

# Unit 2 - Digital Information

This unit explores the way large and complex pieces of digital information are stored in computers and the associated challenges. Through a mix of online research and interactive widgets, students learn about foundational topics like compression, image representation, and the advantages and disadvantages of different file formats. To conclude the unit, students research the history and characteristics of a real-world file format.

## Chapter 1: Digital Information

### Big Questions

- How are images and other complex information represented in a computer?
- How can we reduce the size of digital information and what tradeoffs are involved?
- Why are there so many different formats for representing the same kind of information?

### Enduring Understandings

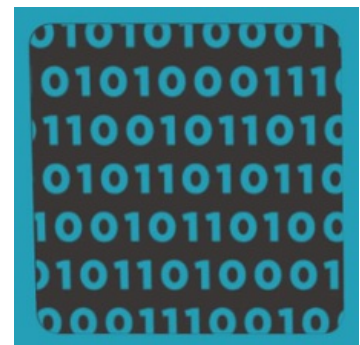
- 1.1 Creative development can be an essential process for creating computational artifacts.
- 1.3 Computing can extend traditional forms of human expression and experience.
- 2.1 A variety of abstractions built upon binary sequences can be used to represent all digital data.
- 3.3 There are trade offs when representing information as digital data.

### Week 1

### Lesson 1: Bytes and File Sizes

#### Research

Students are introduced to the standard units for measuring the sizes of digital files: bytes, kilobytes, megabytes, gigabytes, etc. and research the sizes of files they make use of every day.



## Lesson 2: Text Compression

**Widget | Individual and Group Discovery**

At some point we reach a physical limit of how fast we can send bits and if we want to send a large amount of information faster, we have to find a way to represent the same information with fewer bits - we must compress the data.



## Lesson 3: Encoding B&W Images

**Widget | Concept Invention | Individual Creation**

Students explore methods for encoding digital images in binary which requires representing metadata such as width and height as well as pixel data. Students use the the Pixelation widget to encode simple B&W raster images.



## Week 2

## Lesson 4: Encoding Color Images

Students learn about the RGB color encoding scheme and use an updated version of the pixelation widget to encode color images. Hexadecimal notation is useful for representing larger groupings of binary digits.



## Lesson 5: Lossy vs. Lossless Compression

**Research**

Students learn the difference between lossy and lossless compression. They then research three different real-world file formats for images and sound and compare them using the concepts and vocabulary they've learned through the unit.



## Lesson 6: Rapid Research - Format Showdown

### Project | Research

In this lesson students will conduct a small amount of research to explore a file format either currently in use or from history. Students will conduct research in order to complete a "one-pager" that summarizes their findings. They will also design a computational artifact (video, audio, graphic, etc.) that succinctly summarizes the advantages of their format over other similar ones.

## Chapter Commentary

### Key Concepts and Pedagogy

**Increasing Focus on Research:** Unit 2 includes an increasing focus on research skills students will use throughout the rest of the course and eventually on the Explore PT. Researching computing topics is a challenging skill which students will need to develop separately from the other skills and content knowledge. Lesson 5 and the Rapid Research activity in Lesson 6 in particular explicitly review these research skills as students apply the knowledge they've developed elsewhere in the unit to research the differences between familiar file formats.

**A Familiar Pedagogy:** Many of the lessons in this unit will feel similar to Unit 1 Chapter 1. Students will explore real world concepts and problems in data representation through collaborative problem solving activities. This process is aided through the introduction of two new widgets, the Text Compression Widget in Lesson 2 and the Pixelation Widget in Lessons 3 and 4. As with the Internet Simulator, students can't "break" these tools and so are encouraged to freely explore them with minimal content frontloading to discover the concepts they highlight. In keeping with the "ABC CBV" approach, technical vocabulary is only introduced only once these exploratory activities ensure all students have developed a shared understanding of the concepts.



This curriculum is available under a  
Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

# Lesson 1: Bytes and File Sizes

## Research

## Overview

In this lesson students are introduced to the standard units for measuring the sizes of digital files, from a single byte, all the way up to terabytes and beyond. Students begin the lesson by comparing the size of a plain text file containing “hello” to a Word document with the same contents. Students are introduced to the units kilobyte, megabyte, gigabyte, and terabyte, and research the sizes of files they make use of every day, using the appropriate terminology. This lesson foreshadows an investigation of compression as a means for combatting the rapid growth of digital data.

## Purpose

The simple purposes of this lesson are:

1. Get terminology out in the open
2. Become somewhat conversant with file types and sizes
3. Grapple with orders-of-magnitude differences between things.

The 8-bit byte has become the de-facto fundamental unit with which we measure the “size” of data on computers, and in fact, today most computers only let you save data as combinations of whole bytes; even if you only want to store 1 bit of information, you have to use a whole byte to do it. And many computer systems will require you store even more than that. Messages sent over the Internet are also typically structured as messages with byte-offsets.

Paralleling the explosion of computing power and speed, the sheer size of the digital data now created and consumed every day is staggering. Units of measure (terabytes) that previously seemed unfathomably large are now making their way into personal computing. This rapid growth of digital data presents many new opportunities and also poses new challenges to engineers and programmers. The implications of so-called **Big Data** will not be investigated until later in the course, but it's good and interesting to be thinking about the size of things now.

## Agenda

### Getting Started (10 mins)

File Size Comparison: .txt vs .doc

### Activity (30 mins)

### Wrap-up

Review worksheet

Foreshadow Compression

## View on Code Studio

## Objectives

### Students will be able to:

- Use appropriate terminology when describing the size of digital files.
- Identify and compare the size of familiar digital media.
- Solve small word problems that require reasoning about file sizes.

## Preparation

☐ You should verify that you know how to look at the sizes of files on computers that your students are using (see activity).

☐ For the getting started activity might want a Word processing program (such as MS Word) and plain text editor (such as Notepad or TextEdit) open and ready.

☐ The teaching remarks and content corners in this lesson contain lots of little bits of history that you might choose to share at various points in the lesson.

☐ **KEY Bytes and File Sizes**

## Links

**Heads Up!** Please make a copy of any documents you plan to share with students.

### For the Students

- **Bytes and File Sizes** - Activity Guide

[Make a Copy](#)

- **CS101 - Kilobytes Megabytes Gigabytes Terabytes**



# Teaching Guide

## Getting Started (10 mins)

### Remarks

As we start a new unit about Data and Digital Information we need to get familiar with terminology about data and different types of data files.

**Vocabulary:** Recall that a single character of ASCII text requires 8 bits. The technical term for 8 bits of data is a byte.

A **byte** is the standard fundamental unit (or “chunk size”) underlying most computing systems today. You may have heard “megabyte”, “kilobyte”, “gigabyte”, etc. which are all different amounts of a bytes. We’re going to learn more about them today.


### File Size Comparison: .txt vs .doc

**Prompt:** In addition to the actual text of a document, it is usually necessary to store the formatting information that allows the text to be displayed correctly. We might wonder just how much extra information, i.e. how many extra bytes, we need to store when we include all of this formatting. Let’s find out!

If a single ASCII character is one byte then if we were to store the word “hello” in a plain ASCII text file in a computer, we would expect it to need 5 bytes (or 40 bits) of memory.

What about a Microsoft Word document that contains the single word “hello”? How many more bytes will a Word document require to store the word “hello” than a plain text document?

**Discuss:** Have students silently make their prediction, then share with a partner, then share with the group. Prompt a couple students to share why they chose the size they did.

 **Demonstrate:** Do a live demo where you show the size of the different files. Here are some files you can download to use.

- Plain text document **hello.txt**
- MS word document **hello.docx**

To find the actual size of a file on your computer, do one of the following:

- PC/Windows: Right-click and choose “Properties”
- Mac: Ctrl+click and choose “Get Info”

In general, the Word Doc should be **thousands of times larger** than the plain text. For the files above:

- hello.txt - 5 bytes
- hello.docx = 21,969 bytes

**Review:** Review students predictions to see how close they were.

### Remarks

### Content Corner

**Why is a Byte 8 bits?:** The 8-bit byte was not always standard. Computers used many different “byte” sizes over the course of history, depending on hardware and how addressable memory worked. However, much of the early computing world relied on representing data and computer instructions encoded in ASCII text where every character is 8 bits. Thus, 8-bits was such a common chunk-size for representing information that it stuck and they gave it its own name - **byte**.

There are various accounts about why it was called a “byte” but most point to early days at IBM where “bite” was used to refer to groups of 8-bits that a computer was processing, as in it could “bite” off 8 bits at time. The spelling was changed to “byte” to avoid confusion with “bit”.

Bytes became the fundamental unit with which we measure the “size” of data on computers, and in fact, today most computers only let you save data as combinations of whole bytes; even if you only want to store 1 bit of information, you have to use a whole byte to do it.

### Teaching Tip

**Try a Live Demo:** If you wish, it might be more fun to create these files in front of your students, saving them on the desktop for a quick demo. To make a plain ASCII text file you’ll need to use the correct program:

- PC/Windows: use Notepad
- Mac: use TextEdit (Note: TextEdit needs to be switched into plain text mode from rich text. Go to Format → Make Plain Text)

The big difference in file size between .txt and .docx is due to the extensive formatting information included along with the actual text in .docx. Modern data files typically measure in the thousands, millions, billions or trillions of bytes. Let's get a little practice looking at files and how big they are.

