Appendix J); it's been shown to have good internal consistency [327]. It asks participants to rate the degree to which certain things interfered with their ability to learn and complete the reading and practice assignments on a 5-point Likert scale from “*not at all*” to “*significantly*”.

6.2.3 Protocol

Participants were asked to complete eight sets of reading and programming practice problems (see Table 6.2). They were reminded to read the entire chapter and complete its interactive problems and a practice assignment (some of the end of chapter problems) via email and informed that these were just reminders, not mandatory deadlines. Upon completing each set, they were also asked to complete four self-report/self-assessment instruments each week: an open-ended question about any accessibility barriers/challenges and benefits experienced while using the interactive eBook, a computer science attitude survey [36], the Computer Science Cognitive Load Component Survey (CS CLCS) [253], and the self-efficacy portion of the Motivated Strategies for Learning Questionnaire (MSLQ) [289].

I conducted three think-aloud sessions with each participant online via Zoom after they completed weeks three, six, and eight. I began by asking participants for consent. Then I followed up with them about their responses to the self-report/self-assessment instruments, asked them to solve a set of programming problems while I observed them, and then asked them followup questions about solving the problems. These problems were representative of the topics being studied (see Figure 6.3).

The following program segment should prompt the user for their name and say hello to them. But, the blocks have been mixed up and include extra blocks that aren't correct.

Drag from here

```

3 | print("Hello" name)
5 | name = "yourName"

```

Drop blocks here

```

2 | name = input('What is your name?\n')
4 | greeting = "Hello "
1 | print(greeting + name)

```

Check Reset Help me

Perfect! It took you only one try to solve this. Great job!

Q-1: Which debug command would you use if you want to execute a function on the next line without having to go through that function line by line

- A. Step Into
- B. Step Over
- C. Continue
- D. Step Ignore

Check Me Compare me

✓ Correct.

The following program should print out whether the number, x, is even or odd, but the code is mixed up. For example, if x is 3, the code should print **3 is odd**. Be sure to indent correctly! Not all blocks will be used.

Drag from here

```

6a | elif:
3a | print(x + " is odd")
1a | print(x + " is even")
4b | if x % 2 == 0:

```

Drop blocks here

```

5 | x = 91
4a | if x % 2 == 0:
7a | print(str(x) + " is even")
2a | else:
3b | print(str(x) + " is odd")

```

Check Reset

Perfect! It took you 2 tries to solve this. Click Reset to try to solve it in one attempt.

Create a function `cube(n)` that takes a number, `n`, and cubes it and returns a string in the form `Cube of (n) is (n cubed)`. For example, `cube(4)` returns `Cube of 4 is 64`.

Drag from here

```

1a | return cube_num
1b | return answer
2 | def cube(n):
3a | cube_num = n * 3
3b | cube_num = n ** 3
4 | answer = "Cube of " + str(n) + " is " + str(cube_num)

```

Drop blocks here

Check Reset Help me

Figure 6.3: Example Problems from Chapters One to Four

Table 6.2: Weekly eBook Chapter Assignments

Week	Chapter	# of Pages	Estimated Reading Time (200 words per min.)
Week 1	Variables, Expressions, and Statements	14	40 min.
Week 2	Debugging	5	30 min.
Week 3	Conditional Execution	11	40 min.
Week 4	Functions	13	40 min.
Week 5	Loops and Iterations	8	40 min.
Week 6	Strings	13	40 min.
Week 7	Lists	15	46 min.
Week 8	Dictionaries	7	33 min.

6.2.4 Analysis

Multiple-case studies can be examined through both within-case analysis and cross-case synthesis [421]. Qualitative analysis was performed using ATLAS.ti. Each of the think-aloud observations was transcribed. I developed a codebook using a hybrid approach [326] of deductive (a priori) and inductive (new) codes with a doctoral student [326]. We derived and refined our codes based on previous research the research questions. I trained with one doctoral student to independently code 20% of the transcripts and identify examples until we reached 100% agreement based on [140]. I coded the remaining transcripts single-handedly.

6.3 Multi-Case Study

Here I report on each of the participants who completed the study at the time of writing.

6.3.1 Amanda

Amanda completed the study in 12 weeks and 6 days.

6.3.1.1 Accessibility Barriers/Challenges and Benefits

Each week, participants were asked if they encountered any accessibility barriers/challenges and/or benefits while using the interactive eBook to today? If so, please explain?"

Most weeks, Amanda responded that she did not encounter any accessibility barriers/challenges while using the interactive eBook to read and practice programming. For the week two, on debugging, Amanda wrote, "Today's reading and activities were much more clear than [the reading

and activities for variables, expressions, and statements] and I was able to do them in one sitting whereas I was not able to do so for the [latter].

She did, however, experience some accessibility challenges during the practice portion of the research assignments, Amanda reported:

“I’ve had two seizures....The first three days it didn’t happen. I think [during the chapter on conditional execution] and [functions]. Numbers are sometimes triggering for me...usually when I have a seizure, I’ll put my head down to get blood back to my head. I normally feel kind of faint. I don’t have convulsive seizures. I have focal seizures....It’s not really anything that can be controlled. It’s just the presence of numbers and sometimes if I’m going through it too fast. If it says it’s going to take 40 minutes, it’ll probably take me two hours. But if I don’t have two hours and I’m trying to go through it. If I gave myself two hours, I probably would be able to avoid triggering a seizure, but because I’m trying to go through and finish, it sometimes will trigger [a seizure].”

When I followed up with her weeks later about her seizures, Amanda said, “It’s not even just flashing. Flashing is an obvious seizure trigger...even just quick movements or changes in color while I’m scrolling through and color contrast because of the way I’m scrolling is like a flash. But with this [the research assignments in Runestone], I don’t really ever have to think about that. I haven’t got any triggers. I’ve had number triggers, but then again, I can pause and come back to it. It’s not timed....that’s all helpful in comparison to other things I’ve done this year....If I see a bunch of numbers, I [scroll] slower.”

On the flexibility of learning outside of the classroom, Amanda said, “[The flexible deadlines] work. When I get an email from you, I do it as soon as I can. I don’t have a lot going on right now, so it’s pretty easy to get done. I don’t really pay attention to when it’s due because I know if I do it within the first couple of days of you emailing it, I’ll make the deadline.” During a different think-aloud session, she said, “In terms of accessibility, there have been no barriers for me. I have to pause a lot or come back to something and so having the ability to do that is helpful. Also, that it’s self-paced is helpful because if I’m not feeling good I can stop and I can come back to it.”

When asked if learning outside the classroom was a better fit for her given her disability, she said, “Yes, 100% better.” And when asked to what degree did certain things interfere with her ability to learn and complete the reading and practice assignments, she reported that illness and confusion about the material significantly interfered.

6.3.1.2 Computational Practices, Perspectives, and Attitudes

Amanda’s main computational problem-solving strategy for solving Parsons and write-code problems included the use of trial-and-error. During the first think-aloud session, when solving a Parsons problem, Amanda seemed to arrange the blocks without thought. She had difficulty with indentation and rated investing high mental effort in solving it (see Figure 6.4). When I followed up with her about her computational practices, she said about the practice problems that “if I don’t feel good on the second question, I’ll just go through and do trial-and-error. [Especially if] I feel like I’m just going to get another seizure.” She said, when solving them “I try to think back to the assignment and go in the order that I think makes sense and I almost never get it. I like being able to check it and I like how it will show me [what’s] wrong. It’ll [say] recheck this area. I wish sometimes that when it checked if something was wrong that something would show up that goes back to the reading assignment because I don’t always remember. And even if it says a specific thing, I can get it from trial-and-error. But I feel like I’m not really learning it if I’m just clicking around and trying to get it [right].”

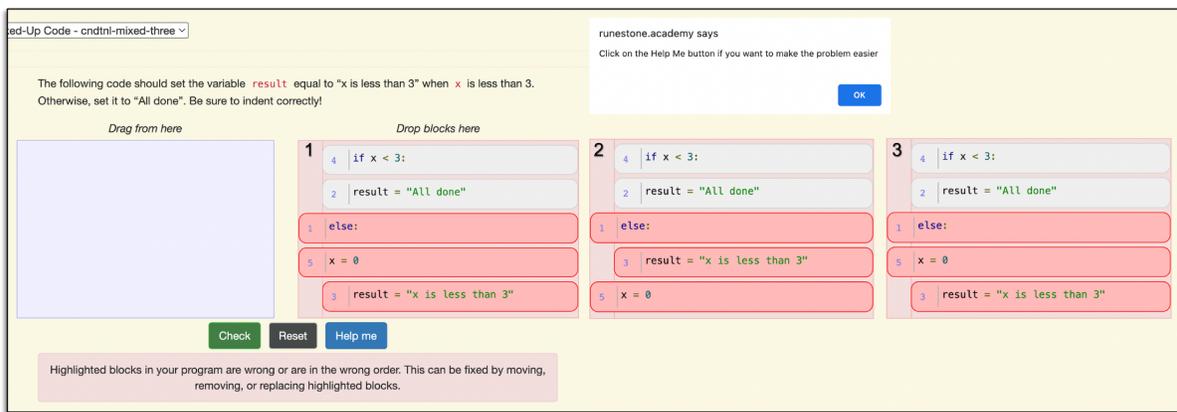


Figure 6.4: Amanda’s trial-and-error Strategy.

Amanda often used her cursor to highlight problem instructions. Upon following up with her about this, she said, “I always do that. It just helps me focus.”

Regarding solving write-code problems, Amanda stated, “I always [rate] those [as investing] the highest mental effort....sometimes these beginning problems help with [solving them]. If I can’t recall the information at all, I go back to these [multiple choice] problems and remind myself of what it is. The Parsons problems sometimes will help because it’s similar. I also wish it would say revisit the text.” During the first think-aloud session, she rated investing ‘very very high mental effort’ in solving one of the write-code problems and said, “I have no idea where to start on problems like this. I give up right away.” She said it would be beneficial if she could use a Chrome extension similar to Grammarly to help with the syntax of write-code problems.

A question about the practice of switching or toggling between Parsons and write-code problems led to Amanda responding that it was helpful but that she originally thought it was something for the researchers to interact with, not her. She expressed that it would increase her self-efficacy. Amanda said, “It would help a lot. Even just looking at a preview [of the Parsons problem] and then going back to write it myself would make me think, ‘I could do this.’ ” She also said she would probably never switch from a Parsons problem to a write-code problem unless “I was doing something that I was going to use for my personal or professional life and I was checking it. But if it was an assignment, the [Parsons problems] requires less effort than writing, so I would never change from this to writing. I would only do the opposite.” Amanda described the toggle question as an accommodation, she said, “Memory is a huge problem for me....and when you showed me how to toggle the questions...that was really helpful as a memory aid.”

During the third think-aloud session, Amanda used the toggle button to solve most write-code problems as a Parsons first and then retyped the Parsons solution into the write-code problem. She also heavily used the “Help Me” button when solving Parsons problems with distractors. When asked about the toggle button she said, “It definitely helped me....I thought once I did it that it would copy it [the code] for me [to the write-code problem space area]. And I was glad it didn’t and that I had to write it out because it forced me to read it...I know I would not have done that if it just [transferred it].” Previewing the Parsons problems solution also helped Amanda solve one of the write-code problems, but she had trouble with indentation and viewing both problem space areas at the same time (see Figure 6.5). When asked if she understood what the “Help Me” button was doing and why, Amanda said, “I’d say 50% of the time [I understood it].” She found the *combine blocks* feature useful but needed more feedback about why it chose to combine certain blocks.

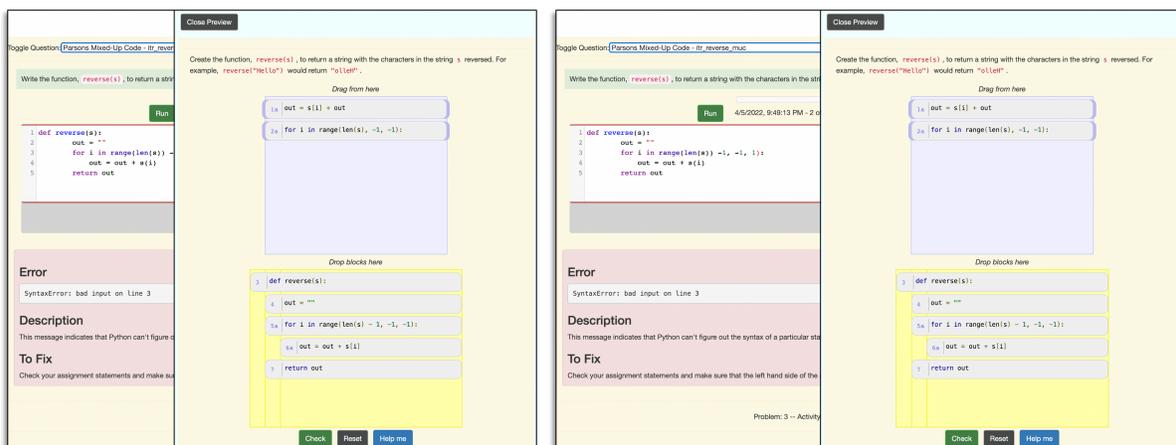


Figure 6.5: Ruenstone’s Toggle Question Feature Before (left) and After (right).

Amanda’s perspectives and attitudes were informed by her background in psychology. She was

consistent in her beliefs about mindset (see Table 6.3 or Figure 6.6). When I followed up with her about whether she perceived intelligence as something that’s fixed or flexible, Amanda said, “I would say malleable. I’m a psychologist and we teach growth mindset. I’m a growth mindset advocate.”

Table 6.3: Amanda’s Responses to the Attitude Survey

Factor	Week One	Week Two	Week Three	Week Four	Week Five	Week Six	Week Seven	Week Eight
(F1) Problem Solving - Fixed Mindset	3	3	3	3	3	3	3	3
(F2) Gender Equity	3	5	5	5	4	5	5	5
(F3) Importance	2	3	2	2	2	2	3	3
(F4) Problem Solving - Strategies	2	3	3	3	4	3	3	3
(F5) Gender Bias	3	1	2	3	1	1	1	1
(F6) Personal Interest	2	3	2	3	2	2	2	3

Note: Likert scale: 1 = Strongly disagree, 2 = Somewhat disagree, 3 = Neither agree nor disagree, 4 = Somewhat agree, 5 = Strongly agree.

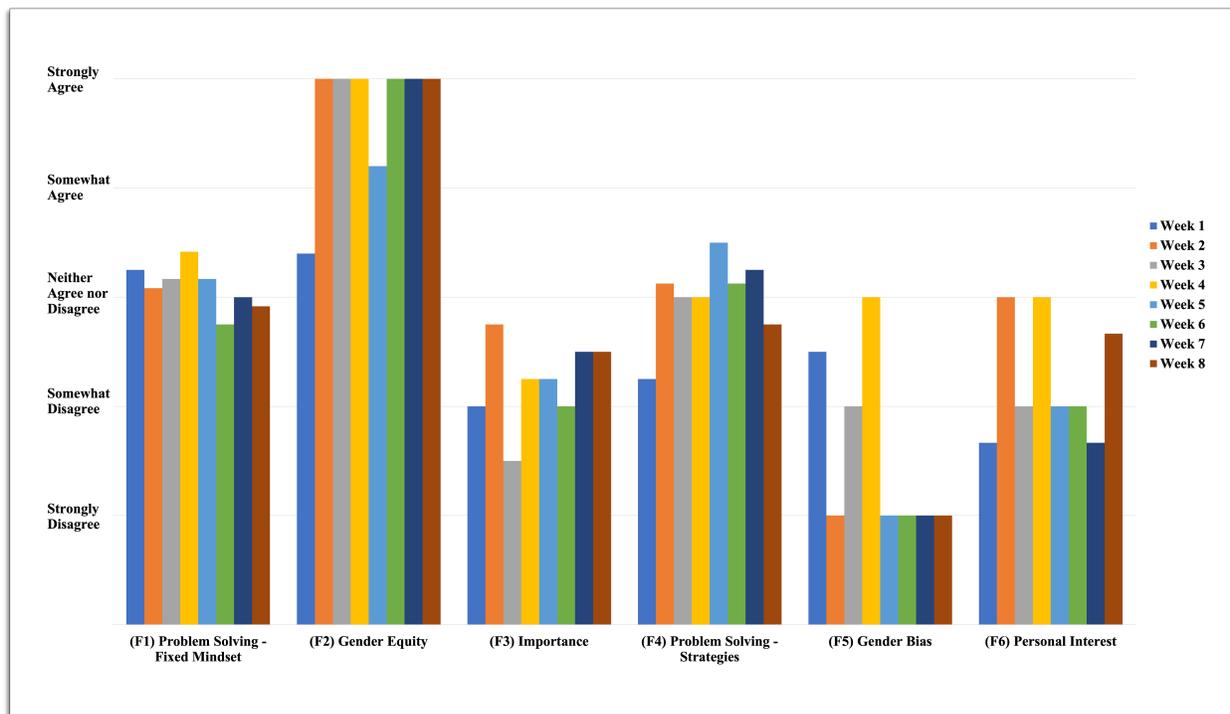


Figure 6.6: Amanda’s Responses to the Attitude Survey

When asked if she believed gender impacted a person’s experience when learning how to program and why or why not, Amanda said, “No. Those are the only questions that I respond to very polarly. Every time I answer no, no, no. I think there is the stereotype for sure. And I’m sure women experience stereotype threat. But I don’t believe it has an impact at all.” And when asked what led to her beliefs, Amanda said, “I probably started noticing gender stereotypes and gender roles while taking some classes in undergrad. It was the first time that I really analyzed it, so I

think my foundation is rooted in my undergrad education on identifying gender stereotypes and roles.”

Amanda’s interest in computing remained low but went up during the second week on debugging (see Figure 6.6). When asked about this, she said it was because it “was one of the shorter [readings]...it was manageable and [she] understood it.”

When I followed up with her about some the responses to the self-efficacy questionnaire (see Table 6.4 or Figure 6.7), Amanda expressed perseverance about completing the research practice assignments. She said, “Even though I’m really bad at them and I’m so confused and so lost, I still feel like I could get it. I just feel like I would need to dedicate way more time to it than what is listed.” And when asked what was motivating her to continue the readings and the practice, she said, “Honestly, to participate in research, give you feedback on what the program looks like for me. You. Your study.”

Table 6.4: Amanda’s Responses to the Self-Efficacy Questionnaire

Question	Week One	Week Two	Week Three	Week Four	Week Five	Week Six	Week Seven	Week Eight
1. I’m certain I can understand the most difficult material presented in the readings.	2	5	5	2	3	3	4	2
2. I’m confident I can understand the basic concepts taught in this eBook.	2	4	5	2	3	3	3	2
3. I’m confident I can understand the most complex material presented in this eBook.	2	4	5	2	3	2	3	2
4. I’m confident I can do an excellent job on the practice assignments in this eBook.	2	5	5	2	2	1	1	2
5. I’m certain I can master the skills being taught in this eBook.	2	3	3	2	3	2	2	2

Note: Likert scale: 0 = Not at all true of me, 2 = Untrue of me, 3 = Somewhat untrue of me, 4 = Neutral, 5 = Somewhat true of me, 6 = True of me, 7 = Very true of me.

I also asked Amanda about her perspectives and attitudes about the assignments, whether she felt they were improving or not improving her knowledge and understanding of programming. She said, “In the beginning they were, and I feel like now it’s just too hard...beyond my reach. I feel like it’s more...it’s frustrating and because I don’t go back to the reading...I’ll forget things or not do anything or have a seizure and then just want to finish it.”

