**Instructor: Mrs. Nichole Niebur**

Computer Science Principles (CSP) curriculum is a full-year, rigorous, entry-level course that introduces students to the foundations of modern computing. The course covers a broad range of foundational topics such as programming, algorithms, the Internet, big data, digital privacy and security, and the societal impacts of computing. The course is designed for typical school settings with teachers in classrooms. In addition to meeting UNO disciplinary and course outcomes, this course also meets the requirements of the Advanced Placement (AP) curriculum. All teacher and student materials are provided for free online.

**Curriculum Overview and Goals**

Computing affects almost all aspects of modern life and all students deserve access to a computing education that prepares them to pursue the wide array of intellectual and career opportunities that computing has made possible. This course is not a tour of current events and technologies. Rather, it seeks to provide students with a “future proof” foundation in computing principles so that they are adequately prepared with both the knowledge and skills to live and meaningfully participate in our increasingly digital society, economy, and culture.

**Performance Objectives/Student Learning Outcomes**

This course is aligned with UNO disciplinary outcomes as reflected in UNO’s master syllabi. Specific UNO course learning objectives include:

- Students will understand the societal need for continued computing innovations.
- Students will be able to express algorithms in a well-defined and unambiguous manner.
- Students will map practical problems to a computational solution.
- Students will develop appropriate abstractions to manage problem complexity.
- Students will use programming as a creative tool.
- Students will learn teamwork implementing a small group project.
- More detailed learning objectives and evidence statements are available from the CS Principles website.

**UNO General Education Student Learning Outcomes (Natural & Physical Sciences)**

This course also fulfills a UNO General Education requirement and is aligned with the following General Education Student Learning Outcomes (SLOs). After completing the course, successful students shall be able to do the following: Successful students shall be able to do the following:

- demonstrate a broad understanding of the fundamental laws and principles of science and interrelationships among science and technology disciplines;
- demonstrate a broad understanding of various natural and/or physical phenomena that surround and influence our lives;
- describe how scientists approach and solve problems including an understanding of the basic components and limitations of the scientific method; and
- solve problems and draw conclusions based on scientific information and models, using critical thinking and qualitative and quantitative analysis of data and concepts in particular to distinguish reality from speculation.
Here is a brief summary of each of the units:

<table>
<thead>
<tr>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unit 1:</strong> The Internet</td>
<td>Learn how the multi-layered systems of the internet function as you collaboratively solve problems and puzzles about encoding and transmitting data, both ‘unplugged’ and using Code.org’s Internet Simulator.</td>
</tr>
<tr>
<td><strong>Unit 2:</strong> Digital Information</td>
<td>Learn how computers store complex information like images, video, and sound. Use interactive widgets to explore concepts like image representation and compression.</td>
</tr>
<tr>
<td><strong>Unit 3:</strong> Intro to Programming</td>
<td>Learn the JavaScript language with turtle programming in Code.org’s App Lab coding environment. Learn general principles of algorithms and program design that apply to any programming language.</td>
</tr>
<tr>
<td><strong>Unit 4:</strong> Big Data and Privacy</td>
<td>Research current events at the intersection of data, public policy, law, ethics, and societal impact. Learn the basics of how and why modern encryption works.</td>
</tr>
<tr>
<td><strong>Explore PT Prep</strong></td>
<td>Practice and then complete the Explore Performance Task (PT).</td>
</tr>
<tr>
<td><strong>Unit 5:</strong> Building Apps</td>
<td>Continue learning how to program in the JavaScript language. Use Code.org’s App Lab environment to create a series of applications that live on the web. Each app highlights a core concept of programming.</td>
</tr>
<tr>
<td><strong>Create PT Prep</strong></td>
<td>Practice and then complete the Create Performance Task (PT).</td>
</tr>
<tr>
<td><strong>Post AP:</strong> Data Tools</td>
<td>Learn how computers allow data to be collected, cleaned, analyzed, and visualized in order to find patterns and draw conclusions.</td>
</tr>
</tbody>
</table>

**Addressing Diversity, Equity, and Broadening Participation in the Curriculum**

A central goal of the curriculum is for it to be accessible to all students, especially those in groups typically underrepresented in computing. To this end, we have worked to provide examples and activities that are relevant and topical enough for students to connect back to their own interests and lives. Wherever possible, and especially in the videos that accompany the curriculum, we seek to highlight a diverse array of role models in terms of gender, race, and profession from which students can draw inspiration and “see themselves” participating in computing.