## Activity (30 mins)

### **Activity Guide: Bytes and File Sizes - Activity Guide**

**Group:** Put students in pairs to find answers or work individually.

**Distribute:** Activity Guide: **Bytes and File Sizes - Activity Guide**

- Introduces the terminology
- Refers to websites for students to use as reference
  - **Stanford CS 101 website**
  - **Computer Hope**
- Has questions and space for students to write answers to questions like:
  - How many bytes are in a Megabyte?
  - Give an example of a file type that is measured in Gigabytes
  - What is the typical size of a .jpg image, .mp3 audio etc.
- Allow students time to finish this activity either individually or in pairs by conducting online research.
- There are 6 practice questions on the 2nd page of the activity guide.

## Wrap-up

### Review worksheet

**Share:** Provide students an opportunity to clear up any remaining confusion and share interesting pieces of information they came across.

**Review:** Answers to the questions on the Activity Guide.

## Foreshadow Compression

### Remarks

As you have seen data file size can grow very quickly in size. In the modern world there is a lot of data around us and usually we want it transmitted over the internet.

**There is a problem though :** If you want to transmit a lot of data you are limited by the speed of your internet connection. Even if you have a fast Internet connection there is a **physical limit** to how fast you can transmit bits.

### Content Corner

NOTE: A 5-byte file is so small that some computers won't allocate a chunk of memory that small. For example you might see something like this:

**Kind: Plain Text Document**  
**Size: 5 bytes (4 KB on disk)**

Which indicates that even though the file is 5 bytes, it's taking up 4 Kilobytes of memory on your computer.

### Content Corner

There are some discrepancies in common usage of the kilo, mega, giga prefixes.

From the **Stanford CS 101 website :**

**It's convenient within the computer to organize things in groups of powers of 2. For example,  $2^{10}$  is 1024, and so a program might group 1024 items together, as a sort of "round" number of things within the computer. The term "kilobyte" above refers to this group size of 1024 things. However, people also group things by thousands -- 1 thousand or 1 million items.**

**There's this problem with the word "megabyte" .. does it mean  $1024 * 1024$  bytes, i.e.  $2^{20}$  which is 1,048,576, or does it mean exactly 1 million,  $1000 * 1000$ . It's just a 5% difference, but marketers tend to prefer the 1 million, interpretation, since it makes their hard drives etc. appear to hold a little bit more. In an attempt to fix this, the terms "kibibyte" "mebibyte" "gibibyte" "tebibyte" have been introduced to specifically mean the 1024 based units (see wikipedia kibibyte article). These terms do not seem to have caught on very strongly thus far.**

If nothing else, remember that terms like "megabyte" have this little wiggle room in them between the 1024 and 1000 based meanings. **For purposes of CS Principles the distinction is not important - "about a million bytes" is a fine, close-enough interpretation for "megabyte".**

### Teaching Tip

**Finding Solutions:** Note that answers to 3 of the 6 questions on the activity guide can be found on the Stanford CS 101 page linked to in the activity guide.

Perfect accuracy is not important for some sections in this activity, but using the correct terminology and achieving a rough estimate of size (one million bytes vs. one billion) is important. Encourage students to practice using terms like megabyte, gigabyte, and terabyte to gain comfort with them.



What if the data you want to send is big enough that it takes an unreasonable amount of time to transmit it, even with a really fast internet connection. Assuming you can't make the Internet connection any faster, could you still transmit the data faster somehow?

💡 **The answer is yes** and it's probably something you've done, or do every day!

#### 💡 Teaching Tip

**Time Saving Tip:** Time permitting you could do the warm up activity from the next lesson (Text Compression)

## Assessment

Use the last 3 questions on the activity guide for assessment.

## Standards Alignment

### CSTA K-12 Computer Science Standards (2011)

- ▶ **CT** - Computational Thinking

#### Computer Science Principles

- ▶ **2.1** - A variety of abstractions built upon binary sequences can be used to represent all digital data.
- ▶ **3.3** - There are trade offs when representing information as digital data.

### CSTA K-12 Computer Science Standards (2017)

- ▶ **DA** - Data & Analysis



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

# Lesson 2: Text Compression

Widget | Individual and Group Discovery

## Overview

At some point we reach a physical limit of how fast we can send bits and if we want to send a large amount of information faster, we have to find a way to represent the same information with fewer bits - we must **compress** the data.

In this lesson, students will use the Text Compression Widget to compress segments of English text by looking for patterns and substituting symbols for larger patterns of text. After some experimentation students are asked to come up with a process (or algorithm) for arriving at a "good" amount of compression despite the fact that there is no way to know what is best or optimal. In developing a so-called "heuristic approach" to this problem, students will grapple with the tradeoffs in compressing data and begin to develop a sense of computing problems that are "hard" to solve.

## Purpose

This is a big lesson that covers a lot of bases. It should easily take 2 or more days of class. First and foremost it covers two or three topics directly from the CSP framework.

### 1. lossless compression

The basic principle behind compression is to develop a method or protocol for using fewer bits to represent the original information. The way we represent compressed data in this lesson, with a "dictionary" of repeated patterns is similar to the **LZW compression scheme**, but it should be noted that LZW is slightly different from what students do in this lesson. Students invent their own way here. LZW is used not only for text (zip files), but also with the GIF image file format.

### 2. heuristics

The lesson touches on computationally hard problems and heuristics but please note that **computationally hard problems and heuristics will be revisited later on**. A general "hand-wavy" understanding is all that's needed from this lesson.

We do want students to see, however, that there is no single correct way to compress text using the method we use in this lesson because a) there is no known algorithm for finding an optimal solution, and b) we don't even know a way to verify whether a given solution is optimal. There is no way to prove it or derive it beyond trying all possibilities by brute force. **This is an example of an algorithm that cannot run in a "reasonable amount of time"** - one of the CSP learning objectives.

[View on Code Studio](#)

## Objectives

**Students will be able to:**

- Collaborate with a peer to find a solution to a text compression problem using the Text Compression Widget (lossless compression scheme).
- Explain why the optimal amount of compression is impossible or "hard" to identify.
- Explain some factors that make compression challenging.
- Develop a strategy (heuristic algorithm) for compressing text.
- Describe the purpose and rationale for lossless compression.

## Preparation

- ☐ Test out the Text Compression Widget
- ☐ Review the teaching tips to decide which options you want to use

## Links

**Heads Up!** Please make a copy of any documents you plan to share with students.

**For the Teacher**

- **Activity Recap - Decode this Message** - Activity Recap [Make a Copy](#)

**For the Students**

- **Decode this message** - Activity Guide [Make a Copy](#)
- **Activity Guide - Text Compression** - Activity Guide [Make a Copy](#)
- **Video: Text Compression with Aloe Blacc** - Video ([download](#))
- **Activity Guide - Text Compression Heuristics** - Activity Guide [Make a Copy](#)

## Vocabulary

### 3. Foreshadowing programming behaviors

Lastly, the Text Compression Activity is an **important lesson to refer back to when students start programming**. The activity engages students in thinking and problem solving behaviors that foreshadow skills that are particularly useful for programming later down the line. In particular, when students recognize patterns that repeat, and then represent those patterns as abstract symbols, and then further recognize patterns within those patterns, it is very similar to the kinds of abstractions we develop when writing **functions and procedures when programming**. Decoding the message in the warm-up activity is very similar to tracing a sequence of function calls in a program.

## Agenda

### Getting Started (5-7 mins)

Warm up: Abbr In Ur Txt Msgs (5-7 mins)

### Activity (45 mins)

Decode this Mystery Text (10-15 mins)

Use theText Compression Widget

Discuss properties and challenges with compression.

### Activity 2 (30 mins)

Develop a heuristic for doing compression

What's best?

### Wrap-up (20 mins)

Recap Questions

Compression in the Real World (.zip)

### Assessment

### Extended Learning

- **Heuristic** - a problem solving approach (algorithm) to find a satisfactory solution where finding an optimal or exact solution is impractical or impossible.
- **Lossless Compression** - a data compression algorithm that allows the original data to be perfectly reconstructed from the compressed data.

# Teaching Guide

## Getting Started (5-7 mins)

### Warm up: Abbr In Ur Txt Msgs (5-7 mins)

Prompt:

- "When you send text messages to a friend, do you spell every word correctly?"
  - Do you use abbreviations for common words? List as many as you can.
  - Write some examples of things you might see in a text message that are not proper English.

Give students a minute to write, and to share with a neighbor?

- "Why do you use these abbreviations? What is the benefit?"
  - Possible answers:
    - to save characters/keystrokes
    - to hide from parents/teachers
    - to be cool, clever, funny
    - to "speak in code"
    - to say the same thing in less space

### What's this about? - Compression: Same Data, Fewer Bits

- Today's class is about **compression**
- When you abbreviate or use coded language to shorten the original text, you are "compressing text." Computers do this too, **in order to save time and space**.
- The art and science of compression is about figuring out how to represent the SAME DATA with FEWER BITS.
- Why is this important? One reason is that storage space is limited and you'd always prefer to use fewer bits if you could. A much more compelling reason is that there is an upper limit to how fast bits can be transmitted over the Internet.
- What if we need to send a large amount of text faster over the Internet, but we've reached the physical limit of how fast we can send bits? Our only choice is to somehow capture the same information with fewer bits; we call this **compression**.

