Overview: This is a 4 week unit zero, basic skills, for a 7th grade Introduction to Computer Science course. It will give an overview of the following topics: history/evolution of computers; computing languages; how computers work: hardware, software, memory, processing; universal programming concepts and keyboarding and shortcuts . The purpose is to provide students with foundational skills for a middle school computer science class because there is a disconnect between students and past and present technology and how we got here. Also, students have various exposures to computer education so this unit will help to build background knowledge.

Content & Skills: At the end of this unit, students will know/be able to…

Lesson 1 Learning Target: I can analyze the evolution of computers (2-3 days)
Success Criteria/Objectives:

- ☐ summarize key events in the history of computers
- ☐ discuss the different forms and types of computers that emerged throughout history
- ☐ list key players in the history of computers
- ☐ explain how and why inventions can change the way we live

Lesson 2 Learning Target: I can investigate how computers work (2-3 days)
Success Criteria/Objectives:

- ☐ explore the inside of a computer.
- ☐ explain the function of each part of a computer
- ☐ Determine the meaning of unfamiliar computer terms (input, output, processing, storage)

Lesson 3 Learning Target: I can investigate why programming languages exist (about 2-3 days)
Success Criteria/Objectives:

- ☐ Present an overview of assigned programming language
- ☐ Explain the qualities that differentiate natural languages and programming languages
- ☐ Justify the existence of programming languages to precisely communicate instructions
- ☐ identify and explain the two types of programing language
- ☐ explain the two categories of low-level language
- ☐ define 'compiler' and 'interpreter' and explain what each is
- ☐ provide examples of interpreter and compiler language

Lesson 4 Learning Target: I can explore the big ideas of universal programming with Scratch Creative Computing tutorials (10-15 days)
Success Criteria/Objectives:

- ☐ Use Variables, Conditionals, Loops, Events, Procedures in Scratch Creative Computing tutorials

Lesson 5 Learning Target: I can use proper keyboarding technique when typing (10-15 days)
Success Criteria/Objectives:

- ☐ use correct hand posture on every key
- ☐ Use keyboard shortcuts
- ☐ Increase typing speed and accuracy

Enduring Understandings: 40 years from now, students will understand…

Thousands of years ago, people needed a way to visualize doing mathematical calculations, like addition and subtraction. So, they invented the abacus. An abacus was a wooden rack containing rows of beads. Calculations were made by moving beads on the rack to left or to the right. Abacuses are still in use today. During the 17th century, a teenager named Pascal wanted to help his father add up the money he received as a tax collector. Pascal invented a mechanical calculator that used moving parts, such as gears, to add and subtract. A man named Charles Babbage devised a calculating machine known as a calculation engine. This was a very large and complicated device, with lots of rods and moving gears. It had four components: an input, a processor, memory, and an output. These four components are still a part of all computers today. The very first computer programmer was Ada Lovelace, who was writing sequences for computers that didn't even exist yet. She also predicted that sound, music, text, and pictures could be made digital.

In the computer world there are about 500+ programming languages with their own syntax and features. A program is a set of instructions that tells a computer what to do in order to come up with a solution to a particular problem. Programs are written using a programming language. A programming language is a formal language designed to communicate instructions to a computer. There are two major types of programming languages: low-level languages and high-level languages.

Computing affects many aspects of the world at local, national, and global levels. Computational thinking involves thinking about and solving problems in ways that can be carried out by a computer. Computational thinking includes concepts such as variables, conditionals, loops, events, procedures and algorithms to solve problems across disciplines.

Essential Questions:

1. What is a computer? How have computers changed over time?
2. What are the basic components of a computer? How do they work?
3. What are coding languages? What are the different languages used for? How have computer languages changed over time?
4. What are universal programming concepts?
    a. Variables
    b. Conditionals
    c. Loops
    d. Events
    e. Procedures
5. How can proper keyboarding technique and shortcuts increase accuracy and productivity?

Lesson 1
Learning Target: I can analyze the evolution of computers
Success Criteria/Objectives:

- ☐ summarize key events in the history of computers
- ☐ discuss the different forms and types of computers that emerged throughout history
- ☐ list key players in the history of computers
- ☐ explain how and why inventions can change the way we live

Standards: 7-8.DL.5 Transfer knowledge of technology in order to explore new technologies; New technologies could include different tools for collaboration, creation, etc. that the student has not used before.

