|  |
| --- |
| **Computer Science Principles**  **Syllabus for AP CS Principles Course Audit** |

**Course Description**

Students work in teams to develop computational thinking and solve problems. Structured activities progress to open-ended projects and problems that require planning, documentation, communication, and other professional skills. Problems aim for ground-level entry with no ceiling: all students can successfully engage the problems while students showing greater achievement are challenged to work further. There are five primary course objectives.

* To develop problem solving and computational thinking skills
* To generate excitement about the field of computing
* To introduce computational tools that foster creativity
* To build awareness of career opportunities in all fields for people with computational skills
* To consider issues raised by the present and future societal impact of computing

**CS Principles: Computational Thinking Practices and Big Ideas**

Students develop the six Computational Thinking Practices and achieve enduring understandings of the seven Big Ideas by collaborating to create solutions to problems. They evaluate and communicate solutions and connect computing to all areas of work and society. In considering any problem, students build the habit of considering five ways in which computing can be applied to a problem.

* Express ideas with creativity

Use tools to create and remix artifacts including sound, images, video, and text, and visualizations. Understand how such data are represented digitally.

* Safely and effectively use the Internet

Use the Internet for collaboration, crowd-sourcing, research, publication, social networking, and economic participation. Understand principles of cyber hygiene.

* Collect, visualize, analyze, and communicate data

Formulate problems so that computation can help solve them, logically organize data, and generalize patterns in data.

* Model and simulate

Identify abstractions in a model and understand, predict, and communicate phenomena with simulation.

* Create and improve algorithms and automate

Describe and improve an algorithmic procedure. Create software for networked and embedded computing and physical automation.

**Course Style**

Each lesson is framed by a project or problem. Students begin each lesson by considering the problem and the lesson's essential questions, and they reflect throughout the lesson on the essential questions. The knowledge and skills necessary for students to successfully engage the problem are taught during activities guided by step-by-step instructions, instructional videos, teacher presentations, and unplugged activities. Students complete the vast majority of the work in the course with a partner, typically changing partners each lesson.

**Texts and Learning Resources**

Abelson et al. (2008). *Blown to Bits*. Addison Wesley.

Passages are used in specific assignments.

PLTW. (2016). *Computer Science Principles.*

Activities, projects, and problems and supporting resources are provided to the student in offline and online formats. The offline format includes Word documents, PowerPoint presentations, MP4 instructional videos, and zip files of source code and data. These resources are also available through a web-based interface with the Instructure Canvas learning management system, with hyperlinks to additional external resources.

**Lab Computing Resources and Coding Opportunities**

Lab resources are sufficient to provide each pair of students with two computers meeting PLTW computer specifications and one Android device with WiFi connectivity for live testing in MIT App Inventor. SSH access is provided to a Linux environment for learning about the Internet, the Web, and cybersecurity.

The syllabus calls for students to be working with code on a computer during 87 days plus additional time for the *Create* performance task. The course software installation list indicates software provided to the student, including Canopy Python, NetLogo, XaoS, MEGA, and tools for SSH, FTP, and hex inspection.

**Computer Languages Used**

During the unit on algorithms, students are introduced to programming with Scratch™ and MIT App Inventor and then use *Python®. Python* is the primary language used in Unit 1 and in the course overall. During the unit on the Internet, students use *Python* and also work in a Linux environment manipulating HTML, CSS, JavaScript, PHP, and SQL. During the unit on data, students use *Python* and Microsoft® Excel®. During the unit on simulation, students use *Python* and NetLogo.

**Learning Objectives**

The course follows the Understanding by Design model in which each lesson is designed to produce evidence that students have achieved specific course objectives. Course objectives include all CS Principles learning objectives. Course materials explicitly show alignment to objectives at the lesson level and at the activity level, including PLTW understanding, skills, and knowledge objectives; CS Principles learning objectives and essential knowledge; and objectives specified in NGSS, CCSS, CSTA, and other standards. There are many opportunities to meet the learning objectives within each of the big ideas; several are described below. In all of these examples, students work in a team to combine team members' perspectives, skills, and knowledge to address a complex problem. This team approach gives students opportunities to describe, explain, and justify the design and appropriateness of their computational choices, to analyze and describe computational artifacts and the behavior of programs, to communicate the meaning of knowledge discovered in data, and to use simulations to understand, predict and communicate about natural phenomena.