Transition:

Let's look at an example of a text message that's been compressed in a clever way.

## Activity (45 mins)

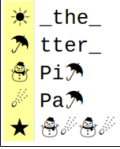
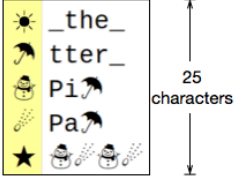
### Decode this Mystery Text (10-15 mins)

- **Distribute or Display** the Activity guide: **Decode this message - Activity Guide**
- Put students into partners or work individually.
- **Task:** What was the original text?
- Give students a few minutes to decode the text. The text should be a short poem (see activity recap below)

#### Discussion

As a warm up to thinking about Text Compression, connect to ways that most people already compress text in their lives, through abbreviations and acronyms with which most people have some experience in text messages.

Motivate some ideas about why someone would want to compress text.

Student Activity Guide	Activity Recap
<p><b>Distribute or Display</b> Activity guide: <b>Decode this message - Activity Guide</b></p>	<p>(Display or draw yourself) Activity Recap: <b>Activity Recap - Decode this Message - Activity Recap</b></p>
<p>Encoded Message: ★listen_to★rain_★on★window_pane</p> <p>Original Message:</p> <p>Key:</p> 	<p><b>How Much?</b> 40% compressed!</p> <p>Original: 93 characters Compressed: 56 characters Difference: 37 characters (~40%)</p> <p><b>Original Message</b> 93 characters</p> <p>Pitter_patter_pitter_patter_listen_to_the_rain_pitter_patter_pitter_patter_on_the_window_pane</p> <p><b>Compressed</b> 56 characters</p> <p>★listen_to★rain_★on★window_pane</p> <p>31 characters</p> <p>Total number of characters needed to represent compressed version is:</p> <p>31 (message) + 25 (key) = 56</p> 

### Recap: How much was it compressed?

To answer, we need to compare the number of characters in the original poem to the number of characters needed to represent the compressed version.

Let's break it down.




- **Display** or **Demonstrate** yourself ideas from: **Activity Recap - Decode this Message - Activity Recap** (shown in table above)
- **Important Note:**
  - The compressed poem is not **just** this part: ★listen\_to★rain\_★on★window\_pane If you were to send this to someone over the Internet they would not be able to decode it.
  - The full compressed text includes BOTH the compressed text **and** the key to solve it.
  - Thus, you must account for the total number of characters in the message **plus** the total number of characters in the key to see how much you've compressed it over the original.

### Transition

Now you're going to get to try your hand at compressing some things on your own.

## Use theText Compression Widget

### Code Studio levels

Lesson Overview 	Student Overview
Text Compression 	Student Overview
Widget: Text Compression 	Student Overview

# Check Your Understanding

4

5

6

7

(click tabs to see student view)



## Video: Text Compression with Aloe Blacc -

### Video

- Video explains compression
- Demonstrates the use of the Text Compression Tool.
- NOTE: This video pops up automatically when students visit the text compression stage in Code Studio.



- Divide students into groups of 2
- Assign each pair one of the poems provided and challenge them, as a pair to compress their poem as much as possible.
- Deliver or put simple instructions on the board so students can follow.
  - **Challenge:** compress your assigned poem as much as possible.
  - Compare with other groups to see if you can do better.
  - Try to develop a general strategy that will lead to a good compression.
- After some time, have pairs that did the same poem get together to compare schemes. As a group their job is to come up with the best compression for that poem for the class.
- Optionally: you may hand out **Activity Guide - Text Compression - Activity Guide** and have students complete it individually. It may work well as an out-of-class activity or assessment.

## Discuss properties and challenges with compression.

Ask groups to pause to discuss the questions at the end of the activity.

### Prompts:

- **"What makes doing this compression hard?"**
  - Invite responses. Some of these issues should surface: You can start in lots of different ways. Early choices affect later ones. Once you find one set of patterns, others emerge.
  - There is a tipping point: you might be making progress compressing, but at some point the scale tips and the dictionary starts to get so big that you lose the benefit of having it. But then you might start re-thinking the dictionary to tweak some bits out.
- **"Do we think that these compression amounts that we've found are the the best? Is there a way to know what the best compression is?"**
  - We probably don't know what's best.
  - There are so many possibilities it's hard to know. It turns out the only way to guarantee perfect compression is brute



### Content Corner

The video explains a little bit about compression in general - the difference between lossless compression and lossy compression. Today's class is about lossless compression we'll do lossy compression in a class or two after looking at image encoding.

### Teaching Tip

**Teacher's Choice** whether to show the video to the whole class or let students watch it from within Code Studio. There are benefits and drawbacks to each.

**Option to Consider:** Get students into the text compression tool BEFORE showing the video. You might find students are more receptive to some of the information in the video if they have tried to use the tool first.

**Communication and Collaboration:** To develop communication and collaboration between students, include one of the following scenarios in class:

- Have students who were assigned the same poem compare results, or seat them in the same area of the room.
- Have a little friendly competition - but be careful not to let "bad" competition seep in - to see which pair can compress a poem the most. Use a poem that none of the students have compressed yet.
- For each poem, have the group(s) who did it figure out the best in the class, and record it on the board or somewhere that people can see.
  - Have a class goal of getting the compression percentages for the four poems as high as possible.
  - The groups with the best compression percentages may be asked to share their strategy with the class.

Students may be reluctant to share if they feel they don't have the best results, but students should see others' work and offer advice and strategies.

force. This means trying every possible set of substitutions. Even for small texts this will take far too long. The “best” is really just the best we’ve found so far.

- **“But is there a process a person can follow to find the best (or a pretty good) compression for a piece of text?”**
  - Yes, but it’s imprecise -- you might leave this as a lingering question that leads to the next student task.

## Activity 2 (30 mins)

### Develop a heuristic for doing compression

#### Distribute or Display: Activity Guide - Text Compression Heuristics - Activity Guide

In computer science there is a word for strategies to use when you’re not sure what the exact or best solution to a problem is.

**Vocabulary:** heuristic a problem solving approach (typically an algorithm) to find a satisfactory solution where finding an optimal or exact solution is impractical or impossible.

#### Instructions:

- Continue working on compressing your poem using the Text Compression Widget. As you do so, develop a set of rules, or a “heuristic” that generally seems to provide good results.
- **Record your heuristic** as a list of steps that someone else unfamiliar with the problem could follow and still end up with decent compression.
- **Trade your heuristics with another group.** Are they clear and specific enough that you always know what to do? If not, provide feedback to one another and improve your heuristics to provide clearer instructions.
- Using another group’s heuristic, attempt to compress one or more of the poems in the tool. Record the amount of compression you achieve.

#### Teaching Tip

You may elect to not do this heuristic activity and instead get the key take-aways (see Activity Goal below) across through discussion following the previous activity.

#### Activity

The point here is to establish:

- There is no real way to determine for sure that you’ve got the **best** compression besides trying everything possible by brute force.
- Heuristics are techniques for at least making progress toward a “good enough” solution.
- Following the same heuristic might lead to different results.

### What’s best?

#### Share Findings:

Have one member of each group give a summary of their heuristic and the results on each of the poems. If time is limited, these presentations can be done between groups instead in front of the entire class. The discussion questions below could also be done group to group.

#### Reflection Prompts (from the Activity Guide)

- **“Do you think it’s possible to describe (or write) a specific set of instructions that a person could follow that would always result in better text compression than your heuristic? Why or why not?”**
  - Some compression programs (like zip) do a great job if the file is sufficiently large and has reasonable amounts of repetition.
  - However, it is also possible to create a “compressed file” that is larger than the original because the heuristic does work in every single case.
- **“Is there a way to know that a compressed piece of text is compressed the most possible? If yes, describe how you could determine it. If no, why not?”**

- Stress that there is no perfect solution.
- The size and shape of the data will determine what the “best” answer is and we often cannot even be sure it is the best answer (only that it is better than other answers we have tried.)

## Wrap-up (20 mins)

### Recap Questions

**“What did all groups’ processes for compression have in common?”**

- Pattern Recognition
- Abstraction (patterns referring to other patterns)

**“Will following this process always lead to the same compression? (i.e. two people following the process for the same poem, will result in the same compression?)”**

- No. It’s imprecise, but still OK. The text still gets compressed, no matter what.
- Since there is no way to know what’s best, all we need is a process that comes up with some solution, and a way to make progress.