Her main recommendation for a type of support or intervention was “...having a box that shows up which tells you to go back to the reading that says ‘You don’t understand this, so reread this part.’...It’s really frustrating when you can’t complete a problem....especially the [Parsons problems]...so if some thing could popup and then I could redo it and complete it successfully, I would feel much better about myself and think, ‘Oh, I could learn this.’ ”

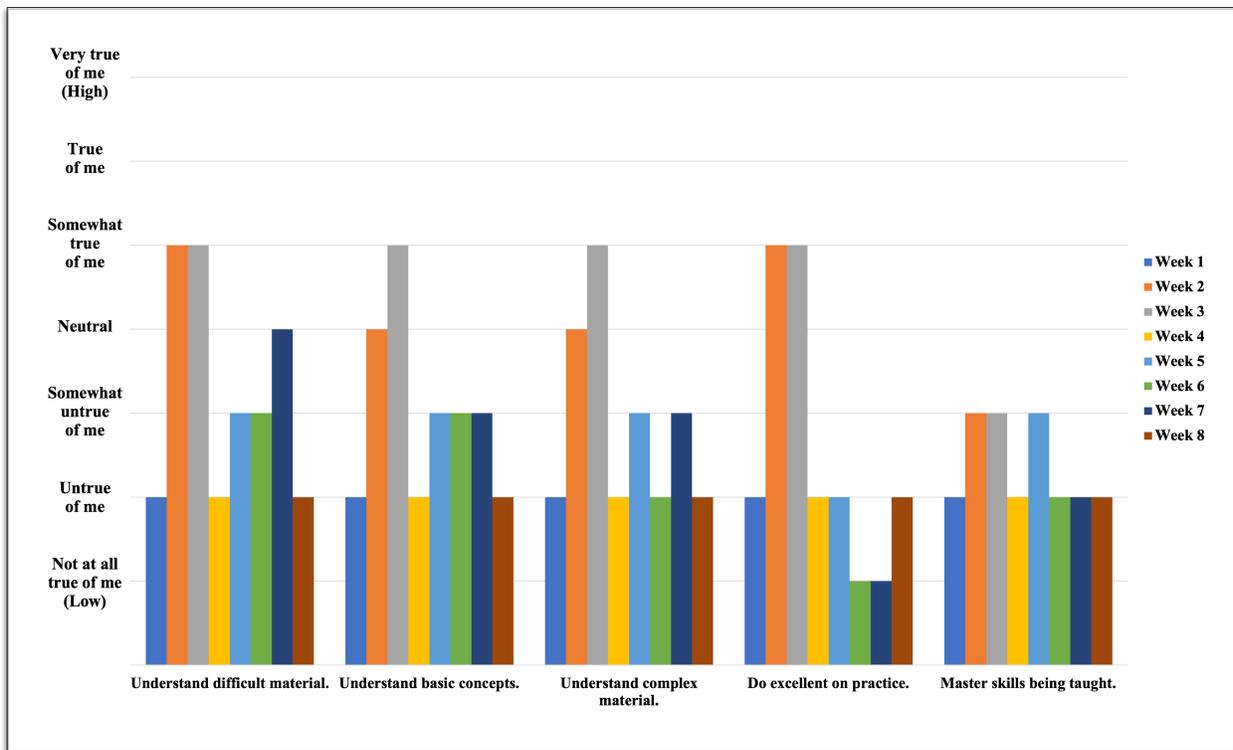


Figure 6.7: Amanda’s Responses to the Self-Efficacy Questionnaire

6.3.1.3 Self-Assessment

Amanda reported no concerns about the frequency of filling out instruments in the study, but had concerns about seeing how much mental effort others reported investing in solving problems.

When asked about filling out the attitude survey, Amanda said, “It doesn’t affect me in any way. I definitely think if I was answering those questions before an assignment it would be different. I’m glad to have them after. But it doesn’t really affect me. My answers just never change. I’m curious if you’d find people’s answers change. I imagine they don’t. I could see doing a pre-post test and a woman, in particular, doing this and then failing at it and maybe re-evaluating and saying, maybe women aren’t as good at this. I think people already have their preconceptions about these stereotypes...and I don’t think it would trigger anything.”

Amanda said she was also fine with the frequency of filling out the Paas scale after every problem and the Computer Science Cognitive Load Component Survey (CS CLCS) each week (see Table 6.5 or Figure 6.8). And when asked how she assessed how much mental effort she invests in solving problems, Amanda said, “I just feel like in general it’s always high effort and then I decide...could I really not figure this out. Was I really engage in thinking about it or was it just kind of difficult and then I figured it out.” In particular, for the Paas scale, she said, “I think it would be harmful to see [how] other people [responded] and possibly think, ‘Oh, people

don't think this is hard and I don't get it.'...in this cohort [of students with disabilities]....It would be helpful if they also were having a really hard time. I'd think, 'Ok. I'm not the only one who doesn't get this. I guess it's like context dependent.' ”

Amanda said she would not have found it helpful to see a learning analytics dashboard displaying how she or others responded to the different instruments over time.

Table 6.5: Amanda's Responses to the Computer Science Cognitive Load Component Survey

Factor	Week One	Week Two	Week Three	Week Four	Week Five	Week Six	Week Seven	Week Eight
(F1) Intrinsic load	10	2	7	10	9	8	9	9
(F2) Extraneous load	3	0	3	5	4	3	1	4
(F3) Germane load	2	8	3	2	3	3	5	3

Note: Likert scale: 0 = Not at all to 10 = Completely the case.

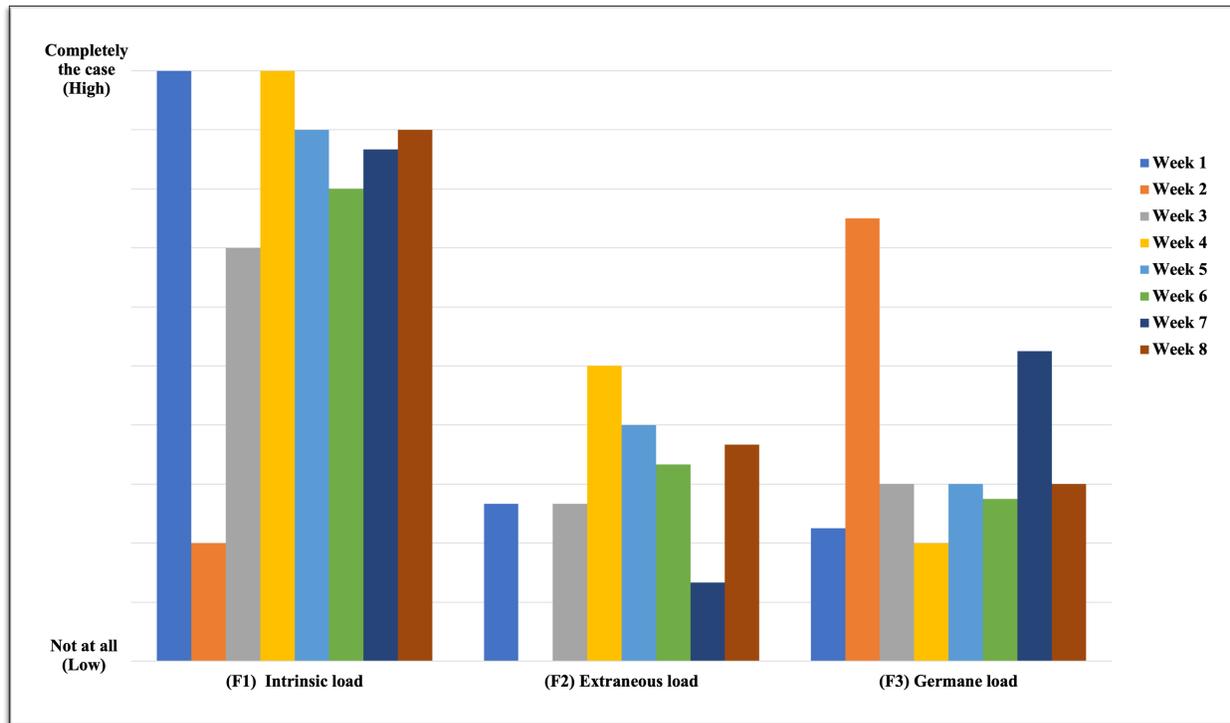


Figure 6.8: Amanda's Responses to the Computer Science Cognitive Load Component Survey

6.3.2 Claire

Claire started the study on October 27th, 2021 and ended it on January 10th, 2022 (10 weeks and 5 days).

6.3.2.1 Accessibility Barriers/Challenges and Benefits

Akin to Amanda, most weeks Claire responded that she did not encounter any accessibility barriers/challenges while using the eBook to read and practice programming. During week seven, when learning about lists, Claire wrote, “I had a lot more trouble with this assignment. I feel like the practice problems were a lot harder than the examples shown, and I didn’t really know how to begin (even with the Parsons questions).”

When asked if there was anything about the eBook and the programming practice that she thought worked particularly well for learners with ADHD, Claire said:

I would say what works best for me is everything being broken down into small chunks. If I see a long reading...sometimes I just don't want to read it and take the time to read it. Or sometimes I do read it, but then I don't actually absorb what I'm reading. Then it's just a waste of reading. And so having things in small chunks makes it more approachable and helps me retain more. Also, having exercises or short multiple choice questions after a short reading kind of holds me accountable which I need. I think that's different from how most books are structured which just want you to read like 20 pages and hope you got what you needed to do. I can do it, but it's just a lot less pleasant for me.

She said, “Yes,” when asked if the shorter chunks helped her focus and gave an example: “Let’s say there’s eight sections. Even if it could be three longer sections. The shorter sections helps me it feel better about starting it and then finishing a couple of sections. And I could go back and start on the new section [if I need a break]. I don’t like leaving off in the middle of things.”

Claire had said during the onboarding process that she preferred print books because they were less of a distraction, so I followed up with her about her experience with the eBook. I asked if she’d experienced any distractions while using the eBook and Claire said, “Yes, it’s kind of inevitable when I’m on a computer. I can go to any other website and do anything else. I think this is definitely working better. I think the fact that it’s cut into shorter sections, so that I’m not just scrolling through pages and pages of stuff. I like to finish things and then move on to the next section even if that means there’s more sections.”

When asked if the flexible assignment deadlines and learning outside of the classroom worked for her and why or why not, Claire said, “Yes. Definitely. My college workload from my classes is always fluctuating, so I work on this when I have downtime rather than when it needs to get done...that lets me take my time with things because if I’m just trying to get something done it’s just to get it done and not to absorb any of the material. I like to learn things. I don’t like to rush to get stuff done, but sometimes you have to do that. Also, I was sick for a lot of November. It kind

of threw me off and put me behind on a lot of things so not having to get this done was also nice because I was already behind and had to catch up...having it be less of an obligation is nice.”

Claire reported that requirements for other classes and procrastination significantly interfered with her ability to learn and complete the reading and practice assignments. During week six, when learning about strings, Claire also wrote, “My meds were wearing off near the end, which made this a little more difficult (more of my own problem, not a problem caused by the eBook).”

6.3.2.2 Computational Practices, Perspectives, and Attitudes

Claire’s main computational practices included note-taking, information-seeking, and highlighting portions of texts to help her focus.

When in need of help with a problem or when struggling with a problem, she said she would, “Google things or [she’d] go back to one of [the Parsons problems] and look at the syntax it used or where the parentheses are going...that kind of thing and try to apply it to what [she was] doing.”

Claire did not really have trouble solving the write-code and Parsons problems during the first think-aloud session. She did select two blocks and then hit reset and highlighted the instructions when solving one of them (see Figure 6.9). Then she proceeded to put all the blocks in the correct order in one attempt. During this session, she also used the solutions from previous problems to help her solve later problems and completed all of them in one attempt. She did not engage in help-seeking. Claire indicated that she highlighted that portion of the instructions “just to group those together because otherwise I just read this and I just see numbers. [Highlighting it] pulls my attention to the price [and how it’s] all grouped...it’s just for my attention.”

I followed up and asked her if she would find it helpful to have a feature that helped her focus on the instructions. Claire suggested using bullet points, “I like bullet points, so I guess if this were bullet pointed then that would call my attention to it. I highlight it because I have to go back and figure out where it was in the paragraph [instructions]. [Highlighting helps me get] back to it immediately.”

Claire also had trouble with indentation when solving Parsons problems (see Figure 6.10). During the second think-aloud session, she said, “I think this embedding is like a minor error because I struggle more with getting the code right...it’s in the right order, so I understand it, but then I just need to fix my understanding of the indents.”

During another think-aloud session, I noticed that she used her notes to solve one of the problems (see Appendix E). Her notes on strings helped her to remember the `len` function and syntax—whether the “colon [went] after the define function or not.” When I asked if it would be useful to have a feature that kept notes in the eBook, Claire said, “Maybe. But two things about that. One, I am very much a handwritten note taker. It helps me understand it and remember it a lot more than if I just copy and paste or highlight. And the second thing is, I don’t like to switch

The following code should set `price` to 1.5 if `weight` is less than 2, otherwise set `price` to 1.3, then set `total` to the `weight` times `price`. For example, if `weight` is 0.5 then `price` should be set to 1.5 and `total` will be 0.75. Be sure to indent correctly!

Drag from here

```

1 total = weight * price
2 price = 1.50
3 weight = 0.5
  numItems = 5
  if weight < 2:

```

Drop blocks here

```

5 if weight >= 2:
4   price = 1.30

```

Check
Reset
Help me

Figure 6.9: Claire Highlights the Instructions.

Create the function, `countup_str(start)`, to return a string with the numbers from 1 to `end`. For example, `countup_str(5)` would return "12345".

Drag from here

```

1 out = ""
2a out = out + str(i)
2b out = out + i
5a for i in range(1, end + 1):
5b for i in range(1, end):

```

Drop blocks here

```

4 def countup_str(end):
3   return out

```

Check
Reset
Help me

Create the function, `countup_str(start)`, to return a string with the numbers from 1 to `end`. For example, `countup_str(5)` would return "12345".

Drag from here

```

1 out = ""
2b out = out + i
5b for i in range(1, end):

```

Drop blocks here

```

4 def countup_str(end):
5a for i in range(1, end + 1):
2a out = out + str(i)
3   return out

```

Check
Reset
Help me

Your program is too short. Add more blocks.

Create the function, `countup_str(start)`, to return a string with the numbers from 1 to `end`. For example, `countup_str(5)` would return "12345".

Drag from here

```

2b out = out + i
5b for i in range(1, end):

```

Drop blocks here

```

4 def countup_str(end):
  out = ""
  for i in range(1, end + 1):
    out = out + str(i)
3   return out

```

Check
Reset
Help me

These blocks are not indented correctly. To indent a block more, drag it to the right. To reduce the indentation, drag it to the left.

Create the function, `countup_str(start)`, to return a string with the numbers from 1 to `end`. For example, `countup_str(5)` would return "12345".

Drag from here

```

2b out = out + i
5b for i in range(1, end):

```

Drop blocks here

```

4 def countup_str(end):
1   out = ""
5a for i in range(1, end + 1):
2a out = out + str(i)
3   return out

```

Check
Reset
Help me

Perfect! It took you 3 tries to solve this. Click Reset to try to solve it in one attempt.

Figure 6.10: Claire's Attempts to Solve Problem Two - Second Think-Aloud Session

between screens. I like to be able to have whatever I'm working on in front of me and then have notes next to me or a second browser open so that I can be looking at two things at one time. If I had to switch back and forth then I just lose what I'm looking at and it doesn't work as well for me."

When asked about the practice of switching between problem types, Claire said she didn't notice the toggle question button until I mentioned it. She said, "I would assume that it would go to [a Parsons problem] because now I see that's the next step. But honestly, even if I had seen it, I probably would not have thought it was for me to mess around with." She said she would use the toggle question's preview mode for "guidance if [she] got stuck on write-code problems....and to learn from it." Claire said she would toggle from solving a Parsons problem to a write-code problem if she "struggled with remembering the flow of the problem—what [blocks go] where. I think that this sort of active recall [solving write-code problems] reinforces it a little bit more."

During one of the think-aloud sessions, when solving a problem with three distractor blocks (see 6.11), Claire expressed that she was not distracted by them. She was focused and knew what she was looking for. Claire said, "I think I assumed it meant that one of them would be correct. But to be honest, I don't know that I totally pay attention to that when solving it because I just was like, 'Okay, this is what I'm looking for.'" She said she was not distracted by them because she "didn't pay enough attention them." When I followed up with her at the end about distractor blocks, she said, "I actually think the distractors are beneficial because I have to think about the code itself rather than just the order...having the [distractor blocks] makes me think about which one of these is the correct way to do this so for the future I've already thought about that and processed it a little more because without it, I'm kind of prone to just glancing at it and saying, 'Oh, that looks right'. But then I didn't have to think about exactly what it is."

She said she thought distractor blocks would be better for some learners with ADHD. Claire said "because if I read some thing, sometimes I'll just read it for....this is an analogy, but I've learned a lot of foreign languages and I found when I'm trying to learn vocab and I just look at a word, I'll understand what the word. But then if I have to reproduce it, I won't know how to spell it correctly....the [distractors] make me think about the spelling or how it's written. And then I understand it better. I actually have to look at it and think about it." She agreed with the statement that the distractor blocks caused her to focus and pay attention.

Claire reported having a growth mindset consistently (see Table 6.6 or Figure 6.12). When I asked her whether she perceived intelligence as something that's fixed or flexible, Claire said, "I think in some senses it is a little bit rigid. I think people have a range or a boundary of capability, but then can work within that range. I don't know think it determines your life anyways. I feel like intelligence is a weird concept, and I don't totally know how much it determines about your life or your capabilities. I think motivation and work ethic are more important than intelligence."

Arrange the code to calculate and print the cost of a 14 mile cab ride. If the distance traveled is less than or equal to 12 miles the cost is \$2.00 a mile, and if the distance traveled is more than 12 miles the cost is \$1.50 a mile. Be sure to indent correctly and look out for extra code blocks!

Drag from here

```

1 | rate = 1.50
2 | print("Total cost of trip: " + str(total))
3a | total = distance * rate
3b | total = distance / rate
4a | if distance is 12:
4b | if distance <= 12:
5a | if distance < 12:
5b | if distance > 12:
6 | distance = 14
7 | rate = 2.00

```

Drop blocks here

Check Reset Help me

Figure 6.11: Claire Encounters Distractor Blocks

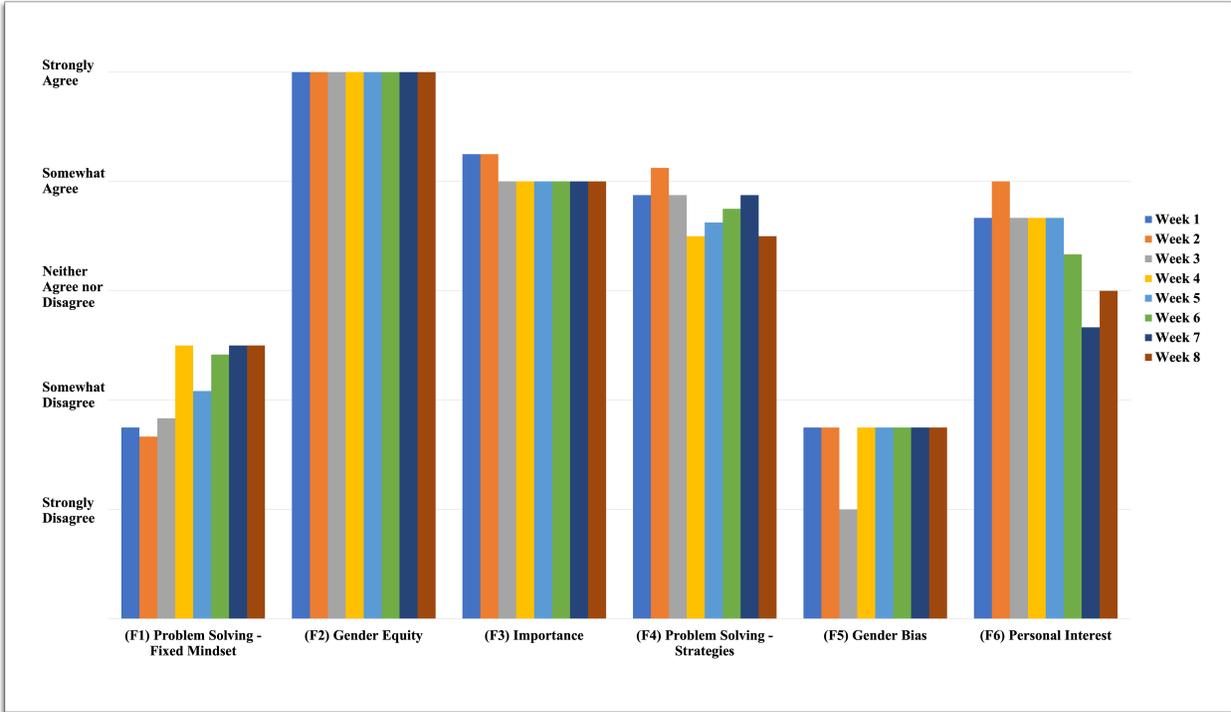


Figure 6.12: Claire's Responses to the Attitude Survey

Table 6.6: Claire’s Responses to the Attitude Survey

Factor	Week One	Week Two	Week Three	Week Four	Week Five	Week Six	Week Seven	Week Eight
(F1) Problem Solving - Fixed Mindset	2	2	2	3	2	2	3	3
(F2) Gender Equity	5	5	5	5	5	5	5	5
(F3) Importance	4	4	4	4	4	4	4	4
(F4) Problem Solving - Strategies	4	4	4	4	4	4	4	4
(F5) Gender Bias	2	2	1	2	2	2	2	2
(F6) Personal Interest	4	4	4	4	4	3	3	3

Note: Likert scale: 1 = Strongly disagree; 2 = Somewhat disagree; 3 = Neither agree nor disagree; 4 = Somewhat agree; 5 = Strongly agree.