**Example activities, projects, and problems**

Each activity, project, or problem provides opportunities to meet learning objectives within several big ideas, often focused primarily within one big idea. Students engage in all six computational thinking practices: Connecting Computing [P1], Creating Computational Artifacts [P2], Abstracting [P3], Analyzing Problems and Artifacts [P4], Communicating [P5], and Collaborating [P6].

**Big Idea 1: Creativity**

* In Problem 1.2.6 Designing an App, student teams create an app for a mobile device using MIT App Inventor. Teams pair program and apply concepts of Agile development. Students present their creations and reflect in writing on their development process and on their collaboration. (LO1.1.1 [P2], LO1.2.1 [P2], LO1.2.2 [P2], LO1.2.3 [P2], LO1.2.4 [P6], LO1.2.5 [P4], LO1.3.1 [P2], LO2.2.1 [P2], LO2.2.3 [P3], LO4.1.1 [P2], LO4.1.2 [P5], LO5.1.1 [P2], LO5.1.2 [P2], LO5.1.3 [P6], LO5.2.1 [P3], LO5.3.1 [P3], LO5.4.1 [P4], LO5.5.1 [P1])

**Big Idea 2: Abstraction**

* In the unplugged Activity 1.4.1 Procedural Abstraction, students act out the instantiation of ping pong balls and implement methods for drawing colored shapes on the surface of the ping pong balls. Students generalize to consider how classes and methods abstract away details. (LO2.2.1 [P2], LO2.2.2 [P3], LO2.2.3 [P3], LO5.1.1 [P2], LO5.2.1 [P3], LO5.3.1 [P3])
* In Activity 4.1.3 Introducing Simulation, students explore hypotheses about a wolf-sheep-grass ecosystem abstracted by a simulation in NetLogo. Continuing this work in Activity 4.1.4 Varying Simulation Parameters, students analyze the wolf sheep predation model using Monte Carlo methods and use parallel processors to automate exploration of the behavior of the model across a range of parameter values. In another model using moths, students analyze data to determine the characteristics of moth flight that reproduce real motion captured in a video. (LO1.2.5 [P4], LO2.2.1 [P2], LO2.3.1 [P3], LO2.3.2 [P3], LO3.1.1 [P4], LO3.1.3 [P5], LO3.2.1 [P1], LO5.3.1 [P3])

**Big Idea 3: Data and Information**

* In Project 3.2.6 Genomic Data, each student team picks a human protein, retrieves a published DNA sequence from the National Center for Biotechnical Information, and executes a BLAST search for similar DNA sequences in the Center's database. Students use MEGA, a professional biologist's tool, to create a phylogenetic tree visualizing the similarities in the genetic data across organisms. (LO1.2.2 [P2], LO1.2.3 [P2], LO1.2.4 [P6], LO3.1.1 [P4], LO3.1.2 [P6], LO3.1.3 [P5], LO3.2.2 [P3], LO4.1.1 [P2], LO7.1.1 [P4], LO7.1.2 [P4], LO7.2.1 [P1])

**Big Idea 4: Algorithms**

* In Activity 2.3.2 Security by Encryption, student teams analyze different algorithms that decide whether a number is a prime. Students use Python's timeit library to empirically compare the algorithms and analyze how the time efficiency of the algorithms depends on the length of the data input. (LO3.3.1 [P4], LO4.1.1 [P2], LO4.1.2 [P5], LO4.2.1 [P1], LO4.2.2 [P1], LO4.2.3 [P1], LO4.2.4 [P4], LO5.2.1 [P3], LO5.3.1 [P3], LO5.4.1 [P4], LO5.5.1 [P1], LO6.2.2 [P4], LO6.3.1 [P1])

