Formative Assessment Using Diagrammatic Code Queries: Engaging Students with a Notional Machine

By

Lauren Rachel Himbeault

A thesis submitted to Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements of the degree of

Master of Science

Under the supervision of
Dr. Celine Latulipe

Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, Canada
July 2023

Copyright ©2023 by Lauren Rachel Himbeault

Abstract

Introductory Computer Science classes often start by teaching students to write simple programs in the language being taught. The ability to competently write code that solves problems encompasses many skills and foundational concepts. Students need a basic understanding of how the programming language is structured and how the underlying machine executes the instructions. This mental model of execution of a specific programming language is called a 'notional machine' in computer science. Authors such as Suh et al. [1], Lewis [2] and Fincher et al. [3] emphasize the importance of helping students develop accurate mental models. Various tools such as code visualizations, can be used to help students develop accurate mental models, but getting students to regularly engage with these tools early and frequently can be difficult to fit into a curriculum. My research addresses this issue by paying particular attention to helping students form accurate mental models early.

Dr. Celine and I created a framework to guide Code Diagram Query (CDQs) development and deployment in an introductory programming class. Code Diagram Queries (CDQs) are a formative tool that I designed to aid students in developing accurate mental models of programming language NMs. CDQs are a type of question a student is asked to answer about a snippet of code that is accompanied by a visual of the code execution. I hypothesized that these questions would act as a forcing function to engage students with diagrammatic material that a) evaluated their current mental models and b) challenged and corrected their mental models about the notional machine of the language they learned. A diagrammatic representation of mental models is often used in direct instruction of notional machines

or provided as supplemental resources. Students may ignore these resources if formative activities do not drive them to interact with the diagrammatic materials. In this thesis, I describe how I created and deployed a set of CDQs for an introductory programming class. I then indirectly measured student mental model development through course performance and examined question-type effectiveness. I hypothesized that student engagement with these CDQs may help them accurately visualize and comprehend coding concepts to support their learning. Results from my study suggest CDQs helped to reveal misconceptions in understanding the topics related to the flow of execution programming concept. I found very limited evidence suggesting a slight increase in optional usage of Python Tutor due to its use in the formative assessments for the CDQ section. Overall, my findings suggest that CDQs may enhance understanding of some concepts when paired with specific question types. These findings indicate that CDQs could have a place in classrooms and that their implementation could benefit students learning abstract concepts in computer science.

Dedication

This thesis is dedicated to my incredible husband, Eric and my children, Evan and Alison. No words can describe how thankful I am to these three humans. I could not have finished this without their unconditional love and support. Thank you, and I love you.

Acknowledgements

First, I express my most considerable appreciation to my fantastic advisor, Dr. Celine Latulipe. I am so fortunate to have worked with such a kind, caring, intelligent and empathetic individual. I am honoured to have performed under her supervision. Thank you for being a second mom and helping me become a better student, educator, researcher and human.

I would also like to thank my committee members, Dr. Andrea Bunt and Dr. Amy De Jaeger, for their time, support, and guidance.

I am forever grateful to my lab mates and HCI faculty. I appreciate their feedback and support throughout this journey. Whether it was a poster or paper feedback, snacks for my kids, or a general reminder to sit down and finish my thesis, they have made this not only journey possible but fun. Thank you to Anik, Ananta, Phil, Afsane, Danika, James, Skyla, Pouya, James, Zach, and Yengin. An extra special shout out to my lab mates Patrick, Min and Raquel for being a shoulder to lean on when things became especially tough and for reviewing multiple iterations of the same paragraphs until I was sure you'd shoo me away (but still didn't). Thank you to my lab mate Carter for all his help with question design and classroom support. A final lab thanks to former HCI lab members: Nabila, Lorena, and Rozhan for their unconditional support and assistance throughout this program.

Many thanks to my thoughtful friends who supported me from near and far and still want to be my friend after months of ignored texts while I completed this degree. An extra thank you goes to our neighbourhood friends, Dan and Shay, for feeding my family and me when I was too busy to cook, planning play dates with the kids so I had peace to write, listening to my defence presentation on repeat, teaching me statistics I had long forgotten, and for your general love and support through this journey. Another thank you goes to Christina Penner, whose mentorship and wisdom have guided me. I am fortunate to have your shining light on my side for the dark moments.

Last but certainly never least, I would like to thank my family. My parents, brothers, and in-laws for their constant support and unconditional love. For bringing us food and being our childcare, for listening to me cry and offering support as you could. If you don't realize already, I want to tell you that you are all a shining light in my life and inspire me every day.

Contents

List of Figures ix
List of Tables xi

1 Introduction **1**
1.1 Reflexivity Statement 7

2 Background **9**
2.1 Concreteness Fading 9
2.2 Bloom's Taxonomy 10
2.3 Constructivism in Education 11
2.4 Zone of Proximal Development 11
2.5 Notional Machines in Computer Science Education 12
2.6 Summary 14

3 Related Work **15**
3.1 History of Notional Machines in Computer Science Education 16
3.2 Visualizations and Notional Machines 17
3.3 Building Programming Skills 19
3.4 Mental Model Assessment 20
3.5 Formative Assessment 21
3.6 Summary 22

4 CDQs & the CDQ Framework **24**
4.1 Code Diagram Queries 24
4.2 The CDQ Framework 25
4.3 Computer Science Concepts 26
4.3.1 Bloom's Taxonomy Layers for Targeting Cognitive Processes 26
4.3.2 Question Styles 27
4.3.3 Visualization Resources 30
4.4 Example CDQs 31
4.4.1 Example 1: Analyzing Temporal Flow 31
4.4.2 Example 2: Applying an Understanding of Structure & Scope 32
4.5 Summary 33

5 Course Context **34**
5.1 Grade Breakdown . . . 35
5.2 Formative Assessment Materials . . . 37
5.3 Course Director: Role and Responsibilities . . . 37
5.4 Summary . . . 38

6 Methods **39**
6.1 Creating Course Content . . . 40
6.2 Experiment: Classroom Implementation of CDQs . . . 42
6.2.1 Participant Recruitment . . . 43
6.2.2 Weekly Formative Assessments . . . 44
6.3 Summative Evaluation of Mental Model . . . 45
6.4 Data Collection . . . 46
6.4.1 Course Performance Data . . . 46
6.4.2 Interviews . . . 47
6.5 Data Analysis . . . 47
6.6 Pre-Processing Data Logs . . . 48
6.7 Summary . . . 48

7 Results **50**
7.1 Consenting Participant Information . . . 50
7.2 Overall Engagement with Formative Assessments . . . 51
7.3 Course Performance as a Proxy for Mental Model Evaluation . . . 52
7.3.1 Term Test One Scores . . . 52
7.3.2 Term Test One: Targeted Analysis . . . 54
7.3.3 Final Examination Analysis . . . 54
7.3.4 Formative Performance Analysis . . . 55
7.4 Impacts on Engagement with Optional Resources . . . 59
7.4.1 Immediate Repeated Weekly Attempts . . . 60
7.4.2 Formative Assessments as a Final Exam Study Aid . . . 60
7.4.3 Python Tutor Engagement . . . 61
7.5 Exploring CDQ Dimensions Through Student Performance . . . 62
7.5.1 Recollection versus Recognition Skills . . . 63
7.5.2 Computer Science Topics . . . 65
7.5.3 Usefulness by Cognitive Processes and Overarching Programming Concepts . . . 66
7.6 Interview Case Studies . . . 68
7.6.1 Participant 1: . . . 69
7.6.2 Participant 2: . . . 69
7.6.3 Participant 3: . . . 70
7.7 Summary . . . 71

8 Discussion **73**
8.1 RQ1: *How does the use of CDQs impact the accuracy of mental models, using course performance as a proxy?* . . . 74
8.1.1 Summative Analysis Considerations . . . 74
8.1.2 Formative Analysis Considerations . . . 76
8.2 RQ2: *How do CDQs impact the level of optional engagement with PythonTutor?* 78
8.2.1 Engaging with Formative Assessments . . . 78
8.2.2 Formative Assessments as a Final Exam Study Tool . . . 81
8.2.3 Final Considerations for Python Tutor Engagement . . . 81
8.3 Exploration of CDQ Dimensions and Student Performance and Experience . 82
8.3.1 Question Type Considerations: MSMC versus WR Questions . . . 83
8.3.2 Computer Science Concept Considerations . . . 83
8.3.3 An Educator's Perspective . . . 85
8.4 Limitations & Obstacles . . . 85
8.4.1 The Deployment Tool . . . 85
8.4.2 Education Research . . . 87
8.5 Summary . . . 88

9 Conclusion **89**
9.1 Contributions . . . 89
9.2 Future Work . . . 90

Appendices **92**
A Approved REB Document . . . 92
B Consent Form: In Class . . . 116
C In Class Script: Consent Form Distribution and Recruitment . . . 119
D Email Consent Form: Absent Students . . . 120
E Email Script: Absent Student Recruitment . . . 121
F Consent Form: Interview . . . 122
G Interview: Question Set . . . 125
H Certification: TCPS Core 2 . . . 126
I Weekly Question Sets . . . 127
J COMP 1012 Winter 2023 ROASS . . . 140

List of Figures

1.1 A student missing a strong and accurate mental model of how their code is executing may make a mistake and not understand the process of rectifying it. Students tend to find themselves in a 'code soup' loop whereby they make small changes hoping they are 'close enough' to an answer that they will stumble upon the correct solution. When this doesn't work, they become frustrated and create more small changes, slowly trapping themselves in this cycle until they no longer recognize any part of their code. 3

1.2 Python Tutor shows code alongside a diagrammatic representation of what the code does at a given step in the execution. The different frames of scope are labelled and depicted as a list of boxes with the active frame highlighted in blue. Variable/object storage inside the frames is seen as boxes and arrows as appropriate. A green arrow alongside the code line numbers indicates which line of code just ran, and a red arrow indicates the next line of code that will run. The current step of the execution is shown beside the total number of steps that will occur in the execution of the provided code. A user can navigate forward and backward through the execution using the 'Prev' and 'Next' buttons. 4

1.3 How a student's mental model of code is formed: students take in information from class materials and their own experiences of what does or does not give them their expected output and build their understanding of how a computer will use their code (NM). The "real" machine then executes the provided code. Each programming language interacts with computer architecture uniquely, exhibited by that language's NM. A student is trying to build a mental version of this interaction for the language they are learning. 6

2.1 Left: Visualization of Bloom's hierarchy. Right: Mapping of specific layers of Bloom's taxonomy to programming specific comprehension. 10

2.2 Illustration of the ZPD in concentric circles. The innermost circle depicts a student's current understanding of a topic which a student can currently do. The outermost ring represents information that is too far from a student's existing knowledge for them to grasp. The middle ring, or ZPD, describes the information that a student can feasibly comprehend with guided resources from an expert. 13

2.3 Python representation of a factorial function. Python stores the function in memory as an object. 13

2.4 Java representation of a factorial method. Java stores methods as activation records on the run-time stack. 14

3.1 "Notional Machine" title keyword search across Research Articles from the ACM Digital Library plotted over time. Since Sorva's work in 2013, the use of the term' Notional Machine' in research has exploded exponentially 17
3.2 When a student is learning to code, we can apply Bloom's taxonomy to see how the skills required increase in difficulty. While writing new code is at the top of Bloom's taxonomy, building a solid foundation in understanding existing code, applying that knowledge to programs and analyzing coding problems to find solutions helps a student become a more adept programmer. 20

4.1 The CDQ framework connects visualizations, NMs, and identifies the different cognitive levels at which formative assessments can take place. 25
4.2 A modified screenshot from Python Tutor depicting a method, *summation(n)*, which sums up the values of 0 to n-1 and returns the summation. 31
4.3 A modified screenshot from Python Tutor depicting the function, *check_age(n)*, which determines if an inputted age is above the specified age of majority. . 32

5.1 A screenshot of the student view of the formative assessments. In each week's content folder, a student would see the practice problem set they were to submit that week, labelled with their enrolled section and the current week number. The section has been intentionally blurred. 35
5.2 Visual flow of the COMP 1012 class by week, showing each week's topic. The summative evaluations, important cut-off dates, and student weekly actionable items are highlighted. CDQs were only used in topics 1-7. 36

6.1 An overview of the methods described in this chapter. Starting with CDQ question creation and finishing with analyses of performance and interview data. 40
6.2 Example of a question in the initial implementation of CDQs in the Runestone Academy Textbook for SCI 1002. 42

7.1 First Attempt Percentages: % of students who submitted at least one response for a given week of the maximum possible submissions for a given week. . . . 52
7.2 Term Test 1 question added which specifically tests a student's understanding of what a computer is doing with a snippet of Python code. 53
7.3 Term Test 1 scores of consenting students. 53
7.4 Box plot of the final examination scores for consenting students 55
7.5 Box plots of first attempt scores for each week using the original data of all questions in each week. This includes answers to written response questions, which students from both sections did poorly on. 56
7.6 Grouped by section, the letter grade breakdown for each weekly formative assessment, considering only first attempts. This data set includes both MSMC and WR type questions. 57

7.7 Box Plots of first attempt scores for each topic, omitting the written response (fill-in-the-blank/short answer) questions. Yellow stars on the x-axis indicate statistically interesting topics. 58
7.8 Subsequent Formative Attempt Percentages: % of consenting students who submitted at least 2 responses for a given week. 60
7.9 Paired box plot of the CDQ section showing average student performance on all MSMC style questions versus WR questions. Each point is a student's average mark across all 7 weeks of MSMC questions and WR questions. The red line pairs a student's average mark on the MSMC box plot with their average mark on the WR box plot. 64
7.10 Paired box plot of the nCDQ section showing average student performance on all MSMC style questions versus WR questions. Each point is a student's average mark across all 7 weeks of the MSMC questions and WR questions. The red line pairs a student's average mark on MSMC box plot with their average mark on the WR box plot. 64
7.11 The CDQs implemented in the CDQ section, categorized by CS concept and cognitive process from the framework. They are labelled using the week number, a hyphen, and the question number, matching the labelling in Appendix I. The MSMC questions are in bold font and WR are not. 67
7.12 Average question score on formative assessments for the CDQ section, including MSCS and WR questions. 68
7.13 Average question score on only the MSMC questions on formative assessments for the CDQ section. 68
7.14 These tables are organized by concept and cognitive process based on the CDQ framework. The row and column averages are calculated and shown in red. 68
7.15 Screenshot of Python Tutor code after execution line 13, used during participant interviews. The screenshot focuses on the scope and the use of shadowing global variables with locally created ones. The visualization shows that more than one version of the variables MIN and MAX exist in the current program state. 71

List of Tables

4.1 Examples of types of questions that can be asked when code is accompanied by a diagrammatic representation of the code in mid-execution. Examples of generic introductory programming questions that can be asked when targeting my outlined cognitive processes across the target concepts. 28

6.1 This table outlines the various data sets I collected, how I analyzed them, and which research questions and exploratory pathway this analysis aimed to answer. FA is shorthand for formative assessment. For reference, (1) How does using CDQs impact the accuracy of students' mental models? [Accuracy], (2) How do CDQs impact student engagement with provided optional resources? [Engagement], and (3) An exploration into the dimensions which affect student experiences and performance with CDQs. [Question Dimension] 41

7.1 The number of formative first attempts per topic (February 2023), separated by section, at the time of the data export (February 2023), approximately seven weeks into the course. 51
7.2 Descriptive statistics from term test 1 scores (as percentages). 53
7.3 Descriptive statistics from question 1 on term test 1 (as percentages). 53
7.4 Weekly average correctness for first attempts by topic, all question types included. Note that Iterable Objects includes: Lists, Tuples, Dicts, & Sets . . 56
7.5 Weekly Average correctness for first attempts by topic, considering only the multiple choice/multiple select questions. t-test results by section comparing all topics: $(t = 0.39563, p = 0.69931, df = 12)$. Note that Iterable Objects includes: Lists, Tuples, Dicts, & Sets. 58
7.6 Topics broken down by category of multiple-choice/select (MSMC) and short answer/fill-in-the-blank (WR) for each section. 59
7.7 The number of formative assessment repeat attempts for each week that occurred, during the final exam study period (April 2023), organized by section. W# indicates the week number the repeat attempts were observed. Note that Iterable Objects includes: Lists, Tuples, Dicts, & Sets 61
7.8 Two-way ANOVA test of average scores across the two sections and considering question types (MSMC and WR). Results indicate significant difference between sections and between question types and a significant interaction between section and question type. 65

7.9 Inferential statistics on formative assessment scores for each week. T-tests run for each topic comparing sections on a topic level. Only multiple select/choice style question scores were included in the data sets. Topics with significant results are in bold. Week 3 data was omitted due to a question imbalance. . 66

Chapter 1

Introduction

A mental model is an individual's internal representation of how something works [4]. In Computer Science, much of our work revolves around abstract concepts and comprehension. A robust mental model of what a computer does with our code is essential when writing programs to ensure our programs output our expected results. This mental model of *how* a computer is interacting with the code we write is called a Notional Machine (NM) [5]. When a student is developing their mental model, they may use class materials such as lectures, labs, videos, and activities to ground their understanding of how a physical computer uses a programming language (such as Python) to manipulate data and complete a task (Figure 1.3).

Du Boulay defined the Notional Machine (NM) as an abstract model of a computer program or programming language construct [5]. The scope of an NM can be for a particular concept or defined for an entire programming language. NMs are representational forms that inform users how computer code executes. These machines can be a physical model, a story, a visualization or a semantic rule. For example, a notional machine can be a story we tell about a piece of code, like describing a variable as a box. NMs can aid in student comprehension of the flow of code execution, how functions execute, and how data is stored. A student's development of an accurate mental model, i.e. a correct notional machine, is a critical skill that separates adept coders from others [6].

Novice programmers often struggle with learning to code [7]. Many factors are attributed

to this difficulty of learning to program, which can make isolating and overcoming problems a challenge [7]. One possible reason for this challenge is that current teaching methods are often too subtle in their notional machine use [8]. One of the implications of this struggle is the creation of inaccurate mental models of computer processes [9], which leads to further complications in a student's educational journey. The explicit inclusion of NMs into a curriculum can support students in producing more accurate mental models of what a computer is doing with code [8].

Programming languages interact with data and computer memory in different ways. A student who struggles with a strong understanding of the language's notional machine may need help using the language effectively [10]. The primary responsibility of educators in introductory programming courses is to support their students in developing accurate mental models of code execution. Teaching and understanding abstract topics in programming, such as arrays, which students often need more familiarity with, can be difficult. A student's understanding of what happens "under the hood" in a computer program has been proven to impact their self-efficacy and course performance [7, 11–14], therefore there is benefit in supporting this process of abstract concept comprehension. Helping students develop accurate mental models is a pertinent part of becoming successful programmers.

The modes used when teaching programming vary greatly, including what languages are taught and which topics are covered. While ACM curriculum guidelines exist for teaching programming at a post-secondary level, research suggests that when first teaching students how to program, getting them to start coding immediately is less effective than we may expect [15]. At the University of Manitoba, a fail rate of 34% is observed in students with high school programming experience when taking a university-level introductory programming course. This percentage soars to 60% for students who have not previously been exposed to programming (C. Latulipe, personal communication; COMP 1000 Proposal, March 12, 2021). These numbers show that students who have had prior experience and time to practice code, routinely perform better than those who have no experience. Forcing students to write

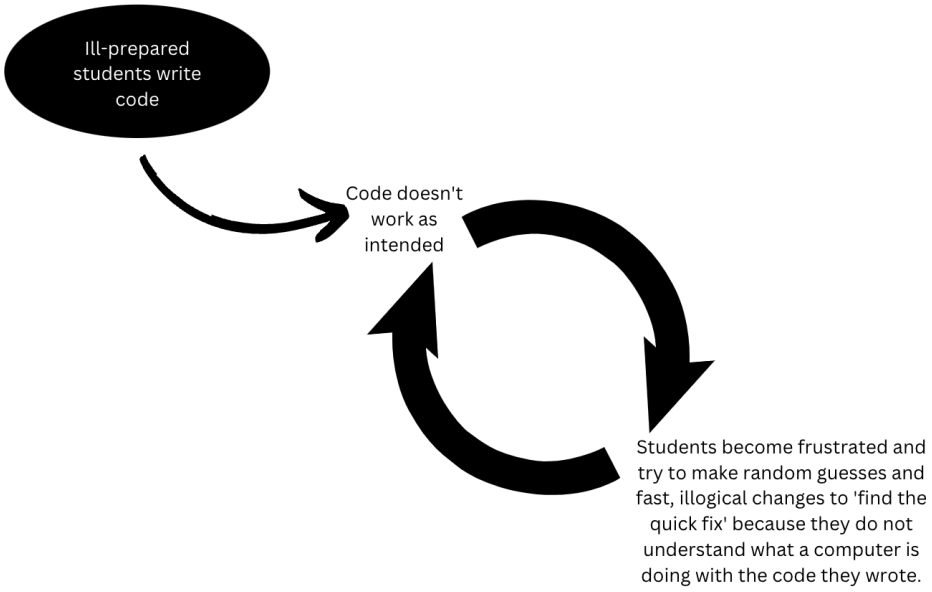


Figure 1.1: A student missing a strong and accurate mental model of how their code is executing may make a mistake and not understand the process of rectifying it. Students tend to find themselves in a 'code soup' loop whereby they make small changes hoping they are 'close enough' to an answer that they will stumble upon the correct solution. When this doesn't work, they become frustrated and create more small changes, slowly trapping themselves in this cycle until they no longer recognize any part of their code.

code before they fully comprehend what a computer is doing likely contributes to failure because students make random guesses about what instructions to write and why the code they wrote does or does not work, and then they quickly become frustrated. This novice programmer workflow is cyclical, and these frustrations can quickly spiral (Figure 1.1). It is known that inaccurate mental models contribute to failure in understanding programming concepts in novice programming classes [9, 16].

Giving students a visualization depicting an NM may support their developing mental models. Various programming visualization software and applications exist to help students solidify their understanding of NMs [17–19]. PythonTutor (PT), the tool used in this research, is a web-based application that visually represents user-written code executing in Python [18]. Its animation feature allows students to trace through their code and understand how it interacts with their computer at a deeper level, allowing them to control the execution step-by-step (Figure 1.2).

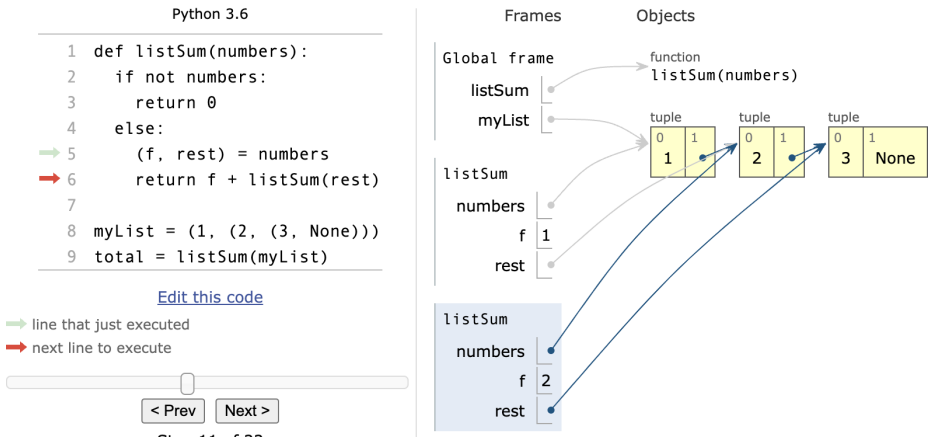


Figure 1.2: Python Tutor shows code alongside a diagrammatic representation of what the code does at a given step in the execution. The different frames of scope are labelled and depicted as a list of boxes with the active frame highlighted in blue. Variable/object storage inside the frames is seen as boxes and arrows as appropriate. A green arrow alongside the code line numbers indicates which line of code just ran, and a red arrow indicates the next line of code that will run. The current step of the execution is shown beside the total number of steps that will occur in the execution of the provided code. A user can navigate forward and backward through the execution using the 'Prev' and 'Next' buttons.

My research investigates ways of supporting novice programming students by paying special attention to helping them form accurate mental models early. I do this by designing and implementing formative assessments guided by a novel framework collaboratively designed by my advisor, Dr. Celine Latulipe and myself, called the Code Diagram Query Framework. The CDQ framework is designed to guide the development of assessment questions that leverage visualization tools for teaching core programming concepts across three levels of Bloom's taxonomy. Using this framework, I created a set of Code Diagram Queries (CDQs) to engage students with the Python Tutor visualization tool. CDQs are intended to be low-stakes formative questions using visualizations to help students understand *how* code is being executed. I hypothesized that CDQs could be used as a forcing function to engage students with visual or diagrammatic material that helps them develop a more accurate mental model of how a computer executes code in the language they are learning.

I use these visually stimulating queries to urge students to engage more deeply and frequently with Python Tuthor (PT), an optional course resource they may otherwise ignore.

The design of CDQs prioritizes asking students pointed questions about their understanding of *how* a computer executes code. Having a student visualize conditional structures mid-execution allows educators to assess their knowledge of how the computer reached that point in their code. In Figure 1.2, the code just finished executing line 5. Checking that a student understands that before line 5, line 2 is executed may ensure a deeper understanding of control structures and how the computer interprets and executed the code. These questions should help to engage students in the material and develop a more accurate mental model of what is happening inside the computer when it is executing the code they write. Using our CDQ framework, I created a set of Python-based CDQs and deployed them as formative assessments in one section of an introductory CS class, while giving similar questions without visualization components as formative assessments to the other section of the class. Finally, I measured the effectiveness of CDQs in improving a student's mental model construction of the Python NM by comparing student engagement and performance across the two sections. My thesis hypothesis is that CDQ questions will be an effective formative assessment tool to help students learn.

The primary focus of my research was to design and implement the Code Diagram Query (CDQ) question type and assess the effectiveness of CDQs in student development of accurate mental models of the Python notional machine. Focusing on this area of understanding, I investigated two research questions:

RQ1. How does using CDQs impact the accuracy of students' mental models, using course performance as a proxy?

RQ2. How do CDQs impact student engagement with provided optional resources?

In addition to these two research questions, I explored the various dimensions that may affect student performance and experience with CDQs. These include the different computer science concepts, cognitive processes, and question types.

This thesis presents the work I completed on these research questions. By analyzing

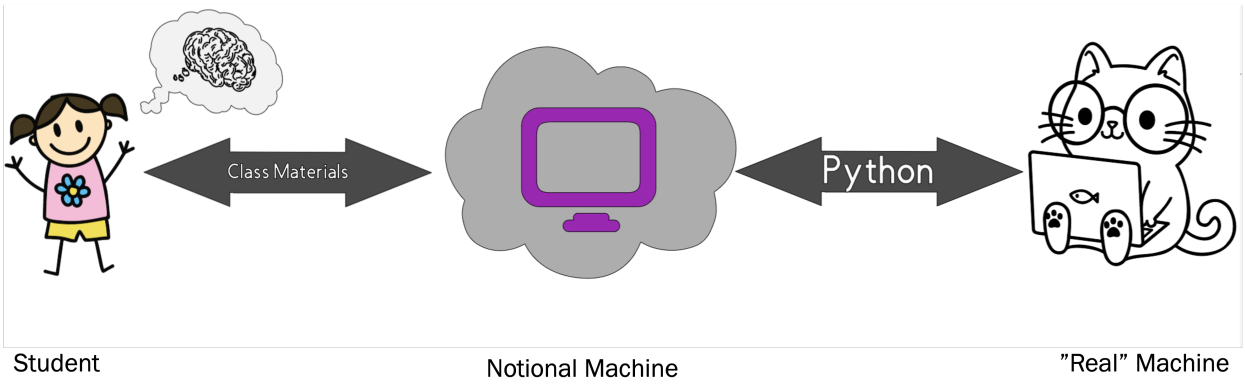


Figure 1.3: How a student's mental model of code is formed: students take in information from class materials and their own experiences of what does or does not give them their expected output and build their understanding of how a computer will use their code (NM). The "real" machine then executes the provided code. Each programming language interacts with computer architecture uniquely, exhibited by that language's NM. A student is trying to build a mental version of this interaction for the language they are learning.

student engagement, course performance and course interaction logs, I indirectly assessed student mental models as a way to measure the effectiveness of the PT-based CDQs. Since I could not directly interact with a student's brain and query their mental model, I used course performance on examinations and questions designed to target a deeper understanding of computer/code interactions as a proxy. Specifically, I wanted to determine if CDQs could assist students in tracing their code and comprehending code execution, both critical skills in becoming adept programmers [6].

This thesis is structured as follows. First, I present a chapter on the contextual background of my research. A chapter on the related works provides the theoretical and empirical grounding for my hypotheses. Next, I outline the framework we designed and implemented to test my hypotheses in my study. I then outline the course I ran my study in, COMP 1012. This outline includes the course layout, a breakdown of evaluations, and my role in the class. The subsequent chapter outlines my methodology and summarizes the steps to recruit participants, implement and deploy CDQs, and collect and analyze the data gathered. The results chapter reports on the analysis of the data I collected and is followed by the discussion chapter, where I compile all the results and theorize and justify their meanings.

Lastly, I identify the limitations and contributions of my work and future opportunities to investigate CS student support and mental model development more deeply.

1.1 Reflexivity Statement

As both a current student and educator, I would like to briefly acknowledge how my background played an important role in interpreting my findings and conducting this research [20].

At the time of writing this thesis, I am a 32-year-old woman who has studied computer science for almost eight years. I worked in industry for four years before having my two children. I have also been an sessional instructor at the post-secondary level for five years, teaching CS0, CS1, CS2, Software Engineering and Professional Practices in Computer Science. I have been a student and instructor, often both in the same term and have had an opportunity to see the peaks and valleys of how computer programming is currently taught.

I am also a minority in computer science. As a women, I am often left looking around a room only to find no one looks like me. Making friends in my first years of computer science was difficult and while there are many people I now call my friends, my first introduction to computer science was intimidating and isolating. I often felt stupid when asking questions others seemed to already understand or less than when I struggled with concepts others were confident in. These feelings have inspired my drive to make computer science feel approachable and accessible to all. Having these experiences and feelings motivates me to create as many accessible opportunities to learn as possible.

It's important to me that students feel supported and safe in those foundational years. Furthermore, I strongly believe in the importance of remembering that educators are people too. We are constantly iterating on our own successes and failures, and looking to welcome learners.

In this thesis, I deployed Code Diagram Queries (CDQ) in a formative assessment setting and investigated their impact on student understanding of the Python Notional Machine.

These assessments were deployed in a class that I have taught previously but did not teach during this study. I am familiar with the learning management system used and Python Tutor, the online visualization tool that I used during the study.

Other people also helped build the formative assessments. Students and researchers with various backgrounds offered feedback on the questions to help ensure their clarity and consistency. For example, the CDQs were written by more than one person and tested together. Furthermore, they were tested in an undergraduate class the term before my study ran where errors in spelling or explanations in the questions were uncovered and modified accordingly. The quantitative data was interpreted by my advisor and I, individuals who have worked together in more than one programming course and while all my education experience comes from the University of Manitoba, Dr. Celine has a wider lens when exploring this data.

Chapter 2

Background

As outlined in Chapter 1, my hypothesis is that we can better support our students in developing their mental programming models through CDQs. To help ground my interdisciplinary investigation, I provide contextual definitions for several core constructs in this chapter. These include the Concreteness Fading methodology, Bloom's Taxonomy of cognitive engagement levels, learning using Constructivism, the Zone of Proximal Development and Notional Machines.

2.1 Concreteness Fading

A standard methodology used when teaching novice programmers is an instructional style that involves taking a learning objective and teaching it, initially in a concrete and familiar way (e.g. using jellybeans to teach math)—following that, transferring the student's concrete understanding of the concept to the abstract (e.g. mathematical formula comprehension). This style is known as Concreteness Fading. A representation of notional machines, such as a box for a variable assignment, takes the learning objective of understanding variables and gives students something concrete to aid their comprehension [2]. Teaching programming through notional machines and different visualizations is an example of concreteness fading and has been an effective way to learn computer programming [21].

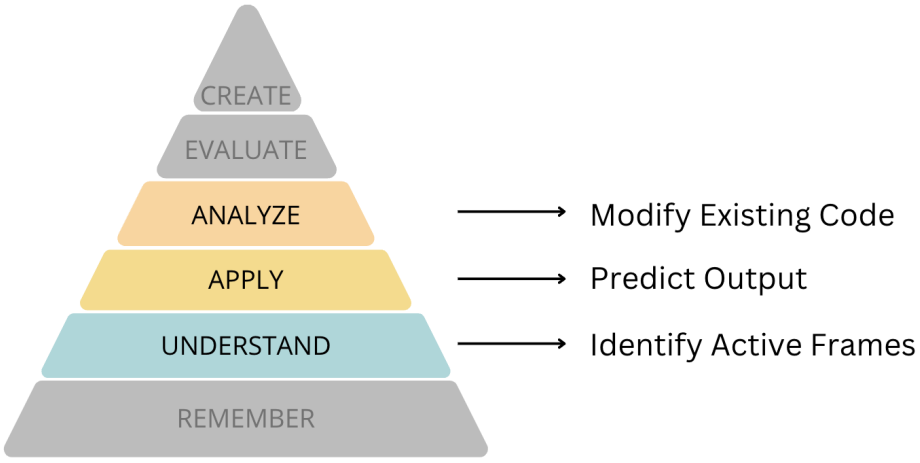


Figure 2.1: Left: Visualization of Bloom's hierarchy. Right: Mapping of specific layers of Bloom's taxonomy to programming specific comprehension.

2.2 Bloom's Taxonomy

Bloom's taxonomy is a model used in education to classify learning activities by cognitive engagement level [22]. When learning to program, Bloom's taxonomy can inform the development of activities and assessments (Figure 2.1). While CS activities are often built on this taxonomy, many educators have found categorizing Computer Science assessments into specific levels of the taxonomic hierarchy difficult [23]. The skills required to learn to code can be separated and mapped to the taxonomic groups; however, it is also possible to take this one step further and for a specific skill to be broken down and mapped. We can break down a particular concept, such as conditionals, and create activities targeting Bloom's hierarchy's different layers (Figure 2.1). At the Understand level, we can ask students to identify control structures like if/else statements, while, and for loops. At the Apply level, we can query a student's depth of understanding by having them determine which situations are better suited for which control structures. Finally, at the Analyze level, we can ask students to examine code snippets and categorize them by which type of loop they would use and why. This allows us to create customized activities targeting each skill required to grasp an individual concept fully.

2.3 Constructivism in Education

Constructivism is a learning theory which posits that people build a knowledge base by engaging with the world around them and taking part in experiences that help form their understanding in a domain [24]. When students learn a new skill that they are unfamiliar with, an expert guiding them through new material is logical. Constructivism has a student-centred focus on learning, which is critical in capturing students' attention and having them engage meaningfully with materials [25]. Various disciplines use altered theories of Constructivism to cultivate the learning outcomes appropriate for that domain [24, 26, 27]. For example, Constructionism focuses on physically building something to interact with and understand information [26, 27], and Social Constructivism describes learning through interactions within the context of a social environment [28]. One example of where Social Constructivism can be applied is through visualizations and an expert explanation. While direct instruction and constructivism are not often thought to pair well, using an expert, such as an educator, facilitating activities and using instructional material to benefit a student's understanding can be considered a mix of direct instructional aspects and social constructivism. Pairing aspects of direct instruction and visualizations can help students develop correct mental models. Some visualization tools allow students to create their example code snippets and build their understanding of what the execution is doing. For this to be effective, we need students to actively and regularly engage with the provided visualizations and resources.