When asked if she believed gender impacted a person’s experience when learning how to program and why or why not, Claire said, “Yes. I think that a lot of our society and institutions have structural or inherent bias towards men, and I’d say white men, especially in STEM fields. I think people inherently think that they’re more capable of doing something. And so I think that comes back in two ways. I think a lot of woman, because that’s what I can speak to, but I would add a lot of minorities also, probably feel that they aren’t capable of doing something like that so they don’t even try. And then if they do, people might not hold their work or their abilities to the same....just think it’s not as good as something that somebody else is doing. But I can only speak to my experience as a woman.” I followed up and asked if there was anything about her experience in general with the research assignments that had impacted her beliefs in a negative or positive way. She said, “I think the way the eBook is set up is very neutral. And I think a lot of issues comes from how our society is as a whole, which is a whole other issue that would take a lot to tackle....I don’t think there’s really anything in here that you can target towards one gender or another. It’s very neutral to me, which I feel is how it should be....I think it’s always good to see representation of people who fit your identity, so in this case, gender identity. I’ll be honest, I don’t know that I totally pay that much attention to [gender bias and equity] though. I just pay more attention to what’s going on in the problem.”

Based on Claire’s responses to the attitude survey about importance and personal interest (see Table 6.6) and the dip in her responses to the self-efficacy questionnaire during week seven on lists (see Table 6.7 or Figure 6.13), I followed up to ask her what was motivating her to continue the readings and the practice, and why she might or might not be interested in learning more about computer science. Claire said, “I think what’s motivating me is, one I feel like everyone knows how to code and I’ve never learned how to code. I think I learned HTML in a high school course once when I was 14, and I like to learn new things. Also, if I’m being honest, I told you I would finish this study and I want to finish it to help you out....I want to make sure you have the data that you need for your study.”

Table 6.7: Claire’s Responses to the Self-Efficacy Questionnaire

Question	Week One	Week Two	Week Three	Week Four	Week Five	Week Six	Week Seven	Week Eight
1. I’m certain I can understand the most difficult material presented in the readings.	5	5	6	5	4	5	3	5
2. I’m confident I can understand the basic concepts taught in this eBook.	7	7	6	5	5	6	5	6
3. I’m confident I can understand the most complex material presented in this eBook.	5	5	5	4	4	5	3	4
4. I’m confident I can do an excellent job on the practice assignments in this eBook.	4	5	5	4	4	5	2	4
5. I’m certain I can master the skills being taught in this eBook.	4	5	5	4	4	5	2	3

Note: Likert scale: 0 = Not at all true of me; 2 = Untrue of me; 3 = Somewhat untrue of me; 4 = Neutral; 5 = Somewhat true of me; 6 = True of me; 7 = Very true of me.

6.3.2.3 Self-Assessment

Claire had low scores for intrinsic and extraneous cognitive load up until week four on functions (see Table 6.8 or Figure 6.14). When I followed up with her about her self-assessment process, Claire said, “I think it was fine. I don’t think it took a very long or an unnecessarily long amount of time. I don’t really remember having any issues with it or being super frustrated. I don’t think I totally have the mindset of somebody who codes. I think that’s something that comes along with time....I think this is where I started to see that there is a certain way that things are done and I just haven’t had enough exposure yet. But I don’t know that I necessarily struggled with the readings.”

Table 6.8: Claire’s Responses to the Computer Science Cognitive Load Component Survey

Factor	Week One	Week Two	Week Three	Week Four	Week Five	Week Six	Week Seven	Week Eight
(F1) Intrinsic load	4	2	4	7	5	6	9	6
(F2) Extraneous load	0	1	2	7	4	3	5	5
(F3) Germane load	9	7	7	5	6	6	5	6

Note: Likert scale: 0 = Not at all to 10 = Completely the case.

She was not confident in her ability to assess how much mental effort she invested in things. When asked about it, Claire said, “I don’t know. I feel like I’m just bad at judging how much mental effort things take me.” When asked to rate items like “the activity really enhanced my knowledge and understanding of computing,” Claire’s germane load remained high because of the interactivity of the eBook. Claire said, “I think the interactivity of it not just being a textbook. To finish a section I actually had to do something to show that I processed what I read. If I just read something, even if I understood it, I probably didn’t process it, which is why I take notes. I have to do that for all of my classes. I also like things being in shorter chunks. Getting it broken down into smaller bits helped me process it. I think the book is done well for a very basic way of teaching Python...at least for me, this is a lot better than most of my textbooks for college....I also want to

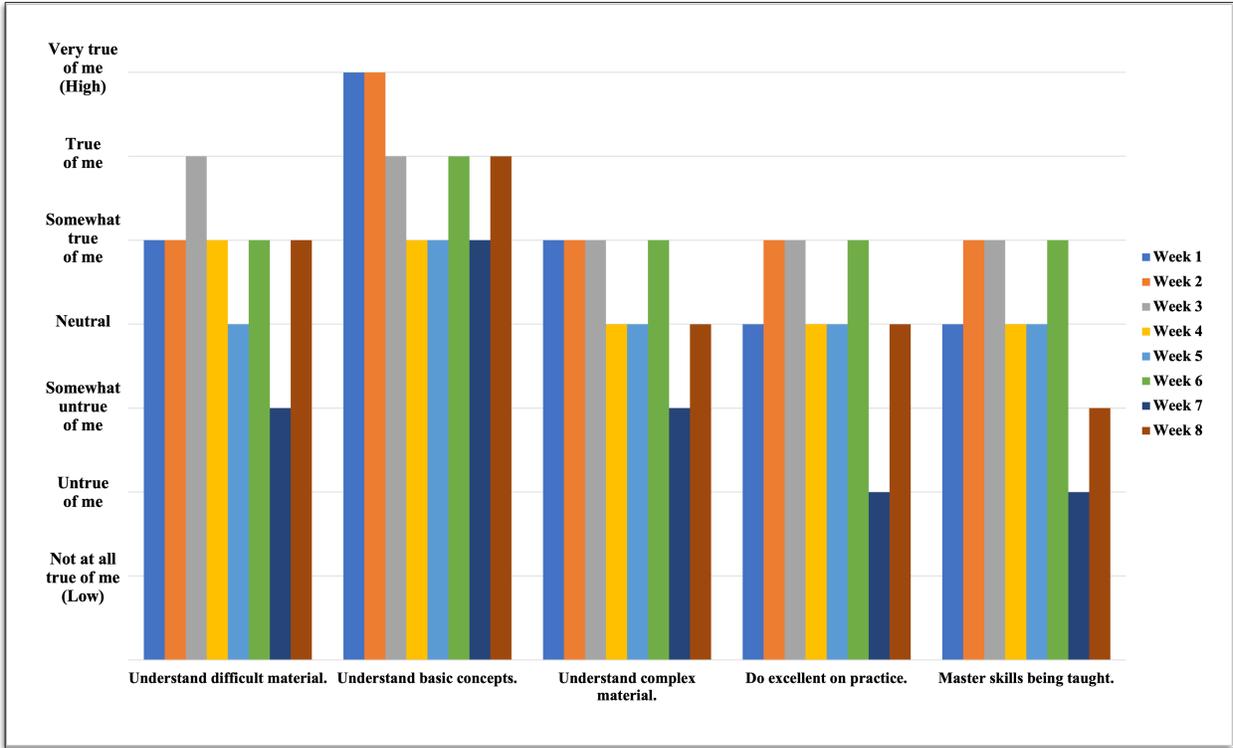


Figure 6.13: Claire's Responses to the Self-Efficacy Questionnaire

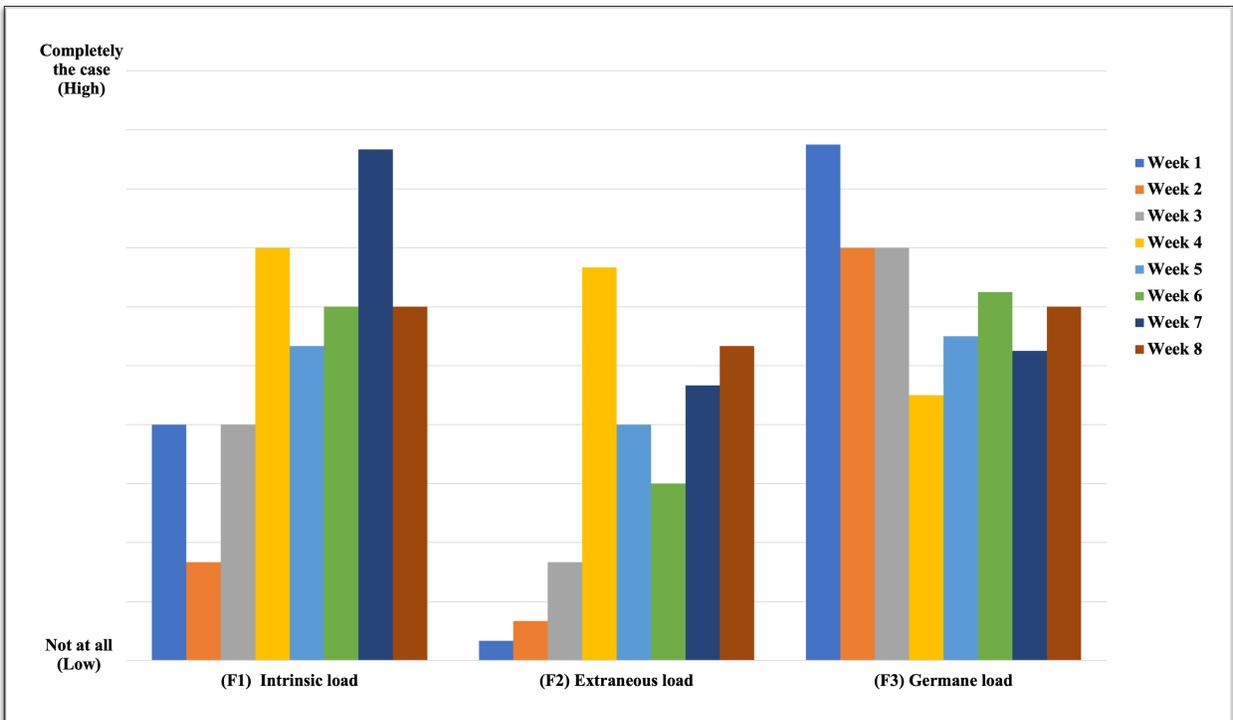


Figure 6.14: Claire's Responses to the Computer Science Cognitive Load Component Survey

say that I'm not used to things being made for accessible for me, so this is a big improvement from stuff I'm normally dealing with. I don't really know what more accessible would look like. I can't fathom it."

When asked if she would find it beneficial to you to see a learning analytics dashboard based on the surveys, Claire said, "I'm going to say no...I'm pretty aware of my frustration level and my understanding...and in terms of seeing what other people are doing. I am a very competitive person, and I compare myself to people a lot in a way that's unhealthy. And so it's better for me to not see what other people are doing and feeling and to just focus on myself."

6.3.3 User

User started the study on October 27th, 2021 and ended it on March 28th, 2022 (21 weeks and 5 days).

6.3.3.1 Accessibility Barriers/Challenges and Benefits

When asked whether or not they encountered any accessibility barriers/challenges while using the interactive eBook to read and practice programming, User raised concerns about time-management, chunking information, reading problem instructions, design, and sources of struggle related to learning. User said:

"I needed to take frequent breaks to complete assignments. I also get very little warning before my brain shuts down completely." Week One - Variables, Expressions, and Statements

"There was one window that was too narrow for the code and I had a bit of trouble using the slider, since I couldn't see the line all at once." Week Two - Debugging

"The same as the first two weeks: it takes me a long time to complete the practice tasks. When an assignment asks to assign values to a variable, it sometimes puts the value first and sometimes second, and that kind of sentence construction is a bit confusing to me, I'd prefer that all clauses would be of the same type." Week Three - Conditional Execution

"Not regarding the eBook per se, but my life is nothing but barriers and challenges in general lately." Week Six - Strings

"None, except I'd like to learn Python when the world is not on fire and I'm not stretched so thin." Week Seven - Lists

“No, but systemic ableism poses many barriers and challenges to my learning of Python. I’m too tired to see this through properly.” Week Eight - Dictionaries

User suggested that the eBook provide more explanation when asked what would be helpful in the eBook itself. They said, “I feel a lot of the times this type of book presupposes some knowledge that I don’t have, like math knowledge. In order to solve the problem, I first have to look up math, because I’m very rusty on that, and then do the problem. It’s kind of like it is directed at a certain population because if you’re a computer science major you probably have your math down, but I don’t....Just don’t make the assumption that this would be easy...that the math itself would be easy for people. And if you have to, if you absolutely have to use the math to explain the programming, then break down the math first. Explain it a little bit....Sometimes I get this feeling that I do the reading and everything is clear to me in the reading. And then I’m being asked to do something that we didn’t study in the practice part. I did get that feeling before during weeks one to four, but week five [on loops and iterations] became really overwhelming....There is always practice in the reading. Interestingly enough, the practice in the reading doesn’t stump me.”

User said they would find it more helpful to have most of the practice interleaved in the reading. They said, “I never have big problems with the reading. If I don’t understand it, I just read it twice or I google something. The reading [assignments] are great. The practice [assignments] are what I have trouble with.” Yet on another occasion, they said, “Sometimes [the practice in the readings] are too easy. Like [see Figure 6.15]. This is first grade stuff. I feel maybe this eBook is not a good fit to how my knowledge works and where I’m at because some things strike me as too easy and some things strike me as too hard.”

User also struggled a bit with some of the problem instructions. During one of the think-aloud sessions, I followed up to ask if they could explain why they struggled with the wording of one of the problems (see Figure 6.16). User said, “Return a new list with... First, I did a double take on ‘passed list’ because I did not encounter this term, but I filled in the blank and though it might have been the list that is given to us in the same order. Yeah, and ‘return a new list with all the strings from the passed list in the same order that have a greater length than three.’ So there are two subordinate clauses, ‘from the past list’ and ‘in the same order.’ And by the time I’m finished with those two subordinate clauses, I have no idea what the third refers to....I worked as a journalist for a number of years and as an editor as well....and then I worked for ten years as a professor. So this is how I would explain to a student why the writing isn’t clear.”

In particular, User stated they had a mental health disability, so I followed up with them about it for context and recommendations. User said,

“I started out with chronic depression and anxiety my entire life, and I was diagnosed with PTSD just under a year ago. For the longest time, I was going just with the

When you update a variable by adding 1 it's called an *increment*; subtracting 1 is called a *decrement*.

Q-3: Before you can update a variable, you have to _____ it.

Check me

Compare me

Q-4: When you subtract 1 from a variable, you _____ it.

Check me

Compare me

Figure 6.15: User's Too Easy

Finish the function, `filter_strings(str_list)`, below to take a list of strings, `str_list`, and return a new list with all the strings from the passed list in the same order that have a length greater than 3. For example, `filter_strings(["Run", "she", "said"])` should return `["said"]` and `filter_strings(["It", "was", "a", "dark", "night"])` should return `["dark", "night"]`.

Run

Load History

Show CodeLens

Share Code

```
1 def filter_strings(str_list):
2     str_list = ["Run", "she", "said"]
3     if string in str_list:
4         len(string) <= 3
5
6     return new_list
```

Figure 6.16: User's Struggled with the Subordinate Clauses in the Instructions

diagnosis of depression and anxiety. And that I have received a lot of treatment for and I'm doing much better on those things...I would say I have been in remission of depression for maybe six years now. It is a marked difference to how I feel now compared to back then. But there is also an issue of low energy and as my depression and anxiety goes down that goes up. I almost feel like I took care of one onion layer and now I'm experiencing all those lower onion layers that I didn't even pay attention to because my mind was busy dealing with immediate things like pain....Now, I'm just starting to notice how I processed things like sounds and social interactions and how it makes me feel drained because I'm more in touch with my body and emotions now that I can actually notice them....When I was in alarm mode all the time, I only knew to take care of immediate depression triggers and kind of manage depression. That was all that I could think about. I was in active mode all the time, so I didn't really have that much low energy. But now that I've been able to relax a little bit in the past couple of years or so, I noticed that my energy levels went down and that [has led to] being tired, being exhausted by the smallest things like a lot of social interaction...just life, normal life....problems with attention and focusing and just basic energy levels."

When asked what kinds of positive content or activities would raise their energy and/or mood, User said, "One of the things that I do for emotional regulation is look at videos of cats....social interaction is also a big one for me. I have to manage it carefully because it is also draining, but it is a requirement to have kind a genuine connection with the person. Having a support network of people that I want in my life is really important and I do have that. I would say empowering social justice content like an article about a social work problem or social media content in which a person understood how taking care of themselves is actually resistance and a revolutionary practice. There's a whole range of things that feel affirming....a social justice program, people would be all over that...gender fluidity. I don't think that kind of thing exists though because programming has been a refuge for marginalized people, but as a mechanism of escapism because it is so removed from the world that there is no pain and discrimination there. It has been associated with that indirectly, but if somebody made something like this people would actually be invested in solving these problems because it's cool."

When asked if the flexible assignment deadlines and learning outside of the classroom worked for them and why or why not, User said, "Well, my procrastination... my anxiety around deadlines is tied to the stakes around the deadline. This has no stakes because this is just for fun and that makes me more likely to engage with it because this is so unimportant that it will not be a reflection of who I am and how I perform if I do dwell on this or get stuck on an assignment and then just skip it. I feel more freedom because it's not graded or embedded into the fabric of my social life. Yesterday I was feeling a bit sick...too sick to actually do schoolwork, but not to actually do

nothing, so I decided today's Python day. And that's how I completed week four."

In the end, User said, "I came to a conclusion that the biggest barrier for me to learning this stuff, which I do want to learn, has not been specific stuff, the eBook, or the course organization, or anything, but just not having time and energy...systemic ableism. [By that] I mean the university and academia not taking into account the personal abilities and boundaries and energy levels. Everyone has to conform to the same standard and this same standard is really harmful for people who do not have any disabilities or lower energy levels. It is just too much. And that has to do with the school trying to provide as much education in as little time so that they can make as much money from students. It's tied in with capitalist foundations of academia and stuff."

They went on to express how much the flexibility helped but that there was room for improvement. User said "I was supposed to finish this stuff in December, and now it's March, and I just finished this stuff, so that's flexibility, right? Without that flexibility, I would not have finished this stuff at all. And with the flexibility, I have been able to finish this stuff, but I have not enjoyed this as much....If a programming course would be geared for people in professions where they need to use Python or where they can enhance their professional skills by using Python, that would, of course, be really cool, and that would probably attract many more people...[for example] how to handle client databases and stuff."

When I asked if they felt learning outside of the classroom made learning how to program more or less complex, User said, "Well, I would prefer to have a teacher because I don't think that we have a teacher while we self-learn out of the book. And I do ask my partner, but my partner knows other languages, but not Python....so that is also not very good counsel. I think it's fine. I think the textbook is okay....sometimes a problem is raised in a way that is a little bit jumbled. I don't have a baseline of how else you could learn it, so I'm like, Okay, this is how you learn Python."

I checked in with participants every now and then to remind them about the research assignments. My February email to User read:

Hi User,

I hope you're doing well! It looks like you only have the following to complete: Week 6 Strings, Week 7 Lists, and Week 8 Dictionaries.

Don't forget to fill out the following surveys after each set of assignments:

1. [Accessibility Barriers and Challenges](#)
2. [Attitudes Survey](#)
3. [Computer Science Cognitive Load Component Survey](#)
4. [Self-Efficacy Questionnaire](#)

Best,

Carl

They responded:

Thanks for checking in! Right now I'm just trying to calm down from the news of the war and the Texas anti-trans letter, and tomorrow I'm going to Chicago until Sunday night but during the spring break I think I'll be able to finish all the assignments.

I followed up with them about the role of life while learning outside the classroom because they were impacted by the 2022 Russian invasion of Ukraine and anti-trans events. They also reported that requirements for other classes, work obligations, confusion about the material, getting stuck on bugs, and self-doubt/lack of confidence significantly interfered with their ability to learn and complete the reading and practice assignments. User said, "Many of [instructors] reached out, and a lot of them were really supportive, super supportive. They just said, 'Whatever you need and if you need to submit something later, then you can,' and I did. But even with that flexibility, it's too much...and in the field specifically, they have not been really good with accommodations."

6.3.3.2 Computational Practices, Perspectives, and Attitudes

User's computational practices during problem-solving included the use of trial-and-error. During the second think-aloud session, User said, "This is very representative of how I got through week five [on loops and iterations]. I don't understand what's happening. And I wouldn't be able to write the code on my own. But I got an idea of the syntax enough to do it by trial-and-error."

Furthermore, when asked what would you say are some practices you have while learning how to program via the eBook, User said, “I mean, I definitely lookup concepts if I don’t understand them. If something in the book is confusing and not explained in a way that I understand it, I will definitely google and see other tutorials. That is something that I didn’t have to do much in this book because the explanations were pretty clear. Until the time that I had to code myself, I could understand pretty well, but this is something that I would do if I don’t understand things. Then, I would do all the assignments in the reading section because those, I feel that they relate to a specific explanation better than the assignments in the practice session because if they just explained lists and indexing, then I know that the assignment is going to be on the indexing. I know what to pay attention to. But when you have to be more creative and apply what you have learned in a more freestyle way, that’s where I get lost.”

I also asked what their perspective was about distractors blocks, were too many or too little when solving a problem with three and they said, “I thought that was a reasonable amount.” And when asked if they felt distractor blocks increased their cognitive load or lessen it, User said, “As opposed to just using all the blocks, it increases the cognitive load. But it also provides education, so I guess it’s a good trade-off.”

