# An exploration of the instruction of Middle School Computer Science

# Table of Contents

# Chapter 1: Purpose and Rationale

CS education is currently very theoretical, and most students who complete a highschool CS course are not well equipped to apply their knowledge in writing efficient and well thought out programs. The goal of this project is to develop a curriculum that would act as an introductory CS class offering students an explanation on how to write code, but also explain what the code does and how it works on a fundamental level. The goal is to explore the feasibility of a middle school level course (Grades 6-8) that would expose students to various aspects of computing and help build an intuitive understanding of programming that will allow them to leverage computers to help aid in other subjects.

## Hypothesis

It is feasible to create a computer science curriculum that can expand on student intuition and understanding in mathematics to help students “invent” the study of computer science for themselves.

# Chapter 2: Capstone Literature Review

Computer accessibility has dramatically increased in the past two decades, however, there has been increasing inequality in Computer Science education. While some schools are able to kindle a passion in their students for the subject, very few have been able to teach the subject effectively to a diverse group of students, especially from lower socioeconomic backgrounds.[1] Some of the largest barriers to entry into Computer Science are a general lack of understanding of the importance of the field, an incomplete approach to teaching programming, and limited access to a diverse set of teaching styles for the subject. For the purposes of this study, information on CS education for english speaking students from ages 10-14 was examined as this age tends to be the point where students begin to have the appropriate mathematical tools necessary to begin writing code, transitioning away from block based programming.

**The Importance of Computer Science:**

Whilst those inside the field of computer science often see its applications all around them, for those outside the field, computers can seem like magic boxes which allow people to use the internet, make spreadsheets and write essays. To most people, learning Computer Science can seem daunting, especially since most people associate it with the fancy spreadsheet and web applications that they see in their day to day lives. However, that could not be further

[1] Guzdial, Mark. “3 Proposals to Change How We Teach Computing In Order to Reduce Inequality.” ACM, August 2, 2020. https://cacm.acm.org/blogs/blog-cacm/246550-3-proposals-to-change-how-we-teach-computing-in-order-to-reduce-inequality/fulltext.

from the truth. The goal of the computer science class is really to teach students the tools to become problem solvers. Instead of being stuck using somebody else's solution for a slightly different problem, the goal of the CS class is to equip students with understanding of programming so that they can easily pick up the syntax of the language they intend to use.

One of the largest issues is when parents are unaware of what the goals of a CS course are. In some cases parents see computer science courses as a course on how to use a computer and therefore tend not to encourage their children to participate as younger generations become more and more proficient in using electronic devices[2]. As such, a large part of a CS program is marketing in the community, helping to educate people about the field of CS. Additionally, the role of a CS program is to help make CS accessible. When it comes to the magic box that computers are, it is often easy to overlook the similarities between block based coding and written code[3]. Especially in a high stakes academic environment, where a poor grade in CS can hurt a student's future opportunities, computer science understanding becomes neglected due to a lack of exposure. Students in the 5-8th grade level benefit the most from CS education since they are able to explore the subject in parallel with the mathematical knowledge acquired at this time, whilst also benefiting from the relative unimportance of grades at this age range.

---

[2] Arnaud. "Six Reasons Why Computer Science Education Is Failing Students." LiveCode, October 10, 2012. https://livecode.com/six-reasons-why-computer-science-education-is-failing-students/.

[3] "Problems in CS Education - High School Computer Science Education." Google Sites. Accessed December 18, 2020. https://sites.google.com/site/highschoolcseducation/background/problems-in-cs-education.

**An Incomplete Curriculum:**

Current Computer Science curriculum faces many challenges unique to the field. Unlike in an english class per say, the language being used varies wildly from decade to decade with different languages gaining prominence before dropping off, and different ideas are constantly being explored. In such a dynamic subject the greatest challenge lies in making sure the curriculum is relevant, not only for the present day, but also for the years to come. As a result of the field's nascency, Computer Science faces several hurdles to becoming a widespread teachable subject. The first of which is a lack of teachers. "Often computer science teachers are qualified for either teaching or technology, but do not have training in both"[4] This is often aggravated by the high demand for computer scientists and programmers in industry, pulling tallent away from the field of computer science under the allure of good pay. In order to develop a set of teachers who are adept in computer science, it is necessary to develop a concrete set of guidelines on what teaching programming actually entails.