Key Ideas: People have used mechanical calculators like the abacus since ancient times. Greek astronomers took it a step further, with precision instruments that tracked the stars. A computer is an electronic device that accepts data (input), manipulates the data (process), produces information based on the manipulation (output) and stores the results (storage). In 1884, Charles Babbage, an English mathematician, tried to build a complicated machine called the "analytical engine." It was mechanical, rather than electronic, and Babbage never completed it, but computers today are based on many of the principles he used in his design.

1. **Introduction**
    a. What is a computer?
    b. When were computers created?
    c. What do you want to learn about the history of computers?
2. **Mini Lesson:**
    a. Video History of Computers – How were Computers Invented Short Documentary or BrainPOP Computer History
        i. Watch, think, write:
            1. Jot the gist: big ideas, key points, important details, new vocabulary terms, models, diagrams, concept maps
            2. Jot any questions you have about the topic
        ii. Discuss-Talk about your notes using the sentence starters
        iii. Write-Summarize the big ideas using the key points, new vocabulary, etc.
3. **Activity**
    a. Each group is assigned to a decade from the 1930-2015 of the Timeline of Computer History and reports back to the class on the major developments in computing history during that time period.
4. **Summary**
    a. Draw a timeline that shows 4 major developments in the evolution of computing/computers.
    b. Who are 2 pioneers in the history of computing? What were their contributions?
    c. How and why can inventions change the way we live?
5. **Out of class practice/homework**
    a. https://quizizz.com/admin/quiz/5e1898ae9cbbdf001f57e15a/computing-history
    b. https://quizizz.com/admin/quiz/60988a1c00f5c2001cc9f426/history-of-computer

6. **Resources**
    a. https://www.computerhistory.org/timeline/computers/
    b. https://sciencing.com/history-computers-kids-7982362.html
    c. https://www.britannica.com/technology/computer/The-first-computer
    d. https://www.commonsense.org/education/lesson-plans/computer-history-timeline-with-google-slides
    e. https://www.commonsense.org/education/lesson-plans/parts-of-a-computer
    f. https://educators.brainpop.com/lesson-plan/inventions-lesson-plan-origins-computers-technology/
    g. https://www.tsl.texas.gov/sites/default/files/public/tslac/ld/ld/LibrariesLiteracy/1-1%20Computer%20Basics%20Lesson%20Plan.pdf
    h. http://www.nysed.gov/common/nysed/files/programs/curriculum-instruction/computer-science-digital-fluency-standards-k-12.pdf
    i. Modern Marvels: How the First Computer Changed the World (S2, E11) | Full Episode | History https://www.youtube.com/watch?v=OSQld7xVMn0&t=13s
    j. https://www.windham-schools.org/docs/DOK%20Wheel%20Slide%20for%20Teachers-0.pdf

Lesson 2 Learning Target: I can investigate how computers work
Success Criteria/Objectives:

- ☐ explore the inside of a computer.
- ☐ explain the function of each part of a computer
- ☐ Determine the meaning of unfamiliar computer terms (input, output, processing, storage)

Standards: 7-8.NSD.2 Design a project that combines hardware and software components.

Key Ideas: The main part is the central processing unit, which is sometimes abbreviated as CPU. It's the main part of the computer that uses a processor to follow directions. Information is put into the CPU, and this is called input. A processor is a device that helps the computer read the input and determines what it needs to do. It's very small, and sometimes it's referred to as a chip. This chip might be tiny, but it is powerful. Processors perform many complicated calculations (Arithmetic Logic Unit performs the arithmetic and logical functions (addition, subtraction, multiplication, division, equals, not equal, equal to or greater than, equal to or less than, greater than, less than, etc.). A computer is composed of many different circuits. Inside of each computer is the main circuit board called a motherboard. This is where important circuits are located. The actual motherboard is made of many different materials. The board itself is composed of fiberglass, which is an insulator. This helps control the flow of electricity to other parts of the computer. The actual circuits are usually made from copper, which is a conductor. This allows the electricity to flow within the motherboard. A chipset is a group of microchips designed to work as a unit in performing one or more related functions. BIOS (Basic Input/Output System) is the program a computer's microprocessor uses to get the computer system started after it is turned on. An operating system (sometimes abbreviated as "OS") is the program that manages all the other programs in a computer. Memory is the name for the electronic holding place for instructions and data that a computer's microprocessor can reach quickly.