Different sources of struggle resulted in User taking frequent breaks, so I followed up with them about their computational practices related to learning intervals. When asked if they preferred spaced repetition or mass repetition, User said, “Since programming is so far from everything that I deal with in my professional life, I feel it is trickling down over time. My first attempt trying to learn Python was with my girlfriend teaching me. And she’s not a teacher, so that went incredibly awry with me not learning much and her being frustrated. But I feel that I am doing better in this course because I had that little bit of explanation way back and now I’m like, ‘Oh, this is what [my girlfriend] meant by this.’ I feel that I’m not going to learn Python in this iteration, and I’m pretty sure of that. But if in a years time, I decide I actually want to be able to do something with it. I feel like this would be the third layer in which I would understand better. I like spaced out and returning to the material regularly. Having new layers of stuff would be helpful for me...I would probably enjoy a python intensive [course].” When asked if they would prefer the system recommend when to take breaks, User said, “No. I have specifically worked on being in touch with myself and how I feel. I would prefer to determine that by myself.”

When asked about the practice of switching between problem types, User preferred that toggling to and completing the Parsons problem would result in the solution being transferred to the write-code problem, but they found it helpful to write the code to “get a little bit more of a feel [for writing the] syntax.” User said they would toggle to Parsons problems only if they got stuck on a write-code problem. When commenting on the different problem types, they said, “Well, drag and drop is definitely easier because the input is already there. Writing code from scratch is definitely

harder....but it is more beneficial because I learned more this way. ” User did not find the help me button helpful because “most of the time [they] solved [Parsons problems] before three tries.”

User’s responses to the attitude survey are shown in Table 6.9 and Figure 6.17. User’s perspectives and attitudes were informed by their sexual orientation and gender identity. When asked if they believed gender impacted a person’s experience when learning how to program and why or why not, User said, “Absolutely, because of how we have been conditioned. If women and various sorts of non-men have been conditioned from childhood to believe that this is not their domain, they’re less likely to venture into that domain. And when they do venture into that domain, they’re more likely to doubt themselves.” I asked what kind of intervention they thought might improve this situation and User said, “I don’t know if programs [geared toward] women in programming would actually be effective because they might feel like you’re singling out a specific population. You’re trying to correct the existing inequality, but you are focusing on the inequality, which might feel uncomfortable for some people, I think. And also excluding other gender identities that might need this kind of intervention. Take what’s happening right here. I very much was condition to think programming was not my thing. But now, I’m actually being taught how to program and maybe having more people from marginalized backgrounds in any kind of course or study—just do affirmative action around that—without necessarily showing the people that that is what you’re doing may be a good way to go. I’m enjoying this course a lot.”

Table 6.9: User’s Responses to the Attitude Survey

Factor	Week One	Week Two	Week Three	Week Four	Week Five	Week Six	Week Seven	Week Eight
(F1) Problem Solving - Fixed Mindset	3	3	3	3	4	3	4	3
(F2) Gender Equity	5	5	5	5	5	5	5	5
(F3) Importance	4	4	5	5	3	4	3	3
(F4) Problem Solving - Strategies	3	3	3	4	2	3	3	3
(F5) Gender Bias	2	2	2	2	2	3	3	3
(F6) Personal Interest	3	4	5	4	1	2	2	2

Note: Likert scale: 1 = Strongly disagree; 2 = Somewhat disagree; 3 = Neither agree nor disagree; 4 = Somewhat agree; 5 = Strongly agree.

User’s responses to the self-efficacy questionnaire are shown in Table 6.10 and Figure 6.18. With the exception of question two, User’s self-efficacy responses declined. When I followed up to ask them how they felt about the readings and their impact on their understanding and self-efficacy, User said, “I mean this is hard. I haven’t done anything like this before and it takes me a while. And I can do most things, but some things I just can’t. If I didn’t have this very big workload, I probably would have figured them out. But it would have taken me a lot of time so some things I just skip but not much.”

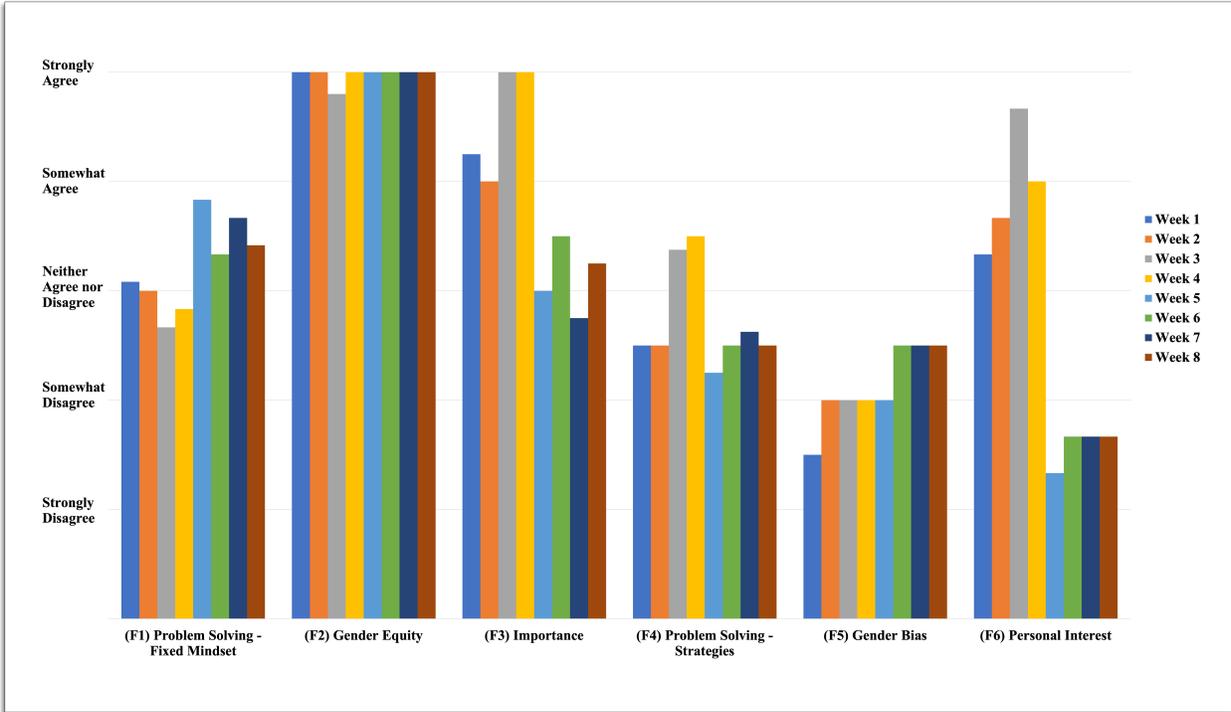


Figure 6.17: User's Responses to the Attitude Survey

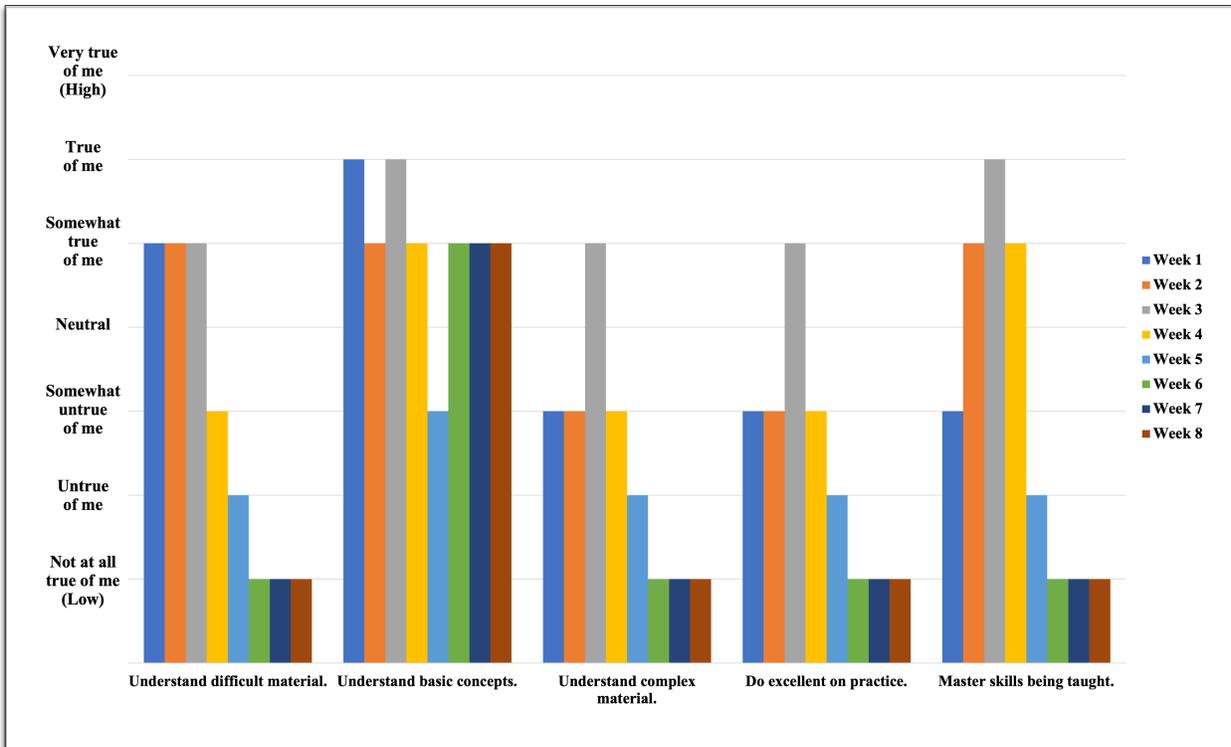


Figure 6.18: User's Responses to the Self-Efficacy Questionnaire

Table 6.10: User’s Responses to the Self-Efficacy Questionnaire

Question	Week One	Week Two	Week Three	Week Four	Week Five	Week Six	Week Seven	Week Eight
1. I’m certain I can understand the most difficult material presented in the readings.	5	5	5	3	2	1	1	1
2. I’m confident I can understand the basic concepts taught in this eBook.	6	5	6	5	3	5	5	5
3. I’m confident I can understand the most complex material presented in this eBook.	3	3	5	3	2	1	1	1
4. I’m confident I can do an excellent job on the practice assignments in this eBook.	3	3	5	3	2	1	1	1
5. I’m certain I can master the skills being taught in this eBook.	3	5	6	5	2	1	1	1

Note: Likert scale: 0 = Not at all true of me; 2 = Untrue of me; 3 = Somewhat untrue of me; 4 = Neutral; 5 = Somewhat true of me; 6 = True of me; 7 = Very true of me.

6.3.3.3 Self-Assessment

During the first think-aloud session, User had some comments about the attitude survey. They said, “I have a little feedback on how your attitude survey is designed. There were a couple of questions that I was confused about. There are a couple where I have answered neither or neutral. One of them is ‘It makes sense that there are less women than men in computer science’. It makes total sense because of sexism and all the barriers. I think the answer it is looking for is that it doesn’t make sense, but socially it does. I think I haven’t really been answering what the question is asking....There is one more question about women who are interested in computer science being a bit peculiar and for me that’s like a compliment, so I also answered neutral. Those are the only two questions that were kind of unclear. I feel like my answers to those questions will be pretty consistent except for those two unclear questions because I’m confused about how to answer them. My answers to those two questions might fluctuate because I forget what I chose last time, but the answers that reflect my actual beliefs [such as those about whether or not I] would have more faith in a program written by a man, should be consistent....I just answer about women and men because those are populations, but if you want to [have] less gender binary questions....I just think at this point that that kind of language, women and men, is just a little bit passé.”

User’s responses to the Computer Science Cognitive Load Component Survey (CS CLCS) are shown in Table 6.11 and Figure 6.19. They found week six on strings the most confusing and this showed in their cognitive load response. When asked about it, User said, “I think before I went on this long break, I understood everything....I feel I would be able to understand it if I just had more brain space. I get to this assignment after I’ve had a full day at school and my brain is so foggy and it’s kind of unpleasant and also an unusual situation for me because I’m used to my brain feeling sharp. I’m used to being able to understand anything that I put my mind to. It has been getting worse in the past two years since pandemic.”

User expressed that they use time-on-task and comprehension to assess how much mental effort they invested in the readings and problem-solving. They said, “Mainly time, how much time I spend on it, and if it is something that matches my current state of mind, good. I would basically read the instructions, then read the task for a problem. Then it would make sense to me, and I would understand how to do it, and then I would do it. That amount of time I have in my brain as a baseline. So if it was just like, ‘Oh, this is too easy. I know how this goes. Let me just put them in order,’ this would be easier cognitive load.” Another time, they remarked, “I actually did not understand how it worked. I just did what the system required of me, and that’s why it was low mental effort.”

Later on User said, “If I read the task and I don’t even understand how to approach it, and then I have to do several tries, then that would be high cognitive load. So for the latest ones, I might have marked easy cognitive load because at some point, I stopped understanding what was going on and started doing assignments purely out of understanding of syntax [I’d think], ‘Where would particular block of code go?’ because the syntax, I do know, and that is a lighter cognitive load than actually understanding because this is just mix and match.”

At one point during the last think-aloud, when talking about Parsons problems, they explained why they rated investing high cognitive load in solving a problem. User said, “For this one, I would say high mental effort because I actually tried something. It didn’t work. I tried to reason why it didn’t work...It wasn’t easy. I didn’t feel that it was easy. If I had to write the code, it wouldn’t have increased my understanding because I just wouldn’t have done it.” When asked if they could think about ways to reduce cognitive load that they would’ve liked to see, User said, “Like going slower because the pace at which the material was introduced was definitely very high. It was basically just, ‘Here’s a new topic that you know nothing about. Read a couple of sentences about each concept, and then you have to be able to do problems,’ and I just can’t.”

Table 6.11: User’s Responses to the Computer Science Cognitive Load Component Survey

Factor	Week One	Week Two	Week Three	Week Four	Week Five	Week Six	Week Seven	Week Eight
(F1) Intrinsic load	6	7	9	9	8	10	8	9
(F2) Extraneous load	4	1	5	4	5	4	3	5
(F3) Germane load	9	9	9	8	6	4	3	1

Note: Likert scale: 0 = Not at all to 10 = Completely the case.

When asked if they would find it beneficial to see a learning analytics dashboard based on the surveys, User said, “Yeah. I would probably want to see my answers. I feel that they might have been quite random because I’ve been doing it over a long period of time. Each time I filled out a new survey, I had no idea how I filled out the last one, so I had to think about the stuff. Remember, we had this discussion about the questions about such as, ‘Do you think that women program are a bit peculiar?’ and how the equivalence of that question is not very good because it could be

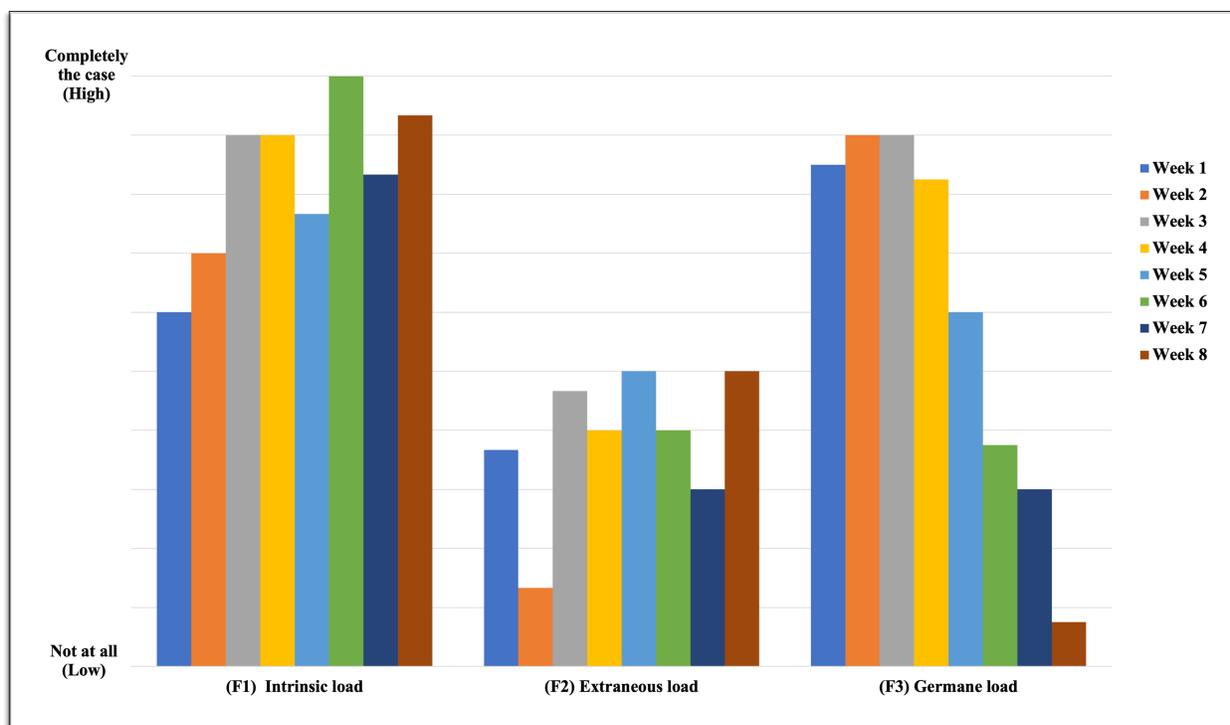


Figure 6.19: User’s Responses to the Computer Science Cognitive Load Component Survey

interpreted differently. So for one, I might have thought, ‘Oh, this is a bad question. I’m going to disagree...strongly disagree,’ and the another time, I might have though, ‘Oh, I’m actually going to answer how I understand it, which is yeah, they are peculiar. That’s a good thing.’ My answers might have been strongly disagree or somewhat agree, but they would basically indicate the same attitude.”

They went on to say, “I would like to see how I did just to have an overview of how differently I answered. I definitely know that my attitudes about confidence and ability to learn Python went down by the end of it because I was just very tired and not understanding much. I remember that happened, but everything else, it would probably be interesting to see how I answered. I don’t think that my attitudes changed much, except the ability to learn Python, but maybe they did, and I didn’t notice. That would be interesting as well to see....Also, maybe if I see my answers and I see, ‘Oh, why did I do that? This is clearly a mistake,’ then maybe I could tell you that, ‘Oh, I misclicked,’ or, ‘I don’t know what I was thinking at the moment,’ because if it’s like, yeah, strongly agree, agree, agree, and then one time strongly disagree, then something must have happened there.”

They said yes to wanting to see their ratings in comparison to others, but during an earlier think-aloud session, they said about the Paas scale results that it was “way too intimidating” to see other peoples’ rating of how much mental effort they invested in solving problems and “would want [it] to be focused on [their] own experience.”

6.3.4 Sophia

Sophia started the study on December 17th, 2021 and ended it on March 11th, 2022 (12 weeks).