The curriculum assumes no prior knowledge of computing and is written to support both students and teachers who are new to the discipline. Activities are designed and structured in such a way that students with diverse learning needs have space to find their voice and to express their thoughts and opinions. The activities, videos, and computing tools in the curriculum strive to have a broad appeal and to be accessible to a student body diverse in background, gender, race, prior knowledge of computing, and personal interests.

Broadening student participation in computer science is a national goal, and effectively an issue of social justice. Motivational marketing messages only get you so far. We believe that the real key to attracting students to computer science and then sustaining that growth has as much to do with the teacher in the classroom as it does with anything else. The real “access” students need to computing is an opportunity to legitimately and meaningfully participate in every lesson regardless of the student’s background or prior experience in computing coming into the course. For example, the course begins with material that is challenging but typically unfamiliar even to students who have some prior experience or knowledge of computer science.

**Who Should Take This Course?**

The course requires a significant amount of expository writing (as well as writing computer code, of course).
For students wishing to complete this course, we recommend they be in 10th grade or above due the expectations of student responsibility and maturity.

The curriculum itself does not assume any prior knowledge of computing concepts before entering the course. It is intended to be suitable as a first course in computing though students with a variety of backgrounds and prior experiences will also find the course engaging and with plenty of challenges. While it is increasingly likely that students entering this course will have had some prior experience in computer science (particularly with programming), that experience is equally likely to be highly varied both in quantity and quality. It is for this reason that the course does not start with programming, but instead with material that is much more likely to put all students on a level playing field for the first few weeks of class. Read more about this below in the description of Unit 1.

**CS Principles Course At-A-Glance**
The chart on the following page is intended to show the big picture of the entire course. While the layout of units may appear to be modular, the units of study are scaffolded and sequenced to build students’ skills and knowledge toward the Enduring Understandings of the CSP Course Framework. The lessons for each unit assume that students have the knowledge and skills obtained in the previous units. There are also many thematic connections that can be made between and among lessons and units. A summary of how the course is arranged is below.

- The course is split into several units.
- Larger units are divided into two chapters - groups of lessons that address a related set of topics.
- Each lesson is intended to be a complete thought that takes the student from some motivational question or premise to an activity that builds skills and knowledge toward some learning objective(s). Lessons are typically designed to be taught in one ~50 minute class session though exceptions are noted throughout.
- Units will contain at least one and often multiple major projects and assessments. Projects are typically designed as Practice PTs and mimic some elements of the Create or Explore Performance Tasks.
- Pacing guidance is provided for each lesson, visually indicating roughly how many weeks each chapter or unit should take. Lessons are roughly allocated to weeks.
- Preparation for the AP® Create and Explore Performance Tasks is integrated throughout the course. The Create PT Prep and Explore PT Prep units are included to help you prepare students to understand the requirements of the tasks and set aside the required time to complete them.
Unit 1 - The Internet
Ch. 1: Representing and Transmitting Info

wk 1
Personal Innovations
Sending Binary Messages
Sending Messages with the Internet Simulator

Number Systems
Binary Numbers

2
Sending Numbers
Sending Text
Unit 1 Chapter 1 Assessment

Ch. 2: Inventing The Internet

3
The Need for Addressing
Routers and Redundancy

4
Packets and Making a Reliable Internet
HTTP and Abstraction

5
Practice PT - The Internet and Society
Unit 1 Chapter 2 Assessment

Unit 2 - Digital Information

wk
Bytes and File Sizes
1
Text Compression
Encoding B&W Images

Encoding Color Images

2
Lossy Compression and File Formats
Rapid Research - Format Showdown
Unit 2 Chapter 1 Assessment

Unit 3 - Intro to Programming

wk
The Need for Programming Languages
1
The Need for Algorithms
Creativity in Algorithms