**Big Idea 5: Programming**

* In Activity 1.4.5 Image Algorithms, student teams design and implement an algorithm to create derivative images from all images in a folder. Students use the numpy library for scientific computing to code pixel-level analysis and manipulation. Students also use the Python Imaging Library to code image manipulations at a higher level of abstraction. Students explain abstractions they use and explain how their program functions. (LO1.1.1 [P2], LO1.2.2 [P2], LO1.2.3 [P2], LO1.2.4 [P6], LO1.3.1 [P2], LO2.2.2 [P3], LO2.2.3 [P3], LO4.1.1 [P2], LO4.1.2 [P5], LO5.1.1 [P2], LO5.1.2 [P2], LO5.1.3 [P6], LO5.2.1 [P3], LO5.3.1 [P3], LO5.5.1 [P1])

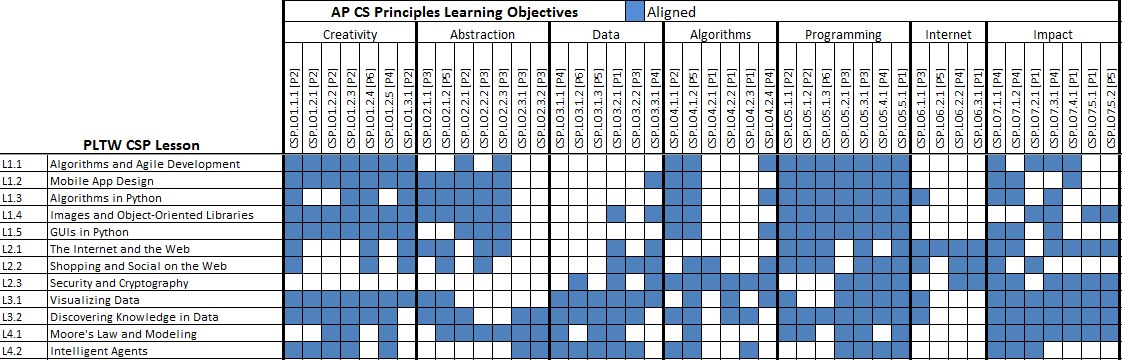
**Big Idea 6: Internet**

* In Activity 2.1.3 Protocols and Bandwidth, students act out the TCP/IP protocol and recursive DNS lookup. Student use a Linux environment provided by PLTW to use various tools for understanding DNS, routing, latency, and bandwidth. Tools used include ping, the Domain Information Groper, ifconfig, nslookup, and tracepath. (LO2.1.1 [P3], LO2.1.2 [P5], LO2.2.2 [P3], LO2.2.3 [P3], LO3.3.1 [P4], LO6.1.1 [P3], LO6.2.1 [P5], LO6.2.2 [P4], LO6.3.1 [P1], LO7.4.1 [P1])

**Big Idea 7: Global Impact**

* In Activity 2.1.1 Rise of the Internet, student teams compare Internet connectivity between groups of countries over time. Students then explore the global impact of the Internet, selecting one topic from choices including social interaction, retail business, scientific research, and governance. Each team identifies reliable sources of information and prepares a two- to five- minute presentation on their topic. (LO6.1.1 [P3], LO6.2.1 [P5], LO6.2.2 [P4], LO7.1.1 [P4], LO7.3.1 [P4], LO7.4.1 [P1], LO7.5.1 [P1], LO7.5.2 [P5])
* In Activity 2.3.3 Security and Liberty, students read, discuss, and write about data generated by and about citizens and consider impact on privacy, liberty, and law enforcement. Students explore what data sets are available from the government and describe the content of one of them with consideration of the impact the shared data has on society. Students read one or more ACM Tech News articles and write about beneficial and harmful effects of a computing innovation. (LO3.1.2 [P6], LO3.2.1 [P1], LO3.3.1 [P4], LO7.1.1 [P4], LO7.1.2 [P4], LO7.2.1 [P1], LO7.3.1 [P4], LO7.4.1 [P1], LO7.5.1 [P1], LO7.5.2 [P5])

Lesson-Level Alignment with CS Principles Learning Objectives



Activity-level alignment is provided at the end of the document.