6.3.4.1 Accessibility Barriers/Challenges and Benefits

When asked whether or not she encountered any accessibility barriers/challenges while using the interactive eBook to read and practice programming, Sophia raised issues related to note-taking and instructional design. Sophia said:

“I did come across a few words I didn’t know or wasn’t familiar with, so I had to take time to google them. Most concepts were easy to understand, and I really like the breakdown of each concept and how interactive it is. There were parts that I wish I could highlight, but I couldn’t log [into Hypothesis—the system Runestone using for annotation]. In the chapter titled ‘asking the user for input’, once the examples used multiple lines of code with no space in between them, my brain could not process each sentence. I think having some spaces between each lines for examples might help me understand even more. Again this was only with the birthyear example.” Week One - Variables, Expressions, and Statements

“I did not encounter any barriers, everything was extremely easy to follow and comprehend.” Week Two - Debugging

“Not particularly, but I do find it difficult to notice small details such as commas or colons. I feel like my eyes are not noticing it when a command turns a different color (this is obvious). I am also experiencing high fatigue when I do multiple practice problems, even when they are the same type of question.” Week Three - Conditional Execution

“I had a great deal of fatigue for the past two days. This chapter took me multiple days to finish, due to having to hand write notes (it is the only way I process) and needing to read everything out loud. Reading it out loud gives me two forms of ‘reading’ (hearing/reading visually). I find that I have to do this with longer or more complex passages. I also do this on Zoom or while watching tv (cc’d on). At certain parts, having to write out code becomes difficult, especially if there was no similar example. When I first started this book, I didn’t need too many similar examples, but today I felt very confused in the practice. In the reading, the math functions were difficult for me (I’ve failed trigonometry before and calculus and I was discouraged). I did not gain more confidence until there was discussion of random numbers. I found myself

wanting to see the words in the first chapter that encouraged me along the way (rarely do I see textbooks that say 'it's okay if you don't understand.' ” Week Four - Functions

“I think I am just encountering more and more fatigue on the practice questions. I understand what the questions are asking for to a point, but I noticed a lot more things weren't covered (in terms of Python reserved words). It makes it harder when I don't even know what to google. The more stress I am under, the more my disability hinders me.” Week Five - Loops and Iterations

“I believe I didn't encounter anything until I started getting to more complex topics like format operators. I understood it towards the end, but I felt there were little examples beyond the camels example. I have confidence in most of my answers in the practice, but I found myself checking each line more thoroughly to make sure I understood. I would like to have more examples in the practice that reflect the different aspects of the chapter because it seems like the same format of questions where we had to return a string. Topics like parsing strings, format operator, and methods can sometimes be confusing to me. I think for specific commands that are in the glossary it would be easy to go back if the glossary had a Python example for each definition.” Week Six - Strings

“While finishing these problems it made me think that if this was a class textbook, I would feel more confident with my notes (open book test). Unlike undergrad where I had to do statistics with no help, grad school was a lot more manageable due to open book tests. However, with the practice problems I could not solve any write-code problems. It felt like I did not have the prior knowledge of code that was needed. All of the reading sections were easy to understand, and I get the feeling that I understand lists in a more complex way. Maybe the problems could slowly increase in difficulty? Or is there the option for a help me similar to the mixed-code that would suggest which code to use. This is the most confused I have felt in all of the practices I completed.” Week Seven - Lists

“I didn't really encounter any issues with the reading for this day. Everything seemed to be straight forward and many of the references were related to me. I had to reread some of the reading to do the practice because I forgot everything and needed a stimulus to help me remember. The practice can be difficult when I have to build a loop that calls out a specific element (e.g., line in lines or word in dictionary). I feel like this would be easier if we saw the data or given a hint like some of the other eBooks.” Week Eight - Dictionaries

When asked if there's was any kind of help feature she would like to see to support memorization and concentration, Sophia said, "I think it was difficult because I couldn't highlight stuff. I've worked with PDFs and editing PDFs before. Usually it helps me at least to highlight it or write notes to myself....I really like the interactivity of the actual reading. I feel like I'm learning alongside the actual textbook rather than this being just something I have to read that I don't want to do....writing it makes me more anxious. I haven't had this disability before, but now I am learning how to deal with it. It's taking me even longer than it usually did to think about what I have to actually type. And if I get a red error message, I just feel frustrated....I'm frustrated with my disability....I'm frustrated that it's taking me longer rather than having instant gratification. But I also know that programming and coding was naturally difficult for me, and it's even more difficult now. I get migraines and sometimes it's not migraines, sometimes my brain just hurts physically. It's like two different types of frustration that I'm not dealing with properly."

When asked if it would be helpful if the eBook adapted the content to her as opposed to encountering subjects like trigonometry, which frustrated her, Sophia said, "I would say so. I mean, even the one about mileage reminded me of taxis in New York, so I was like, that makes sense. It helps me relate to it...why this is important and when this is being used. I really like it, I haven't called a cab in a minute, but...[laughter]."

Sophia also struggled with comprehending the error messages for write-code problems. When asked if she understood them, she said, "I've seen them before, it makes me frustrated to be honest, 'cause I'm think... 'I didn't do it right.' But then I realized that the question says this is the exact output. And the only way to test to see if it is working is to put a random number. When I saw the error, I thought it was wrong and then I thought about it and it makes sense. Then that's when I tried different numbers. And the program is doing what I'm telling it...that's what the question asked for." Sophia often had trouble with testing incomplete code, so she used **print** statements and CodeLens (see Figure 6.20). What she's referring to here is that fact that unit test run when the code is complete, but she wanted to see output for subsections of her code.

When asked about her short-term memory and if it would be helpful for the eBook to prompt her to review chapters after being away for a while, Sophia said, "Yes. In regards to my short-term memory, I have a really bad problem with it. I could give you an example [of some things that work for me]. I take Cantonese classes online. It's self-paced as well. You can mark down things that you want to review again."

She also thought it would be helpful if there was a feature that took ones notes and digitized them and also linked them to certain concepts in the eBook. Sophia said, "Yeah, because I have a written notebook that I can scan and that's how I made the PDF. The problem is, it doesn't have a transcription service. That would be a lot easier so I can control F and find exactly what it is I'm looking for. But it takes the literal notes and just makes that into a PDF. I can't really search for

things in the PDF... Yes, it would be helpful.”

When reflecting on the length of the chapters in the eBook, Sophia said, “I like when the texts are kind of spread out and slowly built up over time. Another thing I really like about it, actually, which is something I haven’t seen is that I think one or two of the questions have a Wikipedia or a Google link to it....it helped a lot because something that professors don’t see is that I struggle a lot with big words and just things that my colleagues who have a great educational background don’t struggle with, but I did not get that....to be honest, there’s not really much that I feel needs to be improved. If I really had to say anything, it’s just I would like to see more of the links.” She thought it would be helpful to break down the sections of the eBook and add more external links.

When asked about the experience of learning to outside the classroom and the flexibility of doing so, Sophia said, “I guess I feel more confident. It makes sense. Every time I go to my professor to ask how I run a program, or if it’s really just anything aside from SPSS...I can’t get any help. When I was engaging with R in my first year in grad school, I was so confused and nobody was helping me and nobody was...I don’t wanna say it, dumbing it down for me. I didn’t pay attention to statistics, I was looking at what new shoes I could get....because I wasn’t confident. I was just like, I’m just gonna look at this because this makes me feel better. I would say I feel more confident right now.” Later on she said, “I think if it wasn’t self-paced I would’ve been a lot more stressed out.”

Sophia reported that illness, family obligations, social/personal life issues, and self-doubt/lack of confidence moderately interfered with her ability to learn and complete the reading and practice assignments.

6.3.4.2 Computational Practices, Perspectives, and Attitudes

Sophia’s main computational practices included note-taking while reading, using her notes to solve programming problems, and using CodeLens to trace code (see Figure 6.20). In particular, when solving problem two during the last think-aloud session, Sophia used her notes on lists to help (see Appendix F.3).

I followed up with Sophia about any other practices she had and she said, “Aside from taking notes, sometimes I would look at my partners’ because he’s also coding in Python. I watched him because he’s taking an actual class and it seems very fun. There’s a little robot that you control to move....I also supplement the [glossary with Stack Overflow].” Sophia wanted more peer-support and interaction. She said, “I think that it would be fun. Sometimes when doing practice questions, I would ask my partner for his input.”

To manage her cognitive load, Sophia would put some problems off until later. Once she said, “I wasn’t really understanding how it was looping...I was really just trying to understand where I messed up, but it seemed like it didn’t even get past the first line of code. I couldn’t really

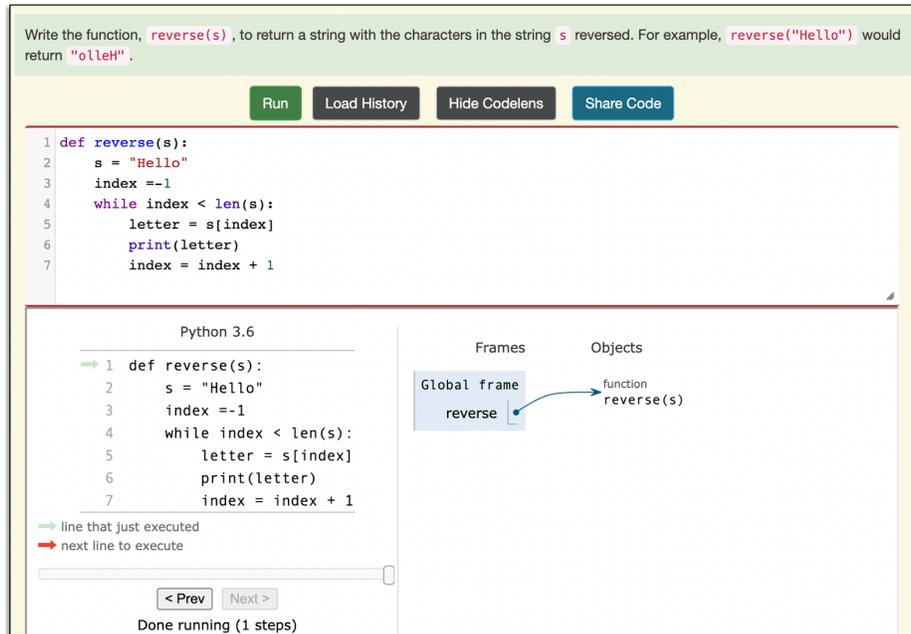


Figure 6.20: An example problem for which Sophia used her notes to solve.

figure out why or how to change it. I got a little bit frustrated....[so I thought] let me just go to something else, get less frustrated. Finish that and then come back.” She also consistently reported experiencing low extraneous load (see Table 6.14 and Figure 6.24), so I followed up to see what about the assignments was working well and Sophia said, “I like the glossary in terms of instruction....maybe having the definition and then an example so that my brain kind of associates [things there]. For the instructions, have there been points when I have not understood. Not really. I feel like it’s more the code itself in terms of reading code. I like to be able to understand....I want to be able to read it like I read the definitions and the reading itself.”

When asked about the practice of switching between problem types, Sophia said that seeing a Parsons problem solution similar to hers would “validate her understanding.” She chose not to copy the solution from the Parsons problem when solving problem two (see Figure 6.5). Sophia remarked, “I feel like I get confused because the [function] doesn’t start out with the word ‘Hello’. It starts out with an empty string or an idea of a string. I think I get confused when there’s no actual [parameter] being passed. That’s where I stopped. I didn’t see the [parameter] in [the solution].” Sophia often used **print** statements to run her incomplete code or used CodeLens to check her comprehension of write-code problems.

She preferred each problem type (Parsons and write-code problems) for different reasons. Sophia said, “I feel mixed [Parsons]...personally whenever I do the problems and I have to drag it, I like doing those more because I feel like it takes me a lot to think of what the code should be. I have to think of finer details, such as realizing that there’s [a colon] after this versus the [write-code

problems], so I feel like [Parsons problems] test me more [to see if] I’m really paying attention to what I’m writing [or if] I’m just trying to get the right answer. And rather than me just typing it, where I’m struggling to find the right answer, I feel like the one that we’re looking at right now [the Parsons problem] tests if I have actually learned the material.”

Sophia’s responses to the attitude survey are shown in Table 6.12 and Figure 6.21. When I followed up with her about whether she perceived intelligence as something that’s fixed or flexible, she said, “I would say that I view intelligence as always changing. It might even be the way that I learned a long time ago is different then the way I learn now....I think ways of learning are always changing and intelligence is always changing because we’re able to get things that are more accessible to people with disabilities.”

Table 6.12: Sophia’s Responses to the Attitude Survey

Factor	Week One	Week Two	Week Three	Week Four	Week Five	Week Six	Week Seven	Week Eight
(F1) Problem Solving - Fixed Mindset	3	3	3	3	3	3	3	3
(F2) Gender Equity	5	4	5	5	5	5	5	5
(F3) Importance	5	5	5	5	5	5	5	5
(F4) Problem Solving - Strategies	4	4	4	4	4	4	4	4
(F5) Gender Bias	2	2	3	3	3	2	2	2
(F6) Personal Interest	4	5	4	3	3	4	3	4

Note: Likert scale: 1 = Strongly disagree; 2 = Somewhat disagree; 3 = Neither agree nor disagree; 4 = Somewhat agree; 5 = Strongly agree.

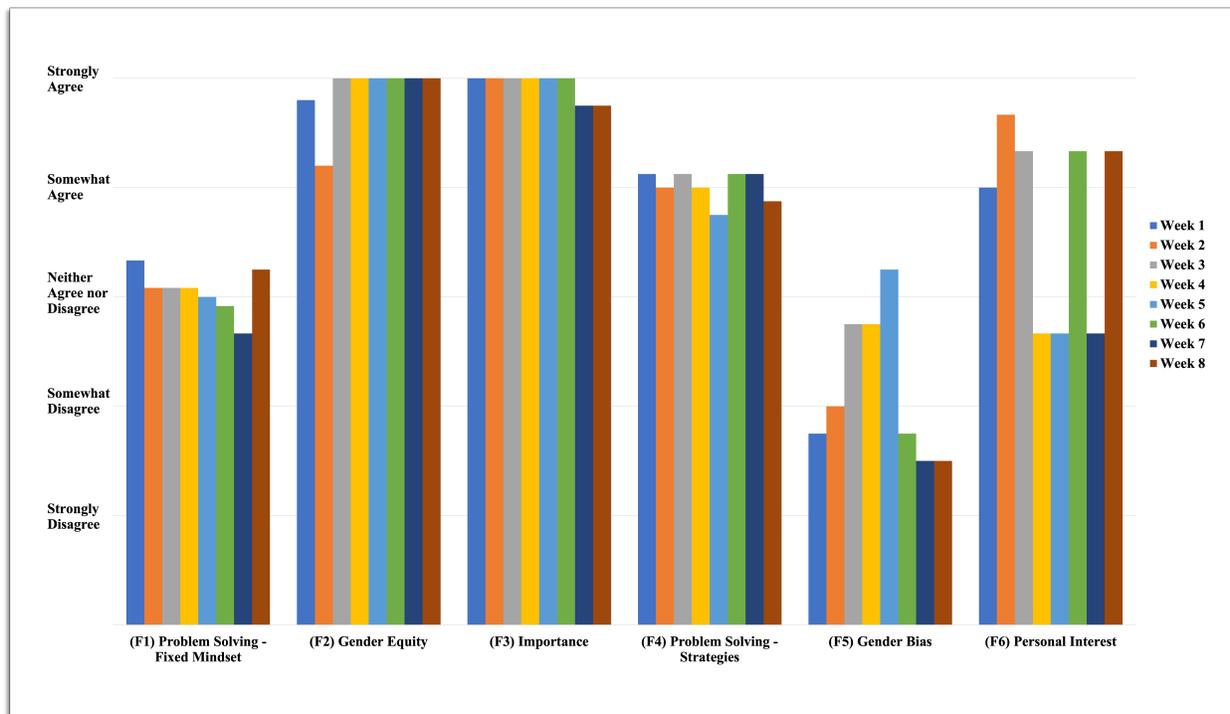


Figure 6.21: Sophia’s Responses to the Attitude Survey

When asked if she believed gender impacted a person's experience when learning how to program and why or why not, Sophia said, "Yeah, I mean, I originally wanted to go into computer science so I could develop video games, but I gave that up because I didn't feel confident in myself, and nobody really backed me. I mean, my mom was like, 'Go for it,' but at the end of the day, that was not my educational experience in high school and college. And even here, I'll go to a lab and if you go to people who are majoring in things related to programming or computer science, it's mostly male. Every time I see a woman, I'm just like, 'Good for you,' because I wanted to do it back then. It was pretty obvious that the teachers and professors that I had growing up did not believe in me as a woman...that I could do more than what was expected of me."

On another occasion, she said, "I would say [gender impacts learning]. At least for programming, the reason I'm so focused on a question is because in the back of my head I hear 'You don't know what you're doing' and that's constantly been reinforced in many of my math classes. In one of my classes in high school, the teacher would reward students with a dollar or two for being in the top three and the top three were always the same dudes. It was always three men and I don't think I ever saw myself or any of my female friends have that. So I thought 'Oh, I'm not good enough to get a high score and make money off of it.' In terms of sexuality? I am not too sure. I mean, I identify as pansexual, but I'm also not out. I know a lot of my friends who are out and being either put down or the professor or teacher won't interact with them as much. In terms of learning, if I am presenting as straight and I'm talking to an educator, they seem fine with me coming and asking questions. But I've seen somebody who was a queer man come in and the teacher just refused to interact with them."

When following up with her about her responses to the attitude survey, in particular, the statement, 'I would have more faith in the answer for a programming problem solved by a man than a woman.', Sophia said, "I had the fortune of having to decide when one was a woman and one was a man, and I thought because the man was friendlier that he would be able to explain things better. But when it came down to it, my female GSI broke it down for me and actually listened to what I was asking. So at least during those experiences, I've usually had a better understanding based on the women who have taught me, but maybe it's just because they're more understanding of the frustration that I've had. I don't really think particularly that there's really a difference in gender in terms of like intelligence and communication or even understanding. But I do understand why there's more men in programming than women. If that makes sense, I understand the stereotypes."

Sophia also felt like the eBook influenced her beliefs about gender equity and gender bias. She said, "One of the things was in the beginning or the first two chapters was, I think, talking about errors and the textbook literally said, 'This doesn't make sense, right?' And I thought, 'No, it does not.' I think it reflects thoughts that I have when I read, so if I am getting frustrated...it's probably that I'm still not getting the concept or the code that's being presented to me and how it fully works."

I guess it offered more humility than I've ever had.”

In general, she said she felt the eBook improved her perspective of computing. Sophia said, “I would say overall it’s improved my view of it, and more specifically my perception that people can actually understand, and conceptualize, and work with Python overall.”

Sophia’s responses to the self-efficacy questionnaire are shown in Table 6.13 and Figure 6.22. When asked about ways the reading and practice assignments might increase her self-efficacy. Sophia said, “There was a discussion tab [see Figure 6.23]. I thought that was interesting because when I opened it up, it led me to something like a statistics website [I’ve used] where people post a comment and a question and then other people answer...I saw that somebody asked the same question me. I like the discussion. I’ve always been too scared to post something, but it would be nice to talk with other students about things.” She said, “I particularly don’t want answers or just to see people’s answers. I want to see if somebody has asked similar questions. I want to make sure I comprehend things, not just copy and paste somebody else’s stuff.”

Table 6.13: Sophia’s Responses to the Self-Efficacy Questionnaire

Question	Week One	Week Two	Week Three	Week Four	Week Five	Week Six	Week Seven	Week Eight
1. I’m certain I can understand the most difficult material presented in the readings.	5	6	5	4	5	6	5	5
2. I’m confident I can understand the basic concepts taught in this eBook.	6	7	6	6	6	7	7	6
3. I’m confident I can understand the most complex material presented in this eBook.	5	6	6	5	3	6	5	6
4. I’m confident I can do an excellent job on the practice assignments in this eBook.	3	3	5	2	2	3	1	3
5. I’m certain I can master the skills being taught in this eBook.	5	6	6	5	5	7	5	7

Note: Likert scale: 0 = Not at all true of me; 2 = Untrue of me; 3 = Somewhat untrue of me; 4 = Neutral; 5 = Somewhat true of me; 6 = True of me; 7 = Very true of me.

6.3.4.3 Self-Assessment

Sophia thought the use and frequency of filling out the attitude survey helped with reflection. She said, “I think that attitude survey was good. I think in terms of helping me...the gender [items] helped me think about the past and make me realize how much negativity, as a woman, I’ve experienced. I think that might be helpful for some other people who don’t realize that they have certain biases....I think it’s helpful in terms of reflection.”