2.4 Zone of Proximal Development

When building mental models of how computers execute code (i.e. NMs), novice programmers need a frame of reference to develop their models, so direct instruction is often imperative. Without expert guidance, students construct their mental models independently and often incorrectly [29]. To build a complete mental model of a programming language's

NM, students need a more comprehensive understanding of computer architecture and code execution, but due to the complexity of modern computer architecture, that is not taught until later. Having subject matter experts steer a student's knowledge allows students to comprehend more than they would independently [28]. This learning occurs in the zone of proximal development (ZPD) using various resources, known as scaffolding [30]. A visualization of the ZPD shows a student's comfort zone of their current understanding and how far-reaching the information in a given domain can be, much of it too complex to reach from a student's initial standpoint. The ZPD, depicted in Figure 2.2, shows the middle circle, representing the new knowledge an individual may gain when working with an expert and scaffolding. Common scaffolding examples include reading textbooks, attending lectures, pair programming, and peer-instruction quizzes. Students have been found to benefit from learning in the ZPD across various attributes including, but not limited to, topic comprehension and confidence [29]. Any assistance a student can use to further their learning can be considered scaffolding [31, 32]. Scaffolding is how students connect new information to their current understanding of a topic or subject [30]. Scaffolding is often seen as the building blocks used in conjunction with a concreteness fading methodology to support students in moving their knowledge from the concrete to the abstract. By creating visualizations of code snippets for students to engage with, instructors can steer their learning into the ZPD.

2.5 Notional Machines in Computer Science Education

The term Notional Machine (NM) was defined by Du Boulay as 'the idealized model of the computer implied by the constructs of the programming language' [5]. Simply put, it is the abstraction of how a computer interacts with a program. Our mental model is our inner visualization of this. Teaching NMs often incorporates the concept of concreteness fading and is shown with familiar physical, tangible representations [1, 2, 33].

Notional Machines vary by language since different programming languages interact with

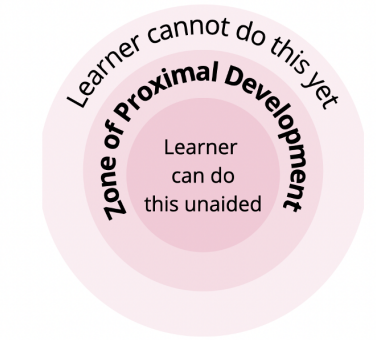


Figure 2.2: Illustration of the ZPD in concentric circles. The innermost circle depicts a student's current understanding of a topic which a student can currently do. The outermost ring represents information that is too far from a student's existing knowledge for them to grasp. The middle ring, or ZPD, describes the information that a student can feasibly comprehend with guided resources from an expert.

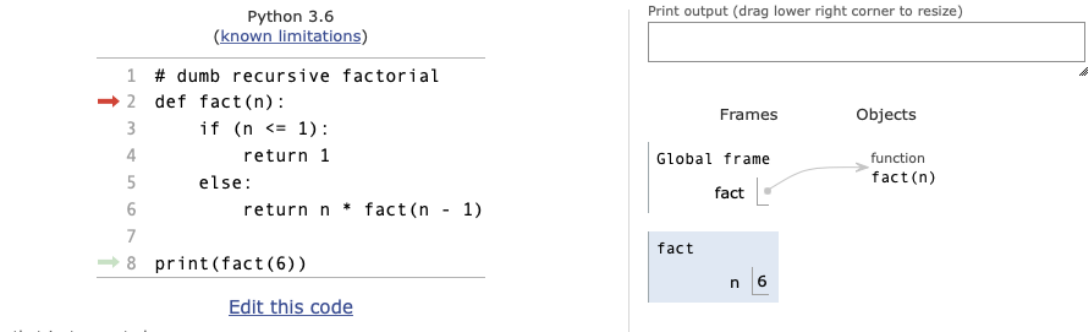


Figure 2.3: Python representation of a factorial function. Python stores the function in memory as an object.

computer architecture in unique ways. An example of this is "calling a function." This concept has a different visual representation in Python (Figure 2.3) and Java (Figure 2.4) thereby making the NMs for this operation different in each language. Python stores functions as objects with a reference in heap memory. Hence, its depiction typically shows an arrow pointing to a labelled box representing the memory where the function can be found. Java does not treat methods as objects, so it stores the function information in an auxiliary storage area on the JVM (Java Virtual Machine). These functions often appear visually as activation records (or a drawn rectangle) on the run-time stack (a tower of rectangles). Therefore, there is not a universal notional machine for students to learn and understand but potentially many that differ in various ways.

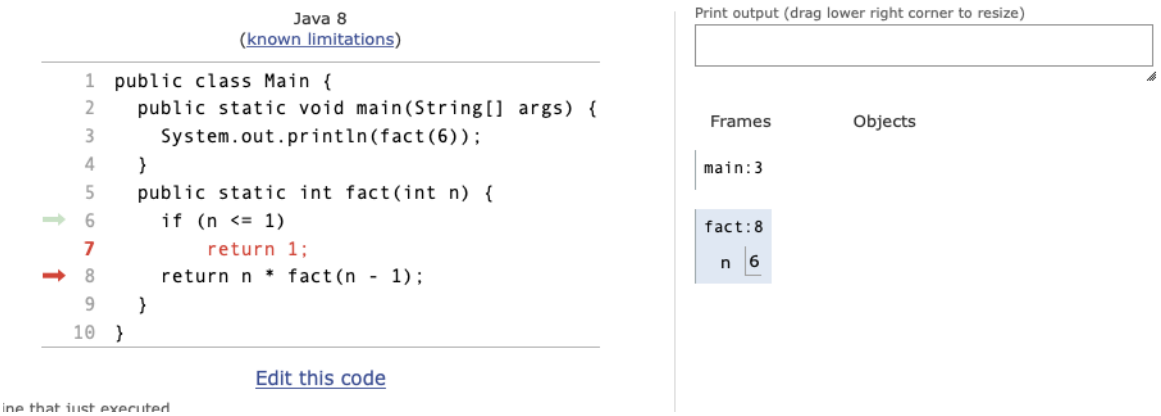


Figure 2.4: Java representation of a factorial method. Java stores methods as activation records on the run-time stack.

2.6 Summary

In this chapter, I provided background on educational concepts relevant to programming. These include the academic concepts of Concreteness Fading, i.e., moving from the familiar to the abstract, Bloom's taxonomy, the model used in education to classify learning activities by cognitive engagement level, and the Zone of Proximal Development (ZPD), the learning space where subject matter experts steer a student's knowledge allowing them to comprehend more than they would independently. I also define the learning theory of Constructivism as the way students learn by constructing their understanding of a topic or concept through experiences in the targeted domain. Finally, I defined Notional Machine as the abstraction of a programming language's interactions with a computer; an individual's mental model is their inner visualization of this phenomenon.

Chapter 3

Related Work

This chapter reviews relevant research in the Computer Science Education field. First, I look at existing work which studies the benefits of combining the explicit teaching of Notional Machines with visualization resources in first-year programming classes. There is a consensus that the direct introduction of NMs with visualizations has led to students more easily understanding code and CS concepts. Next, I consider prior explorations in using code-tracing activities as a basis for learning to code. Studies suggest that building on the skills learned through tracing activities benefits students more than starting novice programmers with coding activities. I then discuss the different strategies that researchers use to evaluate mental models. Lastly, I explore previous research in formative assessments at the K-12 and post-secondary levels. The importance of formative assessments is well documented for K-12 classrooms and beginning to emerge at the post-secondary level. The introduction of post-secondary level challenges such as class sizes means this population is often under-served with respect to formative assessment opportunities, and engaging university students with low-stakes assessments still needs to be improved and evaluated.

3.1 History of Notional Machines in Computer Science Education

In 2006, teaching NMs was considered unimportant compared to other educators' tools [34]. This mindset may have had to do with educators debating how the term "Notional Machine" is defined. Bennedsen and Schulte agreed that some topics were best taught using concrete NMs, meaning physical, digital, or visual representations, regardless of how "Notional Machine" was defined [34]. While Bennedsen and Schultz believed NMs have their benefits, they emphasized that NMs are a learning tool that should not overshadow other learning devices used in programming courses [34]. In 2013, Sorva published a paper which brought together a variety of perspectives on notional machines, and their perceived usefulness [35]. Since Sorva's work, the investigation of NMs and their use has exploded (Figure 3.1). While all of these authors comment that all introductory programming courses incorporate NMs in one way or another, explicit or not, there are varying beliefs on their importance.

We see the use of NMs in CS classrooms regularly, even where the term 'Notional Machine' is unused [34]. Examples of this include the use of diagrams to understand algorithms or control structures in computers [36, 37]. Classrooms that have begun to use NMs formally are achieving great success in integrating activities that explicitly aid in students developing a more profound understanding of abstract programming concepts [9]. Lewis implemented physical representations of the Java NM and found that their students asked fewer repeat questions, and their grades improved [2].

Teaching NMs often incorporates the concept of concreteness fading and is shown with familiar physical, tangible representations. Over time, a learner can better comprehend the learning objective without requiring the familiar representation. The concepts taught through these concrete representations gradually fade into abstract concepts. Fyfe and Nathan discuss the topic of concreteness fading and its viability as a pedagogical tool [38]. Their initial foundation of investigating concreteness fading, and its usefulness has been

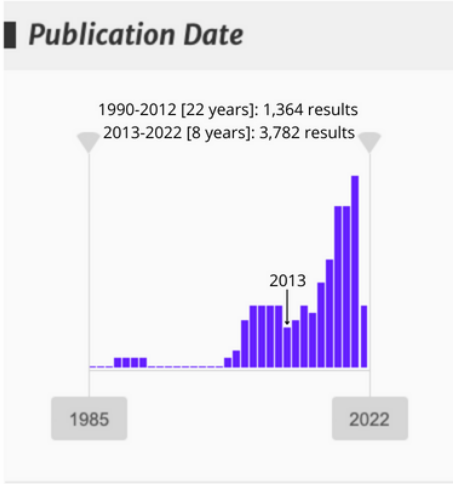


Figure 3.1: "Notional Machine" title keyword search across Research Articles from the ACM Digital Library plotted over time. Since Sorva's work in 2013, the use of the term' Notional Machine' in research has exploded exponentially

expanded on by Suh [21] and Trory et al. [33], who each separately discuss the concept. These papers explore concreteness fading as a practical learning tool for students across many disciplines. Concreteness fading is especially relevant when teaching CS, as programming is inherently abstract [21, 33].

As coding languages interact with computer architecture in various ways, NM models vary by programming language. Because of this, generalized research evaluating their efficacy is sparse. Classroom experiences of understanding notional machines using a physical representation were highly effective when introducing various topics [2, 3]. While the previously mentioned research did not directly evaluate the correlation of explicit NM references and their impact on learning, both Lewis and Fincher observed vast improvements in students' comprehension of the targeted concepts. They each describe these successes by noting improved evaluation scores and fewer repeat questions in class.

3.2 Visualizations and Notional Machines

I suspect diagrammatic queries will aid in student mental model development because of work that has been done which suggests how well-constructed and explanative diagrams can

support students effectively when learning to program. Dickson et al. explored notional machines in 2020 and how explicitly integrating them into a curriculum can help students produce more accurate mental models of what a computer is doing with the code [8]. The authors also provide a detailed definition of the differences and the relationship between NMs and visualizations. According to their report, it is possible to have many different visualizations that all refer to the same notional machine, and each type of visualization has its pros and cons. Dickson et al. also explain how some visualizations may be associated with multiple NMs. For example, array memory is allocated similarly for Python and Java; therefore, the visualizations would be similar. The opposite is also true: different languages may handle other operations differently, and each process would have a unique NM in these cases. For example, primitive values in Python are stored as objects; however, Java stores these on the stack.

Visualizations are a tool that educators commonly employ to help students learn. Visualizations can be static images, drawings on whiteboards, or animated media. Coding Strips is an example visualization which uses the familiar approach of comics to teach coding concepts [21]. Preliminary evidence suggested that students and educators enjoyed using Coding Strips it they effectively support students in understanding code and concepts. Mazumder, Latulipe, and Pérez-Quiñones's [39] work on the importance of diagrams further illustrate how a well-constructed visual can ensure the accuracy of a student's mental model and efficacy of their problem-solving skills. Mazumder et al.'s work showed a lack of effective diagrammatic materials in textbooks and the need for additional visuals to aid students in their mental model development. A drawback of the static visualizations found in textbooks is their lack of interactivity. When a student is solidifying their understanding of a concept, the ability to interact with and modify a visual to test how things work is crucial [40]. There are a plethora of interactive coding visualization tools available such as Labster [17], Python-Tutor [18], CrossCode [41] and various other code visualizers [42–46], that allow students to explore how code is executed.

PythonTutor (PT) is a web-based program for teaching and visualizing code [18]. PT can be embedded into many web-based applications or accessed directly through the website. Although PythonTutor started as a visualization/animation program for Python, it has since expanded to include visualizations for other languages such as C, C++, JavaScript, Ruby, and Java, making it a versatile tool for different courses [47]. PT allows users to write their code and see how the computer system will interact with the program during execution. PT is often considered a Notional Machine because it visualizes and animates the execution of code in a specific language. Unlike Lewis's concrete representation of physical memory using pockets and iClickers [2] or Mazumder et al.'s explanation of useful static textbook diagrams [39], PythonTutor uses digital visualization and animation to aid students in developing accurate mental models of an NM for a programming language. PythonTutor has been effectively used in various educational studies when investigating different ways to support novice programmers [47, 48].

3.3 Building Programming Skills

Prior work has established that expecting students to write code as an initial learning skill when studying coding is less successful than once believed [49–51]. Studies suggest building on skills such as sketching and tracing through pre-existing code helps students build a more robust understanding of computer science concepts [52, 53]. Too often, students are asked to write code before being given the opportunity to understand and apply their knowledge [9, 54, 55]. A core skill leading up to learning to write code is tracing through code snippets [56]. When looking at Bloom's taxonomic levels and learning to code, writing new code is the most difficult, while working to understand existing code is a foundational block (Figure 3.2). Teaching students to build a code tracing skill often involves using visualizations. Research in this domain includes investigation into a multitude of techniques. These include various online visualizations and pencil and paper tracing activities [50, 57].



Figure 3.2: When a student is learning to code, we can apply Bloom's taxonomy to see how the skills required increase in difficulty. While writing new code is at the top of Bloom's taxonomy, building a solid foundation in understanding existing code, applying that knowledge to programs and analyzing coding problems to find solutions helps a student become a more adept programmer.

Qian and Lehman approach the novice programming problem from the educator's perspective [54]. Alongside Jenkins [58], Qian and Lehman [54] state that even if an educator is considered an expert in their subject, that does not mean they can see the given problem through a novice's eyes. This contributes to the difficulty students face in introductory courses. The authors state that novice learners need to be taught by someone who can *teach* programming well, not just by someone who can program well. This brings up the exciting topic of how educators can be supported in teaching introductory programming effectively. The CDQ framework may give educators another tool to guide their question creation.

3.4 Mental Model Assessment

As a mental model is an inner interpretation and visualization of a given topic, it is not something tangible. When an individual is confronted with an inconsistency in their mental model, given new information, they look for an explanation. Rather than ignoring these inconsistencies, it is important to understand them so that we can alter our model to appropriately fit correct information [59]. As researchers are unable to directly see an individual's mental model, we must rely on various techniques and proxies to evaluate a student's internal understanding of a topic or concept. There is general agreement that when students are asked questions designed to assess their level understanding, they use their current mental model to inform their answer [60]. Relying on this well-grounded assumption, researchers use different strategies to probe mental models. Some researchers such as Ngo [61] and

Horvath [62] rely on interviews and open-ended questions. They ask participants to explain why they believe their solution to a problem is correct or what facts are they relying on to answer the question. Other researchers such as Mazumder et al. [63] ask multiple choice questions which directly probe for the understanding of a specific topic. In my study, I use a mixed approach, asking multiple-choice and open-ended questions throughout the formative assessments. I also conducted interviews to gain additional perspective from participants.

3.5 Formative Assessment

There are two main types of assessments: evaluative or summative assessments, which aim to measure how much a student has learned at the end of a topic or class and formative assessments, which are offered throughout the learning process and are used to encourage a student to assess their learning of the material. Any assessment can be formative or summative; the difference is in the stakes, the frequency and the amount of feedback. Examples of typical formative assessments can include question-and-answer sessions, drawing out comprehension maps, and low-stakes quizzes [64]. While summative assessments are essential to ensure that appropriate comprehension has occurred for a student, formative assessments allow for frequent feedback and allow students to assess and correct their learning path should their understanding stray off its intended course.

The importance of formative assessments when teaching CS to primary school students is well studied [65–67]. Formative assessments are shown to be both a crucial and often unaddressed need in schools. This is the case when we look at post-secondary level programming and see limited data on measuring the effectiveness of formative assessments at this level [68]. As not all students feel confident with the same question styles [69, 70], it is important to structure assessments using different types. Roberts explores the use of multiple select/choices (MSMC) style questions in CS classrooms in formative settings and describes the gains that can come from using these question types effectively [71]. This is likely made

more challenging by larger class sizes as Computer Science enrollments explode [72]. When considering creating a formative assessment framework, intersecting visual resources to help students develop more accurate mental models seems promising.

An assessment is only truly formative if a student does not feel the same high-stakes pressure that comes with a summative or evaluative assessment; however, from my experience in the CS classrooms, if an evaluation is low-stakes, students might ignore it or may not feel as compelled to engage with it when balancing their busy schedules. A significant challenge of engaging students with optional assessments, specifically at the post-secondary level, is motivating students to interact with them, and current work is investigating the types of students who use formative assessments and how best to engage all students with optional resources [73]. Picking a technique to effectively engage students with optional resources intended to allow them to assess their understanding and enhance their comprehension of course materials is a challenge likely faced by many post-secondary programming instructors.

When designing practice problems that will be both engaging and effective for students to assess their level of comprehension, we need to ensure our problems target specific learning objectives and concepts [74]. The methodology of using Bloom's taxonomy in classrooms is well studied [75, 76] although has often been considered challenging to incorporate in CS education [77]. We can utilize Bloom's taxonomy [22] to map activities to cognitive processes. Building on previous and emerging research, I theorize that low-stakes weekly practice problems will help post-secondary students gauge their comprehension of programming topics.

3.6 Summary

This chapter lays the groundwork for the CDQ Framework we designed and implemented as described in Chapter 4. The prior work shows that visual representations of abstract notional machines are found to be a productive medium, but there are few examples that quantify how productive this medium is. Second, using techniques such as tracing through

existing code is a more effective protocol when learning to program than immediately writing new code, however it is unclear how to use tracing and visualization tools to target computer programming learning on a cognitive process level. Lastly, I have described prior work which outlines the importance and usefulness of formative assessments as a learning tool that are often overlooked at the post-secondary level due to various challenges. By combining the existing knowledge of these elements, I have decided to investigate the usefulness of low-stakes non-coding visual assessments at the post-secondary level.

Chapter 4

CDQs & the CDQ Framework

In this section, I present Code Diagram Queries and the framework my advisor, Dr. Celine Latulipe, and I built to guide how to write CDQs. A CDQ is a formative question that uses diagrammatic representations of notional machines (NMs) to help students understand how computer code executes.

The CDQ framework brings together visualizations, notional machines, and the different cognitive levels at which the foundation of learning to code can take place (Figure 4.1). The framework defines a design space for formative questions that span three core introductory programming concepts, three different cognitive skill levels from Bloom's taxonomy, and various question formats. Table 4.1 shows the defined design space resulting from the intersection of these elements.

4.1 Code Diagram Queries

Code Diagram Queries (CDQs) are an assessment tool designed to aid students in developing accurate mental models of programming language NMs. CDQs are a visual question type where a student is asked questions about a snippet of code accompanied by a visual of the code execution. The diagrammatic aspect of these questions is a snapshot of code mid-execution from an existing visualization tool that a student must comprehend to answer the

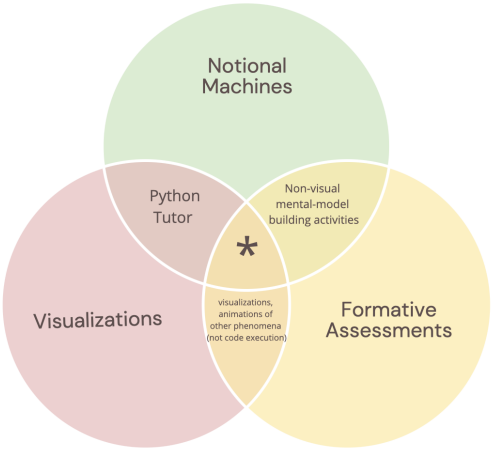


Figure 4.1: The CDQ framework connects visualizations, NMs, and identifies the different cognitive levels at which formative assessments can take place.

question correctly.

While students may be introduced to visualization tools as part of direct instruction, CDQs will act as a forcing function to get students to interact with NM visualizations more profoundly and frequently. The expectation is that this will help them think about how a computer interprets and executes code. These questions are intended to support students' understanding of a notional machine for the language they use as they learn to program.

Allowing students to explore concepts and attempt to understand them in a low-risk, formative environment can help build confidence in coding [64, 66]. I have designed the CDQs in our study to be formative low-stakes assessments to allow students to explore and experiment with coding concepts, without the fear of grade impacts and failure.

4.2 The CDQ Framework

The CDQ framework aims to aid instructors in the practical design of CDQs that probe and correct a student's understanding of code execution for a given course and programming language. This framework guides the creation of formative assessments that use diagrammatic representations of NMs (CDQs) to ensure coverage of multiple cognitive processes and

coverage across the most critical early programming concepts. Using this framework should enable the development of a wide range of questions that can help students in building a more accurate understanding of how a computer executes code in a given language. Even for educators who are experts in their subject, working with novice programmers can be complex, and understanding problems through their eyes can be challenging [54, 58]. The CDQ framework allows for an educator to create questions that pinpoint targeted levels of understandings about topics and may be a resourceful tool when looking for new teaching techniques. The framework is generalized to be language and visualization tool agnostic.

4.3 Computer Science Concepts

The CDQ framework addresses three main introductory programming concepts introduced to novice programmers, regardless of the language in use. First is the *Memory State*: the memory state refers to the object/variable values at any given point in a program's execution and how they are stored. The second concept is the *Structure & Scope*: this concept refers to the frames and stack of the program. Activation records placed on the stack dictate the current function call in a program and the comprehension of the different frames of a program. Specifically, these records dictate the global or local resources available to a program at a given point in execution. The third concept is the *Temporal Path*, referring to the flow of execution and the comprehension of the program flow's previous, current and future state.

4.3.1 Bloom's Taxonomy Layers for Targeting Cognitive Processes

Bloom's taxonomy is a hierarchical model that classifies educational learning objectives into levels differing by complexity and specificity [22]. Our framework targets three layers of Bloom's taxonomy: Understanding, Applying, and Analyzing. The methodology of using Bloom's taxonomy in classrooms is well studied [75, 76] although it has often been considered

challenging to incorporate in CS education [77].

To develop the framework, we outlined core introductory programming skills. We then detailed the concepts that students were expected to understand from each overarching skill. Compared to reading a sentence in a new language and constructing a sentence in said language, modifying existing code is considerably simpler than writing your own. Furthermore, like learning the alphabet of a new spoken language, identifying the parts of code is a critical skill in writing well-constructed and working programs. We outline this mapping visually in Figure 2.1.

With this in mind, we created a depth of understanding which began with simple skills and led to complex creations (Figure 3.2). Recognizing that the 'Remember' layer of the taxonomy often contains relatively shallow learning objectives related to memorizing definitions, we narrowed our focus to the specific layers in Figure 2.1 rather than the entire hierarchy. We decided to omit the remaining top two layers as they involved a more advanced level of understanding concerning writing code. This framework intends to guide question creation for novice programmers. The taxonomy's 'Evaluate' and 'Create' layers are often the center of a coding course's learning objective. Utilizing Bloom's taxonomy in this way allows us to effectively support students in building solid foundations.

Table 4.1 depicts the interaction between the cognitive layers we targeted from Bloom's taxonomy and the introductory CS concepts targeted. Each cell demarcates the learning objectives in that development space. Using the verbiage found in Bloom's taxonomy, we can highlight the types of questions we want to ask students at each level to aid them in self-assessing their skill set.

4.3.2 Question Styles

As not all students feel confident with the same question types [69, 70], we thought it was essential to offer a variety of different CDQ question styles to increase the learning opportunities for students. As this framework was designed to be flexible, the type of questions that

	Understand	Apply	Analyze
Memory State	**Classify** variables by their type in the given frame: circle integers, underline strings	**Determine** the variable that will update after the highlighted code executes	**Explain** why the list in the global frame is updated when *func_x()* runs
Structure & Scope	**Identify** how objects and primitives are presented in the visualization differently	**Examine** frames to understand if the *copy()* is deep or shallow	**Select** all variables that are currently in scope
Path of Execution	**Predict** which frame will have a variable updated next	**Sketch** the remaining flow of execution over the visualization	**Analyze** the remaining number of iterations of the displayed loop

Table 4.1: Examples of types of questions that can be asked when code is accompanied by a diagrammatic representation of the code in mid-execution. Examples of generic introductory programming questions that can be asked when targeting my outlined cognitive processes across the target concepts.

can be used is numerous. When designing the framework, we initially considered five questions: Explanation/Short Answers, Fill-in-the-Blank, Drawing/Sketching, Target Selection, and Multiple Choice/Select.

Explanation, Short Answer, and Fill-in-the-blank questions are all written response questions that allow students to use their own words to explain why something is happening in their code. This type of question relies on strong recall skills as they are often more open-ended and require students to recall information learned in order to successfully answer the question. These open formats allow students to demonstrate what they know about a topic. The subjective nature of an open response makes efficient grading a challenge, especially in large class sizes. One of the main benefits of formative assessments is the feedback a student may receive. Open-ended questions allow a student to express the entirety of their understanding, and this can give an educator additional insight into the type of help a student needs to understand their mistakes. In large classrooms, however, individualized feedback

becomes impractical. Regular expressions to match correct answers or auto graders can be helpful but limited, especially when considering the visual nature of these questions.

Drawing/Sketching questions allow students to represent their code flow and data storage comprehension visually. This type of question is similar to the written response questions (WR), also categorized as recall-type questions, allowing students to demonstrate what they remember and understand about a code snippet. This questioning style was appropriate when developing this framework because students could draw mental models. However, drawings can rarely be auto-graded, limiting the scalability of this question type.

Target Selection is an innovative way to evaluate a student's understanding of code execution and relies more heavily on the recognition skill set, unlike the previous question styles. When looking at a visualization, asking students to pinpoint where in the code something is occurring, where a value is stored, or where the code will execute next forces students to engage with the code and the visualization. The main challenge is more support for target selection style questions in software. Since target selection can be domain-specific, it is not as widely supported as Fill-in-the-Blank or Multiple-Choice question styles. WR and multiple-choice style questions can mimic target selection by asking students to identify what line number something is occurring on or by adding numbered labels to visualizations. Unfortunately, this feels limiting and takes away from the direct interaction with the visualization.

The Multiple Select/Choice (MSMC) styles were the last question type we considered when developing this framework. This question type is an educator's favourite in large class sizes and many scientific domains [69, 71, 78]. From the ease of marking to definitive answers, these questions can give students immediate and helpful feedback. Due to the finite number of options in this question style, students utilize a recognition-based skill set to be successful in MSMC questions. This style's limited answer pool may lend itself to students who need help parsing question objectives. The limited responses also allow educators to give consistent feedback that is inherently more difficult with other question styles. The two

main drawbacks of MSMC questions are luck and rigidness. When given a finite number of options, it is more probable for students to guess the correct answer, at least occasionally, rather than constructing an entire answer on their own in open-ended questions. Second, the rigidness of this style means that the depth of questions an instructor can ask could be improved. Without the free-flowing abilities of the other described techniques, students are limited in responses and educators are limited in what they can learn about their students from these responses. While some prefer the multiple-choice question style for reasons such as ease of marking, students are unable to showcase all they know, as is more often the case with open-ended responses. This style is successfully used when asking pointed questions with few variations of incorrect answers. This is especially beneficial within this framework's context as we specifically target skills and learning objects on a detailed scale rather than asking students to create overarching responses to dissect later.

4.3.3 Visualization Resources

We built the CDQ Framework without considering a specific visualization tool or resource. This was done to ensure flexibility in the framework's use by CS educators. While the study I conducted focused on using Python Tutor [18], this framework allows an educator to select their preferred visualization tool and use it to create CDQs that may help their students. The framework intends to enable an educator to choose a visualization and ask pointed and meaningful questions based on it, allowing students to assess their knowledge gaps better. The visualization chosen may be a physical representation, drawing, image, GIF, code execution recording, or any visual tool that an educator can utilize as a notional machine representation.

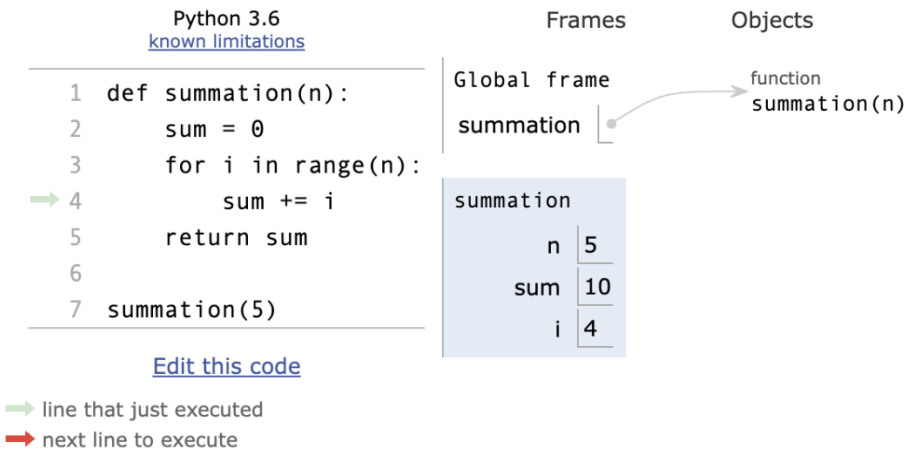


Figure 4.2: A modified screenshot from Python Tutor depicting a method, *summation(n)*, which sums up the values of 0 to n-1 and returns the summation.

4.4 Example CDQs

This section outlines two examples of Code Diagram Queries. Each sample includes a screenshot from Python Tutor and a set of questions I can ask about the image and the solution to those questions. Additional CDQs may be found in Appendix I.

4.4.1 Example 1: Analyzing Temporal Flow

A question we could ask about Figure 4.2 is a short answer, written response style question. This question focuses on the Path of Execution programming concept, targeting the *understand* level of Bloom's taxonomy. Here, we are asking a student to identify where the computer will execute next without asking for a deeper understanding of why. *What line should the red arrow be on? That is, which line will run next?*. To solve this problem, a learner would first need to look at the code portion of the screenshot and notice where the green arrow is. The green arrow indicates that line 4 finished executing when this snapshot was taken. Looking at the visualization of the frame, a learner can see that $n = 5$, telling them that this loop has executed for each value of i in 0-5 (exclusive). Since line 4 has just finished executing, it follows that **line 3** would run next, **because a computer must go back to the top of the loop and check to see if another iteration is required**. As

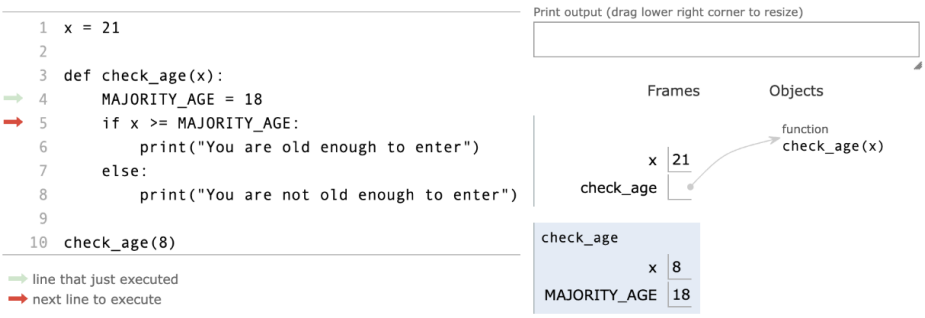


Figure 4.3: A modified screenshot from Python Tutor depicting the function, *check_age(n)*, which determines if an inputted age is above the specified age of majority.

humans, we can see that $n = 5$ and $i = 4$ and since $0 <= i < n$, another iteration is unnecessary. However, the computer must check this condition to know this, and that happens by continuing to attempt to execute the loop and going back to line 3 where the condition will be checked and found to be false.

4.4.2 Example 2: Applying an Understanding of Structure & Scope

When considering Figure 4.3, we can ask a question about the scope of the variable x. Consider a multiple-select style question: *When looking at the given screenshot, what can we say about the variable x?* and the possible options are as follows: [(A) x is globally scoped, (B) x is locally scoped, (C) There is an error in the program, (D) x is defined twice, (E) x is defined once and updated]. For learners to select all correct options, they must understand that the current place of execution is irrelevant, given the possible options. When looking at the defined frames, we can see that the variable x appears twice, meaning it is defined twice, making (D) a correct selection. Although the words 'local' and 'global' do not appear in the visualization, a learner must recognize the terms and recall their definitions to know that A, B, and D are all true, because there are two x variables defined, one is global and one is local. This example falls under the cross-section of the Structure & Scope CS concept and the Apply level of Bloom's taxonomy. Asking students to map the scope to the variables is

considered an analytical level of understanding.

4.5 Summary

To summarize, this chapter describes the CDQ framework design and creation process. By targeting three cognitive processes and pinpointing programming concepts that cross-cut topics and languages, we designed a space where comprehension at a detailed level may be incorporated into formative assessments. The chapter also outlines two examples of CDQs using Python Tutor and walks through the problem statement a learner would see and the solution. In the next phase of my research, I used this framework to guide the creation of a set of Python Tutor CDQs. Then I deployed them in a post-secondary first-year programming class to evaluate their effectiveness as a learning tool.

Chapter 5

Course Context

To evaluate CDQ effectiveness, I conducted a deployment in an introductory programming course. In this chapter, I describe the context in which that deployment occurred. COMP 1012 is an introductory Python programming course for scientists and engineers, most commonly completed by non-computer science majors. In Winter 2023, this course had two sections. I implemented weekly formative assessments using CDQs in one section (CDQ) and equivalent non-diagrammatic formative assessments in the other (nCDQ). Both sections ran as flipped classroom environments where students had three hours of in-class time each week with their peers. Students worked on activities and tutorials facilitated by their instructor and a learning assistant. All course content was found on the university's learning management system (LMS) called UMLearn, including study materials, surveys, formative assessments, quizzes, tests and evaluations. Within UMLearn, students accessed weekly videos, problem sets, and slides. While UMLearn is the university's LMS, its foundation is BrightSpace [79], a cloud-based learning platform that various institutions and educators use to manage their courses. I used UMLearn's built-in assessment tools to implement the formative assessments and distribute them to the students of the two sections of COMP 1012. I used the built-in quiz functionality as the basis of these assessments. As described the section 5.1, these assessments were graded on completeness, rather than correctness.



Figure 5.1: A screenshot of the student view of the formative assessments. In each week's content folder, a student would see the practice problem set they were to submit that week, labelled with their enrolled section and the current week number. The section has been intentionally blurred.