1. **Introduction**
    a. Complete the Computer Basics Quiz preassessment and note what topics you need to practice https://edu.gcfglobal.org/en/computerbasics/computer-basics-quiz/1/
2. **Mini Lesson:**
    a. Video Inside your computer - Bettina Bair
        i. Watch, think, write:
            1. Jot the gist: big ideas, key points, important details, new vocabulary terms, models, diagrams, concept maps
            2. Jot any questions you have about the topic
        ii. Discuss-Talk about your notes using the sentence starters
        iii. Write-Summarize the big ideas using the key points, new vocabulary, etc.

3. **Activity**

   Complete the self paced lessons with your thought partners

   Mild - Goodwill Community Foundation computer basic tutorials

   Medium - Khan Academy computer basic tutorials

   Spicy - Bradley Kjell Hardware & Software Part 1 Introduction to Computer Science using Java

4. **Summary**

   Complete the 9 multiple choice and open-ended questions at TedTed

   Complete the 4 Kjell quizzes in part 1 of hardware - software

5. **Out of class practice/homework**

   Readings -

   How computers work 1

   How computers work 2

   Build a computer simulation

6. **Additional practice**

   computer science discoveries chapter 2 code.org

   Computer images and videos

7. **Resources**

https://chortle.ccsu.edu/Java5/index.html#03
https://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading04.htm
https://www.khanacademy.org/computing/code-org/computers-and-the-internet#how-computers-work
https://www.codeforfun.com/grade-5-unit-1
https://www.edutopia.org/article/teaching-students-code-using-free-simulators
http://www.nysed.gov/common/nysed/files/programs/curriculum-instruction/computer-science-digital-fluency-standards-k-12.pdf
https://www.windham-schools.org/docs/DOK%20Wheel%20Slide%20for%20Teachers-0.pdf

Lesson 3 Learning Target: I can investigate why programming languages exist

Success Criteria/Objectives:

- ☐ Present an overview of assigned programming language
- ☐ Explain the qualities that differentiate natural languages and programming languages
- ☐ Justify the existence of programming languages to precisely communicate instructions
- ☐ identify and explain the two types of programing language
- ☐ explain the two categories of low-level language
- ☐ define 'compiler' and 'interpreter' and explain what each is
- ☐ provide examples of interpreter and compiler language

Standards:

4-6.CT.6 Compare two or more algorithms and discuss the advantages and disadvantages of each for a specific task. 7-8.CT.6 Design, compare and refine algorithms for a specific task or within a program. 9-12.CT.6 Demonstrate how at least two classic algorithms work, and analyze the trade-offs related to two or more algorithms for completing the same task.

Key Ideas:

A program is a set of instructions that tells a computer what to do in order to come up with a solution to a particular problem. Programs are written using a programming language. A programming language is a formal language designed to communicate instructions to a computer. There are two major types of programming languages: low-level languages and high-level languages. Low-level languages are referred to as 'low' because they are very close to how different hardware elements of a computer actually communicate with each other. Low-level languages are machine oriented and require extensive knowledge of computer hardware and its configuration. There are two categories of low-level languages: machine language and assembly language.

1. **Introduction**

   What is a language?

   How do computers know what to do?

2. **Mini Lesson:**

If you're listening to morning announcements at school and hear that there will be a practice fire drill, how does this information process into your knowledge of understanding? Draw how this information flows. For example, draw a circle with the words 'Fire drill announcement' inside, then an arrow, then a circle with the words 'Exit the building at the sound of the alarm.' Think of a more complicated command structure: learning to cook a dish, or doing a drill in sports. Have groups first write this structure, then draw on a flow-chart map on chart paper, then share with the class.

3. **Activity**

a. Paired Reading - Programming Languages (see reading below)
b. Presentation on a programming language - student pairs are assigned to one programming language and give an overview of its history use and example code (C, Python, C++, Java, SCALA, C#, R, Ruby, Go, Swift, JavaScript)

4. **Summary**

What are three things you learned, two things you're still curious about, and one thing you don't understand?