Sophia’s responses to the Computer Science Cognitive Load Component Survey (CS CLCS) are shown in Table 6.14 and Figure 6.24. When asked how she assessed how much effort she put into a problem, Sophia said, “I think I do it fine to be honest, I write whatever is the first... I’ll write what my thought process was. So if it’s like, Oh, I didn’t put much effort. Why is it that I

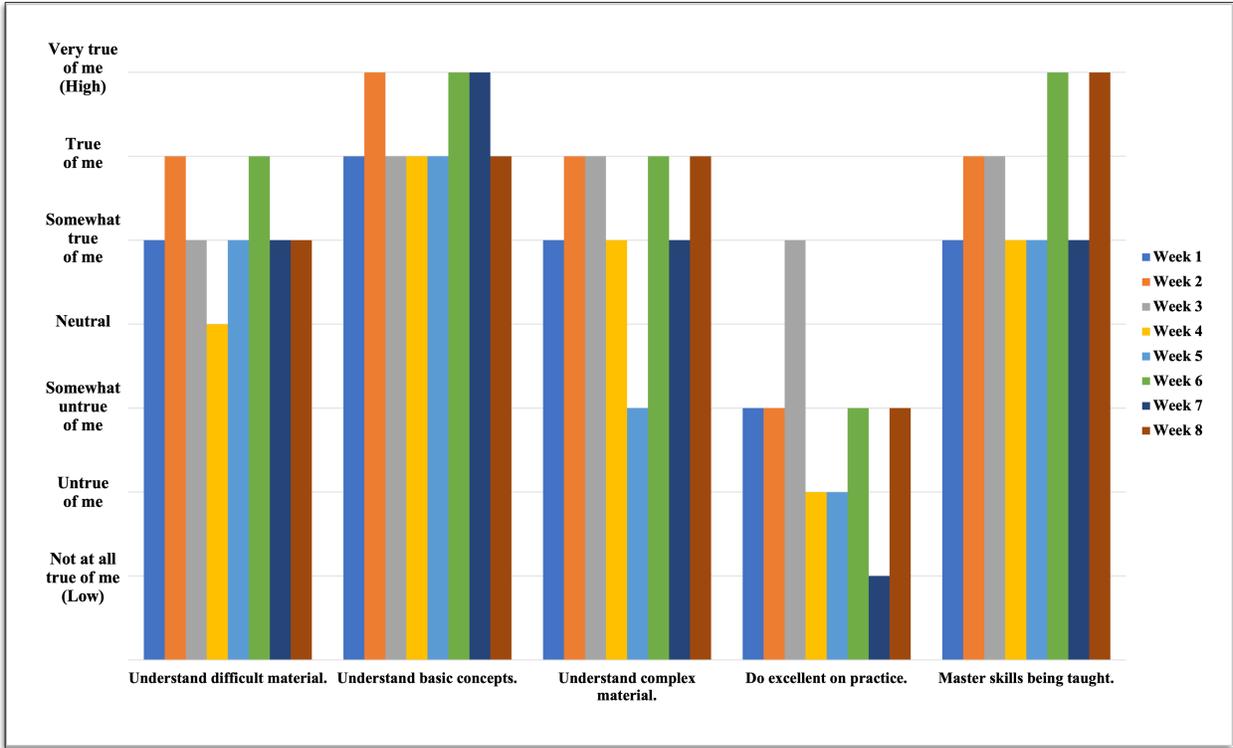


Figure 6.22: Sophia's Responses to the Self-Efficacy Questionnaire

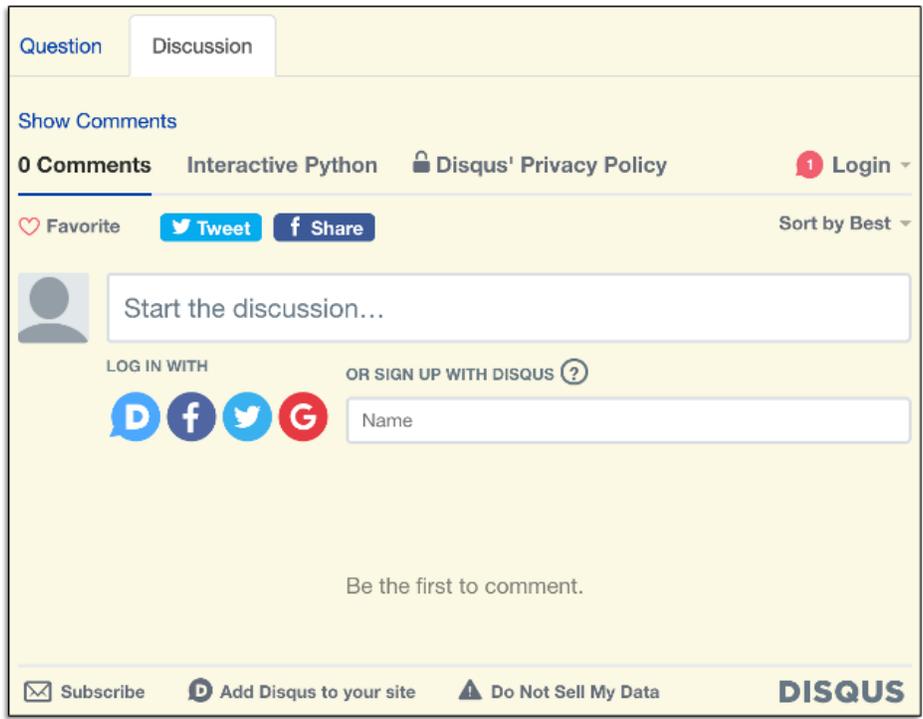


Figure 6.23: Runestone's Discussion Tab.

didn't put much effort or remember what the readings were, or I've seen this problem before, and that's kind of why it didn't take as much effort, it's more effort in like just remembering it rather than like once I remember it, it's like, Oh, I've done this before."

When asked how she described her ability to self-assess in general, Sophia said, "I would say I'm pretty good at it because sometimes after the practice, I feel, I guess down because it depends on how long it took me, but if it took me a while, then I probably feel worse than I usually would when I was reading, but reading, I don't think I've ever. While reading the eBook, I've never felt bad. It's usually once I get to practice questions. Once I get to the questionnaires afterwards, I'm actually thinking about them because it'll ask, 'Was this comprehensible?' And I think, 'Yeah, technically the eBook was, even if I didn't do that great on the practice.' Rather than just assessing how I felt right after the practice questions, I take a step back and think about the learning [process]. Rather than just solely the practice because if it was just about the practice, then I would feel horrible most of the time. But I actually feel really great doing reading." On another occasion she said, "...I think, 'Did it take me five minutes to figure this out, [and if yes,] then it was probably easy and I knew it.' Rather than thinking, 'This is actually a difficult question and I know this stuff, but it took me a while...I reassess constantly.' "

She also commented on seeing other people's responses to the Paas survey: "It feels weird to see other responses. I put low effort and that made sense to me. Then I saw it and was more difficult for other people. Sometimes in school, I feel if I'm not feeling the same things as other people then maybe I didn't do it right. But then I also think everybody learns at their own different speed. It did affect me, and I don't mind answering it. But I do second guess myself once I see other responses."

Table 6.14: Sophia's Responses to the Computer Science Cognitive Load Component Survey

Factor	Week One	Week Two	Week Three	Week Four	Week Five	Week Six	Week Seven	Week Eight
(F1) Intrinsic load	5	1	6	7	6	6	6	4
(F2) Extraneous load	0	0	2	3	4	2	0	0
(F3) Germane load	9	10	9	7	6	10	4	10

Note: Likert scale: 0 = Not at all to 10 = Completely the case.

When asked if she would find it beneficial to see a learning analytics dashboard based on the surveys, Sophia said, "I don't know. I feel two ways. Seeing a comparison to other people, it could make me feel worse or better. If I'm stuck, but then a bunch of people are stuck then I'm might think, 'Okay, I'm not the only one.' And then that lessens self-deprecation. I think for me personally, I would probably want maybe a progress [report] in the beginning, in the middle, and then in the end rather than overtime."

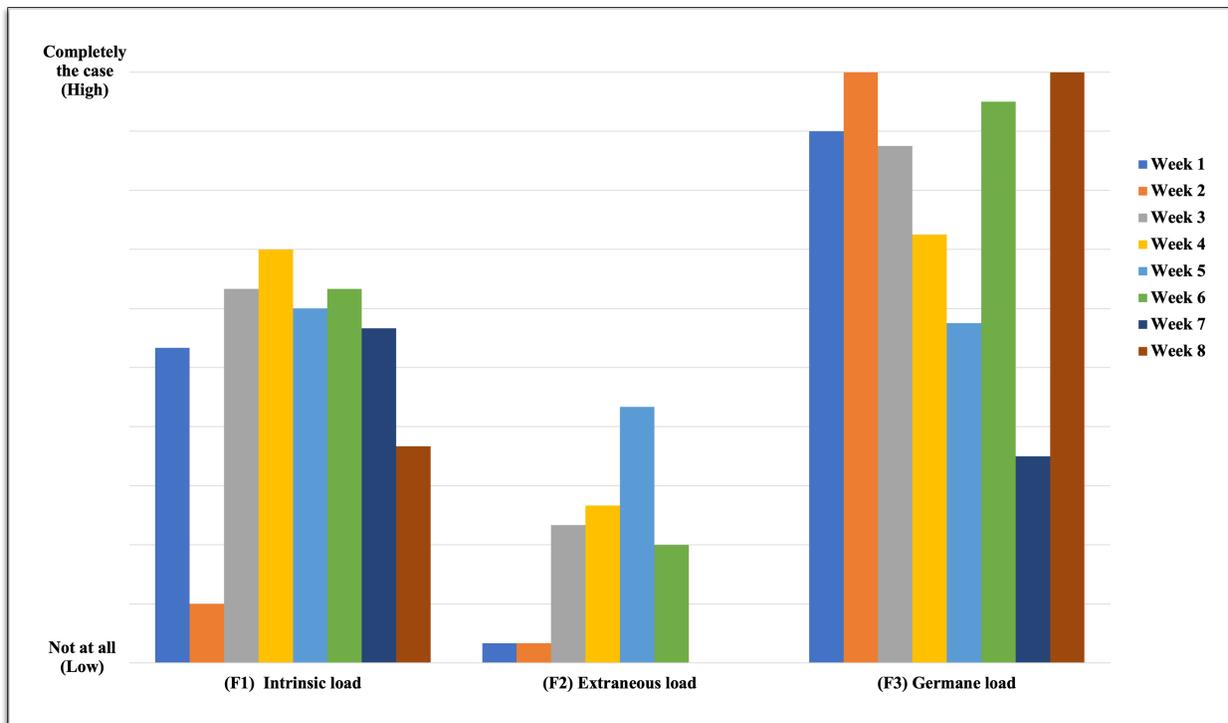


Figure 6.24: Sophia’s Responses to the Computer Science Cognitive Load Component Survey

6.4 Analysis, Discussion, and Future Work

The purpose of this study was to gain a better understanding of neurodiverse learners’ experience using an interactive eBook to learn how to program and to generate working hypotheses for future studies [421]. Specifically, I explored (1) whether they experienced any accessibility barriers/challenges and benefits when reading or solving Parsons and write-code problems, (2) their perceptions of the usability and usefulness of adaptive Parsons problems, (3) their computational practices, perspectives, and attitudes, and (4) their self-assessment processes.

6.4.1 Within-Subject Analysis

Within-subject analyses lead to several observations about accessibility barriers/challenges for each case with respect to the first research question: What are the accessibility barriers/challenges and benefits reported when neurodiverse learners use an interactive eBook to learn how to program?

6.4.1.1 Amanda

Amanda reported experiencing focal seizures when solving Parsons and write-code problems due to the presence of numbers. She mistakenly thought it was something that couldn't be controlled and attributed it to the pace with which she went through the material. While that may be a factor, this finding is consistent with previous research on the relationship between mental arithmetic and seizures [129, 335, 364, 410]. Ingvar and Nyman [168] termed this *epilepsia arithmetices* to describe a sort of reflex epilepsy in which mental arithmetic results in clinical seizures. Amanda also reported that her seizures come from her right temporal lobe which is consistent with research on number processing in the temporal lobe and epilepsy [78]. The main observation I made regarding Amanda's accessibility barriers/challenges was that she experienced seizures when solving programming problems with numeric calculations.

6.4.1.2 Claire

Claire reported that distractor blocks helped with the inattention she experienced while solving Parsons problems and improved her comprehension. "Inattention involves difficulties with keeping the learner's attention focused and to shift the focus of attention as necessary" [288, p. 75]. Learners with an attention-deficit/hyperactivity disorder (ADHD) experience an increase in distractibility [2]. Distractor blocks are extra statements that are not part of a correct Parsons problem solution. Whereas past researchers have found distractors increased extraneous cognitive load, but not intrinsic or germane cognitive load, the present study shows distractors improved focus and understanding (or germane cognitive load) for a learner with ADHD. This is a desirable difficulty [57]. Moreover, researchers who've investigated how learners with ADHD learn introductory computer programming concepts state one approach is to use completed examples [288] of which Parsons problems can be considered a variant (see [153, 373]). The main observation I made regarding Claire's accessibility barriers/challenges was that she experienced a negative correlation between germane cognitive load and the number of distractor blocks in a Parsons problem.

6.4.1.3 User

User reported that they watched videos of cats to regulate their emotions while learning. This relates to what researchers have found when investigating the impacts of using media computation to teach programming [346] and the need to support self-regulated learning (SRL) strategies such as emotional regulation in Computer Science Education (CSEd) [122]. Media Computation (or MediaComp) is media-focused pedagogical approach to teaching computing through the manipulation of pixels, for example, to create image effects (see Figure 6.25 [134, 135]). Sound values, movie frames and text can also be manipulated. Kinnunen et al. [188, 189] used a media computa-

tion approach to explore learners' emotional experiences with Java programming assignments and found some students with positive programming experiences still reflect negative views of their efficacy, while students with negative programming experiences reflect positively. Guzdial reported improvements in the success rates of underrepresented groups taking media computation courses [135]. Media Computation is an approach used to contextualize learning. Context and culture are critical aspects for learning. Learning scientists [263, p. 22] posit that "Because learning is influenced in fundamental ways by the context in which it takes place, schools and classrooms should be learner and community centered." Other related concepts and research agendas include interest-driven learning [11, 173] and culturally relevant computing [67]. The main observation I made regarding User's accessibility barriers/challenges was that if they were presented with media computation programming problems that involve modifying, editing, and/or creating pictures of contextualize and culturally relevant content, then they might experience more positive emotions.

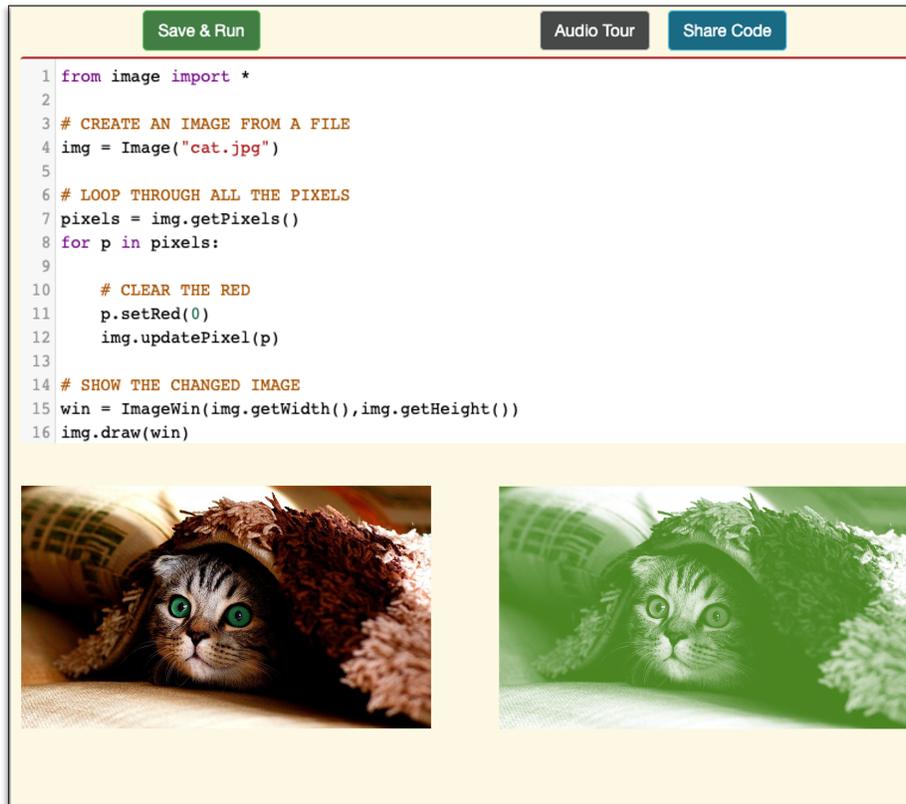


Figure 6.25: Example Media Computation Problem with Cat Image to Support Emotional Regulation

6.4.1.4 Sophia

Sophia reported that longer and/or more complex chapters in the eBook required note-taking and that she interacted better with text on paper than eBooks because of her memory impairment. She took copious notes (see Appendix F). This observation is consistent with research linking note-taking, memory, and comprehension [37] as well as the usefulness of note-taking and concept mapping in introductory computer science courses [178]. Note-taking is an active learning strategy that improves problem-solving performance [62, 384]. Sophia engaged in both *constructive* (free-form) and *active* (summarized/copy-and-pasted) note-taking [62]. The main observation I made regarding Sophia's accessibility barriers/challenges was that engaging in active learning strategies such as note-taking when learning how to program helped improve her problem-solving performance.

6.4.2 Cross-Case Synthesis

Cross-case synthesis revealed several key findings across and between all cases.

6.4.2.1 Accessibility Barriers/Challenges and Benefits

My research question was: What are the accessibility barriers/challenges and benefits reported when neurodiverse learners use an interactive eBook to learn how to program?

Participants reported that they encountered both accessibility barriers/challenges and benefits while using the eBook. One barrier/challenge for all of the participants was the length and structure of some of the reading and practice assignments. Amanda had trouble completing some of the reading and practice assignments in one sitting and gauging how much time it would take to complete them for someone with her disability. Claire expressed dread when faced with long chapters and said that it was demotivating. User said they needed to take multiple breaks and would prefer more practice interleaved throughout the reading as opposed to a separate assignment; User also found it hard to read some of the problem instructions because of how the instructions were written. They wanted problems that required math skills to supply more information similar to Edabit's notes section (see Figure 6.26) and more of a gradual introduction to new concepts. Sophia reported experiencing fatigue, wanting more encouragement and validation from the text, spending multiple days completing chapters such as the one on functions and wanting PDFs of the chapters to read offline. These findings can be interpreted in light of UDL principles as the participants wanting multiple means of engagement and multiple means of representation.

In contrast, all of the participants reported benefiting from readings in the eBook that chunked information into smaller chapters and sections. In particular, for week two on debugging, each participant reported it had the lowest intrinsic cognitive load (i.e., complexity of topics, program

Instructions Code Resources Solutions Comments

Body Mass Index

Published by [yanni](#) in [Python](#)

complete math strings

Body Mass Index (BMI) is found by taking your weight in kilograms and dividing by the square of your height in meters. The BMI categories are:

- Underweight: <18.5
- Normal weight: 18.5–24.9
- Overweight: 25–29.9
- Obesity: BMI of 30 or greater

Create a function that will accept weight and height (in kilos, pounds, meters, or inches) and return the BMI and the associated category. Round the BMI to nearest tenth.

Examples

```
BMI("205 pounds", "73 inches") -> "27.0 Overweight"
BMI("55 kilos", "1.65 meters") -> "20.2 Normal weight"
BMI("154 pounds", "2 meters") -> "17.5 Underweight"
```

Notes

- 1 inch = 0.0254 meter
- 1 pound = 0.453592 kilo

Figure 6.26: Edabit's Instructions Tab

code, concepts and definitions covered). Amanda remarked that her interest in computing went up because this reading was short and manageable. And Claire said that short chapters and sections help with inattention and retention. These results are consistent with the claim that chunking information accommodates the limits of our short-term memory and lowers cognitive load making it easier to encode new information [128, 348] especially when learning how to program [216]. Future studies should explore the organization of interactive eBooks for programming and how to manage the limitations of short-term memory and cognitive load for difficult topics. Also, the results imply that we should change the instructions to use bullet-pointed step-by-step instructions similar to programming practice websites like Edabit; this might help learners Amanda, Claire, and User. These findings can be interpreted in light of UDL principles as removing instructional barriers related to cognitive load.

The learners reported that one of the main benefits to learning outside of the classroom was the interactivity of the eBook and the flexibility to complete the reading and practice assignments at ones own pace. Claire, User, and Sophia associated the interactivity of the eBook with their high self-reported germane cognitive load. And Amanda, User, and Sophia expressed that the flexibility of completing the assignments was an accommodation that helped them manage their time, complete everything, and avoid stress. This is consistent with the literature on how to combat ableism in academia and support learners with disabilities [87]. Begel and Ko [21] call for researchers to investigate whether learning technologies should “structure learning for learners” or whether learners should “be taught how to structure their own independent learning?” [p. 763]. Future studies should explore the differences between self-regulated learning (SRL) in the classroom and self-directed learning (SDL) outside of the classroom [see 193] for students with and without disabilities in computing education. This has implications for curriculum design as there are some topics learners can readily learn on their own [see 241].

The results also imply that students with disabilities could use peer-support when learning how to program outside of school. User mentioned wanting a teacher and social interaction with other learners. Sophia said that seeing discussion posts similar to things she had questions about could increase her self-efficacy. Both of them had partners with programming knowledge that they received help from during the study. This is consistent with the literature on how creating an accessible and engaging learning experience for a broad range of learners through activities like pair programming [244, 403]. These findings can be interpreted in light of UDL principles as the participants wanting multiple means of action and expression.

6.4.2.2 Computational Practices, Perspectives, and Attitudes

My research question was: What are neurodiverse learners’ computational practices, perspectives, and attitudes? And what are their perceptions of the usability and usefulness of adaptive Parsons

problems versus write-code problems for learning how to program?

First, two of the participants had similar computational practices related to problem-solving and note-taking. Both Amanda and Sophia reported engaging in a problem-solving strategy of trial-and-error and expressed feeling like they were not learning when using this strategy. When asked what could be done to help, Amanda and User's responses implied that prompts to reread portions of the eBook and spaced repetition would help. This is consistent with the literature on how students with disabilities learn how they learn. Getzel and Thoma reported on what a focus group of postsecondary-level students with disabilities thought about problem-solving skills and self-awareness. Learners said about problem-solving skills and self-awareness that they needed time to organize and think about how to solve problems and how they learned with their disability [124]. Furthermore, spaced versus massed learning/practice can improve long-term retention [74, 148, 375]. Researchers argue that "the computer as a medium itself induces a more trial-and-error-like behavior as opposed to problem or goal-oriented behavior" [262, p. 1128]. Future research should investigate how we can create environments that support spaced learning and the development of SRL and SDL skills amongst students with disabilities to combat the use of learning by trial-and-error.

Both Claire and Sophia prefer to take handwritten notes that they then scanned to keep track of. Future research should explore how to make digitizing these notes more accessible to learners so they can merge them with eBooks.