5.1 Grade Breakdown

The grade breakdown for the course is as follows:

- Weekly labs: 10% total
- Weekly in-class quizzes: 10% total
- Four assignments throughout the term: 12% total
- Final examination: 30% total
- In-class participation: 5% total
- Weekly formative assessments: 5% total

In order to receive the full 5% in each case of the in-class participation and the weekly formative assessments, a student had to complete a minimum of 80% of the assigned tasks throughout the term (Algorithm 1). I use this formula to attempt to mitigate the concern of missing a week or two of work. These assessments are supposed to be low-stakes, and students often have busy schedules. Students were reminded regularly that the formative assessments were scored based on submission, not correctness. The weekly assessments were labeled with their week number and the term 'Practice Questions' rather than the more intimidating 'formative assessment' (Figure 5.1). A breakdown of the course schedule by major milestones and topics can be found in Figure 5.2. For additional context, the course syllabus can be found in Appendix J

Algorithm 1: Formula used when calculating the student's formative assessment participant grade. A maximum of 5% can be earned.

```
submitted_assessments ;          /* number of weekly assessments submitted,
 regardless of correctness */
total_assessments ← 11 ;          /* number of weekly assessments that were
 administered throughout the term */
if submitted_assessments/total_assessments ≥ 80% then
    student_grade ← the full 5% for assessment submission
else
    student_grade ← submitted_assessments/total_assessments * 5%
end
```

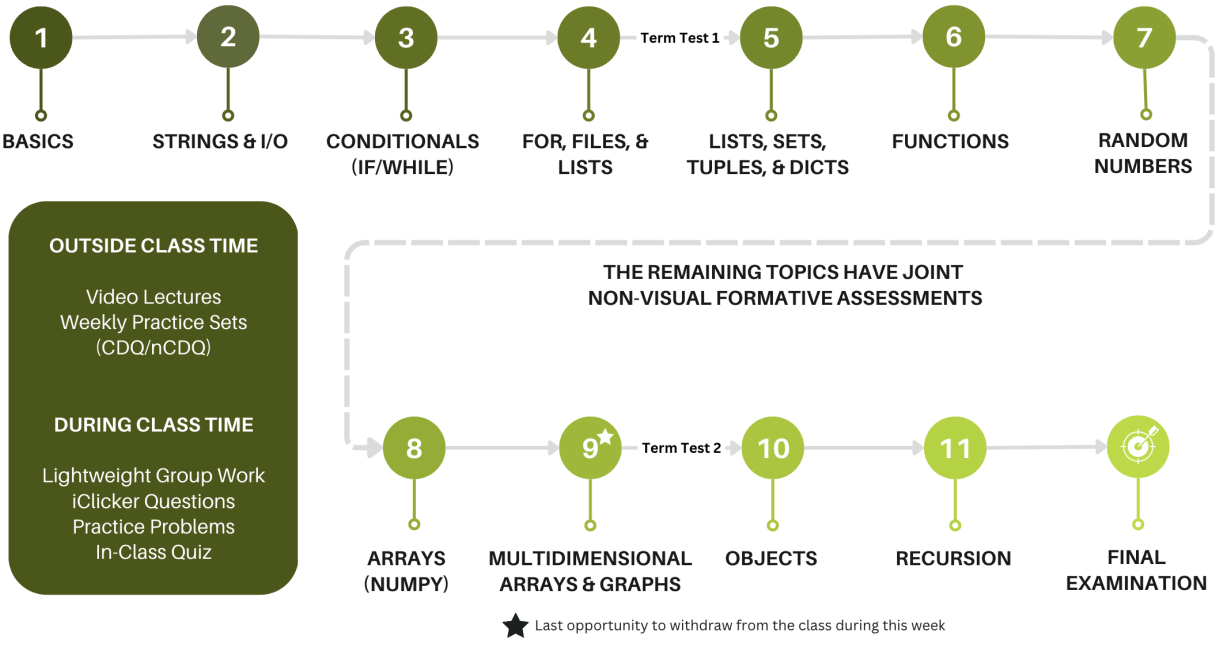


Figure 5.2: Visual flow of the COMP 1012 class by week, showing each week's topic. The summative evaluations, important cut-off dates, and student weekly actionable items are highlighted. CDQs were only used in topics 1-7.

5.2 Formative Assessment Materials

Students were provided additional optional resources: practice problems and embedded access to the Python Tutor visualizer. The CDQ section's weekly formative assessments were based on visualizations of coding examples in Python Tutor, while the nCDQ section had non-visual questions on the same topics. Both sections received practice problems with accompanying code; however, only the CDQ section included the PT visualization. Students in both sections were made aware of Python Tutor at the beginning of the course, and it was embedded into UMLearn directly under course resources. Thus, all students could use PT to work through problems throughout the term.

5.3 Course Director: Role and Responsibilities

I was the course director for COMP 1012 in the Winter of 2023, and this role required that I guide and support the teaching of the two instructors hired to run the classes. This position also meant that I had no direct interactions with students. This helped avoid conflict of interest but did introduce limitations such as limited control over the classroom environment. I had no grading responsibilities, nor was I required to create any summative evaluations. My primary role was to act as a guide and support person for the instructors when they were creating assignments and tests, interacting with students, or if they experienced administrative issues. I also guided the instructors to use a flipped class approach and provided pedagogical support. This helped with consistency because both sections of students received the majority of the course content by watching the same set of pre-recorded videos that had been created by a former instructor. While the instructors were of varying levels of expertise and came from different backgrounds, the material taught and summative evaluations were identical between the two sections. The only difference between the sections is the weekly formative assessment questions (CDQs and non-CDQs). The instructor of the CDQ section was a first-time instructor, new to the flipped classroom dynamic. They had recently com-

pleted their masters degree in computer science and started working in the computer science industry. The non-CDQ section instructor has been a sessional instructor for a number of years, having completed their PhD in 2014. While this individual has taught for a number of years, this was the first time they had taught this course and the first time they have taught a class with this many students.

5.4 Summary

This chapter outlined the COMP 1012 class in which I conducted my study. I described the 11 topics this class covered and highlighted the seven foundational programming topics the formative assessments centred on each week (Figure 5.2). I presented the course evaluation breakdown and the general course configuration. I described the classroom environment and the roles and responsibilities of the two instructors as well as my role as course coordinator. The next chapter will describe how I implemented my study in this course, outlining the recruitment and consent procedures, data collection and planned analysis methodology.

Chapter 6

Methods

To answer my research questions I:

1. Developed a set of CDQs using the framework and piloted them in a class in the Fall of 2022.

2. Deployed the question set in a first-year programming class, COMP 1012, in the Winter 2023 term.

3. Collected and analyzed data from the consenting students in COMP 1012.

4. Conducted and analyzed data from semi-structured interviews to gain student perspective on the weekly formative assessments and the CDQs.

The primary goal of the CDQ framework is to guide the creation of CDQs which support students in their understanding of how code interacts with computer architecture, thereby enabling them to build up increasingly complete mental models. as they learn more programming concepts This section presents how I deployed CDQs in a first-year programming class and investigated the use of these questions through weekly formative assessments. Initially, I created a set of CDQs and piloted them in an introductory programming class, as described in the next section. Once that class ended, I took the feedback received and altered the questions appropriately, mapping them to the learning objectives in the COMP

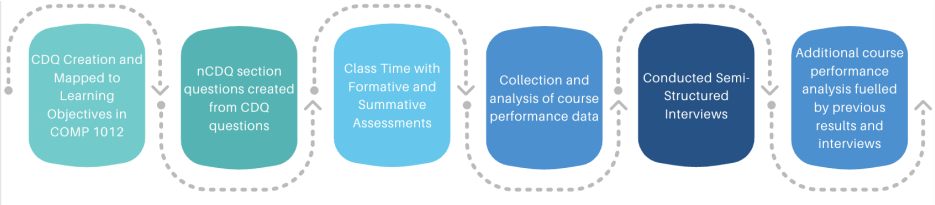


Figure 6.1: An overview of the methods described in this chapter. Starting with CDQ question creation and finishing with analyses of performance and interview data.

1012 class. This resulted in CDQs which spanned the first seven weeks of the COMP 1012 course. Next, I created problem sets for the nCDQ section based on the questions in the CDQ section to ensure an equivalent concept coverage. Following the question creation, COMP 1012 commenced in January 2023, and students worked through these weekly assessments. Students wrote their first term test in February 2023. This was the only term test I deployed a question on with the intention of assessing their understanding of one of the framework cells (Analyzing the Flow of Execution), thereby indirectly evaluating their mental models. At the end of the seven weeks of foundational concepts, I collected the responses and scores from the weekly formative assessments for both sections and their term test grades. I then performed a preliminary analysis surrounding the course performance data. Near the end of the term, in April 2023, three semi-structured interviews took place with students from the CDQ section. I performed additional course performance analysis based on the content analysis of these interviews. Finally, in May 2023, following the course's final examination, I reviewed the usage logs from the formative assessments for each section, looking for information on additional attempts and final grades. A workflow visualization can be found in Figure 6.1, and the analytical approach to using the collected data to answer my research questions is found in Table 6.1.

6.1 Creating Course Content

An initial draft of CDQs was deployed between July and November 2022 in the introductory programming course SCI 1002: How to Think Like a Computer. Course content for SCI

Data Source	Analysis	Research Question
Term Test Scores	Inferential Statistics	Accuracy
Targeted Term Test Question	Inferential Statistics	Accuracy
Final Term Scores	Inferential Statistics	Accuracy
FA Rate of Correctness Average	Inferential Statistics	Accuracy
FA Rate of Correctness by Question Type	Inferential Statistics	Accuracy
FA Rate of Correctness by Concept	Inferential Statistics	Question Dimension
FA Rate of Correctness by Cognitive Process	Inferential Statistics	Question Dimension
Usage Statistics: Immediate Repeated Attempts	Inferential Statistics	Engagement
Usage Statistics: April Repeated Attempts	Inferential Statistics	Engagement
Semi-Structured Interviews	Content Analysis	Engagement

Table 6.1: This table outlines the various data sets I collected, how I analyzed them, and which research questions and exploratory pathway this analysis aimed to answer. FA is shorthand for formative assessment. For reference, (1) How does using CDQs impact the accuracy of students' mental models? [Accuracy], (2) How do CDQs impact student engagement with provided optional resources? [Engagement], and (3) An exploration into the dimensions which affect student experiences and performance with CDQs. [Question Dimension]

1002 was developed in the Summer of 2022 by Dr. Celine, an undergraduate student, Carter Lupkowski, and myself. While designing this new course, Carter and I created 3-5 end-of-chapter questions each week for students to assess their understanding. Each question required the use of a Python Tutor image of a program mid-execution. The initial iteration of these questions was implemented into the SCI 1002 Interactive Textbook in Runestone Academy [80] (Figure 6.2), which has Python Tutor built-in as 'Codelens' questions.

The original plan was to implement these questions in SCI 1002 in Winter 2023. However, we realized that this class would need more students enrolled to produce valuable results, and my plan pivoted. I decided to deploy my questions in COMP 1012, the first-year Python programming course for science and engineering students who do not pursue a Computer Science Major/Honours degree. This shift meant I reorganized my initial questions to fit into the topic structure for COMP 1012. Once this was completed, I saw that in a 12-week term, we had seven weeks of questions that were tested using CDQs. Since SCI 1002 is meant to be for students with no prior experience, the materials covered in that course did not map to the entire 12 weeks of COMP 1012. The remaining five weeks in COMP 1012 cover

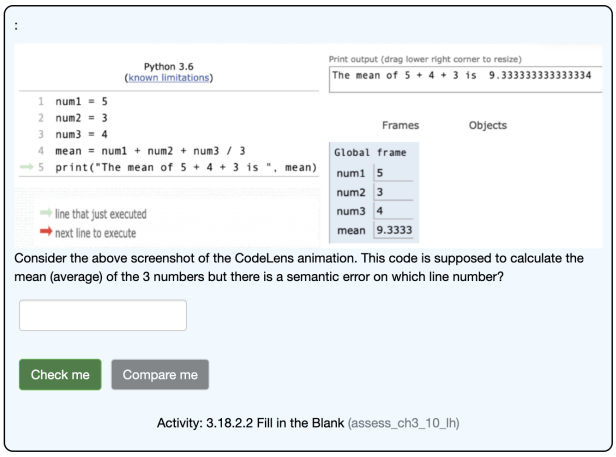


Figure 6.2: Example of a question in the initial implementation of CDQs in the Runestone Academy Textbook for SCI 1002.

advanced introductory topics and Python libraries that are not compatible with Python Tutor. The final set of CDQ questions which I used in the first seven weeks of COMP 1012, can be found in Appendix I. My final set of CDQs uniquely required students to refer to code displayed in a PythonTutor diagram and would be more challenging to answer without using the provided diagram.

After the questions had been ported over, I created equivalent questions in the non-CDQ section. I accomplished this task by looking at each CDQ I was planning to present to the students, identifying the learning goal as per the framework and then writing a similar non-visual question. An example of this mapping is found in Appendix I under topic 1. These questions would then span over the first eight weeks of the course (the first week was a syllabus-focused assessment to become familiar with the assessment quiz structure on the LMS, and the following seven weeks were using the CDQs).

6.2 Experiment: Classroom Implementation of CDQs

In the Winter 2023 term, there were two sections of the COMP 1012 course. Two instructors were hired to teach this course, and I acted as the course director, ensuring consistency between the two sections and overseeing the implementation of my formative assessments.

Both sections of this course employed a flipped class, active learning pedagogy, with direct instruction on content delivered through weekly required videos in UMLearn. These videos were available to all enrolled students. As course director, I provided additional resources, including other practice questions and the embedded Python Tutor tool. Both sections were administered the same summative assessments throughout the course, including the same final examination. One section had weekly formative assessments with CDQ questions (CDQ); the other had similar non-diagrammatic formative assessments (nCDQ). UMLearn's ability to restrict access to materials by section allowed me to implement CDQs in just one section and investigate my research questions.

6.2.1 Participant Recruitment

All students enrolled in COMP 1012 in Winter 2023 were asked to consent to collecting and analyzing their data for my research. I attended a class for each section during the third week of the term to describe my study and distribute consent forms (Appendix C. I chose to distribute the consent forms early in the term rather than waiting until I planned to collect the data to gather a more diverse set of consenting students. The consent form can be found in Appendix B. As the term progressed, students may begin to feel self-conscious about the grades they are achieving, and therefore, those who were doing poorly may be less inclined to consent to collect their data. This approach also enabled me to collect data from students who might have dropped the course. I believed I would gain valuable insight from as much data from students as possible, rather than just those that made it to the final exam.

Following the consent form distribution, I contacted students who were not present via email so they would also have the opportunity to consent to me analyzing their data for my research (Appendix E and D). On the form for the CDQ section, students were asked to indicate their willingness to be contacted for a follow-up interview. To determine the order of priority for interviews, I grouped the consenting students by CDQ usage at the end of the term. Starting with the highest usage group, students were contacted to be interviewed.

6.2.2 Weekly Formative Assessments

Students were required to complete weekly UMLearn formative assessments. Students in the CDQ section received questions correlating the course materials to diagrammatic visuals from PT. For the seven weeks of foundational material, each question set reflected that week's objectives with questions in multiple formats, as discussed in section 4. It was pertinent that the questions not just allude to the code of the program but instead use the visualization of machine execution provided by Python Tutor. This includes Python Tutor's ability to step through code and ask questions regarding mid-execution values and its ability to aid students in visualizing differences in variables, modules, and objects.

Students in the nCDQ section also completed weekly formative assessments. These questions were based on the same learning objectives in the CDQ assessments. However, they did not contain a diagrammatic component. These formative assessments included code or possible output from code without the added visualization of how the data was being stored or what the values looked like mid-execution for a piece of code. Both sets of questions contained feedback on incorrect answers. This feedback included explanations of why the answer they chose was incorrect and a suggestion for the student to navigate to Python Tutor or their Python code program to walk through the code and try again. Potential additional examples were also included in the feedback, as appropraite, so students would be able to attempt the problem using a different subset of values.

While UMLearn has a formative assessment tool called "self-assessments", the question sets for both sections were distributed in a quiz-like format as the self-assessment tool was inappropriate for my situation. I wanted students to be able to see what they did wrong and how they could fix it, which required them to submit the assessments in a quiz-like format. The formative assessment tool had limitations surrounding the data I could collect and the granularity at which I could give feedback. Thus, I used the quiz tool, although I set up each assessment so that no grade was assigned for correctness. Students were informed multiple times that the grade shown was merely for their understanding and would not affect their

final score. Once the set of questions was attempted, they received participation points for their attempt. The assessment told them how many questions they answered correctly but did not provide immediate information as to which questions were right or wrong. This level of feedback was designed to encourage repeat attempts in full.

Each week the students were given seven days to attempt the question set. No marks were deducted from students who answered incorrectly; all they had to do was try the assessments. After seven days, the correct answers were released, and additional feedback was given. Assigning a participation grade to these assessments ensured their use while keeping the interactions as stress-free as possible. If a student was interested in solving the problems before I released the correct answers, it was relatively simple for a student to recreate the code themselves on Python Tutor. Students wishing for instant correctness had to engage with the code in a more hands-on manner, helping them further their understanding. This process was encouraged but not required. The delayed feedback was intentional, and research shows that the more students practice coding and manipulating code, the stronger their resulting understanding would be [52, 53].

6.3 Summative Evaluation of Mental Model

While any question where a student explains their understanding of code execution or concepts requires them to draw on their inner model of the topics, I had created a single question which specifically targeted the students understanding of execution flow of a program using for loops. This question was worth 2 marks with very limited variability in correct answers. When creating this question, I intentionally chose a non-visual style question to ensure fairness between the sections. The nCDQ section had not had regular exposure to Python Tutor and using it in a summative assessment felt unfair. Ultimately I realized, by choosing to include one question using one of the two question styles, one section would be at a disadvantage. By choose a non-visual question, the CDQ section students may have struggled to

comprehend the question full. In a future iteration, I would include one of each question style on summative assessments.

6.4 Data Collection

The data collected in my study includes both quantitative and qualitative components. Quantitatively, course performance logs were collected. These included the content of their attempts on the weekly assessments, their summative grades throughout the term, and their final grades. As part of the initial consent form, we asked participants to indicate whether they would participate in an interview toward the end of the term. I conducted semi-structured interviews with several students from the CDQ section. I grouped the consenting students based on their level of engagement with the CDQs, so I could engage in purposeful recruiting of students who had both high and low levels of engagement with the materials. The interviews were approximately 30 minutes long and provided an opportunity to discuss CDQ usage, structure and experience with the students.

6.4.1 Course Performance Data

CDQ usage data came from UMLearn. UMLearn tracks how students interact with the quiz tool across a few axes. At the highest level, UMLearn records assessment attempts. An extensive list of each attempt in chronological order is displayed, and a grading distribution is shown, among other statistics. Course performance data included student grades from each grade item in the course: weekly quiz marks, lab grades, assignments, summative evaluations, and formative assessments.

Focusing my attention on the formative assessments, on a student level, I could see details such as how many times a student attempted the assessment. Data at this level include the date and time of the attempt, the length of the attempt, and the "grade" achieved on the attempt. For students who attempted the assessment more than once, this data is available

for each attempt. Finally, I had access to the most granular data within a given try: the submitted responses to each question.

6.4.2 Interviews

Interviews were conducted on UM-licensed Zoom between me and individual students. Each interview was approximately 30 minutes long. Interview audio was recorded and manually transcribed to be coded, grouped, and analyzed. Students were compensated with a $15 Tim Horton's gift card for their time. Given the number of consenting students in the CDQ section, I hoped to interview approximately eight of them.

The interviews were semi-structured, roughly 30 minutes in length. We discussed the general course structure, material, and their study habits. I showed the participant a CDQ during the interview, and we reviewed it together, allowing them to share their thoughts on the CDQ structures and efficacy. Finally, we discussed Python Tutor and their usage of the tool.

6.5 Data Analysis

Using a variety of existing learning, analytical, and statistical methods, I analyzed the usage rate of the formative assessments (weekly practice problem sets). I looked for trends in the accuracy of student understanding of the Python notional machine and how these low-stakes assessments, alongside the use of the additional optional resources, affected their course grades at a granular level (not the overall final grade but on a grade item-by-grade item basis). I was looking for any impact the usage of these questions had on students' performance as a proxy for their mental models.

I intended to engage in a thematic analysis of eight interviews. The interviewees would be participants grouped by engagement with the weekly assessments. Due to a lack of responses, I only completed three interviews, and my qualitative analyses became a set of mini-case

studies following a content analysis of each interview.

6.6 Pre-Processing Data Logs

Before performing any analysis, I preprocessed and cleaned the data logs exported from UMLearn. The reason for preprocessing is two-fold. Pre-processing this data was a way to mitigate any inaccuracies or outliers in the participants. There were instances in my data where students would include blanks or random characters. This was because completing these formative assessments contributed only to the student's participation grade, and no importance was placed on answering correctly. Some students submitted garbage data, including empty strings or random characters. I added logic in my preprocessing code to filter out these responses. Through a few simple rules, I removed these outliers. These rules included checks for students who may have had the weight from their term test move to a later evaluation or vice versa to attempt to mitigate any oddities in the data. I considered participants with many zeroes and no/a zero-term test score as having withdrawn from the class. I removed them from my summative analyses but still included their responses for attempted weekly formative assessments.

6.7 Summary

In this section, I described the methods used to design and implement my study, including the various pivots I had to make. I created CDQs designed using our CDQ Framework and deployed them in a first-year computer science programming class as weekly formative assessments. After completing the first seven foundational weeks, I collected the student data from the learning management system and began to analyze student performance. Near the end of the term, I recruited and interviewed students who took part in the CDQ section of this class to gain insights into their thoughts on the course structure and the use of the CDQ-based weekly formative assessments. In the following section, I present the results of

my research.

Chapter 7

Results

When beginning my study design, I had two research questions I wanted to answer: How does the use of CDQs impact the accuracy of mental models, using course performance as a proxy? How do CDQs impact the level of engagement with provided optional resources? And I wanted to explore how different questions within the framework (concept, cognitive level and question style) affected student performance and experience. In this chapter, I describe my study's main findings, reporting on my participants' demographic data and the general understanding of their background with programming as well as detailing the quantitative analysis I performed to attempt to answer each of my research questions. I also report on the three interviews with participants in the CDQ section at the end of the Winter 2023 term.

7.1 Consenting Participant Information

In the third week of the 2023 winter term, I collected 304 consent forms from the two sections of COMP 1012. Each class had more than the maximum number of enrolled students in the room due to additional individuals such as note-takers, wait-listed students, and students auditing the course. This resulted in a slightly higher number of collected consent forms than the number of seats in the course. There were 151 consent forms collected from the CDQ

	Basics	Strings & I\O	Conditionals & While	For & Files	Iterable Objects	Functions	Random Numbers
CDQ	100	84	83	77	58	70	66
nCDQ	94	94	92	89	69	85	82

Table 7.1: The number of formative first attempts per topic (February 2023), separated by section, at the time of the data export (February 2023), approximately seven weeks into the course.

section (CDQ) and 153 from the non-CDQ (nCDQ) section. Of these, 82 students (54%) in CDQ and 96 students (63%) in nCDQ consented to collecting and analyzing their data for this research, totalling 178 consenting students. The remaining forms collected were from students whose responses indicated that they did not consent to their data being collected and analyzed for this research.

I had 49 students from the CDQ section who consented to be contacted for a follow-up interview. I hoped to interview approximately eight students from the CDQ section; however, likely due to the timeline of this study aligning with final examinations, only some students responded to the recruitment email. Four students agreed to be interviewed, with three participants completing the interview.

7.2 Overall Engagement with Formative Assessments

Following the seventh week of classes, an initial export of student course performance data was completed with submission numbers found in Table 7.1. When considering all attempts made by students for a given week's assessment, I report a maximum of 100 submissions in a single week (Basics: Week 1) and a minimum of 58 submissions (Lists, Tuples, Sets, and Dicts: Week 5) for the CDQ section and a maximum of 94 submissions (Weeks 1/2) and a minimum of 82 submissions (Random Numbers: Week 7) for the nCDQ section.

The number of submitted formative attempts varied weekly, as only some students submitted every assessment. As the term progressed, students withdrew from the course, af-

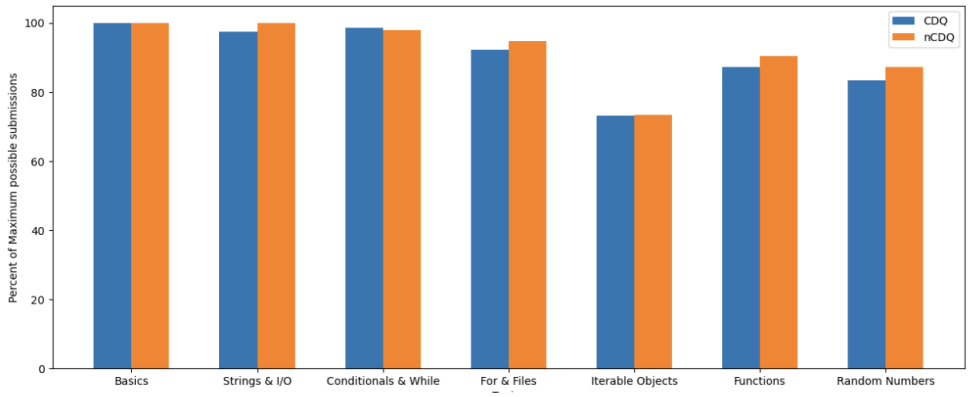


Figure 7.1: First Attempt Percentages: % of students who submitted at least one response for a given week of the maximum possible submissions for a given week.

fecting the number of attempts to analyze. Figure 7.1 shows the number of consenting participants who submitted a first attempt each week.

7.3 Course Performance as a Proxy for Mental Model Evaluation

The first research question I wanted to answer was *How does using CDQs impact the accuracy of students' mental models, using course performance as a proxy?*. As I could not directly probe the inner mental models forming inside the participant's brains, I used course performance as a proxy for mental model assessment. I incorporated a question on the first term test specifically designed to test the student's understanding of a computer's interaction with a Python for loop, but without a visualization (Figure 7.2). I then used their overall term test scores and their scores on this question to look for differences between the two sections.

7.3.1 Term Test One Scores

I looked at the overall term test performance of consenting students. I saw no visual difference in the overall term test grades (Figure 7.3), further solidifying this with the inferential

Question 1: Flow of Execution [2 marks]

Before the final print command on line 6 is executed for the code below, what **line number** is executed in the code and **why**?

```
1 str_greeting = "Hi there!"
2
3 for character in str_greeting
4    print(character, end="") # prints each character
5
6 print("All done")
```

Figure 7.2: Term Test 1 question added which specifically tests a student's understanding of what a computer is doing with a snippet of Python code.

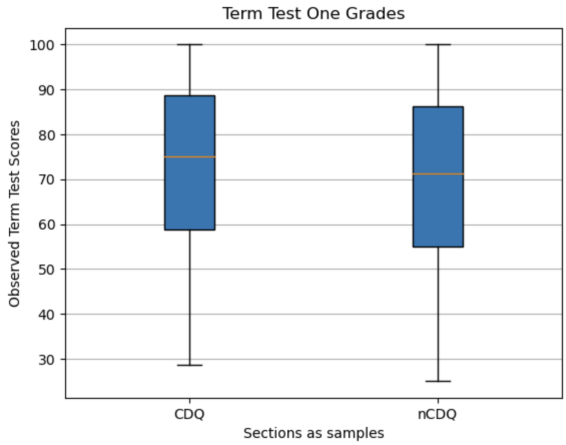


Figure 7.3: Term Test 1 scores of consenting students.

statistics I calculated (Table 7.2 and ($t = 0.5594, p = 0.5767, df = 173$)). While this gave me some insight into students' consistency and abilities between the sections, it did not show any statistically significant impact of the formative assessment intervention.

Section	mean	std
CDQ	71.667%	20.182
nCDQ	69.958%	18.659

Table 7.2: Descriptive statistics from term test 1 scores (as percentages).

Section	mean	std
CDQ	30.555%	44.790%
nCDQ	30.555%	45.069%

Table 7.3: Descriptive statistics from question 1 on term test 1 (as percentages).

7.3.2 Term Test One: Targeted Analysis

I then considered the marks on the question I designed to test students' mental models. I created a single question which specifically targeted the students' understanding of the flow of execution with for loops. I designed this question without a visualization to be sure to avoid bias in the question in favour of the CDQ section. This question was worth two marks with minimal variability in correct answers. I saw no difference in abilities between the sections (Table 7.2 and ($t = -0.4006, p = 0.6892, df = 173$)). The average grade for each section hovered around 30%. More students got this question wrong which lowered the average given its all-or-nothing nature. Students did not seem to understand that the code would go back up to the top of the for loop to check for additional iterable elements, misunderstanding how the flow execution worked.

As measured through course performance, CDQs neither helped nor harmed mental model development. Student scores were consistent between sections which suggest that CDQs had a little direct impact on students' summative scores and overall understanding of the concepts presented.

7.3.3 Final Examination Analysis

In May 2023, after the final exam for the class, an additional export of data logs was performed and analyzed. When analyzing the final exam scores, there were no significant differences between the two sections (Figure 7.4 and ($t = -0.5673, p = 0.5713, df = 164$)). I did not include any specific questions to probe student's mental models on the final examination as the final exam had an emphasis on later topics and there were several weeks of topics toward the end of the course where I did not implement CDQs due to the more advanced nature of topics covered.

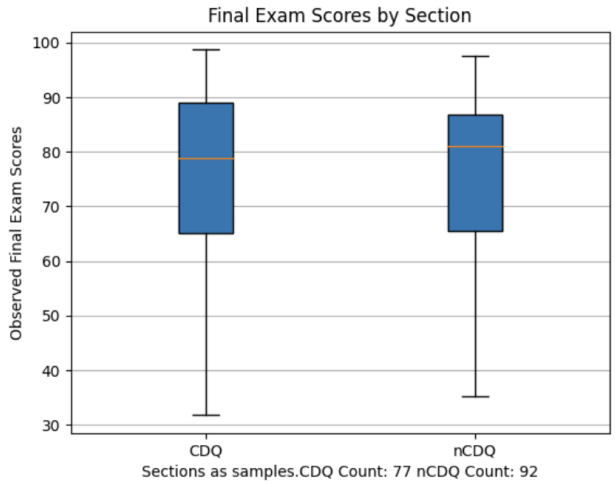


Figure 7.4: Box plot of the final examination scores for consenting students

7.3.4 Formative Performance Analysis

I also investigated students' first attempts at the formative assessment activities, as these provided a lens through which to analyze students' initial comprehension of a question and topic and any impact the visualization may have on immediate understanding. I speculated that more straightforward topics would have higher correctness rates, and as the term progressed and topics became more complex, the average correctness rate would vary more. Since the summative scores showed course performance consistency between sections, if the CDQs had no impact on understanding the learning objectives, variability in correctness by topic between the two sections would be minimal.

7.3.4.1 Overall Formative Assessment Scores

I began this level of investigation by looking for significant differences in the correctness of weekly first attempts for participants in each section (Figure 7.4). Considering the first attempts averaged across all seven topics ($t = -0.934, p = 0.351$), I found no statistical significance.

Next, I considered differences in performance at the weekly/topic level. To explore this possibility, I created box plots for each week to visually inspect the distribution of scores

	Basics	Strings & I\O	Conditionals & While	For & Files	Iterable Objects	Functions	Random Numbers	Average	Std
CDQ	84.777	76.102	50.464	35.301	64.444	70.931	75.087	65.3166	25.1984
nCDQ	78.411	87.383	56.300	42.603	63.198	73.839	62.915	66.6974	23.6625

Table 7.4: Weekly average correctness for first attempts by topic, all question types included. Note that Iterable Objects includes: Lists, Tuples, Dicts, & Sets

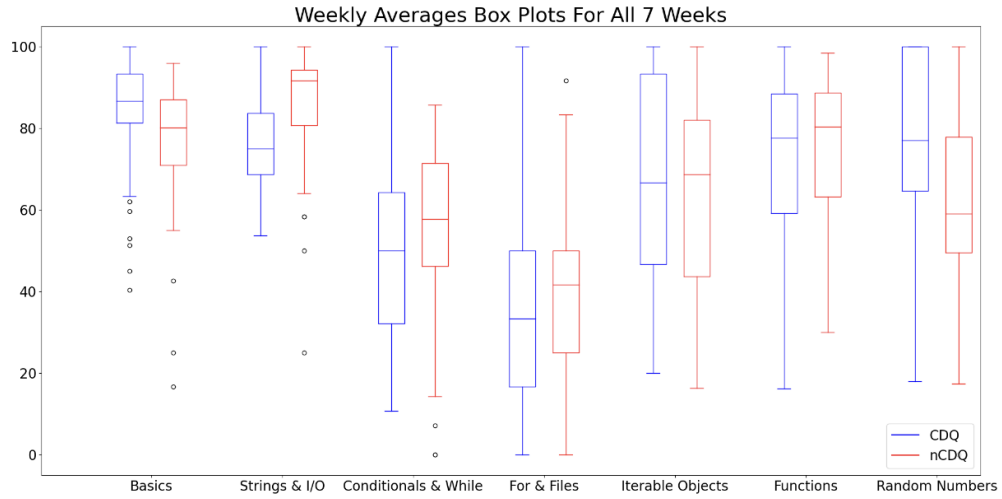


Figure 7.5: Box plots of first attempt scores for each week using the original data of all questions in each week. This includes answers to written response questions, which students from both sections did poorly on.

and how they might vary by section and week (Figure 7.5). I also report on the mean of each week by section (Figure 7.4). I wanted to visualize this score in a different setting and consider their letter grades. I took the students' grades by the week and graphed them using a divergent bar chart. Using the grading schema found in Appendix J, I assigned a letter grade to the participants' first attempt scores (Figure 7.6). These analyses showed the same general understanding of the data from the box plots, highlighting the same topics having significant differences by sections. This finding led me to start dissecting the weekly student logs.

The weekly formative performance results suggested that there were likely to be meaningful differences in student performance between sections for the Strings & I\O and Random Numbers topics (weeks 2 and 7, respectively). When I further dissected the significant dif-

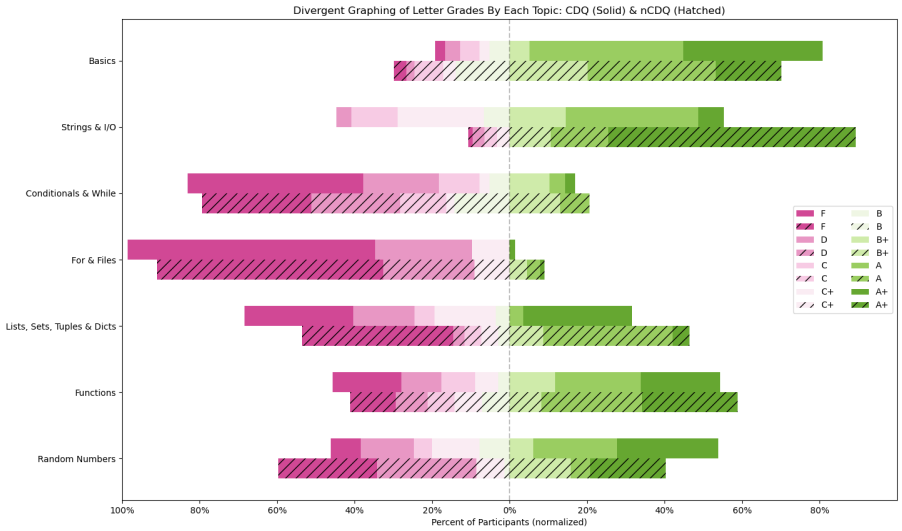


Figure 7.6: Grouped by section, the letter grade breakdown for each weekly formative assessment, considering only first attempts. This data set includes both MSMC and WR type questions.

ference in the Strings & I\O topic (week 2), the data revealed that students in both sections performed poorly on the written response style questions. The number of written response questions each section completed was unequal (Table 7.6).