**Performance Assessment Tasks**

Students create artifacts and associated writing for the CS Principles *Create* and *Explore* performance assessment tasks. Several opportunities exist to practice components of each performance task.

|  |  |
| --- | --- |
| Opportunities to *Create* | Opportunities to *Explore* |
| Problem 1.1.7 Scratch Game or Story  Problem 1.2.6 Designing an App  Problem 1.4.7 Image Artist  Problem 1.5.4 Design a Python GUI  Project 2.1.4 HTML and CSS  Project 2.2.4 Dynamic Data-Driven Design  Problem 3.2.7 Investigating with Data  Project 4.2.3 Modifying a Simulation's Assumptions | Activity 1.4.6 Digital Property and Forensics  Activity 1.5.1 Human Computer Interaction  Activity 2.3.3 Security and Liberty  Activity 3.1.3 Data Innovations and Parallel Algorithms  Activity 4.1.1 Computing Impacts All Fields |

Five to ten days (45-60 minutes each) are allotted for each of the projects and problems that offer practice opportunities for the *Create* task. One or two days are allotted for each of the activities that offer *Explore* task practice opportunities. Students are provided with 8 hours of class time to complete the *Explore* performance task and 12 hours of class time to complete the *Create* performance task using the College Board Performance Task descriptions and rubrics. Categorized as Problem 4.2.5, these 20 hours are distributed throughout the course, with roughly 5, 10, and 5 hours devoted in Units 2, 3, and 4, respectively.

**Additional Assessment**

Students are encouraged but not required to take the AP CS Principles exam. Students are required to complete the 80-minute PLTW end of course exam. A handful of check-for-understanding quizzes are used to assess progress during course activities.

Student work on the projects and problems forms the primary evidence of student achievement. PLTW rubrics are provided for student deliverables including presentations, writing, collaborations, project documentation, and software. The College Board Performance Assessment Rubrics are also used during the practice opportunities, focusing on different elements of the rubrics at each of the opportunities.

**Syllabus at a Glance**

**Unit 1: Algorithms, Graphics, and Graphical User Interfaces (68 days, 9 additional days as time permits)**

Comprising the first half of the course, Unit 1 emphasizes programming, creativity, algorithmic thinking, and abstraction. Students work with Scratch, App Inventor, and *Python* programming languages to tell graphical stories, publish games and Android applications, and explore various development environments and programming techniques. Students create original code and read and modify code provided from other sources. An Agile software development process is emphasized, and personal, professional, and collaborative skills take center stage. Students debate policy questions about the ownership and control of digital data and examine the implications for creative industries and consumers. In this unit students begin their exploration of career paths tied to computing.

Lesson 1.1 Algorithms and Agile Development (15 days)

Lesson 1.2 Mobile App Design (15 days)

Lesson 1.3 Algorithms in *Python* (20 days)

Lesson 1.4 Images and Object-Oriented Libraries (17 days)

Lesson 1.5 GUIs in *Python* (10 days, optional)

**Unit 2: The Internet (30 days)**

The goal of Unit 2 is for students to have a more concrete understanding of the Internet as a set of computers exchanging bits and the implications of these exchanges. Students use PHP and SQL to structure and access a database hosted on a remote server, learn how HTML and CSS direct the client computer to render a page, and experiment with JavaScript™ programming language to provide dynamic content. The focus of the unit is on the protocols that allow the Internet to function securely as it delivers social media and eCommerce content. Students work briefly in the various web languages to understand how the languages work together to deliver this content. The history and workings of the Internet are explored, and issues of security, privacy, and democracy are considered. Practical cybersecurity hygiene is included. Career paths in cybersecurity, web development, and information technology are highlighted.