Second, most participants self-reported that (1) having a growth mindset was important to their success, (2) negative beliefs about gender equity and gender bias in computing could be curtailed by knowledge of stereotype threat, lived experiences, and personal views about gender identity, and (3) that they were motivated to participate in the study and complete the assignments because of their "nothing about us without us" attitude [see 359]. Future research should explore how computing educational instruction and assessment materials can prompt a growth mindset and sense of belonging. Having a growth mindset can improve interest in computer science [45, 259] and sense of belonging is important for cultivating diversity in computing related to different social markers (i.e., race, gender, dis/ability, sexuality, etc.) [256, 363].

6.4.2.3 Usability and Usefulness of Adaptive Parsons Problems

My research question was: What are neurodiverse learners' perceptions of the usability and usefulness of adaptive Parsons problems versus write-code problems for learning how to program?

Most participants found the Parsons problems useful for learning but there were two issues. Both Amanda and Claire had trouble with indentation and Amanda had trouble with the combine blocks feature. They both wanted more feedback. Interestingly enough, neurotypical learners from a previous study (see Chapter 4) were confused when the "Help Me" button provided indentation

or combined blocks that were already adjacent. Both Amanda and Sophia found adaptive Parsons problems versus write-code problems useful for learning. Amanda used Parsons problems to help her solve write-code problems and Sophia remarked that Parsons problems tested attention to details and resulted in learning rather than struggling to solve write-code problems and not learning. When asked about the toggle feature, all of the participants viewed it as an accommodation—a memory aid. Sophia said that seeing a Parsons problem with a solution similar to hers would also validate her understanding. Amanda said she would never switch from a Parsons problem to a write-code problem unless she was motivated to solve it for personal reasons. In contrast, Claire said she would toggle from a Parsons problem to write-code problem if she did not recognize the solution to the Parsons problem. How do we get learners to challenge themselves to solve a write-code problem after solving an isomorphic Parsons problem? Future research should investigate how to improve learners engagement so that they take on intellectual challenges and engage in learning transfer; I plan to explore gameful learning in CSEd to increase student engagement in programming practice [113].

6.4.2.4 Self-Assessment

My research question was: What are neurodiverse learners’ perceptions about the surveys and/or questionnaires used in the study?

Self-assessment is defined as a “...wide variety of mechanisms and techniques through which students describe (i.e., assess) and possibly assign merit or worth to (i.e., evaluate) the qualities of their own learning processes and products” [275, p. 2]. Its accuracy [43] and impact on self-regulated learning and self-efficacy [130, 276] are of major concern to educational researchers because of its impact on cognitive learning gains [109, 352]. In this study, I explored participants’ perceptions about the surveys and/or questionnaires used in the study (i.e. assessment instruments) and learners’ self-assessment process. My findings highlight the (1) importance of developing nonbinary/genderqueer items, (2) impact of engaging in frequent self-assessments on learners with disabilities, and (3) concerns about learning analytics.

User and Sophia identified as being a part of the LGBTQIA+ community and expressed how “passé” some of the computer science attitude survey items were. Human-Computer Interaction researchers have explored how to develop gender sensitive surveys [46] and data about disabilities [33]. Future CSEd researchers should not only explore what we are asking computing education students about [423], but how we are asking them about gender identity [46] and their disabilities [91, 114] as well.

None of the participants reported negative reactions to engaging in frequent self-assessments; they expressed that it helped them to reflect and learn how to engage in realistic self-assessment. This is important for two reasons. One, CSEd researchers have posited that “By reducing the

frequency that students self-assess negatively while programming, we maybe able to improve self-efficacy and decrease dropout rates in CS” [130, p. 170]. And two, helping learners with and without disabilities develop self-assessment skills can help them improve how they engage in each phase of SRL [418] and contribute to the research on SRL strategies used while learning how to program online [122].

Most participants reported that their self-assessment process for determining the cognitive load of the readings and/or practice problems included retrospectively monitoring their level of engagement, time-on-task, any issues, frustrations, misconceptions, and comprehension. This is important for determining the accuracy of learners’ self-judgments and how they align with performance in order to provide feedback about the development of their self-assessment skills [275].

Finally, most participants reported that displaying learning analytics related to the self-assessment instruments would be harmful/intimidating and/or unhealthy if shown comparison information in contrast to their performance or beliefs. Surprisingly, most cases expressed they would like to see learning analytics about learners struggling with things similar to themselves. And the display of learning analytics was viewed as a way to validate answers to unclear questions and see how ones responses changed over time. This is important for developing learning analytics dashboards that do not reinforce inequities in education [412] for learners with disabilities.

6.4.3 Limitations

First, this study only examined four novice programmers each with different and sometimes overlapping disabilities; one can not assume these findings will generalize to other contexts. Replication of this study with a larger sample of learners who identify with each of the participants’ disabilities would further support each corresponding working hypothesis and whether these findings generalize across learners and contexts.

Second, participants in this study were not exposed to other eBooks and tools for learning how to program; future investigations into the effectiveness of the eBooks for learners with disabilities should compare their experiences with different eBooks.

Third, learners can simultaneously use interactive eBooks while attending massive open online courses (MOOCs) that support peer learning which may benefit learners with cognitive, learning, and/or neurological disabilities [198]. Future studies should consider a research design that involves more peer learning/instruction as it is a cost-effective way to increase learning gains [383, 427].

6.5 Conclusion

With the appropriate scaffolding and support, neurodiverse learners can excel at computer programming. This work contributed the first empirical multi-case study on how neurodiverse learners learn how to program with an interactive eBook with adaptive Parsons problems. Observations of the accessibility barriers/challenges that each case/participant had with the eBook and adaptive Parsons problems led to the generation of four working hypotheses about how to improve accessibility related to (1) mental arithmetic and learners with seizures disorders, (2) learners with ADHD and adaptive Parsons problems with distractor blocks, (3) learners with ADHD and/or mental health disabilities and interest-driven learning and culturally relevant computing, and (4) learners with memory impairments and active learning strategies such as note-taking. Future research should explore the generalizability of these observations with larger groups of learners who specifically identify with each case. Cross-case synthesis provided design recommendations in line with Universal Design for Learning (UDL) principles for computing education eBooks and adaptive Parsons problems. Participants also benefited from readings in the eBook that chunked information into smaller chapters and most found the Parsons problems useful for learning. This study also revealed issues and recommendations associated with self-report/self-assessment instruments used to understand learners' perceptions of computer programming instruction and assessment materials. And finally, it identified ways to meet the needs of learners with a board range of abilities and support the development of positive attitudes toward computing.

CHAPTER 7

Conclude

Traditional introductory computer programming practice includes code-tracing and code-writing. Code-tracing involves using paper and pencil to hand trace the execution of a program [202], and code-writing requires students to write-code from scratch. These types of problems are time-intensive and frustrating which decreases students' engagement and motivation [23, 344]—and not much has changed in the last decade [327]. In particular, programmers from historically under-represented groups in computing typically have less prior programming experience and are more vulnerable to being negatively affected by poorly designed learning environments where difficult programming tasks are introduced without appropriate support [175, 183, 220].

In contrast to traditional programming practice, mixed-up code (Parsons) problems require learners to place blocks of code in the correct order. Parsons problems are designed to challenge but not frustrate programmers. The idea is to keep learners in the Zone of Proximal Development (ZPD) [397]. Practice problems such as these can increase the diversity of programmers who successfully complete these courses by improving the efficiency and quality of knowledge acquisition. This is argued to be particularly beneficial for struggling students and students with disabilities “who typically require more time to master important information” [161, p. 2].

This dissertation envisions a future in which both neurotypical and neurodiverse novice programmers can actively engage in solving computer programming problems that are both adaptable and adaptive. Adaptability refers to the ability of the learner to control the system to, for example, switch between the type (i.e., write-code problems or adaptive Parsons problems) and difficulty of problems based on information or rather student-facing learning analytics they receive from the system [414]. Adaptivity refers to the system changing, for example, (1) the difficulty of problems based on different measures (performance, problem-solving efficiency, cognitive load, self-efficacy, etc.) [414] or (2) the solution of a Parsons problem based on maintaining desirable difficulties and facilitating learning transfer.

7.1 Contributions

This research makes contributions to several areas. First, research on active learning pedagogical techniques for computing education. Second, research on problem-solving efficiency and cognitive load for explicit learning of common and uncommon solutions to programming problems. Third, research on adaptive learning features for scaffolding computer programming skills. And fourth, research on how students with cognitive, learning, and/or neurological disabilities learn to program.

- *Parsons problems as active learning lecture activities.* Most novice programmers like solving adaptive Parsons problems as active learning (lecture) activities. They find them helpful for learning, but some students would rather write the code themselves. Parsons problems are a type of scaffolding which should fade as students develop expertise. We added the ability to switch (toggle) to the equivalent write code problem when given a Parsons problem.
- *Problem-solving efficiency and cognitive load of adaptive Parsons vs. write-code problems.* Novice programmers were significantly more efficient at solving Parsons problems than writing the equivalent code, but not when the solution to the Parsons problem was uncommon.
- *The impact of solving Parsons problems with (un)common solutions.* Novice programmers were significantly more efficient at solving a Parsons problem created with the most common student written solution. They were significantly less efficient at solving a Parsons problem with an uncommon solution than solving the equivalent write-code problem. When first presented with a Parsons problem that had an uncommon solution, learners tended to use that solution to solve the equivalent write-code problem.
- *How neurodiverse students learn to program outside of the classroom.* The observations from this study led to several working hypotheses for future research. First, learners with seizure disorders will experience a positive correlation between seizures and solving programming problems with numeric calculations. Second, learners with Attention Deficit Hyperactivity Disorders (ADHD) will experience a negative correlation between germane cognitive load and the number of distractor blocks in a Parsons problem. Third, if learners with ADHD and/or mental health disabilities are presented with media computation programming problems that involve modifying, editing, and/or creating pictures of contextualize and culturally relevant content, then they will experience more positive emotions. Fourth, if learners with memory impairments engage in active learning strategies such as note-taking when learning how to program then their problem-solving performance will improve. A cross-case analysis also revealed several benefits to learners. Participants reported benefiting from readings in the eBook that chunked information into smaller portions because it

increased their interest and lowered their cognitive load. The flexibility of learning via the eBook was a major accommodation and reason for completing the study. Participants also remarked that adaptive Parsons problems were useful for learning because, for example, Parsons problems caused them to focus on demonstrating what they knew instead of struggling to write code from scratch.

7.2 Future Work

This dissertation has provided evidence that adaptive Parsons problems can improve problem-solving efficiency, lower cognitive load, impact learning transfer, and that neurotypical neurodiverse learners find them helpful for learning. It has also led me to start codeveloping a computer programming practice environment called Codespec (see Figure 7.1 and Appendix K).

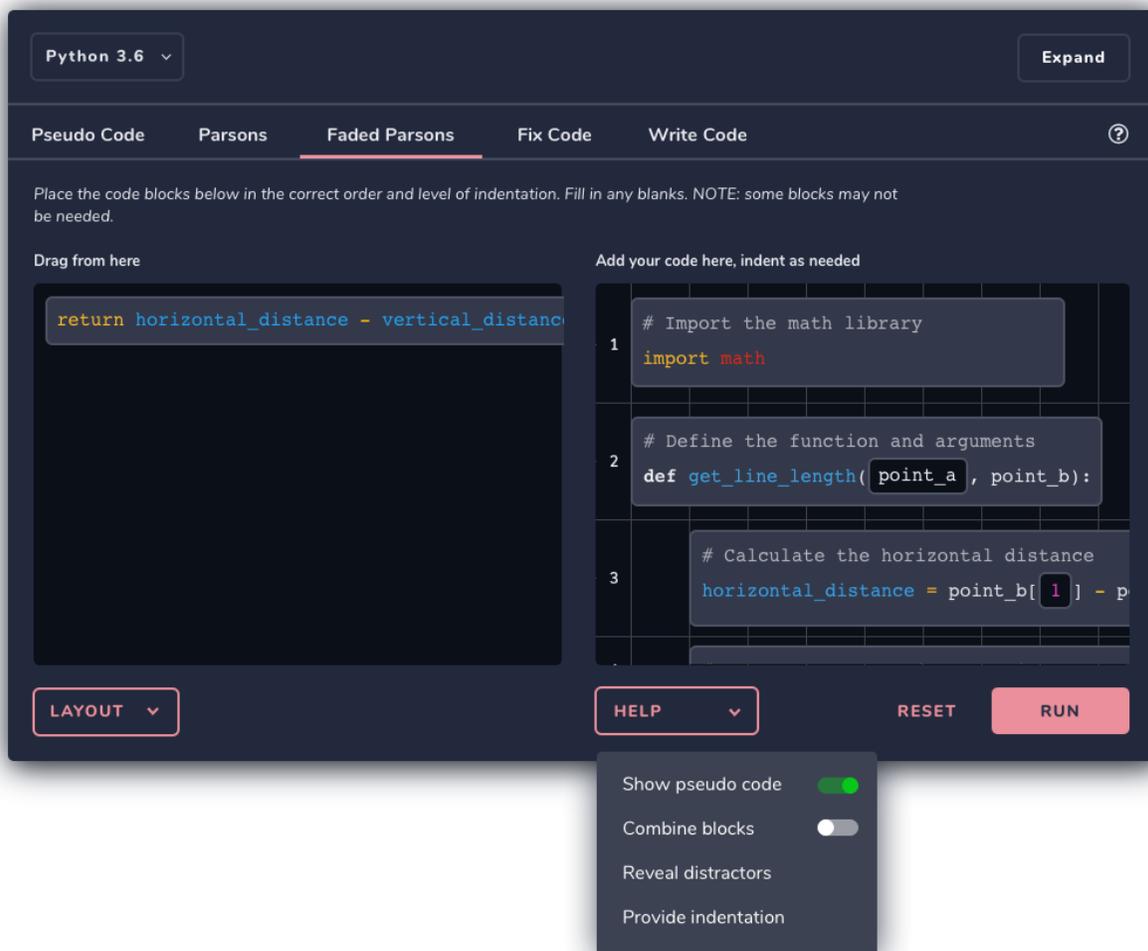


Figure 7.1: Faded Parsons Problem with Pseudocode Feature

Learning how to program requires the development of at least four skills: code reading and tracing, code writing, pattern comprehension, and pattern application [see 416]. Developing these skills requires deliberate practice which involves engaging in tasks specifically designed to improve ones' skills in a particular domain [106]. Web-based interactive programming practice environments such as CloudCoder [278], CodeChef [84], CodeLab [18], CodeWars [86], CodingBat[282], Edabit [230], and LeetCode [213] can be considered informal and/or open-ended learning environments (OELEs) that support deliberate practice—the learner decides what to learn and when [21, 142, 279]. OELEs support self-directed learning [193, 210] of computing topics by inspiring programmers to complete projects that are meaningful to them [242], by helping programmers build self-confidence, and by increasing programmers' sense of control over their learning process [39]. But the absence of a predefined curriculum and/or scaffolding, which are tenets of OELEs [142, 210], has led to such environments being criticized for failing to help learners grasp basic computer programming concepts [28, 238]. Learners may form inadequate mental models of basic programming concepts [205], struggle to find good resources, and struggle to get help from real experts when learning in these environments [39]. Hence, Begel and Ko [21], in their chapter on informal learning, call for researchers to investigate whether learning technologies should “structure learning for learners” or whether learners should “be taught how to structure their own independent learning” outside of the classroom [p. 763].

There are several types of computer programming practice problems to consider when developing these learning technologies. Parsons problems require learners to place blocks of code in the correct order. Faded Parsons problems are variant of Parsons problems that require learners to fill the blanks that are inside blocks with valid expressions and put the blocks in the correct order [405]. Parsons problems provide scaffolding for struggling learners; they help learners solve programming problems significantly more efficiently than writing the equivalent code [153] and Faded Parsons problems vs write-code problems significantly increase pattern acquisition [405]. However, scaffolding should fade as novice programmers develop expertise to keep them in the Zone of Proximal Development (ZPD)—a critical space between what learners can do independently and what they can do with assistance [397]. Prior research provides evidence that advanced learners want harder programming problems while novice learners struggle with the exact same problems [101, 104, 102]. Thus, we take a ‘software-realized scaffolding’ approach [133] to design a ‘gentle-slope system’ [293] that guides learners from drag-and-drop block-based programming practice to writing code from scratch.

7.2.1 Provide meaningful choices for computer programming practice.

The within-subjects experiment and think-aloud in chapter four showed some students had strong negative reactions to Parsons problems; these students also had more prior programming experience [153]. Consequently, I conceptualized a feature for toggling between code-writing problems and Parsons problems to increase the ability of the Runestone platform to scaffold learning for students with more prior programming knowledge (see Figure 7.2). This platform serves free computing education eBooks to over 25,000 people—increasing access to those who cannot afford expensive textbooks [103]. To my knowledge, BlockPy [19] and Runestone [153] are the only environments that offer programmers the option to solve the same problem as a Parsons problem or write-code problem, and Crescendo aims to support scaffolding for independent learning [399].

Codespec will provide learners with the opportunity to solve different types of programming problems along a gentle slope [see 312]. Its problem space area offers learners the option to switch between solving a problem as either a pseudocode problem (also described as subgoals and programming plans) [254, 72], Parsons problem [283, 98], Faded Parsons problem [405], fix-code problem [102], or write-code problem. Currently, there is no environment in which students can switch between each of these types of problems. The goal is to provide learners with flexibility to demonstrate what they know and curtail the criticism that some types of computer programming practice “do not give students the opportunity to show which concepts they understand and which they do not” [227, p. 14].

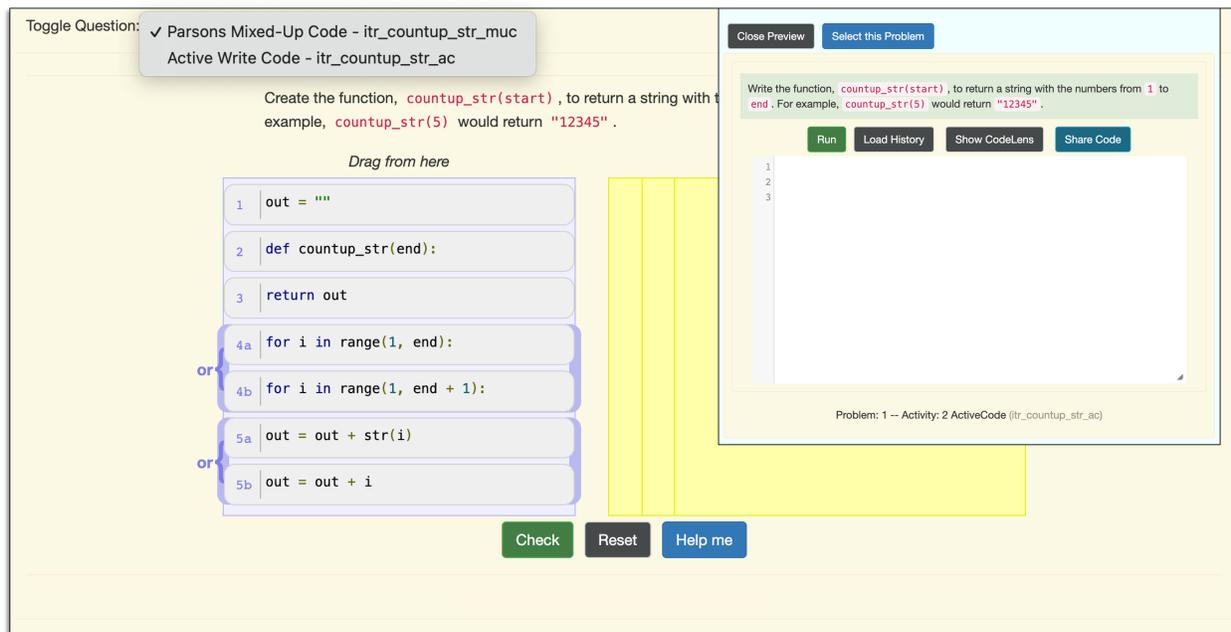


Figure 7.2: Runestone’s Toggle Question Feature

The research study in chapter four also provided evidence that to be more efficient than writing the equivalent code, Parsons problems should be generated from the most common student written code. I also clustered students' solutions to code-writing problems using OverCode [127] to explore relationships between efficiency, cognitive load ratings, programming solution use and acquisition (i.e., commonality of solution), and self-efficacy. Preliminary exploratory results showed it is possible to input this data into a sorting algorithm and produce a scale I can use to scaffold learners' introduction to different solutions. My goal is to create a machine learning algorithm that can automatically generate practice problems tailored to each individual learners' needs and provide students with multimodal learning analytics [34] based on this and other types of data such as eye-tracking.

7.2.2 Explore human-AI symbiosis and cognition.

Upon asking participants if they would like some type of help for write-code problems, one person said that they would like to see “a ‘Help Me’ button that either highlights the error or provides a hint” and another said, “Maybe if it gave you a hint. A piece of code that you’re missing...”