Using Simple Commands

2
Creating Functions
Functions and Top-Down Design

APIs and Function Parameters

3
Creating Functions with Parameters
Looping and Random Numbers

4
Practice PT - Design a Digital Scene
Unit 3 Chapter 1 Assessment

Unit 4 - Big Data and Privacy

wk
What is Big Data?
1
Finding Trends with Visualizations
Check Your Assumptions

2
Rapid Research - Data Innovations
Identifying People with Data

The Cost of Free

3
Simple Encryption
Encryption with Keys and Passwords
Public Key Crypto

4
Rapid Research - Cybercrime
Unit 4 Chapter 1 Assessment

Explore PT Prep

wk
Explore PT - Review the Task
1
Explore PT - Make a Plan

2
Explore PT - Complete the Task (8 total class hours)

Unit 5 - Building Apps
Ch. 1: Event-Driven Programming

wk
Buttons and Events
1
Multi-screen Apps
Building an App - Multi-Screen App

Controlling Memory with Variables
Building an App - Clicker Game

2
Unit 5 Assessment 1
User Input and Strings

If-Statements Unplugged
Boolean Expressions and If-Statements

3
"if-else-if" and Conditional Logic
Building an App - Color Sleuth

4
Unit 5 Assessment 2
Building an App - Color Sleuth

Ch. 2: Programming with Data Structures

While Loops
1
Loops and Simulations
Introduction to Arrays

Building an App - Image Scroller
Unit 5 Assessment 3

6
Processing Arrays
Functions with Return Values
Building an App - Canvas Painter

7
Unit 5 Assessment 4
Unit 5 Assessment 5 - AP Pseudocode Practice

Create PT Prep

wk
Create PT - Review the Task
1
Create PT - Make a Plan

2
Create PT - Complete the Task (12 total class hours)

3
Create PT - Complete the Task (continued)

Post AP - Data Tools

wk
Intro to Data
1
Good and Bad Data Visualizations
Making Data Visualizations

Discover a Data Story
2
Cleaning Data
Creating Summary Tables

3
Practice PT - Tell a Data Story
Post AP Assessment
Technical and Material Requirements
The course requires and assumes a 1:1 computer lab or setup such that each student in the class has access to an Internet-connected computer every day in class. Each computer must have a modern web browser installed. All of the course tools and resources (lesson plans, teacher dashboard, videos, student tools, programming environment, etc.) are online and accessible through a modern web browser. For more details on the technical requirements, please visit: code.org/educate/it

While the course features many “unplugged” activities designed to be completed away from the computer, daily access to a computer is essential for every student. It is not required that students have access to internet-connected computers at home to teach this course. But because almost all of the materials are online, it is certainly an advantage. PDFs of handouts, worksheets and readings are available on the course website.

Computational Tools, Resources and Materials Provided
The Code.org CSP curriculum includes almost all resources teachers need to teach the course including:

Lesson Plans
- Instructional guides for every lesson
- Activity Guides and handouts for students
- Formative and summative assessments
- Exemplars, rubrics, and teacher dashboard

Videos
- Student videos - including tutorials, instructional and inspirational videos
- Teacher videos - including lesson supports and pedagogical tips and tricks

Computational Tools
- Widgets for exploring individual computing concepts
- Internet Simulator - Code.org’s tool for investigating the various “layers” of the internet
- App Lab - Code.org’s JavaScript programming environment for making apps

Required Materials / Supplies:
One potentially significant cost to consider is printing. Many lessons have handouts that are designed to guide students through activities. While it is not required that all of these handouts be printed, many were designed with print in mind and we highly recommend their use.

Beyond printing, some lessons call for typical classroom supplies and manipulatives such as:
- Student Journal
- Poster paper
- Markers
- Post-it notes
- Graph paper, etc.

There is a complete materials list in the curriculum front matter. Besides printing costs, all other materials are highly suggested, and are low cost (cups, string, playing cards, etc.).

Suggested Text
Blown to Bits http://www.bitsbook.com/
This course does not require or follow a textbook. Blown to Bits is a book that can be accessed online free of cost. Many of its chapters are excellent supplemental reading for our course, especially for material in Units 1, 2 and 4. We refer to chapters as supplemental reading in lesson plans as appropriate.
AP® Assessment
The AP Assessment consists of a 74-question multiple choice exam and two “through-course” assessments called the AP Performance Tasks (PTs). The tasks can be found in the official AP CS Principles Exam and Course Description.
- Explore Performance Task (p. 108)
- Create Performance Task (p. 113)

Academic Integrity: “The maintenance of academic honesty and integrity is a vital concern of the University community. Any student found responsible for violating the policy on Academic Integrity shall be subject to both academic and disciplinary sanctions.” Via studentlife.unomaha.edu/integrity

Coverage of the CS Principles Framework and Computational Thinking Practices
The CS Principles Framework outlines seven “Big Ideas” of computing, and six “Computational Thinking Practices”. Activities in the course should ensure that students are engaging in the Computational Thinking Practices while investigating the Big Ideas.