Terminology: Verify students know or use an **\*exit ticket** on this vocabulary:

- lossless compression v. lossy compression
- heuristic

## Compression in the Real World (.zip)

### Zip Compression

- There is a compression algorithm called **LZW compression** upon which the common “zip” utility is based. Zip compression does something very similar to what you did today with the text compression widget.
- Here is an animation of **lzw in action**. You can see the algorithm doesn’t compress it the most, but it is following a heuristic that will lead to better and better compression over time.
- Do you want to use zip compression for real? Most computers have it built in:
  - **Windows:** select a file or group of files, right-click, and choose “Send To...Compressed (zipped) Folder.”
  - **Mac:** select a file or group of files, ctrl+click, and choose “Compress Items.”
- Warning: if you try this results may vary.
  - Zip works **really** well for text, but only on large files. If you try to compress the simple hello.txt file we used in a previous lesson, you’ll see the resulting file is actually **bigger**.
  - Zip is meant for text. It might not work well on non-text files very well because they are already compressed or don’t have the same kinds of embedded patterns that text documents do.

#### 💡 Teaching Tip

- You **do not** have to review or demo LZW compression in depth here. It is an interesting real-world application of the activity done in class.
- While details of LZW compression are not part of the AP course content, but the idea of lossless compression is.
- Recommendation: demonstrate zip quickly.
- Have a large text file at the ready, such as the **plaintext version of Hamlet**
- Use the .zip utility on your computer to compress into a zip file and then compare the file size to the original. (We learned how to do this in the previous lesson).

## Assessment

**Code Studio:** Assessment questions are available on the Code Studio



# Extended Learning

## Real World: Zip Compression

- Experiment with zip using text files with different contents. Are the results for small files as good as for large files? (On Macs, in the Finder choose “get info” for a file to see the actual number of bytes in the file, since the Finder display will show 4KB for any file that’s less than that.)
  - Warning: results may vary. Zip works really well for text, but it might not compress other files very well because they are already compressed or don’t have the same kinds of embedded patterns that text documents do.

## Challenge: Research the LZW algorithm

- .zip compression is based on the **LZW Compression Scheme**
- While the idea behind the text compression tool is similar to LZW (zip) algorithm, tracing the path of compression and decompression is somewhat challenging. Learning more about LZW and what happens in the course of this algorithm would be an excellent extension project for some individuals.

# Standards Alignment

## CSTA K-12 Computer Science Standards (2011)

- ▶ **CL** - Collaboration
- ▶ **CPP** - Computing Practice & Programming
- ▶ **CT** - Computational Thinking

## Computer Science Principles

- ▶ **2.1** - A variety of abstractions built upon binary sequences can be used to represent all digital data.
- ▶ **2.2** - Multiple levels of abstraction are used to write programs or create other computational artifacts
- ▶ **3.1** - People use computer programs to process information to gain insight and knowledge.
- ▶ **3.3** - There are trade offs when representing information as digital data.
- ▶ **4.2** - Algorithms can solve many but not all computational problems.

## CSTA K-12 Computer Science Standards (2017)

- ▶ **DA** - Data & Analysis



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

# Lesson 3: Encoding B&W Images

Widget | Concept Invention | Individual Creation

## Overview

In this lesson, students will begin to explore the way digital images are encoded in binary. The class begins by asking students to invent their own image encoding protocol in order to familiarize themselves with some of the subtle complications of encoding images, namely the need for other data, called metadata, that describes properties of the image necessary for rendering it. Students will learn about pixels, raster images, and what an image file format is. Students will encode binary image data using a widget in Code Studio.

## Purpose

The main purpose of this lesson is for students to exhibit some creativity while getting some hands-on experience manipulating binary data that represents something other than plain numbers or text. Connections to abstraction in data can be made here. Connections can be made back to file sizes and file formats here as well - e.g. how many bytes does it take to store an image v. text? If you want to broach the subject, the concept of data compression can come in here too - it is interesting to think about how a black and white image might be compressed. You should be aware that this lesson largely acts a stepping stone to the **next lesson** which addresses how RGB colors are represented in binary.

Image file types have some similarities to data packets we saw in the Internet unit -- because images must include metadata, or data about the data. The data of a black-and-white image is the list of bits that represent whether each pixel is on or off. To create the image, however, we must also know how wide and tall the image is in order to recreate it accurately. This necessitates the creation of a file format which clearly defines how this metadata will be encoded, since it is crucial for interpreting the subsequent data of the image. It is similar to how an internet packet doesn't only contain the data you need to send, but must also include metadata like the to and from addresses and packet number.

Digital images can be stored in many formats, but one of the most common formats is "raster". Raster images store the image as an array of individual pixels, each of which has a particular color. Higher-quality images can be obtained by decreasing the size of the pixels (resolution). While full color will be addressed in the next lesson, an important idea here is that images on computer screens are created with **light** by illuminating pixels on the screen. This is why it is typical

[View on Code Studio](#)

## Objectives

Students will be able to:

- Explain how images are encoded with pixel data.
- Describe a pixel as an element of a digital image.
- Encode a B&W image in binary representing both the pixel data (intensity) and metadata (width, height).
- Create the necessary metadata to represent the width and height of a digital image, using a computational tool.
- Explain why image width and height are metadata for a digital image.

## Preparation

☐ (Optional) Graph or grid paper for drawing pixel images by hand

☐ **KEY B&W Pixelation Widget**

☐ **KEY Extension: Magnify an Image (optional)**

## Links

**Heads Up!** Please make a copy of any documents you plan to share with students.

**For the Teacher**

- **Teaching Tips & Tricks Video - Encoding Images** - Video ([download](#))

**For the Students**

- **B&W Pixelation Widget** - Activity Guide [Make a Copy](#)
- **Extension: Magnify an Image (optional)** - Activity Guide [Make a Copy](#)
- **B&W Pixelation Tutorial** - Video ([download](#))
- **Invent a B&W image encoding scheme** - Activity Guide [Make a Copy](#)

in a black and white image for the value 1 to represent white - it means turn the light on - and 0 represents black - light off. If you were drawing on paper you might do the inverse.

## Agenda

**Getting Started (10 mins)**

**Invent An Encoding Scheme for B&W Images**

**Activity (40 mins)**

**Video: The Pixelation widget**

**Wrap-up (10 mins)**

**Assessment**

**Extended Learning**

## Vocabulary

- **Image** - A type of data used for graphics or pictures.
- **metadata** - is data that describes other data. For example, a digital image may include metadata that describe the size of the image, number of colors, or resolution.
- **Pixel** - short for "picture element", the fundamental unit of a digital image, typically a tiny square or dot that contains a single point of color of a larger image.

# Teaching Guide

## Getting Started (10 mins)

### Remarks

Back in the Internet Unit you encoded a line-drawing image as a list of numbers that made up the coordinates of the points in the image. That works for line drawings, but how might you encode a different kind of image? Today we're going to consider how you might use bits to encode a photographic image, or if you like: how could I encode vision?

Today, we're going to start to learn about images, but we're going to start simple, with black and white images.

## Invent An Encoding Scheme for B&W Images

### • Distribute Invent a B&W image encoding scheme - Activity Guide

- Separate students into pairs, and hand each student a copy of the activity guide.
- Students should work the first two pages
- Give groups time to work

#### Discuss:

Ask students to **share-out** their file format to identify commonalities and patterns.

- As a class, address students' questions that arise from the concept invention activity.
- Use the questions below to spur conversation. If the concept of metadata, or data about data, arises naturally, then address it here.

#### Prompts:

- How have you encoded white and black portions of your image, what do 0 and 1 stand for in your encoding?
- Are your encodings flexible enough to accommodate images of any size? \* How do they accomplish this?
- Is your encoding intuitive and easy to use?
- Is your encoding efficient?

### Activity

The purpose of this little concept invention activity is to be creative and to get the mind moving. There is no exact right answer that we're going for here.

There are many clever and interesting ways this could be done. Most students will likely end up saying that each pixel should be represented with either a 0 or a 1.

But what we really want to draw out is the idea of "metadata". Simply encoding the pixel data is not enough. We also need to encode the width and height of the image, or the image could not be recreated - other than through trial and error

### Remarks

- **Vocabulary:** each little dot that makes up a picture like this is called a pixel. Where did this word pixel come from? It turns out that originally the dots were referred to as "picture elements", that got shortened to "pict-el" and eventually "pixel".

- What we've discovered is that the data for our image file must contain more than just a 0 or 1 for every pixel. It must contain other data that describes the pixel data.
- This is called metadata. In this case the metadata encodes the width and height of the image.
- We've seen forms of metadata before. For example: an internet packet. The packet contains the data that needs to be sent, but also other data like the **to** and **from** address and packet number.

### Content Corner

There is some mystery about the etymology of the word "pixel". You can read more about it on the **Wikipedia: pixel page**

## Activity (40 mins)




### Introduction

- The pixelation widget in Code Studio will allow us to play with these ideas a little more.
- This widget follows a particular encoding scheme for images that you'll have to follow.

## Video: The Pixelation widget

- Show the tutorial video: **B&W Pixelation Tutorial - Video**
- NOTE: This video pops up the first time you visit the pixelation widget in Code Studio. You might prefer to have students watch it there on their own.

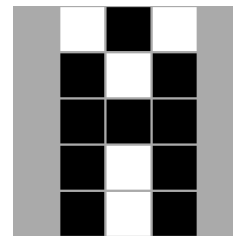
## Code Studio levels

- Levels
-  2
-  3
-  4

## Student Instructions

### Task 1: Make a 3x5 letter 'A'

Start by trying to recreate the 3x5 letter "A" depicted (at right) using the pixelation widget.