AI paired programming may be a solution that can expand our ability to capture which concepts students understand and which they do not while increasing their ability to write code from scratch. Leading HCI researchers posit that, “as machines acquire capabilities to learn deeply and actively from data, fundamental changes can occur to what humans are expected to learn [362]. And it is hypothesized that programmers need explicit and incremental instruction to develop at least four basic skills to prepare for computer programming jobs: code reading and tracing, code writing, pattern comprehension, and pattern application [416]. Yet AI paired programming suggests programmers will need to focus on code reading and tracing to debug programs—review code and decide “what is good and what is not” [246]. Researchers should explore how learning with artificial intelligence affects what needs to be learned and the cognitive processes required to learn through field experiments with mixed methods.

Artificial intelligence technologies such as Copilot and Codex suggest next lines of code and write complete computer programs [284]. And currently, there is no environment in which students can both switch between different types of programming problems and receive both code predictions and suggested changes. Codespec's prototype features a “Help” menu with the options to “Predict Code”, “Show Pseudocode”, “Suggest Changes”, “Reveal Distractors”, “Combine Blocks”, and “Provide Indentation”. Code prediction is a feature available to professional programmers (for example GitHub Copilot [284]) that, to our knowledge, has not been added to learning environments dedicated to computer programming. And currently, there is no environment in which students can both switch between different types of programming problems and

receive both code predictions and suggested changes. Furthermore, evidence shows programming plans (pseudocode) and purpose-first scaffolds can motivate novice programmers to learn programming [see 72]. And, finally, research shows learners may benefit from more information about distractor blocks [153]. The prototype also features an option to toggle between ‘Dark Mode’ and ‘Light Mode’ and guidelines to increase the accessibility and adaptability of the problem space area [150, 197, 414].

7.2.3 Increase the representation of underrepresented groups in computing.

I will continue to explore how we can increase the presence of underrepresented learners in computing by designing for sense of belonging [256, 363] that supports people based on their social markers (i.e., race/ethnicity, gender, dis/ability, sexuality, class, etc.). Much of the research on learning design is focused on neurotypical learners. Based on the multi-case study in chapter six, researchers should explore the four working hypothesis and their generalizability to understand how to improve the accessibility adaptive Parson problems and equivalent write-code problems for neurodiverse learners.

APPENDIX A

Average Task Completion Times and Standard Deviations

A.1 Average Task Completion Times for Parsons to Write

Table A.1: Task Completion Times for Parsons to Write

Problem (Diff.)	<i>n</i>	Parsons Problem	Write-Code Problem
		<i>M (SD)</i> in seconds	<i>M (SD)</i> in seconds
1 has22 (H) [≡]	57	73.51 (72.95)	194.14 (297.57)
2 countInRange (M) [≡]	29	137.14 (72.02)	93.31 (72.34)
3 diffMaxMin (E) [≡]	59	14.58 (12.53)	76.61 (158.41)
4 dictTotal (M) [≡]	30	23.50 (6.50)	49.50 (50.88)
5 dictNames (H) [≡]	50	117.32 (119.67)	350.18 (481.65)

Notes: E = Easy, M = Medium, H = Hard; The equivalent symbol [≡] indicates that students who solved the adaptive Parsons problem first used the same solution to solve the equivalent write-code problem.

A.2 Average Task Completion Times for Write to Parsons

Table A.2: Average Task Completion Times for Write to Parsons

Problem (Diff.)	<i>n</i>	Parsons Problem	Write-Code Problem
		<i>M</i> (<i>SD</i>) in seconds	<i>M</i> (<i>SD</i>) in seconds
1 has22 (H) [≡]	30	40.47 (18.93)	379.47 (465.11)
2 countInRange (M) [≡]	61	110.46 (71.80)	257.20 (383.89)
3 diffMaxMin (E) [≡]	30	11.40 (7.75)	215.07 (203.98)
4 dictTotal (M) [≡]	62	29.11 (24.79)	97.44 (166.43)
5 dictNames (H) [≡]	26	58.69 (25.89)	476.77 (455.31)

Notes: E = Easy, M = Medium, H = Hard; The equivalent symbol [≡] indicates that students who solved the adaptive Parsons problem first used the same solution to solve the equivalent write-code problem.

APPENDIX B

Problem Set with Instructions and Solutions

B.1 Problem One

Finish the function `has22` below to return `True` if there are at least two items in the list `nums` that are adjacent and both equal to 2, otherwise return `False`. For example, return `True` for `has22([1, 2, 2])` since there are two adjacent items equal to 2 (at index 1 and 2) and `False` for `has22([2, 1, 2])` since the 2's are not adjacent.

The adaptive Parsons problem solution presented to students and the most common student written solution:

```
def has22(nums):  
    for i in range(len(nums)-1):  
        if nums[i] == 2 and nums[i+1] == 2:  
            return True  
    return False
```

B.2 Problem Two

Finish the function to define `countInRange` that returns a count of the number of times that a target value appears in a list between the start and end indices (inclusive). For example, `countInRange(1,2,4,[1, 2, 1, 1, 1, 1])` should return 3 since there are three 1's between index 2 and 4 inclusive.

The adaptive Parsons problem solution presented to students:

```
def countInRange(target, start, end, numList):
    count = 0
    for index in range(start, end+1):
        current = numList[index]
        if current == target:
            count = count+1
    return count
```

Most common student written solution for participants who solved this problem as a write-code problem first:

```
def countInRange(target, start, end, numList):
    count = 0
    for i in range(start, end+1):
        if numList[i] == target:
            count += 1
    return count
```

OverCode clusters of solutions for students who solved this problem as an adaptive Parsons problem first (see Table 5.4):

Cluster 1 (n = 7):

```
def countInRange(target, start, end, numList):
    count = 0
    for i in range(start, end+1):
        current = numList[i]
        if current == target:
            count += 1
    return count
```

Cluster 2 (n = 6):

```

def countInRange(target , start , end , numList):
    count = 0
    for i in range(start , end+1):
        if numList[i] == target:*
            count +=1
    return count

```

Cluster 3 (n = 6):

```

def countInRange(target , start , end , numList):
    count = 0
    for current in numList[start:end+1]:*
        if current == target:
            count += 1
    return count

```

Cluster 4 (n = 5):

```

def countInRange(target , start , end , numList):
    count = 0
    for i in range(start , end +1):
        current = numList[i]
        if current == target:
            count = count + 1*
    return count

```

Cluster 5 (n = 2):

```

def countInRange(target , start , end , numList):
    return numList[start:end+1].count(target)*

```

Cluster 6 (n = 2):

```

def countInRange(target , start , end , numList):
    count = 0
    for i in range(len(numList)):*
        if i>= start and i<= end:*
            if numList[i] == target:*
                count += 1
    return (count)

```

B.3 Problem Three

Finish the function `diffMaxMin` to return the difference between the largest and smallest value in the passed list of numbers (`nums`). For example, `diffMaxMin([1,2,3])` should return 2 since the difference between 3 and 1 is 2.

The adaptive Parsons problem solution presented to students and the most common student written solution:

```
def diffMaxMin(nums):  
    large = max(nums)  
    small = min(nums)  
    return large - small
```

B.4 Problem Four

Finish the function `total_values` that takes a dictionary (`dict`) and returns the total of the values in the dictionary. For example, `total_values('red': 3, 'blue': 2, 'green': 20)` would return 25.

The adaptive Parsons problem solution presented to students and the most common student written solution:

```
def total_values(dict):  
    total = 0  
    for key in dict:  
        total += dict[key]  
    return total
```

B.5 Problem Five

Finish the function `get_names` that takes a list of dictionaries and returns a list of strings with the names from the dictionaries. The key for the first name is 'first' and the key for the last name is 'last'. Return a list of the full names (first last) as a string. If the 'first' or 'last' key is missing in the dictionary use 'Unknown'. For example, `['first': 'Ann', 'last': 'Brown', 'first': 'Darius']` should return `['Ann Brown', 'Darius Unknown']`.

The adaptive Parsons problem solution presented to students and the most common student written solution:

```
def get_names(list_of_dict):
    name_list = []
    for p_dict in list_of_dict:
        first = p_dict.get('first', 'Unknown')
        last = p_dict.get('last', 'Unknown')
        name = first + "_" + last
        name_list.append(name)
    return name_list
```

APPENDIX C

Paas Scale

In solving the proceeding problem, I invested: (Choose one of the following ratings.)

1. Very, very low mental effort
2. Very low mental effort
3. Low mental effort
4. Rather low mental effort
5. Neither low nor high mental effort
6. Rather high mental effort
7. High mental effort
8. Very high mental effort
9. Very, very high mental effort

APPENDIX D

Prior Programming Experience Survey

Overall Experience

1. How much experience (i.e., months, years) have you had programming?

High School Experience

2. Did you learn to program as part of a high school class (either formal or informal)?
3. If yes, how many semesters did you take courses involving programming?
4. Was this programming experience part of another course (e.g. math, business, science...)?

College Experience

5. Did you learn to program as part of a college course?
6. If yes, how many semesters did you take courses involving programming?
7. Was this programming experience part of another course (e.g. math, business, science...)?

Work Experience

8. Did you program for work or an internship?
9. If yes, was your work programming experience full time, part time, or less than part time?
10. Roughly how long did you have this work and/or internship programming experience?

APPENDIX E

Claire's Handwritten Notes

E.1 Chapter Five: Functions

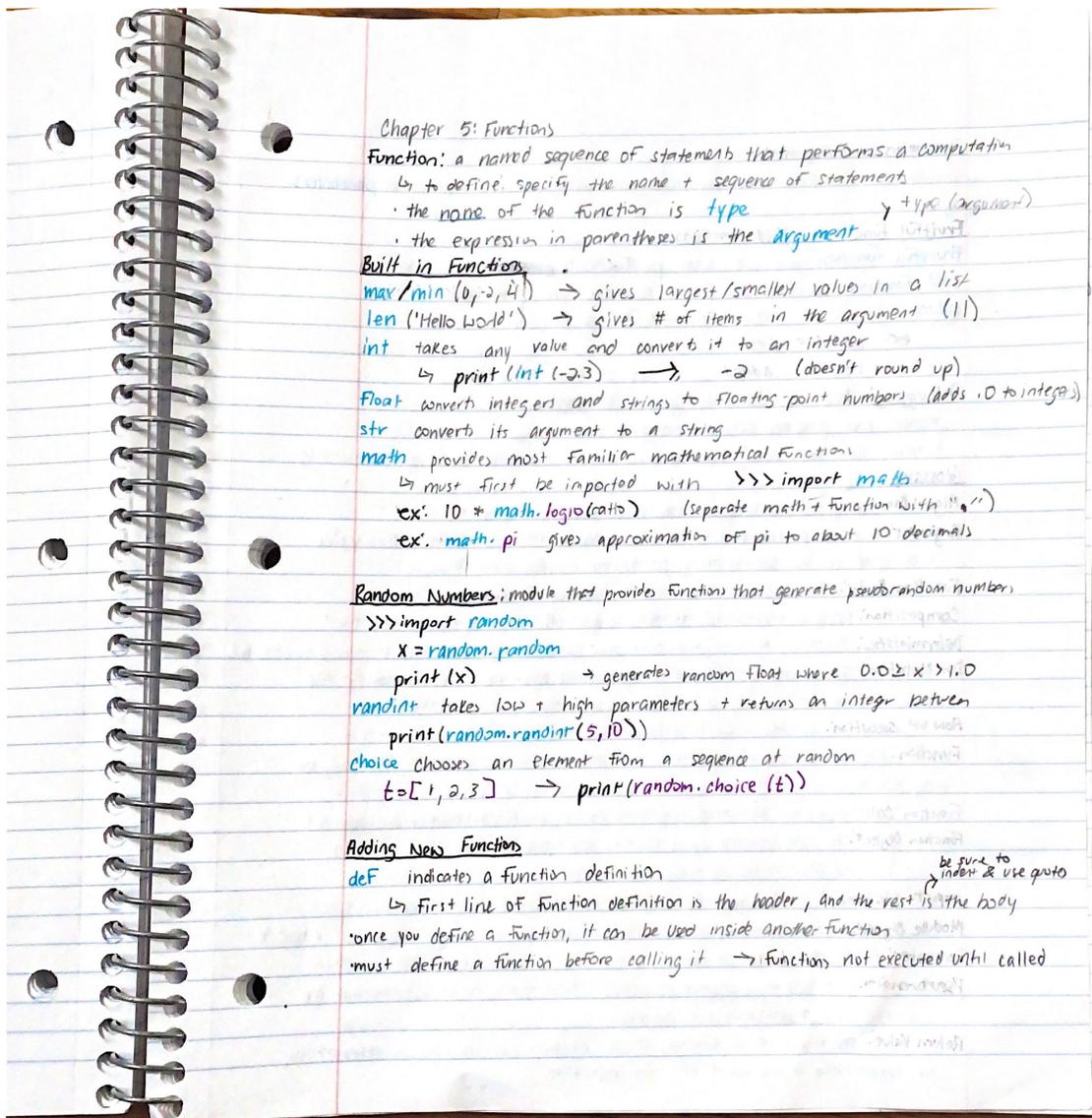


Figure E.1: Claire's Handwritten Notes on Functions - for Chp. Five - Page One

Parameters and Arguments

Inside the function, the arguments are assigned to values called **parameters**

Fruitful Functions + Void Functions

Fruitful Functions are ones that yield results (like a math function)

void functions perform an action but don't return a value

return is used to return a result from a function

ex: `added = 3 + 5`

`return added` ← (8 is then stored)

Debugging: best to use only spaces (no tabs)

* save your program before running it

* make sure the code you're looking at is the same code you're running

Glossary

Algorithm: a general process for solving a category of problems

Argument: A value provided to a function when the function is called. This value is assigned to the corresponding parameter in the function.

Function Body: The sequence of statements inside a function definition

Composition: Using an expression/statement as part of a larger expression/statement

Deterministic: Pertaining to a program that does the same thing each time it runs given same input

Dot Notation: The syntax for calling a function in another module by specifying the module name followed by a dot (period) and the function name

Flow of Execution: the order in which statements are executed during a program run

Function: A named sequence of statements that performs a useful operation. They may or may not take arguments & may or may not return a value

Function Call: A statement that executes a function. Consists of function name + argument list

Function Object: A value created by a function definition. The name of a function is a variable that refers to a function object

Import Statement: a statement that reads a module file & creates a module object

Module Object: a value created by an import statement that provides access to data/code in a module

Parameter: A name used inside a function to refer to the value passed as an argument

Pseudorandom: Pertaining to a sequence of numbers that appear to be random but are generated by a deterministic program

Return Value: The result of a function. If a function call is used as an expression, the return value is the value of the expression

Figure E.2: Claire's Handwritten Notes on Functions - for Chp. Five - Page Two

Chapter 6: Loops and Iterations

While: evaluate the condition, yielding True or False

2. if the condition is false, exit the while statement + continue execution @ now

3. if the condition is true, execute the body + go back to step 1

• if in an **infinite loop**: use **break** to jump out

↳ infinite loop means the while statement is always true

• this way of writing commands to "stop when this happens" (not "go until that happens")

Continue: used to skip the next iteration w/o finishing the body of the loop for the current iteration

construct a **definite loop** using **for**

• **for** loops through a known set of items, so it runs through as many iterations as there are items in the set

• syntax still needs a **for statement** and a **loop body**

↳ variable is a list → "for" loop goes through the list

```
names = ('name', 'name', 'name')
```

```
for name in names:
```

```
    print("Hello,", name)
```

```
print("All done!")
```

accumulator: a variable that counts the sum of elements in a loop

largest: the largest value we have seen so far

↳ if set to "None": the value is marked as empty

Glossary:

Accumulator: a variable used in a loop to add or accumulate a result

Loop Counter: a variable used in a loop to count the # of times something happened.

Initialized to 0 → then increment the counter each time we want to "count" something

Decrement: to decrease the value of a variable (often by 1)

Initialize: to give an initial value to a variable that will be updated

Increment: to increase the value of a variable (often by 1)

Infinite Loop: a loop in which the terminating condition is never satisfied or doesn't exist

Iteration: one repeated execution of a set of statements using either a function that calls itself or using a loop

Figure E.3: Claire's Handwritten Notes on Functions - Page Three

Chapter 7: Strings

String: a sequence of characters that can be accessed one at a time w/ the bracket operator

```
>>> fruit = 'banana'
>>> letter = fruit[1]
```

↳ the second statement extracts the character at index position 1 from the fruit variable + assign it to the letter variable (starts with 0 (so 'I would' = a))

the expression in brackets is called the **index** (must be an integer)

len() is a built-in function that returns the # of characters in a string

↳ negative indices count backwards from the end of a string

transversal: computations that involve processing a string 1 character at a time

slice: a segment of a string (selecting slice is similar to selecting a character)

```
s = 'Hello'
```

```
print(s[0:2]) → this will print 'He'
```

- [:] would select the entire string

- include 1st index but excludes the last

- strings are **immutable**, meaning they can't be modified

- you can reassign a variable to a completely new string

in: a boolean operator that takes 2 strings and returns 'True' if the first string appears as a substring in the second string

Ex: `print('a' in 'banana')` → returns True

`print('seed' in 'banana')` → returns False

the **comparison operator** works on strings

Ex: `if word = 'banana'`

```
print('All right')
```

↳ < means before

other comparisons (>, <) put words alphabetically (all uppercase before lowercase)

Figure E.4: Claire's Handwritten Notes on Functions - Page Four

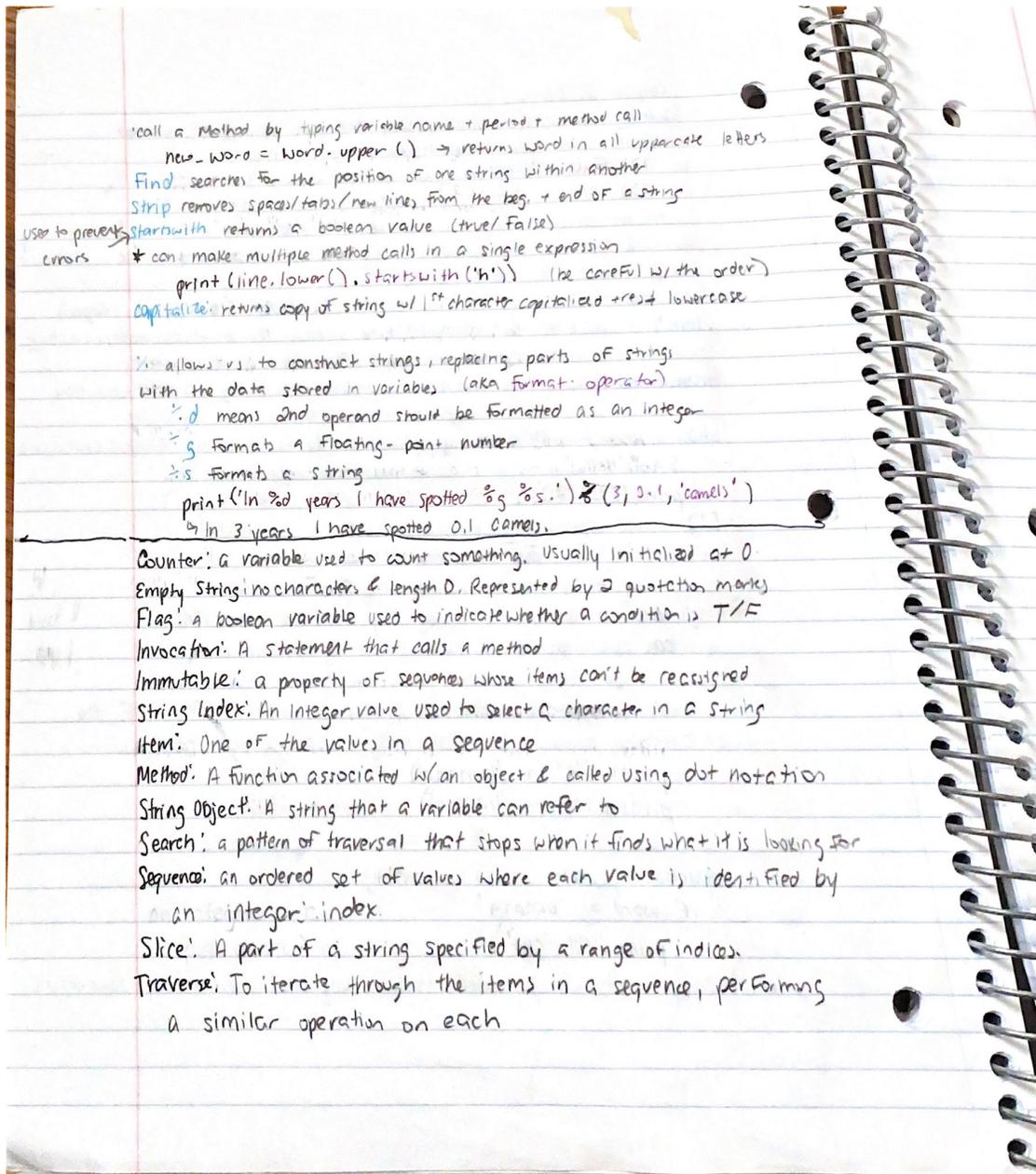


Figure E.5: Claire's Handwritten Notes on Functions - Page Five