7.3.4.2 Multiple Choice Formative Assessment Scores

Knowing recall style questions are typically more challenging for students than recognition-based question styles [81], I decided to rerun the first attempts of correctness analysis on the weekly formative assessments without the written response questions. I further strengthen the justification for this analysis by considering what I learned from the interviews (described in Section 7.6) that the students did not put much effort into answering WR questions on these formative assessments because they took too long and would submit blank answers. Results for this analysis are shown in Figure 7.7 and Table 7.5. These box plots and means show slightly different results. The Conditionals & Whiles and Iterable Objects (Weeks 3 and 5) display significant variations in means, instead of the Strings & I\O topic. The Random

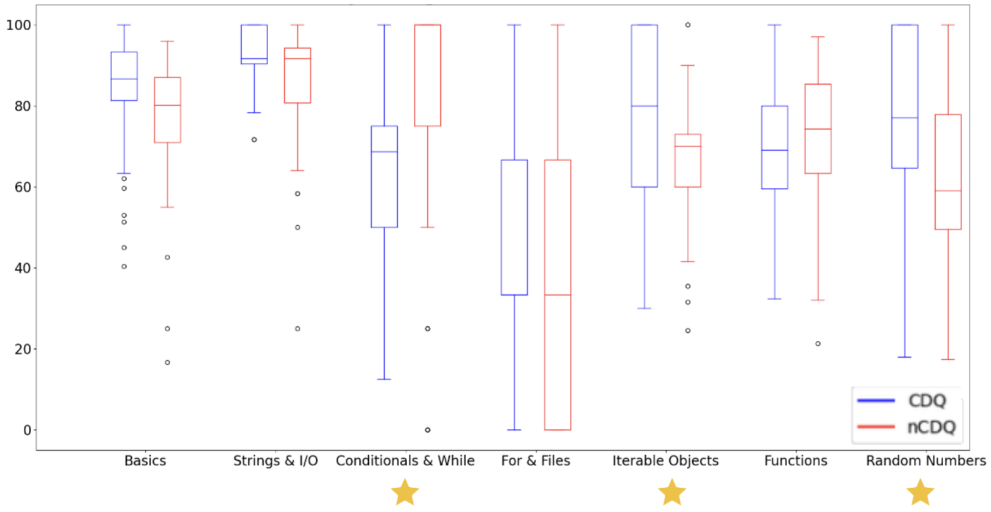


Figure 7.7: Box Plots of first attempt scores for each topic, omitting the written response (fill-in-the-blank/short answer) questions. Yellow stars on the x-axis indicate statistically interesting topics.

	Basics	Strings & I\O	Conditionals & While	For & Files	Iterable Objects	Functions	Random Numbers	Overall Average	Overall Std
CDQ	84.777	92.259	64.610	43.519	78.246	69.314	75.087	72.5828	14.6051
nCDQ	78.411	87.383	80.707	37.079	65.812	71.6	62.915	69.1294	15.2894

Table 7.5: Weekly Average correctness for first attempts by topic, considering only the multiple choice/multiple select questions. t-test results by section comparing all topics: $(t = 0.39563, p = 0.69931, df = 12)$. Note that Iterable Objects includes: Lists, Tuples, Dicts, & Sets.

Number topic continued to show a substantial disparity between sections as there were no written response questions in that topic. Given the adjusted data set, no such differences emerged for the remaining weeks. I further explore this weekly difference when investigating question type differences in Section 7.5.2.

7.3.4.3 Outlier Detection: Conditionals & While Loop

Given the results reported on MSMC style questions, I further investigated the differences in student performance given the questions in a specific week. Table 7.6 displays the number and types of questions each section saw in their assessments for each topic. The Conditional & While topic data is unreliable as the two sets of questions by section were heavily

	Basics	Strings & I\O	Conditionals & While	For & Files	Iterable Objects	Functions	Random Numbers
CDQ							
MSMC	3	3	4	3	2	3	3
WR	0	1	3	3	1	3	0
nCDQ							
MSMC	3	3	1	3	2	3	3
WR	0	0	6	3	1	3	0

Table 7.6: Topics broken down by category of multiple-choice/select (MSMC) and short answer/fill-in-the-blank (WR) for each section.

imbalanced, with the nCDQ section answering a single MSMC question compared to the 4 MSMC questions in the CDQ section. Upon reviewing this finding, this topic's results were no longer considered relevant as the difference between sections may be due to the question type differences.

Following this analysis, I report no difference in the summative indicators I measured as a proxy for assessing student mental models. Overall formative results suggest little differences for some topics but inconsistently for all seven weeks. Once I removed the written response questions, which had a low correctness rate across both sections, and the Conditionals & While Loop topic, we see a borderline significant difference between the sections ($t = 0.67813, p = 0.51307, df = 10$) suggesting students in the CDQ section performed better on the weekly multiple choice questions than the students in the nCDQ section.

7.4 Impacts on Engagement with Optional Resources

To investigate *How do CDQs impact the level of optional engagement with PythonTutor?*, I attempted to explore immediately repeated attempts on the formative assessments, the usage statistics from April 2023, during the time approaching the final examination in the course, and the engagement with the Python Tutor tool itself.

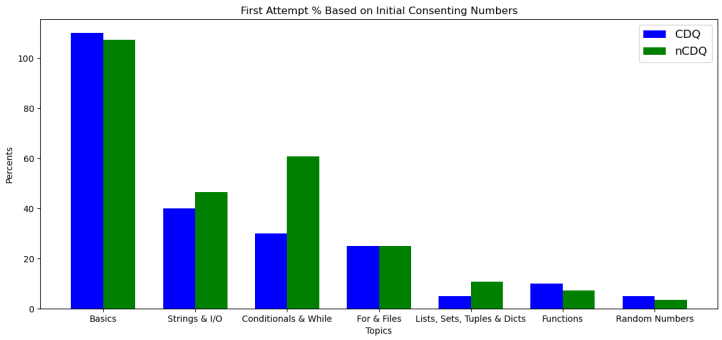


Figure 7.8: Subsequent Formative Attempt Percentages: % of consenting students who submitted at least 2 responses for a given week.

7.4.1 Immediate Repeated Weekly Attempts

I believed that immediately repeated attempts on formative assessment activities could indicate that students found value in that activity. I hoped to see more repeated attempts by students in the CDQ section, suggesting that they found the visualization-based questions to help develop or correct their mental models. Unfortunately, there was insufficient data to investigate this possibility (Figure 7.8). Given the number of participants with first attempts, the number of subsequent student attempts was insufficient. It is possible that this lack of secondary engagement resulted from only receiving an overall score on the assessment, and so students who did well enough just left the assessment instead of trying to figure out which questions they answered incorrectly. Due to a lack of immediately repeated attempts, this line of investigation was not fruitful for exploring my second research question.

7.4.2 Formative Assessments as a Final Exam Study Aid

The second place I looked for optional engagement was using the formative assessments as a study aid approaching the final examination. I found that each week in the CDQ section had additional attempts in April 2023, while only some weeks in the nCDQ section had further attempts, as seen in Table 7.7. Using a paired t-test, I found a borderline significant difference in these attempt numbers, $(t = 2.091290, p = 0.058441, df = 13)$, although my N

	Basics	Strings & I\O	Conditionals & While	For & Files	Iterable Objects	Functions	Random Numbers	Total
CDQ	6	3	4	3	2	1	2	21
nCDQ	3	0	0	1	2	2	2	10

Table 7.7: The number of formative assessment repeat attempts for each week that occurred, during the final exam study period (April 2023), organized by section. W# indicates the week number the repeat attempts were observed. Note that Iterable Objects includes: Lists, Tuples, Dicts, & Sets

is small compared to the class size.

7.4.3 Python Tutor Engagement

The third investigation I intended to conduct to find evidence of optional engagement was to gather logs from the participants on their access to and usage of Python Tutor. Unfortunately, this did not transpire. The learning management system, UMLearn, does not effectively track the navigation or time spent in an embedded or external link. When setting up the formative assessments, I ensured that the feedback given for questions referred students to the embedded version of Python Tutor, hoping to track how many students interacted with this link and the time spent there. Regrettably, UMLearn does not record these statistics, so this third line of inquiry into optional engagement could have been more fruitful.

Overall, only a tiny amount of evidence shows that students in the CDQ section engaged more with PythonTutor than required. However, in Section 7.6, I describe how the interviews I completed with three participants lend some perspective into how the students used PythonTutor in conjunction with the formative assessments.

7.5 Exploring CDQ Dimensions Through Student Performance

In addition to my two main research questions, I also wanted to explore the different dimensions of CDQs that may affect student performance. First, I wanted to explore the questions that most students answered incorrectly and investigate why that might be. Specifically, I wanted to focus on reattempts to see if participant responses improved over time. However, to explore this possibility, I would have needed more subsequent attempts than I observed (Figure 7.8).

While the first attempts did not show how a student's mental model grew over time, the formative assessment attempts did allow me to observe a participant's initial comprehension of a question and the topic being taught, so I focused on the first attempts and the rate of correctness in questions. I speculated that simpler topics would have higher correctness rates, and as the term progressed and topics became more complex, the average correctness rate would vary more. Since the summative scores showed course performance consistency between sections, if the CDQs had no impact on understanding the learning objectives, the variability in correctness by topic would be minimal.

Next, I examined the differences in recall versus recognition skill-based questions. Separating questions into multiple select/choice (MSMC) style questions and open-ended written response questions (WR) allowed me to take a closer look at these skill sets. Students often find recognition-based questions that provide a stimulus or cue that a student can use as a point of reference when answering a question to be easier than those that require them to recall information [69, 81].

The last dimension of CDQs I explored was along the axes identified in the CDQ framework. I categorized the CDQs that participants in the CDQ section saw into one of nine cells relating to the cognitive processes and overarching programming concepts. I used this breakdown to understand how the CDQs can support student learning and where student

difficulties may lie.

7.5.1 Recollection versus Recognition Skills

The CDQ framework we developed does not explicitly account for recognition vs. recall skill sets, but my analysis of RQ1 revealed a difference between these types of questions. I categorized MSMC-style questions into the recognition-based category. MSMC questions give students all possible answers when they look at the provided information, and those can be the cue they need to fully comprehend a question and remember how to work through the problem. Written response (WR) questions are more open-ended, leaving a larger space for possible correct answers. The open-endedness in this type of question forces students to rely on their understanding of concepts more heavily than MSMC, thus requiring them to use recollection-based skills to answer questions correctly.

To further understand this difference in performance, I graphed the student performance data by question type in a paired points box plot, creating a separate graph for each section. I averaged student scores across all multiple-choice/select questions (MSMC) and their scores across all written response questions (short answer/fill-in-the-blank) (Figure 7.9). I then created the same graph for the participants in the nCDQ section (Figure 7.10)). The graphs reveal a clear relationship with almost every participant performing more poorly on the WR questions than the MSMC questions. This was the case in both sections, but the relationship appeared more robust in the CDQ section.

When exploring the efficacy of CDQs, participants scored higher on the MSMC questions than on the WR questions. While both sections performed poorly on WR questions, likely due to the recall skills required to be successful with this question type, the CDQ section had more trouble with them (Figure 7.9). Similarly, when looking at the MSMC questions, while both sections performed better on this style, the CDQ section, on average, performed better than the nCDQ section. This difference may relate to using a visual aid as a solid cognitive cue for answering these questions.

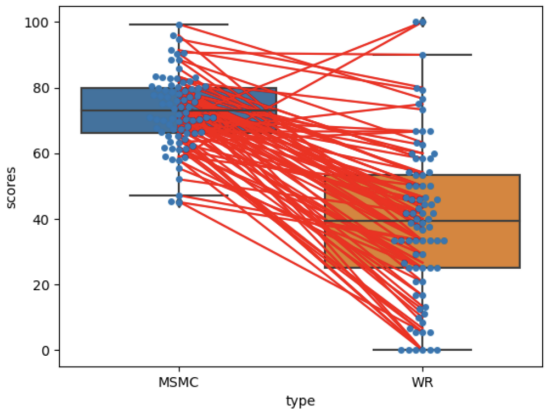


Figure 7.9: Paired box plot of the CDQ section showing average student performance on all MSMC style questions versus WR questions. Each point is a student's average mark across all 7 weeks of MSMC questions and WR questions. The red line pairs a student's average mark on the MSMC box plot with their average mark on the WR box plot.

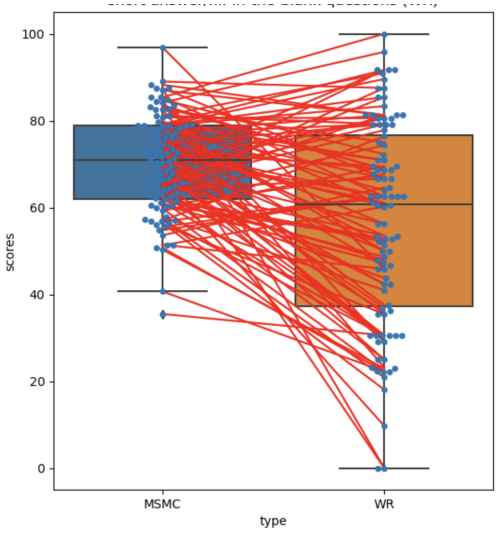


Figure 7.10: Paired box plot of the nCDQ section showing average student performance on all MSMC style questions versus WR questions. Each point is a student's average mark across all 7 weeks of the MSMC questions and WR questions. The red line pairs a student's average mark on MSMC box plot with their average mark on the WR box plot.

	df	sum_sq	mean_sq	F	PR(>F)
C(Section)	1.0	5499.275975	5499.275975	15.346095	$1.076354e-04$
C(type)	1.0	46022.000467	46022.000467	128.427450	$1.432079e-25$
C(Section):C(type)	1.0	9491.072828	9491.072828	26.485469	$4.435357e-07$
Residual	350.0	125422.564558	358.350184		

Table 7.8: Two-way ANOVA test of average scores across the two sections and considering question types (MSMC and WR). Results indicate significant difference between sections and between question types and a significant interaction between section and question type.

I performed a two-way ANOVA test to determine the effect on a participant's score based on their section and the question types discussed here. The results, shown in Table 7.8, indicate that the different question types statistically affected student scores. The p-value for the interaction effect (section*type) is also statistically meaningful. This means the observed interaction effect between the section and question type are most likely not due to chance. In this context, since the formative assessments in which the different sections interacted with the different question types was unique, a significant interaction makes sense. While both sections performed poorly on the WR questions, the CDQ section struggled more with these, possibly suggesting the visualizations and written response style questions did not work well together.

7.5.2 Computer Science Topics

I performed an independent two-sample t-test on data from each week to compare formative assessment scores in the CDQ and nCDQ sections on a computer science topic level. These tests were performed on the modified data set, omitting the written response style questions (Table 7.9).

The average mark was drastically affected by the type of questions for both sections. I report on weeks 1, 2, 5, and 7 having significant differences between the two sections and in all four, the CDQ section scores were better. This analysis suggests that CDQs help these

	t-stat	p-value	df
Week 1: Basics	3.0345	0.00279	170
Week 2: Strings & I\O	2.9041	0.0042	168
Week 3: Conditional & Whiles	N/A	N/A	N/A
Week 4: For Loops & Files	1.3683	0.17314	159
Week 5: Iterable Objects	3.83116	0.00020	124
Week 6: Functions	−0.9099	0.3643	151
Week 7: Random Numbers	3.2046	0.00166	145

Table 7.9: Inferential statistics on formative assessment scores for each week. T-tests run for each topic comparing sections on a topic level. Only multiple select/choice style question scores were included in the data sets. Topics with significant results are in bold. Week 3 data was omitted due to a question imbalance.

topics. The For Loops & Files topic was not as well-received by students using CDQs, and this may be because Python Tutor doesn't show File system structure.

7.5.3 Usefulness by Cognitive Processes and Overarching Programming Concepts

As my deployed CDQs were created through the guidance of our CDQ framework, I also investigated the usefulness of the questions along the cognitive processes and programming concept axes. The framework provides a way to examine student performance and understand where more attention needs to be given and what notional machine aspects students struggle with. As the CDQs were created with the guidance of the framework, the following analysis was completed only on the CDQ section's question set.

First, I categorized all my CDQs into cells in the framework. I have at least one question of each type across all deployed CDQs. However, only some cells have multiple choice and written response style questions (Figure 7.11). I calculated averages from the first

Week#-Q# **Multiple Choice/Select** Written Response	**Understand**	**Apply**	**Analyze**
Memory State	**1-1** **2-3** **5-1** 5-2 **5-3** 6-1 6-4 **7-3**	**1-2** 3-6	**1-3** **2-2** **3-7** **7-1**
Structure & Scope	3-5a **6-3** **7-2**	**3-2** **3-3** 3-5b 4-6b **6-2** **6-5**	2-4 4-6a
Path of Execution	4-3	**4-1** 4-5	**2-1** **3-1** **3-4** **4-2** **4-4** **6-6**

Week #	Topic #
1	Basics
2	Strings & I/O
3	Conditionals & While
4	For & Files
5	Iterable Objects
6	Functions
7	Random Numbers

Figure 7.11: The CDQs implemented in the CDQ section, categorized by CS concept and cognitive process from the framework. They are labelled using the week number, a hyphen, and the question number, matching the labelling in Appendix I. The MSMC questions are in bold font and WR are not.

attempts of the participants in the CDQ section across each row and column to explore the evaluative differences (Figure 7.12). This result suggests that the visualization helped with an immediate understanding of the memory state and helped reveal misconceptions about Structure & Scope, especially about the path of execution. No real difference is seen across the cognition levels. I also conducted this same analysis, but considering only the MSMC questions and calculated those averages by cell type; here, only some cells have an entry (Figure 7.13).

Scores decrease as questions progress across the cognitive levels to 'harder' levels. Also, memory state concepts are the most straightforward; structure and scope are a bit harder, and the path of execution was very challenging for students. Specifically, a point of interest is the difference in performance on Structure & Scope questions. This result may indicate that students need help communicating effectively in their own words about structure and scope.

This exploration of CDQ dimensions through student performance indicated that CDQs don't help with recall-style questions. Most topics benefit slightly from CDQs when omitting the WR questions. The expected difference in cognitive level shows up in the CDQ questions, with analysis showing lower scores, and the CDQs seem to reveal some misconceptions on

All Deployed CDQs

Number of Questions / Average % Correct	Understand	Apply	Analyze	Average by CS Concept
Memory State	8 75.1996%	2 53.1935%	4 72.0608%	70.7574%
Structure & Scope	3 61.6190%	6 60.4328%	2 21.2838%	53.4755%
Path of Execution	1 12.5%	2 39.5833%	6 57.0702%	48.3663%
Average by Cognitive Process	66.0829%	54.8442%	56.1005%	

Figure 7.12: Average question score on formative assessments for the CDQ section, including MSCS and WR questions.

ONLY MSMC CDQs

Number of Questions / Average % Correct	Understand	Apply	Analyze	Average by CS Concept
Memory State	5 82.7477%	1 86.4744%	4 82.3287%	83.0635%
Structure & Scope	2 78.4962%	4 79.2414%	0	79.0071%
Path of Execution	0	1 25.0%	6 57.0702%	49.6076%
Average by Cognitive Process	81.5343%	71.6477%	64.6795%	

Figure 7.13: Average question score on only the MSMC questions on formative assessments for the CDQ section.

Figure 7.14: These tables are organized by concept and cognitive process based on the CDQ framework. The row and column averages are calculated and shown in red.

the path of execution. Additional research is required to determine this revelation's impact on students on subsequent evaluations.

7.6 Interview Case Studies

Interviews were semi-structured and approximately 25 minutes in length. Given the tiny sample size, the planned thematic analyses could not be performed on these interviews. The three interviewed participants were in different stages of their educational journey. Each participant had taken at least one year of University level courses before enrolling in this course. The students interviewed were all "not typical students", meaning second degree, not their first year, had different experiences, etc. and could confidently comment on their study styles and habits from experience in other courses. All the participants interviewed had used the formative assessments more than required. While their usage details vary weekly, each participant found that at least one of the weekly assessments was valuable enough to attempt more than once. I transcribed the interview data and then performed a top-down deductive content analysis. I present three mini-case studies here to provide contextual clues

to help understand the quantitative findings.

7.6.1 Participant 1:

Participant 1 had a non-technical background, and this course was their first introduction to Computer Science. They stated that the class had too much to do each week, and often they had to pick and choose between different graded items to determine what was feasible to complete during class time. While the participant noted the benefits of the formative assessments, they found the written answer questions challenging to decipher. The participant stated that the written response questions seemed to have multiple correct responses, and second-guessing their understanding of the question took much work with little payoff. The participant went on to state that the other styles of question were enjoyable to complete, specifically noting they were helpful to check their understanding, more so than the in-class iClicker questions.

7.6.2 Participant 2:

This participant already had a penchant for Computer Science before taking this course, having previously received a formal education in Computer Science. The participant noted an appreciation for each week, clearly outlined expectations and a list of actionable items. Each week the participant would review that week's lecture material and use the formative assessments as a 'check your understanding' opportunity. The participant noted that they felt confident answering questions with or without a visual assessment given their background but stated that they wished they had the visual assessments when they were first learning these topics. Finally, the participant said they did use PythonTutor outside of these assessments. When seeing how the CDQ questions utilized Python Tutor for finding errors or understanding how a segment of code was running, the participant would regularly use it to check their work for their assignments or when practicing questions for a new topic.

7.6.3 Participant 3:

Participant 3, an end-of-degree student, found this course overwhelming and noted that this term was more than they signed up for. This participant found the formative assessments helpful and described how they would use Python Tutor outside of the assessments every week, specifying that they used it when working through assignments and classroom activities. This participant spoke about finding the formative assessments and visualizations most useful regarding data storage topics such as lists, sets, dictionaries, etc. Specifically, they stated the usefulness of the optional material when it came to the release of the correct answers as a way to gauge where they needed to focus more of their efforts.

My interview results support the hypothesis that students did not want to spend time on open-ended recall-style questions. I asked Participant 1 to walk me through the CDQ I presented to them during the interview (Figure 7.15). I asked the participant which of the variables presented was locally scoped, globally scoped, and why. They walked me through their answer to this question, and they started to explain it when they realized they had to double back on their response and tried to re-explain it. When the participant and I walked through the visual together, they finally understood how the explanation was connected to the visual. These results indicate ways CDQs can be used effectively in multi-modal learning environments but may need more hands-on reason and demonstrated uses.

All three interviewed participants stated that the weekly assessments were beneficial overall, explicitly mentioning the helpfulness of a mid-execution snapshot question. Unanimously, participants also said they found the course overwhelming in the material, which often resulted in submitting blank or 'garbage' responses to receive participation marks. Lastly, all 3 participants preferred the multiple choice/select style of questions and disliked the vagueness of short-answer/fill-in-the-blank responses. Two out of three participants stated they wished there were more practice questions each week.

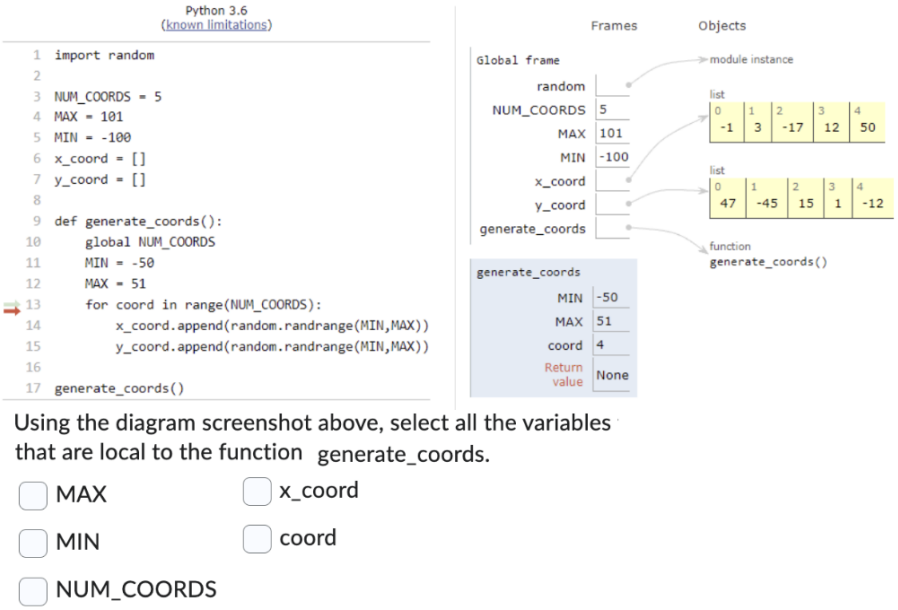


Figure 7.15: Screenshot of Python Tutor code after execution line 13, used during participant interviews. The screenshot focuses on the scope and the use of shadowing global variables with locally created ones. The visualization shows that more than one version of the variables MIN and MAX exist in the current program state.

7.7 Summary

In this section, I presented my findings following my seven-week implementation of formative assessments in the COMP 1012 first-year programming class. This section presents quantitative analyses as direct results from UMLearn course performance and event logs and a content analysis conducted on the transcripts from the semi-structured interviews. I report inferential statistics suggesting significant differences between the question types, multiple-choice/select (MSMC) and short answer/fill-in-the-blank (WR) used for the formative assessments and a difference in student performance based on computer science topic by section. My findings suggest CDQs were similar to the nCDQ practice problems in terms of performance and better in certain topics suggesting they were helpful in specific contexts and could be further enhanced. I have limited evidence to support my first and second research questions regarding the impact on course performance as a proxy for mental model accuracy and PT engagement. My findings also indicate that CDQs help with specific topics, as well

as help reveal possible misconceptions related to the path of execution concept.

Chapter 8

Discussion

When beginning my study, I had two research questions I wanted to investigate: How does using CDQs impact the accuracy of students' mental models? How do CDQs impact the level of student engagement with provided optional resources? I also explored how different dimensions of questions within the framework (concept, cognitive level and question style) affected student performance and experience. Overall, my study results indicate limited evidence to support my first and second research questions regarding mental model accuracy using summative performance as a proxy for mental model assessment and Python Tutor engagement. When considering the different types of questions and how they affected student performance and experience, my findings suggest that CDQs helped with specific topics and helped reveal possible misconceptions related to the path of execution concept. In this chapter, I discuss my interpretation of these results, and hypotheses about how different approaches may have led to stronger results.

8.1 RQ1: *How does the use of CDQs impact the accuracy of mental models, using course performance as a proxy?*

8.1.1 Summative Analysis Considerations

As I could not directly evaluate the participant's mental models, I used their performance on their first term test summative evaluation as a proxy. I used the participant's overall term test scores to compare the two sections and found no statistically meaningful difference between the scores. Considering why we see a lack of difference in the summative evaluation is essential. Since the course structure for the sections was the same, I had to consider many of the less obvious differences that occur in an educational environment. Two different instructors led the two sections with varying styles of facilitating their classroom activities. Due to this, it is possible there was different levels of attendance and engagement in the classroom. With large class sizes such as these, students also have significant variability. I did not realize how difficult it would be to mitigate the various confounds that may have prevented me from seeing a difference at the summative level alone.

I then considered the scores on the question I designed to evaluate their mental models, which specifically looked at their understanding of temporal flow in code execution. I found no difference in results and low scores across both sections. I believe that there were three main factors affecting this question's effectiveness. First, the evaluation structure reflected that of the nCDQ section. Since both sections were administered the same evaluations, I planned the questions to not include visualizations because it would be unfair to students in the nCDQ section to see visualizations on summative evaluations when they were not accustomed to them. The non-visual medium could have affected the CDQ section scores because structuring these questions with no visuals may have caused the summative questions to be more difficult for them to understand. The CDQ section was used to seeing visuals

alongside their questions so this may have been biased against them.

The second factor that I believe impact my results is the choice of topic to evaluate the students on. During my analysis, I learned that the flow of execution was the lowest-scoring over-arching programming concept for students on the formative assessments, suggesting it was the concept they needed help with the most. When I created this question, I hadn't considered the correctness rates by concept but focused on the term test structure and the topics that had to be covered during the evaluation. These poor scores may reflect this concept being more complex. Additional practice on this topic or more questions to analyze could have given me a clearer picture of the effectiveness of the formative assessments.

Finally, I chose to administer a written response question on the term test and this may have been ill-fated. I decided giving them a written response style question would allow students to show me what they did know and limit the possibility of randomly selecting the correct answer, like an MSMC question. Again, this question was deployed before my analysis, and I did not realize the impact the type of question would have on the students. In the future, if I wish to use summative evaluations as a proxy, I will need to include more questions using different approaches. All of this meant I was not able to glean a broad enough picture of the students' mental models given my implementation. Had I done some form of analysis on the performance of the questions by framework cells leading up to the evaluation, I may have added more questions, basing them off the gaps I saw when categorizing questions in the framework.

Design Consideration 1:

Categorize learning objects into the framework cells

Categorizing the learning objectives into the different CDQ Framework cells will both make question creation easier and ensure instructors are targeting the programming concept and level of understanding that instructors are expecting to target.

8.1.2 Formative Analysis Considerations

There was no statistical difference in formative evaluation performance when considering all question types (MSMC and WR). Written response style questions are more challenging for students than MSMC style questions [81], and I believe students were not able or willing to spend additional time working on their answers for the WR questions. I found many students submitted blanks or garbage responses for the WR question types. If these assessments had a higher final weight or were used as in-class activities with written response questions instead of individual self-assessment quizzes, it is possible that students would engage with them in a more meaningful way. I expect another challenge with these assessments is the lack of readily available feedback. Students who completed an assessment immediately knew how many questions they answered incorrectly but not which questions until the solution was published with their feedback. I chose this feedback system thinking if students knew what they got wrong they would skip over the questions in their reattempts. I thought students would be driven to get them all correct in this way but had not considered students who truly may not know which questions they answered correctly. This lack of immediate feedback may have contributed to students' low engagement with these questions. Students who did 'well-enough' on the assessment may have just moved on, not driven to attempt to determine which ones they answered incorrectly.

I also found there was a distinct inability to provide meaningful question-level input based on their answers meant the usefulness of the written response questions as a formative tool was limited. I could only write so many regular expressions for capturing correct answers. The lack of personalized feedback based on a student's understanding of a question makes the feedback given in WR questions weak whereas the limited response choices in MSMC is easier to predict feedback for.

Once I removed the WR questions and saw a difference in scores, I realized that a future iteration of this study would need equal numbers of MSMC and WR questions for each cell in the framework. The scoring was so different once I had adjusted the data set, which may help

explain student engagement. If an individual assessment had more MSMC questions, they might have been more inclined to try it repeatedly; if it had more WR questions, students might have glossed over the assessment, not seeing it as worth their time.

Data suggested the CDQs were better than the nCDQ questions when it came to the formative assessment performance. I saw a heavy preference for recognition-based questions (MSMC) given the correctness rates in the CDQ section. This makes sense since I asked them to look at a picture to answer questions. Students did well with these questions, and having access to a screenshot seemed to helped overall. Both sections struggled with open-ended style questions but even more so in the CDQ section. I think this may have been because students were trying to find an easy recognition-based solution in an open-ended question. Having a visualization meant they thought they could find the answer more explicitly in the image rather than using the visual as a tool to understand the problem at all. Tweaking how CDQs are written could help overcome this obstacle. The empirical data suggests little summative difference in course performance for the two sections.

Design Consideration 2:

Confine the answer space to ensure useful feedback or consider AI-based feedback systems

MSMC questions leave a finite amount of room for interpretation and varied results allowing for a simpler feedback process. Open-ended question types in formative assessments, are most useful when they give student personalized feedback, allowing them to understand their mistakes. This can be difficult with many different levels of understanding and question interpretations. Considering an AI-based feedback system or extremely pointed questions may help mitigate this, keeping these questions impactful for students.

8.2 RQ2: *How do CDQs impact the level of optional engagement with PythonTutor?*

8.2.1 Engaging with Formative Assessments

My original hope was to track how frequently students accessed Python Tutor outside the formative assessments and look at how the two classes interacted with the tool outside the context of the practice questions. While this type of data tracking is possible using SCORM objects or other means, due to software limitations in UMLearn, extracting empirical data surrounding this research question was not possible.

Next, to expand on the idea of engagement, I wanted to investigate engagement with the formative assessments as a learning tool through immediately repeated attempts. If students repeatedly tried the assessments, it could indicate they found value in them and would want to continue to engage with them. The actual number of immediate repeated attempts varied by week but was small. I do not think students felt there was enough reason for them to continue reattempting the assessments. If a student did 'well enough' on their first attempt, they did not seem to think there was a gain for them to try again. I considered measuring student earnestness [82, 83] using time between attempts as a metric and how it would affect a student engaging with optional materials too late into my study. Due to a lack of immediately repeated attempts and lack of interaction data, processing the data to calculate earnestness was not worthwhile nor possible for some metrics.

If I were to redo this experiment, I would consider using a tool that allows me to set a threshold of correctness on assessments before it is considered "complete". I want to keep these assessments low-stakes, so students can redo them as much as they want. However, I would set a minimum correctness rate to receive the marks instead of relying solely on submitting them for participation. This would force students struggling with a topic or assessment to repeatedly attempt it with minimal stress since they can try it as often as they would like. Using a grading system like this would allow me to see a progression in

understanding better than the UMLearn tool allows. Then, by tracking data such as time spent between questions or attempts, I could gather additional information on the level of engagement with these tools.

I also considered how much students used the formative assessments as a study tool when approaching the final exam in in April. Students in the CDQ section returned to the formative assessments as a study tool in April more than those in the nCDQ section. However, the number of attempts was small, meaning the results were only borderline significant. Subjectively, the interviewed participants described their enjoyment of these practice problems and interactions with the Python Tutor. This feedback could be a sign of higher engagement with the CDQ questions. With some modifications to the questions and additional research, there is room for CDQs to enhance student understanding and be a practical formative resource. These findings concur with similar research results using visualizations to understand notional machines [1, 21, 39]. Furthermore, I build on these works by exploring additional ways to support the creation of effective diagrammatic representations of coding concepts and data structures. While the overall course performance was not meaningfully affected, visual assessments could enhance other aspects of a student's educational journey, such as making fewer mistakes in coding activities or affecting their collaborative work dynamics.

To further support my hypothesis that the formative assessments were useful for students, my mini case studies suggest that, in theory, the problem sets I implemented were valuable but sub-optimal. Participant responses related to the heavy coursework load and low interest in responding to written response style questions concurred with the course performance findings. Participants enjoyed interacting with Python Tutor and the practice problems. Still, they viewed them as a quick check-in of their understanding rather than wanting to devote much time to them. Given what is already known about the challenges associated with post-secondary student management [72], it follows that these students, with their busy schedules, are unwilling and unable to devote significant amounts of time to practice problems. Creating short and practical formative assessments is vital in engaging students

regularly. Alternatively, it may be worth considering using CDQs as part of graded homework assignments or during in-class active learning activities when students have the time and energy to devote to them [84].

One participant noted that the workload was not worth the percentage tied to it (5% for submitting a minimum of 8 practice problem sets throughout the term). So, they often left questions blank or selected random answers to ensure they were submitted on time. All three interview participants shared this sentiment. The average weekly scores for the problem sets were higher when I omitted the WR questions, leading me to believe hurried answers were more typical on the written response style questions over the MSMC questions. This suggests that a series of rapid-fire multiple-choice/select-style questions that are not time-consuming may be more effective from a student standpoint. Considering a more flexible deadline schedule so students may work on questions when their schedule allows may lead to more engaged and earnest responses. Participants also found the number of items they must complete each week overwhelming, a common complaint with flipped class structures [85]. This may have inadvertently led students to cut corners on these questions. Discussions with two of the three participants included suggestions on what other activities/grade items could be cut in future course iterations to make these practice problems more meaningful and worthwhile, which may increase engagement.

Design Consideration 3:

Ensure the level of work matches the grade students receive for completing it

It is possible for formative assessments to be both low-stakes/low-stress and worth the work students put in. Adopting an unlimited submission with a correctness threshold model may help students interact repeatedly with this without causing them stress to answer correctly in a limited number of attempts. Alternatively, quick and short assessments that require little effort can be made available for fewer marks, helping to ensure students feel they will get out of the assessments what they put into them.

8.2.2 Formative Assessments as a Final Exam Study Tool

I also considered the engagement levels of students with the formative assessments as a study aid approaching the final examination. I found that each week in the CDQ section had additional attempts in April 2023, while only some weeks in the nCDQ section had further attempts. Although I had a small N for this data set compared to the number of participants in the classes, I saw a significant use trend. While there is not much to say regarding these attempts, this may indicate a slight trend in the usefulness of CDQs. Creating additional question sets to be used as final exam tools may help students solidify their understanding of these topics, rather than having them use the same questions they had already answered. No participants specifically commented on their use of the formative assessments as a study aid for the final exam however all three participants did comment on their frequent use of the tool throughout the term.