Seven Big Ideas
The course is organized around seven big ideas, which encompass ideas foundational to studying computer science.
- Big Idea 1: Creativity
- Big Idea 2: Abstraction
- Big Idea 3: Data
- Big Idea 4: Algorithms
- Big Idea 5: Programming
- Big Idea 6: The Internet
- Big Idea 7: Global Impact

Six Computational Thinking Practices
Computational thinking practices capture important aspects of the work that computer scientists engage in.
- P1: Connecting Computing
- P2: Creating Computational Artifacts
- P3: Abstracting
- P4: Analyzing Problems and Artifacts
- P5: Communicating
- P6: Collaborating

These Big Ideas and Practices are not intended to be taught in any particular order, nor are they units of study in and of themselves. The Big Ideas overlap, intersect, and reference each other. The practices represent higher order thinking skills, behaviors, and habits of mind that need to be constantly visited, repeatedly honed, and refined over time.

Unit Overviews
On the pages that follow are more in-depth prose descriptions of each unit of study which explain the topics covered, what students will be doing and how the lessons build toward the Enduring Understandings in the framework.

Each unit also highlights a particular lesson, project or assignment of interest, explaining what students do and showing which learning objectives and computational thinking practices that particular assignment addresses.
Unit 1: The Internet

This unit explores the technical challenges and questions that arise from the need to represent digital information in computers and transfer it between people and computational devices. Topics include: the digital representation of information - especially, numbers, text, and communication protocols. The first unit of this course purposefully addresses material that is fundamental to computing but with which many students, even those with computers at home or who have some prior experience with programming, are unfamiliar. This levels the playing field for participation and engagement right from the beginning of the course.

Chapter 1 of the unit begins with a consideration of what is involved in sending a single bit of information from one place to another. In the Sending Binary Messages lesson students work with a partner to invent and build their own bit-sending “device.” Complexity increases as students adapt their machines to handle multi-bit messages and increasingly complex information. Students use an Internet Simulator that allows them to develop and test binary encodings and communication protocols of their own invention. These should be an illustrative set of activities that helps build toward the Enduring Understandings that: A variety of abstractions built upon binary sequences can be used to represent all digital data (2.1) and that characteristics of the Internet influence the systems built on it (6.2).

Chapter 2 of the unit is called “Inventing the Internet” because through the unit students continue to solve problems similar ones that had to be solved to build the real Internet. Students design their own versions of protocols, each one layered on the previous one, in a process that mimics the layered sets of protocols on the real Internet and builds toward the Enduring Understandings that The Internet is a network of autonomous systems (6.1) and that How the Internet was designed (layers of protocols) affects the systems built on top of it (6.2) as well as its ability to grow and adapt.

For the practice Performance Task at the end of the unit students research a modern societal issue related to the Internet such as “Net Neutrality” or internet censorship, which layers in Enduring Understandings about the fact that computing has a global affect -- both beneficial and harmful -- on people and society (7.3).

Unit 1: Framework Mappings (see figure)
Unit 2: Digital Information

This unit further explores the ways that complex digital information is encoded and represented. In this unit students will use a variety of tools including several Code.org widgets.

The unit begins by exploring the sheer size of files used by computers today. Using the Text Compression Widget students will then learn how compression helps decrease the amount of space necessary to store a piece of information. Next they’ll use a Pixelation Widget to explore how both black and white and color images are stored in computers. Along the way they’ll learn the importance of metadata, or data about data, when storing information in a computer and how hexadecimal notation allows humans to quickly represent binary sequences. Throughout this sequence students are considering both the possibilities and limitations of encoding information in binary and building on the Enduring Understanding that a variety of abstractions built upon binary sequences can be used to represent all digital data (2.1).

In the last two lessons of the unit students hone their research skills by investigating how familiar real world file formats use these concepts. In the Rapid Research activity that ends the unit students have a “showdown”, arguing the relative benefits and drawbacks of their format compared to others. This continues to build the Enduring Understanding that there are trade offs when representing information as digital data (3.3).

Unit 2: Practice PT Highlights

This unit further explores the ways that complex digital information is encoded and represented. In this unit students will use a variety of tools including several Code.org widgets.