5. **Out of class practice/homework**

Readings - coding languages computer programming language

6. **Resources**

a. https://www.geeksforgeeks.org/the-evolution-of-programming-languages/
b. https://study.com/academy/lesson/machine-code-high-level-language-lesson-plan.html
c. https://www.w3schools.com/ai/ai_history_languages.asp
d. https://ny.pbslearningmedia.org/resource/first-programming-language-crash-course-cs/first-programming-language-crash-course-cs/
e. Coded bias teacher guide
f. https://chortle.ccsu.edu/Java5/index.html#03
g. https://studio.code.org/s/csp3-2021/lessons/5
h. http://www.nysed.gov/common/nysed/files/programs/curriculum-instruction/computer-science-digital-fluency-standards-k-12.pdf
i. https://www.windham-schools.org/docs/DOK%20Wheel%20Slide%20for%20Teachers-0.pdf

Reading: **Machine Code and High-level Languages: Using Interpreters and Compilers**

VOCABULARY
Programming language
Low-level language
Machine language
Machine code
Assembly language
High-level language
Compiler
Interpreter

Machine language, or machine code, is the only language that is directly understood by the computer, and it does not need to be translated. All instructions use binary notation and are written as a string of 1s and 0s. A program instruction in machine language may look something like this:

10010101100101001111101010011011100101
Technically speaking, this is the only language computer hardware understands. However, binary notation is very difficult for humans to understand. This is where assembly languages come in. An assembly language is the first step to improve programming structure and make machine language more readable by humans. An assembly language consists of a set of symbols and letters. A translator is required to translate the assembly language to machine language. This translator program is called the 'assembler.' It can be called the second generation language since it no longer uses 1s and 0s to write instructions, but terms like MOVE, ADD, SUB and END.

Many of the earliest computer programs were written in assembly languages. Most programmers today don't use assembly languages very often, but they are still used for applications like operating systems of electronic devices and technical applications, which use very precise timing or optimization of computer resources. While easier than machine code, assembly languages are still pretty difficult to understand. This is why high-level languages have been developed.

A high-level language is a programming language that uses English and mathematical symbols, like +, -, % and many others, in its instructions. When using the term 'programming languages,' most people are actually referring to high-level languages. High-level languages are the languages most often used by programmers to write programs. Examples of high-level languages are C++, Fortran, Java and Python.

To get a flavor of what a high-level language actually looks like, consider an ATM machine where someone wants to make a withdrawal of $100. This amount needs to be compared to the account balance to make sure there are enough funds. The instruction in a high-level computer language would look something like this:

```
x = 100
if balance  x:
print 'Insufficient balance'
else:
 print 'Please take your money'
```

This is not exactly how real people communicate, but it is much easier to follow than a series of 1s and 0s in binary code.

There are a number of advantages to high-level languages. The first advantage is that high-level languages are much closer to the logic of a human language. A high-level language uses a set of rules that dictate how words and symbols can be put together to form a program. Learning a high-level language is not unlike learning another human language - you need to learn vocabulary and grammar so you can make sentences. To learn a programming language, you need to learn commands, syntax and logic, which correspond closely to vocabulary and grammar.

The second advantage is that the code of most high-level languages is portable and the same code can run on different hardware. Both machine code and assembly languages are hardware specific and not portable. This means that the machine code used to run a program on one specific computer needs to be modified to run on another computer. Portable code in a high-level language can run on multiple computer systems without modification. However, modifications to code in high-level languages may be necessary because of the operating system. For example, programs written for Windows typically don't run on a Mac.

A high-level language cannot be understood directly by a computer, and it needs to be translated into machine code. There are two ways to do this, and they are related to how the program is executed: a high-level language can be compiled or interpreted.

A compiler is a computer program that translates a program written in a high-level language to the machine language of a computer. The high-level program is referred to as 'the source code.' A typical computer program processes some type of input data to produce output data. The compiler is used to translate source code into machine code or compiled code. This does

not yet use any of the input data. When the compiled code is executed, referred to as 'running the program,' the program processes the input data to produce the desired output.

When using a compiler, the entire source code needs to be compiled before the program can be executed. The resulting machine code is typically a compiled file, such as a file with an .exe extension. Once you have a compiled file, you can run the program over and over again without having to compile it again. If you have multiple inputs that require processing, you run the compiled code as many times as needed.