8.2.3 Final Considerations for Python Tutor Engagement

I could not track students who accessed Python Tutor outside of UMLearn. This would be exciting data; however, short of polling each student on their use of Python Tutor, this data was not feasible to collect. That being said, each interview participant explicitly stated finding Python Tutor engaging and using it as part of their studies or checking their answers in assignments and activities. Even without targeted access logs, the interview participant's opinions suggest that Python Tutor was a tool they found valuable when developing a deeper understanding of their code and the concepts they were learning. Interview participants noted that they were unaware of Python Tutor before its introduction in this class and found it immensely helpful. It seems likely that the student's having repeated exposure of Python Tutor through the CDQs helped them understand the value of the tool. With additional investigation, it would be possible to dissect student engagement and earnestness [82] with these questions and measure their effectiveness as predictors for long-term success. Given the large gap in post-secondary research surrounding formative assessments [68], there is

more to explore here regarding effectively engaging students with formative resources.

8.3 Exploration of CDQ Dimensions and Student Performance and Experience

When investigating this, I considered these results alongside the foundational difference between recall and recognition skills [86]. Often the lowest levels of Bloom's taxonomy are associated with the basic skills of recalling and recognizing facts [87], and these skills build the foundation for further success. Students often prefer multiple-choice questions as these questions engage the recognition part of the brain. Selecting a cued, recognizable option over open-ended questions, where the recall skill is necessary to succeed, is often easier for students [81]. I also found that students were frustrated with answering the open-ended style (WR) questions because they were guessing what string of words would be the correct answer. Often, I found that students with vastly incorrect responses did not read questions correctly and, therefore, responded inappropriately. Educators in CS often attempt to use regular expressions to give students effective feedback on open-ended questions. Automated feedback based on regular expressions can only take a student so far. It's impossible to think of every possible way a student may attempt to answer a question, account for spelling and grammatical errors, or specific comprehension issues that lead to incorrect answers. Given the large class sizes and the array of solutions a student may give, it is worth exploring alternative ways to ask WR questions and provide feedback so that students are more encouraged when interacting with them on a functional level.

8.3.1 Question Type Considerations: MSMC versus WR Questions

The choice to focus on the MSMC questions in my analysis was based on the poor performance and variety of answers given in the written responses and the prior research completed by Roberts suggesting the appropriate use of MSMC questions can be powerful [71]. Given my deployment of CDQs, I decided I could learn the most from my implementation of MSMC CDQs over the WR attempts, which did not work out as planned. While most attempts seemed genuine, meaning they did not just submit random 'keyboard mash' garbage, many students seemed to lack reading comprehension skills. When reviewing the submissions to questions, many students seemed to have genuine answers but they did not address the question they were being asked. This contributed to misunderstanding questions and answering in entirely unexpected ways, and often, MSMC leaves less room for this type of misunderstanding. A secondary factor as to why there is a difference in correctness based on question type is the student's level of engagement. Based on my interview discussions, students wanted to avoid interacting with more extended answer questions. There was a hesitancy to even spend time on these because of frequent misunderstandings about what a question was asking. Even if an instructor writes good questions, it is a waste of time if students do not feel confident when interacting with them.

8.3.2 Computer Science Concept Considerations

Regarding my research into visualizations, investigating CDQs vs traditional questions (nCDQs) on a topic and question style basis, my findings indicate a statistically significant difference between the sections at different levels. Focusing on computer science concepts, I saw the CDQ section significantly outperform the nCDQ section in five of the seven weeks. My findings suggest that some topics covered in the class were better understood by students using a visual medium, such as Strings & I\O, Conditionals & While Loops, Iterable Objects, and

Random Numbers.

When considering the over-arching CS concepts that define the framework, we see Memory State being a less abstract concept at the introductory level and Flow of Execution being the most abstract at this level. Having a visual component should help students understand the flow of execution since it is so abstract, but this is not what happened. I believed that a tangible visualization of loops, objects, and random numbers could only help a student comprehend such abstract concepts. For novice programmers, there is a familiarity with concepts such as numbers, math, and possibly even files meaning visualizations may not be as necessary for a strong understanding of these topics. I think it is possible that Python Tutor shows concepts like Memory state and Scope particularly well but does not visualize the Path of Execution as well, which is only depicted using the green and red arrows in the simulation for the current and next line. I think creating CDQs for other visual mediums that represent the flow of execution better may be able to help with this. Specifically, showing side-by-side states of a program would help students understand how a computer runs code and execution of flow versus a single snapshot mid-execution.

Design Consideration 4:

Design CDQ formative assessments holistically to ensure expected distribution of question types

Once an assessment is written, summative or formative, it is important to go back and ensure instructors are focusing on your expected learning outcomes. Categorizing each question into the framework ensures each question targets a specific area of understanding as well as allows you to verify an even distribution of assessment across the programming concepts and depth of understanding.

8.3.3 An Educator's Perspective

From an educator's standpoint, question creation is a difficult skill to acquire and master, given the different techniques and methods used to ensure learning objectives are what students take away from the course material. I found visualization-based question creation using our framework was a straightforward process. For example, when considering the simplest of programming concepts, such as variables or data types, in the past, I have found it challenging to create meaningful practice problems, given the simplicity of these topics. Using our framework as a guide, I succeeded in creating questions that accurately reflected the course's learning objectives. As seen in the weekly average table, these early weeks, which still included students who would later withdraw from the system, had the highest average score in both sections (Table 7.4). Having a framework to guide question creation and the constraint of utilizing the visual allowed me to create practical, creative, and targeted formative assessments.

Furthermore, I find that as soon as I, as an educator, learn something, I forget the road I took to reach that moment of comprehension, leading to a disconnect between myself and my students. Taking time to create and use our framework as a formulaic resource to create questions that resonate with novice programmers made my job easier. While the data I collected came from students and my thesis focused on something other than the educator aspect of this process, as a technical educator, I couldn't help but become aware of the benefits of the CDQ framework when creating the questions for my study.

8.4 Limitations & Obstacles

8.4.1 The Deployment Tool

Understanding the limitations of the resources used is pertinent in designing experiments effectively. The framework's malleability to be used regardless of LMS is a benefit of my

work. Hence, a significant limitation of successfully implementing the framework falls to the medium in which CDQs are offered. While many different question types could be implemented using the framework, the UMLearn quiz tool allowed for implementing only a limited set of question styles. Our focus on multiple choice/select and short answer/fill-in-the-blank stemmed from the LMS and the limitations of it. While UMLearn did have alternative question styles, such as ordering questions and matching questions, these types were not implemented effectively, and previous attempts to use these questions seemed to cause more confusion than they cleared up for students. Since different question styles may benefit other students, it is hard to say on a small scale why a question may have worked differently than intended. Approaching the creation of formative assessments from a multi-modal standpoint is considered the most effective way to create accessible content [88]. This requires creating multiple questions that probe the same concept but use different styles and tools. That being said, visual-based question formats may lend themselves better to specific questions. The hope in keeping these types of assessments low-stakes is that even if students are not fans of a particular style of question, they may still attempt the assessment if they say that they can only benefit from trying.

UMLearn's self-assessment tool would have been, psychologically, more effective than the quiz tool for low-stakes assessments; however, it needs to provide more information about how a student interacts with it. There is no submission option, so feedback is shown on a selection-by-selection basis for a question, and the details of their attempts are not tracked or logged. From a research data collection standpoint, self-assessments only record whether the content item was opened. As I required more data than this, I explored ways to use the quiz tool without making the assessments feel like quizzes. Self-assessments and Quizzes are built on the same foundation in UMLearn, so the question types I had access to were the same. The additional capabilities from making the weekly question set into quizzes meant I could see how many attempts were made, what the students answered on each attempt, how long they spent in the attempt before submitting, and more, which was a vast improvement

over the self-assessment option. The main limitation of quizzes is that each question must have a mark value associated with it since it is a quiz. I believe that for some students, seeing a score of any kind may be triggering and put superfluous stress on the student, impeding their abilities to interact with the assessment in the intended way.

8.4.2 Education Research

A common issue in educational research is consistency. The content of each course may be identical, but the set of students and instructors in each section will vary. It would be impossible to have two classes with the same distribution of students in terms of characteristics such as motivation, background, and experience. To mitigate this, we must rely on a larger random sample over a small variable producing reasonably robust results. There's also the issue of consent. If I was able to look at data of all students, not just consenting students, I might have seen more of a difference, though this could go in either direction.

Another variable that could not be controlled is the instructor difference, as each section had a different educator. The course content is streamlined for consistent experiences, including the in-class activities and assessments, to mitigate as much bias as possible from two different people teaching the course. Furthermore, students in both sections had to watch the same video lectures in preparation for each week, prepared and recorded by a previous course instructor. That being said, much of what we know about classroom learning is the importance of the educator and the impact an engaging and earnest instructor can have on the learning experience. While both instructors were well suited for their position, different levels of experience, classroom management and presence likely impacted students' experiences in a way that could not be controlled for with respect to my data collection.

The final poignant limitation I saw was the structure of the summative evaluations reflecting that of the formative assessments used in the nCDQ section. Since both sections were administered the same summative evaluations, I planned for these evaluations to not include visualizations because it would be unfair to students in the nCDQ section to see

visualizations on the assessments when they were not accustomed to them. Following the first term test, I discussed the implications of this with Dr. Celine. This discussion made me realize that structuring these questions with no visuals may have caused the summative questions to be more difficult for the CDQ section to understand, as they were used to seeing visuals alongside their questions. I realized there was no easy way to address this bias as the two sections took identical summative assessments other than to include both types of questions on summative assessments. The drawback of including both styles on the assessments being it would increase the length of these examinations.

8.5 Summary

In this section, I presented my interpretations of the study results. My findings uncovered meaningful insights into the use of visualizations for teaching novice programming concepts. Based on my interpretations of the results, I have suggested some preliminary design considerations for instructors using CDQs and the framework to help students in introductory CS courses build better mental models. However, these considerations should be thought of as exploratory and more evidence is needed to support them. To summarize, I saw themes emerge which suggest the use of multiple-choice/select style questions were preferred by students and more effective alongside the visualizations. Findings from this study also provide recommendations for future research into mental model construction and formative assessment engagement.

Chapter 9

Conclusion

In this thesis, I presented an in-situ comparative study of CDQs as formative assessment questions and their effectiveness in engaging students with optional resources and evaluating their mental model development. Results from my research indicated that CDQs might enhance immediate understanding of some topics when used in formative assessment activities and provide initial evidence of positive trends in optional engagement. These results suggest CDQs helped to reveal misconceptions in understanding the topics related to the flow of execution programming concept. Overall, my findings suggest that CDQs have a place in classrooms and that their implementation could benefit students learning abstract concepts in introductory programming.

9.1 Contributions

This thesis makes three contributions. The first contribution is a framework for educators to use when creating targeted formative visual questions. The CDQ framework allows for the flexibility of different programming languages and visualization mediums while still supporting students in comprehending core computer science concepts. The CDQ framework provides guidance to instructors and may help with question creation.

I also contribute the results from an in-situ comparative study using questions developed

using the CDQ framework in a first-year programming class. My results show a small positive impact for some introductory programming topics for students using visual formative assessments created from our framework. Students using the CDQ visual assessments did as well, or better, in most of the weekly topics presented in class. My results suggest that using our framework to aid in targeting specific concepts and cognitive processes may continue to support students through additional topics. Moreover, my study shows we can implement formative assessments and test their efficacy in large classroom settings. The use of formative assessments in large classes is a complex process which may be one of the reasons why current research in the area is limited, but it is doable.

Finally, I contribute some design considerations that can be used to support educators when using the CDQ framework to create questions. For example, some suggestions include focusing the question type to multiple select/choice to reduce the number of garbage and blank responses and increasing the question pool for additional variety, increasing the re-usability of these assessments. These changes may affect the levels of student earnestness when engaging with these optional materials. Student engagement levels and the small cohort of interviewees suggest that these considerations could positively impact students' learning experiences.

9.2 Future Work

Working on this research has been incredibly rewarding and left me with a few open questions that build on CDQs and would be interesting to explore. First, I want to look at the earnestness of students when completing these assessments. Earnestness is often evaluated along a few different axes; the principal axes being how quickly a student completes an assessment [83], and the time between taking scored assessment attempts [82]. Determining how earnest a student is in conducting CDQ assessments or how their earnestness changes over time would help educators create more meaningful assessments to support students

better. Even though my summative assessment data did not show a significant difference in final course grades between sections, it would be worthwhile to explore the number and types of programming mistakes students make and how they are affected by the use of CDQs.

Second, given the frustrations students remarked on when responding to open-ended style questions, I want to investigate conversation-based assessments and their implications. With new natural language processing tools emerging regularly [89], finding ways to capitalize on them and assess their usefulness in providing automated feedback on open-ended style questions would be an exciting area of research to explore [90].

Finally, my experience suggests that the CDQ framework could benefit the instructor question creation process. Thus, it is worth studying the use of the CDQ framework by other instructors creating questions related to their courses to gauge their impression of the usefulness and effectiveness of the framework. Testing the framework in classes taught in different ways (not flipped) and with other programming languages and visualizations would be worthwhile. Question creation is a skill that takes time to develop, and furthering the research on aiding instructors and their professional development is essential. It would be interesting to find ways to make question creation easier on educators while maintaining the objective that questions target specific levels of skill mastery for students.

Appendices

A Approved REB Document



Research Ethics & Compliance
Human Ethics Office-Fort Garry
208-194 Dafoe Road
Winnipeg, MB R3T 2N2
T: 204 474 7122
humanethics@umanitoba.ca
umanitoba.ca/research

Amendment
HE2022-0362
Celine Latulipe

Table of Contents

1. Amendment Summary....1
2. Summary - Summary....2
3. Summary - Purpose of the Research....3
4. Summary - Research....5
5. Summary - General Questions....6
6. Research Personnel....7
7. Participants - Participants....9
8. Participants - Compensation....10
9. Participants - Recruitment....11
10. Consent - Informed Consent Process....12
11. Data - Confidentiality....13
12. Data - Data....14
13. Data - Data Transfer....18
14. Risks/Benefits - Benefits....19
15. Risks/Benefits - Risks....20
16. Dissemination/Withdrawing - Feedback....21
17. Dissemination/Withdrawing - Withdrawing....22
18. Attachments....23

HE2022-0362	Amendment	Celine Latulipe

1. Amendment Summary

Please provide a brief summary of the requested changes	Another graduate student from the Human-Computer interaction lab under Dr. Celine Latulipe will have access to the students' UMLearn data as they will help with the analysis of that data.
Please provide a justification for these changes	A preliminary analysis of data will occur after the collection of consent forms but prior to complete data collection. This data will be collected, aggregated and analyzed by Lauren with the help of another Masters student, Min Kyu Jung. Min has also completed his TCPS 2 and will follow all appropriate privacy and confidentiality policies as described in this protocol.
Will there be changes to the number of participants?	No
Will there be any changes in recruitment?	No
Will there be changes in recruitment material?	No
Will there be any changes to the consent form?	No
Will participants need to be re-consented?	No

2. Summary - Summary

Protocol Title	Studying CDQ Effectiveness in First-Year Computing Courses
Principal Investigator	Celine Latulipe
Primary Department	Computer Science
Application Initiated By	Celine Latulipe

3. Summary - Purpose of the Research

Select the appropriate REB for review:	REB 2
Is this a study that has already been approved by another Canadian institution?	No
Does this study only involve the use of secondary data?	No
Provide a brief statement about the project written in lay language. Do not cut and paste directly from the study proposal.	During the Winter 2023 semester, the CS Department will be testing the effectiveness of a new type of formative assessment called Code Diagram Queries (CDQs). These questions ask students to look at diagrams related to computer code and answer questions that probe their understanding of how the code executes. CDQs will be tested by deploying them in one section of the COMP 1012 class while deploying traditional multiple choice and fill in the blank questions that probe the same concepts in the other section of the COMP 1012 class. Data will be collected from both sections of the course to understand the effectiveness of using CDQs in future semesters. This REB adds a research component and involves asking students' permission to analyze their data and to participate in interviews that probe students' experiences with the CDQs. By asking for student consent to analyze their data (interactions with CDQs in an interactive textbook, performance on related test questions, and survey responses), we will be able to share the results of this pedagogical investigation with the broader CS education community. Note that regardless of REB approval for the research study, the data analysis of student performance in the two sections will go ahead because the CS department is interested in understanding the effectiveness of CDQs and whether to use them in the future.
Describe the research question(s) and objectives for this research study.	The study is intended to uncover how effective CDQs are at helping students develop a correct mental model of how code executes.
Describe the research methods.	This study involves two types of data collection and analysis: 1. Analysis of student work, surveys and grades collected from UMLearn. For the purposes of this study analysis, we will only include data from students who consent. 2. Semi-structured interviews with up to 8 students.
Describe briefly in a step-by-step manner what the research team will be doing with participants, after they have been recruited and consented.	Midway through the Winter 2023 semester, coinvestigator Lauren will visit both sections of the COMP 1012 class and explain the research project and will present students with consent forms asking to analyze their UMLearn data which will include classroom reflection surveys, course performance data, and formative assessments. Only data from students who consent will then be used in the research. The course instructors will not know which students consent to their data being used for research purposes or which students consent to be contacted for interviews at any

	point. Lauren Himbeault will collect the consent forms and hold sole responsibility over them. Analysis of data will commence once consent forms have been collected and students will be recruited for the semi-structured interviews. Students will be grouped by their activity level with respect to the formative questions so that we can recruit interview participants who are both active CDQ users and users who did not complete many CDQ questions. Interviews will be conducted on Zoom. Students who participate in interviews will be compensated with a $15 gift card to Tim Hortons. Interviews will be manually transcribed by Lauren Himbeault and analyzed jointly by Lauren and Celine.
Where will the study take place?	The consent form distribution and collection takes place during class time and the interviews will take place on zoom
Approximately how long will this study take for participants?	For students to consent to data collection, the process of hearing about the research, reading the consent forms, and asking any questions should take approximately 30 minutes. For students who participate in interviews, the interviews should take no longer than 30 minutes. Edited to Update: the research does not include any surveys. If students are filling surveys out in their class, that data is included in the data collection.

4. Summary - Research

Type of Research (select all that apply):	Master's Thesis
Type of Study (select all that apply):	Focus Group/Interview, Secondary Use of Data

5. Summary - General Questions

Does the study involve participants who are not legally or practically able to give their valid consent to participate?	No
Does the study involve participants who are under the age of 18?	No
Are participants from a population that may be marginalized or vulnerable in the context of research?	Yes
Does this research include the use of personal health information?	No
Does this study use deception (i.e., will participants be intentionally misled about the purpose of the study, their own performance, or other features of the study)?	No
Will the majority of participants identify as First Nations, Inuit, and/or Metis?	No
Will the analysis of the research results use First Nations, Inuit, and/or Metis identity as a variable?	No
Will the interpretation of research results refer to First Nations, Inuit, and/or Metis people, language, history or culture?	No
Will participants be given the choice to waive their anonymity?	No
Does this study require approval from another organization?	No
Will this study include involvement or recruitment from a specific organization?	No
Will this study occur in locations outside of the University of Manitoba?	No

HE2022-0362 **Amendment** **Celine Latulipe**

6. Research Personnel

Name/Department	Role	CORE	PHIA	Access	Contact	Contact Level	Oath Of Confidentiality
Celine Latulipe / Computer Science	PI			Signature Authority		Primary	
Lauren Himbeault / Computer Science	Co-Investigator			Edit Access	Yes	Primary	
Min Kyu Jung / Computer Science	Research Assistant			No Access		Primary	oath_min.pdf

Describe each team member's role in the study (e.g. staff, research assistant, student, statistician, supervisor etc.)	This project is part of Lauren Himbeault's MSc. Lauren will conduct the data analysis under the supervision of Celine. Lauren is collecting data from the COMP 1012 course in Winter 2023 as well as interviewing students from the course. Students in the class are being presented with a variety of new question types and Lauren will be investigating their effectiveness in a first-year computing class. Partway through the term, Lauren will collect student performance and conduct semi-structured interviews with a small number of students, and transcribe those interviews. Celine will oversee all data analysis. Min Kyu will have access to the UMLearn course data. Min Kyu will be helping aggregate and analyze the data.
Are any of the research team members affiliated with institutions other than the University of Manitoba?	No
What type of data (e.g. directly identifiable, anonymized) will each individual have access to?	Lauren and Celine will have access to course data, interview transcriptions, and textbook data. Min Kyu will only have access to the UMLearn course data. This will be directly identifiable when collected but stripped of all identifying features prior to analysis. Lauren will be conducting semi-structured interviews and transcribing them. Min Kyu will have access to the same directly identifiable data as Lauren but stripped of all identifying features prior to analysis.
How will the PI ensure that all research team members are aware of their responsibilities regarding participants' privacy and confidentiality? Research coordinators and assistants must complete an Oath of Confidentiality.	All investigators are made aware of the criticality of maintaining the privacy and confidentiality of the gathered information. All identifiable data (audio files) will be deleted within 2 weeks of transcription and all identifiable course performance information will be deleted within 2 weeks of data collection starting (after the final grade appeal deadline in May of 2023). All members have their TCPS2 certifications
Do any of the research team members have a study related conflict of interest that could compromise or reasonably be perceived to compromise the objective conduct of the research or the integrity of the data generated by the study? Conflicts of interest may include business, commercial or financial interests, dual roles (e.g. teacher and PI), as well as personal matters and career interests	Yes
Please describe. Please explain how you will ensure participants do not feel pressure or obligation to participate or perceive that they may be penalized for choosing not to participate.	While Lauren is the course director for COMP 1012 in Winter 2023, she has no direct contact with students. The role of a course director is to oversee the instructors and ensure consistency between sections so there is no reason a student would feel pressured to participate. Instructors

of the course will be asked to leave the room when the study is explained and consent forms are distributed and collected. Students will be informed that their instructors will have no access to who consented to the release of their data nor who will be interviewed. This process is entirely separate from the course and the instructors will be unaware of who is taking part, so their decision to have their data analyzed as part of this research will not impact their grades or relationship with their instructors. Students will be told that Lauren will be conducting the interviews and data analysis however as a course director, Lauren has no impact on their grades whatsoever. Lauren has no say over any graded materials and will not be taking part in any marking or grade assignment whatsoever. Students will be informed of the role of a course director so they feel confident in understanding that agreeing or not agreeing to participate has no bearing on the course performance. All students will be given a consent form with a checkbox either allowing or prohibiting the use of their data so that we have explicit consent from each student. Students who are not present in class on the day Lauren attends to explain the study will be contacted by email with an embedded consent form. Lauren will compare the collected consent forms against the class lists and send emails to students who were not in class and did not complete consent forms. The scripts, consent forms, and emails can be found in Appendices A through E. Min Kyu has no connection to the COMP 1012 course and its students.

7. Participants - Participants

How many participants do you expect to recruit?	Each class has a capacity of 150 students, totaling a potential 300 students. We anticipate most in the class would consent to have their data analyzed, so at most, we would have 300 participants for the main analysis. For interviews, there will be a maximum of 8 students interviewed.
What is the inclusion criteria to participate in the study?	Students who enroll in COMP 1012 may be included to consent to the release of their course data. Students who wish to participate as an interviewee must also consent to have their audio recorded.
What criteria would someone become ineligible to participate?	If they drop the class anytime before the consent forms are distributed and collected (March 2023)
Will the participants in your study be UNAWARE that they are participants?	No
Will information about the participants be obtained from sources other than the participants?	Yes
Please describe	As course director, while I have access to student information such as grades, I am not responsible for any evaluations or assignments of grades in the course. However, I will have access to student information (email addresses) in UMLearn. Once I have collected consent forms from the students present in class, any students who were not present the day the consent forms have been distributed will be contacted via email. As course director, I already have access to the list of emails of students and no additional person is required to distribute or have knowledge of student information.
Will participants receive any compensation for participating (e.g. honorarium, course credit, food, parking)?	Yes

8. Participants - Compensation

What is the compensation for participating?	Students who are selected for interviews will receive a 15$ gift card to Tim Hortons for their time. Participants who consent to the release of their data will not receive compensation.
Please provide justification for these compensation arrangements	$15 is appropriate monetary compensation for a 30-minute interview. Students consenting to the release of their data are not compensated as we are not actually asking for any of their personal time in order to obtain information from them. All conversations and consent processes will take place during class time.
When will participants receive their compensation?	As all interview communication and processes occur online, consent forms are collected prior to the interview date. Participants will receive their compensation at the beginning of the interview. Gift cards will be sent to their preferred email. Lauren will be the one to give out the gift cards as she is the one conducting the interviews.
Participants must be able to keep their compensation if they withdraw from the study. Please describe this process	Participants will receive the gift card at the beginning of the interview, so the gift card cannot be revoked, even if the participant decides to stop participating before the interview is completed.

9. Participants - Recruitment

Provide a step-by-step description of how you will identify and recruit participants. Describe how prospective participants will be identified, who will contact prospective participants and by what means this will be done	Lauren will distribute the consent forms to both classes in class at the beginning of March in the Winter 2023 semester (Appendix A). During this, Lauren will explain the purpose of collecting their data, Appendix B. They will distribute and collect consent forms from all students present (students check yes or no on the consent forms). The instructors for the sections will not be in the room while this happens. Lauren will have access to the list of students in the two class sections and will send an email to students who were not in class to complete the consent forms, Appendix C. In this way, all students enrolled in the class will be given a chance to consent to participate, even if they aren't present in class the day the consent forms are distributed. This email will contain the consent form and the student will be asked to simply respond to the email. Once consent collections have occurred, Lauren will contact students who consented to a follow-up interview. Lauren will email those students to see if they still want to participate (Appendix D). Those who are interested will respond via email and an interview date will be scheduled for the near future. A single reminder email will be sent to those who consent to be contacted in the event that we do not get an initial response within 2 weeks.

Attach copies of all material that will be given/read to participants and/or third parties:

Type	Name	Document
recemail	Appendix D: Interview Email Script	D-Interview-Email.pdf
recscript	Appendix B: MARKED UP	AppendixB-MARKEDUP-ConsentDistribution-Script.docx
recscript	Appendix B: CLEAN	AppendixB-CLEAN-ConsentDistribution-Script.pdf
recemail	Appendix C: MARKED UP	AppendixC-MARKEDUP-ConsentEmail.docx
recemail	Appendix C: CLEAN	AppendixC-CLEAN-ConsentEmail.pdf

10. Consent - Informed Consent Process

Describe the consent process. Where and how will consent be obtained?	Lauren will distribute the consent forms that ask for the release of their data and if a student consents to be contacted for an interview about the course material. Lauren will inform students of the outline of the study, what their data will be used for, and answer any questions they may have. Those contacted via email by Lauren for an interview will receive the interview consent form by email to review. Students will be asked to return their consent forms via email. They may be printed and filled out and then scanned/photographed or they may fill them in directly on their computers and return an electronically filled-in version. Lauren will read the script found in Appendix B and distribute the classroom consent forms in Appendix A. Lauren will send an email to absent students (Appendix C) and an email that Lauren will send to those willing to be contacted about a follow-up interview in Appendix D along with the consent form for the interview in Appendix E.

Attach consent form(s):

Type	Name	Document
consentdoc	OLD IN CLASS - deleted old file not realizing I would lose the notes made by the REB	NEW-AppendixA-CDQ-Consent.pdf
consentdoc	OLD EMAIL - deleted old file not realizing I would lose the notes made by the REB	NEW-AppendixE-Interview-Consent-Form.pdf
consentdoc	Appendix A MARKED UP	JAN-18-MARKEDUP-AppendixA-CDQ-Consent.docx
consentdoc	Appendix A CLEAN	JAN18-CLEAN-AppendixA-CDQ-Consent.pdf

11. Data - Confidentiality

Are there conditions in which privacy or confidentiality cannot be guaranteed (e.g., focus groups)?	No

12. Data - Data

Please review the different types of information researchers may seek to collect, use, share and access based on the TCPS 2, Chapter 5

Anonymous information – the information never had identifiers associated with it (e.g., anonymous surveys) and risk of identification of individuals is low or very low.

Anonymized information – the information is irrevocably stripped of direct identifiers, a code is not kept to allow future re-linkage, and risk of re-identification of individuals from remaining indirect identifiers is low or very low.

Directly identifying information – the information identifies a specific individual through direct identifiers (e.g., name, social insurance number, personal health number).

Indirectly identifying information – the information can reasonably be expected to identify an individual through a combination of indirect identifiers (e.g., date of birth, place of residence or unique personal characteristic).

Coded information – direct identifiers are removed from the information and replaced with a code. Depending on access to the code, it may be possible to re-identify specific participants (e.g., the principal investigator retains a list that links the participants' code names with their actual name so data can be re-linked if necessary).

Will this study include a survey or questionnaire?	No
Does this study involve interviews?	Yes
Who will conduct the interview?	Lauren will be conducting all semi-structured interviews. A copy of the general outline of this interview is found in Appendix F
Will any individual(s) other than the research personnel be present during the interview?	No
Will these interviews involve audio recording or, video recording?	Yes
List the procedures that will be recorded.	the recording feature of UM licensed Zoom will be used.
State the purpose of recording.	In order to appropriately anonymize and transcribe the data.
Will participants be permitted to review, edit, and/or erase the recording?	No
Where will this data be stored? How will you maintain participant confidentiality	This information will be stored on OneDrive in password-protected folders. Lauren and Celine will have access to the

	raw audio files. Interview recordings will be saved locally (not to the cloud) and then will be immediately transferred to the secure UM OneDrive location and deleted from local machines. Identifiable data files will be encrypted until they are transcribed and then deleted.
What will happen to the recordings once the data is transcribed? How long will you keep it? If you will destroy the data, when (MMYY)? Please provide justification if identifiable data will be kept indefinitely.	All raw audio files will be deleted as soon as they have been transcribed which, as stated, is 2 weeks from the interview date (05/23)
Who will transcribe the data?	Lauren will transcribe the data manually.
Will the transcripts be anonymized, coded, or identifiable? Please explain	Anonymized. Lauren will be aware of which interview transcript belongs to which participant however will not transcribe or label transcripts with any information that could potentially identify a participant. Once the data is transcribed and anonymized, it will all be kept in a password-protected folder with no codings or identifying features.
Where will this data be stored?	In a private folder in MS Teams (OneDrive) that is only accessible to the investigators, Dr. Latulipe and Lauren Himbeault. This has been reflected in the consent form.
What will happen to the transcripts? How long will you keep it? If you will destroy the data, when (MMYY)? Please provide justification if identifiable data will be kept indefinitely.	Transcripts will be deleted within three years of completing the study (12/27). This is reflected in the consent form

Attach interview questions:

Type	Name	Document
	Appendix F: Interview Doc	F-InterviewQuestions.pdf

Does this study involve focus groups?	No
Does your study involve observations?	No
Are there any other data mediums that are not listed above? This includes paper consent forms, paper assent forms, photos, participant ID keys/codebooks, handwritten notes, observation notes, log files, articifacts (e.g. notes, gifts, cards).	Yes

Please select all of the data you may create, collect, access, store or share during or related to this study

Data Medium	Other

Is this data anonymous, anonymized, coded, indirectly identifiable or directly identifiable? Please explain	Email Communication, Identifiable but stored on UM servers
Where will this data be stored?	All email communication will go through the University's secure servers using university email addresses.
What will ultimately happen to this data? How long will you keep it? If you will destroy the data, when (MMYY)? Please provide justification for why the data will be kept indefinitely.	Emails pertaining to the study will be deleted within three years of completing the study (12/27).
Data Medium	Paper Consent Forms
Is this data anonymous, anonymized, coded, indirectly identifiable or directly identifiable? Please explain	Consent form, Identifiable
Where will this data be stored?	Paper consent forms will be scanned immediately after collection and will be stored on Lauren's password-protected computer in password-protected folders. All paper copies will then be securely shredded
What will ultimately happen to this data? How long will you keep it? If you will destroy the data, when (MMYY)? Please provide justification for why the data will be kept indefinitely.	Consent forms will be deleted within three years of completing the study (12/27).
Data Medium	Artifacts eg. notes, gifts, cards, from participants
Is this data anonymous, anonymized, coded, indirectly identifiable or directly identifiable? Please explain	EGiftcard
Where will this data be stored?	Egiftcards will be sent once an interview has concluded.
What will ultimately happen to this data? How long will you keep it? If you will destroy the data, when (MMYY)? Please provide justification for why the data will be kept indefinitely.	Once a gift card has been accepted, a note will be made on the consent form for confirmation. This data will not be used for anything other than record keeping and will be deleted alongside the consent forms (12/27)
Data Medium	Other
Is this data anonymous, anonymized, coded, indirectly identifiable or directly identifiable? Please explain	Umlearn data will be identifiable when collected but will be anonymized and stripped of all identifying features immediately before being stored on OneDrive with other study data.
Where will this data be stored?	This information will be stored on OneDrive in password protected folders.

What will ultimately happen to this data? How long will you keep it? If you will destroy the data, when (MMYY)? Please provide justification for why the data will be kept indefinitely.	Local anonymized copies will be deleted within three years of completing the study (12/27). This is reflected in the consent form.
Data Medium	Honorarium receipt forms
Is this data anonymous, anonymized, coded, indirectly identifiable or directly identifiable? Please explain	Emails, Identifiable but stored on UM servers. Once a participant has received their honorarium, we will ask them to respond to the email with confirmation of receipt.
Where will this data be stored?	This data is stored on UM secure servers.
What will ultimately happen to this data? How long will you keep it? If you will destroy the data, when (MMYY)? Please provide justification for why the data will be kept indefinitely.	Emails pertaining to the study will be deleted within three years of completing the study (12/27).

13. Data - Data Transfer

Will data be transferred from one site to another?	No
Will data be transferred between research team members?	Yes
Please describe in detail how the data will be transferred. What identifiable data will be transferred? How and where will it be stored? What safeguards will be used to protect the data during transfer and storage?	No data will be transferred. All data will be stored on MS Teams and OneDrive. Dr. Latulipe and Lauren have an MS Teams group that only they have access to. All data will live there so that no data will be transferred via email or other processes. UMLearn data will be in a separate channel on teams which Min Kyu has access to with no access to interview data.
Will the data be archived or made accessible to the public and/or other researchers?	No

14. Risks/Benefits - Benefits

What are the expected benefits of the research?	After completing these investigations, we hope to share the findings with other CS Educators, so they can understand the benefits and drawbacks of using CDQs when teaching programming. We hope to learn how these question types impact student self-efficacy, course performance, and student's understanding of computer programming foundations.
What are the indirect benefits for participants participating in the research?	The indirect benefit to all participants is an opportunity to reflect on the course they took part in and to aid in its future development. Interview participants will benefit from having a lasting impact on how programming is taught and the pedagogical resources used in future versions of the course.
What are the direct benefits for participants participating in the research?	Interview participants have an opportunity to voice their thoughts on COMP 1012 and the structure of the material and help us propose new ways of conducting this course when it is offered again in the future.