In addition to this, identifying a programming language to use for instruction is a difficult balance to strike. Over the last 50 years, several programming languages have come to prominence and quickly fallen from grace due to their lack of features. As such, the current landscape, though more stable, still is littered with variance, on top of which great debates rage between several programming languages, arguing over functionality and ease of use. "Teachers try to figure out how much to isolate students at the lower level from the massive systems usual to CS."[4] Whilst some languages make this task easier, and are often preferred by teachers, this

---

[4] "Problems in CS Education - High School Computer Science Education." Google Sites. Accessed December 18, 2020. https://sites.google.com/site/highschoolcseducation/background/problems-in-cs-education.

approach has come under great scrutiny as many argue that isolation often hinders conceptual understanding of the theories that bind computer science together, leading student to memorize certain commands.

**New Methodologies:**

In order to develop a robust Computer Science curriculum, it is important to build several skill sets. On the one hand is the conceptual understanding of the mathematical and scientific concepts and formulas that govern the activities that the user would like the computer to perform. On the other hand is the code itself and familiarizing students, not to any one language, but to teach them how to think about code. In his critique of the Khan Academy Programing curriculum, Bret Victor writes, "The programming environment exhibits the same ruthless abbreviation as this hypothetical cooking show. We see code on the left and a result on the right, but it's the steps *in between* which matter most."[5] The issue that exists with current coding platforms is to balance the information given so as not to overwhelm the student, whilst also attempting to provide them the tools necessary to continue programming, without building a reliance on the teaching tools.

Additionally, Victor writes, "The example above allows the programmer to follow the program's execution over time. But she's peeking through a pinhole, only seeing a single point in time at any instant. She has no visual context."[5] this example serves to illustrate his point of guiding students through the programming process. At the same time, it is important to make sure that students do not build an over reliance on teaching tools and

[5] Victor, Bret. Learnable Programming, September 2012. http://worrydream.com/LearnableProgramming/.

acquire the necessary skills to apply their knowledge in more traditional settings. While the articles seem to indicate that students need to understand why the functions they use work, this ignores the nuance of the extremely complicated structure of modern computing, instead requiring a curriculum that needs to provide a high level understanding of the computer and a holistic understanding of the code and what it means.

Computer Science education seems to engulf much more than just teaching students how to code. A class in computer science needs to establish its value to students, parents, and community members alike, proving its practical worth to communities around the world. Unlike disciplines like History, Language, Math, and Science, Computer Science's age proves to be the limiting factor in its teachability and thus, teachers need to adopt a style that teaches their students a new way of thinking that requires them to try new methods and problem solve, not just follow the steps of a process. Additionally, the teaching process of CS has to provide students with some familiarity and guidance. A class should spend time learning how to read Documentation pages and teaching students how to debug their code, while linking core concepts back to counterparts in block based programming and in English. Students need to be taught to vocalize their ideas in a format that can be easily translated into code.

# Chapter 3: Methods

In order to observe the feasibility of combining computer science and computer engineering curriculum, a six week test course will be conducted where students will be taught different elements of each subject in such a manner that will help build their knowledge of both subjects and allow students to understand the operations they are doing on both a micro and macro level. In order to test the effectiveness of this curriculum, it will then be taught by a professional teacher to a diverse group of students with prior exposure to block based programming.

During the creation of the curriculum, the main considerations at hand were to help build a fundamental intuitive understanding of CS in students in a pseudocode with syntax reflecting that of existing programming languages designed to give students a way to vocalize their ideas. Ideally, students would have been able to implement their understanding in a language like Python that is simple to understand and used industry wide, however, due to a lack in availability for teachers through Covid-19 this was not possible.

Before creating the course, some assumptions were made about the students' prior knowledge. Students were expected to have mastery over basic arithmetic, being able to fluently add, subtract, multiply, and divide 1 digit numbers in their heads, with the tools necessary to perform these operations on larger numbers. Students were also expected to have prior exposure to block based programming so that they are able to smoothly grasp basic concepts in written code.

The course material is structured in the form of a book, and has several “Chapters” that each cover a Major Concept. While this is by no means comprehensive, it aims to provide students with the intuition necessary to learn to code in a real programming language such as Python. The material created would span about 1 semester (about 12 weeks), and includes the concepts, but does not include exercises which can be found in other external references.