Lesson 4 Learning Target: I can explore the big ideas of universal programming with Scratch Creative Computing tutorials

Success Criteria/Objectives:

- ☐ Use Variables, Conditionals, Loops, Events, Procedures in Scratch Creative Computing tutorials

Standards:

7-8.CT.4 Write a program using functions or procedures whose names or other documentation convey their purpose within the larger task.
7-8.CT.5 Identify multiple similar concrete computations in a program, then create a function to generalize over them using parameters to accommodate their differences
7-8.CT.6 Design, compare and refine algorithms for a specific task or within a program.
7-8.CT.7 Design or remix a program that uses a variable to maintain the current value of a key piece of information.
7-8.CT.9 Read and interpret code to predict the outcome of various programs that involve conditionals and repetition for the purposes of debugging.

Key Ideas:

- sequence- identifying a series of steps for a task
- loops- running the same sequence multiple times
- parallelism- making things happen at the same time
- events- one thing causing another thing to happen
- conditionals- making decisions based on conditions
- operators- support for mathematical and logical expressions
- being iterative and incremental- developing a little bit, then trying it out, then developing some more
- testing and debugging- making sure that things work – and finding and fixing mistakes
- reusing and remixing- making something by building on what others – or you – have done
- abstracting and modularizing- building something large by putting together collections of smaller parts

1. **Introduction**

- What are the different ways you interact with computers?
- How many of those ways involve you creating with computers?

2. **Mini Lesson:**

- Explain that over the next several sessions they will be creating their own interactive computational media with Scratch.
- Show a basic demo of Scratch

- You build projects by snapping blocks together, just as you can build things in the physical world by snapping LEGO bricks together.
- There are more than 100 blocks in 8 different categories.
- As a small example, let's make the cat do a dance.
- Start by dragging out the "move 10 steps" block from the "Motion" blocks palette to the scripting area.
- Every time you click on the block the cat moves a distance of 10.
- You can change the number to make the cat move a greater or smaller distance.
- From the "Sound" palette, drag out the "play drum" block. Click on the block to hear its drum sound. Drag and snap the "play drum" block below the "move" block. When you click on this stack of two blocks, the cat will move and then play the drum sound.
- Copy this stack of blocks (either using the Duplicate toolbar item or by right clicking the stack and selecting "duplicate") and snap the copy to the already placed blocks.
- Change the second "move" block to -10 steps, so the cat moves backward. Every time the stack of four blocks is clicked, the cat does a little dance forward and back.
- Go to the "Control" blocks palette and grab the "repeat" block. Wrap the "repeat" block around the other blocks in the scripting area. Now when you click on the stack, the cat dances forward and back 10 times.
- Finally, drag the "when Sprite clicked" block and snap it to the top of the stack. Click on the cat (instead of the blocks stack) to make the cat dance.
- Show the range of projects they will be able to create, by reviewing the self paced tutorials
- Tell students they will maintain a design notebook, for recording their coding notes, ideas and plans, as well as for responding to the questions

3. Activity

Give students 10 minutes to explore the Scratch interface in an open-ended way. One prompt is: "You have 10 minutes to make something surprising happen to a sprite." Students are encouraged to work together, ask each other for help, and share what they are figuring out during the 10 minutes.

4. Summary

Ask for 3 or 4 volunteers to share with the entire group one thing that they discovered. Optionally, after the volunteers have shared, offer several challenges to the students: Did

anyone figure out how to add sound? Did anyone figure out how to change the background? Did anyone figure out how to access the help screens for particular blocks?

5. **Out of class practice/homework**

Students complete the Scratch tutorials at home and in class choose any 13 of the 26 tutorials.

- Students explore the arts by creating projects that include elements of music, design, drawing, and dance. The computational concepts of sequence and loops, and the computational practices of being iterative and incremental are highlighted.
- Students explore storytelling by creating projects that include characters, scenes, and narrative. The computational concepts of parallelism and events and the computational practices of reusing and remixing are highlighted.
- Students explore games by creating projects that define goals and rules. The computational concepts of conditionals, operators, and data, and the computational practices of testing and debugging are highlighted.
- Students develop independent projects by defining a project to work on, collaborating with others to improve the project, and presenting the project and its development process. The computational practices of abstracting and modularizing are highlighted.