15. Risks/Benefits - Risks

What are the risks (psychological, physical, emotional, social, legal, economic, or political) to participants, or to a third party?	The risk to participants is minimal. Interview participants who are not doing well in the course may feel slight discomfort discussing the course.
Provide a description of the risks, the steps that will be taken to reduce or eliminate them, and the steps that will be taken to improve any actual harm to participants, including (if appropriate) providing a list of helpful resources.	It is possible that participants could feel uncomfortable discussing their experiences with Lauren. We will remind all participants that their involvement in the study is totally optional and they do not have to continue if they feel uncomfortable or answer any questions that make them uncomfortable. We will also mitigate the risk of identification by ensuring directly identifiable data is removed from all transcripts. As the sample size is small, it is possible the investigators could remember which transcript maps to which participant. Consent forms include a reminder that data is anonymized, they do not have to answer if they do not want to and no prejudice or action will be taken regarding any positive or negative feedback given during the interview. Students are also reminded that their instructors will not know who chose to participate in the research and will not see any identifiable responses.
Is there a possibility that abuse of children or persons in care might be discovered in the course of the study?	No

16. Dissemination/Withdrawing - Feedback

Will you be providing participants with the opportunity to review their data?	No
How will information from or about your participants be presented (e.g., summary statistics for the whole group, direct quotations from their interviews)?	Course data will be aggregated; participants from interviews will be directly quoted. Each student participant's quotes will be labeled P1, P2, etc. Possible Direct Quote permission is expressed in the consent document.
Will your publications refer to individual participants?	Yes
How will they do so (e.g., by their real name, by a pseudonym, by a general descriptor such as “one female student” or “one factory worker”)?	By pseudonyms with no descriptors. Students will be referred to as S1, S2, etc with no identifying descriptors. If any data requiring gender or classroom section identifiers are required, pseudonyms such as "F1, M1, NB1, etc" will be used, or "a student in the diagrammatic question section" vs " a student in the traditional question style section". Consider how many students know each other in a 300-person course. On top of that, no one other than Lauren and Dr. Celine will know who agree to participate in the release of their data or consented to an interview, and further to that, were selected to participate in the interview. Student confidentiality is of our highest concern and by limiting the individuals who are aware of the identity of the participating students to just Lauren and Dr. Celine, the probability that someone can accurately guess who participated in an interview based solely on a pseudonym is exceedingly low.
How will the research results be disseminated, to whom, and for what intended purpose?	The research will be a part of Lauren's final master's thesis. This is explicitly stated in the attached consent forms. As such this will also be disseminated on MSpace (UofM’s public thesis repository), and also the results may be disseminated in a journal or conference publication.
Steps should be taken to provide participants with a brief, non-technical summary of research results as soon as possible after the data collection phase of the study is completed should they want it. Provide your plans for providing project results to participants. Participants should be given a choice of how they wish to receive a summary and should be told approximately when (MMYY) to expect it.	Participants will be able to read a summary of the research results online on Dr. Celine Latulipe's lab webpage. Participants will be informed that this will be made available within 1 year of Lauren's thesis defense (12/24). Participants will also be informed that they may reach out with any questions about the research or its results via email. This is made clear in the consent form.

17. Dissemination/Withdrawing - Withdrawing

How and when are participants informed of their right to withdraw?	For the interview, participants will be informed of their right to withdraw in the consent form as well as at the start and end of the interview. Participants will not have the ability to revoke the consent given to release their course data anytime after consent form collection as anonymization will begin immediately.
What procedures will be followed for participants who wish to withdraw at any point during the study?	For the interview, participants may withdraw at any time during the process, including at any point during the interview. This information is on the consent form and will be re-iterated at the start of the interview. If a participant wishes to withdraw, they will simply need to e-mail Lauren or say so during the interview. For revoking consent of data release, participants will not be able to do this once forms are collected. The ability to do this is stated in the consent form.
Please indicate what will be done with the participants data when they request to withdraw	When a participant withdraws during the interview, all data collection will stop. If a participant is comfortable with the already collected data being transcribed, that data will be used as specified in the protocol, if not we will then immediately delete all collected data. There are no consequences or penalties for withdrawing from the study and participants who begin the interview will still be compensated, even if they withdraw without completing the interview.
Is there a deadline after which the nature of your data analysis would make it impossible for participants to withdraw? Please provide a MMYY.	Once the study is complete and the audio recordings have been deleted, it wouldn't be possible to withdraw a participant's information as it is all anonymized. Interview participants may withdraw their data from the start of the study until the end of the interview. As interviews will be transcribed within 2 weeks of completion, following that, there will be no ability to withdraw. Interviews will hopefully take place in May which means by April/23 all data will be anonymized.

18. Attachments

Type	Name	Version	Status	Filename	Uploaded Date
Recruitment Documents	AppendixD-Interview-Email.pdf	1	Approved	D-Interview-Email.pdf	11/25/2022
Interview Document	AppendixF-InterviewQuestions.pdf	1	Approved	F-InterviewQuestions.pdf	11/25/2022
Recruitment Documents	AppendixB-MARKEDUP-ConsentDistribution-Script.docx	1	Approved	AppendixB-MARKEDUP-ConsentDistribution-Script.docx	12/19/2022
Recruitment Documents	AppendixB-CLEAN-ConsentDistribution-Script.pdf	1	Approved	AppendixB-CLEAN-ConsentDistribution-Script.pdf	12/19/2022
Recruitment Documents	AppendixC-MARKEDUP-ConsentEmail.docx	1	Approved	AppendixC-MARKEDUP-ConsentEmail.docx	12/19/2022
Recruitment Documents	AppendixC-CLEAN-ConsentEmail.pdf	1	Approved	AppendixC-CLEAN-ConsentEmail.pdf	12/19/2022
Consent Documents	AppendixA-CDQ-Consent.docx	1	Approved	NEW-AppendixA-CDQ-Consent.pdf	12/29/2022
Consent Documents	AppendixE-Interview-Consent-Form.docx	1	Approved	NEW-AppendixE-Interview-Consent-Form.pdf	12/29/2022
Consent Documents	JAN18-CLEAN-AppendixA-CDQ-Consent.pdf	1	New	JAN18-CLEAN-AppendixA-CDQ-Consent.pdf	01/18/2023
Oath of Confidentiality:Min Kyu Jung	oath	1	New	oath_min.pdf	01/24/2023

B Consent Form: In Class



Winnipeg, Manitoba
Canada R3T 2N2
(204) 474-8313
FAX: (204) 474 - 7609

Appendix A - Study Consent Form
Project Title: Studying Code Diagram Query (CDQ) Effectiveness in First-Year Computing Courses

Researchers:

Dr. Celine Latulipe, Associate Professor, Department of Computer Science, University of Manitoba. (celine@cs.umanitoba.ca)

Lauren Himbeault, Masters Candidate, Department of Computer Science, University of Manitoba. (lauren.himbeault@umanitoba.ca)

Min Kyu Jung, Masters Candidate, Department of Computer Science, University of Manitoba. (jungmk-@myumanitoba.ca)

Please take the time to read this carefully and to ensure you understand all the information.

This consent page should give you the basic idea of what this research is about and what your participation will involve. If you would like more detail about something mentioned here, or information not included here, you should feel free to contact Lauren Himbeault (lauren.himbeault@umanitoba.ca). You may also contact Lauren's advisor, Dr. Latulipe as (celine.latulipe@umanitoba.ca).

You are being asked to consent to your data being collected and analyzed in a thesis on the effectiveness of CDQs in first year computing courses as well as consenting to being contacted to participate in an optional interview. This semester you have been using different types of questions as learning aids.

The objective of this study is to investigate the value of these questions for novice programmers. We are looking to see if these questions increased your understanding of computer programming and code execution as well as your own feelings of confidence when coding. We are also investigating if these questions are worthwhile as well as how they can be improved upon to create better experiences for the future. Our hope is to share the CDQ framework with the wider CS education community, so that other CS Educators can use these to teach beginner programming at other institutions.

Participation in this study has two parts. As such, there are two sets of checkboxes below you need to fill out. The first is consenting to releasing your UMLearn information such as your grades, practice question logs, and background survey data. This data will be initially identifiable but stripped of identifying information prior to being analyzed by Lauren and Min Kyu. All data is stored on encrypted, and password protected drives with Lauren, Min and Dr. Celine being the only two with access. Anything published will be clustered and unidentifiable.

The second set of checkboxes are for consent to being contacted for an online interview to discuss your experiences in this class. Lauren will be conducting interviews on UM Zoom, to gather your thoughts and reflections on this course and the use of these diagrammatic questions. Risks to participating in this study are minimal. You may find discussing your experiences, if critical of these questions and the course, causes slight discomfort.

Participation in this study is voluntary and if you agree to participate, please note that withdrawal from data release will not be possible as data anonymization will start immediately making it impossible to differentiate participant data. If you agree to participate in the interview but change your mind at any point, you may email lauren.himbeault@umanitoba.ca and your name will be removed from potential interview candidates. All consenting interview candidates will receive a 're-invitation' email with details on coordinating the interview providing you are still willing and able to participate. You will have the opportunity to request a withdrawal from the interview portion of the study during and within the 2 weeks following the interview as well. There is no penalty for choosing to not participate or withdrawing your data. A summary report of the research will be available on Dr. Latulipe's website (https://celinelatulipe.net) and Lauren's full thesis will be available on her website following the successful defense of the thesis (www.laurenhimbeault.com). The expected date of defense is August 2023 and so the availability of the report and thesis will be available before the end of 2023. By consenting to release your course performance and logs for research purposes, you indicate that you have understood to your satisfaction the information regarding the research project. In no way does this waive your legal rights nor release the researchers, sponsors, or involved institutions from their legal and professional responsibilities. You are free to withdraw from the study at any time, and /or refrain from answering any questions you prefer to omit, without prejudice or consequence. Your continued participation should be as informed as your initial consent, so you should feel free to ask for clarification or new information throughout your participation. This research will be submitted as part of Lauren's master's thesis and be available on MSpace. The results of this research may also be presented at future academic conferences and those papers which are accepted will also be available on the above listed websites.

The University of Manitoba may look at your research records to see that the research is being done in a safe and proper way.

This research has been approved by the Research Ethics Board at the University of Manitoba, Fort Garry campus. If you have any concerns or complaints about this project, you may contact any of the above-named persons or the Human Ethics Coordinator at 204-474-7122 or

humanethics@umanitoba.ca. You should feel free to keep a digital copy of this consent form for your records and reference, by saving it to your hard drive.

The benefits of this research are:

After completing these investigations, we hope the findings will help other CS Educators understand the benefits of Code Diagram Queries. We hope to discover the value of these question types such as increased self-efficacy, high course performance, deeper understanding of computer programming foundations and going forward be able to create a more valuable experience for students.

The indirect benefits to the participants are an opportunity to reflect on the course they took part in and be a voice in shaping it for the future.

By checking this box, I certify that I consent to the release of my course grades and runestone logs for Lauren's research:

☐ I consent ☐ I do not consent

By checking this box, I indicate my consent to be contacted in the future for a short interview:

☐ I consent ☐ I do not consent

By checking this box, I indicate that I understand and consent to my **anonymized** data being analyzed, and the results will be published in Lauren's thesis:

☐ I consent ☐ I do not consent

Name [Printed]: _____

Signature: _____

C In Class Script: Consent Form Distribution and Recruitment

Appendix B Script For Consent Form Distribution
Project Title: Studying Code Diagram Query (CDQ) Effectiveness in First-Year Computing Courses

Hello class and thank you for your time

I am Lauren Himbeault and I am a graduate student in the HCI lab here at the university of Manitoba. I am doing research in Computer Science Education. I am also the course director for COMP 1012 this term. If you haven't seen me before, that would be because my role does not have a connection to you as students. As someone who has taught this class many times before, my job is to make sure your instructors are staying consistent between the two sections of this course and teaching you the material you are expected to learn throughout the term. It's important to understand this because it means that even though I have access to your course this term, I do not make any decisions around your marks. I do not grade assignments or labs or tests, nor do I assign any grades to any evaluative assessments you may have. My reason for being here today is regarding my graduate studies.

I am studying ways to help first year students learn to program successfully. I am investigating different types of questions that have been used as practice in this course alongside your course performance to gauge how well the resources you are given are helping you learn and understand what computers are doing. I am also interested in seeing how a student's confidence in their skills may affect how they interact with the course materials. I would like access to your course performance and logs from this term as part of my research. Your instructors will not have access to these consent forms, and therefore not know who is consenting or not at any point during this process, including after the term ends. Your instructors do not have any connection to my research. There is no penalty for not consenting to the release of your information. All data collected will be aggregated and anonymized so your name, student number, or any identifying information will be stripped from the data set prior to analysis and being included in my final thesis.ß

I will then distribute and read the consent forms aloud

I will now give you a few minutes to review the form I have just distributed and read to you, and you may ask me any questions or voice any concerns you may have.

Please fill out the consent form in its entirety and return it to me before you leave today.

D Email Consent Form: Absent Students

Appendix C – Consent EMAIL
Project Title: Studying Code Diagram Query (CDQ) Effectiveness in First-Year Computing Courses

Subject Title: COMP 1012 – Data Release consent: Studying Code Diagram Query (CDQ) Effectiveness

Hello,

I am Lauren Himbeault, and I am a graduate student in the HCI lab here at the university of Manitoba. I am writing to you I am doing research in Computer Science Education. I came to your class this week, but you were not present, and I want to be sure you have equal opportunity to respond to this request. I am studying ways to help first year students learn to program successfully. I am investigating different types of questions that have been used as practice in this course alongside your course performance to gauge how well the resources you are given are helping you learn and understand what computers are doing. I am also interested in seeing how a student's confidence in their skills may affect how they interact with the course materials. I would like access to your course performance and logs from this term as part of my research. Your instructors will not have access to who consenting at any point during or after the course. There is no penalty for not consenting to the release of your information. I have attached the consent form distributed in class. All data collected will be aggregated and anonymized so your name, student number, or any identifying information will be stripped from the data set prior to analysis and being included in my final thesis.ß
ß

Please reply to this email directly and state whether you consent the release of your information or not.

--- I will attach the Consent form as seen in Appendix A---

If you have any questions or which to discuss any concerns you may have, please do not hesitate to email me back. You may also contact my advisor, Celine Latulipe at celine.latuliple@umanitoba.ca if you have any questions you would prefer to direct her way.

I look forward to hearing from you.

E Email Script: Absent Student Recruitment

Appendix D - Interview Recruitment Email

Project Title: Studying Code Diagram Query (CDQ) Effectiveness in First-Year Computing Courses

Good day,

I am investigating the effectiveness of CDQs in beginner programming classes. If you are receiving this email, it is because you consented to being contact by me in the consent form you filled out in COMP 1012.

My goal of this research is to see if these CDQ questions help students form a better understanding of how computer code interacts with the system and if incorporating more of these questions would benefit future students when learning to program.

As a reminder, you have been completing various formative assessments to serve as practice for you through the term. I am interested in knowing if you found these questions to be beneficial to your learning as well as identify any challenges you may have experienced.

Benefits:

- You will help me understand the pros and cons of these CDQ questions across various traits such as usability, performance, and self-efficacy
- As a student of COMP 1012, you will have an opportunity to reflect on the course you took and have a voice in shaping it for the future

This study will consist of a 30-minute interview. The interview will be on UM Zoom and audio will be recorded. For your time, you will be given a $15 coffee e-gift card as compensation.

I understand it may be difficult to be critical of course materials, please know that all interviews will be transcribed and anonymized before any data is reviewed and analyzed. Anything you say will have no negative effects or penalties on you. You do not have to say anything you do not want to, nor do you have to answer any questions you do not wish to answer. Any direct quotes that may be used will have identifying data stripped. Your instructors will not have access to who is being interviewed or what is being said at any point during this process.

If you are still interested in participating or have any questions, please fill out the attached consent form and email it back to me at: lauren.himbeault@umanitoba.ca. We will set up an interview on Zoom at your earliest convenience. Please don't hesitate to reach out if you have additional questions or concerns as well.

If you would like to contact my advisor, Celine Latulipe, her email can be found below as well:
Celine.Latulipe@umanitoba.ca

This research has been approved by the Research Ethics Board at the University of Manitoba, Fort Garry campus. If you have any concerns or complaints about this project you may contact any of the above-named persons or the Human Ethics Coordinator at 204-474-7122 or humanethics@umanitoba.ca.

Thank you for your time and I look forward to hearing from you,

Lauren Himbeault
Department of Computer Science
University of Manitoba

F Consent Form: Interview



Winnipeg, Manitoba
Canada R3T 2N2
(204) 474-8313
FAX: (204) 474 - 7609

Appendix E - Interview Consent Form

Project Title: Studying Code Diagram Query (CDQ) Effectiveness in First-Year Computing Courses

Researchers:

Dr. Celine Latulipe, Associate Professor, Department of Computer Science, University of Manitoba. (celine@cs.umanitoba.ca)

Lauren Himbeault, Masters Candidate, Department of Computer Science, University of Manitoba. (lauren.himbeault@umanitoba.ca)

Please take the time to read this carefully and to ensure you understand all the information.

This consent page should give you the basic idea of what this research is about and what your participation will involve. If you would like more detail about something mentioned here, or information not included here, you should feel free to contact Lauren Himbeault (lauren.himbeault@umanitoba.ca). You may also contact Lauren's advisor, Dr. Latulipe as (celine.latulipe@umanitoba.ca).

You are being asked to consent to being interviewed on your experiences in COMP 1012 for Lauren's thesis on the effectiveness of CDQs in first year computing courses. This semester you've interacted with different types of questions as learning aids.

The objective of this study is to investigate the value of these questions for novice programmers. We are looking to see if these questions increased your understanding of computer programming and code execution as well as your own feelings of confidence when coding. We are also investigating if these questions are worthwhile as well as how they can be improved upon to create better experiences for the future. Our hope is to share the CDQ framework with the wider CS education community, so that other CS Educators can use these to teach beginner programming at other institutions.

Participation in this study is voluntary and if you agree to participate, you can withdraw your participation at any time. You do not have to answer any questions you do not want to without any penalty or prejudice. There is no penalty for withdrawing consent.

The interview will take place on UM Zoom. The interview will take place at a time that is convenient to you and should last between 30 minutes. Interviews are recorded (audio and video) however all video recordings are immediately deleted, there is no requirement to have your video on during the interview. All audio files are stored in encrypted, and password protected folders with Dr. Celine and Lauren being the only two individuals with access to them. Lauren will then transcribe the interview (within 2 weeks of the interview taking place) and delete all audio recordings at this time. The transcripts will be stored in encrypted, and password protected folders with Dr. Celine and Lauren being the only two individuals with access. Lauren will be conducting the interviews alone and manually transcribing and anonymizing the audio files for subsequent analysis. It is possible that direct quotes will be included in the final report. No names or identifiers will accompany direct quotes. If direct quotes are used, pseudonyms such as "S1", for student 1 will be used or "a female identifying student said …". Any negative or critical feedback is welcome and will not be met with any penalty at any future point. As a thank-you for your time, a $15 e-gift card (Tim Hortons) will be sent to you.

If you consent to an interview on this form and change your mind, you can also withdraw from the study during the interview or after the interview by emailing lauren.himbeault@umanitoba.ca and request to withdraw your information. Audio files will be transcribed and anonymized within 2 weeks of the interview, after this time, withdrawing participation will not be possible as data is no longer identifiable.

All additional study data (transcripts, notes, consent forms) will be deleted from OneDrive **within three years of completing the study (12/27)**.

Lauren's full thesis will be available on her website following the successful defense of the thesis (www.laurenhimbeault.com). The expected date of defense is August 2023 and so the availability of the report and thesis will be available before the end of 2023. This research will be submitted as part of Lauren's master's thesis and be available on MSpace. The results of this research may also be presented at future academic conferences and those papers which are accepted will also be available on the above listed websites.

By consenting to participate in this study, you indicate that you have understood to your satisfaction the information regarding participation in the research project and agree to participate in the interview. In no way does this waive your legal rights nor release the researchers, sponsors, or involved institutions from their legal and professional responsibilities. You are free to withdraw from the study at any time, without prejudice or consequence. Your continued participation should be as informed as your initial consent, so you should feel free to ask for clarification or new information throughout your participation.

The University of Manitoba may look at your research records to see that the research is being done in a safe and proper way.

This research has been approved by the Research Ethics Board at the University of Manitoba, Fort Garry campus. If you have any concerns or complaints about this project, you may contact any of the above-named persons or the Human Ethics Coordinator at 204-474-7122 or

humanethics@umanitoba.ca. You should feel free to keep a digital copy of this consent form for your records and reference, by saving it to your hard drive.

The benefits of this research are:

After completing these investigations, we hope the findings will help other CS Educators understand the benefits of Code Diagram Queries. We hope to discover the value of these question types such as increased self-efficacy, high course performance, deeper understanding of computer programming foundations and going forward be able to create a more valuable experience for students.

The indirect benefits to the participants are an opportunity to reflect on the study they took part in and be a voice in shaping it for the future.

By checking this box, I certify that I consent to participating in the interview and having it recorded:

☐ I consent

☐ I do not consent

Name [Printed]: _____

Signature: _____

G Interview: Question Set

Appendix F – Semi-structured Interview Questions

Project Title: Studying Code Diagram Query (CDQ) Effectiveness in First-Year Computing Courses

Recording will start immediately, is that ok?

Thank you for agreeing to participate in this interview. As a reminder, you may withdraw your consent at any time. You do not have to answer any questions you do not want to, and you may stop the interview at any time. The interview is approximately 30 minutes in length but as this interview is semi-structure, the time may vary slightly. Here is your gift card as an honorarium for participating,

Do you have any final questions for me before we begin?

Questions such as the following cover the general objectives of the semi-structured interview

1. Please tell me a bit about your background such as your year of university and any previous programming exposure you may have had
2. What lead you to take COMP 1012?
3. Thoughts about computer science when coming into COMP 1012?
4. Tell me about how you used the Runestone textbook
5. **[Show pictures of example CDQs)**
    a. Questions like these from class, thoughts on them?
    b. Question type preference? Why or why not?
6. In a course with no interactive textbook/practice questions, how would you study?
7. Can you describe your confidence with respect to continuing with additional computer science classes?
8. What would you tell a friend asking about coding and how you learned in it?

H Certification: TCPS Core 2



I Weekly Question Sets

I.1 Topic 1: Python Basics

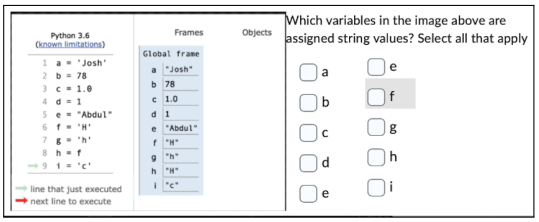


Figure 1: CDQ Question 1

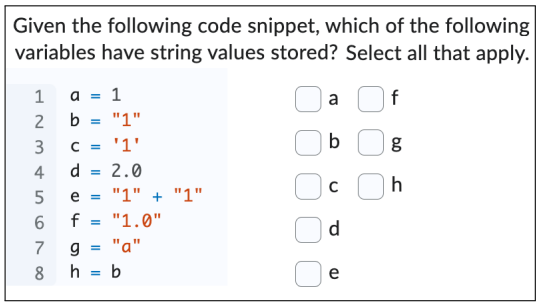


Figure 2: nCDQ Question 1

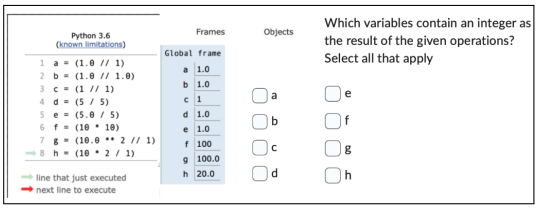


Figure 3: CDQ Question 2

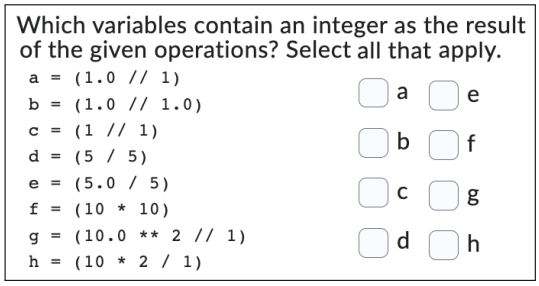


Figure 4: nCDQ Question 2

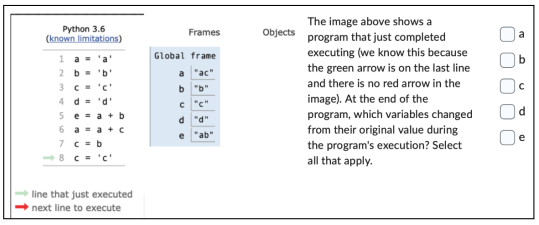


Figure 5: CDQ Question 3

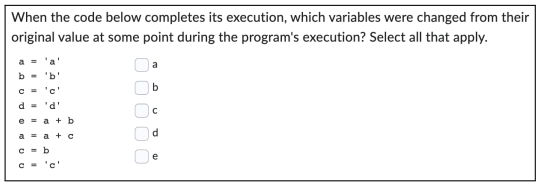


Figure 6: nCDQ Question 3

I.2 Topic 2: Strings and User Input

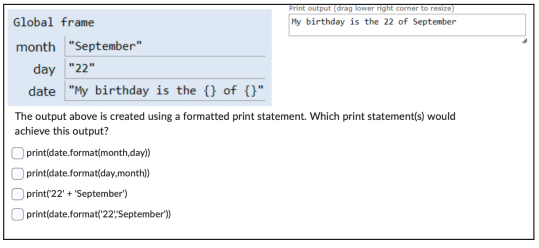


Figure 7: CDQ Question 1

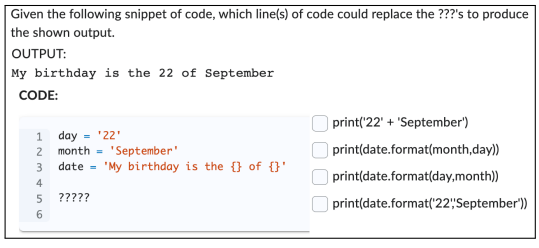


Figure 8: nCDQ Question 1

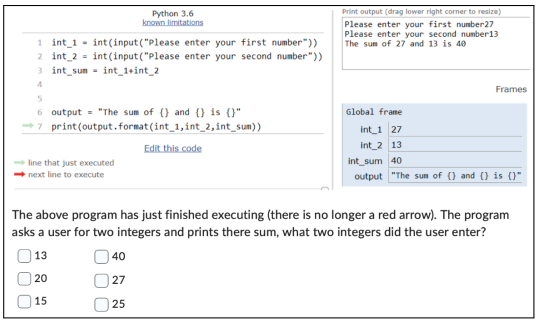


Figure 9: CDQ Question 2

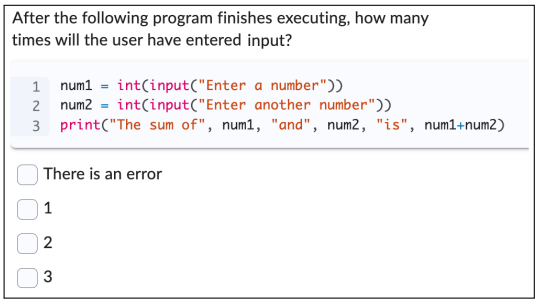


Figure 10: nCDQ Question 2

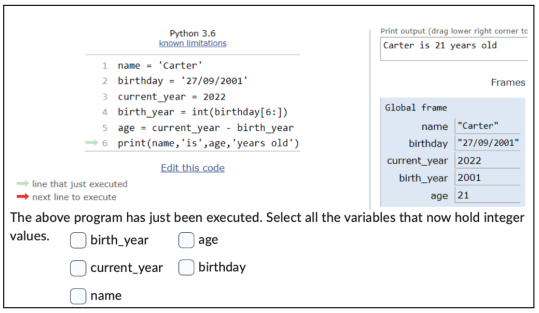


Figure 11: CDQ Question 3

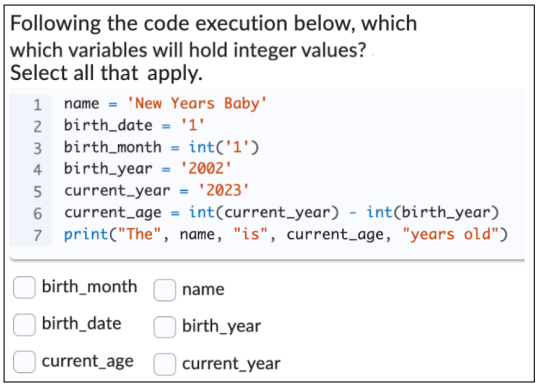


Figure 12: nCDQ Question 3

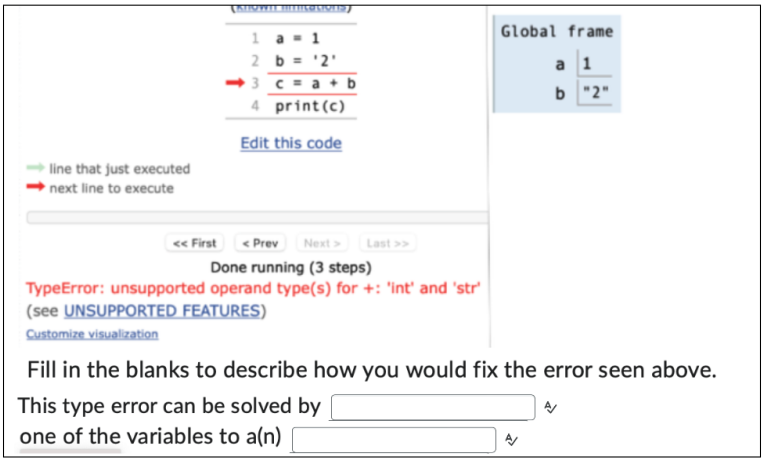


Figure 13: CDQ Question 4

I.3 Topic 3: Conditionals (If/While)