Lesson 2.1 The Internet and the Web (9 days)

Lesson 2.2 Shopping and Social on the Web (13 days)

Lesson 2.3 Security and Cryptography (8 days)

**Unit 3: Raining Reigning Data (30 days)**

The goal of Unit 3 is for students to see the availability of large-scale data collection and analysis in every area they can imagine. Students examine very large data sets tied to themselves as well as to areas of work and society. They learn a variety of data visualization techniques and work to recognize opportunities to apply algorithmic thinking and automation when considering questions that have answers embedded in data. The complexity of the data sets, visualizations, and analysis increases in the second lesson of the unit, challenging students to generalize concepts developed in the first lesson.

Lesson 3.1 Visualizing Data (14 days)

Lesson 3.2 Discovering Knowledge from Data(16 days)

**Unit 4: Intelligent Behavior (24 days)**

In Unit 4 the emergence of intelligent behavior is explored from two distinct approaches: from human crowd sourcing of data and from separate algorithmic agents working in parallel. The goal is to galvanize the connections among computing concepts and between computing and society. The first lesson explores the hardware layer of computing, working from discrete components to integrated circuits. The exponential advancement of electronics, low on the ladder of abstraction, is connected to advancements at the highest levels on the ladder of abstraction, where artificial intelligence and simulation and modeling are impacting all fields. In the concluding lesson, students identify problems and questions that can be addressed with computer simulation, incorporating agent-based modeling. Students are challenged to explore the assumptions and parameters built into several simulations and to attach meaning to the results. Having explored a few applications of intelligent behavior emerging from algorithmic components, students reflect on the current and future state of artificial intelligence.

Lesson 4.1 Moore’s Law and Modeling (12 days)

Lesson 4.2 Intelligent Agents (12 days)

**Through-Course and End of Course Assessment (21-22 hours)**

Performance Task *Create* (12 hours)

Performance Task *Explore* (8 hours)

PLTW Assessment PLTW End-of-Course Exam (Two 40-minute sessions)

AP Assessment AP End-of-Course Exam (2 hours, optional)

**Detailed Outline**

**Lesson 1.1 Algorithms and Agile Development (15 days)**

The goal of this lesson is to introduce students to programming at a level appropriate to novice programmers. With an introduction to pair programming and the Agile software development process, students create original programs in Scratch that incorporate audio and visual elements while tackling algorithmic problems. The lesson opens with an introduction to how computing is affecting our lives. Students explore cloud-based tools for collaboration and select tools to manage and document the products they create. The foundations for later algorithmic thinking are built by focusing on the most common roles that variables fulfill, with an introduction to the conventions of object-oriented programming.

Activity 1.1.1 Principles (2 days)

Activity 1.1.2 Lightbot: Input, Output, State (1 day)

Activity 1.1.3 Branching and Iteration (2 days)

Activity 1.1.4 Objects and Methods (1 day)

Activity 1.1.5 Variable Roles I (1 day)

Activity 1.1.6 Variable Roles II (2 days)

Problem 1.1.7 Scratch Game or Story (6 days)

**Lesson 1.2 Mobile App Design (15 days)**

Students learn about binary representation of numbers, letters, and colors in an unplugged activity. Students then build, evaluate, and modify apps in MIT App Inventor. Algorithms involving conditionals, iteration, accumulation, aggregation in lists, and one-way flags are compared side-by-side between Scratch and App Inventor. Students conclude the lesson by developing and presenting their own solution to a need they identify, practicing an Agile development process.

Activity 1.2.1 Bits and Bytes (1 day)

Activity 1.2.2 Introducing App Inventor (2 days)

Activity 1.2.3 Creating Mobile Apps (2 days)

Activity 1.2.4 Analyzing a Program (2 days)

Project 1.2.5 Modifying a Program (2 days)

Problem 1.2.6 Designing an App (6 days)

**Lesson 1.3 Algorithms in *Python* (20 days)**

An unplugged activity introduces the ladder of abstraction and processor-level instructions. Students are introduced to variable types and algorithmic structures in *Python*, comparing programs across Scratch, App Inventor, and Python. After building strength with basic *Python* algorithms, students create algorithms to compete in a round-robin tournament of the Prisoner’s Dilemma, using the collaborative programming platform GitHub in the process.