The unit begins by exploring the sheer size of files used by computers today. Using the Text Compression Widget students will then learn how compression helps decrease the amount of space necessary to store a piece of information. Next they’ll use a Pixelation Widget to explore how both black and white and color images are stored in computers. Along the way they’ll learn the importance of metadata, or data about data, when storing information in a computer and how hexadecimal notation allows humans to quickly represent binary sequences. Throughout this sequence students are considering both the possibilities and limitations of encoding information in binary and building on the Enduring Understanding that a variety of abstractions built upon binary sequences can be used to represent all digital data (2.1).

In the last two lessons of the unit students hone their research skills by investigating how familiar real world file formats use these concepts. In the Rapid Research activity that ends the unit students have a “showdown”, arguing the relative benefits and drawbacks of their format compared to others. This continues to build the Enduring Understanding that there are trade offs when representing information as digital data (3.3).
Unit 3: Intro to Programming

This unit introduces students to programming in the JavaScript language and creating small applications (apps) that live on the web. This introduction places a heavy emphasis on understanding general principles of computer programming and revealing those things that are universally applicable to any programming language.

To start the unit we use unplugged activities to introduce algorithms and highlight the need for a programming language to implement them on a computer. These activities will involve the whole class working in small groups to solve problems using simple manipulatives like playing cards or blocks. We want to draw connections here between the rules of Internet protocols developed earlier in the course, in which students acted as the computer processing the information. Many of the structured and systematic thinking that goes into developing communication protocols feels similar to designing algorithms - ultimately you’re designing a series of steps to solve a problem that a machine could follow. We want to establish the dual Enduring Understandings that algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages (4.1) and people write programs to execute algorithms (5.2).

Students are introduced to the App Lab programming environment by writing programs to control a “turtle”, an imaginary character that moves around the screen and can draw. In the lessons students learn features of the JavaScript language by going through a series of short tutorials to familiarize students with the environment, and new concepts. There is a heavy emphasis on writing procedures (functions in JavaScript), and using top-down program design - a process by which a large problem is broken down into smaller and more manageable parts. These lessons highlight the way multiple levels of abstraction are used to write programs (2.2).

Along the way students create more and more sophisticated drawings culminating in the Practice PT: Design a Digital Scene in which small groups must collaborate to design and share code to create a small vignette created with turtle art. Through the lessons and PTs we want to build toward some Enduring Understandings that creative development can be an essential process for creating computational artifacts (1.1) and that collaboration and computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem (1.2).

Unit 3: Algorithms and Programming

| 1. The Need for Programming Languages | 4.1 4.1.1[P2] (A) |
| 2. The Need for Algorithms | 5.1 5.1.1[P2] (BC) |
| 3. Creativity in Algorithms | 5.3 5.1.1[P3] (ACDFGL) |
| 4. Using Simple Commands | 5.4 5.1.1[P4] (CDEFHJK) |
| 5. Creating Functions | 9. Looping and Random Numbers |
| 6. Functions and Top-Down Design | 10. Practice PT - Design a Digital Scene |
| 7. APIs and Using Functions with Parameters | 2.2 2.2.1[P2] (A) |
| 8. Creating Functions with Parameters | 2.2 2.2.1[P2] (C) |

Unit 3: Framework Mappings (see figure)
Unit 4: Big Data and Privacy

The data rich world we live in also introduces many complex questions related to public policy, law, ethics and societal impact. In many ways this unit acts as a unit on current events. It is highly likely that there will be something related to big data, privacy and security going on in the news at any point in time. The major goals of the unit are 1) for students to develop a well-rounded and balanced view about data in the world around them and both the positive and negative effects of it and 2) to understand the basics of how and why modern encryption works.

During the first two weeks of the unit students will research and discuss innovations enabled by computing in a wide variety of fields (7.2). During this time views about the benefits - “Big Data is great!” - and drawbacks - “Big Data is scary!” - will swing quickly. We primarily want to build toward the dual Enduring Understandings that Computing facilitates exploration and the discovery of connections in information (3.2) and that Computing innovations influence and are influenced by the economic, social, and cultural contexts in which they are designed and used (7.4) while the beneficial and harmful effects (7.3) of these things must be weighed and kept in balance.

The activities later in the unit about data encryption follow a pattern: introduce an encryption concept through an unplugged activity or thinking prompt, and then “plug it in” by using a Code.org widget to explore the concept further. The purpose of the widgets is to allow students time to play with some of the ideas - often mathematical in nature - underlying different methods of encryption and why they might be susceptible to being “cracked.” These explorations lead towards an understanding of computationally hard problems and the fact that algorithms can solve many but not all computational problems (4.2).