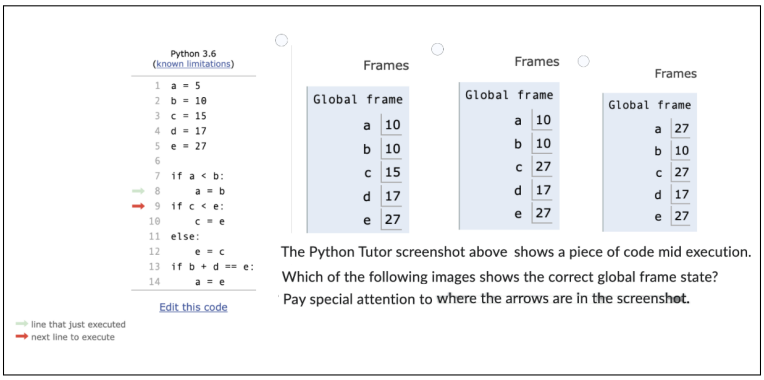


Figure 14: CDQ Question 1

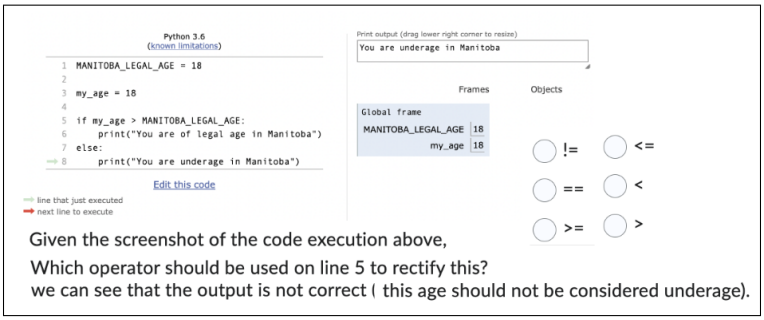


Figure 15: CDQ Question 2

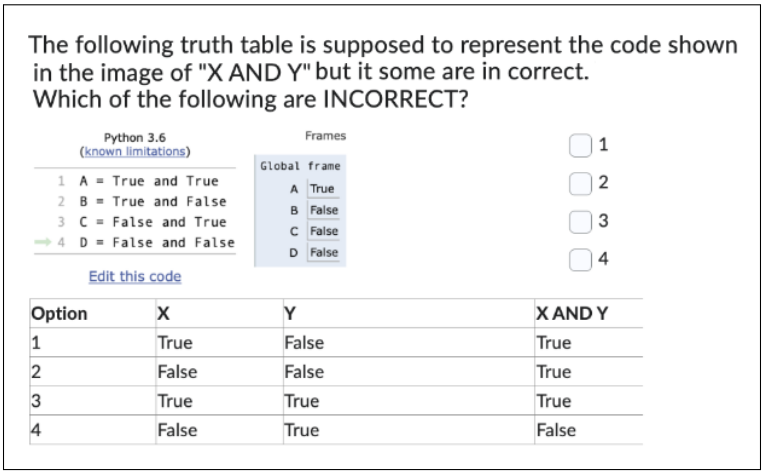


Figure 16: CDQ Question 3

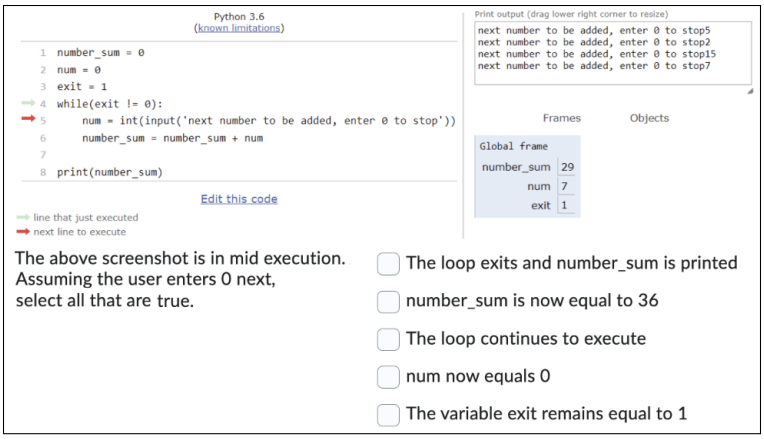


Figure 17: CDQ Question 4

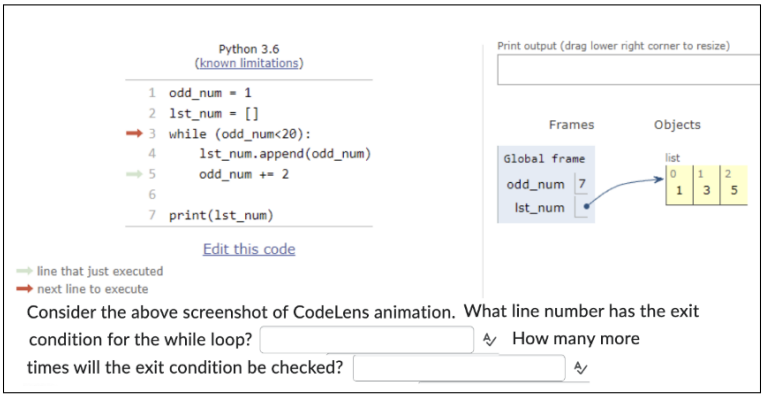


Figure 18: CDQ Question 5

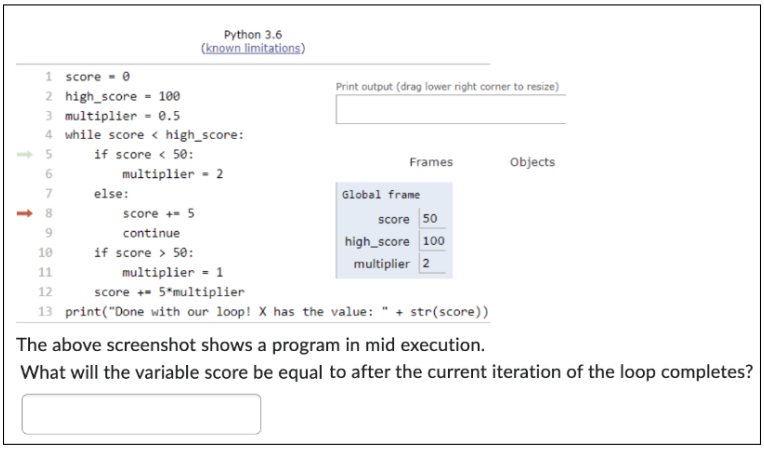


Figure 19: CDQ Question 6

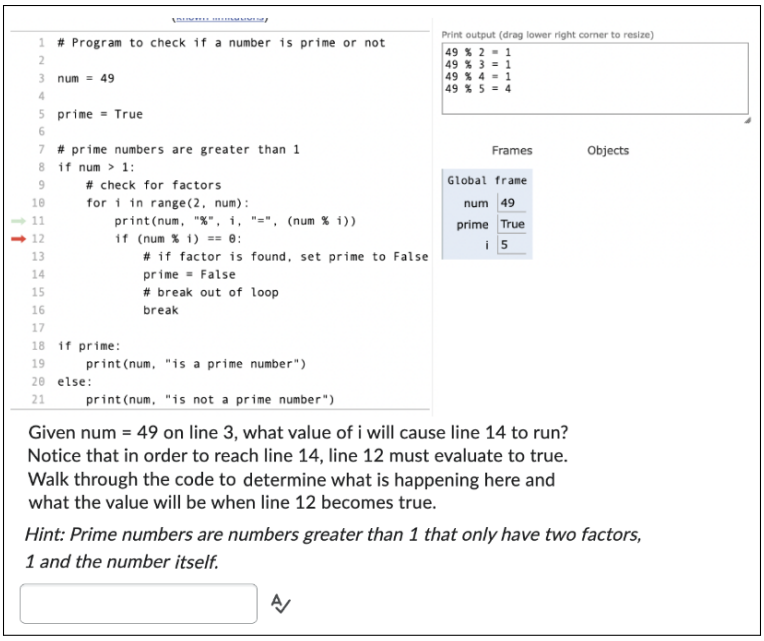


Figure 20: CDQ Question 7

I.4 Topic 4: For loops, Files, Lists

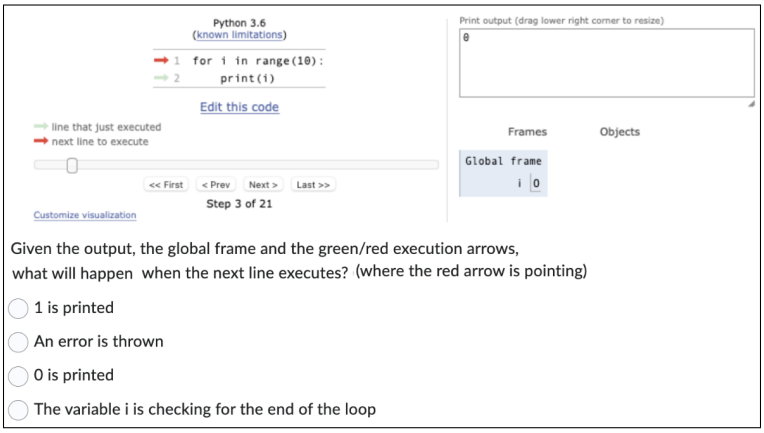


Figure 21: CDQ Question 1

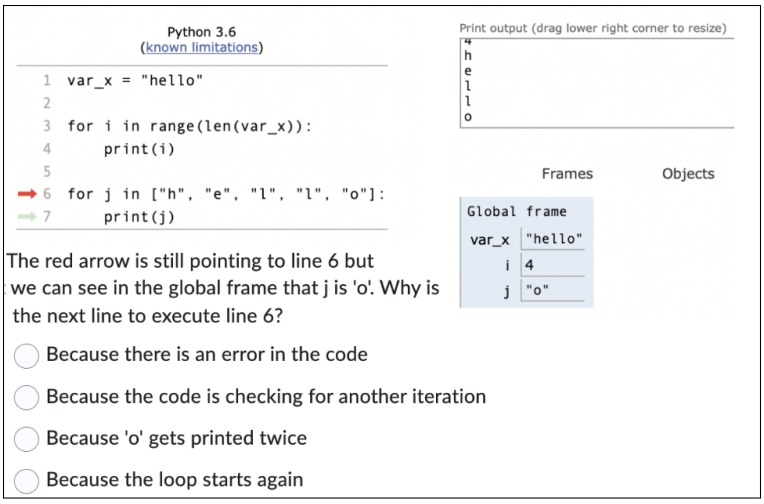


Figure 22: CDQ Question 2

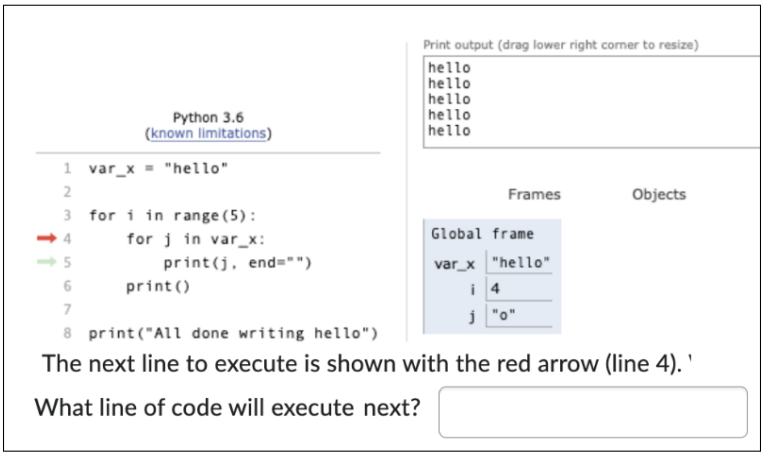


Figure 23: CDQ Question 3

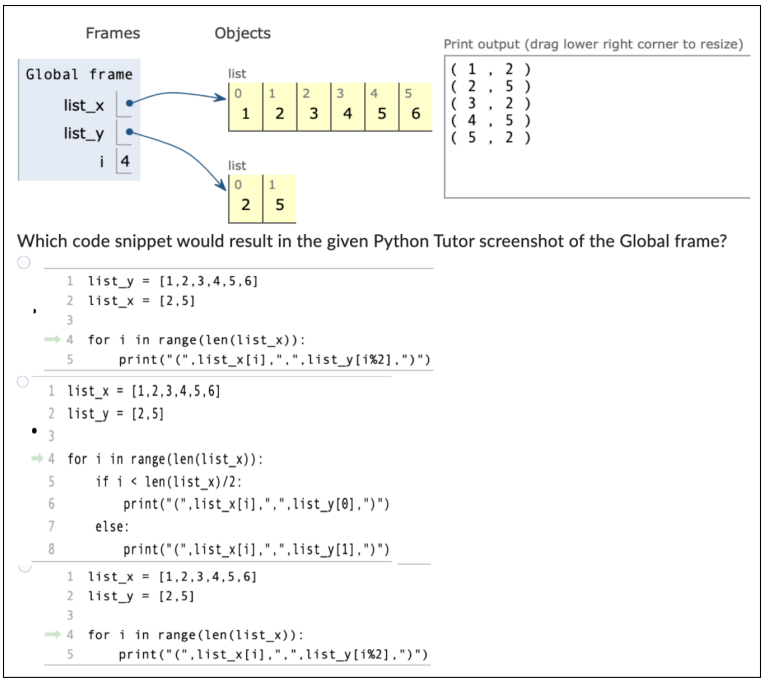


Figure 24: CDQ Question 4

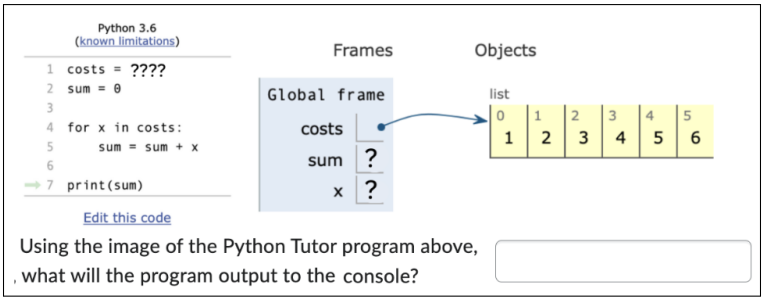


Figure 25: CDQ Question 5

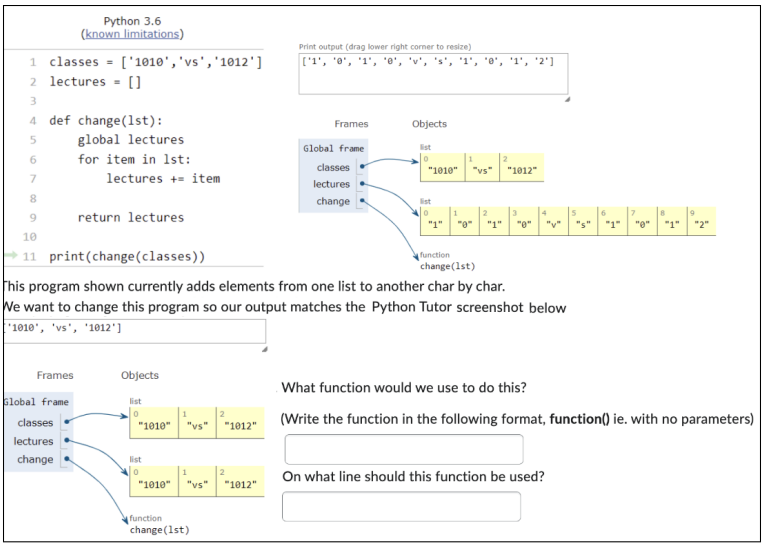


Figure 26: CDQ Question 6

I.5 Topic 5: Lists, Sets, Dicts, Tuples

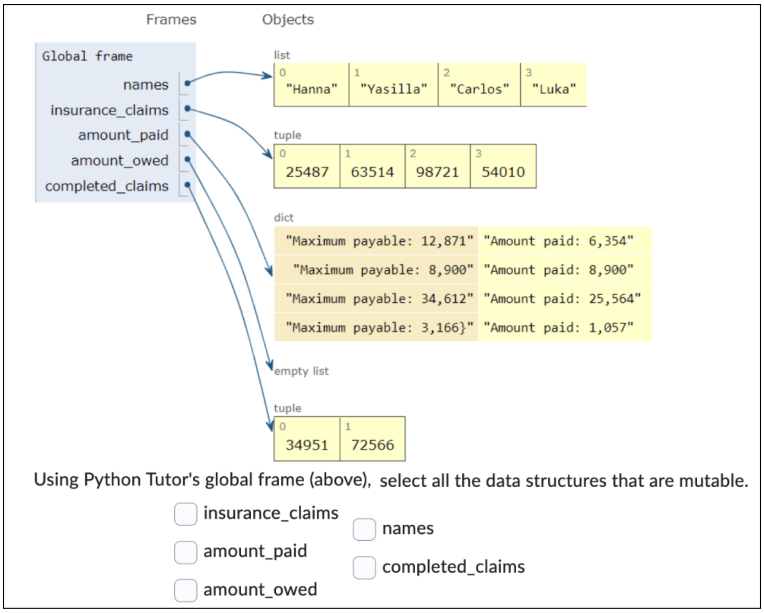


Figure 27: CDQ Question 1

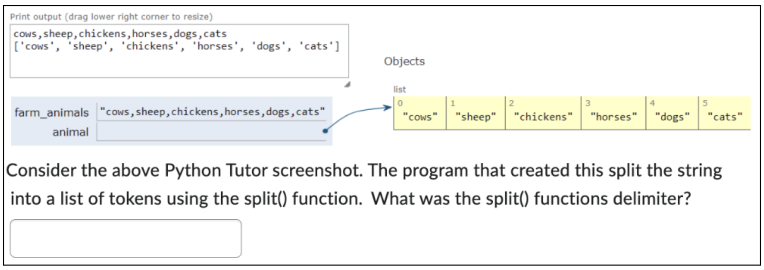


Figure 28: CDQ Question 2

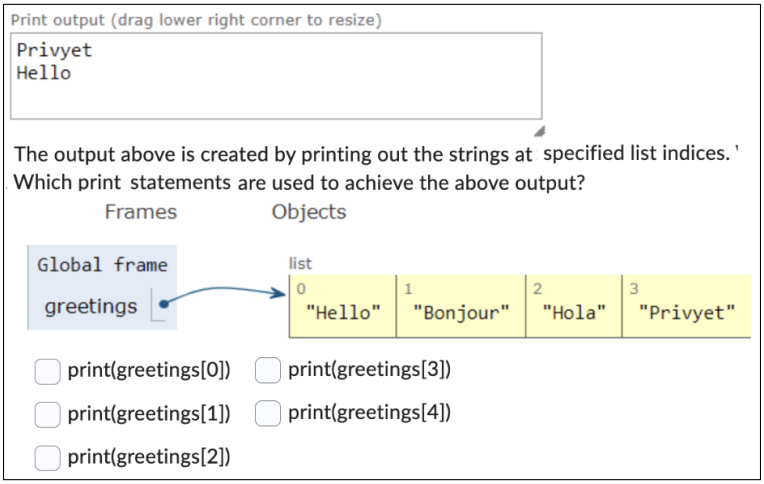


Figure 29: CDQ Question 3

I.6 Topic 6: Functions & Scope

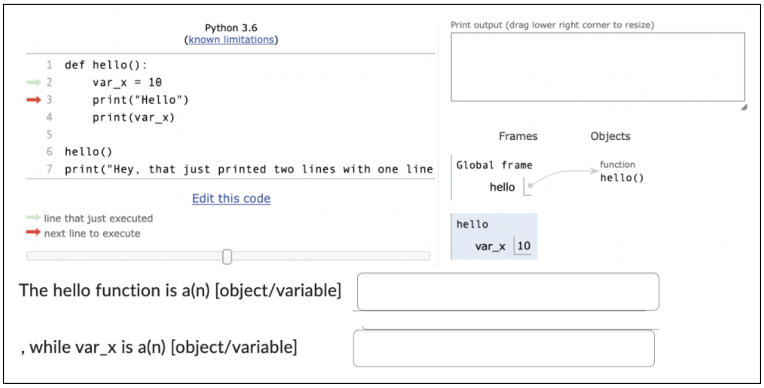


Figure 30: CDQ Question 1

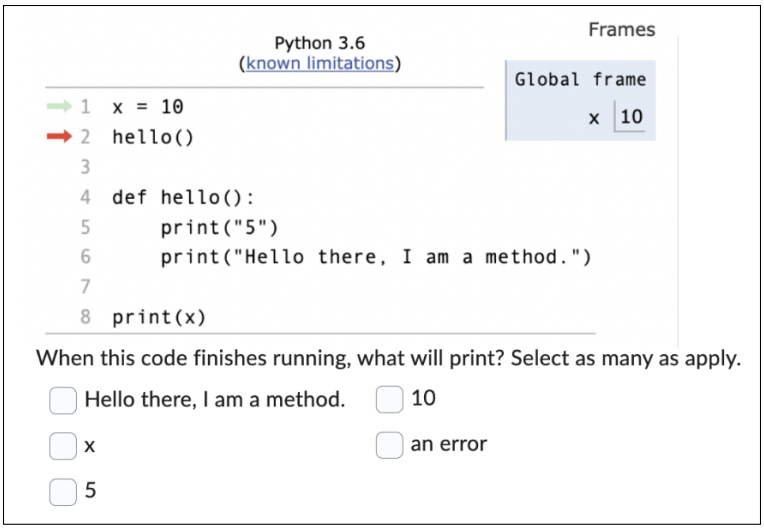


Figure 31: CDQ Question 2

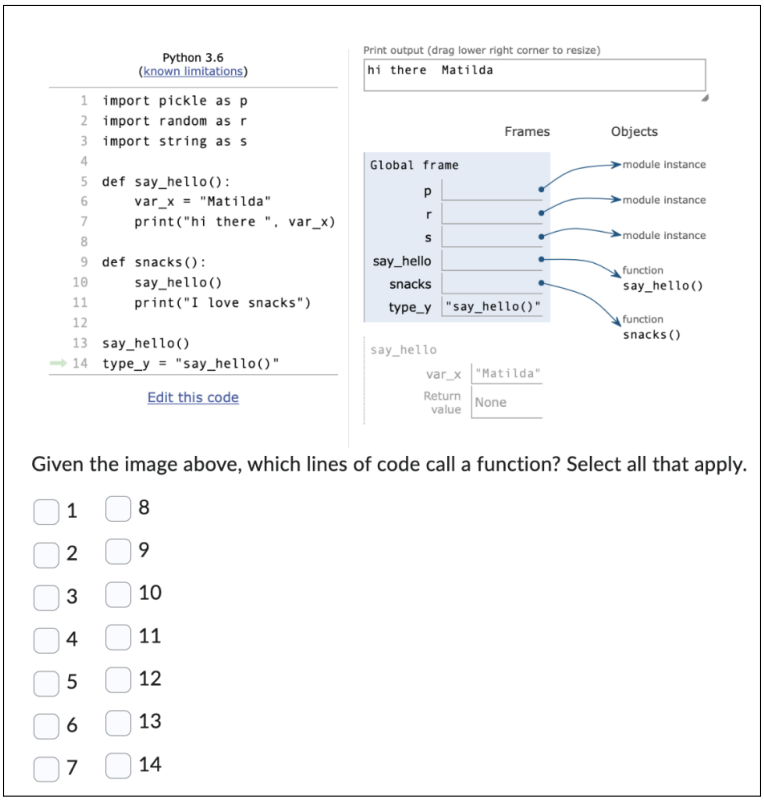


Figure 32: CDQ Question 3

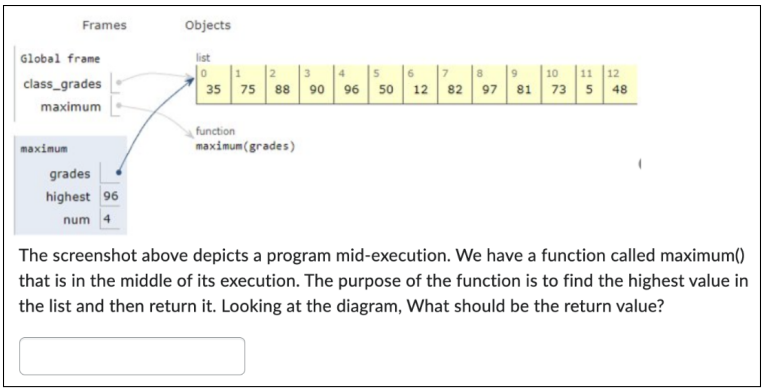


Figure 33: CDQ Question 4

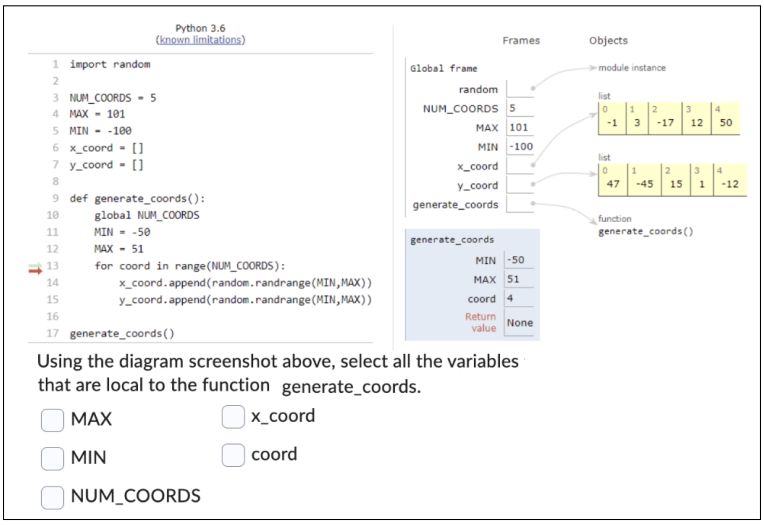


Figure 34: CDQ Question 5

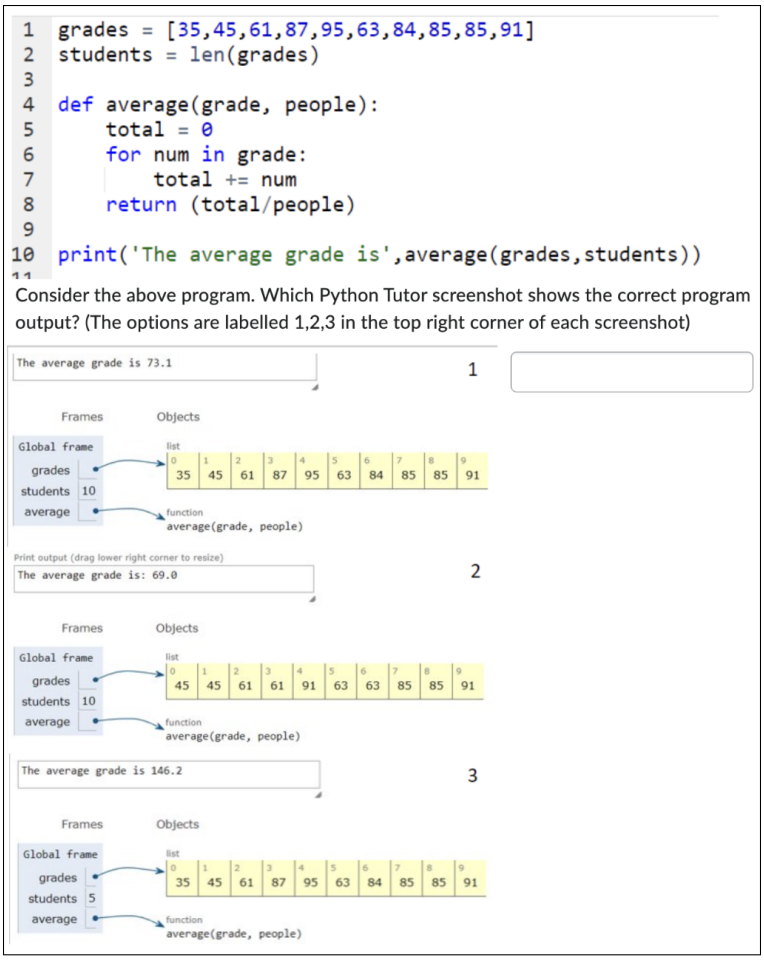


Figure 35: CDQ Question 6

I.7 Topic 7: Random Numbers

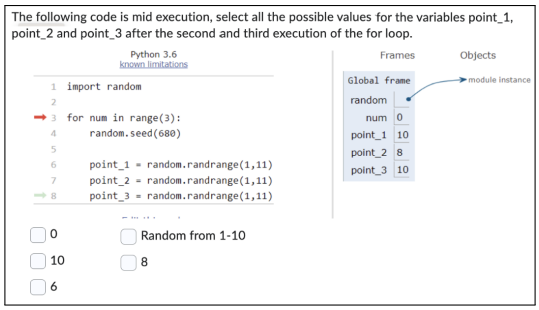


Figure 36: CDQ Question 1

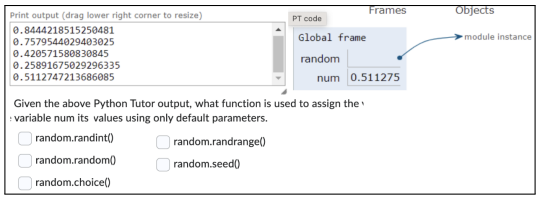


Figure 37: CDQ Question 2

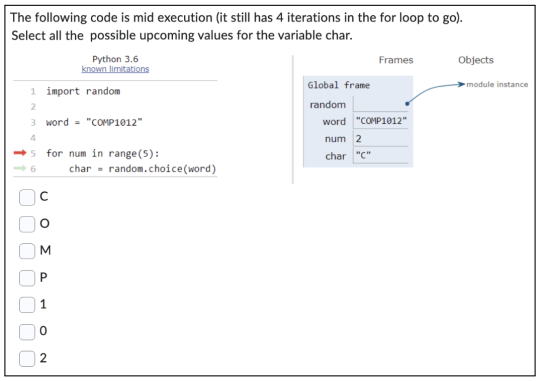
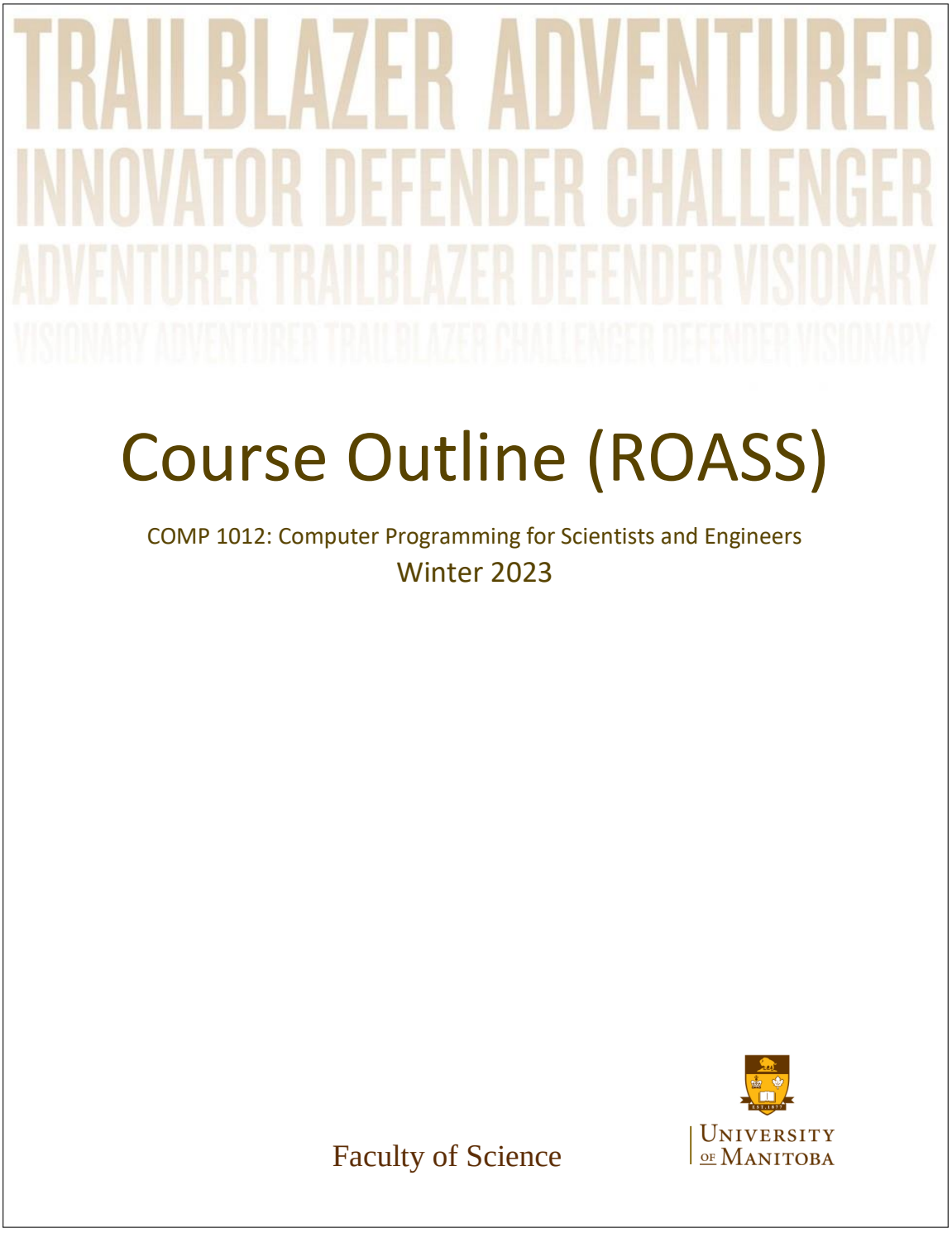


Figure 38: CDQ Question 3

J COMP 1012 Winter 2023 ROASS



Syllabus for COMP 1012: Programming for Engineers and Scientists

DETAILS, DESCRIPTIONS AND OBJECTIVES

Details

Course Title & Number:	COMP 1012: Programming for Engineers and Scientists
Number of Credit Hours:	3
Class Times & Days of Week:	A01 – TR 2:30 – 3:45 pm A02 – MWF 11:30 am – 12:20 pm
Location for classes/labs/tutorials:	A01 – 100 Fletcher Argue A02 – 223 Wallace
Pre-Requisites:	40S Mathematics, or equivalent.
Co-Requisites:	MATH 1230 or MATH 1500 or MATH 1501 (or equivalent).

Instructor Contact Information

(A01) Name	**Japjeet Singh**
Office Hours or Availability:	1-2pm Tuesdays and 5-6pm Thursdays IN PERSON
Office Location	E2-460
Email:*	Japjeet.singh@umanitoba.ca
(A02) Name	**Dr. Meng Cheng (MC) Lau**
Office Hours or Availability:	MWF 3:00 pm - 4:00 pm (book on Webex)
Office Location	
Email:*	Meng.Lau@umanitoba.ca

Instructor Consultation Times

In-person and online consultation will be available. Online drop-in consultation sessions and individual appointments will be available through Webex in UM Learn. Please see UM Learn for available times.

Students are encouraged to contact instructors outside of class as soon as difficulty is encountered. In this course, later topics depend on a strong understanding of earlier topics, and I want you to regularly evaluate your understanding and see me for help as needed.

Emailing your instructor

To ensure that we see your email:

- Put "[COMP 1012]" in the subject, followed by a meaningful description (e.g. "[COMP 1012] Missed Test #2").
- Send from your @myumanitoba.ca university email account or use the e-mail tool in UM Learn.
- **MOST MATTERS SHOULD BE POSTED IN PIAZZA. Please post any course related questions there for the most efficient responses**

E-mail Policy

All e-mail communication must conform to the U of M Electronic Communication with Students Policy.

Instructors will respond to e-mail received on a weekday within one business day. Email on evenings, weekends, and holidays will generally be answered on the next business day. Responses to emails asking for review of grades may take several days, to allow instructors to review all requests at once, or to allow for time for discussion with graders.

Syllabus for COMP 1012: Programming for Engineers and Scientists

Email should be used only for personal matters (e.g., missed test, grade inquiry). **All questions about course material, assignments, labs, etc. should be posted on the course discussion forum**. Replies to assignment questions received through email will be posted to the discussion forum, so that all students can benefit. In large classes, debugging help is not available by email. Code sent by email will not be reviewed. To discuss code with an instructor or teaching assistant, you must attend office hours.

The course discussion forum (Piazza) is the fastest way to receive a response to a question about course material or assignments. Students may choose to post anonymously. Note that when you post anonymously your name is hidden from other students, but still visible to instructors and teaching assistants.

Course Descriptions

U of M Course Description: An introduction to computer programming suitable for solving problems in science and engineering. Students will implement algorithms for numerical processing, statistical analysis, and matrix operations. Not to be held with COMP 1010, COMP 1011 or COMP 1013. Prerequisite: Mathematics 40S or equivalent. Co-requisite: MATH 1230 or MATH 1500 or MATH 1501 (or equivalent).

General Course Description

All scientists and engineers today use computer programs daily, to automate tasks and process data. Even if you are not the original author of those programs, there will be times that you need to modify the programs. It is therefore important that all scientists and engineers have an understanding of basic programming concepts, and the ability to edit and write computer programs. The knowledge you will gain this term includes how to instruct a computer to make choices, perform repetition, and store collections of data. While the programming language used in this course is Python, the programming concepts are almost universal, and you will be able to transfer your skills easily to other languages.

This course is an introduction to implementing simple algorithms by writing computer programs. The course does not require previous programming experience. However, many students find that if they have never written any computer programs or have never been formally introduced to algorithms before, the pace can be challenging. If you would like to learn more about the subject of Computer Science, while also picking up important programming fundamentals, you may want to consider taking COMP 1500 before this course.

In this course students will learn to define and use variables, functions, conditional expressions, iteration via loops, read data from files stored on a computer, and create simple data structures such as arrays, lists, sets, and dictionaries. Recursion and an introduction to object-oriented programming are also discussed. Examples and problem sets will be drawn from all the Sciences (from Physics to Statistics and everything in-between) including those of particular interest to Engineers.

Course Goals

- By the end of this course students will:
- Have a robust mental model for how the computer executes programming instructions, and makes execution flow decisions, by accessing information from memory, performing calculations, and storing results in memory.
- Write and run moderately complex programs using a procedural programming language.
- Devise solutions to simple problems and implement them as computer programs.
- Read and evaluate written programs.
- Describe basic programming concepts and structures in plain English.
- Represent ideas and information in a way that computers can understand and act on.
- Implement and use data structures to solve a problem, with emphasis on arrays, lists, sets, and dictionaries.
- Write software that reads and processes data from files stored on disk.

Syllabus for COMP 1012: Programming for Engineers and Scientists

Course Learning Objectives

A list of learning objectives for each week is provided in UM Learn.

Website

UM Learn is used in this course. Please make sure that you can access COMP 1012 in UMLearn and spend some time browsing to familiarize yourself with the layout.

You are responsible for monitoring the website (UM Learn) for course announcements. You should also regularly visit the discussion forum (Piazza) – you will find useful help when doing assignments and labs or studying for tests. Note that you must not post solutions – not even snippets of solutions – there or anywhere else.

Software and Computer Requirements

Students enrolled in this course must ensure they satisfy the following minimum technological requirements:

1. A laptop or desktop computer where you can create and edit documents and install software (note that for this class you cannot use a tablet computer, such as an iPad. You must use a fully functional laptop or computer).
2. An internet connection capable of streaming videos and downloading software, and
3. Access to a webcam and microphone. Use of a webcam is strongly recommended when attending online office hours.
4. The programming language used in COMP 1012 is Python. You must download and install a Python Integrated Development Environment (IDE). For best results, and compatibility with course requirements, you should use the Anaconda Distribution. For the Fall 2023 term, we're using the Python 3.9 version.
5. You must create an iClicker account, and either install the iClicker app or use a web browser to respond to questions during class. See the separate document in UM Learn on setting up and using iClicker.
6. It is strongly recommended that you use removable storage or cloud storage to back up your work. Computer failure is not an acceptable excuse for late assignments.

Textbook

- **Optional:** *Primer on Scientific Programming with Python (5th Edition)*, 2016 by Hans Petter Langtangen.
  You can download a free PDF of this book via the Library's SpringerLink.
  This textbook is a good reference for anyone that will be writing programs that process numerical data. Optional practice problems will be assigned from this book.
- **Optional:** *Python: A Visual QuickStart Guide (3rd Edition)*, 2013 by Toby Donaldson.
  This textbook is a good starting point for students that have not completed a computer programming course in high school. It is an introduction to many topics we will cover and provides additional examples.
- Links for additional (online, free access) readings are provided in UMLearn

Syllabus for COMP 1012: Programming for Engineers and Scientists

LEARNING APPROACHES

Attendance: Since attendance is so crucial for learning in class, attendance is mandatory. Participation marks are recorded and tracked in each class.

Flipped classroom: This course is taught in a flipped classroom format, meaning you will encounter part of the course material initially by assigned resources done BEFORE the class time. Classroom time will be used for interactive lectures, in-class exercises, group discussions, and peer editing. There may be a period of direct instruction as needed but the expectation is the lectures are watched beforehand. Here is a video that tells you more about the flipped classroom style: https://www.youtube.com/watch?v=qdKzSq_t8k8

In-class Activities: These activities will vary from week to week and lend themselves to the material taught and the assignments. These may be group or individual activities.

UMLearn: This course relies heavily on UMLearn. Students should familiarize themselves with this interface. See your instructor if you have questions about using or accessing UMLearn.

Piazza: The discussion forum (Piazza) is available for students to ask questions and discuss course material. Posts can be made anonymously, so no one should feel uncomfortable posting or responding to questions. In a large class, with good participation, you may be able to get a very fast response to your postings. The discussion board is as useful as you make it. Everyone who participates contributes to the common good.

The one restriction with the discussion board is that you cannot post any complete or partial answers to the labs or assignments. Hints to other students can only be of a very general nature. For example, if an assignment requires you to use a dictionary, posts should not provide code relevant to the assignment, and should instead focus on discussion of how to create and use a dictionary.

Posting as anonymous means that you are anonymous to your classmates. Instructors and teaching assistants know who is posting. Credit will be given whether posts are anonymous or not.

The Respectful Work and Learning Environment Policy should be followed for all exchanges between you and the instructors and/or teaching assistants and/or your classmate

Syllabus for COMP 1012: Programming for Engineers and Scientists

COURSE SCHEDULE WITH ASSESSMENTS

This schedule is subject to change at the discretion of the instructor and/or based on the learning needs of the students but such changes are subject to Section 2.8 of ROASS. More details on UMLearn.