[View on Code Studio](#)



The image is initially setup with the **incorrect** dimensions. Your first task is to set the second byte to the 8-bit binary code for 5: 0000 0101. Then you can start entering pixel data to make the A.

## Student Instructions

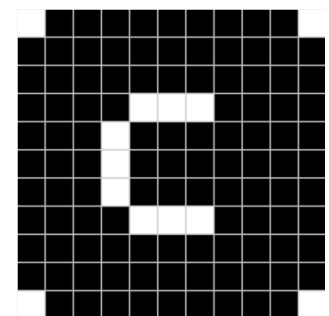
### Oh no! An image got messed up during transmission!

**The problem:** A single extra bit was inserted into the stream of bits that make up the C of the Code.org logo.

That extra bit bumps all of the other bits down the line which makes the logo look messed up.

**Your task:** Hunt down the extra bit and remove it to fix the Code.org logo.

**HINT:** One bit early on would make it look like many bits were out of order.



[View on Code Studio](#)



## Student Instructions

### Make your own image of any size



**Directions:**

- Encode an image of anything you like.
- You might want to do some planning and sketching with graph paper first.
- DO NOT simply make an abstract pattern, like a checkerboard.

[View on Code Studio](#)



- Depict something, perhaps your name written out, your initials, an icon or logo of some sort.
- Get creative! The image doesn't have to be a perfect square, it can be long and skinny.
- Optional: for fun, send your image bits to a friend using the **sending bits widget**. (note: this is just a link to the **sending formatted text level from a couple of classes ago**)

## Distribute: Activity Guide - B&W Pixelation Widget - Activity Guide

### Activity Guide

Page 1:

- Explains the encoding scheme and a bit about how the tool works.
- Describes the 3 student tasks to get familiar with the tool:
  1. Create a small image: Start by trying to recreate the 3x5 letter "A" depicted (shown above) using the pixelation widget.
  2. Correct an error: Oh no! An extra bit was inserted into an image during transmission! Track it down.
  3. Make your own image of any size of anything you like.

The second page asks students to:

- Copy/paste a copy of their personal creation
- Copy/paste the bits that are used to encode it
- Written reflection questions:
  - What are the largest dimensions (width and height) of an image we can make with the pixelation widget?
  - How many total bits would there be in the largest possible image we could make with the pixelation widget?
  - How many bits would it take to represent the smallest possible image (i.e. an image with one pixel)?
  - What would happen if we didn't include width and height bits in our protocol? Assume your friend just sent you 32 bits of pixel data (just the 0s and 1s for black and white pixels). Could you recover the original image? If so, how?

#### Teaching Tip

- You may not need or want to use the first page of the activity guide. It is a reference for students, but the tasks for students are given in Code Studio.
- Similarly for the second page, if you don't intend to collect it for assessment purposes, you can use the questions as group discussion or wrap-up questions.

## Wrap-up (10 mins)

### Review:

- The image file protocol we used contains "metadata": the width and height. **Metadata is "data about the data"** that might be required to encode or decode the bits.
- For example, you couldn't render the B&W image properly without somehow including the dimensions. Prompts:
  - What other examples of metadata have we seen in the course so far?
  - What other types of data might we want to send that would require metadata?

### (Optional) Prompt:

**"Did you think about compression at all while doing this exercise? Can you think of a way that you might represent an image of pixel data with fewer bits? What would have to change about the encoding strategy?"**

- For an answer to this see the **"Color by Numbers"** Activity from CS Unplugged (csunplugged.org).
- It uses something called "run-length encoding"

## Assessment

Check students responses on: **B&W Pixelation Widget - Activity Guide**

- Check to make sure that the bits they submitted actually produce the image as claimed.
- Score the digital artifact as you see fit, with points for creativity and perceived effort.
- The following questions can be found in the Activity Guide and also appear on Code Studio
- Answers these questions can be found here:**KEY B&W Pixelation Widget**
  - Using the B&W file format from the pixelation widget
    - What are the largest dimensions (width and height) of an image we can make with the pixelation widget?
    - How many total bits would there be in the largest possible image we could make with the pixelation widget?
    - How many bits would it take to represent the smallest possible image (i.e. an image with one pixel)?
  - What would happen if we didn't include width and height bits in our protocol? Assume your friend just sent you 32 bits of pixel data (just the 0s and 1s for black and white pixels). Could you recover the original image? If so, how?

## Extended Learning

- Check out the "**Color by Numbers**" from CS Unplugged (csunplugged.org) which uses a different clever encoding scheme for B&W images.
- ◦ Do the **Extension: Magnify an Image (optional) - Activity Guide** activity (double the size of an image on the Pixelation Tool).
- Have students research raster graphics in anticipation of the subsequent lesson.
- Attempting to communicate with possible intelligent life beyond our solar system has been a dream for humans and the goal of scientists for many years. Questions about messages to send, as well as how to send messages deep into space to unknown recipients have been debated. In 1974, scientists sent the Arecibo message to the star cluster M13 some 25,000 light years away. Read about the message they sent using 1,679 binary digits ([https://en.m.wikipedia.org/wiki/Arecibo\\_message](https://en.m.wikipedia.org/wiki/Arecibo_message)).
  - How would you change the content of the message? What would you delete and add? Why would your change be significant in a communication to other intelligent beings?
  - Sketch the segment of the design you would alter. Remember, you must retain the original number of bits.
  - List the details in this article that you understand more deeply because of what you have learned in this class up to this point.

## Standards Alignment

### CSTA K-12 Computer Science Standards (2011)

- ▶ **CL** - Collaboration
- ▶ **CPP** - Computing Practice & Programming
- ▶ **CT** - Computational Thinking

### Computer Science Principles

- ▶ **1.1** - Creative development can be an essential process for creating computational artifacts.
- ▶ **1.2** - Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem.
- ▶ **1.3** - Computing can extend traditional forms of human expression and experience.
- ▶ **2.1** - A variety of abstractions built upon binary sequences can be used to represent all digital data.
- ▶ **2.3** - Models and simulations use abstraction to generate new understanding and knowledge.
- ▶ **3.1** - People use computer programs to process information to gain insight and knowledge.
- ▶ **3.2** - Computing facilitates exploration and the discovery of connections in information.
- ▶ **3.3** - There are trade offs when representing information as digital data.

### CSTA K-12 Computer Science Standards (2017)

- ▶ **CS** - Computing Systems
- ▶ **DA** - Data & Analysis



This curriculum is available under a  
Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.



# Lesson 4: Encoding Color Images

## Overview

In this lesson students are asked to consider how color is represented on a computer and to imagine how it might be encoded in binary. Students then learn about how color is actually represented on a computer - using the RGB color scheme - and create their own images in an new version of the pixelation widget that allows you use more than 1 bit per pixel to represent color information. After grappling with the prospect of possibly many bits just to represent a single pixel, students are shown how using hexadecimal allows us to represent many bits with fewer characters. Students use a new version of the pixelation tool to encode an image with color and create a personal favicon.

## Purpose

The main purpose here, similar to the B&W pixelation activity is for students to get hands-on and "down and dirty" with bits. A major outcome will also be understanding the relationship between hexadecimal (base-16) and binary (base-2), and how useful it is to use hex to represent groups of 4 bits. It's important to realize that using hex is not a form of data compression, it's simply a different view into the bits.

The most common color representation scheme - RGB - typically uses 24 bits (3 bytes) with 8 bits each for Red, Green and Blue intensities. And one of the most common ways you see these colors represented is in hexadecimal. The pixelation widget, with its ability to choose how many bits represent the color value for each pixel, can be a very useful tool for showing the utility of hex representations for bits.

The process of rendering color on a computer screen by mixing red, green and blue light is an important concept of this lesson. The results are not always intuitive, because mixing pigment and mixing colored lights (like what's on a computer screen) lead to different results.

Another important objective of this lesson is to understand how (uncompressed) image file sizes can become quite large. For example, even a relatively small image of 250x250 pixels is a total of 62,500 pixels, each requiring up to three bytes (24 bits) or color information, resulting in a total of 1.5 million bits to store one image! Thus, interesting connections to compression **can** be made here, but note that lossy compression and image formats like .jpg are covered in the next lesson.

## Agenda

Getting Started (5 mins)

[View on Code Studio](#)

## Objectives

**Students will be able to:**

- Use the Pixelation Tool to encode small color images with varying bits-per-pixel settings.
- Explain the color encoding scheme for digital images.
- Use the Pixelation Tool to encode an image of the student's design.
- Explain the benefits of using hexadecimal numbers for representing long streams of bits.

## Preparation

☐ (Optional) Consider demonstrating the color pixelation widget instead of showing the video.