6. **Resources**

a. https://scratched.gse.harvard.edu/sites/default/files/curriculumguide-v20110923.pdf
b. https://ccl.northwestern.edu/netlogo/docs/programming.html
c. UDL for CS https://ctrl.education.ufl.edu/wp-content/uploads/sites/5/2020/05/Copy-of-UDL-and-CS_CT-remix.pdf
d. https://www.helloruby.com/educators
e. http://www.nysed.gov/common/nysed/files/programs/curriculum-instruction/computer-science-digital-fluency-standards-k-12.pdf
f. https://www.windham-schools.org/docs/DOK%20Wheel%20Slide%20for%20Teachers-0.pdf
g. https://code.org/curriculum/docs/csf/CSF_TeacherGuide_CoursesA-F_v2a_small.pdf
h. Exploring CS curriculum
i. https://www.dummies.com/article/technology/programming-web-design/coding/helping-kids-coding-dummies-cheat-sheet-252828

Lesson 5 Learning Target: I can use proper keyboarding technique when typing

Success Criteria/Objectives:

- ☐ use correct hand posture on every key
- ☐ Use keyboard shortcuts
- ☐ Increase typing speed and accuracy

Standards:

7-8.DL.1 Type on a keyboard while demonstrating proper keyboarding technique, with increased speed and accuracy

Key Ideas:

Touch typing makes you faster. This, in turn, means you will be more productive, as it takes you less time to do tasks so you can take on more work or assignments, or alternatively spend less time sitting at a computer. Automatizing the process improves the quality of your writing too. It frees up cognitive energy so you focus on the ideas instead of just the language required to articulate them.

Keyboard shortcuts are keys or key combinations you can press on your computer's keyboard to perform a variety of tasks. Because both of your hands can remain on the keyboard, using a shortcut to perform a task is often faster than using a mouse. Keyboard shortcuts are also universal—meaning once you learn them, you can use many of the same shortcuts in a variety of applications.

1. **Introduction**
    - Check your typing skills in 60 seconds - 1 minute test https://www.typingtest.com/
    - Why is it important to learn keyboarding/touch typing?
    - What are some keyboarding shortcuts that you know?
2. **Mini Lesson:**

    Review the 6 reasons to learn touch typing here and keyboard shortcuts here
    - 1. Speed - This is going to be the first and most obvious benefit of learning to touch type. A touch typist can easily reach typing speeds above 75-80 words per minute, while a non-trained individual is around 10.
    - 2. Accuracy - One of the most important things to learn no matter how hard you type is to type accurately.
    - 3. Time - If you increase your typing speed for example, from say 30 words per minute to 60, you have effectively halved the time it would take you to do the same amount of work.
    - 4. Decrease Fatigue - Typing is both psychologically and physically exhausting when done for long periods of time. Learning to touch type properly reduces mental and physical fatigue. Mentally, it keeps you from having to focus on two things at once. All you have to worry about is your output, not finding the individual keys. Physically, it keeps you from constantly having to bend your head over the keyboard to find your next couple of keystrokes.

- 5. Health - Overall, touch typing is better for your health. You're not hunched over looking at the keys, and using all of your fingers actually reduces the risk for repetitive stress injuries (RSI).
- 6. Productivity - By learning touch typing, you shall become more productive and even increase your own confidence. Time is nearly halved and errors will become near enough non-existent. Touch typing is a skill to be proud of and is desired within within most industries.

3. **Activity**

Student begin self paced touch typing lessons at https://www.typingclub.com/
Students learn about computer shortcuts for Windows and macOS computers here:
https://edu.gcfglobal.org/en/techsavvy/keyboard-shortcuts/1/

4. **Summary**

Students review shortcuts with Quizlet flashcards
https://quizlet.com/3054050/keyboard-shortcuts-flash-cards/

5. **Out of class practice/homework**

Students complete self paced keyboarding practice (2 weeks)

6. **Resources**

25 Essential Windows Keyboard Shortcuts You Need to Know Now

15 Amazing Shortcuts You Aren't Using video

10 Easy Shortcuts Everybody Needs to Know in 2020 video