Week	Dates (Mon-Fri)	Topics (tentative)	Quiz (in class)	Test (in class)	Labs (attend your section)	UMLearn Practice Question Sets (QS) (Completed by Sat)	Assignment Released (Fri at 12:00 pm)	Assignment Due (Fri 11:59 pm)
0	Jan 9 – 13	Introduction/Basics	-	-	-		-	-
1	Jan 16 – 20	Basics	1		Lab 1	QS1		
2	Jan 23 – 27	Strings, I/O	2		Lab 2	QS2		
3	Jan 30 – Feb 3	Conditionals (if), while	3		Lab 3	QS3	1	
4	Feb 6 – 10	for, files, lists	4		Lab 4	QS4		
5	Feb 13 – 17	list, set, dict, tuple	5	1: (Weeks 0 – 4)*	Lab 5	QS5	2	1
	Feb 20 – 24	**Reading Week**	**NO CLASSES**	**NO CLASSES**	**NO LABS**	-	-	-
6	Feb 27 – Mar 3	Functions	6		Lab 6	QS6		
7	Mar 6 – 10	Random numbers	7		Lab 7	QS7	3	2
8	Mar 13 – 17	Arrays	8		Lab 8	QS8		
9	Mar 20 – 24	Multi-dimensional arrays	9		Lab 9	QS9	4	3
10	Mar 27 – 31	Objects	10	2: (Weeks 7 – 9)*	Lab 10	QS10		
11	Apr 3 – 7	Recursion	11		Lab 11 **	QS11		4
12	Apr 10 – 12	Review, optional additional material	-		Lab 12 (Optional) ***	-		
Exam Period	Apr 14 – 28	-	-	-	-		-	-

- *** Term Tests occur on the first class timeslot of the week at the start of class. They are 45 minutes long and the TR slot will have class after the test**
- ****Please note that the week of Good Friday (April 3 – 7) the lab is MUST be submitted online OR you may attend a Wednesday or Thursday section, either is acceptable. If you attend lab, you do not need to submit online.**
- *****There are NO labs the last week of classes, Lab 12 is optional. No marks will be taken as there is no class on Friday. Feel free to work on it on your own time.**

1

Syllabus for COMP 1012: Programming for Engineers and Scientists

ASSESSMENT AND GRADES

Graded Components

Below is an overview of the graded components, and their weight. Details of each component are listed in GRADED COURSE COMPONENTS

Type of Assessment	Weight
Assignments	16%
Labs (lowest is dropped)	10%
Term Tests	24%
Final Exam	30%
Quizzes (lowest is dropped)	10%
In-class iClickers	5%
UMLearn Practice Questions	5%

Letter Grade Cut-Points

Below is a breakdown of grades and resulting letter grades. Depending on final grade distribution, the instructor may adjust this table. Grade adjustments will generally be limited to moving the cut-points .5%, in either direction.

Letter Grade	Percentage out of 100	Grade Point Range	Final Grade Point
A+	90-100	4.25-4.5	4.5
A	80-89.99	3.75-4.24	4.0
B+	75-79.99	3.25-3.74	3.5
B	70-74.99	2.75-3.24	3.0
C+	65-69.99	2.25-2.74	2.5
C	60-64.99	2.0-2.24	2.0
D	50-59.99	Less than 2.0	1.0
F	Less than 50		0

Voluntary Withdrawal

Please note these important dates and details:

- **Jan 20, 2023: This is the last day to drop the class and receive 100% refund.**
- **Jan 23, 2023: This is the last day to ADD the class**
- **March 22, 2023: This is the last day to withdraw (VW) from the course (with no refund).**
- Students who do not formally withdraw from the class before **March 22, 2023**, through the Registrar's Office will be assigned a final grade.
- Withdrawn courses will be recorded on your official transcript.
- Please refer to the Registrar's Office web page for more information.
- You are welcome to discuss your grade and course progress with me if you are considering dropping the course.

Syllabus for COMP 1012: Programming for Engineers and Scientists

GRADED COURSE COMPONENTS

Quizzes (10%)

- Each week there will be a 5-minute in-class quiz on the previous weeks' material. See the schedule above for the outline of that.
- Quizzes can include writing, coding, short answer, fill in the blank, multiple select or matching.
- Quiz questions will be based on in-class activities, lectures, videos, and required resources.
- There are no make-up quizzes. If you miss a quiz, you receive a 0
- Your lowest quiz mark will be dropped
- Quizzes will always take place in the first class of a given week (Monday or Tuesday depending on your class)
- If you show up late, you may not be able to write the quiz, no additional time will be given
- Quizzes are written **IN** UMLearn. You must bring a laptop with you to class (or tablet but a laptop is probably better for most activities)
- Quizzes are individual. Cheating is strictly prohibited, and violations pursued rigorously
- These quizzes are here to help you engage with the material. They are not meant to be scary, in fact, quite the opposite. Weekly quizzes help us stay on top of the material and relieve test anxiety when it comes to the term tests and final. Please use them as an opportunity to learn and practice, not freak out ☺

Labs (10%)

- Labs are completed on a weekly basis, starting with the second week of classes
- Your lowest lab mark will be dropped
- If you know you need to miss a lab, please reach out first and I can have you attend a different section, so you don't lose the marks
- All labs must be completed in a lab slot and shown your lab instructor to receive marks
- Labs are out of 2. One mark for the work and one mark for your coding standards
    - Part marks may be given out for either criterion as appropriate
- Labs are collaborative, and more fun if they are. Talk with a friend or peer, work together to learn something new.
- Coming to lab and copying a friend's answers will not teach you anything. Do not take the concept of collaboration as an easy way out. Studies shows learnings together is more effective than individually and if you coast here, individual assessments will be much more challenging.

Assignments (16%)

Individually, students will be required to complete 4 assignments throughout the term.

- Assignment release and due dates are listed above in the schedule
- No extensions will be given without extenuating circumstances and proof
- You must submit a declaration of honesty in UMLearn under "Honesty Declarations" for **EACH** assignment
- Assignments are to be handed in on UMLearn
- A0 will be available for practice using UMLearn, not knowing how to hand in an assignment is not a valid excuse for late hand ins.

Late Assignment Policy

Assignments will be accepted up to 49 hours after the due date and time stated on the assignment, at a penalty of 2% of the mark per hour or portion thereof. (For an assignment that is from 1 second late, to 59 minutes and 59 seconds late, you will lose 2% of its value.) After 49 hours and 1 second, the assignment loses 100% and can no longer be handed in at all. Exceptions to this policy will be made only

in exceptional circumstances, such as documented medical or family issues. Computer or network problems are *not* exceptional circumstances. Errors made by a student in the submission process (e.g., handing in incorrect files) are *not* exceptional circumstances. You are strongly advised to submit assignments early.

Assignment Extension Policy

The following policy will apply to COMP 1012 in Fall 2023 in recognition of the fact that there are circumstances other than illness that can interfere with your coursework.

You will be allowed to request a 2-day extension on **ONE** assignment this term. You do not need to supply a specific reason. Simply email your instructor and request that your one-time extension be applied to the assessment of your choice. **You can only use this extension once, so choose wisely! You must make this request at least 12 hours in advance of the posted deadline for the assignment.** This policy does not impose any limits on extensions or other accommodations required for medical reasons. All SAS accommodations remain unchanged.

Term Tests (24%)

- There will be 2 term tests. The dates can be found in the schedule above
- The topics they will cover are in the schedule above but may change slightly. It is dependent on how far in we get in class as well.
- Each term test will be 45 minutes in length. If your class time allows for it, there will be a class after the test. **SAS students must plan at least 1 week in advance and must start writing earlier so that class may continue after.**
- These tests will be made up of multiple-choice, short answers, coding questions, drawings, etc. It's all fair game but as we get closer, we will discuss it further.
- There are no make-up tests
- Missed Midterm with valid reason (see attendance policy) will be presented with an alternative. With no valid reason, you will receive a 0.
- Grade Appeals: Midterm grades may be appealed *within 7 days of the date grades were returned using a UMLearn form.* Appeals using email or outside of the 7-day window will usually not be considered.

Final Examination (30%)

- The final examination will take place outside of class time
- It is scheduled by the University so once I know when it is, I will announce it
- Any conflicts with the date need to be brought up to the department or registrar's office immediately so you may take proper steps
- **You must achieve a grade of 40% or higher on the final to pass this course**.

iClickers (5%)

- iClickers will be done in most, if not all classes
- You must participate in at least 80% of the iClicker questions over the course of the term to receive the full percentage for this category. This allows the room for missed classes without a penalty to your grade
- You are more than welcome to join iClickers from home if you are staying home due to illness
- Participation is NOT about getting the right answer, being the fastest, or the loudest. It's about showing up and trying to do your best. Engaging with your peers, your instructor, and the content. Show up and put in the effort and this is the easiest 5% you will ever get

Syllabus for COMP 1012: Programming for Engineers and Scientists

UMLearn Assessments (5%)

- We use UMLearn for additional practice questions.
- Under each week you will find a link to a UMLearn practice assessment
- Practice questions must be completed by the Saturday of each week to receive marks for them.
- This are not marked for correctness; it's not about getting it right on the first try.
- You can attempt the questions as many times as you want as long as they are completed by Saturday at 11:59 pm.
- Disregard the mark total, UMLearn enforces a mark allocation however they do not matter. As long as the questions is attempted you will receive marks for it.
- If you attempt 50% of these practice questions, you receive 2.5/5% of the marks for this grade item
- **As long as you attempt at least 80% of these assessments (by their individual due dates) you will receive the entire 5% of this grade item however it is recommended you attempt them all, even after the due date for practice.**

Bibliography

[1] S. Suh, C. Latulipe, K. J. Lee, B. Cheng, and E. Law, "Using comics to introduce and reinforce programming concepts in cs1," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '21, Virtual Event, USA: Association for Computing Machinery, 2021, pp. 369–375, ISBN: 9781450380621. DOI: 10.1145/3408877.3432465. [Online]. Available: https://doi.org/10.1145/3408877.3432465.

[2] C. Lewis, "Physical java memory models," *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, SIGCSE '21, pp. 383–389, 2021. DOI: 10.1145/3408877.3432477. [Online]. Available: https://doi.org/10.1145/3408877.3432477.

[3] S. Fincher *et al.*, "Notional machines in computing education," *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '20, pp. 21–50, 2020. DOI: 10.1145/3437800.3439202. [Online]. Available: https://doi.org/10.1145/3437800.3439202.

[4] D. A. Norman, *The Design of Everyday Things*. USA: Basic Books, Inc., 2002, ISBN: 9780465067107.

[5] B. du Boulay, "Some difficulties of learning to program," *Journal of Educational Computing Research*, vol. 2, no. 1, pp. 57–73, 1986. DOI: 10.2190/3lfx-9rrf-67t8-uvk9.

[6] R. Lister, C. Fidge, and D. Teague, "Further evidence of a relationship between explaining, tracing and writing skills in introductory programming," *ACM SIGCSE Bulletin*, vol. 41, no. 3, pp. 161–165, Jul. 2009, ISSN: 0097-8418. DOI: 10.1145/1595496.1562930. [Online]. Available: https://doi.org/10.1145/1595496.1562930.

[7] A. Salguero, W. G. Griswold, C. Alvarado, and L. Porter, "Understanding sources of student struggle in early computer science courses," in *Proceedings of the 17th ACM Conference on International Computing Education Research*, ser. ICER 2021, Virtual Event, USA: Association for Computing Machinery, 2021, pp. 319–333, ISBN: 9781450383264. DOI: 10.1145/3446871.3469755. [Online]. Available: https://doi.org/10.1145/3446871.3469755.

[8] P. E. Dickson, N. C. Brown, and B. A. Becker, "Engage against the machine: Rise of the notional machines as effective pedagogical devices," *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '20, pp. 159–165, 2020. DOI: 10.1145/3341525.3387404. [Online]. Available: https://doi.org/10.1145/3341525.3387404.

[9] I. Milne and G. Rowe, "Difficulties in learning and teaching programming–views of students and tutors," *Education and Information Technologies*, vol. 7, no. 1, pp. 55–66, 2002. DOI: 10.1023/a:1015362608943.

[10] J. Hidalgo-Cespedes, G. Marian-Raventos, and V. Lara-Villagran, "Understanding notional machines through traditional teaching with conceptual contraposition and program memory tracing," en, *CLEI Electronic Journal*, vol. 19, p. 3, Aug. 2016, ISSN: 0717-5000. [Online]. Available: http://www.scielo.edu.uy/scielo.php?script=sci%5C_arttext%5C&pid=S0717-50002016000200003%5C&nrm=iso.

[11] V. Ramalingam, D. LaBelle, and S. Wiedenbeck, "Self-efficacy and mental models in learning to program," in *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education - ITiCSE '04*, New York, New York, USA: ACM Press, 2004, p. 171, ISBN: 1581138369. DOI: 10.1145/1007996.1008042.

[12] E. Doyle, I. Stamouli, and M. Huggard, "Computer anxiety, self-efficacy, computer experience: An investigation throughout a computer science degree," in *Proceedings Frontiers in Education 35th Annual Conference*, Indianopolis, IN, USA: IEEE Proceedings Frontiers in Education 35th Annual Conference, 2005, S2H–3. DOI: 10.1109/FIE.2005.1612246.

[13] B. C. Wilson and S. Shrock, "Contributing to success in an introductory computer science course: A study of twelve factors," in *Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '01, Charlotte, North Carolina, USA: Association for Computing Machinery, 2001, pp. 184–188, ISBN: 1581133294. DOI: 10.1145/364447.364581. [Online]. Available: https://doi.org/10.1145/364447.364581.

[14] C. C. Wolverton, B. N. G. Hollier, and P. A. Lanier, "The impact of computer self efficacy on student engagement and group satisfaction in online business courses," *Electronic Journal of e-Learning*, vol. 18, no. 2, p. 14, Feb. 2020. DOI: 10.34190/ejel.20.18.2.006. [Online]. Available: https://doi.org/10.34190/ejel.20.18.2.006.

[15] M. Guzdial, "What's the best way to teach computer science to beginners?" *Communications of the ACM*, vol. 58, no. 2, pp. 12–13, 2015. DOI: 10.1145/2714488.

[16] N. Bubica and I. Boljat, "Programming novices' mental models," in *7th International Conference on Education and New Learning Technologie*, Barcelona: International Academy of Technology, Education and Development, Jul. 2015, p. 9. DOI: 10.13140/RG.2.1.3773.2960.

[17] M. B. J.
bibinitperiod M. T. Bonde. "Labster," Labster. (2011), [Online]. Available: http://labster.com.

[18] P. Guo. "Pythontutor," PythonTutor. (2013), [Online]. Available: http://pythontutor.com.

[19] J. P. [source]. "Algorithm visualizer." (2022), [Online]. Available: https://algorithm-visualizer.org/.

[20] W. Pillow, "Confession, catharsis, or cure? rethinking the uses of reflexivity as methodological power in qualitative research," *International Journal of Qualitative Studies in Education*, vol. 16, no. 2, pp. 175–196, 2003. DOI: 10.1080/0951839032000060635. eprint: https://doi.org/10.1080/0951839032000060635. [Online]. Available: https://doi.org/10.1080/0951839032000060635.

[21] S. Suh, "Using concreteness fading to model & design learning process," *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ICER '19, pp. 353–354, 2019. DOI: 10.1145/3291279.3339445. [Online]. Available: https://doi.org/10.1145/3291279.3339445.

[22] B. S. "Bloom, *"The taxonomy of educational objectives: Handbook 1"*. "Harlow, England": "Longman Higher Education", Nov. 1965.

[23] S. Masapanta-Carrión and J. Á. Velázquez-Iturbide, "A systematic review of the use of bloom's taxonomy in computer science education," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18, Baltimore, Maryland, USA: Association for Computing Machinery, 2018, pp. 441–446, ISBN: 9781450351034. DOI: 10.1145/3159450.3159491. [Online]. Available: https://doi.org/10.1145/3159450.3159491.

[24] D. V and Y. A, "Constructivism: A paradigm for teaching and learning," *Arts and Social Sciences Journal*, vol. 7, no. 4, 2016. DOI: 10.4172/2151-6200.1000200.

[25] D. A. Norman and J. C. Spohrer, "Learner-centered education," *Communications of the ACM*, vol. 39, no. 4, pp. 24–27, Apr. 1996. DOI: 10.1145/227210.227215. [Online]. Available: https://doi.org/10.1145/227210.227215.

[26] S. Papert and I. Harel, "Situating constructionism," in *Constructionism*, New York: Ablex Publishing Corporation, 1991, ch. 1, p. 17.

[27] E. Aivaloglou, S. Baloro, G. Boukeas, and C. B. et al., "Learning through making," in *The Big Book of Computing Pedagogy*, Hello World Magazine, 2021.

[28] L. S. Vygotskij, R. W. Rieber, and N. Minick, *The history of the development of higher mental functions*. Plenum Press, 1997.

[29] J. Stockard, T. W. Wood, C. Coughlin, and C. R. Khoury, "The effectiveness of direct instruction curricula: A meta-analysis of a half century of research," *Review of Educational Research*, vol. 88, no. 4, pp. 479–507, Jan. 2018. DOI: 10.3102/0034654317751919. [Online]. Available: https://doi.org/10.3102/0034654317751919.

[30] K. Shabani, M. Khatib, and S. Ebadi, "Vygotsky's zone of proximal development: Instructional implications and teachers' professional development," *English Language Teaching*, vol. 3, no. 4, pp. 237–248, 2010. DOI: 10.5539/elt.v3n4p237.

[31] N. Balaban, "Seeing the child, knowing the person," in *To Become a Teacher*, W. Ayers, Ed. Teachers College Press, 1995, p. 52.

[32] N. Anderson and T. Gegg-Harrison, "Learning computer science in the "comfort zone of proximal development"," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '13, Denver, Colorado, USA: Association for Computing Machinery, 2013, pp. 495–500, ISBN: 9781450318686. DOI: 10.1145/2445196.2445344. [Online]. Available: https://doi.org/10.1145/2445196.2445344.

[33] A. Trory, K. Howland, and J. Good, "Designing for concreteness fading in primary computing," *Proceedings of the 17th ACM Conference on Interaction Design and Children*, IDC '18, pp. 278–288, 2018. DOI: 10.1145/3202185.3202748. [Online]. Available: https://doi.org/10.1145/3202185.3202748.

[34] J. Bennedsen and C. Schulte, "A competence model for object interaction in introductory programming," *Proceedings of the 18th Workshop of the Psychology of Programming Interest Group*, Feb. 2006.

[35] J. Sorva, "Notional machines and introductory programming education," *ACM Transactions on Computing Education*, vol. 13, no. 2, pp. 1–31, 2013. DOI: 10.1145/2483710.2483713.

[36] A. N. Kumar, "Fork diagrams for teaching selection in cs i," in *Proceedings of the Twenty-Seventh SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '96, Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 348–352, ISBN: 089791757X. DOI: 10.1145/236452.236575. [Online]. Available: https://doi.org/10.1145/236452.236575.

[37] P. E. Dickson and T. Dragon, "A memory diagram for all seasons," in *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, ser. ITiCSE '21, Virtual Event, Germany: Association for Computing Machinery, 2021, pp. 150–156, ISBN: 9781450382144. DOI: 10.1145/3430665.3456317. [Online]. Available: https://doi.org/10.1145/3430665.3456317.

[38] E. R. Fyfe and M. J. Nathan, "Making "concreteness fading" more concrete as a theory of instruction for promoting transfer," *Educational Review*, vol. 71, no. 4, pp. 403–422, Jul. 2019, ISSN: 0013-1911. DOI: 10.1080/00131911.2018.1424116.

[39] S. F. Mazumder, C. Latulipe, and M. A. Pérez-Quiñones, "Are variable, array and object diagrams in java textbooks explanative?" In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '20, Trondheim, Norway: Association for Computing Machinery, 2020, pp. 425–431, ISBN: 9781450368742. DOI: 10.1145/3341525.3387368. [Online]. Available: https://doi.org/10.1145/3341525.3387368.

[40] T. L. Naps *et al.*, "Exploring the role of visualization and engagement in computer science education," in *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, ser. ITiCSE-WGR '02, Aarhus, Denmark: Association for Computing Machinery, 2002, pp. 131–152, ISBN: 9781450374491. DOI: 10.1145/960568.782998. [Online]. Available: https://doi.org/10.1145/960568.782998.

[41] D. Hayatpur, D. Wigdor, and H. Xia, "Crosscode: Multi-level visualization of program execution," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23, Hamburg, Germany: Association for Computing Machinery, 2023, ISBN: 9781450394215. DOI: 10.1145/3544548.3581390. [Online]. Available: https://doi.org/10.1145/3544548.3581390.

[42] N. U. of Singapore. "Visualgo." (2011), [Online]. Available: https://visualgo.net/en.

[43] v. Open source. "Algorithm visualizer." (2019), [Online]. Available: https://algorithm-visualizer.org/.

[44] M. Kölling, B. Quig, A. Patterson, and J. Rosenberg, "The bluej system and its pedagogy," *Computer Science Education*, vol. 13, no. 4, pp. 249–268, 2003. DOI: 10.1076/csed.13.4.249.17496. eprint: https://doi.org/10.1076/csed.13.4.249.17496. [Online]. Available: https://doi.org/10.1076/csed.13.4.249.17496.

[45] P. Gries, V. Mnih, J. Taylor, G. Wilson, and L. Zamparo, "Memview: A pedagogically-motivated visual debugger," in *Proceedings Frontiers in Education 35th Annual Conference*, 2005, S1J–11. DOI: 10.1109/FIE.2005.1612204.

[46] C. D. Hundhausen and J. L. Brown, "What you see is what you code: A "live" algorithm development and visualization environment for novice learners," *J. Vis. Lang. Comput.*, vol. 18, no. 1, pp. 22–47, Feb. 2007, ISSN: 1045-926X. DOI: 10.1016/j.jvlc.2006.03.002. [Online]. Available: https://doi.org/10.1016/j.jvlc.2006.03.002.

[47] P. Guo, "Ten million users and ten years later: Python tutor's design guidelines for building scalable and sustainable research software in academia," in *The 34th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '21, Virtual Event, USA: Association for Computing Machinery, 2021, pp. 1235–1251, ISBN: 9781450386357. DOI: 10.1145/3472749.3474819. [Online]. Available: https://doi.org/10.1145/3472749.3474819.

[48] S. Abdelhamid, T. Mallari, and T. Stower, "Cyed: A cyberinfrastructure for computer education," in *Innovative Approaches to Technology-Enhanced Learning for the Workplace and Higher Education*, D. Guralnick, M. E. Auer, and A. Poce, Eds., Cham: Springer International Publishing, 2023, pp. 3–14.

[49] B. Harrington and N. Cheng, "Tracing vs. writing code: Beyond the learning hierarchy," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18, Baltimore, Maryland, USA: Association for Computing Machinery, 2018, pp. 423–428, ISBN: 9781450351034. DOI: 10.1145/3159450.3159530. [Online]. Available: https://doi.org/10.1145/3159450.3159530.

[50] M. Hassan and C. Zilles, "On students' usage of tracing for understanding code," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, ser. SIGCSE 2023, Toronto ON, Canada: Association for Computing Machinery, 2023, pp. 129–136, ISBN: 9781450394314. DOI: 10.1145/3545945.3569741. [Online]. Available: https://doi.org/10.1145/3545945.3569741.

[51] M. Lopez, J. Whalley, P. Robbins, and R. Lister, "Relationships between reading, tracing and writing skills in introductory programming," in *Proceedings of the Fourth International Workshop on Computing Education Research*, ser. ICER '08, Sydney, Australia: Association for Computing Machinery, 2008, pp. 101–112, ISBN: 9781605582160. DOI: 10.1145/1404520.1404531. [Online]. Available: https://doi.org/10.1145/1404520.1404531.

[52] A. M. Gaweda, C. F. Lynch, N. Seamon, G. Silva de Oliveira, and A. Deliwa, "Typing exercises as interactive worked examples for deliberate practice in cs courses," in *Proceedings of the Twenty-Second Australasian Computing Education Conference*, ser. ACE'20, Melbourne, VIC, Australia: Association for Computing Machinery, 2020, pp. 105–113, ISBN: 9781450376860. DOI: 10.1145/3373165.3373177. [Online]. Available: https://doi.org/10.1145/3373165.3373177.

[53] A. Leinonen, H. Nygren, N. Pirttinen, A. Hellas, and J. Leinonen, "Exploring the applicability of simple syntax writing practice for learning programming," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '19, Minneapolis, MN, USA: Association for Computing Machinery, 2019, pp. 84–90, ISBN: 9781450358903. DOI: 10.1145/3287324.3287378. [Online]. Available: https://doi.org/10.1145/3287324.3287378.

[54] Y. Qian and J. Lehman, "Students' misconceptions and other difficulties in introductory programming," *ACM Transactions on Computing Education*, vol. 18, no. 1, pp. 1–24, 2017. DOI: 10.1145/3077618.

[55] R. Lister, "Objectives and objective assessment in cs1," *SIGCSE Bull.*, vol. 33, no. 1, pp. 292–296, Feb. 2001, ISSN: 0097-8418. DOI: 10.1145/366413.364605. [Online]. Available: https://doi.org/10.1145/366413.364605.

[56] A. N. Kumar, "Solving code-tracing problems and its effect on code-writing skills pertaining to program semantics," in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '15, Vilnius, Lithuania: Association for Computing Machinery, 2015, pp. 314–319, ISBN: 9781450334402. DOI: 10.1145/2729094.2742587. [Online]. Available: https://doi.org/10.1145/2729094.2742587.

[57] A. Gupta and R. Rybarczyk, "Improving long term performance using visualized scope tracing: A 10-year study," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, ser. SIGCSE 2023, Toronto ON, Canada: Association for Computing Machinery, 2023, pp. 137–143, ISBN: 9781450394314. DOI: 10.1145/3545945.3569748. [Online]. Available: https://doi.org/10.1145/3545945.3569748.

[58] T. Jenkins, "On the difficulty of learning to program," in *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, Citeseer, vol. 4, Loughborough, UK: LTSN Centre for Information and Computer Sciences, 2002, pp. 53–58.

[59] P. N. Johnson-Laird, *Mental models and consistency*. The Guilford Press, 2012, pp. 225–244, ISBN: 9781920682590.

[60] R. Bornat, S. Dehnadi, and Simon, "Mental models, consistency and programming aptitude," in *Proceedings of the Tenth Conference on Australasian Computing Education - Volume 78*, ser. ACE '08, Wollongong, NSW, Australia: Australian Computer Society, Inc., 2008, pp. 53–61, ISBN: 9781920682590.

[61] T. Ngo, J. Kunkel, and J. Ziegler, "Exploring mental models for transparent and controllable recommender systems: A qualitative study," in *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, ser. UMAP '20, Genoa, Italy: Association for Computing Machinery, 2020, pp. 183–191, ISBN: 9781450368612. DOI: 10.1145/3340631.3394841. [Online]. Available: https://doi.org/10.1145/3340631.3394841.

[62] A. Horvath, M. Nagy, F. Voichick, M. B. Kery, and B. A. Myers, "Methods for investigating mental models for learners of apis," in *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '19, Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–6, ISBN: 9781450359719. DOI: 10.1145/3290607.3312897. [Online]. Available: https://doi.org/10.1145/3290607.3312897.

[63] S. F. Mazumder and M. A. Pérez-Quiñones, "Eliciting a novice programmer's mental model of arrays," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '21, Virtual Event, USA: Association for Computing Machinery, 2021, p. 1302, ISBN: 9781450380621. DOI: 10.1145/3408877.3439647. [Online]. Available: https://doi.org/10.1145/3408877.3439647.

[64] C. Boston, "The concept of formative assessment," *Practical Assessment, Research, and Evaluation*, vol. 8, no. 9, p. 5, 2002. DOI: 10.7275/kmcq-dj31.

[65] S. Grover, "Toward a framework for formative assessment of conceptual learning in k-12 computer science classrooms," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '21, Virtual Event, USA: Association for Computing Machinery, 2021, pp. 31–37, ISBN: 9781450380621. DOI: 10.1145/3408877.3432460. [Online]. Available: https://doi.org/10.1145/3408877.3432460.

[66] J. Hudesman, S. Crosby, B. Flugman, S. Issac, H. Everson, and D. B. Clay, *Using formative assessment and metacognition to improve student achievement.* Nov. 2012. [Online]. Available: https://eric.ed.gov/?id=EJ1067283.

[67] S. W. Dietrich *et al.*, "Using formative assessment for improving pedagogy: Reflections on feedback informing database visualizations," *ACM Inroads*, vol. 11, no. 4, pp. 27–34, Nov. 2020, ISSN: 2153-2184. DOI: 10.1145/3430766. [Online]. Available: https://doi.org/10.1145/3430766.

[68] R. Morris, T. Perry, and L. Wardle, "Formative assessment and feedback for learning in higher education: A systematic review," *Review of Education*, vol. 9, no. 3, Aug. 2021. DOI: 10.1002/rev3.3292. [Online]. Available: https://doi.org/10.1002/rev3.3292.

[69] C. Wong, P. Denny, A. Luxton-Reilly, and J. Whalley, "The impact of multiple choice question design on predictions of performance," in *Australasian Computing Education Conference*, ser. ACE '21, Virtual, SA, Australia: Association for Computing Machinery, 2021, pp. 66–72, ISBN: 9781450389761. DOI: 10.1145/3441636.3442306. [Online]. Available: https://doi.org/10.1145/3441636.3442306.

[70] K. Cunningham, S. Blanchard, B. Ericson, and M. Guzdial, "Using tracing and sketching to solve programming problems: Replicating and extending an analysis of what students draw," in *Proceedings of the 2017 ACM Conference on International Computing Education Research*, ser. ICER '17, Tacoma, Washington, USA: Association for Computing Machinery, 2017, pp. 164–172, ISBN: 9781450349680. DOI: 10.1145/3105726.3106190. [Online]. Available: https://doi.org/10.1145/3105726.3106190.

[71] T. S. Roberts, "The use of multiple choice tests for formative and summative assessment," in *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*, ser. ACE '06, Hobart, Australia: Australian Computer Society, Inc., 2006, pp. 175–180, ISBN: 1920682341.

[72] C. R. Association, *Generation cs*, Apr. 2017. [Online]. Available: https://cra.org/data/generation-cs/.

[73] A. Aggarwal, N. Puthanveetil, and C. Gardner-Mccune, "Who attempts optional practice problems in a cs1 course? exploring learner agency to foster mastery learning," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, ser. SIGCSE 2023, Toronto ON, Canada: Association for Computing Machinery, 2023, pp. 1055–1061, ISBN: 9781450394314. DOI: 10.1145/3545945.3569854. [Online]. Available: https://doi.org/10.1145/3545945.3569854.

[74] ACM and CCECC. [Online]. Available: https://ccecc.acm.org/files/publications/Blooms-for-Computing-Verb-Table-for-printing.pdf.

[75] L. W. Anderson, D. R. Krathwohl, B. S. Bloom, and B. S. Bloom. New York, NY, USA: Longman, 2001, p. 333.

[76] S. Woessner and N. McNulty, *How the best teachers use bloom's taxonomy in their digital classrooms*, Nov. 2021. [Online]. Available: https://www.niallmcnulty.com/2017/11/blooms-digital-taxonomy/.

[77] U. Fuller *et al.*, "Developing a computer science-specific learning taxonomy," *SIGCSE Bull.*, vol. 39, no. 4, pp. 152–170, Dec. 2007, ISSN: 0097-8418. DOI: 10.1145/1345375.1345438. [Online]. Available: https://doi.org/10.1145/1345375.1345438.

[78] A. Rhodes, K. Bower, and P. Bancroft, "Managing large class assessment," in *Proceedings of the Sixth Australasian Conference on Computing Education - Volume 30*, ser. ACE '04, Dunedin, New Zealand: Australian Computer Society, Inc., 2004, pp. 285–289.

[79] *Brightspace*, Apr. 2023. [Online]. Available: https://www.d2l.com/brightspace/.

[80] (), [Online]. Available: https://runestone.academy/.

[81] L. P. Cousins. "Recognition vs recall," PsychCentral. (2010).

[82] A. Edgcomb, F. Vahid, R. Lysecky, and S. Lysecky, "Getting students to earnestly do reading, studying, and homework in an introductory programming class," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '17, Seattle, Washington, USA: Association for Computing Machinery, 2017, pp. 171–176, ISBN: 9781450346986. DOI: 10.1145/3017680.3017732. [Online]. Available: https://doi.org/10.1145/3017680.3017732.

[83] D. H. Smith, C. Emeka, M. Fowler, M. West, and C. Zilles, "Investigating the effects of testing frequency on programming performance and students' behavior," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, ser. SIGCSE 2023, Toronto ON, Canada: Association for Computing Machinery, 2023, pp. 757–763, ISBN: 9781450394314. DOI: 10.1145/3545945.3569821. [Online]. Available: https://doi.org/10.1145/3545945.3569821.

[84] J. J. McConnell, "Active learning and its use in computer science," *SIGCUE Outlook*, vol. 24, no. 1–3, pp. 52–54, Jan. 1996, ISSN: 0163-5735. DOI: 10.1145/1013718.237526. [Online]. Available: https://doi.org/10.1145/1013718.237526.

[85] A. Thi Van Pham and N. Thi Thao Ho, "Perceptions of lecturers and students on challenges of the flipped classroom," in *2021 3rd International Conference on Modern Educational Technology*, ser. ICMET 2021, Jakarta, Indonesia: Association for Computing Machinery, 2021, pp. 168–173, ISBN: 9781450390354. DOI: 10.1145/3468978.3469006. [Online]. Available: https://doi.org/10.1145/3468978.3469006.

[86] M. J. Watkins and A. K. Todres, "On the relation between recall and recognition," *Journal of Verbal Learning and Verbal Behavior*, vol. 17, no. 5, pp. 621–633, 1978, ISSN: 0022-5371. DOI: https://doi.org/10.1016/S0022-5371(78)90381-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002253717890381X.

[87] D. R. Krathwohl, "A revision of bloom's taxonomy: An overview," *Theory Into Practice*, vol. 41, no. 4, pp. 212–218, 2002. DOI: 10.1207/s15430421tip4104_2. eprint: https://doi.org/10.1207/s15430421tip4104_2. [Online]. Available: https://doi.org/10.1207/s15430421tip4104_2.

[88] B. Bouchey, J. Castek, and J. Thygeson, "Multimodal learning," in *Innovative Learning Environments in STEM Higher Education: Opportunities, Challenges, and Looking Forward*, J. Ryoo and K. Winkelmann, Eds. Cham: Springer International Publishing, 2021, pp. 35–54, ISBN: 978-3-030-58948-6. DOI: 10.1007/978-3-030-58948-6_3. [Online]. Available: https://doi.org/10.1007/978-3-030-58948-6_3.

[89] *Introducing chatgpt*. [Online]. Available: https://openai.com/blog/chatgpt.

[90] S. N. YILDIRIM-ERBASLI and O. BULUT, *Conversation-based assessments: Real-time assessment and feedback*. [Online]. Available: https://elearnmag.acm.org/archive.cfm?aid=3495533.

List of Publications

[1] L. Himbeault, "Diagrammatic question framework: Studying effectiveness in first-year computing courses," in *Proceedings of the SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '23, Toronto, Ontario, Canada: Association for Computing Machinery, 2023. DOI: 10.1145/3545947.3573278. [Online]. Available: https://doi.org/10.1145/3545947.3573278.

[2] C. Latulipe and L. Himbeault, "Using a two-stage final exam in an intro cs course: Student perceptions and grade impacts," in *Proceedings of the Western Canadian Conference on Computing Education Proceedings*, ser. WCCCE '23, Vancouver, British Columbia, Canada: Association for Computing Machinery, 2023. DOI: 10.1145/3593342.3593355. [Online]. Available: https://www.laurenhimbeault.com/published-works/wccce-23.