☐ **KEY Encoding Color Images**

☐ **KEY Video Guide "A Little Bit About Pixels"**

☐ **KEY Hexadecimal Numbers (optional)**

## Links

**Heads Up!** Please make a copy of any documents you plan to share with students.

**For the Teacher**

- **Teaching Tips & Tricks Video - Encoding Images** - Video ([download](#))

**For the Students**

- **A Little Bit about Pixels** - Video ([download](#))
- **Worksheet - Video Guide for "A Little Bit about Pixels"** (optional) - Worksheet ([Make a Copy](#))
- **Encoding Color Images** - Activity Guide ([Make a Copy](#))
- **Hexadecimal Numbers (optional)** - Activity Guide ([Make a Copy](#))
- **Personal Favicon Project** - Activity Guide ([Make a Copy](#))
- **Rubric - Personal Favicon Project** -

**Prompt: How might you encode colors?**

**Activity (40 mins)**

**Video: A Little Bit about Pixels**  
**Color Pixelation Widget**

**Activity 2 (30-40 mins)**

**Personal Favicon Project**

**Wrap-up**

**Submit Favicon**  
**Gallery Walk**

**Assessment**

**Extended Learning**

Rubric **Make a Copy** ▾

## Vocabulary

- **Hexadecimal** - A base-16 number system that uses sixteen distinct symbols 0-9 and A-F to represent numbers from 0 to 15.
- **Pixel** - short for "picture element", the fundamental unit of a digital image, typically a tiny square or dot that contains a single point of color of a larger image.
- **RGB** - the RGB color model uses varying intensities of (R)ed, (G)reen, and (B)lue light are added together in to reproduce a broad array of colors.

# Teaching Guide

## Getting Started (5 mins)

### Prompt: How might you encode colors?

Use a **getting started strategy** to address these questions (for ideas consult: **Teaching Strategies for the CS Classroom - Resource**)

- In the previous lesson we came up with a simple encoding scheme for B&W images. What if we wanted to have color?
- Devise an encoding scheme for color in an image file. How would you represent color for each pixel?
- How many different colors could you represent? Do you have a particular order to the colors?

#### Pair and share ideas

- Discuss some of the difficulties of representing color
- Compare and contrast the different schemes students come up with.

#### Discussion

It is likely that many students will come up with an idea like making a list of colors and just assigning a number to each one. That is fine and reasonable.

Some students may already be aware of a numeric RGB color scheme. If they can describe that here, that is fine as well.

Regardless of their encoding, students should be thinking about the number of bits they will allocate to the encoding and how that will affect the number of colors that can be encoded.

## Activity (40 mins)

### Remarks

The way color is represented in a computer is different from the ways we represented text or numbers. With text, we just made a list of characters and assigned a number to each one. As you are about to see, with color, we actually use binary to encode the physical phenomenon of LIGHT. You saw this a little bit in the previous lesson, but today we will see how to make colors by mixing different amounts of colored light.

### Video: A Little Bit about Pixels

- Show the video: **A Little Bit about Pixels - Video**
- Kevin Systrom, founder of Instagram, explains pixels and RGB color.
- (Optional) complete the video worksheet: **Video Guide KEY for "A Little Bit about Pixels" - Answer Key**

#### Discuss:

Following the video, you might address any questions (or give students time to complete the video worksheet)

Important ideas from this video include:

- Image sharing services are a universal and powerful way of communicating all over the world.
- Digital images are just data (lots of data) composed of layers of abstraction: pixels, RGB, binary.
- The RGB color scheme is composed of red, green, and blue components that have a range of intensities from 0 to 255.
- Screen resolution is the number of pixels and how they are arranged vertically and horizontally, and density is the number of pixels per a given area.
- Digital photo filters are not magic! Math is applied to RGB values to create new ones.



### Color Pixelation Widget

- Distribute the Activity Guide: **Encoding Color Images - Activity Guide**
- Direct students to work in Code Studio.
- There are 3 tutorial videos that appear in Code Studio that guide students through using the widget.
- This activity guides students through a few levels to get used to representing pixel data with more than one bit per pixel.

It works up to full 24-bit RGB color and will present hexadecimal as a convenient way to represent binary information for humans to read.

### **Guide: Encoding Color Images**

Each of the items below are presented to students on the activity guide and in Code Studio.

#### **Step 1: 3-bit color**



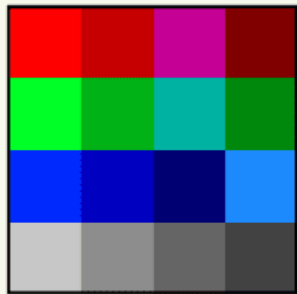
- **Color Pixelation widget tutorial video - Part 1 - Video** : How to use the pixelation widget to control color.
- Task 1: Fill in the last two pixels with the missing colors

#### **Step 2: 6-bit color**



- **Color Pixelation widget tutorial video - Part 2 - Video** : more bits per pixel for more colors
- Task 2: Experiment with 6-bit color

#### **Step 3: 12-bit color and Hex**



- **Color Pixelation widget tutorial video - Part 3 - Video** : Using hex to type bits more quickly
- Task 3: Experiment with Hex

#### **Teaching Tip**

If you are comfortable **you might consider demonstrating the pixelation tool** for each of the 3 steps in the activity guide rather than having students watch the tutorial video. Demonstrating might be a more efficient and interactive/engaging way bring students through each step.

## **Activity 2 (30-40 mins)**

### **Personal Favicon Project**

Students will create a 16 by 16 pixel personal favicon in RGB color using the Pixelation Tool. This project will likely require some time to complete, and should serve as a practice with hexadecimal numbers, metadata, and the underlying encoding of images in a raster file.

- Distribute the Activity Guide: **Personal Favicon Project - Activity Guide** and review the criteria for the project.
- Students will need a decent amount of work time to create their favicon. You might get them started in class and then assign it as homework.

### **Personal Favicon**

**(From the activity guide)**

#### **Directions**

- Create a personal 16x16 favicon and encode it using the Pixelation Widget on the final level of this lesson in Code Studio.



- The image you make should



represent your personality in some distinctive way. You will be using this favicon in future lessons and web sites that you make, so be creative and thoughtful.

- After you have finished your favicon, share it with others in the class by sending them the bits with the Internet Simulator Widget!

### Requirements

- The icon must be 16x16 pixels.
- You must use the Pixelation Widget to encode the bits of color information.
- The image must be encoded with at least 12 bits per pixel.

### Things to think about

- A simple design with a few basic colors is probably the best solution. How could you use more colors?
- Plan ahead: Sketch your design before starting to encode the bits. You might want to use a tool to help you draw small images. Suggestions:
  - Favicon Maker: <http://www.favicon.cc/>
  - Make Pixel Art: <http://makepixelart.com/free/>

## Wrap-up

### Submit Favicon

You should ask students to submit a .png version of their favicon, blown up to a larger size. And ask them to send you the bits that made up the image.

### Gallery Walk

- With the images you can make a class favicon “quilt” by printing them out.
- And you can copy/paste the bits into the pixelation tool to verify that image is correct.

## Assessment

### Content Corner

#### RGB color model - Additive Light

Computer screens emit light, so when you mix RGB colors, you are really mixing light together. This is counterintuitive for many students who have grown up mixing paints in school. When you mix paint it absorbs light.

It is illustrative to look at how you make black and white with paint vs. light:

- To make black: with paint, mix a full spectrum of colors together; with light, turn off all the lights.
- To make white: with paint, don't use any paint (assuming canvas is white); with light, turn on all lights for a full spectrum of color.

This can make mixing colors a little bizarre too:

- With paint, mix full red and full blue to make Purple
- With light, mix full red and full blue to make Pink

The Pixelation Tool is in RGB mode, as long as the number of bits per pixel is a multiple of 3 (3, 6, 9, 12, etc.) This allows for the same number of bits to be allocated to each color channel. Other bits-per-pixel settings will set the image to grayscale, with more bits allowing finer control over the shade of gray.

#### Hexadecimal Numbers:

When working through the Activity Guide for the color version of the Pixelation Tool, students will be introduced to the concept of hexadecimal numbers, so-called because there are 16 unique symbols that can appear in each place value, 0-9, A, B, C, D, E, and F.

#### MISCONCEPTION ALERT

**It is important to note that hexadecimal numbers are used to aid humans in reading longer strings of bits, but they in no way change the underlying data being represented. Instead, they allow us to read 4 bits at a time rather than 1, and so allow us to more easily parse binary information. Hexadecimal representation is NOT a form of compression, since the underlying binary representation is not changing at all. Rather it is a more convenient way of representing that binary information when humans need to read and interact with it.**

You may wish to separately address this topic as a class. Students can practice with the **Hexadecimal Odometer** and can complete this **Hexadecimal Numbers (optional) - Activity Guide** if you deem more practice necessary.

### Questions:

- How many bits (or bytes) are required to encode an image that is 25 pixels wide and 50 pixels tall, if you encode it with 24 bits per pixel?
  - To help students understand how quickly the bit size of images expands as the image is enlarged, start with smaller numbers (5 X 10) and then incrementally increase the width and height to illustrate the concept.
- Imagine that you have an image that is too dark or too bright. Describe how you would alter the RGB settings to brighten or darken it. Give an example.

## Extended Learning

- If you had to send your favicon using the sending bits widget, it would probably take a long time. Could you compress your image? How? Describe in broad strokes the kinds of things you could do.
- Read **Blown to Bits (www.bitsbook.com), Chapter 3**, Ghosts in the Machine, pp. 95-99 (Hiding Information in Images), then answer the following questions:
  - Besides hiding information sent to others, what other uses can steganography have for everyday users? For example, what uses would steganography have for an American businessman in China?