The book is not dense with material, but aims to build up the concepts of programming in an intuitive sense, and coupled with teacher instruction in a programming language and adequate exposure to documentation & debugging can allow students to have a holistic understanding of Computer Science.

# Chapter 4: Results

The course is contained in the format of a book titled *A Guide to Intuitive Programming* which, when complimented with supplementary exercises related to each of the chapters can form a course. This is by no means a complete teachable curriculum, but it contains the key ideas necessary to guide teachers in following the best practices for teaching Computer Science.

Link:

https://github.com/ghostlypi/A-Guide-to-Intuitive-Programming/blob/main/A%20Guide%20to%20Intuitive%20Programming.pdf

# Chapter 5: Reflections

Having had some level of formal education in CS through all four years of high school, I set out upon this endeavour to in part, formalize the means by which I learnt CS and became interested in the field which I'm now pursuing so fervently. Initially, I was inclined to focus on the interaction between Computer Science and Computer Hardware, looking into approaches that can be used to help use Hardware to accelerate software processes. Although this was a noble goal, there are many complexities which I realized could not be covered in full in a basic computer science course, notwithstanding the nacency of the idea of using hardware acceleration in modern computing applications.

Pivoting away from this methodology, I settled on developing a curriculum that mentions hardware, but focuses on building up the intuition and state of mind for students such that programming becomes second nature, allowing them to verbalize a problem, break it down and solve it in parts. With a rough idea on how to do this, I began work on trying to use a student's mathematical and language knowledge as a jumping off point to their understanding of Computer Science, writing out core concepts in terms of english words with pseudocode examples to accompany them.

In this project, one of the biggest surprises was in the body of best practices knowledge for CS. In addition to being sparse, to say the least, much of the information was contradictory, arguing on everything from languages to teaching styles to whether students should be allowed to bootstrap (copy) code from the internet. In order to synthesize such a diverse and

contradictory body of knowledge, I developed a scheme that relied on industry practices and their equivalents in a learning environment. The issue of bootstrapping from the internet was a particularly complex issue as some teachers argued that it was essentially a means for students to “cheat” on their code by copying it off the internet without understanding what it did. On the other hand, teachers argued that if students could copy similar code and modify it to meet their use case, then they were essentially quoting the code in their program, using some components as reference whilst really doing the task themselves. I consolidated these opinions with the industry practice and decided that bootstrapping should be not only encouraged, but taught. The risk of cheating only comes when students are not able to read documentation properly and are not taught how to read and interpret code, all problems that can be eliminated in the classroom through lecture and practice.

Additionally, the task of picking a language to teach was one which I was only partly willing to engage in. At the rate at which new programming languages are popping up it is difficult to provide a definitive reason for which language is the best for students to learn, thus I chose to write the book entirely in Pseudocode. With this in mind, I did recommend at the end of the book that Python is a great language to get started on due to its industry prominence and also because of its ease of use. While it is by no means the best language for every job, it offers an easy gateway into CS allowing students to dive deeper into languages like C and Java and explore different frameworks for how they can write code.

# Bibliography

"1971: Microprocessor Integrates CPU Function onto a Single Chip." 1971: Microprocessor Integrates CPU Function onto a Single Chip | The Silicon Engine | Computer History Museum. Accessed December 18, 2020. https://www.computerhistory.org/siliconengine/microprocessor-integrates-cpu-function-onto-a-single-chip/.

Ambalam, Visaal. "The Invisible Curriculum: What Colleges Fail To Teach." Visaal. Visaal, August 6, 2020. https://www.visaalambalam.com/the-invisible-curriculum/.

Arnaud. "Six Reasons Why Computer Science Education Is Failing Students." LiveCode, October 10, 2012. https://livecode.com/six-reasons-why-computer-science-education-is-failing-students/.

Guzdial, Mark. "3 Proposals to Change How We Teach Computing In Order to Reduce Inequality." ACM, August 2, 2020. https://cacm.acm.org/blogs/blog-cacm/246550-3-proposals-to-change-how-we-teach-computing-in-order-to-reduce-inequality/fulltext.

"Problems in CS Education - High School Computer Science Education." Google Sites. Accessed December 18, 2020. https://sites.google.com/site/highschoolcseducation/background/problems-in-cs-education.

Victor, Bret. Learnable Programming, September 2012.

http://worrydream.com/LearnableProgramming/.