Activity 1.3.1 Programs are Data (1 day)

Activity 1.3.2 *Python* Variables and Functions (2 days)

Activity 1.3.3 Branching and Output (2 days)

Activity 1.3.4 Nested Branching and Input (2 days)

Activity 1.3.5 Strings (2 days)

Activity 1.3.6 Tuples and Lists (3 days)

Activity 1.3.7 For Loops (3 days)

Activity 1.3.8 While Loops (2 days)

Activity 1.3.9 Tools for Collaboration (1 day)

Project 1.3.10 Game Theory (2 days)

**Lesson 1.4 Images and Object-Oriented Libraries (17 days)**

The goal of this lesson is for students to become independent learners of a programming language who are able to interpret documentation for object-oriented libraries commonly available. The lesson begins with an unplugged activity to teach object-oriented concepts. Students build additional strength with *Python* algorithms, manipulating image files by modifying pixel data and using code libraries to work at higher levels of abstraction. As part of that work, they learn to use a variety of documentation including application programming interfaces (APIs). Students read, discuss, and debate intellectual property issues associated with digital data. In the culminating problem of the lesson, they collaborate to create an image processing function that highlights the power of automation.

Activity 1.4.1 Procedural Abstraction (1 day)

Activity 1.4.2 Objects and Methods (2 days)

Activity 1.4.3 Images and Arrays (2 days)

Activity 1.4.4 *Python* Imaging Library API (2 days)

Project 1.4.5 Image Algorithms (3 days)

Activity 1.4.6 Digital Property and Forensics (2 days)

Problem 1.4.7 Image Artist (5 days)

**Lesson 1.5 GUIs in *Python* (10 days, optional)**

Activity 1.5.1 is an unplugged activity in which students generalize the user interface topic of this lesson to the field of human-computer interaction. The remainder of the lesson is optional, implemented if time permits. In the lesson, students work with interface objects as an abstraction. Students create a graphical user interface (GUI) with considerations of audience and accessibility. Students practice using an API to learn methods that affect an object’s state. Students work with two APIs: the Tkinter Canvas for drawing and animation and the Tkinter toolbox of GUI widgets. Students are provided code for a simple GUI that implements a model-view-controller (MVC) pattern. Students will modify the elements of that pattern to suit their own needs. The lesson concludes with a problem in which students create a model-view-controller GUI using Scratch or *Python*.

Activity 1.5.1 Human-Computer Interaction (1 day)

Activity 1.5.2 The API for the Tkinter Canvas (2 days)

Activity 1.5.3 The MVC Pattern with Tkinter (2 days)

Problem 1.5.4 Design a *Python* GUI (5 days)

[Lesson 2.1 The Internet and the Web](file:///C:\Users\bbrown\Documents\BB%20Documents\CSE\Content\Notes%20Units%201-4%20from%20DetailedOutlines.docx) **(9 days)**

In this lesson the goal is to build student understanding of the Internet as a set of computers exchanging bits in the form of packets. Students will learn to identify the components of their digital footprint and compare the designs, strengths, and weaknesses of their favorite web pages. In this context students use an unplugged activity to understand (in broad brushstrokes) the content and flow of data when browsing the web. They compare results from different search engines and learn to refine their search techniques. They review how to assess the trustworthiness of web-based media and consider the data flow that permits targeted advertisements. Students employ appropriate tools to explore the hierarchical nature of DNS and IP. Students identify ways that a web developer’s decisions affect the user and ways that the user’s decisions impact society. The tree structure of web documents is introduced alongside HTML and CSS. Students exchange keys and messages, use *Python* functions to encrypt and decrypt, and explain how paired key encryption and certification authorities provide security and authentication.