In particular students should come away with a high level understanding of how asymmetric encryption works and why it makes certain things possible (sending encrypted data without a shared key) and certain things basically impossible (cracking a key). By investigating some of the mathematical foundations of encryption we build toward the Enduring Understanding that cybersecurity is an important concern for the Internet and the systems built on it (6.3) and as always There are trade offs when representing information as digital data (3.3).

Unit 4 Framework Mappings (see figure)
Explore PT Prep
In Units 1-4 students learn the content and practice the skills they need in order to succeed on the AP CSP Explore Performance Task. This short unit prepares students for the logistics of completing and submitting the Explore Performance Task.

The first two lessons of this unit include whole-class activities designed to prepare students for the task itself. Students will explore sample submissions and complete activities designed to help them understand the task requirements and develop a plan for the time given to complete the task. The third lesson guides students through the process of completing the task and gives guidance to the instructor on how to effectively manage the classroom during the 8 hours of class time a teacher is required to give students to complete the task. Guidance is also provided to teachers on how much assistance they are able to provide students during this largely independent project.

The 8 hours are the minimum amount of class time required, and do not need to be contiguous.

Unit 5: Building Apps
This unit continues to develop students’ ability to program in the JavaScript language, using Code.org’s App Lab environment to create a series of small applications (apps) that live on the web, each highlighting a core concept of programming. In this unit students transition to creating event-driven apps. The unit assumes that students have learned the concepts and skills from Unit 3, namely: writing and using functions, using simple repeat loops, being able to read documentation, collaborating, and using the Code Studio environment with App Lab.

The first chapter begins by introducing App Lab’s “Design Mode” which allows students to rapidly prototype an app. Again, we want to highlight the Enduring Understanding that Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem (1.2). As students construct simple apps that respond to user actions and inputs, the lessons progress through some core concepts of programming; Specifically, the lessons cover variables, boolean logic and conditionals, which enforces the understanding that Programming uses mathematical and logical concepts (5.5).

The second chapter goes deeper into core programming concepts including looping, arrays, and the use of models and simulation to develop new insight and knowledge (2.3). We want to reinforce the idea that programs are developed, maintained, and used by people for different purposes (5.4). Each app also emphasizes a different core concept and skill of programming allowing us to further the connections that people write programs to execute algorithms (5.2) and that programs employ appropriate abstractions (5.3) (such as list structures) as well as mathematical and logical concepts (5.5) to extend traditional forms of human expression and experience (1.3).

The Practice PT: Create Your Own App asks students to look back at the apps they’ve created during the unit and use one as a point of inspiration for creating their own app. This project is designed to highlight the way programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems (5.1) and is designed to closely mirror the actual Create PT which students will complete in the following unit.

Unit 5: Framework Mappings (see figure)
Create PT Prep

In Units 1-5 students learn the content and practice the skills they need in order to succeed on the AP CSP Performance Tasks. This short unit prepares students for the logistics of completing and submitting the Create Performance Task.
The first two lessons of this unit include whole-class activities designed to prepare students for the task itself. Students will explore sample task submissions and complete activities designed to help them understand the task requirements and then develop a plan for the time given to complete the task. The third lesson guides students through the process of completing the task and gives guidance to the instructor on how to effectively manage the classroom during the 12 hours of class time a teacher is required to give students to complete the task. Guidance is also provided to teachers on how much assistance they are able to provide students during this largely independent project.

The 12 hours are the minimum amount of class time required, and do not need to be contiguous.

**Post AP: Data Tools**

In this unit students develop skills interpreting visual data and using spreadsheet and visualization tools to create their own digital artifacts. Through an ongoing project - the “class data tracker” - students learn how to collect and clean data, and to use a few common tools for computing aggregations and creating visualizations. These activities build toward the Enduring Understandings that people use computer programs to process information to gain insight and knowledge (3.1) and that computing facilitates exploration and the discovery of connections in information (3.2).

As students explore ways in which computing enhances communication, interaction, and cognition (7.1), they also examine how human error during data analysis can lead to inaccurate and potentially damaging conclusions. This leads to a deeper understanding that there are trade offs, and potentially beneficial and harmful effects when representing information as digital data (3.3, 7.3).

This unit does not necessarily need to be taught after the AP test but due to the length of the course it is recommended that this be the last unit of instruction.