## Standards Alignment

### CSTA K-12 Computer Science Standards (2011)

- ▶ **CL** - Collaboration
- ▶ **CPP** - Computing Practice & Programming
- ▶ **CT** - Computational Thinking

### Computer Science Principles

- ▶ **1.1** - Creative development can be an essential process for creating computational artifacts.
- ▶ **1.2** - Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem.
- ▶ **1.3** - Computing can extend traditional forms of human expression and experience.
- ▶ **2.1** - A variety of abstractions built upon binary sequences can be used to represent all digital data.
- ▶ **2.2** - Multiple levels of abstraction are used to write programs or create other computational artifacts
- ▶ **2.3** - Models and simulations use abstraction to generate new understanding and knowledge.
- ▶ **3.1** - People use computer programs to process information to gain insight and knowledge.
- ▶ **3.2** - Computing facilitates exploration and the discovery of connections in information.
- ▶ **3.3** - There are trade offs when representing information as digital data.

### CSTA K-12 Computer Science Standards (2017)

- ▶ **CS** - Computing Systems
- ▶ **DA** - Data & Analysis



This curriculum is available under a  
Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

# Lesson 5: Lossy vs. Lossless Compression

## Research

### Overview

Students learn the difference between lossy and lossless compression by experimenting with a simple lossy compression widget for compressing text. Students then research three real-world compressed file formats to fill in a research guide. Throughout the process they review the skills and strategies used to research computer science topics online, in particular to cope with situations when they don't have the background to fully understand everything they're reading (a common situation even for experienced CS students).

### Purpose

The first goal of this lesson is straightforward: understand what lossy compression is and when/why it might be used. Students should see a number of examples of this distinction throughout the lesson and should leave the lesson being able to describe the relative benefits of each.

The second goal of this lesson is to build up students' research skills both for the project they will complete in the next lesson and for the Explore PT at the end of the year. Students will need practice finding reliable sources, reading technical articles, and synthesizing information. The teacher's role in calling out the skills being used, not merely the facts being found, is significant.

### Agenda

#### Getting Started (15 mins)

**Quick Discovery: Lossy Text Compression  
Lossless vs. Lossy Compression**

#### Activity (40 mins)

#### Wrap-up (5 mins)

## View on Code Studio

### Objectives

#### Students will be able to:

- Explain the difference between lossy and lossless compression.
- Explain the relative benefits or drawbacks of different file formats, particularly in terms of how they compress information.
- Identify reliable sources of information when doing research
- Explain the difference between open source and licensed software.

### Preparation

☐ Prepare print or digital copies of File Formats and Compression Activity Guide

☐ **KEY File Formats and Compression**

### Links

**Heads Up!** Please make a copy of any documents you plan to share with students.

#### For the Students

- **Lossy Text Compression App** - App Lab
- **File Formats and Compression** - Activity Guide [Make a Copy](#)

### Vocabulary

- **Lossless Compression** - a data compression algorithm that allows the original data to be perfectly reconstructed from the compressed data.
- **Lossy Compression** - (or irreversible compression) a data compression method that uses inexact approximations, discarding some data to represent the content. Most commonly seen in image formats like .jpg.



# Teaching Guide

## Getting Started (15 mins)

### Quick Discovery: Lossy Text Compression

**Prompt:** With a partner, go to the **Lossy Text Compression App - App Lab**. Use it for a couple minutes, discuss with your partner what's happening, and then answer the following questions.

- Should this “count” as text compression? Why or why not?
- What do you think the word “lossy” refers to?

🗨️ **Discuss:** Give pairs of students a couple minutes to explore the app. Then ask discuss what they are seeing. Finally ask them to write down brief responses to the two prompts. Once all pairs have had a chance to write responses have a few groups share opinions with the groups.

### Lossless vs. Lossy Compression

#### 🗣️ Remarks

This is an example of compression since the total number of bits needed to represent the message is reduced. As we've seen, though, it's different than the compression we saw with the text compression widget.

The text compression widget uses “lossless” compression. This means that it is possible to reverse the compression and recover the original information (message) in its entirety. This is what the dictionary was for.

The text compression we just saw is “lossy” compression. This means that it isn't possible to perfectly reverse the compression and recover the original (message). Some of it is lost forever. If you saw the word “fd” it could be “food”, “feed”, “feud”, or “fad”. You might be able to guess what it was supposed to be, but there's no algorithm that will always give you the original message.

**Vocabulary:** Display these definitions

- **Lossless Compression:** a data compression algorithm that allows the original data to be perfectly reconstructed from the compressed data.
- **Lossy Compression:** (or irreversible compression) a data compression method that uses inexact approximations, discarding some data to represent the content. Most commonly seen in image formats like .jpg.

🗨️ **Prompt:** Have students briefly discuss the prompt below with a neighbor.

- When you use lossy compression you lose the ability to decompress your information and get back a perfect copy. Even so, people use lossy compression all the time. Can you think of reasons or situations where someone would still use lossy compression?

**Discuss:** Have pairs briefly discuss with one another. Circulate and listen to ideas as they're discussed. Then have a couple groups share with the whole room.

**Transition:** We've been looking at image file formats. And we've also seen text compression. Both of those attempted to render perfectly every piece of information.

#### Discussion

**Goal:** As you circulate first make sure that all groups try typing their own text into the app. They should see that the app keeps the first character of every word but then removes all vowels.

During the share out aim to hear from multiple groups. If you ask leading questions you might direct the conversation towards the following points.

- The text is certainly “smaller” than before so at least it might be compression.
- This is different from what they saw with the text compression widget though. Some information is being entirely thrown away and if you only had the compressed text there's no way to get back the original text.
- That said, you can usually still read the text. The meaning hasn't been lost even if some parts of the original message could never be perfectly recovered.
- The word “lossy” probably has something to do with the fact that some letters (information) are entirely lost here.

Your goal in this discussion isn't that students arrive at a definition. You're aiming for all students to see that this type of compression is different from what they've seen before and prep them all for the following remarks.



Both the image file format and the text compression scheme we used were lossless. Lossy compression schemes usually take advantage of the fact that a human is supposed to interpret the data at the other end, and human brains are good at filling the gaps when information is missing.

## Activity (40 mins)

**Group:** Place students in pairs. Each group will only need a single computer.

**Distribute: File Formats and Compression - Activity Guide** to each group.


### *Rapid Research - Compression Formats*

**JPG:** Ask all groups to research the answers to the first column in the activity guide together. In particular remind them to keep track of their sources of information.

**Share:** Have pairs share their answers with another pair. If there are disagreements have them discuss with one another the sources of their information and try to arrive at a shared solution.

**Discuss:** Share the solution to the first column as a class. Resolve any discrepancies. Afterwards briefly acknowledge the following points.

- JPG is lossy compression. Many images on the web are converted to JPG. Odds are if you've seen a grainy or pixelated image on the Internet, it's a JPG.
- Lossy compression like this results in a lower quality image with fewer details, but as humans we can still tell what's in the picture. Especially in instances where you don't need a high quality image and bandwidth is limited JPG makes a lot of sense.
- This is a free and open format, but even so there have been disputes about its ownership.

 **Prompt:** Let's think for a minute about how you are doing your research. What kinds of sources are you finding? How are you actually reading these sources?

**Discuss:** Have students briefly share their thoughts at their tables before discussing as a class. Use the discussion to lead into to the comments below.

### *Remarks*

Conducting research online about CS topics is a skill. Often it means hunting through articles you don't completely understand for the key pieces of information that you do. Other subjects like history or math may have classic agreed upon texts but in the world of CS you'll often end up on Wikipedia or online forums. This is ok. We're going to keep working on this skill of reading technical articles. If you sometimes are confused that's ok and entirely normal. There is no one on earth who understands everything about CS with how large a subject it is and how quickly it's changing. When you're doing research as a computer scientist it usually means sticking with it even if a lot of the content doesn't make sense at first.

**MP3 and PNG:** Have pairs fill in the MP3 and PNG columns of the table.

**Share:** Once both columns have been filled out have pairs share their results across the table again.

**Discuss:** Share out the results of the research with the whole class, reviewing the results.

### Discussion

**Goal:** This prompt serves two roles. It first should get students a chance to practice using the vocabulary you just introduced. Use their responses to judge whether students understand the distinction between the two types of compression. If necessary review the two before continuing to the main activity.

Secondly it prompts students to begin thinking critically about the tradeoffs of the two compression types which they will be doing for the remainder of the lesson.

### Content Corner

**Formats in Everyday Life:** The file extension you often see on a file (for example: myPhoto.jpg) is really just an indicator to the computer of how the underlying bits are organized, so the computer can interpret them. If you change the name of the file to myPhoto.gif, that **does not** magically change the underlying bits; all you've done is confuse the computer. It won't be able to open the file because it will attempt to interpret the file as a **GIF** when really the bits are in **JPG** format.