Activity 2.1.1 The Rise of the Internet (2 days)

Activity 2.1.2 Your Favorite Web Page (1 day)

Activity 2.1.3 Protocols and Bandwidth (2 days)

Project 2.1.4 HTML and CSS (3 days)

Activity 2.1.5 Secure Protocols (1 day)

[Lesson 2.2](file:///C:\Users\bbrown\Documents\BB%20Documents\CSE\Content\Notes%20Units%201-4%20from%20DetailedOutlines.docx) **Shopping and Social** on the Web **(13 days)**

The goal for this lesson is for students to understand the role of client-side code, server-side code, and databases in delivering interactive web content. The hook is a problem in which CS students collaborate with art students to publish content on the web. Students are provided with JavaScript and PHP code and can access an SQL database from a shell command line as well as through PHP. Students compare languages encountered so far to generalize the concepts of sequencing instructions, selection of instructions by conditionals, iteration, and the common roles of variables. Students explore and compare career paths within computing. Students also begin selecting topics for the CS Principles Performance Tasks as described in the final Problem 4.2.5 of the course.

Activity 2.2.1 HTML5 and JavaScript (3 days)

Activity 2.2.2 Introducing PHP (3 days)

Activity 2.2.3 Databases and SQL (2 days)

Problem 2.2.4 Dynamic Data-Driven Design (4 days)

Activity 2.2.5 Career Fields of CS and IT (1 day)

[Lesson 2.3 Security](file:///C:\Users\bbrown\Documents\BB%20Documents\CSE\Content\Notes%20Units%201-4%20from%20DetailedOutlines.docx) **and Cryptography** **(8 days)**

The goal of this lesson is for students to personally invest in maintaining online security and to improve their personal cybersecurity hygiene. Students focus on cybersecurity from the perspectives of the user, the software developer, the business, the nation, and the citizen. In the team competition at the end of the lesson, students explore parallel strands in encryption and security. Encryption is used as a route to explore the efficiency of algorithms and how the time for an algorithm to execute can be dependent on its input.

Activity 2.3.1 The Vulnerable User (2 days)

Activity 2.3.2 Security by Encryption (1 day)

Activity 2.3.3 Security and Liberty (2 days)

Project 2.3.4 The Heist (3 days)

[Lesson 3.1](file:///C:\Users\bbrown\Documents\BB%20Documents\CSE\Content\Notes%20Units%201-4%20from%20DetailedOutlines.docx) **Visualizing Data** **(14 days)**

The goal of this lesson is for students to be able to create visualizations to analyze large sets of data and to meaningfully interpret the patterns they uncover. They draw conclusions relevant to themselves from data, including local weather, the economics of their community, and naming trends with their name. At the beginning of the lesson, students weigh societal concerns around the collection and persistence of Big Data. The students learn how to use *Python* to make useful graphic representations of data, developing from familiar visualizations to more modern visual analyses like scaled-dot or colorized scatter plots of multidimensional data sets. Students are introduced to basic Excel® spreadsheet programming and cell manipulation. A Monte Carlo simulation is used to help students appreciate the meaning of evidence for association between two variables.

Activity 3.1.1 Time Series and Trends (3 days)

Activity 3.1.2 Privacy Issues with Data (2 days)

Activity 3.1.3 Data Innovations and Parallel Algorithms (2 days)

Activity 3.1.4 Pie Charts and Bar Graphs (3 days)

Activity 3.1.5 Histograms and Distributions (4 days)

[Lesson 3.2](file:///C:\Users\bbrown\Documents\BB%20Documents\CSE\Content\Notes%20Units%201-4%20from%20DetailedOutlines.docx) Discovering Knowledge from Data **(16 days)**

As in the previous lesson, the goal of this lesson is for students to be able to create a range of visualizations to analyze large complex sets of data and to meaningfully interpret the patterns they uncover. Students use statistics to deepen the meaning of knowledge gained by visualization. The hooks are again conclusions they can draw from data relevant to themselves, including various geographic perspectives on their life and facial recognition of their own features. The lesson uses Excel as well as *Python* to manipulate and visualize data. Students examine multidimensional data sets using scatter plot arrays and view geographic and social data using heat maps and directed graphs. Students experiment with object recognition and face recognition. They are challenged to discover clustering and linear correlation patterns lurking in data sets distributed across sources. Genomic data are investigated with tools of the professional biologist. Finally, student teams choose a question to investigate and using large sets of data.

Activity 3.2.1 Inferential Statistics (2 days)

Activity 3.2.2 Image Data (1 day)

Activity 3.2.3 Linked Data (1 day)

Activity 3.2.4 Geographic Data (2 days)

Activity 3.2.5 Simulation Data (1 day)

Project 3.2.6 Genomic Data (3 days)

Problem 3.2.7 Investigating with Data (6 days)

[Lesson 4.1](file:///C:\Users\bbrown\Documents\BB%20Documents\CSE\Content\Notes%20Units%201-4%20from%20DetailedOutlines.docx)  **Moore’s Law and Modeling** **(12 days)**

**I**n this lesson students construct an understanding of how the explosion of technology over the last two decades has impacted every realm of study and employment. Students begin by researching the impact of computer modeling and simulation which have been made possible by the rapid increase in computational power due to the continued applicability of Moore's Law. They then manipulate discrete electronic components to create logic gates and then produce comparable results using integrated circuits. Students comprehend what it means to double the number of transistors that can fit in a given area. Students explore simulation in NetLogo directly by manipulating a model of predation and a model of the spread of viruses in humans. The lesson concludes with an examination of the code of ethics for simulationists and reflection on the necessity of adhering to such a code.

Activity 4.1.1 Computing Impacts All Fields (2 days)

Activity 4.1.2 Basic Control Circuits (3 days)

Activity 4.1.3 Introducing Simulations (3 days)

Activity 4.1.4 Varying Simulation Parameters (2 days)

Activity 4.1.5 Assumptions, Abstractions, and Ethics (2 days)

[Lesson 4.](file:///C:\Users\bbrown\Documents\BB%20Documents\CSE\Content\Notes%20Units%201-4%20from%20DetailedOutlines.docx)2 **Intelligent Agents (12 days) and Performance Tasks (20 hours)**

In this lesson students experiment with materials designed to illuminate the rise of intelligent and complex behavior from simple rules and seemingly unintelligent agents. Students begin by studying a model of Langton's ant, a simple Turing machine with some surprising emergent behavior. The students manipulate models of neurons and neural networks. Students design and conduct their own experiments on a model of their own choosing using Monte Carlo methods. Students explore the generation and observation of fractals and study a diffusion-limited aggregation model for producing fractal behavior. In the final project of the course, students complete the CS Principles *Create* and *Explore* performance tasks. The class time allotted for the performance tasks is categorized as Problem 4.2.5, but students begin exploring ideas for their work midway through the course, devoting 5, 10, and 5 hours in Units 2, 3, and 4, respectively. At the end of the course, students complete an 80-minute PLTW end of course exam.

Activity 4.2.1 Emergent Behavior (2 days)

Activity 4.2.2 Neural Networks (3 days)

Project 4.2.3 Modifying a Simulation’s Assumptions (5 days)

Activity 4.2.4 Beauty in Chaos and Fractals (2 days)

Problem 4.2.5 Computer Science Principles (20 hours)

Android is a trademark of Google Inc.

App Inventor is used without the permission of MIT under Creative Commons Attribution 3.0 license.

Excel is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

JavaScript is a trademark or registered trademark of Oracle in the U.S. and other countries.

NetLogo is courtesy Wilensky, U. (1999). Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Tkinter Canvas and Tinkter toolbox are free software released under a *Python* license.

All other marks are properties of their respective owners.

**Activity-Level Alignment with CS Principles Learning Objectives**