### Discussion

**Researching CS Topics:** When researching a computer science topic (e.g. as students will have to do on the Explore PT) it is important to remember that this is a unique and separate skill. It often requires students to read text that they don't completely understand or which uses advanced vocabulary. At other times they'll need to consult sources typically considered unreliable like online forums or Wikipedia. Part of the goal of this lesson is calling out the fact that this is a separate skill, practicing the individual component, and preparing students to do it more independently in the following lesson.

## 💡 Wrap-up (5 mins)

🗣️ **Prompt:** Lossy compression seems to be "worse" than lossless compression but obviously both are being used all the time. Write down three reasons or situations where someone would be willing to use lossy compression even though it means some loss of quality.

**Discuss:** Students should silently list their responses, then share with a neighbor, then finally discuss with the entire class. Have multiple pairs or tables share their responses.

### 🎤 Remarks

Today we accomplished a lot. We explored the difference between lossy and lossless compression, we practiced reading and researching a CS topic, and we learned a little bit about the complicated world of file formats! Next time we'll close out this unit by digging a little deeper on all of these topics.

### 💡 Teaching Tip

**Record Research Strategies:** Throughout this activity consider writing and recording strategies that students are using in order to conduct their research. This could be on a poster, the board, or some kind of shared digital notes. Students are going to have another chance to research formats in the following lesson, so an important goal in this lesson is highlighting how students are succeeding at doing this research, not just the information they're finding.

### 💬 Discussion

**Goal:** This discussion is primarily a way to check that students have understood the difference between lossy and lossless compression. Lossy compression is fine if you just need a "good enough" version of something and care about saving space or bandwidth. For example, if you don't want to use up the data plan on your phone you would use compressed images.

## Standards Alignment

### CSTA K-12 Computer Science Standards (2011)

- ▶ **CD** - Computers & Communication Devices
- ▶ **CL** - Collaboration
- ▶ **CT** - Computational Thinking

### Computer Science Principles

- ▶ **3.3** - There are trade offs when representing information as digital data.
- ▶ **7.3** - Computing has a global affect -- both beneficial and harmful -- on people and society.
- ▶ **7.5** - An investigative process is aided by effective organization and selection of resources. Appropriate technologies and tools facilitate the accessing of information and enable the ability to evaluate the credibility of sources.

### CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming
- ▶ **DA** - Data & Analysis



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

# Lesson 6: Rapid Research - Format Showdown

Project | Research

## Overview

In this lesson students will conduct a small amount of research to explore a file format either currently in use or from history. Students will conduct research in order to complete a "one-pager" that summarizes their findings. They will also design a computational artifact (video, audio, graphic, etc.) that succinctly summarizes the advantages of their format over other similar ones.

This lesson is intended to be a quick, short version of a performance task in which students rapidly do some research and respond in writing. It might take 2 class days but should not take more. The goal is to develop skills that students will use when they complete the actual Explore PT later in the year.

## Purpose

This lesson concludes Unit 2's investigation of file sizes, formats, metadata, and compression. The activity is designed for students to practice research skills that will be beneficial when completing the Explore PT while also applying the knowledge they have developed across the unit. The "showdown" aspect of the project is designed to both encourage students to think more deeply about the inherent tradeoffs of different file formats and also provide a motivating context in which to perform their research. The computational artifact completed in this project has a direct parallel to the computational artifact students are expected to design for the Explore PT.

## Agenda

**Warm Up (5 mins)**

**Activity (90 mins)**

**Rapid Research - Format Showdown**

**Day 1 - Choose Format, Research, Begin One-Pager**

**Day 2 - Complete One-pager and Computational Artifact**

**Wrap Up (10-60 mins)**

**Presentation (Optional):**


[View on Code Studio](#)

## Objectives

**Students will be able to:**

- Identify reliable sources of information when doing research
- Synthesize information taken from multiple online sources
- Create an artifact (video, image, slide, poster, etc.) to communicate information about a computing topic.

## Preparation

 Prepare printed / digital copies of the

**Rapid Research - Format Showdown - Activity Guide** and **Rapid Research - Format Showdown - One-pager Template** to share with students

## Links

**Heads Up!** Please make a copy of any documents you plan to share with students.


**For the Students**

- **Rapid Research - Format Showdown - Activity Guide** [Make a Copy](#)
- **Rapid Research - Format Showdown - One-pager Template** [Make a Copy](#)

# Teaching Guide

## Warm Up (5 mins)

**Prompt:** Imagine I was comparing two different file formats for representing images. All I tell you that one of them is "better" than the other. Based on everything that you've learned in this unit, what might "better" mean? Try to come up with at least three ideas and incorporate vocabulary from the unit.

 **Discuss:** Have students quietly write their responses, then share with a neighbor, then discuss with the class. Some potential responses might include:

- Better compression ratio or better quality image even when compressed
- Can hold larger or more complex types of information
- Different kinds of metadata
- Includes features not found in other formats
- Used by more people
- Open, no licensing fees
- Newer and optimized for modern technology

### Discussion

**Goal:** There's no expectation that students come up with every reason listed. The goal is to get students in the mindset of comparing formats. This discussion should also highlight that there's lots of different ways to consider one format better than the other. This is also why there's so many different formats!

### Remarks

Today we're going to dig a little deeper into this question by doing a Rapid Research project. Not only are you going to research your own file format, you're going to make the case why yours is the best!

## Activity (90 mins)

### Rapid Research - Format Showdown

**Distribute:** **Rapid Research - Format Showdown - Activity Guide** and **Rapid Research - Format Showdown - One-pager Template** and review as a class.

Below is a suggested schedule for completing the project.

### Teaching Tip

#### Day 1 - Choose Format, Research, Begin One-Pager

##### Review Activity Guide and Rubric:


At the beginning of the project, emphasize the importance of reviewing the **one-pager template** and **rubric**. Students may assume that more is required of them than is actually the case.

**Choosing Your Format:** It is recommended that you place a time limit on this process (e.g. 10 minutes). You may even consider simply assigning formats yourself.

They are all essentially equivalent in terms of difficulty. There's minimal difference in terms of "quality" so long as students take a creative angle towards the presentation. All these formats have been used by someone at some point, so their job is to find out why and creatively play up that angle.

**Differences from the actual Explore PT:** The actual Explore Performance Task will be completed over 8 class hours. The fact that this schedule is significantly shorter reflects several differences in this Practice PT.

- Students are not choosing a computing innovation by the College Board's definition (file formats and protocols do not process data, include program code, etc.)
- Students are not completing most of the prompts of the Explore PT
- Research in this project is presumed to be less detailed

 **Conduct Your Research:** This document is intended to serve primarily as a guide to students for identifying online sources of information. The skill students need to develop is identifying useful resources on their own and then synthesizing this information. Being presented with a structured way of doing this means students will have a model for how to complete their research when completing the actual Explore PT.

The "Key Information to Find" highlights specific terminology from the Explore PT that students will benefit from having seen earlier in the course.

**Begin One-Pager:** Have students begin working on filling in the details of their one-pager as they research.

## Day 2 - Complete One-pager and Computational Artifact

**Complete One-Pager:** Students should continue filling in their one-pagers. Remind students to record their sources of information as they go.

**Computational Artifact:** Decide if you will encourage students to all create separate kinds of computational artifacts or if they will do them in a unified way. You might have everyone create a slide, a short video (e.g. a 30 second "campaign ad" for their format) etc. Remind students of the requirements for this artifact (noted below):

- A list of other formats you'd specifically like to target in your computational artifact
- Three key points you'd like to make in your artifact explaining the benefits of your format

### 💡 Teaching Tip

**Reinforce Research Skills from the Previous Lesson:** As students saw in the previous lesson, performing research on computing topics is a skill unto itself. Reinforce skills reviewed in the previous lesson as students move to this part of the activity.

## Wrap Up (10-60 mins)

### 💡 Presentation (Optional):

If time allows, students may wish to have an opportunity to share their one-pagers and computational artifacts with one another. Consider other options like creating a "File Formats Hall of Fame" by posting links to all their documents in single shared document. Or even print them out and post them in the room to be reviewed in a gallery walk.

### 💡 Teaching Tip

**Timing Consideration:** Having opportunities to share work promotes a communal classroom atmosphere but be thoughtful about time. If you feel you are not pressed for time using a full class period for presentations may be a great way to end the unit, but otherwise you may be looking for a quicker way to share highlights from this project.

## Standards Alignment

### Computer Science Principles

- ▶ **1.1** - Creative development can be an essential process for creating computational artifacts.
- ▶ **1.2** - Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem.
- ▶ **2.1** - A variety of abstractions built upon binary sequences can be used to represent all digital data.
- ▶ **2.2** - Multiple levels of abstraction are used to write programs or create other computational artifacts
- ▶ **3.2** - Computing facilitates exploration and the discovery of connections in information.
- ▶ **3.3** - There are trade offs when representing information as digital data.
- ▶ **7.3** - Computing has a global affect -- both beneficial and harmful -- on people and society.
- ▶ **7.5** - An investigative process is aided by effective organization and selection of resources. Appropriate technologies and tools facilitate the accessing of information and enable the ability to evaluate the credibility of sources.

### CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming
- ▶ **DA** - Data & Analysis



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.